BAB II

LANDASAN TEORI

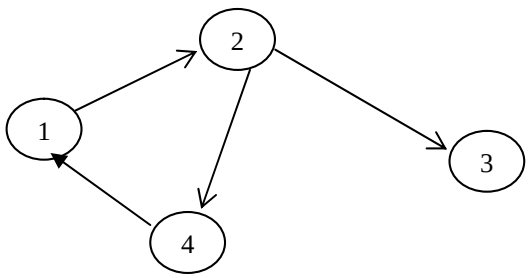
Untuk keperluan pembahasan tentang masalah pencarian jarak terpendek maka jaringan dimodelkan kedalam suatu graf. Pemodelan jaringan dengan graf akan diuraikan paada bab ini yang meliputi : terminologi, macam dan karakteristik graf, masalah penentuan jarak terpendek (*Simulated Annealing* dan algoritma Djikstra).

2.1 Tinjauan Teori Graf

Suatu graf dapat menjadi suatu solusi jika masalah yang timbul terlebih dahulu direpresentasikan informasinya secara jelas dan terarah.

2.1.1 Definisi Graf

Graf adalah suatu diagram yang memuat informasi yang dapat diinterprestasikan secara tepat. Definisi formal dari graf G = (*V,E*) terdiri dari sekumpulan objek V = {*vi,v2,v3.....*} yang disebut sebagai simpul (*vertex/node*) dan sebuah himpunan lain yang disebut dengan busur (*edge/arc*), dimana setiap busur e_k diidentifikasi dengan simpul tak terurut (*vi,vj*).Contoh sebuah graf dengan empat simpul dan empat busur dibawah ini :



V= { 1,2,3,4 } dan E = { (1,2),(2,3),(2,4),(4,1) }

Gambar 2.1. Contoh gambar graf G = (*V,E*)

2.1.2 Macam – Macam Graf

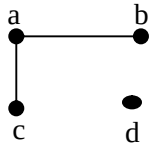
Suatu graf G dapat dibedakan menurut jenis garisnya, yaitu :

2.1.2.1 Graf Tak Berarah

a. Graf Bipartite

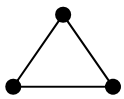
Graf bipartite terbagi dua bagian :

1. Graf sederhana adalah graf yang tidak mempunyai loop ataupun garis paralel. Contoh graf sederhana :



Gambar 2.2. Graf sederhana

2. Graf lengkap dengan n titik adalah graf sederhana dengan n titik, di mana setiap 2 titik berbeda dihubungkan dengan satu garis.Banyaknya garis dalam suatu graf lengkap dengan n titik adalah $\frac{n(n-1)}{2}$ buah.



Gambar 2.3. Graf lengkap

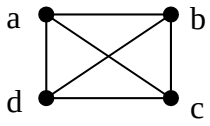
b. Komplemen Graf

Komplemen suatu graf G (simbol $\overline{G}$) dengan n titik adalah suatu graf dengan :

1. Titik-titik $\overline{G}$ sama dengan titik-titik G. Jadi $V(\overline{G}) = V(G)$

2. Garis-garis $\overline{G}$ adalah komplemen garis-garis G terhadap graf lengkapnya (K_n), maka $E(\overline{G}) = E(K_n) - E(G)$. Titik-titik yang dihubungkan dengan garis

dalam G tidak terhubung dalam *G* . Sebaliknya, titik-titik yang tidak terhubung dalam G menjadi terhubung dalam *G*. Contoh graf komplemen :



Gambar.2.4. Graf komplemen

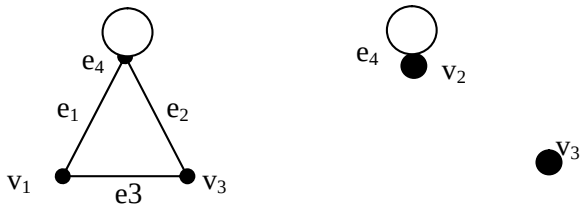
c. Sub Graf

Misalkan G adalah suatu graf. Graf H dikatakan subgraf G bila dan hanya bila :

a. V (H) $\subseteq$ V (G)

b. E (H) $\subseteq$ E (G)

c. Setiap garis dalam H mempunyai titik ujung yang sama dengan garis tersebut dalam G. Contoh sub graf



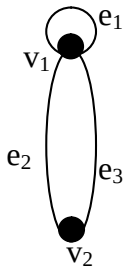
G H

Gambar 2.5. Sub graf

E(H) = {e_4} dan E(G) = {e_1,e_2,e_3,e_4} sehingga E(H) $\subseteq$ E(G). Garis e_4 di H merupakan loop pada v_2 dan garis e_4 juga merupakan loop pada v_2 di G.

d. Derajat

Misalkan v adalah titik dalam suatu Graf G. Derajat titik v (simbol d(v)) adalah jumlah garis yang berhubungan dengan titik v dan garis suatu loop dihitung dua kali. Derajat total g adalah jumlah derajat semua titik dalam G.

