Penerapan Algoritma *Branch and Bound* dalam Pencarian Rute Terpendek untuk Menyelesaikan *Task* pada Gim *Among Us*

Kent Liusudarso - 13520069
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520069@std.stei.itb.ac.id

***Abstrak*—*Among Us* merupakan gim *multiplayer* dimana setiap pemain dapat saling bekerja sama atau saling menghianati satu sama lain. Pemain terbagi menjadi dua peran, *crewmate* dan *impostor*. Objektif seorang *crewmate* yaitu mengisi *task bar* atau mengeluarkan *impostor*. Sedangkan objektif *impostor* yaitu membunuh *crewmate*. Untuk mengisi *task bar* sebagai *crewmate*, diperlukan pengerjaan *task* diberbagai tempat. Maka dari itu, untuk mengisi *task bar* secara efektif, diperlukan rute terpendek yang mengujungi lokasi *task* yang harus dikerjakan. Salah satu metode untuk menyelesaikan optimasi rute yaitu dengan algoritma *branch and bound*.**

Kata Kunci*—*Among Us; branch and bound; crewmate; gim; impostor; rute; sus; task; travelling salesman problem

I. PENDAHULUAN

Among Us merupakan gim multiplayer yang dapat dimainkan sebanyak 4 – 15 pemain. Pada awal permainan, setiap pemain akan diberikan peran sebagai *crewmate* atau *impostor*. Jumlah *crewmate* harus lebih banyak dibanding *impostor*. Jika *crewmate* memenuhi *task bar* sebelum jumlah *crewmate* berkurang hingga sama dengan jumlah *impostor*, maka *crewmate* akan memenangkan permainan. Sebaliknya, jika jumlah *crewmate* berkurang hingga sama dengan jumlah *impostor* sebelum *crewmate* memenuhi *task bar*, maka *impostor* akan memenangkan permainan.

Gambar 1 Tampilan Awal Pembagian Peran

Sebagai *crewmate*, salah satu cara efektif untuk memenangkan permainan, yaitu dengan menyelesaikan *task* yang tersebar diberbagai tempat secepat mungkin melalui pemilihan rute terpendek yang mengunjungi lokasi *task* yang harus dikerjakan. Salah satu permasalahan populer yang mirip dengan persoalan ini yaitu *Travelling Salesman Problem* (TSP). Permasalahan tersebut dapat diselesaikan dengan salah satu pendekatan, yaitu dengan penerapan *branch and bound*.

Gambar 2 Tampilan Awal Permainan Sebagai *Crewmate*

II. LANDASAN TEORI

A. *Algoritma Branch and Bound (B&B)*

Algoritma *Branch and bound* merupakan algoritma yang digunakan untuk persoalan optimasi, yaitu persoalan yang meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batasan (*constraints*) persoalan. Branch and bound terdiri dari gabungan *Breadth First Search* (BFS) dengan *least cost search*.

Pada *branch and bound*, setiap simpul diberi *cost* (c(i)), yaitu nilai taksiran lintasan termurah ke simpul status tujuan yang melalui simpul status i. Simpul berikutnya yang akan di-expand tidak lagi berdasarkan urutan pembangkitannya, tetapi simpul yang memiliki *cost* yang paling kecil (*least cost search*) – pada kasus minimasi [1].

Backtracking memiliki persamaan dengan *branch and bound* dalam pencarian solusi dengan pembentukan pohon ruang status dan juga mematikan simpul yang tidak mengarah ke solusi. Sedangkan perbedaannya yaitu *backtracking* tidak memiliki batasan (optimisasi/non-optimasi) dalam persoalan yang bisa diselesaikan (umumnya untuk persoalan non-optimisasi. Branch and bound digunakan untuk persoalan optimisasi, lalu untuk setiap simpul pada pohon ruang-status, diperlukan suatu cara penentuan batas (bound) nilai terbaik fungsi objektif pada setiap solusi yang mungkin, dengan menambahkan komponen pada solusi sementara yang direpresentasikan oleh simpul. *Branch and bound* juga menghasilkan nilai dari solusi terbaik.

