Teknik-Teknik Search, Sebelum membahas tentang teknik-teknik pencarian, maka perlu dibicarakan tentang hal-hal penting yang muncul: Arah search, Topologi proses search tersebut, Memilih aturan-aturan yang dapat diterapkan, Penggunaan fungsi-fungsi heuristik untuk untuk memandu proses search tersebut.

Arah search dapat dilakukan: maju, bermula dari keadaan awal (start state); mundur, diawali dari keadaan tujuan (goal state).

TOPOLOGI PROSES SEARCH, ada dua macam penggambaran problem, yaitu dalam bentuk: pohon (tree) & graf (graph).

POHON (TREE), Merupakan graf dimana dua simbol memiliki paling banyak satu lintasan yang menghubungkannya. Tidak dimungkinkan adanya loop pada pohon.

GRAF (GRAPH), Graf dibedakan antara: Graf berarah dan Graf tidak berarah.

Graf disebut berarah bila lintasannya mempunyai arah, umumnya digambarkan dengan anak panah. Untuk garf berakar mempunyai simpul unik yang disebut akar sedemikian rupa sehingga terdapat lintasan dari akar tersebut ke semua simpul pada graf.

Ada beberapa cara untuk mencari kemungkinan penyelesaian, yaitu: Depth-First Search, Breadth-First Search, Hill-Climbing Search, Least-Cost Search, Best-First Search.

Evaluasi sebuah pencarian(search), Evaluasi penampilan sebuah teknik pencarian (search) akan sangat kompleks. Dasar pengukuran dari evaluasi: seberapa cepat search menemukan penyelesaian, seberapa cepat search menemukan penyelesaian yang baik.

Kecepatan search ditentukan: panjang lintasan, jumlah sesungguhnya penelusuran node.

Penelusuran (pencarian) Depth First berarti setiap kemungkinan path ke goal digali (eksplorasi) ke kesimpulannya sebelum path lainnya dicoba.

Pencarian Depth-First adalah kemungkinan metoda terbaik yang dapat diikuti di mana Heuristic tidak digunakan. Turbo Prolog menggunakan pencarian Depth-First.

Teknik Pencarian Breadth-First Merupakan kebalikan dari teknik pencarian Depth-First. Pada metoda ini diperiksa setiap node pada level yang sama sebelum mengolah ke level berikut yang lebih dalam.

Pada metoda depth-First dan Breadth-First: Pencarian terletak pada pemindahan dari satu goal ke goal yang lain tanpa menggunakan tebakan yang terarah. Kedua metoda tersebut baik untuk situasi-situasi yang terkendali. Bila tidak maka dibutuhkan penambahan heuristic. Dasar dari metoda pencarian heuristic: Maximizing atau minimizing beberapa aspek dari problem (masalah).

HEURISTIK, Dari kata Yunani : heuriskein artinya “to discover” artinya menemukan, adalah sebuah teknik yang memperbaiki hasil efisiensi dari sebuah proses penelusuran / pencarian, kemungkinan dengan klaim-klaim korban dari kesempurnaan.

Keuntungan dari Depth first search: depth-first search membutuhkan sedikit memory, karena hanya nodenode pada path aktif (current) yang disimpan. Ini sangat berbeda dengan breadth-first search, dimana semua part dari tree sepanjang telah dihasilkan harus disimpan. Secara kebetulan (atau jika penanganan diambil dalam urutan state pengganti alternatif). Depth first search dapat menemukan sebuah solusi tanpa uji coba beberapa penelusuran tempat kosong pada keseluruhannya. Ini berbeda dengan breadth-first search, semua bagian dari tree harus di ujicoba pada level n sebelum beberapa node pada level n+1 dapat di ujicoba. Ini secara khusus pasti jika beberapa solusi dapat diterima. Depth-first search dapat berhenti bila salah satu dari mereka ditentukan.

Fungsi/Function Heuristik adalah: Fungsi yang melakukan pemetaan (mapping) dari diskripsi keadaan masalah (problema) ke pengukur kebutuhan, umumnya direpresentasikan berupa angka. Sangat penting dalam Expert system sebagai komponen esensial dalam memecahkan persoalan.

George Polya mendefinisikan heuristik sebagai studi metode & aturan penemuan.

Dalam proses search ruang keadaan (state space), heuristik dinyatakan sebagai aturan untuk melakukan pemilihan cabang dalam ruang keadaan yang paling dapat diharapkan mencapai pemecahan masalah yang dapat diterima.

