BAB II

LANDASAN TEORI

2.1 Kecerdasan buatan

Kecerdasan buatan (*artificial intelligence*) merupakan inovasi baru di bidang ilmu pengetahuan. Mulai ada sejak muncul komputer modern, yakni pada 1940 dan 1950. Ini kemampuan mesin elektronika baru menyimpan sejumlah besar info, juga memproses dengan kecepatan sangat tinggi menandingi kemampuan manusia.

Pentingnya kecerdasan buatan menjadi nyata bagi negara-negara yang berperan sejak tahun 1970. Para pemimpin negara yang mengakui potensialnya kecerdasan buatan mengharap mendapat persetujuan jangka panjang untuk sumber-sumber yang memerlukan dana intensif. Jepang adalah yang pertama kali melakukan itu. Negara ini mengembangkan program yang sangat berambisi dalam penelitian kecerdasan buatan.

2.2 Sejarah Kecerdasan Buatan

Pada awal abad 17, *Rene Descartes* mengemukakan bahwa tubuh hewan bukanlah apa-apa melainkan hanya mesin-mesin yang rumit. *Blaise Pascal* menciptakan mesin penghitung digital mekanis pertama pada 1642. Pada 19, *Charles Babbage* dan Ada *Lovelace* bekerja pada mesin penghitung mekanis yang dapat diprogram.

Bertrand Russell dan *Alfred North Whitehead* menerbitkan Principia Mathematica, yang merombak logika formal. *Warren McCulloch* dan *Walter Pitts* menerbitkan "Kalkulus Logis Program AI pertama yang bekerja ditulis pada 1951 untuk menjalankan mesin *Ferranti Mark* di University of Manchester (UK): sebuah program permainan naskah yang ditulis oleh *Christopher Strachey* dan program permainan catur yang ditulis oleh *Dietrich Prinz. John McCarthy* membuat istilah "kecerdasan buatan " pada konferensi pertama yang disediakan untuk pokok persoalan ini, pada 1956. Dia juga menemukan bahasa pemrograman Lisp. *Alan Turing* memperkenalkan “Turing Test” sebagai sebuah cara untuk mengoperasionalkan test perilaku cerdas. *Joseph Weizenbaum* membangunELIZA, sebuah chatterbot yang menerapkan psikoterapi Rogerian

Selama tahun 1960-an dan 1970-an, Joel Moses mendemonstrasikan kekuatan pertimbangan simbolis untuk mengintegrasikan masalah di dalam program *Macsyma*, program berbasis pengetahuan yang sukses pertama kali dalam bidang matematika. *Marvin Minsky* dan *Seymour Papert* menerbitkan Perceptrons, yang mendemostrasikan batas jaringan syaraf sederhana dan *Alain Colmerauer* mengembangkan bahasa komputer Prolog. *Ted Shortliffe* mendemonstrasikan kekuatan sistem berbasis aturan untuk representasi pengetahuan dan inferensi dalam diagnosa dan terapi medis yang kadangkala disebut sebagai sistem pakar pertama. *Hans Moravec* mengembangkan kendaraan terkendali komputer pertama untuk mengatasi jalan berintang yang kusut secara mandiri.

Pada Pada tahun 1980-an, jaringan syaraf digunakan secara meluas dengan algoritma perambatan balik, pertama kali diterangkan oleh *Paul John Werbos* pada 1974. Tahun 1990-an ditandai perolehan besar dalam berbagai bidang AI dan demonstrasi berbagai macam aplikasi. Lebih khusus *Deep Blue*, sebuah komputer permainan catur, mengalahkan *Garry Kasparov* dalam sebuah pertandingan 6 game yang terkenal pada tahun 1997. DARPA menyatakan bahwa biaya yang disimpan melalui penerapan metode AI untuk unit penjadwalan dalam Perang Teluk pertama telah mengganti seluruh investasi dalam penelitian AI sejak tahun 1950 pada pemerintah AS.

Tantangan Hebat DARPA, yang dimulai pada 2004 dan berlanjut hingga hari ini, adalah sebuah pacuan untuk hadiah $2 juta dimana kendaraan dikemudikan sendiri tanpa komunikasi dengan manusia, menggunakan GPS, komputer dan susunan sensor yang canggih, melintasi beberapa ratus mil daerah gurun yang menantang.

2.3 Definisi Kecerdasan Buatan

Tidak ada definisi yang memuaskan untuk kecerdasan. Kecerdasan dapat diartikan sebagai kemampuan untuk memperoleh pengetahuan dan menggunakannya atau kecerdasan adalah apa yang di ukur oleh sebuah ”test kecerdasan”.

Apa kecerdasan buatan itu? Bagian dari ilmu pengetahuan komputer ini khusus ditujukan dalam perancangan otomatisasi tingkah laku cerdas dalam sistem kecerdasan komputer. Sistem memperlihatkan sifat-sifat khas yang dihubungkan

dengan kecerdasan dalam kelakuan atau tindak-tanduk yang sepenuhnya bisa menirukan beberapa fungsi otak manusia, seperti pengertian bahasa, pengetahuan, pemikiran, pemecahan masalah, dan lain sebagainya.

Kecerdasan Buatan (*Artificial Intelligence*) didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu *entitas* buatan. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (Komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, permainan komputer (*games*), logika fuzzy, jaringan syaraf dan robotika.

Walaupun *Artificial Intelligence*) memiliki konotasi fiksi ilmiah yang kuat, *Artificial Intelligence*) membentuk cabang yang sangat penting pada ilmu komputer, berhubungan dengan perilaku, pembelajaran dan adaptasi yang cerdas dalam sebuah mesin. Penelitian dalam *Artificial Intelligence*) menyangkut pembuatan mesin untuk mengotomatisasikan tugas-tugas yang membutuhkan perilaku cerdas. Termasuk contohnya adalah pengendalian, perencanaan dan penjadwalan, kemampuan untuk menjawab diagnosa dan pertanyaan pelanggan, serta pengenalan tulisan tangan, suara dan wajah. Hal-hal seperti itu telah menjadi disiplin ilmu tersendiri, yang memusatkan perhatian pada penyediaan solusi masalah kehidupan yang nyata. Sistem AI sekarang ini sering digunakan dalam bidang ekonomi, obat-obatan, teknik dan militer, seperti yang telah dibangun dalam beberapa aplikasi perangkat lunak komputer rumah dan video game.

2.4 Paham Pemikiran

Secara garis besar, *Artificial Intelligence* terbagi ke dalam dua faham pemikiran yaitu *Artificial Intelligence* Konvensional dan Kecerdasan Komputasional (CI, Computational Intelligence). AI konvensional kebanyakan melibatkan metoda-metoda yang sekarang diklasifiksikan sebagai pembelajaran mesin, yang ditandai dengan formalisme dan analisis statistik. Dikenal juga sebagai *Artificial Intelligence* simbolis, *Artificial Intelligence* logis, *Artificial Intelligence* murni dan *Artificial Intelligence* cara lama (GOFAI, Good Old Fashioned Artificial Intelligence). Metoda-metodanya meliputi:

1. Sistem pakar: menerapkan kapabilitas pertimbangan untuk mencapai kesimpulan. Sebuah sistem pakar dapat memproses sejumlah besar informasi yang diketahui dan menyediakan kesimpulan-kesimpulan berdasarkan pada informasi-informasi tersebut.
2. Pertimbangan berdasarkan kasus
3. Jaringan Bayesian
4. *Artificial Intelligence* berdasar tingkah laku: metoda modular pada pembentukan sistem AI secara manual

Kecerdasan komputasional melibatkan pengembangan atau pembelajaran iteratif (misalnya penalaan parameter seperti dalam sistem koneksionis. Pembelajaran ini berdasarkan pada data empiris dan diasosiasikan dengan AI non-simbolis, AI yang tak teratur dan perhitungan lunak. Metoda-metoda pokoknya meliputi:

1. Jaringan Syaraf: sistem dengan kemampuan pengenalan pola yang sangat kuat
2. Sistem Fuzzy: teknik-teknik untuk pertimbangan di bawah ketidakpastian, telah digunakan secara meluas dalam industri modern dan sistem kendali produk konsumen.
3. Komputasi Evolusioner: menerapkan konsep-konsep yang terinspirasi secara biologis seperti populasi, mutasi dan “survival of the fittest” untuk menghasilkan pemecahan masalah yang lebih baik.