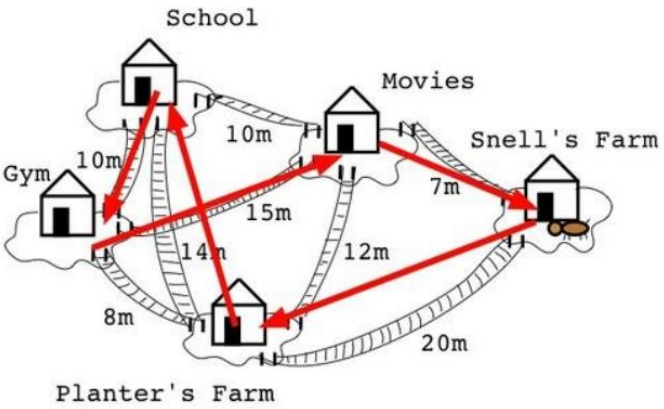
Selain itu dalam hal pembangkitan simpul, *branch and bound* memiliki beberapa aturan tertentu (umumnya menggunakan best-first rule). Sedangkan *backtracking* umumnya menggunakan *depth first search.*

Algoritma *branch and bound* menerapkan "pemangkasan" pada jalur yang tidak lagi mengarah pada solusi. Syarat dalam pemangkasan tersebut yaitu:

- Nilai simpul tidak lebih baik dari nilai terbaik sejauh inis
- Simpul tidak merepresentasikan solusi yang *'feasible'* karena ada batasan yang dilanggar
- Solusi pada simpul tersebut hanya terdiri atas satu titik → tidak ada pilihan lain; bandingkan nilai fungsi obyektif dengan solusi terbaik saat ini, yang terbaik yang diambil

B. *Travelling Salesman Problem (TSP)*

Travelling Salesman Problem merupakan persoalan graf yang populer. Bunyi persoalannya yaitu: "Diberikan n buah kota serta diketahui jarak antara setiap kota satu sama lain. Temukan perjalanan (*tour*) terpendek yang dimulai dari sebuah kota dan melalui setiap kota lainnya hanya sekali dan kembali lagi ke kota asal keberangkatan." Persoalan TSP tidak lain adalah menemukan sirkuit *Hamilton* dengan bobot minimum. Pada graf lengkap terdapat (n-1)! tur dengan n simpul [2].



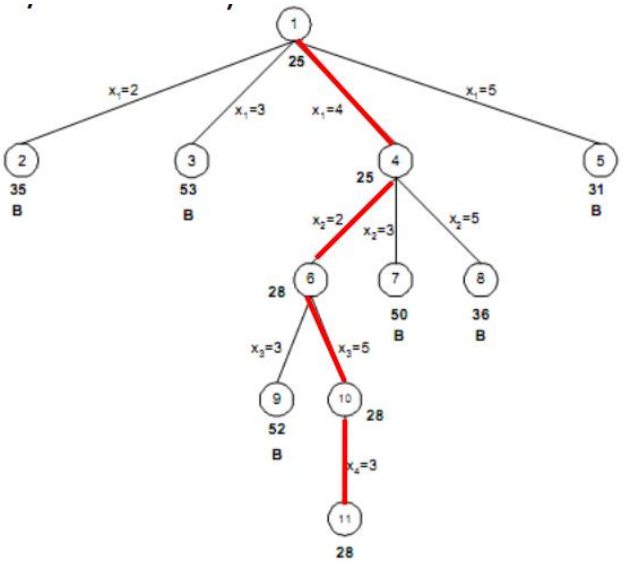
Gambar 3 Contoh Solusi *Travelling Salesman Problem*

Pada graf tak berarah TSP, simpul graf adalah kota, sedangkan sisi dari graf merupakan rute dan bobot sisi merupakan jarak antar kota. Graf ini adalah persoalan minimisasi yang dimulai dan diakhiri pada titik tertentu setelah mengunjungi semua simpul tepat satu kali.