AI menggunakan heuristik dalam dua situasi dasar:

Persoalan/problema yang mungkin memiliki solusi eksak, namun biaya perhitungan untuk menemukan solusi tersebut sangat tinggi dalam kebanyakan persoalan (seperti catur), ruang keadaan bertambah secara luar biasa seiring dengan jumlah.

Persoalan yang mungkin tidak memiliki solusi eksak karena ambiquitas (keraguan atau ketidakpastian) mendasar dalam penyataan persoalan atau data yang tersedia, Diapnosa medis merupakan salah satu contohnya.

Heuristik menangani kerumitan masalah dengan cara memadu proses search pada sepanjang lintas yang paling dapat diharapkan. Namun heuristik juga bisa salah. Oleh karena itu heuristik hanyalah sebuah cara menerka langkah berikutnya yang harus diambil dalam memecahkan suatu persoalan berdasarkan informasi yang ada/tersedia.

Algoritma pencarian mencoba untuk menemukan solusi pertama yang meminimkan jumlah hubungan dengan menggunakan informasi heuristic. Dalam AI metoda pencarian yang menggunakan informasi heuristic disebut: Hill Climbing.

Pada umumnya algoritma Hill Climbing memilih langkah berikut node yang berada ditempat yang terdekat dengan goal.

Kebalikan dari pencarian Hill Climbing adalah pencarian Least Cost. Pada metoda ini: hubungan terpendek akan diambil sehingga route yang ditemukan mempunyai kemungkinan yang baik untuk mendapat jarak terpendek. Jadi Hill Climbing: meminimkan jumlah hubungan. Least Cost: meminimkan jumlah jarak yang ditelusuri.

Best first search merupakan suatu teknik pencarian yang mengkombinasikan yang terbaik diperoleh dari teknik depth-first search dan Teknik breadth-first search ke dalam sebuah metode tunggal.

Prosedur Pencarian Minimax adalah langkah depth-first, idenya adalah memulai pada saat sebagian posisi dan menggunakan plausible-move generator untuk menggerakkan himpunan pada posisi possible successor. Sekarang kita dapat mengaplikasikan fungsi evaluasi statis ke posisinya dan memilih salah satunya yang terbaik.

Setelah melakukannya, kita dapat mengembalikan nilai pada permulaan posisi untuk memperoleh hasil evaluasi. Permulaian posisi sangat tepat sekali untuk dilakukan oleh

kita sebagai posisi yang digerakkan oleh langkah kita yang terbaik untuk selanjutnya. Disini kita mengamsumsikan bahwa fungsi evaluasi statis akan kembali ke nilai yang lebih besar yang mengindentifikasikan kita untuk situasi yang tepat, lalu tujuan akhir (goal) adalah memaksimalkan nilai pada fungsi evaluasi statis pada posisi berikutnya.

Contohnya dapat kita lihat pada gambar 4.1. pada gambar tersebut diamsumsikan bahwa fungsi evaluasi statis akan kembali pada nilai yang ditunjukkan pada –10 to 10, dengan 10 mengindentifikasikan kemenangan untuk kita, -10 menang untuk opponent, dan 0 untuk pertandingan berikutnya. Sejak tujuan akhir kita (goal) adalah untuk memaksimalkan nilai pada fungsi heuristic, kita memilih untuk mengerakkannya ke B, simpan nilai B ke A, kita dapat menyatakan bahwa nilai A adalah 8, sejak kita mengetahuinya kita dapat mengerakkannya ke posisi yang bernilai 8.

Tetapi sejak kita mengetahui bahwa fungsi evaluasi statis tidaklah lengkap, kita memutuskan untuk membawanya ke tempat yang lebih jauh di banding satu permainan. Ini penting misalnya pada permainan catur dimana kita berada ditengah diantara benteng.

Sesudah kita bergerak, situasinya haruslah baik, tetapi jika kita melihat satu pergerakkan raja, kita dapat melihat bahwa salah satu benteng kita akan di makan dan situasi ini tidaklah baik untuk dilihat, lalu kita dapat melihat raja lihatlah apa yang terjadi pada langkah berikutnya yang dapt dilihat oleh opponent-nya.

Paling sedikit pada pengaplikasian fungsi evaluasi statis untuk beberapa posisi yang kita kembangkan, kita dapat mengaplikasikan plausible-move generator, pergerakkan posisi successor untuk beberapa posisi.

