

Penerapan Branch & Bound Berdasarkan Bobot Tur Lengkap untuk Menentukan Rotasi *Farming Jungler* dalam Permainan Mobile Legend

Timothy Stanley Setiawan -13520028

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): timothystanleysetiawan555@gmail.com

Abstract—Seiring dengan perkembangan teknologi yang begitu pesat. Dunia *game* menjadi sangat populer di kalangan masyarakat. Tidak jarang banyak orang yang menjadikan *game* sebagai mata pencaharian mereka. Setiap *game* memiliki genrenya masing-masing. Salah satu genre yang cukup populer adalah *Multiplayer Online Battle Arena* atau biasa disingkat MOBA. Mobile Legend (ML) merupakan salah satu *game online* bergenre MOBA yang dimainkan menggunakan perangkat *mobile*. Sejak perilisannya pada tanggal 14 Juli 2016, ML tidak pernah sepi peminat. Sebagai *game MOBA* yang cukup mudah dimainkan dan tidak memerlukan spesifikasi perangkat yang tinggi untuk memainkannya, membuat ML laku keras di pasar Asia Tenggara khususnya Indonesia. Dalam *game MOBA*, setiap pemain mempunyai rolenya masing-masing. Salah satu role yang cukup diminat dan digemari oleh para pemain (karena mampu membawa tim kepada kemenangan) adalah role *jungler* yang bertugas untuk melakukan *farming*. Untuk menyukseskan proses *farming* tersebut, seorang *jungler* perlu melakukan penentuan rute rotasi *farming* yang efisien. Permasalahan ini mirip dengan permasalahan berkaitan teori graf yang cukup populer, yaitu Travelling Salesman Problem (TSP). Salah satu metode pendekatan untuk menyelesaikan masalah ini adalah dengan menggunakan algoritma Branch & Bound yang costnya didapatkan melalui metode heuristik bobot tur lengkap.

Kata kunci—MOBA; Mobile Legend; Jungler; Rotasi Farming; Travelling Salesman Problem; Branch & Bound; bobot tur lengkap

I. PENDAHULUAN

Dewasa ini, perkembangan teknologi yang semakin pesat sangat berpengaruh dalam berbagai bidang kehidupan. Salah satunya adalah bidang hiburan, khususnya dalam aspek perkembangan dunia industri *game* yang sekarang sedang hangat-hangatnya. Dengan perkembangan tersebut, kini *game* dapat dimainkan diberbagai perangkat. Perangkat-perangkat yang cukup populer untuk memainkan sebuah *game* saat ini adalah *Play Station* (PS), *X-BOX*, *Personal Computer* (PC), dan *Mobile*. Selain itu, *game-game* yang bermunculan pun dapat dimainkan baik secara online, maupun offline. Hal tersebut membuat *game* semakin mudah untuk dimainkan karena bisa dimainkan di mana saja dan kapan saja. Tidak jarang pula suatu *game* tertentu bisa dimainkan di dua atau lebih perangkat tertentu. Dengan kemudahan-kemudahan inilah

membuat dunia industri *game* berkembang semakin kuat dan kokoh.

Tiap *game* memiliki genrenya masing-masing, mulai dari *First-Person Shooting* (FPS), *Role-Playing Games* (RPG), *Massively Multiplayer Online* (MMO), *Real-Time Strategy* (RTS), sampai *Multiplayer Online Battle Arena* (MOBA). Salah satu genre *game* yang cukup digandrungi oleh para pemain adalah *game* bergenre MOBA. *Gameplay* pertarungan 5 vs 5 yang ditawarkan oleh *game-game* sejenis mampu menarik cukup banyak pemain. *Game MOBA* ini sudah berkembang cukup pesat, dari yang awalnya hanya bisa dimainkan di perangkat PC, sekarang sudah bisa dimainkan diperangkat *mobile*.

Mobile Legend atau yang biasa disingkat menjadi ML merupakan salah satu *game online mobile* bergenre MOBA yang cukup digandrungi para pemain *game mobile*. Membawa *game* bergenre MOBA ke dalam perangkat *mobile* adalah sebuah keberanian dari pihak moonton yang pada saat perilisannya *game MOBA* masih banyak dimainkan diperangkat PC (contohnya DOTA, LOL). Bisa bilang ML memiliki *gameplay* yang paling sederhana dibandingkan saudara-saudaranya. *Item* yang bisa dibeli kapan pun dan hero yang bisa digerakan dengan analog digital dalam *game* adalah salah satu contoh kemudahan yang ditawarkan oleh ML. Selain itu, untuk memainkan *game* ini pun tidak perlu memiliki spesifikasi *mobile* (HP) yang tinggi Dengan demikian, ML dapat menggaet banyak pemain (baik yang sudah paham MOBA, maupun pemain baru) yang dapat memperluas pasar dan menjadikannya sebagai *game mobile* yang kuat dan kokoh.