Biasanya, persoalannya berupa graf lengkap, yaitu setiap simpul dihubungkan oleh sisi. Jika jalur antar dua kota tidak ditemukan, ditambahkan sisi yang panjang yang akan melengkapi graf tanpa mempengaruhi tur yang optimal.

C. *Penyelesaian TSP dengan B&B*

Travelling Salesman Problem dapat diselesaikan dengan algoritma *branch and bound* untuk mendapat solusi yang optimal.



Gambar 4 Contoh Pohon Ruang Status

Cost untuk setiap simpul di dalam pohon ruang status menyatakan nilai batas bawah (*lower bound*) ongkos mencapai simpul solusi dari simpul tersebut. Cost setiap simpul dapat dihitung secara heuristik berdasarkan salah satu dari dua cara:

1. Matriks ongkos-tereduksi (*reduced cost matrix*) dari graf
2. Bobot minimum tur lengkap

Untuk *cost* berdasarkan matriks ongkos tereduksi, Ongkos atau nilai batas untuk setiap simpul dihitung dengan menggunakan matriks ongkos-tereduksi (*reduced cost matrix*) dari graf G. Sebuah matriks dikatakan tereduksi jika setiap kolom dan setiap barisnya mengandung paling sedikit satu buah nol dan semua elemen lainnya non-negatif.

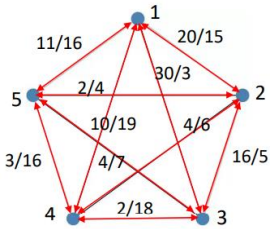
$$M = \begin{bmatrix} 12 & 20 & 30 & 10 & 11 \\ 15 & 8 & 16 & 4 & 2 \\ 3 & 5 & 11 & 2 & 4 \\ 19 & 6 & 18 & 9 & 3 \\ 16 & 4 & 7 & 16 & 8 \end{bmatrix}$$

Reduksi baris dan kolom

$$M' = \begin{bmatrix} 1 & 10 & 17 & 0 & 1 \\ 12 & 6 & 11 & 2 & 0 \\ 0 & 3 & 6 & 0 & 2 \\ 15 & 3 & 12 & 6 & 0 \\ 11 & 0 & 0 & 12 & 4 \end{bmatrix}$$

Tinjau sebuah TSP dengan n = 5, graf dinyatakan dalam matriks ketetanggaan:

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$



Gambar 5 Matriks Ketetanggaan

Karena perjalanan pedagang di dalam graf melalui sisi (i, j), dengan i = 1, 2, …, 5 dan j = 1, 2, …, 5, maka mengurangkan setiap elemen pada suatu baris atau pada suatu kolom dengan konstanta t akan mengurangi panjang (bobot) setiap perjalanan sebesar t.

Jika t dipilih dari elemen minimum pada baris i (kolom j), maka mengurangkan seluruh elemen pada baris i (kolom j) dengan t akan menghasilkan sebuah nol pada baris i (kolom j) tersebut. Dengan mengulangi proses ini berulangkali akan menghasilkan matriks bobot tereduksi.

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \begin{matrix} R_1 - 10 \\ R_2 - 2 \\ R_3 - 2 \\ R_4 - 3 \\ R_5 - 4 \end{matrix} \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix} \begin{matrix} \\ \\ C_1 - 1 \\ \\ C_3 - 3 \end{matrix} \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix} = A$$

Gambar 6 Hasil Reduksi Matriks Ketetanggaan

Jumlah total elemen pengurang dari semua baris dan kolom menjadi batas bawah (*lower bound*) dari tur dengan total bobot minimum. Nilai ini digunakan sebagai nilai untuk simpul akar pada pohon ruang status.

