Pengaplikasian Algoritma Backtracking untuk Menyelesaikan TSP untuk Menentukan Rute Wisata Kuliner di Daerah Bandung

Kristo Abdi Wiguna - 13520058
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13520058@std.stei.itb.ac.id

***Abstrak*—Mengingat Indonesia sebagai salah satu negara yang kaya akan kultur dan budaya, begitu juga dengan dunia makanan yang ada di Indonesia. Bandung sebagai salah satu kota yang diakui dunia yang memiliki variasi makanan yang khas dan kuliner sebagai daya tarik turis. Makalah membahas tentang algoritma runut-balik yang menyelesaikan persoalan *travelling salesman problem* dengan kompleksitas O(N!) dengan waktu eksekusi rata – rata sebesar 0.029 detik.**

***Kata kunci*—tsp, runut-balik, *backtracking*, rute, kuliner**

I. PENDAHULUAN

Bandung adalah kota yang sudah terkenal menjadi destinasi wisata untuk turis yang pergi ke Indonesia. Bandung terkenal dengan kulinernya, *fashion*, dan kota pelajar yang memiliki banyak surga tempat kuliner, tempat baju atau pakaian distro, pakaian modern, dan menampung banyak mahasiswa yang berkuliah di banyak universitas ternama di Bandung.

Pada tahun 2019, Menteri Pariwisata Arief Yahya menetapkan Bali dan Bandung menjadi salah satu destinasi wisata kuliner yang menjadi unggulan di Indonesia. Karena portofolio bisnis pariwisata Indonesia, 60 persen orang datang karena faktor budaya dan mereka menggunakan 45 persen uangnya untuk kuliner.

Dilansir dari jabarprov.go.id bahwa pada tahun 2020, Bandung menduduki peringkat ketujuh di dunia untuk makanan tradisional setelah bersaing dengan kota Naples, Melbourne, Paris, Rome, Lisbon, Bangkok, Mumbai, Hong Kong, New Delhi, dan kota – kota terkenal lainnya. Bandung berhasil mendapat suara berjumlah 63.402 dan sejumlah kritikus restoran profesional berdasarkan survey yang diadakan oleh TasteAtlas Awards 2020. Hal ini menjadi bukti bahwa Bandung merupakan surga kuliner yang memiliki banyak tempat kulinter sebagai daya tarik pariwisata baik domestik ataupun mancanegara.

Gambar 1. Ilustrasi Tempat Kuliner di Bandung
(sumber : https://www.traveloka.com/id-id/restaurants/indonesia/detail/dailycious-sudirman-street-day-night-market-88451)

Oleh karena banyak tempat kuliner yang ada di daerah Bandung, dibutuhkan rute perjalanan yang baik untuk dapat mengunjungi berbagai tempat tersebut agar efisien secara waktu ataupun biaya sehingga semua tempat kuliner dapat dikunjungi oleh seseorang yang berwisata di Bandung.

Persoalan ini dapat ditranslasikan dengan permasalahan Travelling Salesman Problem, yaitu mencari jalur terpendek untuk mengunjungi tiap titik yang ada di sebuah graf lalu kembali tempat semula. Dalam konteks permasalahan terkait tempat kuliner di Bandung, titik merepresentasikan destinasi wisata kuliner dan tempat penginapan menjadi titik awal. Dalam memecahkan persoalan ini, banyak pendekatan yang dapat dilakukan, salah satunya adalah menggunakan algoritma Backtracking.

II. TEORI DASAR

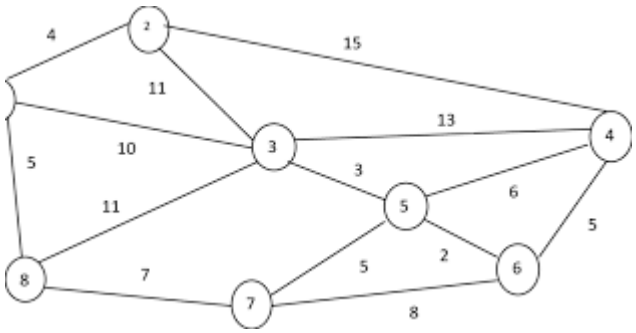
2.1 Travelling Salesman Problem (TSP)

