Penerapan Graf dalam Game dengan Kecerdasan Buatan

Azwar Tamim
Jurusan Teknik Informatika ITB, Bandung 40135, email: if16109@students.if.itb.ac.id

Abstract – *Makalah ini membahas penggunaan graf di dalam pengembangan game dengan kecerdasan buatan. Pembahasan dimulai dengan penggunaan graf untuk navigasi agent atau pathfinding. Lalu dilanjutkan dengan pembahasan penggunaan graf untuk game yang bertipe resource management, yaitu dependency graf. Kemudian terakhir ditutup dengan pembahasan penggunaan state graf untuk pemecahan masalah.*

Kata Kunci: *navigation, pathfinding, agent, state, dependency.*

1. PENDAHULUAN

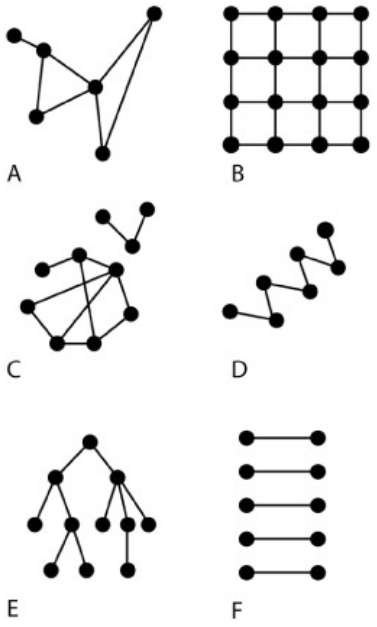
1. 1 Pengenalan Graf

Graf secara formal didefinisikan sebagai himpunan pasangan (V, E). Dituliskan sebagai:

G = {V, E}

yang dalam hal ini V adalah himpunan tidak-kosong dari simpul-simpul (vertices atau node) dan E adalah himpunan sisi (edges atau arcs) yang menghubungkan sepasang simpul.[1]

Gambar di bawah ini memperlihatkan berbagai macam contoh graf.



Gambar 1: Contoh *graf*

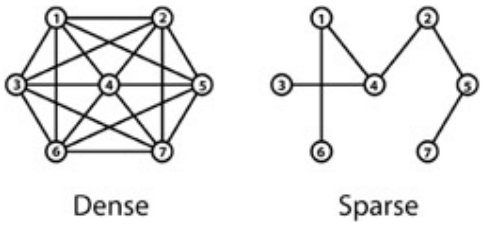
Setiap node dalam graf biasanya diberikan sebuah nama, dapat berupa huruf atau angka. Graf dapat memiliki sisi yang memiliki bobot.

Pohon

Pohon merupakan sebuah graf yang tidak memiliki sirkuit di dalamnya, artinya ia tidak memiliki koneksi yang dapat membuat lintasan kembali ke node awalnya. Struktur data pohon banyak digunakan dalam game dengan kecerdasan buatan untuk pengambilan keputusan.

Kepadatan Graf

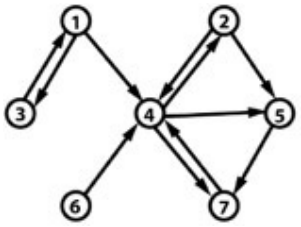
Ratio dari jumlah sisi berbanding jumlah node disebut sebagai kepadatan graf. Kepadatan graf mengindikasikan apakah suatu graf adalah sparse atau dense. Sparse graf memiliki lebih sedikit koneksi dibanding dense graf. Di bawah ini contoh kedua graf tersebut.



Gambar 2: Contoh *graf dense dan sparse*

Untuk mengurangi kompleksitas dan untuk meminimumkan penggunaan CPU dan memori, maka adalah lebih baik untuk memilih sparse graf jika memungkinkan, misalnya untuk mendesain graf untuk perencanaan lintasan dalam game.

Digraf

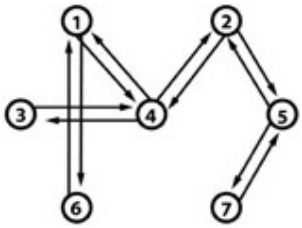


Gambar 3: Contoh digraf sederhana

Digraf adalah graf yang memiliki arah. Salah satu contoh penggunaan graf berarah ini dalam game misalnya dalam kasus mencari cost perjalanan

melalui gunung. Cost untuk menaiki gunung dan turun dari gunung dapat berbeda, sehingga diperlukan graf dengan representasi arah ini.