Metoda-metoda ini terutama dibagi menjadi algoritma evolusioner (misalnya algoritma genetik) dan kecerdasan berkelompok (misalnya algoritma semut).

Dengan sistem cerdas hibrid, percobaan-percobaan dibuat untuk menggabungkan kedua kelompok ini. Aturan inferensi pakar dapat dibangkitkan melalui jaringan syaraf atau aturan produksi dari pembelajaran statistik seperti dalam ACT-R. Sebuah pendekatan baru yang menjanjikan disebutkan bahwa penguatan kecerdasan mencoba

untuk mencapai kecerdasan buatan dalam proses pengembangan evolusioner sebagai efek samping dari penguatan kecerdasan manusia melalui teknologi.

2.5 Macam – Macam Kecerdasan Buatan

Ada banyak jenis kecerdasan buatan, setidaknya ada lima jenis kecerdasan buatan yang sering kita temui, yaitu :

1. **Jaringan Syaraf Buatan (Artificial Neural Networks),** Merupakan sekelompok jaringan saraf *(neuron)* buatan yang menggunakan model matematis atau komputasi untuk pemrosesan informasi berdasarkan pendekatan terhubung pada komputasi. Pada kebanyakan kasus, JST merupakan sistem adaptif yang merubah strukturnya berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut.

2 **Logika Fuzzy (Fuzzy Logics)**

   Adalah peningkatan dari logika Boolean yang berhadapan dengan konsep kebenaran sebagian. Di mana logika klasik menyatakan bahwa segala hal dapat diekspresikan dalam istilah binary (0 atau 1, hitam atau putih, ya atau tidak), logika fuzzy menggantikan kebenaran boolean dengan tingkat kebenaran.Logika Fuzzy memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistik, konsep tidak pasti seperti "sedikit", "lumayan", dan "sangat". Dia berhubungan dengan set fuzzy dan teori

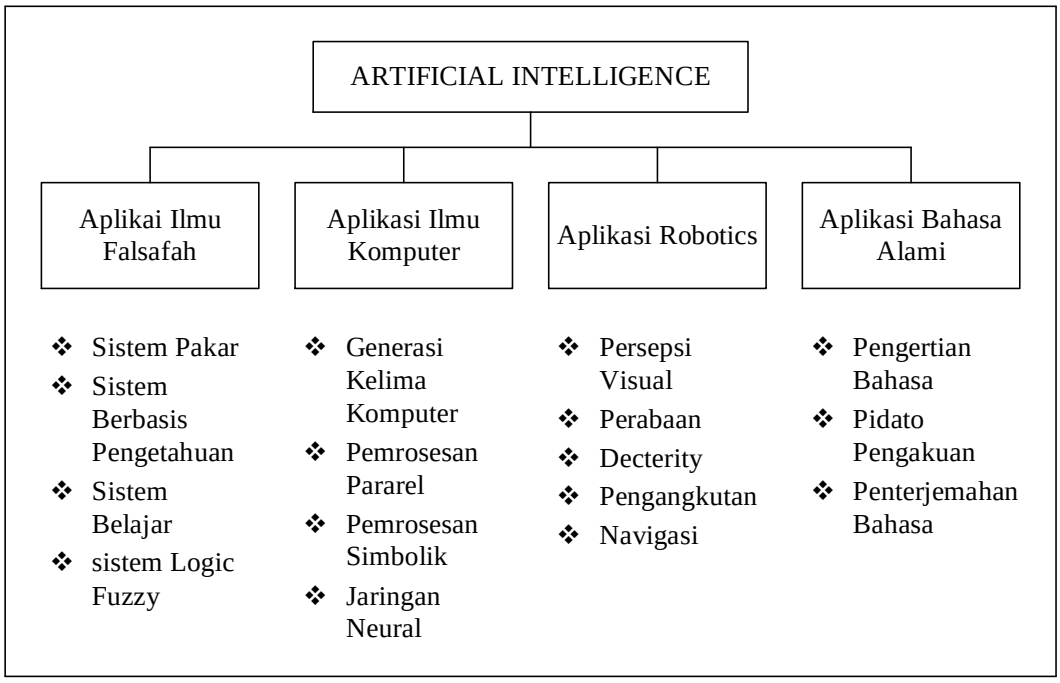
kemungkinan. Dia diperkenalkan oleh Dr. Lotfi Zadeh dari Universitas California, Berkeley pada 1965.

3 **Algoritma Genetik (Genetic Algorithms),**

Adalah teknik pencarian yang di dalam ilmu komputer untuk menemukan penyelesaian perkiraan untuk optimisasi dan masalah pencarian. Algoritma genetik adalah kelas khusus dari algoritma evolusioner dengan menggunakan teknik yang terinspirasi oleh biologi evolusioner seperti warisan, mutasi, seleksi alam dan rekombinasi (atau *crossover)***.**Algoritma Genetik biasanya digunakan dibidang kedokteran, misal untuk menganalisis DNA.

2.5.1 Bagian-bagian Utama dari Aplikasi Artificial Intelligene (AI)

Artificial Intelligence (AI) dapat dikelompokkan ke dalam empat bagian utama, seperti terlihat pada gambar di bawah ini :



Gambar 2.1 Bagian-bagian Utama dari Aplikasi *Artificial Intelligence* (AI)

Seperti terlihat pada gambar di atas, *Artificial Intelligence* (AI) dapat dikelompokkan ke dalam empat bagian utama, yaitu ilmu falsafat, ilmu komputer, aplikasi robotic, dan bahasa alami yang akan dijelaskan berikut ini.

Aplikasi Ilmu Falsafat

Untuk aplikasi ini, *Artificial Intelligence* (AI) berbasis pada penelitian di bidang biologi, neurologi, psikologi, matematika, dan berbagai disiplin ilmu terkait lainnya. Fokus penelitian dari aplikasi ini adalah meneliti bagaimana otak manusia dapat bekerja, dan bagaimana manusia dapat berfikir dan belajar. Aplikasi ilmu falsafah ini mencakup pengembangan di bidang sistem pakar, sistem berbasis pengetahuan, sistem belajar, dan sistem logic fuzzy.

Aplikasi Ilmu Komputer

Untuk aplikasi ini, *Artificial Intelligence* (AI) memfokuskan diri pada perangkat keras komputer dan sistem perangkat lunak yang dibutuhkan untuk menghasilkan superkomputer yang kuat seperti yang dibutuhkan oleh berbagai aplikasi *Artificial Intelligence* (AI). Aplikasi ilmu komputer ini mencakup pengembangan genarasi kelima komputer, pemrosesan pararel, pemrosesan simbolik, dan jaringan neural.

Aplikasi Robotic

Robotic berbasis pada bidang *Artificial Intelligence* (AI), teknik, dan psikologi. Teknologi inilah yang menghasilkan robot. Robot diartikan sebagai mesin dengan kecerdasan komputer dan dikontrol oleh komputer, dan memiliki kemampuan fisik seperti manusia. Aplikasi dari robotic ini mencakup pemberian kemampuan untuk melihat atau persepsi visual, menyentuh atau kemampuan meraba, decterity atau kemampuan untuk memegang dan memanipulasi, pengangkutan atau kemampuan fisik untuk bergerak, dan navigasi atau kecerdasan untuk menemukan atau mencapai jalan keluar.