Travelling Salesman Problem adalah permasalahan yang sudah ada sejak lama di dunia algoritma dan pemrograman. Pada permasalahan ini, ada sebuah kota awal dan sejumlah n kota

yang ingin dikunjungi. Salesman diminta untuk memulai perjalanan dari kota A sebagai kota awal ke seluruh kota yang harus dikunjung sebanyak n tepat satu kali. Perjalanan berawal dan berakhir ke kota A misalnya sebagai kota awal dan tidak boleh kembali ke kota awal sebelum semua kota tujuan dikunjungi. Tujuan dari persoalan ini adalah mencari total jarak paling minimum yang dapat ditempuh salesman dengan mengatur urutan kota yang harus dikunjungi.

Persoalan TSP yang merupakan persoalan NP, berbeda dengan Lintasan Hamilton yang seringkali disama-artikan. Lintasan Hamilton menyatakan bahwa ada lintasan yang melalui tiap simpul pada graf tepat satu kali. Karena berbagai jenis lintasan hamilton ada dalam suatu graf maka, TSP mencari lintasan Hamilton dengan berat paling minimum.

Seringkali persoalan TSP dapat menjadi cukup rumit jika jumlah kota yang harus dikunjungi cukup banyak. Algoritma yang dapat digunakan sebagai pendekatan untuk pemecahan persoalan TSP ada banyak, misalnya Bruteforce, Backtracking, Genetic Algorithm, Greedy, Program Dinamis, dan Branch and Bound.



Gambar 2. Ilustrasi Persoalan *Travelling Salesman Problem*
(sumber: https://www.researchgate.net/figure/Traveling-Salesman-Problem_fig3_301325477)

2.2 Algoritma Backtracking

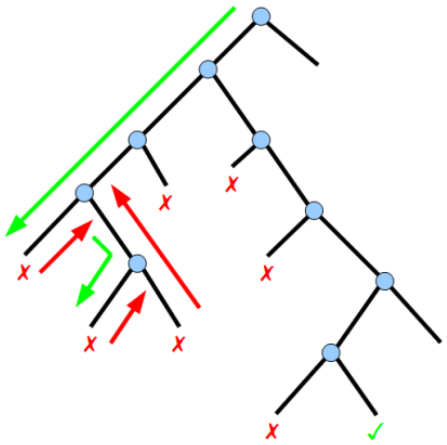
Backtracking dapat dipandang sebagai salah satu dari dua hal yaitu pertama, sebagai sebuah fase di dalam algoritma traversal DFS atau kedua sebagai sebuah metode pemecahan masalah yang mangkus, terstruktur, dan sistematis, baik untuk persoalan optimasi ataupun non-optimasi.

Algoritma runut-balik atau *backtracking* merupakan perbaikan dari *exhaustive search*. Pada algoritma *backtracking*, hanya pilih pilihan yang mengarah ke solusi yang dieksplorasi, pilihan yang tidak mengarah ke solusi tidak dipertimbangkan lagi. Sehingga terdapat langkah *pruning* simpul yang tidak mengarah ke solusi.

Pada umumnya, algoritma runut-balik memiliki properti umum yaitu solusi persoalan, fungsi pembangkit, dan fungsi pembatas. Solusi persoalan adalah himpunan yang dinyatakan sebagai vektor dengan n-tuple sebagai vektor solusi persoalan. Fungsi pembangkit dinyatakan sebagai predikat T() membangkitkan nilai x_k yang merupakan komponen vektor solusi. Fungsi pembatas dinyatakan sebagai predikat B(x_1, x_2, …, x_k) dan bernilai true jika (x_1, x_2, …, x_k) mengarah ke solusi yaitu tidak melanggar *constraints*. Jika fungsi pembatas bernilai true, maka pembangkitan nilai untuk x_k +1 dilanjutkan, tetapi jika false, maka (x_1, x_2, …, x_k).

Proses pencarian solusi dalam algoritma runut-balik ini dapat diilustrasikan dengan *space state tree*. Solusi dicari dengan membangkitkan simpul – simpul status sehinga menghasilkan lintasan dari akar ke daun. Aturan pembangkitan simpul yang dipakai adalah mengikuti aturan *depth-first-order*. Beberapa jenis simpulnya yaitu, simpul hidup yang sudah dibangkitkan, simpul-E atau simpul hidup yang sedang dieksplorasi atau diperluas, dan jika simpul-E yang sedang membentuk lintasan karena tiap kali diperluas tidak mengarah ke solusi maka dimatikan sehingga menjadi simpul mati. Fungsi yang berguna untuk mematikan simpul-E adalah fungsi pembatas.

