Pengantar Kecerdasan Buatan (Artificial Intelligence / AI)

Pengertian AI :

1. Suatu cara yang sederhana untuk membuat komputer dapat "berpikir" secara inteligent.
2. Bagian dari ilmu komputer yang mempelajari perancangan sistem komputer yang inteligent, yaitu suatu sistem yang meperlihatkan karakteristik yang ada pada tingkah laku manusia, seperti mengerti suatu bahasa, mempelajari,mempertimbangkan dan memecahkan suatu masalah.
3. Suatu studi bagaimana membuat komputer dapat mengerjakan sesuatu, yang pada saat ini, orang dapat mengerjakan lebih baik
4. Bidang ilmu komputer yang memungkinkannya untuk memahami, bernalar dan bertindak.

Tujuan AI:

1. Untuk mengembangkan metode dan sistem untuk menyelesaikan masalah,masalah yang biasa diselesaikan melalui aktifivitas intelektual manusia, misalnya pengolahan citra,perencanaan, peramalan dan lain-lain, meningkatkan kinerja sistem informasi yang berbasis komputer.
2. Untuk meningkatkan pengertian/pemahaman kita pada bagaimana otak manusia bekerja.

Arah AI:

1. Mengembangkan metode dan sistem untuk menyelesaikan masalah AI tanpa mengikuti cara manusia menyelesaikannya (sistem pakar / expert systems).
2. Mengembangkan metode dan sistem untuk menyelesaikan masalah AI melalui pemodelan cara berpikirnya manusia, atau cara bekerjanya otak manusia (neural networks).

Paradigma AI:

Symbolic or sub-symbolic (connectionist).

Bidang-bidang Aplikasi AI

Menurut Elaine Rich : Masalah pada AI *(Task Domain) :*

1. Task Keduniaan (*Mundane Task*)
   a. *Perception: Vision, Speech Recognition.*
   b. *Natural Language: Understanding, Generation, Translation.*
   c. *Commonsense Reasoning*
   d. *Robot Control*

2. Task Formal (Formal Task)
   a. Games (Chess, Backgamon, checkers,Go)
   b. Mathematics (Geometry, Logic, Integral Calculus, Proving properties of programs)

3. Task Ahli (Expert Tasks)
   a. Engineering (Design,Fault Finding, Manufacturing Planning)
   b. *Scientific Analysis, Medical Diagnosis, Financial Analysis*

Asumsi Dasar AI

Physical Symbol System terdiri dari himpunan entitas yang dinamakan simbol, berpola fisik yang dapat menjadi komponen dari entitas tipe lain yang dinamakan Ekspresi (struktur simbol)

Struktur simbol terdiri dari sejumlah *instant* (*Token*) dari simbol-simbol yang berhubungan pada beberapa cara fisik. Selain struktur tersebut, sistem juga berisi koleksi proses-proses yang beroperasi pada ekspresi, untuk menghasilkan ekspresi lain : proses pembuatan (*create*), modifikasi, reproduksi, dan penghancuran (destruksi).

Jadi PSS adalah mesin yang memproduksi suatu koleksi penyusunan struktur simbol. Sistem seperti itu terdapat dalam suatu objek dunia yang lebih luas dari hanya ekspresi simbolik itu sendiri.

Perbedaan antara Pemrograman AI dan Konvensional

AI	**Komputasi Konvensional**
Representasi dan Manipulasi simbol	Algoritama
Memberitahu komputer tentang suatu masalah	Memerintah komputer untuk menyelesaikan masalah
Komputer diberi pengetahuan dan kemampuan inferensi	Memberi data kepada komputer dan program

Pemrograman AI :

1. Bila terjadi perubahan dalam program, maka tidak mengganggu seluruh “Facts” yang tersimpan dalam “Otak” (layaknya pikiran manusia/seperti informasi yang terdapat pada pikiran manusia)
2. Independen
3. Dapat Dimodifikasi tanpa mempengaruhi struktur kesluruhan program
4. Fleksibel → efisien dan mudah untuk dimengerti