Aplikasi Bahasa Alami

Pengembangan aplikasi ini berhubungan dengan lingkungan atau bagian utama dari *Artificial Intelligence* (AI) dan merupakan inti dari ilmu falsafat dan robotic. Dapat berkomunikasi atau berbicara kepada komputer dan robot dakam bahasa percakapan manusia dan dapat membuat komputer “mengerti” kita seperti kita saling mengerti satu sama lain merupakan tujuan dari *Artificial Intelligence* (AI).

2.6 Permainan

Permainan merupakan sebuah aktivitas rekreasi dengan tujuan bersenang-senang, mengisi waktu luang, atau berolahraga ringan. Permainan biasanya dilakukan

sendiri atau bersama-sama.Permainan komputer adalah permainan video yang dimainkan pada komputer pribadi, dan bukan pada konsol permainan, maupun mesin ding-dong. Permainan komputer telah berevolusi dari sistem grafis sederhana sampai menjadi kompleks dan mutakhir. Namun, pasar permainan komputer di Amerika Serikat mulai menurun sejak tahun 1999. Permainan teori adalah salah satu cabang diterapkan matematika yang digunakan dalam ilmu sosial (terutama ekonomi), biologi, rekayasa, sains politik, hubungan internasional, ilmu komputer (terutama untuk kecerdasan buatan), dan filosofi. Permainan teori matematis upaya untuk menangkap perilaku dalam *situasi strategis,* di mana individu yang sukses dalam membuat pilihan tergantung pada pilihan lain. Walaupun pada awalnya dikembangkan untuk menganalisa kompetisi di mana satu individu tidak lebih baik di lain penghasilan (nol jumlah permainan), telah diperluas untuk merawat berbagai kelas interaksi, yang berdasarkan beberapa kriteria. Hari Ini, "permainan teori adalah satu bentuk payung atau 'unified lapangan' teori untuk rasional samping ilmu sosial, dimana 'sosial' diterjemahkan luas, termasuk untuk manusia serta pemain non-manusia (komputer, binatang, tanaman)" (Aumann 1987).

Aplikasi permainan tradisional dari teori mencoba untuk menemukan equilibria permainan ini di-set strategi yang individu juga tidak mungkin untuk mengubah perilaku mereka. Banyak keseimbangan konsep telah dikembangkan (yang paling terkenal Nash keseimbangan) dalam upaya untuk menangkap ide ini. Keseimbangan konsep ini diharapkan akan dapat memotivasi berbeda tergantung

pada bidang aplikasi, meskipun mereka sering tumpang tindih atau bersamaan. Metodologi ini tidak tanpa kritik, dan perdebatan atas tepat untuk melanjutkan tertentu keseimbangan konsep, tepat equilibria dari semuanya, dan kegunaan dari model matematika umumnya.

2.6.1 Aplikasi dan tantangan

Permainan teori telah digunakan untuk kajian berbagai perilaku manusia dan binatang. Ia mula-mula dikembangkan dalam bidang ekonomi yang besar untuk memahami koleksi perilaku ekonomi, termasuk perilaku perusahaan, pasar, dan konsumen. Penggunaan permainan teori dalam ilmu sosial telah diperluas, dan permainan teori telah diterapkan untuk politik, sosiologis, psikologis dan perilaku juga.

Permainan teori analisis awalnya digunakan untuk studi perilaku hewan oleh Ronald Fisher pada tahun 1930an (meskipun bahkan Charles Darwin membuat beberapa permainan teori pernyataan informal). Ini bekerja predates nama "teori permainan", tetapi saham banyak fitur penting dengan bidang ini. Perkembangan ekonomi tersebut kemudian diterapkan untuk biologi sebagian besar oleh John Maynard Smith dalam bukunya *Evolution dan Teori Permainan.*

Selain digunakan untuk memprediksi dan menjelaskan perilaku, permainan teori juga telah digunakan untuk mencoba untuk mengembangkan teori dari etika normatif atau perilaku. Dalam ekonomi dan falsafah, cendekiawan telah menerapkan permainan teori untuk membantu dalam memahami perilaku baik atau tepat. Permainan teori argumen dari jenis ini dapat dilihat sebagai jauh kembali sebagai Plato.

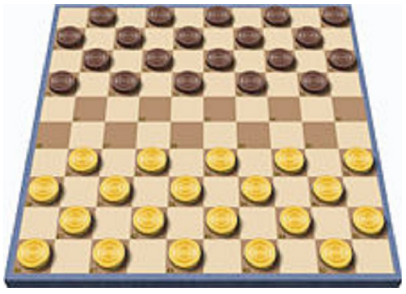
2.6.2 Ilmu komputer dan logika

Permainan teori telah datang untuk memainkan peran yang semakin penting dalam logika dan dalam ilmu komputer. Beberapa logis teori memiliki dasar dalam permainan semantik. Selain itu, para ilmuwan komputer telah digunakan untuk model permainan interaktif computations. Juga, permainan teori menyediakan teori dasar kepada bidang sistem multi-agen.

Secara terpisah, permainan teori telah memainkan peranan dalam line algoritma. Secara khusus, k-masalah server, yang di masa lalu telah disebut sebagai *permainan bergerak dengan biaya* dan *permintaan-Jawaban permainan* (Ben Daud, Borodin & Karp dkk. 1994). Yao prinsip adalah permainan teori-teknik untuk membuktikan batas lebih rendah pada komputer kompleksitas randomized algoritma, dan algoritma khususnya online

2.7 Permainan Checkers

Permainan checkers (dalam bahasa Inggris Amerika) atau disebut draughts (dalam bahasa Inggris British) merupakan permainan yang menggunakan strategi abstrak dimainkan oleh dua pemain dengan menggunakan langkah diagonal token dan menangkap dengan melompati token musuh. Permainan ini telah dimainkan di Eropa sejak abad ke- 16, dikembangkan dari permainan alquerque. Bentuk yang paling populer dari pemainan ini adalah international draughts, yang dimainkan pada papan 10x10. Bentuk yang juga populer adalah English draughts, yang disebut American checkers, dimainkan pada papan 8x8.



Gambar 2.12 International Checkers

2.7.1 Peraturan Checkers

Dimainkan oleh dua orang, dengan pemain berada pada sisi yang berlawanan dari papan. Salah satu pemain memiliki kepingan berwarna gelap dan pemain lain berwarna terang. Pemain dengan kepingan berwarna gelap melakukan langkah pertama, kecuali telah ditentukan sebelumnya.Kepingan akan bergerak diagonal dan kepingan lawan ditangkap dengan meloncatinya. Kepingan yang ditangkap akan dihilangkan dari papan.Gerak kepingan pada papan hanya dapat dilakukan pada kotak yang tidak ditempati. Permukaan yang dapat menjadi papan permainan hanya kotak dengan warna gelap. Pemain yang kalah adalah pemain yang tidak memiliki kepingan yang tersisa atau tidak dapat melakukan langkah lagi.

Kepingan tanpa mahkota disebut orang, akan bergerak satu langkah maju diagonal dan menangkap kepingan dengan melakukan dua langkah pada arah yang sama, melompati kepingan lawan pada kotak tengah. Sejumlah kepingan lawan dapat ditangkap dengan satu loncatan, tidak harus pada arah yang sama tapi bisa zigzag.

Pada English draughts kepingan hanya dapat ditangkap maju, tetapi pada international draughts kepingan dapat ditangkap mundur.

Ketika mencapai baris terjauh, kepingan berubah menjadi raja, ditandai dengan memberikan mahkota. Kepingan raja ini memiliki kekuatan tambahan untuk berjalan dan menangkap mundur (pada jenis yang tidak dapat melakukaknnya). Pada international draughts, raja dapat begerak sejauh yang ia inginkan secara diagonal

2.8 Algoritma

Dalam matematika dan komputasi, algoritma merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan catatan untuk setiap masalah, ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria, dalam hal ini berbeda dengan heuristik. Algoritma sering mempunyai langkah pengulangan (iterasi) atau memerlukan keputusan (logika Boolean dan perbandingan) sampai tugasnya selesai.Desain dan analisis algoritma adalah suatu cabang khusus dalam ilmu komputer yang mempelajari karakteristik dan performa dari suatu algoritma dalam menyelesaikan masalah, terlepas dari implementasi algoritma tersebut. Dalam cabang disiplin ini algoritma dipelajari secara abstrak, terlepas dari sistem komputer atau bahasa pemrograman yang digunakan. Algoritma yang berbeda dapat diterapkan pada suatu masalah dengan

kriteria yang sama.Kompleksitas dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Secara informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalahnya mempunyai kompleksitas yang tinggi.