Gambar 2 Hero Mobile Legend yang memiliki role jungler

Sumber : https://jurnalapps.co.id/assets/img/content/1600078296_logo.jpg

Untuk mencapai kemenangan dalam game ML, salah satu tim harus berhasil menghancurkan base utama musuh. Sebelum dapat menghancurkan base utama, ada tiga jalur berbeda (*safe lane* atau biasa disebut juga *goldlaner* dalam ML, *off lane*, *mid lane*) yang bisa dilalui untuk mencapai base utama musuh. Tiap-tiap jalur tersebut dilengkapi dengan *turret* yang akan menjaga jalan masuk ke base utama. *Turret-turret* di ketiga jalur tersebut harus dijaga oleh masing-masing tim sesuai dengan strategi mereka sendiri agar tidak dapat ditembus oleh musuh. Selain itu dalam permainan game MOBA, ada role-role tertentu yang dimainkan oleh masing-masing pemain. Role-role tersebut adalah *jungler*, *safelaner* atau biasa disebut juga *goldlaner* dalam ML, *offlaner*, *roammer*, dan *midlaner*. Semua role tersebut memiliki tugasnya masing-masing. *jungler* bertugas melakukan *farming*, *safelaner* bertugas menjaga *safe lane*, *offlane* bertugas menjaga *off lane*, *midlaner* bertugas menjaga *mid lane*, dan *roamer* bertugas untuk berpatroli membantu menjaga ketiga jalur tersebut. Tugas *farming* sendiri adalah menghabiskan seluruh monster *jungle* yang ada sehingga hero yang dimainkan mendapatkan exp dan gold lebih banyak. Dengan kondisi demikian, hero akan menjadi level dan *item* yang lebih tinggi dan mudah untuk mengalahkan hero-hero musuh.



Gambar 2 Hero Mobile Legend biasa dimainkan sebagai *jungler*

Sumber : Arsip pribadi

Permasalahan *farming* ini mirip seperti permasalahan TSP atau *Travelling Salesman Problem*. Banyak pendekatan untuk menyelesaikan masalah ini, salah satunya adalah pendekatan dengan algoritma Branch & Bound yang menggunakan metode heuristik bobot tur lengkap untuk penentuan costnya. Dengan memanfaatkan algoritma ini, diharapkan rotasi *farming* yang dilakukan menjadi lebih efektif dan total 7 monster *jungle* (8 ditambah *turtle*) dapat dikunjungi dan diperoleh (dibunuh) secepat mungkin sehingga level dan *item* yang dimiliki oleh hero akan lebih cepat terkumpul dan *damage output* yang diberikan akan lebih besar dibanding hero musuh. Dengan demikian, *chance* kemenangan akan lebih besar karena hero musuh tidak mampu untuk mengalahkan hero tersebut.

II. LANDASAN TEORI

A. Algoritma Branch & Bound

Algoritma Branch & Bound (B&B) adalah algoritma yang digunakan untuk menyelesaikan persoalan optimisasi. Persoalan optimisasi adalah persoalan meminimalkan atau memaksimalkan suatu fungsi objektif dengan tidak melanggar batasan (*constraints*) persoalan. Branch & Bound bisa dibilang sebagai optimisasi dari algoritma BFS yang menggunakan sistem FIFO dalam pembangkitan simpul ekspansinya,

sedangkan untuk algoritma B&B aturan tiap simpulnya adalah sebagai berikut:

- Setiap simpul memiliki cost $\hat{c}(i)$ yang berarti nilai taksiran lintasan paling optimal ke simpul status tujuan yang melalui simpul status i .
- Simpul ekspansi dibangkitkan berdasarkan urutan simpul yang memiliki cost terkecil (untuk minimasi) atau cost terbesar (maksimasi)

Selain BFS, ada algoritma lain yang mirip dengan B&B. Algoritma tersebut adalah algoritma Backtracking, berikut ini perbandingan keduanya:

a. Persamaan :

- Pencarian solusi dilakukan dengan membentuk pohon ruang status
- “Memotong” simpul yang tidak “mengarah” ke solusi

b. Perbedaan :

- Persoalan yang diselesaikan algoritma Backtracking lebih umum, sedangkan B&B hanya untuk persoalan optimisasi dengan melakukan penentuan batas nilai fungsi objektif pada setiap solusi yang mungkin untuk mendapatkan nilai solusi terbaik untuk saat ini.
- Pembangkitan simpul pada Backtracking biasa menggunakan DFS, sedangkan pada B&B dengan aturan tertentu (*best-first rule*)

Seperti yang disebutkan sebelumnya, algoritma B&B menerapkan aturan “pemotongan” terhadap simpul yang tidak mengarah kepada solusi. Kriteria untuk melakukan “pemotongan” tersebut:

- Nilai simpul tidak lebih baik dari nilai solusi terbaik saat ini
- Simpul melanggar batasan yang sudah ditetapkan sebelumnya
- Solusi pada simpul hanya terdiri dari satu titik (tidak ada pilihan). Untuk kriteria ini, nilai fungsi objektif dibandingkan terlebih dahulu dengan solusi terbaik saat ini, kemudian pilih solusi dengan nilai yang lebih baik.

Secara umum, langkah-langkah penyelesaian dengan menggunakan algoritma B&B adalah :

1. Memasukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah solusi, solusi ditemukan dan berhenti jika hanya menginginkan sebuah solusi
2. Jika Q kosong, berhenti.
3. Jika Q tidak kosong, memilih dari antrian Q simpul i yang mempunyai nilai cost $\hat{c}(i)$ paling optimal (terkecil/ terbesar). Jika ada beberapa simpul yang memenuhi, pilih salah satu secara sembarang.