Penyelesaian Masalah berdasarkan teknik AI

Empat hal untuk membangun sistem atau memecahkan masalah tertentu :

1. Definisikan masalah dengan jelas
2. Analisis masalah
3. Kumpulkan dan representasikan knowledge
4. Pilih teknik pemecah masalah terbaik dan gunakan untuk masalah tertentu

Mendefinisikan Masalah sebagai "State Space Search" (SSS), misalnya permainan catur , maka SSS nya adalah : Menspesifikasikan posisi awal dari papan catur, peraturan (rules) yang mendefinisikan langkah-langkah yang legal dan posisi papan yang merepresentasikan pemenang dari satu sisi atau sisi lainnya. Tujuan (*Goal*) dari permainan adalah : memenangkan permainan.

Pendefinisian Masalah Sebagai Pencarian Ruang Keadaan

Masalah utama dalam membangun sistem berbasis AI adalah bagaimana mengkonversikan situasi yang diberikan ke dalam situasi lain yang diinginkan
menggunakan sekumpulan operasi tertentu.

A Water Jug Problem

Anda diberi dua buah gelas, yang satu ukuran 4 galon dan yang lain 3 galon. Kedua gelas tidak memiliki skala ukuran. Terdapat pompa yang dapat digunakan untuk mengisi gelas dengan air. Bagaimana anda mendapatkan tepat 2 galon air di dalam gelas 4 ukuran galon?

Ruang masalah untuk masalah di atas dapat digambarkan sebagai himpunan pasangan bilangan bulat (x,y) yang terurut, sedemikian hingga x = 0, 1, 2, 3, atau 4 dan y = 0, 1, 2, atau 3; x menyatakan jumlah air dalam gelas ukuran 4 galon, dan y menyatakan jumlah air dalam gelas ukuran 3 galon. Keadaan mula-mula adalah (0,0). *State* tujuan adalah (2,n) untuk setiap nilai n.

Operator-opeartor (aturan produksi) yang digunakan untuk memecahkan masalah terlihat pada gambar 2.2.

1.	(x,y) If x < 4	→	(4,y)	Isi penuh gelas 4 galon
2.	(x,y) If y < 3	→	(x,3)	Isi penuh gelas 3 galon
3.	(x,y) If x > 0	→	(x-d,y)	Buang sebagian air dari gelas 4 galon
4.	(x,y) If y > 0	→	(x,y-d)	Buang sebagian air dari galon ukuran 3 galon

5.	(x,y) If x > 0	→	(0,y)	Kosongkan gelas 4 galon
6.	(x,y) If y > 0	→	(x,0)	Kosongkan gelas 3 galon
7.	(x,y) If x+y ≥4 and y > 0	→	(4,y-(4-x))	Tuangkan air dari gelas 3 galon ke gelas 4 galon sampai gelas 4 galon penuh
8.	(x,y) If x+y ≥3 and x > 0	→	(x-(3-y),3)	Tuangkan air dari gelas 4 galon ke gelas 3 galon sampai gelas 3 galon penuh
9.	(x,y) If x+y ≤4 and y > 0	→	(x+y,0)	Tuangkan seluruh air dari gelas 3 galon ke gelas 4 galon
10.	(x,y) If x+y ≤3 and x > 0	→	(0,x+y)	Tuangkan seluruh air dari gelas 4 galon ke gelas 3 galon
11.	(0,2)	→	(2,0)	Tuangkan 2 galon air dari gelas 3 galon ke gelas 4 galon
12.	(2,y)	→	(0,y)	Buang 2 galon dalam gelas 4 galon sampai habis.

Gambar Aturan produksi untuk *Water Jug Problem*.

Jumlah galon dalam gelas 4 galon	**Jumlah galon dalam gelas 3 galon**	**Aturan yang dilakukan**
0	0	-
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 atau 12
2	0	9 atau 11

Gambar Suatu solusi untuk *Water Jug Problem*.