Pencarian pada algoritma runut-balik ditentukan dengan mencari kedalaman dengan membangkitkan simpul dan memperluasnya lalu jika berakhir dengan simpul mati, proses pencarian algoritma runut-balik melakukan *backtrack* ke simpul di atasnya lalu diteruskan de4ngan membangkitkan simpul anak yang lainnya menjadi simpul-E yang baru. Pencarian akan dihentikan bila telah sampai pada goal node.



Gambar 3. Ilustrasi Pohon Pembangkitan Simpul dan Lintasan pada Algoritma *Backtracking*
(sumber: https://www.w3.org/2011/Talks/01-14-steven-phenotype/)

2.3 Penggunaan Backtracking pada TSP

Penggunaan algoritma runut-balik atau *backtracking* hampir sama dengan penggunaan *branch and bound* pada TSP. Misalkan A adalah sebuah array untuk n kota dan l adalah parameter sebuah tempat kota, lengthSoFar adalah variabel penyimpan panjang dari kota A[1], A[2], …, hingga A[l], minCost adalah variabel penyimpan jarak minimum dari penelusuran tiap kota yang ada di array A. Pada penggunaannya, variabel minCost, lengthSoFar, dan l akan terus diperbarui isinya dengan menggunakan pemanggilan rekursif fungsi *backtracking*.

Pseudocode penggunaan algoritma runut-balik atau *backtracking* pada persoalan TSP adalah sebagai berikut.

```
Algorithm TSP_Backtrack(A, ℓ, lengthSoFar, minCost)
1.  n ← length[A]    // number of elements in the array A
2.  if ℓ = n
3.    then minCost ← min(minCost, lengthSoFar + distance[A[n], A[1]])
4.    else for i ← ℓ + 1 to n
5.           do Swap A[ℓ + 1] and A[i]    // select A[i] as the next city
6.              newLength ← lengthSoFar + distance[A[ℓ], A[ℓ + 1]]
7.              if newLength ⩾ minCost    // this will never be a better solution
8.                then skip    // prune
9.                else minCost ←
10.                    min(minCost, TSP_Backtrack(A, ℓ + 1, newLength, minCost))
11.             Swap A[ℓ + 1] and A[i]    // undo the selection
12. return minCost
```

Gambar 4. Pseudocode Penggunaan Algoritma *Backtracking* pada persoalan TSP
(sumber: https://www.win.tue.nl/~kbuchin/teaching/2IL15/backtracking.pdf)

Berdasarkan line 2, dilihat bahwa baris tersebut adalah kondisi dimana semua kota telah ditelusuri. Pada line 7 juga dapat dilihat adalah fungsi pembatas atau *bounding function* yang berguna untuk mem-*pruning* atau membatasi simpul yang dibangkitkan yang tidak mengarah ke solusi yaitu jika jarak yang baru lebih dari minimal cost yang telah dibuat sehingga tidak mengarah ke solusi karena algoritma ingin mencari jarak paling minimum maka rekursif tidak dipanggil dan melewati step tersebut.

Selain itu, dalam algoritma ini, variabel minCost dibuat sebagai batas atas dari total jarak minimum yang dimiliki oleh rute yang terbaik yang sedang ditelusuri. Setiap kali ditemukan rute yang lebih baik, batas atas ini diperbarui sehingga mencari jarak total minimum rute penelusuran.

Fungsi pembatas pada algoritma di atas, hanya memiliki kompleksitas O(1) karena hanya mengecek perbandingan dua variabel yaitu minCost dan newLength.

Fungsi algoritma *backtracking* pada pseudocode tersebut juga dapat dilihat bahwa menelusuri pembangkitan simpul secara DFS. Simpul hidup yang dibangkitkan dengan memanggil rekursif TSP_Backtrack(A, l + 1, newLength, minCost) pada line 10 dengan parameter yang telah diperbarui yaitu l + 1 untuk menambah jumlah kota yang ditelusuri sehingga menjadi simpul ekspan.