4. Jika i adalah simpul solusi, solusi ditemukan dan berhenti jika hanya menginginkan sebuah solusi
5. Jika i bukan simpul solusi, simpul anak-anaknya dibangkitkan. Jika i tidak memiliki anak, kembali ke langkah 2.
6. Untuk semua simpul anak j dari i hitung cost $\hat{c}(j)$, dan masukan anak-anak tersebut ke Q
7. Kembali ke langkah 2.

B. Heuristik Cost Berdasarkan Bobot Tur Lengkap

Bobot Tur Lengkap adalah bobot yang diperlukan untuk melakukan satu kali tur lengkap dari simpul a sampai kembali lagi ke simpul a lagi setelah melalui semua simpul yang ada. Contohnya adalah sebagai berikut :



Gambar 3 Visualisasi tour lengkap

Sumber : Slide perkuliahan mata kuliah IF2211 Strategi Algoritma

Untuk mendapatkan Bobot Tur Lengkap dari graf di atas, bisa dengan menambahkan semua bobot-bobot pada sisinya atau dengan rumus sebagai berikut :

$$\text{bobot tur lengkap} = \frac{1}{2} \sum_{i=1}^n \text{bobot sisi } i_1 + \text{bobot sisi } i_2$$

Gambar 4 Rumus bobot tur lengkap

Sumber : Slide perkuliahan mata kuliah IF2211 Strategi Algoritma

Dengan i_1 dan i_2 adalah dua sisi yang saling bersisian pada simpul i yang berada pada graf. Dari kedua cara tersebut, hasil yang didapatkan adalah:

- Total Sisi:
 $B = 10 + 15 + 8 + 12 = 45$
- Rumus Bobot Tur Lengkap :
 $B = \frac{1}{2} \times [(10 + 12) + (10 + 15) + (15 + 8) + (8 + 12)] = 45$

Rumus Bobot Tur Lengkap tadi dapat digunakan sebagai pendekatan untuk mencari nilai cost dari tiap simpul untuk penyelesaian TSP

$$M \equiv \text{cost} = \text{bobot minimum tur lengkap} \\ \geq \frac{1}{2} \sum \text{bobot sisi } i_1 + \text{bobot sisi } i_2$$

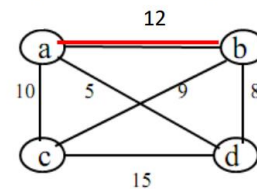
Gambar 5 Pendekatan cost dengan rumus bobo tour lengkap

Sumber : Slide perkuliahan mata kuliah IF2211 Strategi Algoritma

Dalam pendekatan ini, i_1 dan i_2 adalah dua sisi yang saling bersisian pada simpul i (yang berada pada graf) dengan bobot paling minimum. Dalam penggunaannya untuk kasus ini, dalam persoalan mencari cost dari tiap simpul (kecuali akar),

ada sisi yang wajib diambil. Hal ini berarti rute yang diambil akan melewati sisi tersebut. Sebagai contoh dibawah ini :

Untuk $i_2=b$, sisi (a, b) wajib diambil.



Gambar 6 Visualisasi simpul ke-i yang mengambil rute melalui sisi a,b

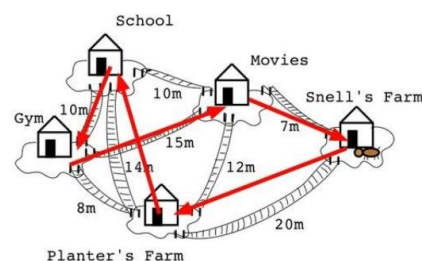
Sumber : Slide perkuliahan mata kuliah IF2211 Strategi Algoritma

Simpul ke-i ini menunjukkan bahwa rute yang diambil untuk menuju simpul b dari a adalah dengan melalui sisi a,b. Sehingga dalam perhitungan nilai costnya sisi a,b ini harus diambil walaupun nilai bobotnya bukan yang terendah. Perhitungan rumus cost untuk simpul i ini adalah :

$$\text{Cost} = \frac{1}{2} \times [(12 + 5) + (12 + 8) + (9 + 10) + (8 + 5)] = 32$$

C. Travelling Salesman Problem

Travelling Salesman Problem (TSP) merupakan salah satu permasalahan berkaitan dengan teori graf yang cukup populer hingga saat ini. Walaupun cukup populer, masih belum ada algoritma yang benar-benar dapat menyelesaikan permasalahan tingkat NP-Hard secara efisien.



Gambar 7 Visualisasi TSP

Sumber : Slide perkuliahan mata kuliah IF2211 Strategi Algoritma

Deskripsi dari persoalan TSP itu sendiri adalah “Diberikan n buah kota (vertex) serta diketahui jarak (bobot) antara setiap kota satu sama lain. Temukan perjalanan (tour) dengan jarak terpendek yang dilakukan oleh seorang pedagang sehingga melalui setiap kota tepat hanya sekali dan kembali lagi ke kota asal keberangkatan” [1]

Singkatnya adalah mencari sirkuit Hamilton terpendek dari suatu graf (baik lengkap maupun tidak, asalkan memiliki sirkuit Hamilton). Kemudian dari deskripsi persoalan tersebut, didapatkan informasi bahwa suatu graf lengkap dengan simpul sebanyak n dapat membuat $(n-1)!$ tur.