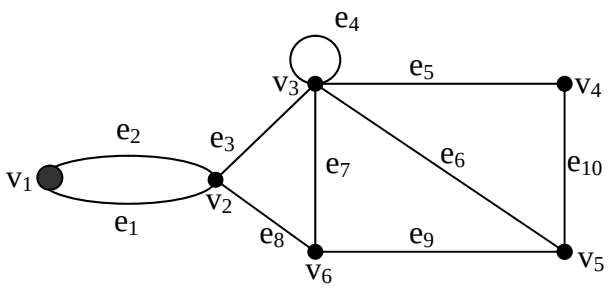


Gambar 2.6. Graf berderajat

e. Path dan Sirkuit

Sirkuit adalah suatu jalan tertutup dimana tidak ada simpul (kecuali simpul awal dan simpul akhir) yang dikunjungi lebih dari sekali. Sirkuit dikenal juga dengan istilah siklus, poligon atau lintasan sirkuler.

Misalkan G adalah suatu graf. Misalkan pula v dan w adalah titik dalam graf G. Suatu Walk dari v ke w adalah barisan titik-titik berhubungan dan garis berselang-seling, diawali dari titik v dan diakhiri pada titik w.



Gambar 2.7 Path dan sirkuit

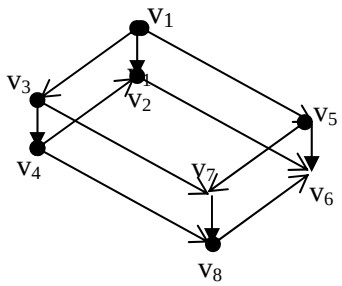
f. Sirkuit Euler

Misalkan G adalah suatu graf. Sirkuit Euler adalah sirkuit dimana setiap titik dalam G muncul paling sedikit sekali dan setiap garis dalam G muncul tepat satu kali.

2.1.2.2 Graf Berarah

a. Path Berarah dan Sirkuit dan sirkuit Berarah

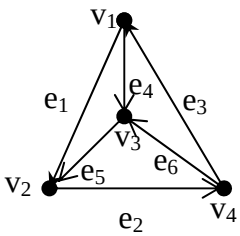
Dalam graf berarah perjalanan yang dilakukan harus mengikuti arah garis. Dan suatu graf berarah yang tidak memuat sirkuit berarah disebut Asklik.



Gambar 2.8. Graf path berarah

b. Graf Berarah Terhubung

Suatu graf tak berarah disebut terhubung jika ada *walk* yang menghubungkan setiap dua titiknya. Misalkan G adalah suatu garaf berarah dan v,w adalah sembarang 2 titik dalam G. G disebut terhubung kuat jika ada path berarah dari v ke w.



Gambar 2.9 Graf berarah terhubung

2.2 Penelusuran Simpul

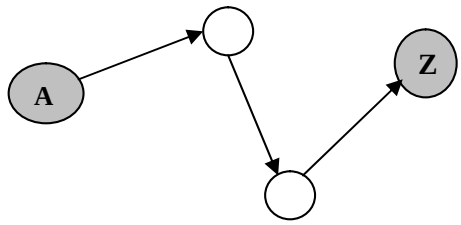
Penelusuran simpul yaitu melakukan pencarain dengan cara mencari jarak dari suatu simpul ke simpul lainnya, mencari simpul mana saja yang dilalui, yang

akhirnya dapat ditemukan jarak yang paling terpendek sebagai solusi pada sistem yang dibuat.

Masalah penentuan jarak atau rute yang menekankan pada penelusuran simpul ke dalam dua kelompok, yaitu rute dengan :

a. Awal dan akhir tujuan berbeda

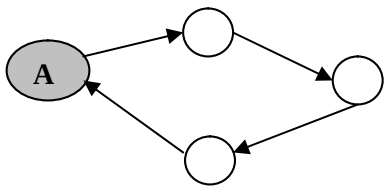
Masalah yang timbul pada proses lintasan yang berdasar daerah awal dan tujuan harus berbeda, yaitu menentukan rute yang dimulai dari simpul X dan diakhiri pada simpul Y sedemikian rupa sehingga diperoleh jarak yang minimal (terpendek). Masalah ini sering disebut sebagai masalah jarak terpendek (*shortest path problem*).



Gambar 2.10 Simpul awal dan simpul tujuan berbeda

b. Awal dan akhir rute sama

Masalah yang timbul adalah bagaimana menentukan jarak atau rute yang dimulai dan diakhiri pada simpul atau tujuan yang sama, sehingga semua simpul yang terdapat pada jaringan terkunjungi dan jarak tempuhnya minimal. Contohnya adalah *Traveling Salesman Problem* (TSP).



Gambar 2.11. Simpul awal dan simpul tujuan sama

2.3 Algoritma Jarak Terpendek

Tujuan dasar dari penjaluran jarak terpendek adalah mencari lintasan antara simpul sumber dengan simpul tujuan yang mengakumulasikan nilai parameter jaringan pada lintasan. Jika parameter berupa panjang lintasan, maka harus diminimasi jaraknya.

Algoritma dan metode yang digunakan untuk mencari rute optimum antara simpul asal dan simpul tujuan dimana nilai parameter yang diasosiasikan pada setiap busur telah diketahui disebut dengan algoritma dan metode rute terpendek.

Terdapat empat kategori dari permasalahan rute terpendek, antara lain :

1. Rute terpendek antara dua simpul yang tidak sama
2. Rute terpendek antara semua pasangan simpul
3. Rute terpendek dari suatu simpul ke semua simpul
4. Rute terpendek dari suatu simpul ke simpul lainnya yang melalui simpul-simpul tertentu yang ditentukan.