2.9 Algoritma pencarian

Dalam ilmu komputer, sebuah algoritma pencarian dijelaskan secara luas adalah sebuah algoritma yang menerima masukan berupa sebuah masalah dan menghasilkan sebuah solusi untuk masalah tersebut, yang biasanya didapat dari evaluasi beberapa kemungkinan solusi. Sebagian besar algoritma yang dipelajari oleh ilmuwan komputer adalah algoritma pencarian. Himpunan semua kemungkinan solusi dari sebuah masalah disebut ruang pencarian. Algortima pencarian brute-force atau pencarian naif/uninformed menggunakan metode yang sederhana dan sangat intuitif pada ruang pencarian, sedangkan algoritma pencarian informed menggunakan heuristik untuk menerapkan pengetahuan tentang struktur dari ruang pencarian untuk berusaha mengurangi banyaknya waktu yang dipakai dalam pencarian.Sebuah algoritma pencarian uninformed adalah algoritma yang tidak mempertimbangkan sifat alami dari permasalahan. Oleh karena itu algoritma tersebut dapat diimplementasikan secara umum, sehingga dengan implementasi yang sama dapat

digunakan pada lingkup permasalahan yang luas, hal ini berkat abstraksi. Kekurangannya adalah sebagian besar ruang pencarian adalah sangat besar, dan sebuah pencarian uninformed (khususnya untuk pohon) membutuhkan banyak waktu walaupun hanya untuk contoh yang kecil. Sehingga untuk mempercepat proses, kadang-kadang hanya pencarian informed yang dapat melakukannya.

2.9.1 Pencarian List

Algoritma pencarian list mungkin adalah algoritma pencarian paling dasar. Tujuannya adalah mencari sebuah elemen dari sebuah himpunan dengan suatu kunci (kemungkinan memuat informasi yang terkait dengan kunci). Oleh karena hal ini adalah masalah yang lazim dalam ilmu komputer, kompleksitas komputasi algoritma-algoritma tersebuh telah dipelajri dengan baik. Algoritma paling sederhana adalah pencarian linear, yang secara sederhana melihat setiap elemen dari list secara berurutan. Waktu pengerjaan algoritma ini adalah O(n), dimana n adalah banyaknya elemen dalam list, dan dapat digunakan langsung pada list yang belum diproses. Algoritma pencarian list yang lebih canggih adalah pencarian biner; waktu pengerjaannya adalah O(log n). Waktu pengerjaannya jauh lebih baik daripada pencarian linear untuk list yang memiliki data banyak, tetapi sebelum dilakukan pencarian list terlebih dahulu harus terurut (lihat algoritma pengurutan) dan juga harus dapat diakses secara acak (pengaksesan acak). Pencarian interpolasi adalah

lebih baik dari pencarian biner untuk list terurut yang sangat besar dan terdistribusi merata. Algoritma Grover adalah sebuah algoritma kuantum yang menawarkan percepatan kuadrat dibandingkan pencarian linear klasik untuk list tak terurut.

Tabel hash juga digunakan untuk pencarian list, hanya memerlukan waktu yang konstan untuk mencari pada kasus rata-rata, tetapi memiliki overhead ruang yang lebih dan pada kasus terburuk waktu pengerjaannya adalah O(n). Pencarian lain yang berdasarkan struktur data khusus, menggunakan pohon pencarian biner yang self-balancing (self-balancing binary search tree) dan membutuhkan waktu pencarian O(log n); hal ini dapat dipandang sebagai pengembangan dari ide utama pencarian biner untuk memungkinkan penyisipan dan penghapusan yang cepat. Lihat array asosiatif untuk diskusi lanjut dari struktur data pencarian list.

Sebagian besar algoritma pencarian, seperti pencarian linear, pencarian biner dan pohon pencarian biner yang self-balancing, dapat dikembangkan dengan sedikit tambahan costuntuk menemukan semua nilai yang kurang dari atau lebih dari sebuah kunci, operasi ini disebut pencarian jangkauan (range search). Pengecualin ada pada tabel hash, yang tidak dapat melakukan pencarian tersebut secara efisien.

2.9.2 Pencarian Pohon

Algoritma pencarian pohon adalah jantung dari teknik-teknik pencarian. Algoritma tersebut mencari node dari pohon, terlepas apakah pohon tersebut eksplisit

atau implisit (dibangkitkan saat pengerjaan). Prinsip dasarnya adalah sebuah node diambil dari sebuah struktur data, suksesornya diperiksa dan ditambahkan pada struktur data. Dengan memanipulasi struktur data, pohon dieksplorasi dalam urutan yang berbeda-beda, dieksplore dari satu tingkat ke tingkat berikutnya (pencarian Breadth-first) atau mengunjungi node pucuk terlebih dahulu kemudian lacak balik/backtracking (pencarian Depth-first). Contoh lain dari pencarian pohon antara lain pencarian iterative-deepening, pencarian berbatas kedalaman, pencarian dwiarah dan pencarian uniform-cost.

2.10 Depth First Search

Pada Depth First Search (DFS), proses akan dilakukan pada semua anaknya sebelum dilakukan pencarian ke node-node (titik) yang selevel. Pencarian dimulai dari node akar ke level yang lebih tinggi. Proses ini diulangi terus hingga ditemukannya solusi. Stack atau tumpukan adalah struktur data yang setiap proses baik penambahan maupun penghapusan hanya bisa dilakukan dari posisi teratas tumpukan. Cara kerja stack adalah LIFO *(Last In First Out)*, dimana data yang terakhir masuk akan keluar pertama.

Berikut analisis ruang dan waktu untuk metode pencarian DFS :

1. Diasumsikan :
    a. Pohon pelacakan memiliki cabang yang selalu sama, yaiu sebanyak b.
    b. Tujuan dicapai pada level ke-d
2. Analisis Ruang

a. Setelah berjalan 1 langkah, stack akan berisi b node.

b. Setelah berjalan 2 langkah, stack akan berisi (b-1) + b node.

c. Setelah berjalan 3 langkah, stack akan berisi (b-1) + (b-1) + b node.

d. Setelah berjalan d langkah, stack akan berisi (b-1) * d + 1 node, mencapai maksimum.

3. Analisis Waktu

a.Pada kasus terbaik, DFS akan mencapai tujuan pada kedalaman d pertama, sehingga dibutuhkan pencarian sebanyak d + 1 node.

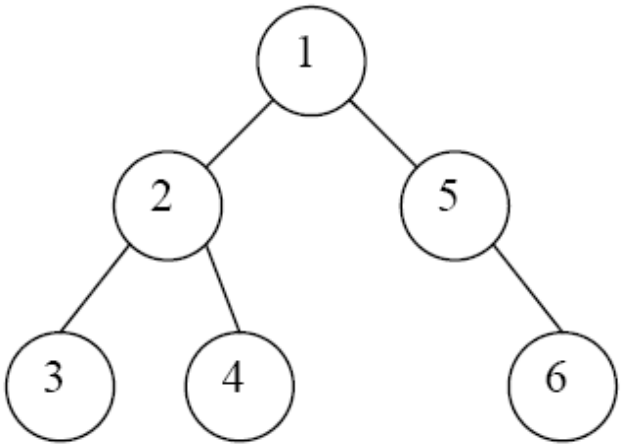
b. Pada kasus terburuk, DFS akan mencapai tujuan pada kedalaman d pada node terakhir, sehingga dibutuhkan pencarian sebanyak $1 + b + b^2 + b^3 + \ldots + b^d = (b^{d+1}-1) / (b-1)$

Keuntungan dari metode ini adalah :

1. Membutuhkan memori yang relatif kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan.
2. Secara kebetulan, metode DFS akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.