Ketika pruning dilakukan maka simpul tidak akan dibangkitkan sehingga program akan backtrack ke kota selanjutnya untuk menelusuri terjadi pada line 7.

2.4 Destinasi Wisata Kuliner di Daerah Bandung

Menurut pencarian penulis pada internet, terdapat beberapa destinasi wisata kuliner yang memiliki daya tarik kuat dan menarik turis – turis untuk berkunjung karena rasa makanan yang lezat, suasana wisata kuliner yang nyaman untuk dikunjungi sehingga ramai, dan sebagainya. Selain itu, penulis menyisipkan hotel terbaik di tengah kota Bandung sebagai referensi dimana turis akan menginap sebagai simpul pertama.

Beberapa destinasi wisata kuliner terpopuler di daerah Bandung Raya yaitu :

1. Kuliner Malam Cibadak
2. Paskal Food Market
3. Pasar Cisangkuy
4. Kawasan Punclut
5. Baso Cuanki Serayu
6. Iga Bakar si Jangkung
7. Kawasan Dipatiukur
8. Kampung Daun
9. Gubung Makan Mang Engking
10. Dusun Bambu

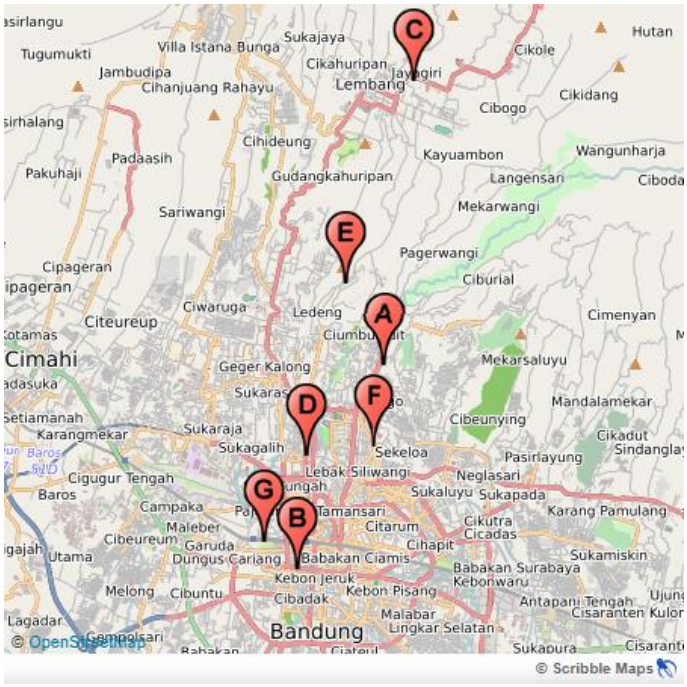
III. PEMBAHASAN

3.1 Persiapan dan Pengambilan Data

Karena ada limitasi waktu untuk mengunjungi semua destinasi kuliner dalam satu hari serta banyaknya tempat yang dikunjungi, maka destinasi yang penulis gunakan pada makalah ini adalah :

1. Sheraton Hotel (titik awal)
2. Kawasan Cibadak
3. Gubug Makan Mang Engking
4. Iga Bakar Si Jangkung
5. Kawasan Punclut
6. Kawasan Dipatiukur
7. Paskal Food Market

Sheraton Hotel dipilih sebagai titik awal karena merupakan hotel yang sangat populer memiliki 5172 ulasan berdasarkan *Google* serta merupakan hotel dengan merk ternama yang berada di Jl. Ir. H. Juanda No.390 Dago.



Gambar 5. Letak titik – titik destinasi kuliner di Bandung
(sumber: milik sendiri, dibuat dengan menggunakan aplikasi

online
scribblemaps (https://www.scribblemaps.com/) pada tanggal 20 Mei 2022 pukul
15.32 WIB)

Keterangan gambar peta di atas :

1. Sheraton Hotel (titik awal)
2. Kawasan Cibadak
3. Gubug Makan Mang Engking
4. Iga Bakar Si Jangkung
5. Kawasan Punclut
6. Kawasan Dipatiukur
7. Paskal Food Market