Karakteristik Masalah Dalam AI :

1. Apakah masalahnya dapat didekomposisi menjadi himpunan sub masalah yang (hampir) independen lebih kecil atau lebih mudah ?
2. Dapatkah langkah penyelesaian diacuhkan paling tidak dibatalkan ketika dapat dibuktikan hal tersebut tidak bijaksana ?
3. Apakah universe masalahnya dapat diprediksi ?
4. Apakah solusi yang baik dari masalah tertentu jelas tanpa membandingkan dengan seluruh solusi lain yang mungkin ?
5. Apakah solusi yang diinginkan sebuah keadaaan dari dunia atau sebuah jalur dari keadaan ?
6. Apa peran dari pengetahuan ?
7. Apakah pekerjaan memerlukan interakasi dengan manusia ?

Sistem Produksi

Sistem produksi terdiri dari:

1. **Himpunan aturan**, masing-masing terdiri dari sisi kiri (pola) yang menentukan kemampuan aplikasi dari aturan tersebut dan sisi kanan yang menggambarkan operasi yang dilalukan jika aturan dilaksanakan.
2. Satu atau lebih **pengetahuan** atau basis data yang berisi informasi apapun untuk tugas tertentu. Beberapa bagian basis data bisa permanen, dan bagian yang lain bisa hanya merupakan solusi untuk masalah saat ini. Informasi dalam basis data ini disusun secara tepat.
3. **Strategi kontrol** yang menspesifikasikan urutan dimana aturan akan dibandingkan dengan basis data dan menspesifikasikan cara pemecahan masalah yang timbul ketika beberapa aturan sesuai sekaligus pada waktu yang sama.
4. **A rule applier** (*pengaplikasi aturan*).

Strategi Kontrol

Syarat-syarat strategi kontrol:

1. ***cause motion.*** Perhatikan kembali *water jug problem*. Jika kita mengimplementasikan strategi kontrol sederhana dengan selalu memilih aturan pertama pada daftar 12 aturan yang telah dibuat, maka kita tidak akan pernah memecahkan masalah. Strategi kontrol yang tidak menyebabkan ***motion*** tidak akan pernah mencapai solusi.
2. ***Systematic.*** Strategi kontrol sederhana yang lain untuk *water jug problem*: pada setiap siklus, pilih secara random aturan-aturan yang dapat diaplikasikan. Strategi ini lebih baik dari yang pertama, karena menyebabkan *motion*. Pada akhirnya strategi tersebut akan mencapai solusi. Tetapi mungkin kita akan mengunjungi beberapa *state* yang sama selama proses tersebut dan mungkin menggunakan lebih banyak langkah dari jumlah langkah yang diperlukan. Hal ini disebabkan strategi kontrol tersebut tidak sistematik. Beberapa strategi kontrol yang sistematik telah diusulkan, yang biasa disebut sebagai metoda-metoda dalam teknik *searching*. Di bab ini, akan dibahas enam metoda, yaitu *Breadth First Search*, *Uniform Cost Search*, *Depth First Search*, *Depth-Limited Search*, *Iterative-Deepening Depth-First Search*, dan *Bi-*

directional search. Masing-masing metoda tersebut mempunyai karakteristik yang berbeda.

Strategi Pencarian

Terdapat empat kriteria dalam strategi pencarian, yaitu:

1. **Completeness:** Apakah strategi tersebut menjamin penemuan solusi jika solusinya memang ada?
2. **Time complexity:** Berapa lama waktu yang diperlukan?
3. **Space complexity:** Berapa banyak memori yang diperlukan?
4. **Optimality:** Apakah strategi tersebut menemukan solusi yang paling baik jika terdapat beberapa solusi berbeda pada permasalahan yang ada?

