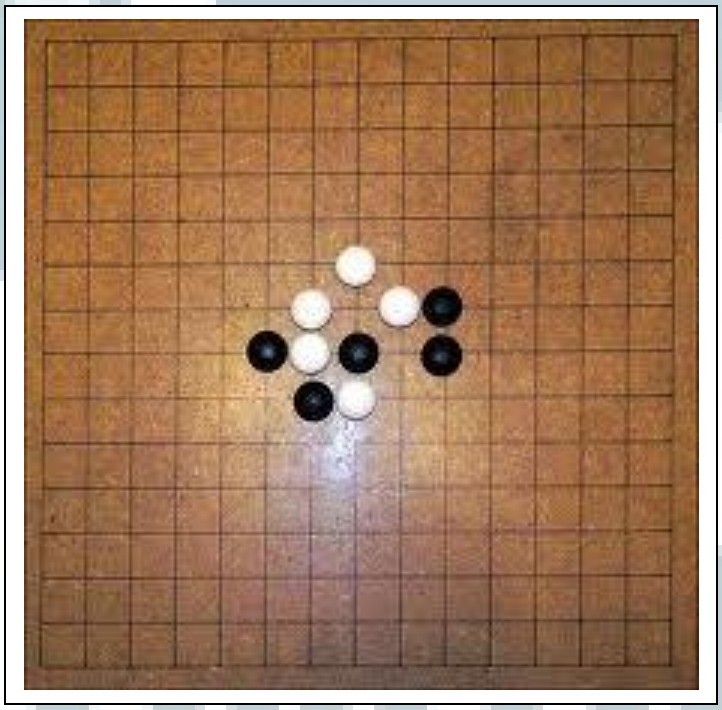
BAB II

TELAAH LITERATUR

2.1 Gomoku

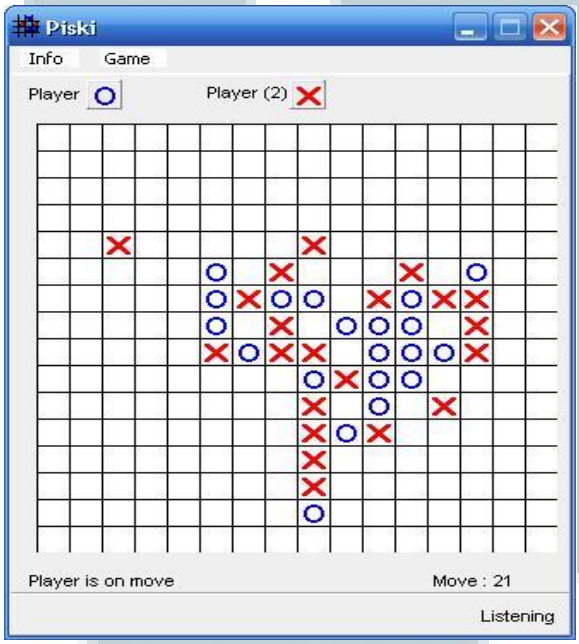
Gomoku atau dikenal sebagai *five in row* adalah sebuah permainan papan tradisional berusia 4000 tahun lebih yang berasal dari cina dan dikenal sebagai Wu Zi Qi. Go berarti lima dan moku berarti bidak. Permainan ini diciptakan untuk 2 pemain yang bergiliran meletakkan batu hitam dan putih di papan hingga tercipta sebuah baris horizontal, vertikal, diagonal dengan warna sama. Umumnya gomoku dimainkan diatas papan kayu berukuran 15 x 15 kotak.



Gambar 2.1 Permainan papan gomoku
(Sumber : www.quantumgambitz.com)

Permainan selesai apabila terdapat pemain pertama yang mencapai lima buah batu berturut-turut membentuk garis horizontal, vertikal atau diagonal. Selain itu apabila seluruh kotak sudah terisi batu, permainan dianggap seimbang.