Rute terpendek yang akan dibahas adalah kategori pertama, yaitu antara dua simpul yang tidak sama. Terdapat metode dan algoritma yang digunakan untuk kategori pertama diantaranya untuk metode yaitu *Simulated Annealing*. Sedangakan untuk algoritma yaitu algoritma Djikstra.

2.3.1 Algoritma Djikstra

Algoritma Djikstra adalah algoritma yang khusus menemukan bentuk unik dari *spaning tree* untuk grafik. idenya adalah memulai dengan sebuah titik yang diberikan label $V_{0,}$ dan setiap langkah menambahkan titik yang terdekat dengan

V_0 ketika dihubungkan kedalam sebuah pohon (*tree*). Spaning tree yang dihasilkan selalu berisi satu dari lintasan terpendek dari V_0 ke setiap titik lainnya.

Misalkan G adalah graf berarah berlabel dengan titik-titik $V(G) = \{v_1, v_2, v_3, ..., v_n\}$ dan path terpendek yang dicari adalah dari v_1 ke v_n. Algoritma Djikstra dimulai dari titik v_1. Pada iterasi yang dihasilkan, algoritma ini akan mencari satu titik yang jumlah bobotnya dari titik 1 terkecil. Titik-titik yang terpilih akan dipisahkan, dan titik-titik tersebut tidak diperhatikan lagi dalam iterasi berikutnya.Misalkan :

$V(G)$ = $\{v_1, v_2, v_3, ..., v_n\}$

L = Himpunan titik $\in$ V(G) yang sudah terpilih dalam alur path terpendek.

D(j) = Jumlah bobot path terkecil dari v_i ke v_j.

w(i,j) = Bobot garis dari titik v_i ke titik v_j.

w*(1,j) = Jumlah bobot path terkecil dari v_1 ke v_j.

Secara jelas, algoritma Djikstra untuk mencari path terpendek adalah sebagai berikut :

1. L = { };

   V = { v_2, v_3, ..., v_n}.

2. Untuk i = 2, ..., n, lakukan D(i) = W (1,i)
3. Selama $v_n \notin$ L lakukan :

   a. Pilih titik $v_k \in$ V-L dengan D (k) terkecil

   L = L $\cup$ {v_k}

   b. Untuk setiap $v_j \in$ V-L lakukan :

Jika D (j) > D (k) + W (k,j) maka ganti

D (j) dengan D(k) + W (k,j)

4. Untuk setiap vj $\in$ V, w* (1,j) = D (j)

Algoritma Djikstra memberikan label pada setiap simpul dari suatu graf. Pada setiap langkah algoritma beberapa simpul memiliki label permanen dan yang lainnya adalah label sementara. Algoritma ini memulai dengan memberikan nilai 0 terhadap simpul asal s, dan label sementara ∞ kepada n-1 simpul sisanya. Selanjutnya dalam setiap iterasi simpul lainnya mendapat label permanen berdasarkan aturan sebagai berikut :

1. Untuk setiap simpul yang belum berlabel permanen dapatkan label sementara yang harganya diberikan oleh :

   min[label lama dari j, (label lama dari I + dij)],

   dimana i adalah simpul terakhir yang berlabel permanen pada iterasi sebelumnya, dan dij adalah jarak langsung antara simpul i dan simpul j. Jika i dan j tidak terhubung dengan suatu busur, maka dij = ∞.
2. Cari nilai terkecil diantara semua label sementara, dan ubah status simpul tersebut menjadi label permanen. Jika terdapat lebih dari satu nilai terkecil, pilih salah satu.
3. Jika label permanen terakhir adalah simpul tujuan t, maka berhenti (rute terpendek telah ditemukan). Jika tidak ulangi langkah 1 dan langkah 2.

Algoritma diatas menghasilkan jarak terpendek antara titik asal dan titik tujuan, tidak menghasilkan urutan jarak terpendeknya. Jarak terpendek bisa diperoleh dengan cara *backward* yang dimulai dari titik tujuan bergerak mencari

simpul predesesornya yang berlabel tetap dan selisih nilai simpulnya sama dengan panjang busur yang menghubungkan dua simpul tersebut. Algoritma ini berhenti jika suatu saat predesesor yang dikunjungi adalah titik asal. Alternatif lain dalam mencari jarak terpendek adalah dengan menyimpan simpul-simpul yang berlabel permanen dalam sebuah vektor (*array*) berukuran n. Jika suatu saat simpul j berlabel permanen maka simpul predesesor simpul j disimpan dalam vektor pada elemen ke-j.

2.3.2 Metode *Simulated Annealing*

Ide dasar *Simulated Annealing* terbentuk dari pemrosesan logam. *Annealing* (memanaskan kemudian mendinginkan) dalam pemrosesan logam ini adalah suatu proses bagaimana membuat bentuk cair berangsur-angsur menjadi bentuk yang lebih padat seiring penurunan temperatur. *Simulated Annealing* digunakan untuk penyelesaian masalah yang mana perubahan keadaan dari satu kondisi ke kondisi yang lainnya membutuhkan ruang yang sangat luas, misalkan perubahan gerakan dengan menggunakan permutasi pada masalah *Traveling Salesman Problem*.