Pada dasarnya, algoritma Brute Force saja sudah mampu untuk menyelesaikan masalah TSP ini. Namun, dengan jumlah tur sebesar itu akan sangat tidak efisien jika harus ditelusuri satu per satu. Perlu waktu yang sangat lama untuk menyelesaikan sebuah persoalan TSP dengan algoritma Brute Force. Oleh sebab itu, salah satu algoritma penyelesaian TSP yang paling efektif untuk saat ini adalah dengan pendekatan

algoritma B&B yang akan memotong banyak sekali kemungkinan simpul yang tidak akan pernah menuju solusi.

III. TEORI IN GAME

A. Farming dan Monster Jungle

Sebagai seorang jungler, kemampuan untuk melakukan *farming* adalah suatu keharusan yang harus ia kuasai. *Farming* adalah kemampuan untuk menghabisi seluruh monster *jungle* yang ada sehingga hero yang dimainkan mendapatkan exp dan gold lebih banyak. Hero yang dimainkan pun akan memiliki level dan *item* yang lebih tinggi. Dengan kondisi seperti ini, hero yang ditempatkan sebagai *jungler* akan memiliki potensi *damage* yang besar sehingga mudah untuk membunuh hero dan menghancurkan *turret*-nya karena sudah tidak ada penjaganya lagi. Pesaing *damage* dari hero jungler ada hero *safelaner* atau *goldlaner* yang sengaja diatur oleh moonton untuk bisa mendapatkan gold lebih banyak dibanding role lain (selain *jungler*). Oleh sebab itu, penting bagi seorang *jungler* untuk bisa melakukan *farming* secepat mungkin agar heronya lebih cepet untuk “jadi”.

Untuk bisa melakukan *farming* dengan efisien, perlu memahami monster-monster *jungle* yang ada di ML. Pemahaman ini dapat dijadikan dasar dalam penentuan bobot-bobot tiap-tiap sisi dari hasil representasi graf. Dalam ML total ada 7 jenis monster *jungle* (9 jika dengan Lord dan Turtle), berikut ini ketujuh monster *jungle* tersebut :

1. Lithowanderer

Monster *jungle* yang berada di area sungai dengan jalur *mid lane* ini memiliki atribut HP sebesar 2501. Ketika dibunuh, monster ini akan memberi 92 gold dan memberikan *buff* berupa ikutnya lithowanderer bersama hero yang membunuhnya dan memulihkan 1% Hp dan Mana setiap detiknya.

2. Crab

Monster *jungle* ini cukup unik dan berbeda dari monster lainnya karena memiliki dua fase, yaitu Crab dan Little Crab. Letaknya berada di area sisi paling atas dari map dan memiliki attribute HP sebesar 3640 di versi biasa dan 1820 di versi kecilnya. Ketika dibunuh monster ini akan memberikan $56 + 6 \times 10$ untuk versi biasa dan $36 + 3 \times 10$ untuk versi kecilnya.

3. Serpent

Monster *jungle* yang sering disebut juga sebagai blue buff ini berada di area hutan. HP yang dimiliki oleh monster ini adalah sebesar 4090. Ketika berhasil di bunuh monster ini akan memberi gold sebesar $22 + 104$ dan juga buff berupa pengurangan seluruh *cooldown skill* sebesar 10%, penggunaan mana sebesar 40% dan penggunaan energi sebesar 25%. [2]

4. Scaled Lizard

Monster *jungle* di area hutan dekat *top lane* ini memiliki HP sebesar 3019. Ketika berhasil di bunuh, monster ini akan memberikan 92 gold dan pemulihan 5% mana dan 350 HP dalam 2 detik.

5. Fiend

Monster *jungle* yang memiliki nama lain monster red buff sama seperti Serpent ini berdasar di area hutan. Atribut HP yang dibawa oleh monster ini adalah 4.901. Gold yang diberikan ketika berhasil membunuh monster ini adalah 120. Selain itu, monster ini juga akan memberikan buff berupa tambahan *damage* dan menyebabkan efek slow kepada target selama 1 detik. Untuk Hero Assassin/Fighter/Tank memberikan $50(+50\% \text{ Total Physical ATK})$ dan menyebabkan efek Slow sebesar 80% sedangkan untuk hero Marksman/Mage/Support akan membuat tambahan *damage* sebesar $50(+80\% \text{ Total Physical ATK})$ dan efek Slow sebesar 30%. [2]

6. Crammer

Monster *jungle* di area hutan dekat *bot lane* ini memiliki HP sebesar 2501. Gold diberikan oleh monster ini adalah $44 + 59$ dan memberikan efek pemulihan 5% mana dan 350 HP dalam 2 detik sama seperti scaled lizard.

7. Rockursa

Monster *jungle* di area hutan dekat *mid lane* ini memiliki HP sebesar 3000. Ketika dibunuh, monster ini akan memberikan gold sebanyak 81 dan memberikan efek yang sama seperti Scaled Lizard dan Crammer.