Kelemahan dari metode ini adalah :

1. Memungkinkan tidak ditemukannya tujuan yang diharapkan.
2. Hanya akan mendapatkan 1 solusi pada setiap pencarian.

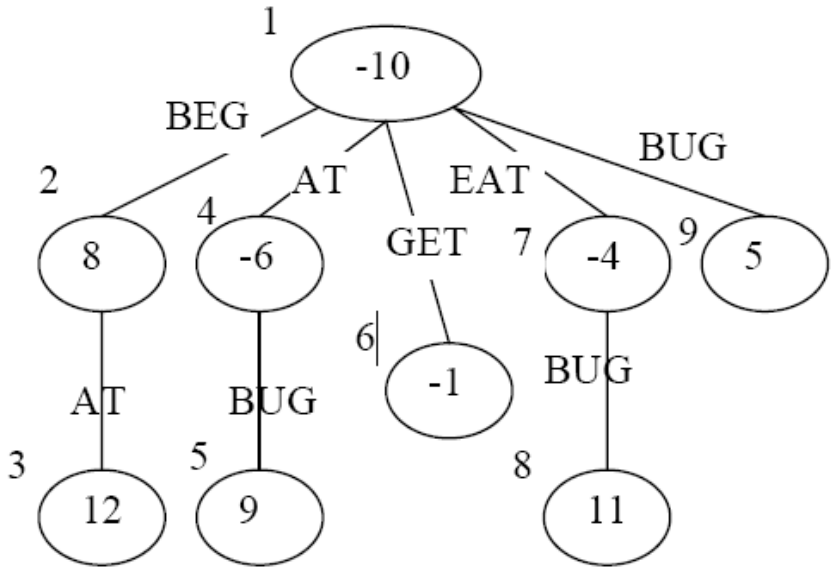


Gambar 2.2 Contoh penelusuran DFS

Pada pohon di atas, penelusuran dimulai dari simpul akar bernomor 1. Simpul berikutnya yang ditelusuri adalah simpul 2 yang bertetangga dengan simpul 1, lalu simpul 3 yang bertetangga dengan simpul 2. Karena simpul 3 sudah tidak memiliki tetangga, penelusuran akan berlanjut ke tetangga simpul 2 yaitu simpul 4. Setelah itu simpul 5 yang bertetangga dengan simpul 1, dan terakhir simpul 6 yang bertetangga dengan simpul 5.

Untuk memecahkan persoalan memaksimalkan f(H), dilakukan penelusuran terhadap semua himpunan bagian Ai yang saling lepas. Setiap simpul berisi himpunan solusi biji-biji yang akan ditaruh dan jumlah nilai dari himpunan solusi. Penelusuran dimulai dari simpul akar yang merupakan himpunan kosong dengan nilai -P. Simpulsimpul berikutnya dibangkitkan dari himpunan biji pada tabel nilai yang saling lepas dengan himpunan solusi yang telah terbentuk sejauh ini. Jika penelusuran

telah mencapai simpul daun dan tidak ada lagi simpul yang dapat dibangkitkan, nilai total dari himpunan solusi disimpan sebagai nilai maksimum sementara. Pencarian dilakukan sampai mendapatkan nilai maksimum yang paling besar. Sebagai contoh digunakan huruf-huruf pada contoh sebelumnya yaitu A, B, E, G, T, O, U. Tabel nilai yang digunakan adalah tabel 3. Nilai simpul akar = - (1+3+1+2+1+1+1) = -10. Mulai dari simpul akar, simpul pertama yang dibangkitkan berisi {B, E, G}. Lalu simpul berikutnya dibangkitkan dengan menambahkan {A, T} menjadi {B, E, G, A, T}. Setelah itu tidak ada lagi simpul yang dapat dibangkitkan, jadi nilai untuk solusi ini adalah 12



Gambar 2.3 Contoh pencarian solusi

Dari simpul 2 juga sudah tidak ada lagi simpul yang dapat dibangkitkan karena {B, E, G} beririsan dengan {G, E, T}, {E, A, T}, maupun {B, U, G}. Jadi penelusuran akan kembali ke simpul akar. Dari simpul akar dibangkitkan simpul {A, T}, lalu {A, T, B, U, G} dengan nilai total 9. Penelusuran dilakukan terus sampai semua simpul dibangkitkan. Dari gambar di atas dapat dilihat bahwa nilai maksimum yang dapat diambil adalah 12 dengan menaruh biji {B, E, G} dan {A, T}.

2.11 Breadth First Search

Pencarian dilakukan pada semua node dalam setiap level secara berurutan dari kiri ke kanan. Jika pada satu level belum ditemukan solusi, maka pencarian dilanjutkan pada level berikutnya. Demikia n seterusnya sampai ditemukan solusi. Dengan strategi ini, maka dapat dijamin bahwa solusi yang ditemukan adalah yang paling baik (Optimal).Tetapi *breadth first search* harus menyimpan semua node yang pernah dibangkitkan. Hal ini harus dilakukan untuk penelusuran balik jika solusi sudah ditemukan.

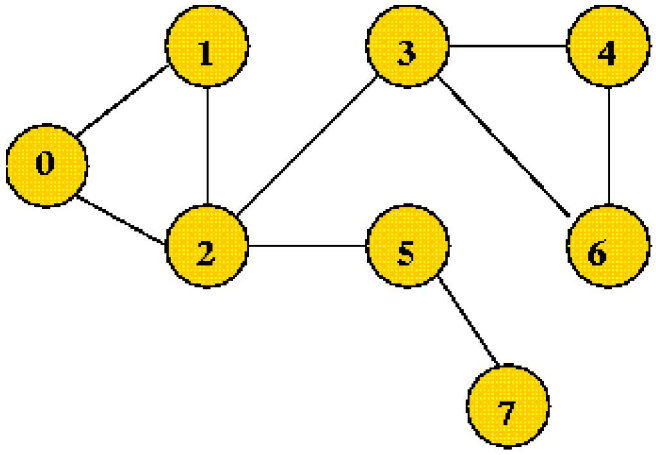
Penelusuran breadth first search :

Traversal dimulai dari simpul *v*.

a. Algoritma:

1. Kunjungi simpul *v*,
2. Kunjungi semua simpul yang bertetangga dengan simpul *v* terlebih dahulu.
3. Kunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul simpul yang tadi dikunjungi, demikian seterusnya.

Jika graf berbentuk pohor berakar, maka semua simpul pada aras *d* dikunjungi lebih dahulu sebelum simpul-simpul pada aras *d* + 1.