Depth-First Search (DFS)

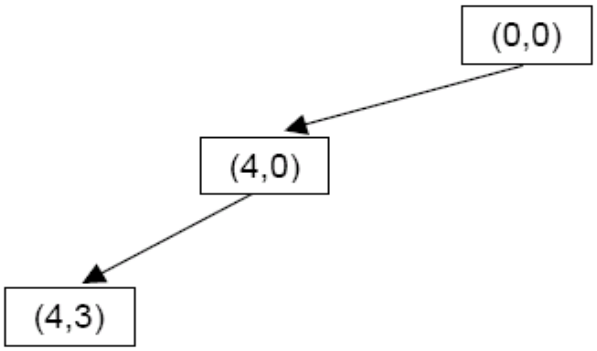
Pencarian dilakukan pada satu node dalam setiap level dari yang paling kiri. Jika pada level yang paling dalam, solusi belum ditemukan, maka pencarian dilanjutkan pada node sebelah kanan. Node yang kiri dapat dihapus dari memori. Jika pada level yang paling dalam tidak ditemukan solusi, maka pencarian dilanjutkan pada level sebelumnya. Demikian seterusnya sampai ditemukan solusi. Jika solusi ditemukan maka tidak diperlukan proses *backtracking* (penelusuran balik untuk mendapatkan jalur yang dinginkan).

Kelebihan DFS adalah:

1. Pemakain memori hanya sedikit, berbeda jauh dengan BFS yang harus menyimpan semua node yang pernah dibangkitkan.
2. Jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka DFS akan menemukannya secara cepat.

Kelemahan DFS adalah:

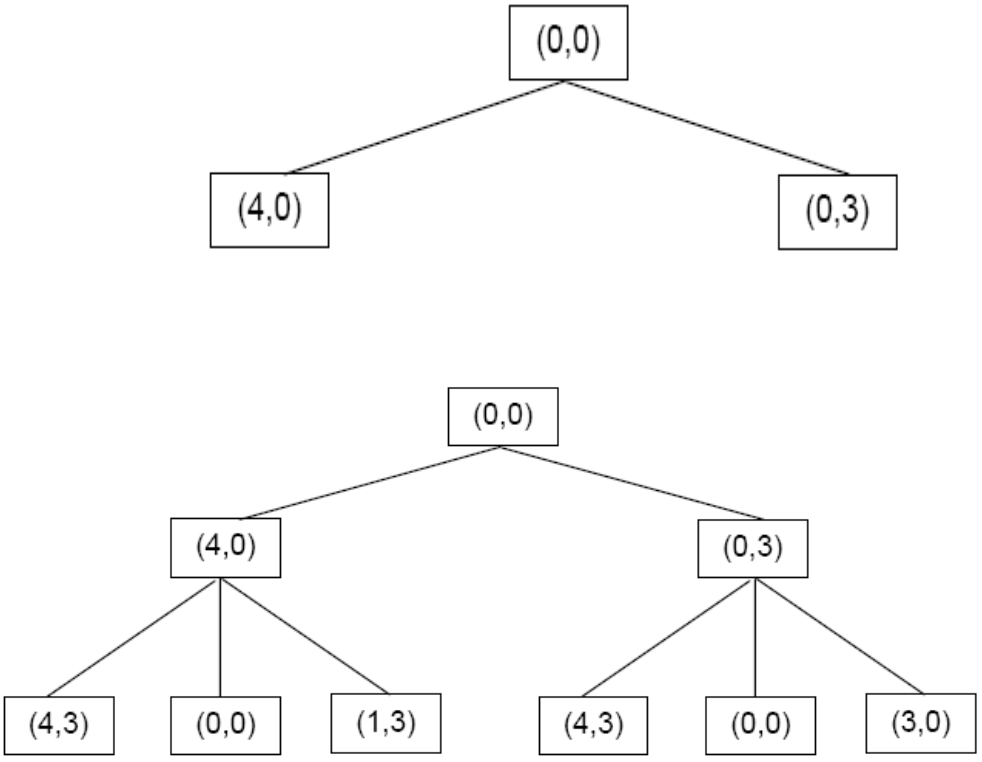
1. Jika pohon yang dibangkitkan mempunyai level yang dalam (tak terhingga), maka tidak ada jaminan untuk menemukan solusi (**Tidak *Complete***).
2. Jika terdapat lebih dari satu solusi yang sama tetapi berada pada level yang berbeda, maka pada DFS tidak ada jaminan untuk menemukan solusi yang paling baik (**Tidak *Optimal***).