Seiring dengan perkembangan teknologi, gomoku dirancang sebagai sebuah permainan yang dapat dimainkan pada sebuah komputer. Pada komputer, gomoku umumnya berukuran 15 x 15 kotak dan menggunakan bidak X dan O. Gomoku ini juga dilengkapi dengan kecerdasan buatan sehingga *player* dapat bermain sendiri melawan *computer*. (Rinaldi, 2007)



Gambar 2.2 Software permainan gomoku
(Sumber : gomoku.en.softonic.com)

2.2 Algoritma

Sebelum angka desimal dikenal luas, orang-orang menggunakan angka romawi untuk menuliskan angka. Sistem desimal ditemukan sekitar tahun 600 di India di mana penulisan angka dan aritmatika dapat dilakukan secara cepat dan efisien. Al Khwarizmi menemukan metode dasar untuk melakukan penjumlahan, pengurangan, perkalian, pembagian, dan akar. Prosedur yang dilakukan presisi, tidak ambigu, efisien dan tepat, dan merupakan akar dari algoritma (Dasgupta, dkk., 2006: 9).

Semenjak itu, sistem desimal dan algoritma numerik berperan penting dalam perkembangan jaman. Kedua sistem tersebut digunakan dalam sains dan teknologi. Dan setelah komputer ditemukan, konsep tersebut digunakan untuk *positional system* pada *bits*, *words*, dan *arithmetic unit* (Dasgupta, dkk., 2006: 10).

Secara umum, algoritma didefinisikan sebagai urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Suatu urutan langkah logis untuk menyelesaikan masalah dapat dikatakan sebagai sebuah algoritma apabila memenuhi syarat-syarat berikut ini.

1. Algoritma harus berhenti setelah mengerjakan sebuah proses,
2. setiap langkah harus didefinisikan secara tepat dan tidak berarti dua (ambiguitas),
3. algoritma memiliki nol atau lebih masukan,
4. algoritma memiliki satu atau lebih keluaran, dan
5. algoritma harus efektif,yaitu setiap langkah dapat dikerjakan dalam waktu yang masuk akal (Knuth, 1997: 4).

Terdapat tiga jenis proses dalam algoritma yaitu *sequence*, *selection*, dan *repetition*. *Sequence* merupakan urutan dari langkah-langkah yang dilakukan. *Selection* adalah proses pemilihan langkah selanjutnya berdasarkan kondisi yang ada. Sedangkan *repetition* adalah proses yang dilakukan berulang-ulang hingga mencapai kondisi tertentu.

Algoritma dapat dituliskan dalam dua cara, yaitu menggunakan *pseudocode* dan *flowchart*. *Pseudocode* menggunakan instruksi berupa kata, dan *flowchart* menggunakan instruksi berupa gambar atau simbol.

2.3 Algoritma Minimax

Algoritma Minimax merupakan algoritma dasar pencarian DFS (*Depth-First Search*) untuk melakukan traversal dalam pohon. DFS akan mengekspansi simpul paling dalam terlebih dahulu. Setelah simpul akar dibangkitkan, algoritma ini akan membangkitkan simpul pada tingkat kedua, yang akan dilanjutkan pada tingkat ketiga, dst (Nadhira , 2008).

Algoritma minimax ini bekerja secara rekursif dengan mencari langkah yang akan membuat lawan mengalami kerugian maksimum. Semua strategi lawan akan dihitung dengan algoritma yang sama dan seterusnya. Ini berarti,pada langkah pertama komputer akan menganalisis seluruh pohon permainan. Dan untuk setiap langkahnya, komputer akan memilih langkah yang paling membuat lawan mendapatkan keuntungan minimum, dan yang paling membuat komputer itu sendiri mendapatkan keuntungan maksimum (Rinaldi, 2007).