Gambar 2.4 Gambar Graf tak berarah

Apabila pada graf di atas digunakan algoritma

Breadth First Search, maka pengunjungan simpulsimpul akan dilakukan seperti berikut:

1. Simpul awal 0, urutan simpul yang dikunjungi adalah 0,1,2,3,5,4,6,7
2. Simpul awal 1, urutan simpul yang dikunjungi adalah 1,0,2,3,5,4,6,7
3. Simpul awal 2, urutan simpul yang dikunjungi adalah 2,0,1,3,5,4,6,7
4. Simpul awal 3, urutan simpul yang dikunjungi adalah 3,2,4,6,0,1,5,7
5. Simpul awal 4, urutan simpul yang dikunjungi adalah 4,3,6,2,0,1,5,7
6. Simpul awal 5, urutan simpul yang dikunjungi adalah 5,2,7,0,1,3,4,6
7. Simpul awal 6, urutan simpul yang dikunjungi adalah 6,3,4,2,0,1,5,7

8. Simpul awal 7, urutan simpul yang dikunjungi adalah 7,5,2,0,1,3,4,6

Jelas bahwa semua simpul akan dikunjungi, tetapi prioritas kunjungan diberikan kepada simpul yang paling dekat

2.12 Best First Search

Metode pencarian best first search merupakan kombinasi dari metode depth first search dan breadth first search dengan mengambil kelebihan – kelebihan dari kedua metode tersebut.Pada metode best first search ini, pencarian node tujuan atau goal, mengizinkan untuk mengunjungi sebuah node yang ada pada level yang lebih rendah jika ternyata node yang berada pada level yang lebih tinggi memiliki nilai heuristik yang lebih buruk (rendah).Hal ini sangat berbeda apabila pencarian dilakukan dengan metode hill climbing, dimana pada metode hilll climbing tidak diperbolehkannya untuk mengunjungi sebuah node pada level yang rendah yang meskipun node tersebut mempunyai nilai heuristik yang lebih baik (tinggi)

Pada setiap langkah proses pencarian terbaik pertama, kita memilih node-node dengan menerapkan fungsi heuristik yang memadai pada setiap node/simpul yang kita pilih dengan menggunakan aturan-aturan tertentu untuk menghasilkan penggantinya.Fungsi Heuristik yang digunakan merupakan prakiraan (estimasi) *cost* dari *initial state* ke *goal state,* yang dinyatakan dengan :

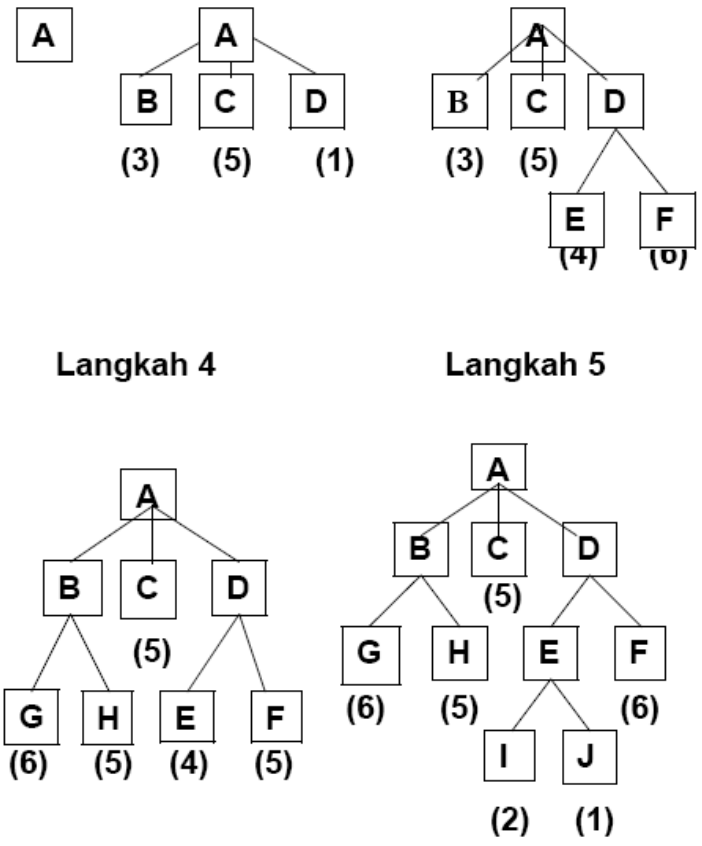
$f' = g + h'$

dimana f’ = prakiraan *cost* dari *initial* ke *goal*

g = *cost* dari *initial state* ke *current state*

h’ = prakiraan *cost* dari *current state* ke *goal state*

Contoh Proses pada *Best First Search:*



Gambar 2.5 Ilustrasi Pencarian Terbaik Pertama *(Best-First Search)*

2.13 Algoritma minimax

Algoritma Minimax merupakan algoritma yang digunakan untuk menentukan pilihan agar memperkecil kemungkinan kehilangan nilai maksimal. Algoitma ini diterpkan dalam permainan yang melibatkan dua pemain seperti tic tac toe, checkers, go dan permainan yang menggunakan strategi atau logika lainnya. Hal ini berarti permainan-permainan tersbut dapa dijelaskan sebagai suatu rangkaian aturan dan premis.

Algoritma ini mulai dikembangkan dari teori game zero-sum. Teori ini mendeskripsikan situasi dimana jika terdapat pemain yang mengalami pendapatan, pemain lain akan mengalami kehilangan dengan nilai yang sama dari pendapatan tersebut, dan sebaliknya. Jumlah pendapatan dari pemain yang dikurangi dengan jumlah kehilangan akan berjumlah nol.

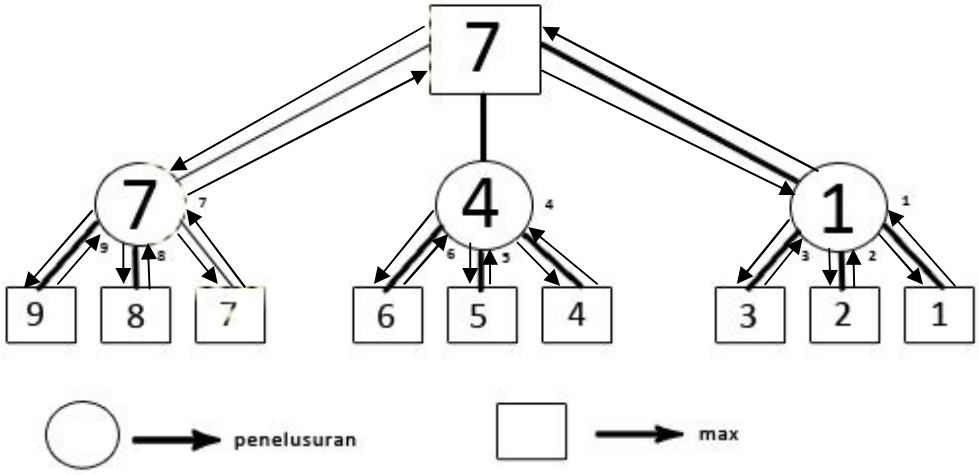
Teori minimax menyatakan :

Untuk setiap dua orang pemain dalam zero-sum game, terdapat nilai V dari strategi yang dimiliki pemain seperti :

1. Stratregi yang ditentukan pemain kedua akan menghasilkan konsekuensi kemungkinan untuk pemain pertama, V
2. Strategi yang dutentukan pemain pertama akan menghasilkan konsekuensi kemungkinan untuk pemain pertama, -V

Secara setara, strategi pemain pertama akan memastikan suatu nilai V tanpa memperdulikan strategi pemain kedua, dan bersamaan dengan itu pemain kedua akan memastikan dirinya kehilangan nilai sebesar –V.Algoritma Minimax merupakan algoritma dasar pencarian DFS (Depth-First Search) untuk melakukan traversal dalam pohon. DFS akan mengekspansi simpul paling dalam terlebih daulu. Setelah simpul akar dibangkitkan, algoritma ini akan membangkitkan simpul pada tingkat kedua, yang akan dilanjutkan pada tingkat ketiga, dst. Dalam melakukan treversal, misalkan dimulai dari suatu simpul i, maka simpul selanjutnya yang akan dikunjungi adalah simpul tetangga j, yang bertetangga dengan simpul k, selanjutnya pencarian dimulai lagi secara rekursif dari simpul j. Ketika telah mencapai simpul m, dimana semua simpul yang bertetangga dengannya telah dikunjungi, pencarian akan dirunutbalik ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul yang belum dikunjungi. Selanjutnya pencarian dimulai kembali dari j. Ketika tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi maka pencarian selesai.

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. Berikut merupakan gambar pohon untuk algoritma Minimax.