Tetapi kita harus dapat membawa ke dalam perhitungan berarti opponent dapat kita pilih berdasarkan successor yang bergerak untuk membuat dan demikian yang mana nilai terminal seharusnya di backup untuk level berikutnya.

Berdasarkan yang kita buat pada langkah B. lalu opponent harus memilih diantara langkah E, F dan G. pada goal opponent adalah meminimalkan nilai pada fungsi evaluasi, lalu dia dapat memilih untuk meggerakkan F. ini berarti jika kita menggerakkan B, posisi aktualnya memungkinkan suatu konfigurasi yang dipersembahkan oleh E, yang sangat baik untuk kita.

Tetapi sejak kita berada pada level tersebut kita tidak dapat melakukan satu langkah, kita tidak akan melakukan pilihan tersebut. Pada gambar 4.1. kita dapat melihat hasil propaganda pada nilai yang baru pada sebuah tree. Pada level ini mempersembahkan pilihan opponent, nilai minimum dapat dipilih dan di back up.

Pada level ini kita mempersembahkan pilihan kita, nilai maksimum yang kita pilih. Pada nilai yang pertama dari permainan kedua di backup, ini akan menjadi langkah yang tepat untuk kita yang kita buat pada level pertama, memberikan informasi yang memungkinkan adalah C. Sejak tidak terdapat opponent maka tidak dapat dilakukan dari sini untuk memproduksi nilai salah lebih dari –2. Proses ini dapat diulang untuk beberapa waktu, dan lebih akurat evaluasinya, sehingga hasilnya dapat digunakan untuk langkah berikutnya pada level teratas. Alternatif pada maximizing dan minimizing pada alternatif permaianan, ketika evaluasi ditekan oleh korespodennya untuk strategi mengopponent pada dua pemain dan memberikan pengertian nama minimax.

Memiliki informasi yang menjelaskan pengoperasian prosedure minimax, kita sekarang menggambarkan secara lengkap. Ini merupakan langkah tepat pada prosedur rekursif bahwa ada dua alasan yang menjelaskan secara spesifik mengenai permainan yang dimainkan.

MOVEGEN (position, Player), Plausible-move generator, yang mengembalikan node yang mempersembahkan langkah yang dapat dibuat oleh pemain dalam suatu posisi.

Kita menyebutnya sebagai dua pemain yaitu pemain 1 dan pemain 2; pada program catur, kita seharusnya menggunakan nama HITAM dan PUTIH.

STATIC (position, player), Fungsi evaluasi statis, yang mengembalikan angka yang mempersembahkan kebaikan pada suatu posisi dari suatu penilaian oleh pemain.

Dengan beberapa program rekursif, isu kritikal dalam suatu design prosedur minimax dapat diberhentikan pada pengrekursiannya dan dapat dipanggil dengan mudah oleh fungsi evaluasi statis. Dapat dinilai factor varietasnya yang menyebabkan suatu keputusan.

Untuk pusat dari prosedur minimax dibahas disini , kita dapat menggunakan DEEP-ENOUGH, yang mengasumsikan untuk megevaluasi semua faktor dan memberikan TRUE jika pencarian dapt dihentikan pada sebagian level dan sebaliknya FALSE.

Kita menggunakan sangat sederhana pada DEEP-ENOUGH yang dapat mengambil dua parameters, posisi dan Depth. Ini dapat menyebabkan ketidakperdulian pada posisi parameters dalam pemberian nilai TRUE, jika pada DEPTH parameter memotong nilai yang konstan.

Suatu permasalahan yang menjelaskan minimax sebai prosedur rekursif yang dibutuhkan untuk tidak mengembalikan salah satunya tetapi terdapat dua hasil: Pemback up-an nilai pada path yang dipilih; Path tersebut. Kita mengembalikannya kedalam bagian path yang memungkinkan hanya sebagian elemen, yang mempersembahkan langka yang baik dari sebagian posisi, yang sangat dibutuhkan.

Kita mengasumsikannya bahwa pengembalian struktur minimax menghubungkan hasil keduanya dan terdapat dua fungsi yaitu VALUE dan PATH, yang mengekstrak sebagian komponen.

Sejak kita menemukan struktur prosedur minimax sebagai fungsi rekursif, kita harus dapat mejelaskan secara spesifik bagaimana ia dapat dipanggil inisialnya. Ini membutuhkan tree parameter, pada board posisi, pada sebagian DEPTH pada suatu pencarian dan pemain untuk menjalankannya. Maka penginisialannya dapat dipanggil untuk menghitung langkah yang terbaik pada posisi CURRENT seharusnya.