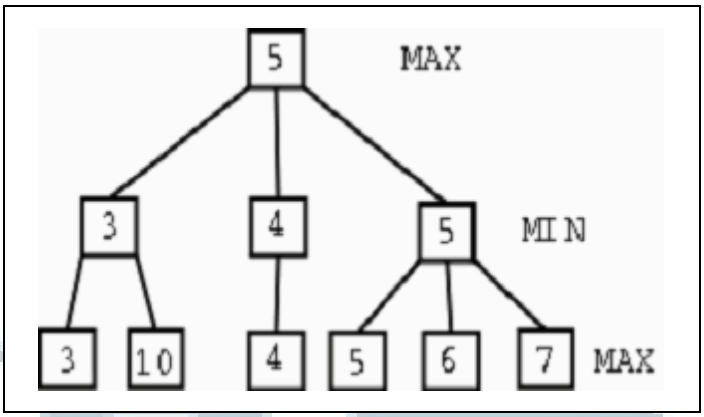
Dalam penentuan keputusan tersebut dibutuhkan suatu nilai yang merepresentasikan kerugian atau keuntungan yang akan diperoleh jika langkah tersebut dipilih. Untuk itulah disini digunakan sebuah fungsi *heurisitic* untuk mengevaluasi nilai sebagai nilai yang merepresentasikan hasil permainan yang akan terjadi jika langkah tersebut dipilih. Biasanya pada permainan *gomoku* ini digunakan nilai n pangkat 2 untuk mewakilkan banyak bidak yang membentuk garis lurus horizontal, vertikal, dan diagonal. Dua bidak yang membentuk garis lurus akan memberikan nilai 2^2 , sedangkan tiga bidak membentuk garis lurus akan memberika nilai 2^3. Dari nilai-nilai *heuristic* inilah komputer akan menentukan simpul mana dari pohon permainan yang akan dipilih, tentunya simpul yang akan dipilih tersebut adalah simpul dengan nilai *heuristic* yang akan menuntun permainan ke hasil akhir yang menguntungkan bagi komputer (Nadhira, 2008).

Berikut ini adalah psedeucode yang umum digunakan dalam gomoku (Rinaldi, 2007),

```
Cari langkah yang dengan nilai maksimum
    IF langkah tersebut merupakan langkah kemenangan THEN pilih lagkah tersebut.
    ELSE
    FOR EACH kemungkinan langkah yang ada
        Cari langkah lawan yang bernilai minimum.
    RETURN nilai dari langkah tersebut.
Pilih langkah yang bernilai maksimum dari langkah-langkah tersebut.
    IF ada langkah kemenangan
        THEN pilih langkah tersebut.
    ELSE IF lawan mempunyai spot terisi dalam satu garis lebih banyak dari spot kita
        THEN tutup langkah tersebut.
    ELSE melangkah ke state yang mempunyai kemungkinan menang tertinggi
        (berdasarkan nilai heuristic yang dibangkitkan).
```

Gambar 2.3 Pseudocode kecerdasan buatan gomoku dengan algoritma minimax (Sumber : Rinaldi, 2007)

Dalam repersentasi pohon dalam algoritma Minimax, terdapat dua jenis node, yaitu node *min* dan node *max*. *Max* node akan memilih langkah dengan nilai tertinggi dan *min* node akan memilih langkah dengan nilai terendah. Pembentukan pohon pencarian solusi digunakan dengan menggunakan konsep *depth-first*, dimulai dari awal permainan sampai akhir permainan. Setelah itu, posisi akhir permainan dievaluasi melalui sudut pandang *Max* seperti gambar 2.3 :



Gambar 2.4 Representasi pohon pencarian pada algoritma *minimax.*
(Sumber : Rinaldi, 2007)

Setelah itu nilai dari setiap simpul diisi dari bawah ke atas dengan nilai yang sudah dievaluasi oleh fungsi *heuristic*. Simpul milik pemain A (MAX) menerima nilai maksimum dari simpul-simpul anaknya. Simpul milik pemain B (MIN) akan memilih nilai minimum dari simpul anak-anaknya (Rinaldi, 2007).