Tambahan

8. Cryoturtle

Monster *jungle* ini berada di area tengah sungai dan memiliki HP sebesar 10000. Monster ini akan memberikan gold sebanyak 80 kepada seluruh tim dan memberikan buff berupa Shield yang dapat menyerap $400(+40 \times \text{Level Hero})$, meningkatkan Physical Attack sebesar $20(+20 \times \text{Level Hero})$ atau Magical Attack sebesar $25(+4 \times \text{Level Hero})$ untuk hero Mage. [2]

9. Sanctuary Lord

Monster *jungle* ini adalah monster *jungle* terkuat di ML yang memiliki attribute HP sebesar 30000 dan ketika dibunuh monster ini akan membantu kita untuk melakukakn *pushing* (menyerang) *turret*.

Catatan : Informasi-informasi di atas merupakan sudut pandang dari pemain yang tentunya akan terlihat terbalik jika dilihat dari sudut pandangan musuh. Untuk lithowanderer dan turtle sendiri bisa muncul dari bagian kiri atau pun kanan dari map. Namun, pada kesempatan ini akan menggunakan asumsi bahwa keduanya muncul di sebelah kiri map.

B. Map ML dan Persebaran Monster



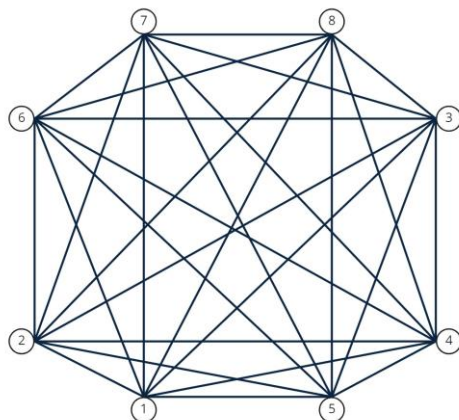
Gambar 7 Visualisasi TSP

Sumber : Arsip pribadi dengan mengambil map dari https://mobile-legends.fandom.com/wiki/Map?file=Sanctum_Island_Map.png

Dari gambar tersebut, berikut ini keterangan dari tiap letak monster *jungle* yang ada pada map adalah:

1. Fiend
2. Serpent
3. Lithowanderer
4. Rockursa
5. Crammer
6. Scaled Lizard
7. Crab
8. Turtle

Kemudian, map tersebut dapat digambarkan dalam visualisasi graf sebagai berikut



Gambar 8 Visualisasi persebaran monster *jungle* dalam graf

Sumber : Arsip pribadi

Seperti yang terlihat pada visualisasi graft sebelumnya, pemodelan graf yang dipilih untuk merepresentasikan rute antara monster *jungle* merupakan suatu graf lengkap. Hal ini bukan tanpa alasan, melainkan karena setiap monster bisa dikunjungi dari monster *jungle* mana pun.

Dengan informasi-informasi di atas dan pemodelan persebaran monster *jungle* dalam bentuk graf, langkah selanjutnya adalah menentukan rute rotasi *farm*ing. Penentuan rute rotasi *farm*ing akan dimodelkan menjadi seperti permasalahan TSP. Dengan bobot-bobot pada setiap sisi graf akan diambil dari perhitungan data monster *jungle* dan juga waktu tempuh untuk mengunjungi monster *jungle* satu ke lainnya.

IV. IMPLEMENTASI DAN PEMBAHASAN

A. Data Monster *Jungle*

Seperti yang sudah dijelaskan pada bab sebelumnya, setiap monster *jungle* mempunyai atribut yang berbeda-beda dan juga “worth” berupa buff yang akan diberikan. Nilai-nilai buff tersebut berasal dari pengalaman penulis sendiri ketika bermain yang berdasarkan seberapa besar pengaruh buff tersebut terhadap permainan. Berikut ini penjelasan lengkapnya :

Tabel 1. Tabel Data Perolehan Gold, Nilai Suatu buff, dan Atribut HP dari Monster *Jungle*

Nama	Gold	Buff	HP
Fiend	22 + 104	4	4090
Serpent	120	4	4901
Lithowanderer	92	3	2501
Rockursa	81	1	3000
Crammer	44 + 59	1	2501
Scaled Lizard	92	1	3019
Crab	56 + 6 x 10/ 36 + 3 x 10	0	3640/ 1820
Turtle	80	5	10000

B. Data Waktu Tempu antar monster *jungle*

Selain memanfaatkan informasi-informasi sebelumnya, untuk mendapatkan bobot tiap sisi, perlu ditambahkan data-data waktu tempuh untuk mengunjungi satu monster *jungle* ke monster *jungle* lainnya. Berikut ini data yang diperoleh dalam bentuk matrik *adjacency*:

Tabel 2 Data waktu tempuh antar simpul

	1	2	3	4	5	6	7	8
1	-	7	11	2	5	12	17	13
2	7	-	5	7	12	4	8	4
3	11	5	-	9	14	4	5	2
4	2	7	9	-	5	11	15	11

5	5	12	14	5	-	18	21	16
6	12	4	4	11	18	-	3	5
7	17	8	5	15	21	3	-	2
8	13	4	2	11	16	5	2	-