Pada *Simulated Annealing*, ada 3 parameter yang sangat menentukan, yaitu: teangga, *gain*, temperatur, pembangkitan bilangan *random* (r). Tetangga akan sangat berperan dalam membentuk perubahan pada solusi sekarang. Pembangkitan bilangan *random* akan berpengaruh implikasi adanya probabilitas

Secara jelas, algoritma Simulated Annealing sebagai berikut :

1. Evalusi keadaan awal. Jika keadaan awal merupakan tujuan, maka pencarian berhasil dan KELUAR. Jika tidak demikian, lanjutkan dengan menetapkan keadaan awal sebagai kondisi sekarang.
2. Inisialisasi BEST_SO_FAR untuk keadaan sekarang.
3. Inisialisasi T sesuai dengan penjadwalan
4. Kerjakan hingga solusi ditemukan atau sudah tidak ada operator baru lagi akan diaplikasikan ke kondisi sekarang.
   a. Gunakan operator yang belum pernah digunakan tersebut untuk menghasilkan kondisi baru.
   b. Evaluasi kondisi yang baru dengan menghitung :

      ΔE = nilai sekarang – nilai keadaan baru.

      i. Jika kondisi baru merupakan tujuan, maka pencarian berhasil dan KELUAR.

      ii. Jika bukan tujuan, namun memiliki nilai yang lebih baik daripada kondisi sekarang, maka tetapkan kondisi baru sebagai kondisi sekarang. Demikian pula tetapkan BEST_SO_FAR untuk kondisi yang baru tadi.

      iii. Jika niali kondisi baru tidak lebih baik dari kondisi sekarang, maka tetapkan kondisi baru sebagai kondisi sekarang dengan probabilitas :

$$P' = e^{-\Delta E / T} : (\text{exp-(Panjang-MinPanjangJalur)/T})$$

Langkah ini biasanya dikerjakan dengan membangkitkan suatu bilangan random (r) pada range [0 1]. Jika r < p', maka

perubahan kondisi baru menjadi kondisi sekarang diperbolehkan. Namun jika tidak, maka tidak proses tidak dikerjakan.

c. Perbaiki T sesuai dengan penjadwalan .

5. BEST_SO_FAR adalah jawaban yang dimaksudkan.

2.3.3 Karakteristik Masalah Kecerdasan Buatan

Tidak semua masalah dapat dipecahkan dengan teknik kecerdasan buatan. Pendekatan kecerdasan buatan dipakai untuk suatu masalah yang memiliki ledakan kombintorial ruang pencarian solusi. Gagasan utama dari pemecahan masalah secara cerdas adalah bahwa suatu sistem harus dapat mengkonstruksi solusi masalah secara selektif dan efisien dari ruang alternatif solusi. Suatu mesin cerdas harus mengimplikasikan kecerdasan manusia. Dengan kata lain program cerdas berkelakuan sebagaimana kecerdasan manusia.

Kecerdasan buatan dapat dikonsentrasikan pada hal-hal berikut :

1. Cocok untuk pemecahan masalah yang kualitatif daripada kuantitatif.
2. Pemecahan masalah dilakukan dengan penalaran simbolis dibanding kalkulasi.
3. Fokus masalahnya adalah untuk masalah yang tidak algoritmis. Teknik pemecahan masalahnya dilakukan dengan dengan organisasi dan eksploitasi pengetahuan untuk pencarian solusi dengan teknik heuristik.
4. Solusi masalahnya tidak bersifat eksak atau optimal, tetapi bersifat mencukupi.

2.3.4 Sudut Pandang Kecerdasan Buatan

Pada penjelasan kecerdasan buatan dapat dipandang dari berbagai sudut pandang, antara lain :

1. Sudut pandang kecerdasan

   Kecerdasan buatan akan membuat mesin menjadi ‘cerdas’ (mampu berbuat seperti apa yang dilakukan oleh manusia).

2. Sudut pandang penelitian

   Kecerdasan buatan adalah suatu studi bagaimana membuat agar komputer dapat melakukaan sesuatu sebaik yang dikerjakan oleh manusia.

   Domain yang sering dibahas oleh para peneliti meliputi :

   a. Mundane task
      1) Persepsi (*vision and speech*).
      2) Bahasa alami (*Understanding, generation and translation*)
      3) Pemikiran yang bersifat *commonsense*.
      4) Robot *control.*
   b. Formal *task*
      1) Permainan atau *games*.
      2) Matematika (geometri, logika, kalkulus integral, pembuktian).
   c. Expert task
      1) Analisis finansial.
      2) Analisis medikal.
      3) Analisis ilmu pengetahuan.
      4) Rekayasa (desain, pencarian kegagalan, perencanaan, manufaktur).