Di dalam implementasi ke dalam game, lebih dianjurkan untuk menggunakan representasi digraf yang setiap koneksinya memiliki dua arah seperti pada gambar berikut.



Gambar 4: Contoh *digraf untuk game*

Mengapa ini penting? Karena kedua tipe graf (berarah dan tidak berarah) dapat direpresentasikan dengan struktur data yang sama, sehingga programmer dapat lebih nyaman menggunakannya.

1. 2 Graf dalam Game dengan Kecerdasan Buatan

Dalam pengembangan game, banyak ditemukan penggunaan teori-teori di bidang ilmu komputer seperti logika, matriks, ilmu peluang, dan juga graf. Untuk pengembangan game dengan kecerdasan buatan, graf memainkan banyak peranan penting, terutama untuk penggunaan yang paling populer, yaitu navigasi atau *pathfinding*. Selain itu, dalam game yang memilki *agent* (*agent* adalah subjek/objek di dalam game yang memiliki kecerdasan buatan), ditemukan beberapa penggunaan graf, yaitu *dependency* graf dan *state* graf. *Dependency* graf banyak digunakan untuk game yang bertipe *resource management* untuk menggambarkan *dependency* antara berbagai macam bangunan, material, unit, dan teknologi yang tersedia untuk pemain. *State* graf digunakan untuk merepresentasikan setiap kemungkinan *state* yang mungkin dicapai sistem dan transisi di antara *state-state* tersebut. Himpunan semua *state* ini biasa disebut sebagai *state space*.

1.3 Representasi Graf

Representasi graf ada bermacam-macam, di sini akan diberikan dua contoh representasi graf dalam program, yaitu dengan matriks dan adjacency list.[1]

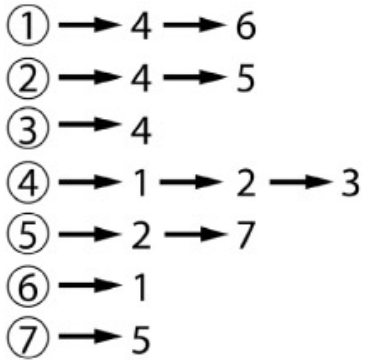
Represntasi graf dengan matriks digambarkan seperti pada gambar berikut.

	①	②	③	④	⑤	⑥	⑦
①	0	0	0	1	0	1	0
②	0	0	0	1	1	0	0
③	0	0	0	1	0	0	0
④	1	1	1	0	0	0	0
⑤	0	1	0	0	0	0	1
⑥	1	0	0	0	0	0	0
⑦	0	0	0	0	1	0	0

Gambar 5: Representasi graf dengan matriks

Setiap angka 1 merepresentasikan adanya koneksi antara node-node yang terdapat pada graf. Setiap angka 0 merepresentasikan tidak adanya koneksi antar kedua node dalam graf.

Representasi dalam adjacency list digambarkan pada gambar berikut.

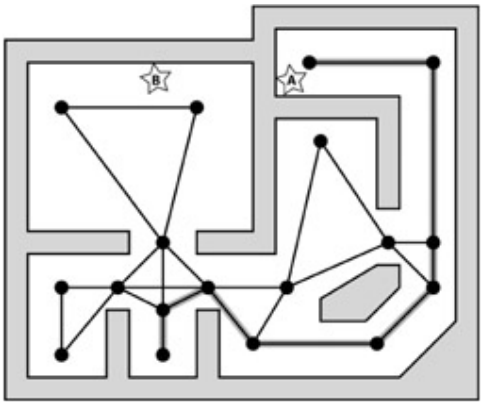


Gambar 6: Representasi graf dengan adjacency list

Representasi ini memperlihatkan setiap node yang terkoneksi dengan node-node lainnya. Misalnya, node 1 terkoneksi dengan node 4 dan node 6, node 2 terkoneksi dengan node 4 dan node 5, dan seterusnya.