2.4 Algoritma Genetik

Algoritma genetika ini ditemukan oleh John Holland dan dikembangkan oleh muridnya David Goldberg. Algoritma Genetika adalah algoritma yang memanfaatkan proses seleksi alamiah yang dikenal dengan proses evolusi (Achmad, 2008).

Dalam proses evolusi, individu secara terus-menerus mengalami perubahan gen untuk menyesuaikan dengan lingkungan hidupnya. Proses seleksi alamiah ini melibatkan perubahan gen yang terjadi pada individu melalui proses perkembangbiakan. Dalam algoritma genetika ini, proses perkembang-biakan ini menjadi proses dasar yang menjadi perhatian utama, dengan dasar berpikir bagaimana mendapatkan keturunan yang lebih baik.

Algoritma genetik terdiri dari delapan komponen, yang bertugas untuk menunjang dari optimasi tersebut, adapun komponen tersebut ialah, skema pengkodean, nilai fitness,

seleksi orang tua, pindah silang, mutasi, elitisme, penggantian populasi dan kriteria penghentian (Achmad, 2008).

Untuk dapat diproses menggunakan algortima genetik, suatu permasalahan harus dikonversi dahulu kedalam bentuk individu yang diwakili oleh satu atau lebih kromosom dengan kode tertentu. Hal ini berbeda dengan teori genetika di dunia nyata. Algoritma genetik merepresentasikan gen secara umum, sebagai bilangan real, desimal atau biner yaitu:

1. *Real number enconding*. Pada skema ini, nilai gen berada dalam interval [0,R] dimana R ialah bilangan real positif dan biasanya R = 1.
2. *Discrete decimal enconding*. Pada skema ini setiap gen bisa berupa deretan bilangan bulat dalam interval [0.9].
3. *Binary enconding*. Setiap gen bisa berupa deretan nilai 0 atau 1.

2.4.1 Nilai Fitness

Nilai *fitness* dalam sebuah algoritma genetik menggambarkan tingkat kovergensi keoptimalan algoritma dimana yang diharapkan adalah nilai *fitness* yang optimal dalam hal ini angka tertinggi ialah nilai terbaik. Dalam evolusi dunia nyata, individu bernilai *fitness* tinggi akan bertahan hidup, sedangkan yang memiliki nilai *fitness* rendah akan gugur atau mati. Pada algoritma genetik, *fitness* biasanya dapat berupa fungsi objektif dari masalah yang akan dioptimalisasi. Kromosom-kromosom diseleksi menurut nilai *fitness* masing-masing Kromosom yang kuat mempunyai kemungkinan tinggi untuk bertahan hidup pada generasi berikutnya.

2.4.2 Seleksi

Seleksi merupakan proses pemilihan kromosom dari generasi lama untuk dijadikan orangtua yang akan saling kawin silang untuk membentuk kromosom baru digenerasi baru, dalam hal ini penulis menggunakan seleksi roda roulette (*roulette wheel selection*). Dalam hal ini pemilihan dua buah kromosom sebagai orang tua (*parent*) dilakukan secara proposional sesuai dengan nilai *fitness*nya. Dimana kromosom yang memiliki nilai *fitness* tertinggi akan menempati potongan yang lebih besar pada lingkaran daripada kromosom dengan nilai *fitness* yang lebih rendah. Contoh metode roulette-wheel dapat diilustrasikan pada tabel 2.1.

Tabel 2.1 Contoh fitness suatu populasi
(Sumber : Ignatius, 2010)

String	Nilai *Fitness*
S1	10
S2	5
S3	7
S4	5
S5	9
Jumlah	36

Pada contoh diatas, nilai dari S1 memiliki nilai fitnes yang paling besar, dengan demikian peluang dari S1 untuk terpilih menjadi orang tua sebesar 0.277 (nilai *fitness* / total *fitness*) sedangkan untuk nilai S2 dan S3 memiliki peluang yang sama sebesar 0.138.