3. Sudut pandang bisnis

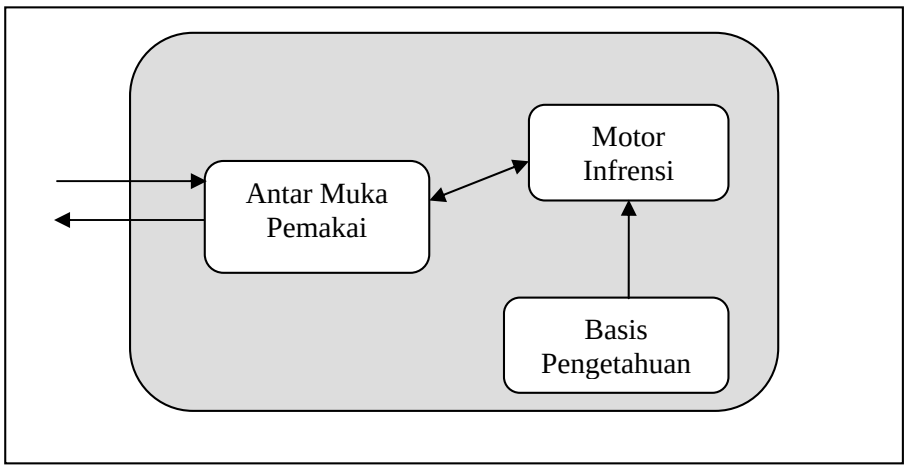
   Kecerdasan buatan adalah kumpulan peralatan yang sangat *powerful* dan metodologis dalam menyelesaikan masalah-masalah bisnis.

4. Sudut pandang pemrograman

   Kecerdasan buatan meliputi studi tentang pemrograman simbolik, penyelesaian masalah (*problem solving*) dan pencarian (*searching*).

   Untuk melakukan aplikasi kecerdasan buatan ada dua bagian utama yang sangat dibutuhkan, yaitu :

   a. Basis Pengetahuan (*Knowledge Base*), berisi fakta-fakta, teori, pemikiran dan hubungan antara dengan lainnya.
   b. Motor Infrensi (*Infrence Engine*), yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.
   c. Antar Muka, yaitu komponen yang menghubungkan motor infrensi dengan lingkungan eksternal.



Gambar 2.12. Arsitektur Mesin Cerdas

2.3.5 Kecerdasan Buatan dan Kecerdasan Alami

Jika dibandingkan dengan kecerdasan alami (kecerdasan yang dimiliki oleh manusia), kecerdasan buatan memiliki beberapa keuntungan secara komersial antara lain:

a. Kecerdasan buatan lebih bersifat permanen. Kecerdasan alami akan cepat mengalami perubahan. Hal ini dimungkinkan karena sifat manusia yang pelupa. Kecerdasan buatan tidak akan berubah sepanjang sistem komputer dan program tidak mengubahnya.
b. Kecerdasan buatan lebih mudah diduplikasi dan disebarkan. Mentransfer pengetahuan manusia dari satu orang ke orang lain membutuhkan proses yang sangat lama; dan juga suatu keahlian itu tidak akan pernah dapat diduplikasi dengan lengkap. Oleh karena itu, jika pengetahuan terletak pada sutu sistem komputer, pengetahuan tersebut dapat disalin dari komputer tersebut dan dapat dipindahkan dengan mudah ke komputer yang lain.
c. Kecerdasan buatan lebih murah dibanding dengan kecerdasan alami. Menyediakan layanan komputer akan lebih mudah dan lebih murah dibandingkan dengan harus mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
d. Kecerdasan buatan bersifat konsisten, hal ini disebabkan karena kecerdasan buatan adalah bagian dari teknologi komputer. Sedangkan kecerdasan alami akan senantiasa berubah-ubah.

e. Kecerdasan buatan dapat didokumentasi. Keputusan yang dibuat oleh komputer dapat didokumentasi dengan mudah dengan cara melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.

Sedangkan keuntungan dari kecerdasan alami adalah :

a. Kreatif. Kemampuan untuk menambah ataupun memenuhi pengetahuan itu sangat melekat pada jiwa manusia. Pada kecerdasan buatan, untuk menambah pengetahuan harus dilakukan melalui sistem yang dibangun.
b. Kecerdasan alami memungkinkan orang untuk menggunakan pengalaman secara langsung. Sedangkan pada kecerdasan buatan harus bekerja dengan *input-input* simbolik.

2.4 Struktur dan Strategi Pemahaman

Setelah dibahas tentang konsep kecerdasan buatan yang menjadi dasar pemahaman dalam pembuatan sistem. Pembahasan selanjutnya adalah tentang bagaimana struktur dan strategi infrensi untuk menemukan solusi permasalahan, juga diperkenalkan teori pencarian ruang keadaan.

2.4.1 Ruang Pencarian Keadaan (*State Space Search*)

Setelah dilakukan representasi pengetahuan pada domain masalah, maka perlu dibentuk model penalaran yang efisien untuk menghasilkan solusi. Masalah direpresentasikan menjadi pencarian ruang keadaan dengan memakai teori graf. Teori graf adalah model yang paling baik untuk penalaran dari struktur objek dan relasinya. simpul dalam graf berasosiasi dengan keadaan (*state*) dalam proses pemecahan masalah, sedang busur merepresentasikan transisi antar keadaan.

Transisi antar keadaan berkorespondensi dengan proses infrensi secara lojik dari suatu keadaan ke keadaan lainnya. Graf ruang keadaan haruslah berbentuk graf berarah asiklik.