6	2.14	0.72	0.86	4.27	6.62	-	1.85	0.78
7	3.67	1.75	1.36	9.44	12.13	1.85	-	0.37
8	1.38	0.43	0.24	1.73	2.46	0.78	0.37	-

C. Bobot tiap simpul dari data yang ada

Dari informasi-informasi yang sudah dikumpulkan di atas, penulis membuat suatu perhutingan untuk mencari nilai bobot dari tiap sisi dengan memperhatikan peroleh gold, buff, dan jumlah darah yang diperoleh berserta jarak tempuhnya. Berikut ini pertimbangan perhitungan yang penulis buat :

$$\text{Worth}(i) = (G(i) \div (H(i) \div 100)) + B(i)$$

$$\text{Bobot}(i,j) = J(i,j) \div (\text{Worth}(i) + \text{Worth}(j))$$

Dengan

Worth(i) adalah keuntungan dari monster *jungle* i

G(i) adalah peroleh gold pada monster *jungle* i

H(i) adalah HP monster *jungle* i

B(i) adalah nilai buff pada monster *jungle* i

Bobot(i,j) adalah bobot antara monster *jungle* i dan monster *jungle* j

J(i,j) adalah waktu tempuh antara monster *jungle* i dan monster *jungle* j

Permasalahan TSP yang diselesaikan dengan menggunakan pendekatan B&B akan mencari ongkos paling minimum untuk menelusuri semua simpul yang ada. Dengan demikian, diambil pertimbangan perhitungan rumus bahwa waktu tempuh akan berbanding lurus dengan bobot, sedangkan “worth”/keuntungan dari membunuh monster *jungle* akan berbanding terbalik. Keputusan ini diambil karena semakin banyak waktu tempuh yang perlukan, maka semakin berat juga bobot untuk melalui sisi tersebut (semakin dihindari). Kemudian untuk “worth”, semakin tinggi nilainya, bobot untuk sisi tersebut akan semakin ringan (semakin diutamakan). Ingat! Karena yang dicari adalah ongkos paling minimum. Semakin kecil bobot suatu sisi, semakin besar juga keuntungan yang diperoleh (asumsi semakin cepat sampai, tetapi pada kasus ini diberi sedikit modifikasi, yaitu semakin tinggi “worth”-nya bobot akan ikut semakin ringan).

Dari perhitungan rumus tadi, berikut ini bobot-bobot yang dimiliki oleh tiap simpul dalam bentuk matrik *adjacency*

Tabel 3 Bobot antar simpul

	1	2	3	4	5	6	7	8
1	-	0.82	1.43	0.36	0.87	2.14	3.67	1.38
2	0.82	-	0.66	1.27	2.12	0.72	1.75	0.43
3	1.43	0.66	-	1.94	2.93	0.86	1.36	0.24
4	0.36	1.27	1.94	-	1.86	4.27	9.44	1.73
5	0.87	2.12	2.93	1.86	-	6.62	12.13	2.46

D. Penerapan algoritma B&B dengan heuristik bobot tur lengkap untuk penentuan rute rotasi *farming*

Dari data-data yang sudah dikumpulkan dan diperhitungkan sebelumnya, data tersebut selanjutnya akan diolah agar bisa diperoleh rute rotasi *farming* paling efektif. Pada penerapan dengan algoritma ini, hal pertama yang harus dilakukan adalah menentukan bobot akar dari pohon ruang status yang akan dibuat. Pada kesempatan ini mengasumsikan bahwa rute rotasi ini akan di mulai dari monster *jungle* serpent. Untuk cost yang dihasilkan pada simpul akar adalah sebagai berikut

$$\text{Cost} = \frac{1}{2} [(0.36 + 0.82) + (0.43 + 0.66) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37)] = 5.685$$

Untuk mempermudah perhitungan, simpul-simpul selanjutnya akan dihitung dengan menggunakan program python. Berikut ini algortima program tersebut dalam bentuk pseudocode berikut ini

1. Utils

FUNCTION Worth(i):
{fungsi ini akan mengembalikan nilai “worth” dari simpul i}
FUNCTION Bobot(i, j):
{fungsi ini akan mengembalikan nilai bobot pada sisi i, j}
PROCEDURE CalculateBobot():
{prosedur ini akan melakukan perhitungan bobot yang akan di simpan di variable MatrikCost}

2. Bobot Tur Lengkap

FUNCTION calculateCost(listSimpul):
{fungsi ini akan mengembalikan cost dari listSimpul yang dimasukkan}
cost <- 0
FOR i in range(1, len(MatrikCost)):
listSisi <- deepcopy(MatrikCost[i])
IF (len(listSimpul) > 1 AND i in listSimpul):
{menghitung cost dengan memperhatikan sisi-sisi yang perlu diambil}
ELSE:
{menghitung tanpa memperhatikan sisi-sisi yang perlu diambil}
ENDIF
ENDFOR
RETURN cost/2
ENDFUNCTION