Gambar 2.6 Pohon Pencarian Algoritma Minimax

Dalam algoritma ini, langkah yang dapat dilakukan pemain ditentukan oleh langkah pemain lawan sebelumnya. Sebagai contoh pada tabel berikut di berikan tabel nilai yang memberitahukan hasil dari pilihan

	B memilih B1	B memilih B2	B memilih B3
A memilih A1	+3	-2	+2
A memilih A2	-1	0	+4
A memilih A3	-4	-3	+1

Gambar 2.7 Tabel Nilai Pilihan

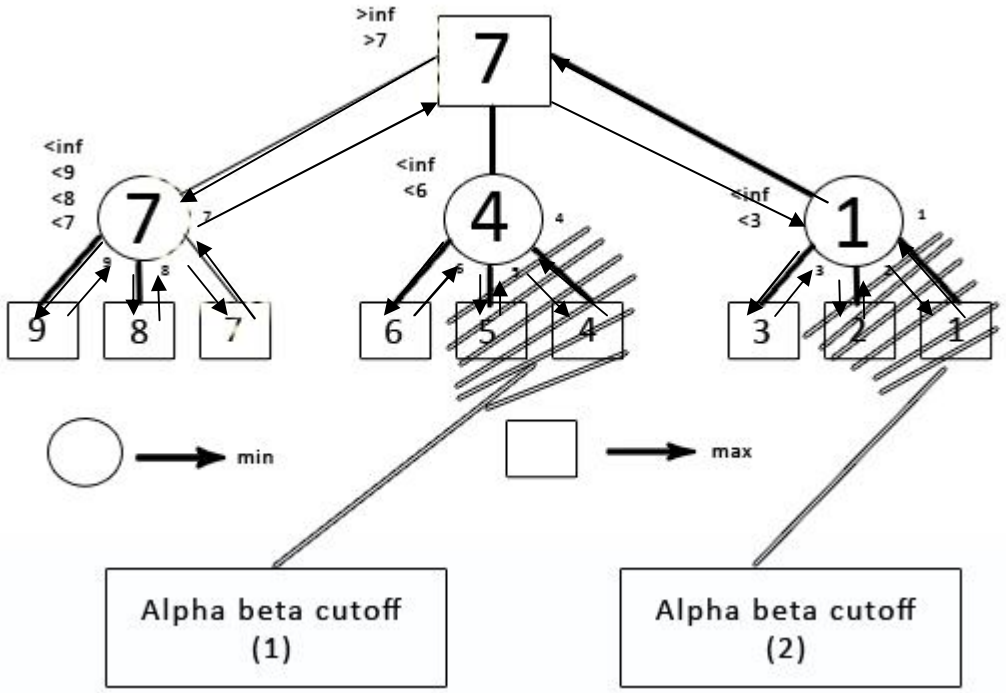
Pada tabel ini diperlihatkan setiap pemain memiliki tiga pilihan yang harus dipertimbangkan. Dengan mengasumsikan nilai pilihan yang dipilih untuk suatu pemain akan bernilai kebalikannya bagi pemain lawan. Maka pilihan minimal untuk A adalah A2 karena nilai terburuk adalah kehilangan -1, dengan pilihan minimax untuk B adalah B3 karena kemungkinan teburuk adalah mendapatkan nilai 1. Bagaimanapun, solusi ini tidak stabil, jika B mengira A akan memilih A2 maka B akan memilih B1 untuk mendapatkan nilai 1. Jika A mengira B akan memilih B1 maka A akan memilih A1 untuk mendapatkan 3, maka B akan memilih B2 yang dimana kedua pemain akan menyadari kesulitan menentukan pilihan. Disinilah dibutuhkan strategi. Pada beberapa pilihan, terlihat dominasi salah satu pemain dan dapat dieliminasi, seperti : A tidak akan memilih A3 karena A1 dan A2 memiliki hasil yang lebih baik, apapun yang B pilih. B tidak akan memilih B3 karena B2 akan memberikan hasil yang lebih baik, apapun yang A pilih. A dapat menghindari kehilangan lebih dari 1/3 dengan memilih A1 dengan kemungkina 1/8 dan A2 dengan kemungkinan 5/6 apapun yang B pilih. B dapat memastikan pendapatan setidaknya 1/3 dengan menggunakan strategi acak untuk memilih B1 dengan kemungkinan 1/3 atau B2 dengan kemungkinan 2/3 apapun yang A pilih.

Berdasarkan contoh tersebut diketahui bahwa dalam algoritma ini terdapat dua peran, yaitu max dan min.Pembuatan pohon dimulai dari posisi awal hingga posisi akhir permainan. Sekanjutnya, posisi akhir dievaluasi dari sudut pandang max,. Setelah itu, node bagian dalam diisi dengan nilai yang telah dievaluasi. Node yang

dimiliki max akan menerima nilai maksimum dari anak-anaknya. Node untuk min akan memilih nilai minumum dari anak-anaknya

2.14 Alpha-Beta Pruning

Dalam algoritma Minimax, pencarian dilakukan pada seluruh bagian pohon, sementara sebagian pohon tidak seharusnya diperiksa. *Alpha-beta pruning* merupakan modifikasi dari algoritma Minimax, yang akan mengurangi jumlah node yang dievaluasi oleh pohon pencarian. Pencarian untuk node berikutnya akan dipikirkan terlebih dahulu. Algoritma ini akan berhenti mengevaluasi langkah ketika terdapat paling tidak satu kemungkinan yang ditemukan dan membuktikan bahwa langkah tersebut lebih buruk jika dibandingkan dengan langkah yang diperiksa sebelumnya. Sehingga, langkah berikutnya tidak perlu dievaluasi lebih jauh. Dengan algoritma ini hasil optimasi dari suatu algoritma tidak akan berubah. Berikut merupakan pohon dengan algoritma *alpha-beta pruning*



Gambar 2.8 Pohon Pencarian Algoritma Minimax dengan *Alpha-Beta Pruning*

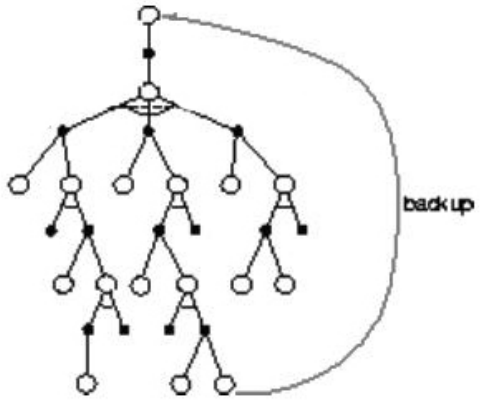
Diperlihatkan, pada pohon tersbut, bahwa terdapat pemotongan pencarian dengan menggunakan algoritma ini.Pada algoritma ini, terdapat dua nilai yang diatur, yaitu alpha dan beta, yang merepresentasikan nilai minum dari *max* yang diyakini dan nilai maksimum dari *min* yang diyakini. Nilai awal alpha adalah tak hingga negatif dan nilai awal beta adalah tak hingga positif. Sebagai hasil dari proses rekursif, area pencarian akan semakin kecil. Ketika beta menjadi lebih kecil dari alpha, akan berarti posisi saat itu tidak dapat menjadi hasil terbaik permainan untuk kedua pemain dan pencarian tidak perlu dilakukan lebih jauh. Pseudocode untuk algoritma Minimax

yang telah mengimplementasikan alpha beta pruning, yaitu : Pembentukan pohon DFS biasa membutuhkan big-O sebesar O(bm) dan dengan *alpha-beta pruning* pohon big- O akan menjadi sebesar O(bm/2).