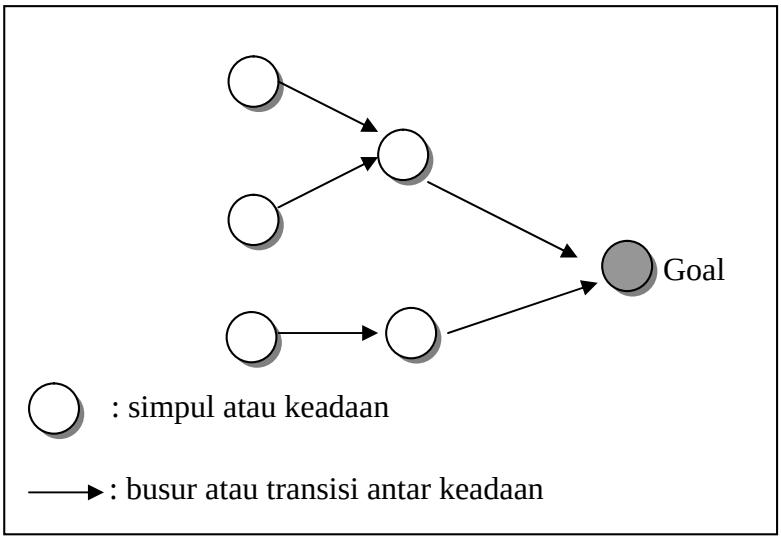
Secara formal ruang pencarian keadaan SP adalah 4 tupel SP = (N,E,S,G) dimana :

a) N adalah himpunan simpul atau keadaan dari graf yang berkorespondensi dengan suatu keadaan dalam proses pemecahan masalah.
b) E adalah himpunan busur antar busur yang berkorespondensi dengan tahapan dalam proses pemecahan masalah.
c) S adalah subset dari N yang tidak kosong yang berisi keadaan awal (inisial) masalah.
d) G adalah subset dari N yang tidak kosong yang berisi keadaan akhir (inisial) masalah. Himpunan keadaan dalam G dideskripsikan menurut kepemilikan suatu keadaan yang muncul dalam pencarian dan kepemilikan jalur yang dibangun dalam pencarian.

Jalur solusi adalah yang dilalui graf dari simpul S ke sebuah simpul di G. Arah pencarian dalam ruang keadaan dapat dilakukan dengan dua cara, yaitu :

1. **Forward Chaining *(data driven search*)**

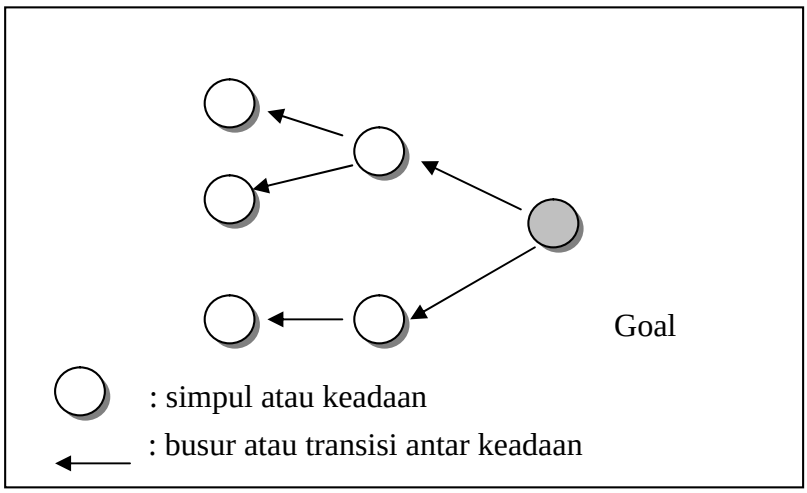
   Penalaran dimulai dari sejumlah fakta masalah dan mengaplikasikannya aturan dan pergerakan yang legal untuk memproduksi fakta-fakta baru yang menuju *goal*.



Gambar 2.13 *Forward Chaining*

2. Backward Chaining (*goal driven search*)

Memfokuskan pada *goal* masalah, kemudian menemukan aturan-aturan yang bisa memproduksi *goal* dan menelusuri aturan tersebut ke sub*goal*-sub*goal* yang dibawahnya menuju ke fakta-fakta dari masalah. Strategi ini efektif pada suatu masalah dimana solusi atau *goal* sudah diketahui.



Gambar 2.14 *Backward Chaining*

2.4.2 Strategi Kontrol dan Pencarian

Salah satu karakteristik program kecerdasan buatan dibanding pendekatan prosedural adalah adanya pemisahan antara pengetahuan dan bagian kontrol program. Kontrol program adalah bagian yang memutuskan aturan yang akan dipakai dalam proses pencarian solusi masalah. Proses pencarian solusi dalam ruang keadaan dilakukan secara heuristik, yaitu strategi atau prosedur yang digunakan untuk pengefisienan pencarian solusi. Teknik pencarian heuristik bekerja berdasar keadaan awal (*initial state*) dan keadaan tujuan (*goal state*). Teknik ini berperan dalam menentukan perggerakan dari keadaan awal ke keadaan selanjutnya mengarahkan ke keadaan yang merupakan keadaan tujuan. Dalam pencarian heuristik menjamin bahwa keadaan tujuan dapat ditemukan, jika memang keadaan tersebut terdapat dalam ruang pencarian. Terdapat beberapa alasan penggunaan teknik pencarian heuristik, yaitu :

1. Teknik pencarian heuristik cocok untuk masalah dimana terjadi ledakan kombinatorial (kondisi banyak alternatif) arah pencarian solusi. Metode ini mengaplikasikan strategi pencarian tertentu sehingga akan mereduksi jalur-jalur pencarian yang tidak penting.
2. Keuntungan pencarian heuristik adalah dalam hal efisien waktu, memori dan menghindari kondisi banyak alternatif.
3. Kemungkinan dicapai solusi besar. Solusi yang dihasilkan adalah solusi optimum, belum tentu solusi mutlak, keampuhannya bergantung pada teknik heuristik dalam mengeksploitasi basis pengetahuan.