2. GRAF UNTUK NAVIGASI ATAU PATHFINDING

Navigation graph adalah abstraksi dari semua lokasi di dalam lingkungan game yang mungkin dikunjungi oleh *agent* dan juga abstraksi semua koneksi antara lokasi-lokasi tersebut. Di dalam implementasinya, *navigation graph* berupa struktur data yang dapat menggambarkan semua kemungkinan jalan (path) dalam lingkungan game. *Navigation graph* ini sangat membantu agent dalam memilih lintasan dari suatu lokasi ke lokasi yang lain.[2]

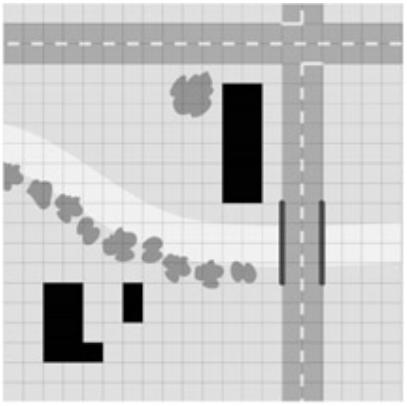


Gambar 7: Contoh *navigation graph*

Setiap node dalam navigation graph biasanya merepresentasikan posisi suatu area penting atau posisi objek dalam game. Setiap sisinya merepresentasikan koneksi/hubungan antara node-node tersebut. Lebih lanjut, setiap sisi dapat memiliki suatu nilai/cost, misalnya nilai jarak antara node-node yang ia koneksikan. Graf tipe ini biasa disebut *Euclidean graph*.

Game agent dapat menelusuri jalan/lintasan di dalam graf ini untuk menemukan lokasi yang dituju. Ia dapat menggunakan graf ini untuk mengkalkulasi lintasan terbaik yang dapat ia pilih.

Tipe game yang lain mungkin menggunakan tipe layout yang berbeda dengan graf sebelumnya. Misalnya, untuk game yang bertipe *real time strategy* (RTS) atau *role-playing game* (RPG), layout-nya dapat terdiri/dibangun dari sel, dimana setiap sel merepresentasikan daerah-daerah yang berbeda, seperti rumput, jalan, atau lumpur. Oleh karena itu, untuk mencari jalan ke suatu lokasi di dalam graf, agent akan menandakan lokasi tempat dia berada saat ini sebagai titik pusat dan akan melakukan algoritma pencarian untuk menemukan jalan terbaik menuju lokasi yang dituju. Pendekatan ini membuat agent mampu mengkalkulasi lintasan untuk menghindari air, memilih berjalan di atas jalan ketimbang lumpur, atau memilih rute berkelok mengelilingi pegunungan untuk melewatinya.

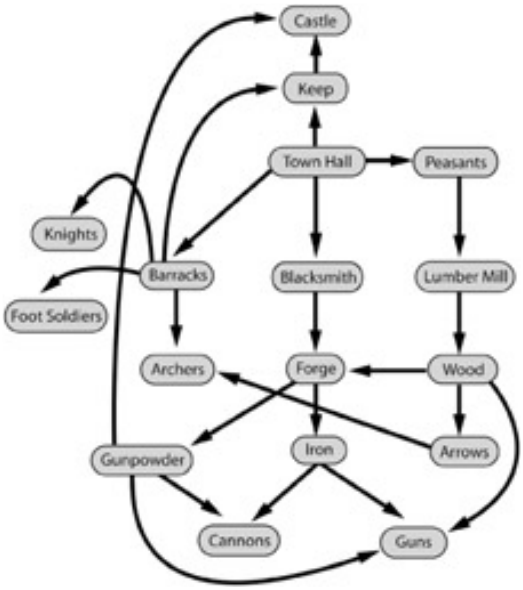


Gambar 8: Contoh graf dengan representasi sel

Representasi dengan sel ini memiliki kelemahan untuk game yang bertipe RTS/RPG karena ukuran graf-nya yang bisa sangat besar, waktu pencarian yang relatif lama, dan akan memakan sejumlah besar ruang di memori. Untuk itu diperlukan beberapa teknik untuk menghindari hal ini – tidak akan dibahas dalam makalah ini.

3. DEPENDENCY GRAPH

Dependency graph digunakan di dalam game yang bertipe resource management untuk menggambarkan kebergantungan antara berbagai macam bangunan, material, unit, dan teknologi yang tersedia untuk pemain. Gambar di bawah ini memperlihatkan sebuah dependency graph sederhana.[2]



Gambar 9: Contoh dependency graph sederhana

Jenis graf ini mempermudah pengembang/sistem untuk melihat persyaratan apa saja yang harus dipenuhi untuk membuat suatu jenis resource. Dependency graph sangat berguna untuk mendesain kecerdasan buatan untuk game yang bertipe seperti ini karena kecerdasan buatan dapat menggunakannya untuk memilih strategi yang cocok, memprediksi masa depan status lawan, dan menempatkan resource dengan efektif.