Tempat – tempat destinasi kuliner tersebut perlu dihubungkan dengan matriks ketetanggaan atau *adjacency matrix*. Matriks ketetanggaan berisi jarak dari satu tempat menuju tempat lain sebagai berikut :

```
[0    7.2   10.2   4.8   4.3    3.2    7.3 ]
[6.8   0    16.9   4.3   11.3   5.2    1.9 ]
[9.9  18.6   0     13.8  5.9    12.4   18.7]
[4.2   5    14.3    0    9.0    2.9    4.9 ]
[4.3   11   5.6    7.1    0     6.6    11.7 ]
[2.5  5.2   12.1   3.8   6.6     0     6 ]
[7.5  2.2   16.3   4.1   11.6   5.5    0 ]
```

Gambar 6. Matriks ketetanggaan destinasi kuliner di Bandung
(sumber: milik penulis)

Dengan urutan titik dari secara baris dari atas ke bawah dan secara kolom dari kiri ke kanan sebagai berikut, Sheraton (0), Kawasan Cibadak (1), Gubug Makan Mang Engking (2), Iga Bakar Si Jangkung (3), Kawasan Punclut (4), Kawasan Dipatiukur (5), dan Paskal Food Market (6). Data jarak antar tempat yang ada pada matriks dicari menggunakan alat bantu *maps.google.com*.

3.2 Eksperimen dalam Bahasa C++

Berikut merupakan matriks ketetanggaan yang dibuat dalam program C++ berbentuk array dua dimensi :

```
float adj_matrix[][V] = {
    {0, 7.2, 10.2, 4.8, 4.3, 3.2, 7.3},
    {6.8, 0, 16.9, 4.3, 11.3, 5.2, 1.9},
    {9.9, 18.6, 0, 13.8, 5.9, 12.4, 18.7},
    {4.2, 5, 14.3, 0, 9, 2.9, 4.9},
    {4.3, 11, 5.6, 7.1, 0, 6.6, 11.7},
    {2.5, 5.2, 12.1, 3.8, 6.6, 0, 6},
    {7.5, 2.2, 16.3, 4.1, 11.6, 5.5, 0}
};
```

Gambar 7. Matriks ketetanggaan array dua dimensi di program C++
(sumber: milik penulis)

Program memiliki fungsi main yang menginisialisasi ukuran dan matriks ketetanggaan, variabel penyimpan jarak total minimal, dan variabel tempat mulai penelusuran. Fungsi main juga menginisialisasi array berukuran n-kota yang diinisialisasi tiap sel nya berisi nilai indeksnya. Array ini nantinya akan digunakan untuk menyimpan rute yang telah berhasil dibuat.

```
int main() {
    int n = 7;

    float adj_matrix[][V] = {
        {0, 7.2, 10.2, 4.8, 4.3, 3.2, 7.3},
        {6.8, 0, 16.9, 4.3, 11.3, 5.2, 1.9},
        {9.9, 18.6, 0, 13.8, 5.9, 12.4, 18.7},
        {4.2, 5, 14.3, 0, 9, 2.9, 4.9},
        {4.3, 11, 5.6, 7.1, 0, 6.6, 11.7},
        {2.5, 5.2, 12.1, 3.8, 6.6, 0, 6},
        {7.5, 2.2, 16.3, 4.1, 11.6, 5.5, 0}
    };

    float current_cost = 0;

    vector<int> path(n);
    for (int i = 0; i < n; i++) {
        path[i] = i; // initialize path
    }

    float min_cost = MAX;
    int s = 1;

    tsp(adj_matrix, path, n, s, current_cost, min_cost);
}
```

Gambar 8. Fungsi main program C++
(sumber: milik penulis)

Selain fungsi main, program C++ yang penulis buat juga memiliki fungsi tsp yaitu fungsi yang melakukan algoritma runut-balik atau *backtracking* pada matriks ketetanggaan. Fungsi tsp adalah sebagai berikut :