Teknik heuristik ada yang bersifat umum (*General Purpose Heuristik*) dan ada yang bersifat pemakaian khusus (*Special Purpose Heuristik*) untuk suatu masalah tertentu. Pada teknik yang bersifat umum heuristik ruang pencariannya berupa pohon pencarian (*search tree*). Pada pohon pencarian, simpul merepresentasikan suatu keadaan tertentu dalam proses pemecahan masalah dan busur merepresentasikan transisi antar keadaan yang berkorespondensi logika infrensi atau satu langkah yang legal. Terdapat beberapa teknik pencarian heuristik yang bersifat umum seperti *Best-First Search*, *Simulated Annealing.* Pada tugas akhir ini hanya dibahas tentang *Simulated Annealing* dan algoritma Djikstra sebagai pembanding.

2.5 Masalah Ruang Keadaan dan Pencarian

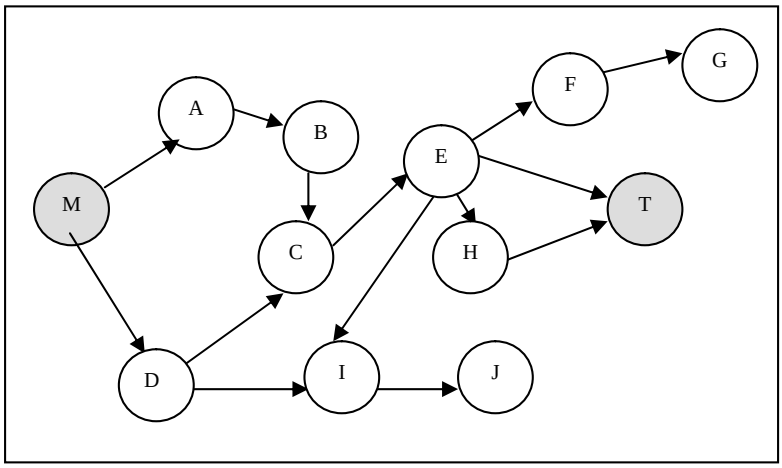
Sistem yang menggunakan kecerdasan buatan akan mencoba memeberikan output berupa solusi dari suatu masalah berdasarkan kumpulan pengetahuan yang ada.

Secara umum, untuk membangun suatu sistem yang mampu menyelesaikan masalah, perlu mempertimbangkan empat hal berikut:

1. Mendefinisikan masalah dengan tepat. Pendefinisian ini mencakup spesifikasi yang tepat mengenai keadaan awal dan solusi yang diharapkan.
2. Menganalisis masalah tersebut serta mencari beberapa teknik penyelesaian masalah yang sesuai.
3. Merepresentasikan pengetahuan yang perlu untuk menyelesaikan masalah tersebut.
4. Memilih teknik penyelesaian masalah yang terbaik.

2.5.1 Graph Keadaan

Graph terdiri-dari node-node yang menunjukan keadaan yaitu keadaan awal dan keadaan baru yang akan dicapai dengan menggunakan operator. Node-node dalam graph keadaan saling dihubungkan dengan menggunakan *arc* (busur) yang diberi arah untuk menunjukan suatu keadaan ke keadaan berikutnya.