Di bawah ini beberapa contoh kasus berdasarkan graph yang baru saja diperlihatkan:

1. Jika AI (kecerdasan buatan) akan menyiapkan pertempuran dan ia mengetahui bahwa tim pemanah merupakan solusi yang menguntungkan maka ia akan memeriksa dependency graph untuk menyimpulkan bahwa sebelum membuat pemanah, ia harus sudha membuat barak-nya terlebih dahulu dan harus sudah memiliki teknologi panahan. Dia juga mengetahui bahwa sebelum memproduksi panah, ia sudah harus memiliki *lumber mill* untuk memproduksi kayu. Oleh karena itu, jika AI telah memiliki lumber mill, ia dapat menempatkan resource untuk memproduksi barak. Jika AI tidak memiliki barak dan lumber mill, maka ia harus memeriksa technology graph untuk lebih jauh untuk menentukan bahwa lebih menguntungkan untuk membangun barak daripada lumber mill dalam kondisi ini. Mengapa? Karena barak merupakan prasyarat untuk membangun tiga unit tempur, sedangkan lumber mill merupakan prasyarat untuk memproduksi kayu. Jika AI sudah mengetahui bahwa pertempuran sudah dekat, maka ia harus menyadari bahwa ia lebih baik membangun tiga unit tempur daripada membangun lumber mill.

2. Jika pasukan infantry musuh yang membawa senjata api datang ke wilayah kekuasaan AI, AI dapat menelisik ke dalam graf tadi bahwa:
   - Musuh pasti sudah membangun pandai besi dan lumber mill.
   - Musuh pasti sudah mengembangkan teknologi bubuk mesiu.
   - Musuh sedang memproduksi kayu dan besi.

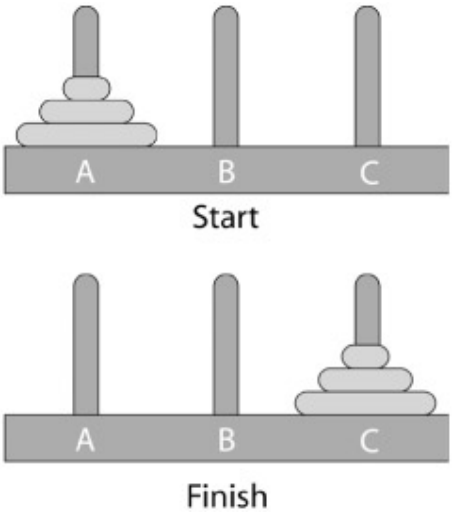
   Pemeriksaan lebih jauh ke dalam graf, dapat mengindikasikan bahwa musuh mungkin sudah memiliki meriam atau sedang memproduksinya.
   AI dapat menggunakan informasi ini untuk menentukan strategi penyerangna terbaik. Sebagai contoh, misalnya untuk menghindari lebih banyak lagi pasukan bersenjata musuh mendekati area kekuasaan AI, maka ia harus menghancurkan tempat pandai besi dan lumber mill milik musuh. AI juga dapat memproduksi assassin untuk menghancurkan pandai besi milik musuh agar musuh tidak dapat membuat pasukan lagi, maka ia harus menyediakan resource untuk membuat assassin.

3. Seringkali, sebuah unit menjadi kunci kemenangan dalam permainan. Jika biaya pembangunan unit ini dapat dilihat di dalam dependency graph, maka AI dapat memilih rute yang efisien untuk membangun unit ini.

4. STATE GRAPH

State graph adalah representasi setiap kemungkinan state yang mungkin dicapai oleh sistem dan juga menggambarkan transisi di antara state-state tersebut. Kumpulan dari state-state ini biasa disebut dengan state space. Graf dengan jenis ini dapat dicari di dalamnya apakah suatu state mungkin ada atau untuk menemukan rute yang paling efisien untuk menuju ke suatu state.[2]

Contoh masalah sederhana untuk state graph adalah masalah “Menara Hanoi”.