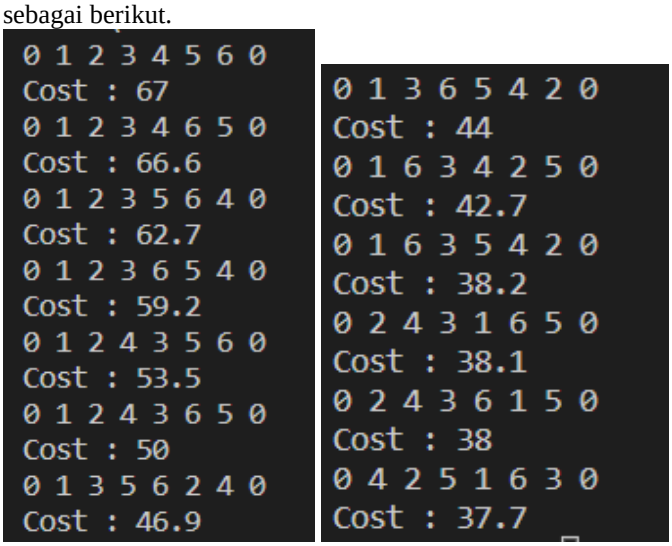
```
void tsp(float adj_matrix[][V], vector<int> path, int n, int& s, float &current_cost, float &min_cost) {
    if (s == n) {
        float x =  adj_matrix[path[n-1]][path[0]];
        if (x != 0 && (current_cost + x < min_cost)) {
            path.push_back(path[0]);
            for (auto it  = path.begin(); it != path.end(); it++) {
                cout << *it << " ";
            }
            cout << endl;
            min_cost = current_cost + x;
            cout << "Cost : " << min_cost << endl;
        }
    } else {
        for (int i = s; i < n; i++) {
            float x = adj_matrix[path[i-1]][path[i]];
            if (x != 0 && (current_cost + x < min_cost)) {
                swap(path[i], path[s]);
                current_cost += adj_matrix[path[s-1]][path[s]];
                int stemp = s+1;
                tsp(adj_matrix, path, n, stemp, current_cost, min_cost);
                current_cost -= adj_matrix[path[s-1]][path[s]];

                swap(path[i], path[s]);
            }
        }
    }
    // waktu yang dibutuhkan
};
```

Gambar 9. Fungsi implementasi algoritma runut-balik program C++
(sumber: milik penulis)

Fungsi yang diimplementasikan sedikit berbeda dari pseudocode yang telah disisipkan pada bagian di atas untuk menyesuaikan implementasi dan sintaks program C++ serta keluaran yang akan diberikan. Fungsi menerima matriks dua dimensi bertipe data float, array satu dimensi yang menyimpen rute terbaik, variabel n jumlah tempat yang dikunjungi, current_cost yaitu variabel penyimpan jarak penelusuran, dan

min_cost yaitu variabel penyimpan jarak total minimum rute terbaik.

Hasil dari program yang dibuat penulis untuk mencari rute terbaik menelusuri tempat kuliner di kota Bandung adalah sebagai berikut.



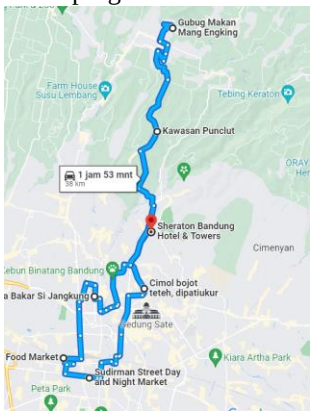
Gambar 10. Hasil Penelusuran Rute Terbaik Wisata Kuliner di Bandung
(sumber: milik penulis)

Dengan hasil terakhir sebagai hasil dengan *cost* atau jarak total paling minimum yaitu 37,7 kilometer dengan rute yaitu 0 menuju 4 menuju 2 menuju 5 menuju 1 menuju 6 menuju 3 menuju 0. Jika rute diterjemahkan sebagai tempat yang telah ditentukan, maka akan sebagai berikut, Sheraton Hotel menuju Kawasan Punclut lalu menuju Gubug Makan Mang Engking lalu menuju Kawasan Dipatiukur lalu menuju Kawasan Cibadak lalu menuju Paskal Food Market lalu menuju Iga Bakar Si Jangkung lalu kembali ke titik awal yaitu Sheraton Hotel.

Jika melihat matriks ketetanggaan jarak di awal, kita juga dapat menjumlahkan total jarak tersebut yaitu 4.3+5.6+12.4+5.2+1.9+4.1+4.2=37.7 kilometer.

Program juga memiliki keluaran rute yang lain yang memiliki cost yang masih lebih besar jika dibandingkan dengan solusi jarak total paling minimum. Rute – rute inilah adalah rute yang berhasil melewati pruning dan mengarah ke solusi karena rute yang lain yang tidak ada dalam keluaran berarti bahwa rute tersebut dibatasi atau di-*pruning*.