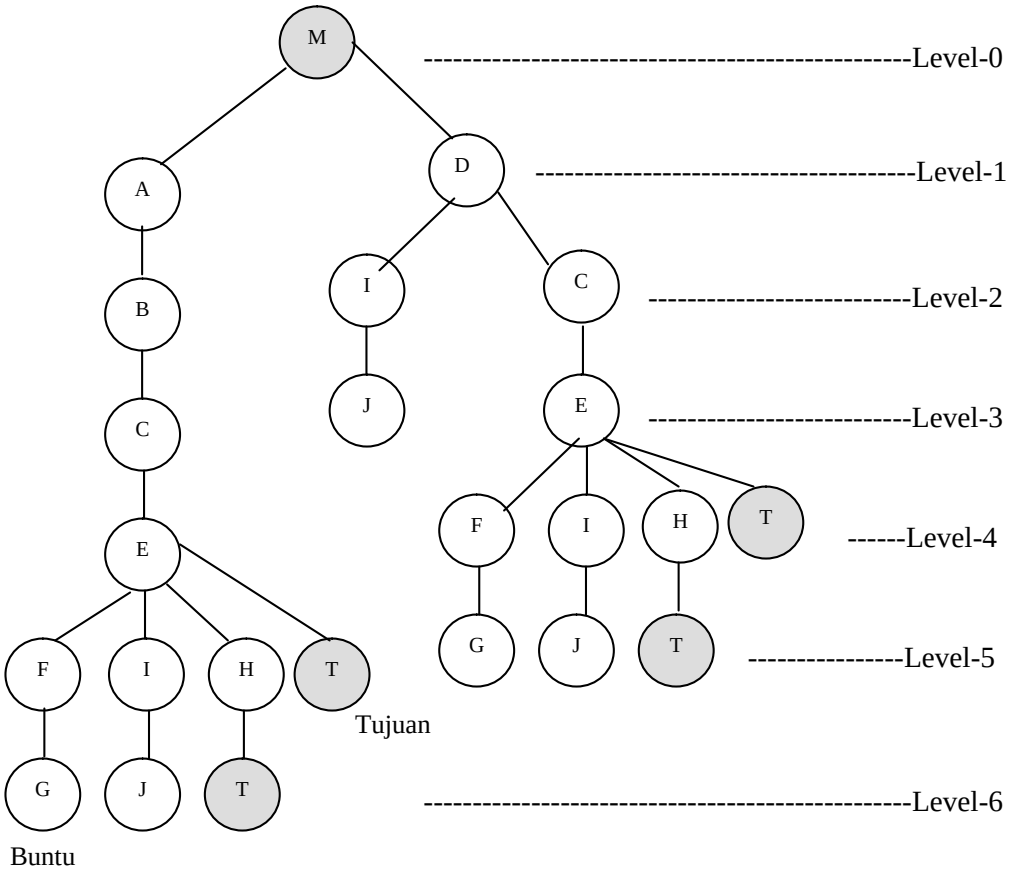


Gambar 2.15 Graph Keadaan

Pada gambar diatas node M menunjukan keadaan awal dan node T adalah tujuan. Bentuk graph seperti ini biasanya sangat sulit untuk direpresentasikan kedalam suatu program *interface*, karena sangat memungkinkan adanya siklus dalam graph tersebut. Sebagai contoh : D-C-E-I-D, node-node ini akan selalu berulang.

2.5.2 Pohon Pelacakan

Untuk menghindari kemungkinan adanya proses pelacakan suatu node secara berulang, maka digunakan struktur pohon.



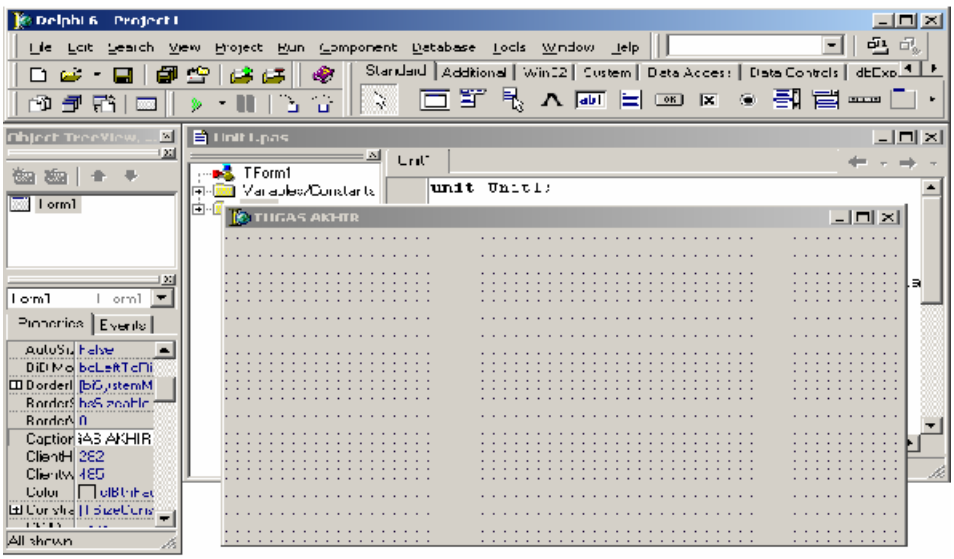
2.16 Struktur Pohon

Struktur pohon digunakan untuk menggambarkan keadaan secara hirarkis. Pohon juga terdiri-dari beberapa node. Node yang terletak pada level-0 disebut dengan nama akar (*root*). Node akar menunjukan keadaan awal yang biasanya merupakan topik atau obyek. Node akar ini terletak pada level ke nol. Node akar memiliki percabangan yang terdiri-atas beberapa node anak (*successor*) dan merupakan node-node perantara. Jika dilakukan pencarian mundur, maka dapat dikatakan bahwa node tersebut memiliki *predecessor*. Node-node yang tidak memiliki anak sering disebut dengan nama node”daun” yang menunjukan akhir dari suatu pencarian, dapat berupa tujuan yang diharapkan (*goal*) atau jalan buntu (*dead end*). Pada gambar diatas sudah tidak terlihat lagi adanya siklus, karena

setiap node tidak diperbolehkan memiliki cabang kembali ke node dengan level yang lebih rendah.

2.6 Perangkat Lunak Pendukung

Perangkat lunak yang digunakan pada penyusunan tugas akhir ini adalah program Delphi 6. Borlan Delphi 6 adalah bahasa pemrograman yang mempunyai cakupan kemampuan yang luas dan sangat canggih, juga program ini dapat menyediakan komponen-komponen dan bahasa pemrograman Object Pascal yang sudah sangat dikenal di kalangan pemrogram sehingga aplikasi yang dibuat dapat dengan mudah dipahami.



Gambar 2.17 Tampilan dasar delphi