Misal tur (perjalanan) dimulai dari simpul 1, A merupakan matriks tereduksi untuk simpul R. S merupakan anak dari simpul R sehingga sisi (R, S) pada pohon ruang status berkoresponden dengan sisi (i, j) pada perjalanan. Jika S bukan simpul daun, maka matriks bobot tereduksi untuk simpul S dapat dihitung sebagai berikut [2]:

1. ubah semua nilai pada baris i dan kolom j menjadi ∞. Ini untuk mencegah agar tidak ada lintasan yang keluar dari simpul i atau masuk pada simpul j.
2. ubah A(j, 1) menjadi ∞. Ini untuk mencegah penggunaan sisi (j, 1).
3. reduksi kembali semua baris dan kolom pada matriks A kecuali untuk elemen ∞.
4. Jika r adalah total semua pengurang, maka nilai batas untuk simpul S adalah c(S) = c(R) + A(i, j) + r
5. Hasil reduksi ini menghasilkan matriks B.
6. Hal ini dilakukan pada semua simpul. Setelah itu, akan didapatkan simpul aktif dengan *cost* minimum.

Kompleksitas kasus terburuk dari *Branch and Bound* tetap sama dengan *Brute Force* karena dalam kasus terburuk, kita mungkin tidak akan pernah mendapatkan kesempatan untuk memangkas sebuah simpul. Padahal, dalam praktiknya kinerjanya sangat baik tergantung pada contoh TSP yang berbeda. Kompleksitas juga tergantung pada pilihan fungsi pembatas karena merekalah yang memutuskan berapa banyak simpul yang akan dipangkas.

III. IMPLEMENTASI

Pada implementasi kali ini, digunakan lima *task* di *The Skeld* sebagai sampel.

Gambar 7 Peta *The Skeld* dengan Lima *Task*

Pada Gambar 7 terlihat ada label disetiap lokasi yang menunjukkan nama lokasi tersebut. Lalu ada simbol lingkaran berwarna kuning dengan tanda seru yang merupakan *task* pada lokasi tersebut.

A. Pengumpulan Data

Data yang dibutuhkan yaitu jarak terdekat antar setiap *task* ke *task* lain pada peta *The Skeld*. Perangkat lunak yang digunakan untuk mengukur jarak pada peta adalah *ruler* dari *photoshop*. Dengan satuan yang digunakan yaitu pixel.

Gambar 8 Peta *The Skeld* dengan Jarak (px)

Untuk mempermudah pengolahan data, tiap *task* diberikan indeks.

TABLE I. INDEKS

Location: Task	**Indeks**
Cafetaria	0
Admin: Swipe Card	1
MedBay: Submit Scan	2
Shields: Prime Shields	3
Reactor: Start Reactor	4
Navigation: Stabilize Steering	5

Berdasarkan lokasi *task*, dapat ditentukan jarak antar setiap *task*. Jarak yang diambil adalah jarak yang minimum dari rute yang mungkin.

TABLE II. MATRIKS KETETANGGAAN JARAK ANTAR *TASK* DALAM SATUAN PIXEL

Indeks	**0**	**1**	**2**	**3**	**4**	**5**
0	∞	585	605	965	1140	1045
1	585	∞	1190	890	1725	1575
2	605	1190	∞	1570	1055	1650
3	965	890	1570	∞	1725	685
4	1140	1725	1055	1725	∞	2185
5	1045	1575	1650	685	2185	∞

B. Penyelesaian dengan Branch and Bound

Implementasi *branch and bound* yang digunakan adalah tentang *travelling salesman problem*, karena memiliki persoalan yang sama.