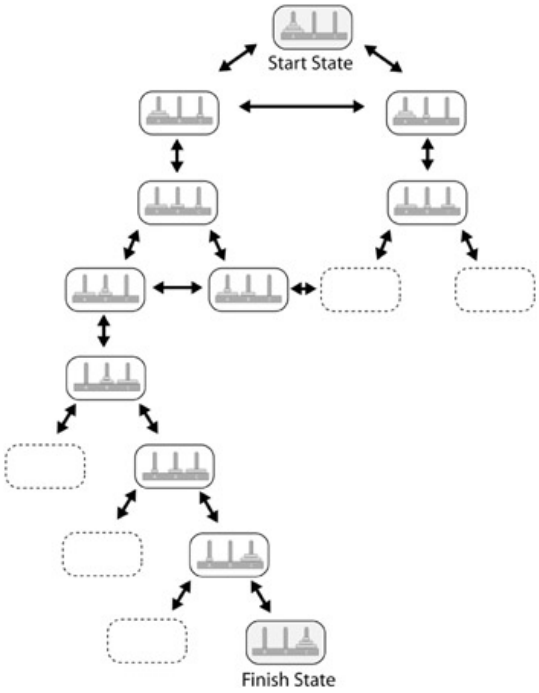


Gambar 10: Menara Hanoi

Pada contoh sederhana ini, terdapat tiga buah pasak dan tiga buah piringan yang berbeda-beda ukurannya. Ketiga piringan ini diletakkan pada pasak A. Tujuan kita adlaah bagaimana memindahkan ketiga piringan ini ke pasak C dengan mengikuti aturan bahwa piringan yang lebih besar selalu di bawah piringan yang lebih kecil.

Kita dapat merepresentasikan state space untuk masalah ini dengan menggunakan graf dimana setiap node pada graf tersebut menggambarkan state-state yang mungkin dicapai, dan setiap sisi menggambarkan trasnsisi dari satu state ke state yang lainnya. Graf ini dibangun dengan cara pertama-tama membangun sebuah node yang merepresentasikan sebuah state awal dari masalah Menara Hanoi ini. Node ini disebut dengan root node. Root ini kemudian diperluas dengan menambahkan state-state yang dapat dicapai dari state tersebut. Lalu tiap state yang baru tersebut juga diperluas dan seterusnya sampai semua kemungkinan state dan transisi telah ditambahkan ke dalam graf. State sebelum suatu state dinamakan parent state, dan state setelah suatu state dinamakan child dari parent state.

Gambar di bawah ini memperlihatkan proses pembangunan graf tersebut.



Gambar 11: Tahapan pembangunan state graph masalah Menara Hanoi

Graf ini bisa jadi bertambah rumit dengan cepat. Oleh karena itu di sini hanya digambarkan bagian yang menuju sebuah solusi.

State graph dapat digunakan untuk mencari solusi dari suatu permasalahan. Dalam contoh ini, solusinya adalah ketika semua piringan telah berada dalam pasak C dengan sesuai dengan aturan yang berlaku, yaitu piringan yang lebih besar berada di bawah piringan yang lebih kecil.

Jumlah rata-rata child node yang tumbuh dari parent node disebut dengan branching factor. Untuk beberapa masalah, seperti contoh ini branching factor-nya relatif rendah, sehingga akan ada cukup ruang di memori untuk merepresentasikan semua state-nya. Untuk kebanyakan masalah/bidang, branching factor-nya relatif besar sehingga semua state-nya tidak mungkin dimuat ke dalam memori. Selain itu, waktu proses pencariannya juga akan sangat lama. Oleh karen itu, graf ini dipecah-pecah pada suatu node dan akan dikerjakan sendiri-sendiri secara paralel.

5. KESIMPULAN

Kesimpulan yang dapat diambil dari studi tentang penggunaan graf di dalam pengembangan game dengan kecerdasan buatan adalah:

1. Bermacam-macam representasi graf dapat digunakan untuk membangun struktur data dalam program untuk penerapan dalam game. Contoh representasi tersebut misalnya: matriks dan adjacency list.
2. Untuk navigasi atau pathfinding dalam game dengan kecerdasan buatan dapat digunakan graf untuk merepresentasikan setiap lokasi di dalam lingkungan game dan melakukan pencarian di dalam struktur graf itu sendiri.
3. Untuk game yang bertipe resource management, dapat digunakan dependency graph untuk mengelola pengetahuan dari sistem kecerdasan buatan agar dapat digunakan untuk pengambilan keputusan, penaksiran, dan pengalokasian resource.
4. State grpah dapat digunakan untuk meprepresentasikan setiap keadaan/state di dalam game yang mungkin dicapai oleh sistem. State-state ini menggambarkan keadaan sistem pada saat tertentu. Representasi ini dapat digunakan untuk mencari solusi atas suatu permasalahan dalam game.
5. Penerapan graf beragam dan dapt berbeda-beda dari suatu game ke game lainnya.

DAFTAR REFERENSI

[1] Munir, Rinaldi, “Matematika Diskrit”, Edisi Ketiga, Penerbit Informatika: Bandung, 2005.

[2] Buckland, Mat, “Programming Game AI by Example”, Wordware Publishing, Inc, 2005.