Menurut George F.Luger dan William A. Stubblefield dalam buku *Artificial Intelegence Structures and Strategies For Complex Promblem Solving.* Kekuatan dari algoritma genetik ialah pada kemampuan pencarian mereka dalam pindah silang, dimana

algoritma genetik menerapkan mempertahankan beberapa solusi terbaik, dan menghilangkan solusi yang tidak bagus (Irving, 2006). Komponen pindah silang digunakan untuk membentuk keturunan baru berdasarkan orangtua yang terpilih. Komponen ini sangat dominan dalam algoritma genetik dibandingkan dengan komponen mutasi.

2.4.3 Pindah Silang

Pindah silang dilakukan dengan harapan kromosom-kromosom baru akan mempunyai bagian baik dari kromosom-kromosom lama dan tidak menutup kemungkinan menjadi kromosom-kromosom yang lebih baik

Skema dari pindah silang ini ialah, dengan mendapatkan dua buah individu orang tua, selanjutnya ditentukan titik pindah silang secara acak.Jika diasumsikan L adalah panjang kromosom, maka titik pindah silang berada diantara 1 hingga L-1, kemudian beberapa bagian dari dua kromosom ditukar pada titik pindah silang yang terpilih. Titik pindah silang ialah titik terjadinya pertukaran gen antar dua individu orang tua.

Tabel 2.2 Tabel pindah silang 1 titik
(Sumber : Ignatius,2010)

Pindah silang satu titik							
orang tua1	1	1	0	0	1	1	0
orang tua2	0	0	1	1	0	0	1
anak	1	1	0	0	0	0	1

Tabel 2.3 Tabel pindah silang banyak titik
(Sumber : Ignatius , 2010)

Pindah silang banyak titik							
orang tua1	1	1	0	0	1	1	0
orang tua2	0	0	1	1	0	0	1
anak	1	1	1	1	0	0	0

2.4.4 Mutasi

Mutasi diperlukan untuk mengembalikan informasi bit yang hilang akibat pindah silang. Mutasi diterapkan dengan probabilitas yang sangat kecil. Jika mutasi dilakukan terlalu sering, maka akan menghasilkan individu yang lemah karena konfigurasi gen pada individu yang unggul akan dirusak. Berdasarkan bagian yang termutasi, proses mutasi dapat dibedakan atas tiga bagian (Irving, 2006).

- Mutasi pada tingkat kromosom, semua gen dalam kromosom berubah.
- Mutasi pada tingkat gen, semua bit dalam satu gen akan berubah, contoh gen nomor 2 mengalami mutasi.
- Mutasi pada tingkat hanya satu bit yang berubah.

2.4.5 Kriteria Penghentian

Terdapat beberapa syarat penghentian yang digunakan pada proses perulangan (Irving, 2006).

1. Memberikan batasan jumlah iterasi, apabila batas iterasi tersebut tercapai dan nilai fitnes tertingi sebagai solusi.
2. Memberikan batasan waktu proses algoritma genetic
3. Menghitung kegagalan penggantian anggota populasi yang terjadi secara berurutan sampai jumlah tertentu.

2.5 Perbandingan Algoritma

Menurut Stuart Russel, performa dari algoritma dapat dilihat dari empat kriteria berikut ini.

1. *Completeness:* Apakah suatu algoritma dapat menemukan sebuah solusi bila solusi tesebut ada?
2. *Optimality:* Apakah strategi yang digunakan mencapai solusi optimal?
3. *Time complexity:* Berapa lama waktu yang dibutuhkan untuk menemukan solusi?
4. *Space complexity:* Berapa banyak memori yang dibutuhkan untuk melakukan pencarian? (Russel, 2010: 80).