Berikut ilustrasi rute yang terbentuk sesuai dengan hasil akhir pengujian menggunakan program C++



Gambar 11. Ilustrasi rute
(sumber: https://www.maps.google.com)

3.3. Pembahasan dan Analisis Kompleksitas

Kompleksitas waktu algoritma program yang dibuat penulis adalah O(N!) dengan N merupakan jumlah tempat yang dikunjungi. Kompleksitas *space* algoritma program adalah O(N) yaitu array berukuran N yang menyimpan rute terbaik yang di-*generate* oleh program.

Jika dibandingkan dengan program algoritma program dinamis yang menyelesaikan persoalan TSP memiliki kompleksitas algoritma O(N^2 * 2^N) yang lebih efisien dibanding O(N!) tetapi masih eksponensial dan cenderung lebih kompleks untuk ukuran N yang lebih tinggi.

Penulis menguji waktu eksekusi program dengan algoritma runut-balik untuk menyelesaikan persoalan TSP sebanyak 10 kali sebagai berikut.

No	Waktu eksekusi
1	26,466 milisekon
2	24,369 milisekon
3	35,059 milisekon
4	32,285 milisekon
5	27,047 milisekon
6	33,431 milisekon
7	31,467 milisekon
8	30,093 milisekon
9	23,016 milisekon
10	28,348 milisekon

Rata – rata waktu yang dibutuhkan program penulis untuk menyelesaikan persoalan TSP menggunakan algoritma runut-balik atau *backtracking* yaitu 0.029 detik atau 29,158 milidetik.

IV. KESIMPULAN

Algoritma *Backtracking* menggunakan pencarian seperti *Depth First Search* (DFS) untuk menyelesaikan persoalan Travelling Salesman Problem atau TSP yang dapat dimanfaatkan untuk mencari rute wisata kuliner di daerah Bandung.

Walaupun dalam makalah ini kasus tur yang diambil adalah di daerah Bandung, tidak menutup kemungkinan bahwa algoritma runut-balik dapat menyelesaikan persoalan serupa lainnya di tempat lain.

Algoritma ini cukuip optimal untuk menyelesaikan sebuah persoalan TSP selama jumlah n-kota masukkan memliki jumlah yang terbatas.

VIDEO LINK YOUTUBE

Video yang berkaitan dapat dilihat pada link berikut : https://youtu.be/dOqeuU9UXd8

V. UCAPAN TERIMA KASIH

Penulis ingin mengucapkan rasa syukur sebesar – besarnya kepada Tuhan Yang Maha Esa karena telah memberikan kesempatan pada penulis untuk menyelesaikan makalah ini.

Penulis ingin menyampaikan rasa terima kasih yang sebesar – besarnya kepada dosen Masayu Leila Khodra karena telah membimbing penulis selama menempuh kuliah IF2211 – Strategi Algoritma pada prodi Informatika, fakultas Sekolah Teknik Elektro dan Informatika di Institut Teknologi Bandung.

Penulis juga ingin mengucapkan terima kasih kepada penulis makalah atau artikel tempat penulis mendapatkan referensi.

Semoga apapun yang ditulis di makalah ini dan yang dipelajari dapat membantu orang lain dalam menyelesaikan tugas – tugasnya, dan mudah – mudahan menambah wawasan para pembaca.

REFERENCES

[1] Levitin, Anany. 2012. Introduction to The Design and Analysis of Algorithms. New Jersey: Pearson Education, Inc.
[2] Munir, Rinaldi. 2007. Diktat Kuliah Strategi Algoritma. Bandung: Prodi Informatika STEI ITB.
[3] https://www.win.tue.nl/~kbuchin/teaching/2IL15/backtracking.pdf (Diakses 15 Mei 2022)
[4] https://dosenit.com/kuliah-it/pemrograman/algoritma-backtracking (Diakses 19 Mei 2022)
[5] https://www.geeksforgeeks.org/backtracking-introduction/ (Diakses 20 Desember 2022)
[6] https://www.geeksforgeeks.org/travelling-salesman-problem-implementation-using-backtracking/ (Diakses 20 Mei 2022)

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, 22 Mei 2022

Kristo Abdi Wiguna
13520058