3. B&B

```
PROCEDURE FarmRoute():
{procedure pencarian rute rotasi Farming}

pq <- PriorityQueue()
start <- [2]
pq.enqueue(Node(None, calculateCost(start), start))

n <- 0
while (not pq.empty()):
    n += 1
    node <- pq.dequeue()
    listSimpul <- node.list_simpul
    IF (len(listSimpul) = len(MatrikCost) - 1):
    {potong simpul yang memiliki nilai > dari solusi
    Terbaik saat ini}
    ELSE:
    {membangkitkan simpul anak-anaknya}
    ENDIF
ENDWHILE
ENDPROCEDURE
```

Seperti yang terlihat dalam pseudocode di atas, untuk pengambilan simpul dengan bobot terkecil diserahkan oleh priority queue sehingga tidak perlu repot-repot untuk melakukan *searching* pencarian simpul dengan cost terendah. Dengan menggunakan priority queue, hanya dengan memanggil *method dequeue* yang sudah diimplementasikan sebelum, simpul yang dipilih sudah merupakan simpul dengan bobot terendah. Oleh karena itu, kita hanya perlu mengecek apakah simpul sudah diambil semua atau belum. Jika sudah, rute tersebut akan menjadi solusi terbaik saat itu yang kemudian akan dilanjutkan dengan pemotongan simpul-simpul yang tidak mungkin menuju ke tujuan (costnya > nilai solusi terbaik saat itu). Jika sebaliknya, cukup melakukan iterasi untuk membangkitkan seluruh anak simpul yang mungkin. Berikut ini contoh perhitungan cost untuk pembangkitan beberapa anak simpul dari iterasi ke 1 dan ke 2 :

1. Iterasi 1

```
Iterasi 1
ID : 0
Serpent
Cost : 5.68
-----
cost = X [(0.36 + 0.82) + (0.82 + 0.43) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 5.76
cost = X [(0.36 + 0.82) + (0.66 + 0.43) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 5.68
cost = X [(0.36 + 0.82) + (1.27 + 0.43) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 5.99
cost = X [(0.36 + 0.82) + (2.12 + 0.43) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 6.42
cost = X [(0.36 + 0.82) + (0.72 + 0.43) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 5.71
cost = X [(0.36 + 0.82) + (1.75 + 0.43) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 6.23
cost = X [(0.36 + 0.82) + (0.43 + 0.66) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 5.68
```

Gambar 9 Perhitungan cost untuk membangkitkan anak simpul pada iterasi 1

Sumber : Arsip pribadi

Pada iterasi satu ini, simpul yang diambil adalah simpul akar karena simpul ini lah satu-satu simpul yang masih ada di dalam atrian. Dengan demikian, simpul ini memiliki nilai cost terkecil saat itu dengan nilai cost sebesar 5.68.

2. Iterasi 2

```
Iterasi 2
cost = X [(0.36 + 0.82) + (0.66 + 0.43) + (1.43 + 0.24) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 6.07
cost = X [(0.36 + 0.82) + (0.66 + 0.43) + (1.94 + 0.24) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 6.31
cost = X [(0.36 + 0.82) + (0.66 + 0.43) + (2.93 + 0.24) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 6.82
cost = X [(0.36 + 0.82) + (0.66 + 0.43) + (0.86 + 0.24) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 5.79
cost = X [(0.36 + 0.82) + (0.66 + 0.43) + (1.36 + 0.24) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 6.04
cost = X [(0.36 + 0.82) + (0.66 + 0.43) + (0.24 + 0.66) + (0.36 + 1.27) + (0.87 + 1.86) + (0.72 + 0.78) + (0.37 + 1.36) + (0.24 + 0.37) + ] = 5.68
```

Gambar 10 Perhitungan cost untuk membangkitkan anak simpul pada iterasi 2

Sumber : Arsip pribadi

Untuk iterasi kedua ini, simpul yang diambil adalah simpul dengan ID 2 (dibangkitkan kedua persis setelah akar). Simpul yang menghubungkan Serpent dengan Lithowanderer ini memiliki bobot yang sama dengan iterasi sebelumnya, yaitu 5.68 dan memiliki nilai cost terkecil saat itu.

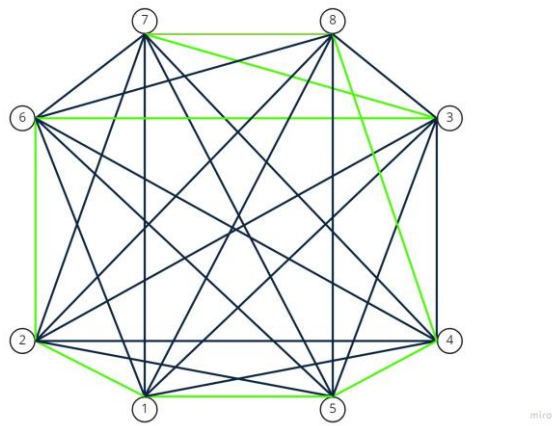
Pada pencarian rute rotasi *farming* yang dimulai dari monster Serpent ini, total memerlukan 82 kali iterasi hingga ditemukan solusi yang paling optimal. Iterasi yang cukup banyak ini bisa terjadi karena total cost dari pembangkitan anak-anak simpul ini agak berdekatan. Dengan demikian, ada kemungkinan priority queue akan memilih simpul-simpul dengan kedalaman lebih rendah terlebih dahulu karena simpul-simpul tersebut masuk ke dalam queue terlebih dahulu. Untuk rute rotasi *farming* paling efektif yang berhasil ditentukan itu sendiri adalah

```
Solusi terbaik:
ID : 173
Serpent → Fiend → Crammer → Rockursa → Turtle → Crab → Lithowanderer → Scaled Lizard
Cost : 6.12
-----
Waktu eksekusi: 0.164 s
```