2.15 Penerapan Algoritma Minimax Pada Permainan Chekers

Penerapan algoritma Minimax dalam *checkers* dibuat berdasarkan prosedur Minimax untuk mendapatkan langkah terbaik dari posisi yang ada. Setiap posisi memiliki nilai yang dapat dihasilkan dari langkah terbaik, dengan berasumsi bahwa AI akan selalu mencoba memaksimalkan nilai, ketika lawan akan mencoba untuk meminimalkannya.Ketika prosedur minimax mencapai akar pada pohon pencarian (posisi saat tersebut), akan menghasilkan langkah terbaik dengan asumsi lawan akan menggunakan kriteria evaluasi yang sama. Beberapa versi program yang dibuat kebanyakan telah menerapkan algoritma pemotongan *alpha-beta*.Terdapat dua macam metode, yang disebut *rote learning*. Metode tersebut memiliki penyimpan untuk setiap posisi yang ditemui selama permainan dengan tidak menghilangkan nilai yang ditentukan oleh prosedur Minimax. Hasilnya adalah jika terdapat posisi yang pernah ditenukan sebelumnya, akan dimunculkan sebagai posisi terminal pada pohon pencarian. Sehingga, pencarian akan semakin mudah karena nilai posisi diambil dari hasil pencarian yang telah dilakukan sebelumnya. Satu masalah awal yang ditemukan adalah program tidak mendukung untuk melangkah langsung menuju kemenangan.

Solusi pencegahan adalah dengan mengurangi sedikit nilai posisi setiap tahap (disebut *ply*) pada analisis Minimax. Jika program berhadapan dengan pilihan posisi dengan nilai yang hanya dibedakan oleh *ply*, maka program akan secara otomatis melangkah pada pilihan yang paling menguntungkan.. Ide ini digambarkan pada gambar berikut ini.



Gambar 2.9 Diagram *Backup* Permainan *Checkers*

Setiap lingkaran putih merepresentasikan posisi langkah program berikutnya dan setiap lingkaran hitam merepresentasikan posisi langkah lawan selanjutnya. *Backup* dibuat untuk setiap nilai pada posisi setelah perpindahan sisi, yang akan menghasilkan langkah berikutnya. Hal ini dibuat berdasarkan nilai yang dihasilkan dari algoritma Minimax. Perkembangan *checkers* menggunakan algoritma Minimax banyak dipengaruhi oleh pembuatan yang dilakukan Samuel tersebut. Untuk

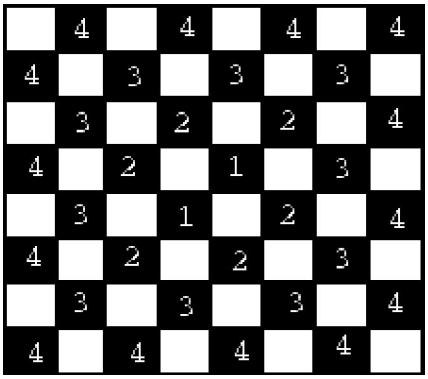
menerapkan algoritma Minimax pada permainan *checkers* diperlukan suatu fungsi optimasi tertentu yang ditambahkan. Salah satu fungsi optimasi yang paling dasar adalah membatasi kedalaman dari pohon pencarian. Jika permainan memiliki pohon 3-*ary*, maka pohon tersebut akan memiliki nilai yang diperlihatkan pada tabel berikut.

Depth	*Node Count*
0	1
1	3
2	9
3	27
...	...
n	3^n

Gambar 2.10 Gambar Kedalaman Pohon Pencarian

Berdasarkan tabel tersebut dapat dilihat bahwa untukpohon pencarian dengan kedalaman 5 akan membutuhkan 1+3+9+27+81+243 = 364 * 1s = 364s = 6m. Waktu ini merupakan waktu yang sangat lama untuk ukuran permainan. Fungsi optimasi selanjutnya yang perlu ditambahkan adalah fungsi yang dibutuhkan untuk melakukan evaluasi posisi permainan dari pemain tertentu. Hal ini dapat dilakukan dengan memberikan nilai pada langkah tertentu pada permainan, seperti menghitung jumlah kepingan di papan atau jumlah langkah yang tersisa di akhir permainan. Sebagai pengganti sebaiknya diperlukan suatu fungsi estimasi yang dapat melakukan penghitungan kemungkinan posisi agar pemain dapat memenangkan permainan. Fungsi ini harus memiliki fungsi heuristik dari permainan tersebut. Pada *checkers*, kepingan pada pojok dan pinggir posisi tidak buat suatu fungsi dapat dimakan. Sehingga, dapat dibuat suatu fungsi yang memberikan nilai yang lebih tinggi pada

posisi tersebut. Sebagai ilustrasi dapat dilihat pada gambar 5. Untuk memperkecil kemungkinan, fungsi heuristik nilai kepingan juga dapat ditambahkan, misalanya raja yang memiliki nilai lebih dibanding kepingan biasa.



Gambar 2.11 Pemberian Nilai pada *Checkers*

Artificial Intelegent pada permainan *checkers* dapat dikembangkan untuk memiliki dua kemungkinan metode pencarian *alpha-beta*. Pertama, pohon pencarian akan mencari hingga kedalaman tertentu, misalkan, pada sebuah permainan ditetapkan kedalaman pohon tingkat 4-5 untuk *Artificial Intelegent* pada level *beginner*, kedalaman 6-8 untuk level *intermediate* dan kedalaman 9-10 untuk level *advanced*. Metode kedua yaitu memungkinkan *Artificial Intelegent* untuk mencari dalam waktu tertentu. Metode ini dianggap lebih baik, karena jika bergantung pada keadaan permainan sejumlah langkah yang mungkin untuk setiap posisi, pencarian berdasarkan kedalaman akan menghasilkan variasi pohon yang sangat berbeda-beda. Pada metode tersebut, pengurangan pohon pada kedalaman rendah tidak diperlukan. Dengan mencari pada waktu tertentu, *Artificial Intelegent* memulai pencarian pada

kedalaman 4 dan melakukan pencarian lebih dalam secara iteratif dengan menambahkan kedalaman sebanyak 1 pada setiap pencarian. Jika waktu yang ditentukan habis pada tengah pencarian, pencarian akan dihentikan pada tingkatan tersebut dan langkah akan dihasilkan dari pencarian sebelumnya.

2.16 Microsoft Visual Basic

Merupakan sebuah bahasa pemrograman yang bersifat event driven dan menawarkan Integrated Development Environment (IDE) visual untuk membuat program aplikasi berbasis sistem operasi Microsoft Windows dengan menggunakan model pemrograman Common Object Model (COM). Visual Basic merupakan turunan bahasa BASIC dan menawarkan pengembangan aplikasi komputer berbasis grafik dengan cepat, akses ke basis data menggunakan Data Access Objects (DAO), Remote Data Objects (RDO), atau ActiveX Data Object (ADO), serta menawarkan pembuatan kontrol ActiveX dan objek ActiveX. Beberapa bahasa skrip seperti Visual Basic for Applications (VBA) dan Visual Basic Scripting Edition (VBScript), mirip seperti halnya Visual Basic, tetapi cara kerjanya yang berbeda. Para programmer dapat membangun aplikasi dengan menggunakan komponen-komponen yang disediakan oleh Microsoft Visual Basic Program-program yang ditulis dengan Visual Basic juga dapat menggunakan Windows API, tapi membutuhkan deklarasi fungsi eksternal tambahan.IDE (Integrated Development Environment) adalah program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan

perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak.

Sebuah IDE, atau secara bebas dapat diterjemahkan sebagai Lingkungan Pengembangan Terpadu, setidaknya memiliki fasilitas:

1. Editor, yaitu fasilitas untuk menuliskan kode sumber dari perangkat lunak.
2. Compiler, yaitu fasilitas untuk mengecek sintaks dari kode sumber kemudian mengubah dalam bentuk binari yang sesuai dengan bahasa mesin.
3. Linker, yaitu fasilitas untuk menyatukan data binari yang beberapa kode sumber yang dihasilkan compiler sehingga data-data binari tersebut menjadi satu kesatuan dan menjadi suatu program komputer yang siap dieksekusi.
4. Debuger, yaitu fasilitas untuk mengetes jalannya program, untuk mencari *bug*/kesalahan yang terdapat dalam program.

Sampai tahap tertentu IDE modern dapat membantu memberikan saran yang mempercepat penulisan. Pada saat penulisan kode, IDE juga dapat menunjukan bagian-bagian yang jelas mengandung kesalahan atau keraguan.