Gambar Penelusuran *Depth First Search* untuk *Water Jug Problem*.

***Breadth-First Search* (BFS)**

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. Demikian seterusnya sampai ditemukan solusi. Dengan strategi ini, maka dapat dijamin bahwa solusi yang ditemukan adalah yang paling baik (***Optimal***). Tetapi BFS harus menyimpan semua node yang pernah dibangkitkan. Hal ini harus dilakukan untuk penelusuran balik jika solusi sudah ditemukan. Gambar 2.4 mengilustrasikan pembangkitan pohon BFS untuk masalah *Water Jug*. Pembangkitan suksesor dari suatu node bergantung pada urutan dari Aturan Produksi yang dibuat (lihat gambar 2.3). Jika urutan dari aturan 4 ditukar dengan aturan 5, maka pohon BFS yang dibangkitkan juga akan berubah.



Gambar Pohon *Breadth First Search* untuk *Water Jug Problem*.

Berikut ini membahas metoda-metode yang terdapat dalam teknik pencarian yang berdasarkan pada panduan (*Heuristic Search*), yaitu *Generate and Test, Simple Hill Climbing, Steepest-Ascent Hill Climbing, Simulated Annealing, Best First Search,Greedy Search, A Star (A*), Problem Reduction, Constraint Satisfaction,* dan *Means-Ends Analysis*.

Generate-and-Test

Metode *Generate*-and-*Test* adalah metode yang paling sederhana dalam pencarian *heuristic*. Jika pembangkitan *possible solution* dikerjakan secara sistematis, maka prosedur akan mencari solusinya, jika ada. Tetapi jika ruang masalahnya sangat luas, mungkin memerlukan waktu yang sangat lama.

Algoritma *Generate*-and-*Test* adalah prosedur DFS karena solusi harus dibangkitkan secara lengkap sebelum dilakukan *test*. Algoritma ini berbentuk sistematis, pencarian sederhana yang mendalam dari ruang permasalahan. *Generate* & *test* juga dapat dilakukan dengan pembangkitan solusi secara acak, tetapi tidak ada jaminan solusinya akan ditemukan.

Algorithm: *Generate*-and-*Test*

1. Generate a possible solution. For some problems, this means generating a particular point in the problem space. For others, it means generating a path from a start state.
2. Test to see if this is actually a solution by comparing the chosen point or endpoint of the chosen path to the set of acceptable goal states.
3. If a solution has been found, quit. Otherwise, return to step 1.

Contoh kasus:

Untuk permasalahan sederhana maka tehnik *generate* & *test* adalah tehnik yang layak. Sebagai contoh, pada teka-teki yang terdiri dari empat kubus segi enam, dengan masingmasing sisi dari setiap kubus dicat dengan 4 warna. Solusi dari teka-teki terdiri dari susunan kubus dalam beberapa baris yang semuanya empat sisi dari satu blok baris yang menunjukkan nasing-masing warna. Masalah ini dapat diselesaikan dengan manusia dalam beberapa menit secara sistematis dan lengkap dengan mencoba semua kemungkinan. Ini bisa diselesaikan dengan lebih cepat menggunakan prosedur *generate* & *test*. Pandangan sekilas pada empat blok yang tampak bahwa masih ada lagi, katakanlah bagian merah dari warna-warna lain yang ada. Sehingga ketika menempatkan blok dengan beberapa bagian merah, ini akan menjadi ide yang baik untuk digunakan jika sebagian darinya sebisa mungkin dibagian luar. Sebagian yang lain sebisa mungkin harus ditempatkan pada blok berikutnya. Menggunakan aturan ini, banyak konfigurasi diperlukan tanpa di-*explore* dan sebuah solusi dapat ditemukan lebih cepat.