NIM : 09021281722032

SOLVING PROBLEMS BY SEARCHING

Intelligent agents seharusnya dapat memaksimalkan ukuran kinerja, hal ini dapat dicapai dengan menetukan tujuan dan sasaran yang akan dicapai. Tujuan membantu mengatur perilaku dengan membatasi obyektif yang berusaha dicapai oleh agen dan karenanya ada tindakan yang perlu dipertimbangkan. Perumusan tujuan berdasarkan situasi saat ini dan ukuran kinerja agen, adalah langkah pertama dalam proses pemecahan masalah. Secara umum, agen dengan beberapa pilihan langsung dengan nilai yang tidak diketahui dapat memutuskan apa yang harus dilakukan dengan mengecek tindakan di masa depan terlebih dahulu yang akan mengarah ke keadaan nilai yang tidak diketahui.

Sebuah masalah dapat didefinisikan dengan lima komponen :

- Keadaan awal agen memulai.
- Deskripsi tindakan yang tersedia untuk agen. Diberikan status tertentu untuk *s*, mengembalikan serangkaian aksi yang dapat dieksekusi *s*.
- Deskripsi tentang yang dilakukan setiap tindakan; disebut model transisi, yang ditentukan oleh fungsi hasil (*s*,*a*) yang mengembalikan keadaan yang dihasilkan dari melakukan tindakan *a* dalam keadaan *s*.
- Tes tujuan, bertujuan menentukan apakah keadaan yang diberikan adalah keadaan tujuan.
- Fungsi biaya jalur, menetapkan biaya numerik untuk setiap jalur.

Komponen ini merupakan model deskripsi matematis abstrak, pada kenyataannya banyak hal yang diabaikan deskripsi keadaan karena tidak relevan dengan masalah yang ingin diselesaikan. Proses menghapus detail dari representasi tesebut disebut abstraksi. Pilihan abstraksi yang baik dengan melibatkan penghapusan sedetail mungkin sambil mempertahankan validitas dan memastikan bahwa tindakan abstrak mudah dilakukan.

Urutan tindakan yang mungkin dimulai dari keadaan awal membentuk search tree dengan posisi keadaan awal pada root; ranting-rantingnya adalah tindakan dan simpul-simpulnya (node) berhubungan dengan keadaan yang merupakan keadaan masalah. Node dapat diperluas dan proses perluasan node berlanjut sampai solusi ditemukan atau tidak ada lagi keadaan untuk diperluas. Pada pohon pencarian ini terdapat keadaan yang berulang (repeated state) disebabkan adanya loopy path. Loopy path adalah kasus khusus dari konsep redundant path, yang muncul setiap ada lebih dari satu cara untuk pergi dari satu state menuju state lainnya. Untuk mengatasinya adalah dengan mengingat di mana sedang berada. Untuk melakukan ini, ditambahkan algoritma tree search dengan struktur data yang disebut exploration set (juga dikenal sebagai daftar tertutup), yang mengingat setiap node yang diperluas. Algoritma baru ini disebut dengan Graph Search.

Algoritma pencarian membutuhkan struktur data untuk melacak pohon pencarian yang sedang dibangun. Untuk setiap node *n* pada pohon, terdapat struktur yang memiliki empat komponen :

- *n*.state : state pada ruang state yang berhubungan dengan node.
- *n*.parent : node pada pohon pencarian yang menghasilkan node lain.
- *n*.action : aksi yang dimiliki oleh node parent untuk menghasilkan node.
- *n*.path-cost : biaya, yang dinotasikan dengan *g(n)*, jalur dari state awal menuju node seperti yang ditunjuk oleh pointer parent.

Setelah memiliki node, maka perlu tempat untuk meletakkannya sehingga algoritma pencarian dapat dengan mudah memilih node berikutnya untuk memperluas sesuai dengan strategi yang digunakan. Struktur data yang digunakan adalah queue. Operasi yang terdapat pada queue adalah :

- empty (queue) mengembalikan nilai true hanya jika tidak ada lagi elemen di dalam queue.
- Pop (queue) menghapus elemen pertama pada queue dan mengembalikannya.
- Insert (element, queue) memasukkan elemen dan mengembalikan hasil queue.

Sebelum membuat algoritma, kita perlu mempertimbangkan kriteria yang dapat digunakan untuk memilih. Mengevaluasi kinerja suatu algoritma dapat dilakukan dengan empat cara:

- Completeness : Apakah algoritma terjamin untuk menemukan solusi ketika ada?
- Optimality : apakah strategi yang digunakan dapat menemukan solusi yang optimal?
- Time complexity : berapa lama waktu yang dibutuhkan untuk menemukan solusi?
- Space complexity : berapa banyak memori yang dibutuhkan untuk melakukan pencarian?

Kompleksitas waktu dan ruang selalu dipertimbangkan sehubungan dengan beberapa ukuran kesulitan masalah. Dalam AI, graph sering direpresentasikan secara implisit oleh state awal, tindakan, dan model transisi dan frekuensi tak terbatas. Untuk alasan ini, kompleksitas dinyatakan dalam tiga kuantitas : *b*, faktor percabangan atau jumlah maksimum penerus dari setiap node; *d*, kedalaman node tujuan yang paling dangkal; dan *m*, panjang maksimum setiap jalur pada ruang state. Untuk menilai keefektifan sebuah algoritma pencarian, dapat menggunakan total cost yang merupakan gabungan dari search cost dan path cost untuk menemukan solusi.