Algoritma *branch and bound* dibuat dengan bahasa pemrograman python yang menerima masukan berupa matriks ketetanggaan.

```
# Matriks ketetanggaan
adj = [[0, 585, 605, 965, 1140, 1045],
       [585, 0, 1190, 890, 1725, 1575],
       [605, 1190, 0, 1570, 1055, 1650],
       [965, 890, 1570, 0, 1725, 685],
       [1140, 1725, 1055, 1725, 0, 2185],
       [1045, 1575, 1650, 685, 2185, 0]]
```

Gambar 9 Input Matriks Ketetanggaan

Matriks tersebut kemudian diolah dengan *branch and bound* untuk problem *travelling salesman problem*. Lalu, yang keluaran yang dihasilkan yaitu:

```
Cost minimal : 6005
Rute yang dihasilkan :  0 1 3 5 2 4 0
```

Gambar 10 Output dari Program

C. Hasil

Rute terpendek yang dihasilkan dimodifikasi sesuai dengan persoalan, yaitu untuk menyelesaikan lima *task*, maka, setelah mengunjungi semua lokasi *task*, rute tidak perlu kembali ke tempat awal, jadi, tanpa kembali ke indeks 0, rute yang dihasilkan yaitu 0 – 1 – 3 – 5 – 2 – 4. Jika indeks diubah kembali menjadi nama *task* maka *Cafetaria* → *Admin: Swipe Card* → *Shields: Prime Shields* → *Navigation: Stabilize Steering* → *MedBay: Submit Scan* → *Reactor: Start Reactor*.

Cost minimum untuk mengunjungi semua *task* dan kembali ke tempat awal yaitu 6005. Tetapi pada persoalan kali ini tidak memerlukan kembali ke tempat awal (*Cafetaria*) untuk menyelesaikan *task*, maka *cost* akrinyanya yaitu *cost* minimum – Jarak *Reactor: Start Reactor* ke *Cafetaria* = 6005 – 1140 = 4765px.

Gambar 11 Ilustrasi Rute Terpendek

IV. KESIMPULAN

Algoritma *Branch and Bound* dapat menemukan rute terpendek untuk menyelesaikan *task* pada gim *Among Us*.

Eksperimen ini dilakukan dengan pengumpulan data berupa aset dari gim *Among Us*, jarak pada peta *The Skeld*, dilanjutkan dengan perhitungan jarak antar *task*.

Dari data – data yang terkumpul, data tersebut diolah lagi dengan menggunakan pendekatan *travelling salesman problem* dengan algoritma *branch and bound* yang diimplementasikan menggunakan bahasa pemrograman python.

Masukan berupa matriks ketetanggan dari jarak antar lima *task* dari peta *The Skeld*, yaitu: *Cafetaria, Admin: Swipe Card, MedBay: Submit Scan, Shields: Prime Shields, Reactor: Start Reactor, Navigation: Stabilize Steering*. Lalu keluaran yang dihasilkan yaitu *cost* minimum dan rute yang dihasilkan yaitu: *Cafetaria → Admin: Swipe Card → Shields: Prime Shields → Navigation: Stabilize Steering → MedBay: Submit Scan → Reactor: Start Reactor* dengan *cost* sebesar 4765px.

Dengan menggunakan rute yang dihasilkan dari eksperimen ini, maka peluang dalam memenangkan permainan sebagai *crewmate* dengan menyelesaikan *task* secepat mungkin melalui rute terpendek akan meningkat.

UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan Yang Maha Esa karena atas berkat dan rahmatnya, sehingga penulis dapat menyelesaikan makalah dengan tepat waktu. Harapannya adalah agar ilmu ini menjadi bermanfaat bagi pembaca.

Ucapan terima kasih kepada dosen pengampu mata kuliah IF2211 Strategi Algoritma K03, Rinaldi Munir, atas segala ilmu dan pedoman yang telah diberikan, sehingga makalah ini dapat diselesaikan dengan baik.

REFERENSI

[1] Rinaldi Munir, Nur Ulfa Maulidevi, dan Masayu Leylia Khodra, "Algoritma Branch & Bound (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, 2021.

[2] Rinaldi Munir, Nur Ulfa Maulidevi, dan Masayu Leylia Khodra, "Algoritma Branch & Bound (Bagian 2)," Bahan Kuliah IF2211 Strategi Algoritma, 2021.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Mei 2022

Kent Liusudarso - 13520069