Gambar 11 Hasil Pencarian

Sumber : Arsip pribadi

Rute rotasi *farming* dengan melalui Serpent → Fiend → Crammer → Rockursa → Turtle → Crab → Lithowanderer → Scaled Lizard sebelum kembali ke Serpent ini menjadi rute terbaik untuk melakukan rotasi *farming* menurut penelusuran dari algoritma B&B yang diimplementasi. Pengambilan kedua buff (merah dan biru) dari monster Serpent dan Fiend di awal-awal menjadi keputusan yang bijak untuk memulai rotasi *farming*. Dari informasi data monster *jungle* pun, kedua monster ini memiliki “worth” yang paling tinggi jika dihitung per satu hero. Namun, jika perhitungan “worth” ini sedikit diubah untuk semua team, ada kemungkinan rute yang diambil akan memilih Turtle terlebih dahulu. Berdasarkan hal-hal tersebut, hasil penentuan rute dari algoritma B&B dapat dipercaya dan dipraktikan langsung ketika bermain sebagai *jungler*.



Gambar 12 Visualisasi rute yang diambil dari hasil penelusuran (sisi dengan warna hijau)

Sumber : Arsip pribadi

V. KESIMPULAN

Sebagai game MOBA, Mobile Legend tentunya tidak bisa lepas dari suasana kompetitif. Tidak hanya dalam ranah kompetisi saja, bahkan untuk sekadar bermain *ranked* pun suasana tersebut sangat terasa kental. Dengan memiliki *skill farming* dengan efektif, cukup membantu untuk memenangkan suatu pertandingan. Mengingat role *jungler* adalah role yang mampu “menggendong” tim untuk membawa kemenangan. Penentuan rute rotasi farming adalah salah satu kemampuan yang harus dimiliki *jungler* untuk menjamin keberhasilan proses *farming*. Dengan memanfaatkan algoritma B&B, permasalahan tersebut bisa terbantu. Memiliki konsep masalah yang mirip TSP tidak heran jika permasalahan rotasi *farming* ini dapat terselesaikan dengan pendekatan algoritma B&B, walaupun algoritma ini dianggap masih belum sepenuhnya mangkus.

Rute yang terbentuk dari penelusuran dengan algoritma B&B bisa dibilang sudah cukup “*make sense*” jika dibandingkan secara langsung dengan data yang ada. Namun, penentuan rute ini dapat lebih tepat lagi apabila diberi tambahan data-data lagi seperti *damage* hero, *exp* yang di dapatkan, dan masih banyak lagi. Dengan adanya tambahan data tersebut, bobot yang dimiliki oleh masing-masing sisi semakin dipercaya.

VI. UCAPAN SYUKUR

Pertama-tama penulis memanjatkan syukur kepada Tuhan Yang Maha Esa, atas berkat rahmat dan penyertaannya penulis dapat menyelesaikan pengerjaan makalah yang berjudul “Penerapan Branch & Bound Berdasarkan Bobot Tur Lengkap untuk Menentukan Rotasi Farming Jungler dalam Permainan Mobile Legend” dengan baik dan tepat pada waktunya. Penulis juga mengucapkan syukur yang sebesar-besarnya kepada Ibu Dr. Masayu Leylia Khodra, ST, MT yang sudah mengajarkan penulis selama satu semester ini. Ilmu yang ibu ajarkan selama satu semester ini sangat berguna dalam kehidupan perkuliahan saya. Terakhir, tidak lupa penulis juga mengucapkan syukur kepada keluarga dan orang tua penulis yang senantiasa memberi dukungan kepada penulis.

LINK PENTING

Youtube : <https://youtu.be/5CZtx6zVhb4>

GitHub : <https://github.com/Stanley77-web/Makalah-Stima.git>

REFERENCES

- [1] Khodra, Masayu Leylia, Maulidevi, Nur Ulfa, Munir, dan Rinaldi. Algoritma Branch & Bound (Bagian 1, 2 dan 3). (2021).
- [2] Hidayanto, Sumahir. Kenali 9 Monster Jungle di Land of Down, Jangan Asal Serang!. (2020). Diakses pada 21.00 tanggal 21 Mei 2022 dari <https://www.idntimes.com/tech/games/sumahir-hidayanto/monster-jungle-di-land-of-down-c1c2/9>
- [3] Gaming, Woles .PENJELASAN MONSTER JUNGLE DAN BUFF DI MAP TERBARU MOBILE LEGEND !!(2020). Youtube. https://www.youtube.com/watch?v=WFsppHYJE0o&t=97s&ab_channel=WolesGaming
- [4] Map. Diakses pada 21.30 tanggal 21 Mei 2022 dari <https://mobile-legends.fandom.com/wiki/Map>

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

Timothy Stanley S (13520028)