2.6 Kecerdasan Buatan

Kecerdasan buatan atau dikenal sebagai *Artificial Intelligence* adalah bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan bisa lebih baik daripada yang dilakukan manusia.

Menurut pendapat dari berbagai sumber, sebagai berikut :

H. A. Simon [1987],"Kecerdasan buatan (*artificial intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas"

Rich and Knight [1991], "Kecerdasan Buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia."

Encyclopedia Britannica, “Kecerdasan Buatan (AI) merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi berdasarkan metode heuristic atau dengan berdasarkan sejumlah aturan”

Bagian utama aplikasi kecerdasan buatan adalah pengetahuan (*knowledge*), yaitu suatu pengertian tentang beberapa wilayah subyek yang diperoleh melalui pendidikan dan pengalaman.Pengetahuan merupakan informasi terorganisir dan teranalisa agar bisa lebih mudah dimengerti dan bisa diterapkan pada pemecahan masalah dan pengambilan keputusan. Pengetahuan terdiri dari fakta, pemikiran, teori, prosedur, dan hubungannya satu sama lain.

Komputer tidak mungkin mendapatkan pengetahuannya sendiri dengan belajar, berpengalaman atau melakukan penelitian, akan tetapi ia memperolehnya melalui upaya yang diberikan oleh seorang pakar manusia.

Hampir semua pangkalan pengetahuan (*knowledge base*) sangat terbatas, dalam arti terfokuskan kepada suatu masalah khusus. Pada saat pangkalan pengetahuan itu sudah terbentuk, teknik Kecerdasan Buatan bisa digunakan untuk memberi kemampuan baru kepada komputer agar bisa berfikir, menalar, dan membuat inferensi (mengambil keputusan berdasarkan pengalaman) dan membuat pertimbangan-pertimbangan yang didasarkan kepada fakta dan hubungan-hubungannya yang terkandung dalam pangkalan pengetahuan itu.

Dengan pangkalan pengetahuan dan kemampuan untuk menarik kesimpulan melalui pengalaman (inferensi), komputer dapat disejajarkan sebgai alat bantu yang bisa digunakan

secara praktis dalam memecahkan masalah dan pengambilan keputusan serta bisa mencapai satu atau lebih solusi alternatif pada maslah yang diberikan.

Problema yang dapat ditangani oleh kecerdasan buatan awalnya adalah pembuktian teorema dan permainan (*game*). Misalnya Newell, ahli teori logika, berusaha untuk membuktikan teorema-teorema matematika dan Samuel yang membuat program permainan catur. Kemudian para peneliti Kecerdasan Buatan terus mengembangkan berbagai teknik baru untuk menangani sejumlah besar persoalan, termasuk persepsi, pemahaman bahasa alamiah, dan problema spesifik seperti diagnosa medis.

Contoh persoalan-persoalan yang ditangani oleh Kecerdasan Buatan adalah ,

1. Permainan (*game*), seperti: *chess*, tic tac toe, dan sebagainya.
2. Pemecahan problema umum (*general problem solving*), misalnya pengambilan keputusan otomatis (*automated decision making*) dan pemodelan kinerja manusia.
3. Persepsi / perception (visi / *vision* dan percakapan / *conversation*)
4. Pengenalan pola (*pattern recognition*), seperti pengolahan citra digital untuk kebutuhan ramalan cuaca, monitor tv, dan sebagainya.
5. Pemecahan problema pakar (*expert*), mencakup bidang matematika simbolik, diagnosa medis, rekayasa rancang bangun, analisis kimia.
6. Pembuatan perangkat lunak otomatis (*automated software generation*).

Sedangkan bidang-bidang teknik kecerdasan buatan diantaranya adalah :

- Robot (*robotics*),
- Fuzzy logic
- Jaringan syaraf (*neural networks*) tiruan, dan
- Pengolahan bahasa alami (*natural language processing*)
- Pengolahan citra.
- Computer vision
- Game playing