Penerapan Algoritma *Branch and Bound* dalam Pengiriman Barang Menggunakan Drone

Raihan Luthfi Haryawan / 13517016
Program Studi Teknik Informatika
*Sekolah Teknik Elektro dan Informatik*a
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517016@std.stei.itb.ac.id

***Abstrack*—Seiring dengan berkembangnya zaman, perkembangan teknologipun juga tak akan terbendung. Teknologi-teknologi ini berkembang dengan tujuan untuk mempermudah kehidupan manusia. Salah satu dari perkembangan teknologi tersebut adalah pengiriman barang dengan menggunakan drone. Pada makalah ini akan diteorikan bagaimana algoritma *branch and bound* dapat memetakan jalur yang efisien dalam pengiriman barang menggunakan drone.**

Keywords—branch and bound, drone, jalur terpendek, pengiriman barang

I. Pendahuluan

Seiring dengan bertambahnya kesibukan manusia, kita membutuhkan layanan untuk dapat mengirimkan barang ke tempat yang kita inginkan. Drone adalah suatu pesawat tanpa pilot. Drone dikendalikan secara otomatis melalui program komputer yang dirancang, dalam kasus pada makalah ini, program dirancang dengan algoritma *branch and bound* dengan tujuan untuk mengantarkan barang-barang kepada rumah-rumah yang telah ditunjukkan oleh *GPS.* Pada makalah ini, drone yang digunakan adalah sebuah drone yang dapat mengantarkan beberapa barang sekaligus untuk sekali pengantaran.

Sebenarnya, masalah ini bukanlah masalah baru. Masalah ini juga dinalakan dengan *Traveling Salesman Problem.* Makalah ini bukan ditulis untuk memproposisikan penyelesaian masalah yang baru, melainkan untuk memberitahu bahwa pengiriman barang dengan drone dapat mengaplikasikan algoritma yang telah ada dengan solusi masalah yang telah ada.

II. Dasar Teori

A. Drone

Drone juga dikenal dengan sebutan *Unmanned Aerial Vehicle.* Awalnya, Drone merupakan pesawat yang dikendalikan secara jarak jauh, tetapi kini mulai banyak direrapkan sistem otomatis untuk mengendalikannya. Drone awalnya dikembnagkan untuk kepentingan militer.

Perkembangan teknologi membuat drone juga mulai banyak diterapkan untuk kebutuhan sipil, terutama di bidang bisnis, industri dan logistik. *Amazon* memulai persaingan industri ini melalui peluncuran layanan *Amazon Prime Air*. Pengangkutan barang menjadi lebih cepat, lebih praktis, minim human error, dan mampu menjangkau lokasi terpencil.

Pengadaan pesawat drone sendiri menjadi salah satu program yang akan diwujudkan oleh Presiden Ir. H. Joko Widodo sebagai salah alat untuk menjaga pertahanan, wilayah Indonesia.

Sesuai Rencana Strategis Kementerian Perindustrian 2015-2019 pada Program Penumbuhan Industri Unggulan Berbasis Teknologi Tinggi, salah satu tahapannya dilaksanakan melalui Penumbuhan Industri Berbasis Kedirgantaraan. Kegiatan pengembangan teknologi dan produksi drone sangat tepat sekali untuk dilakukan terkait rencana strategis tersebut, ditinjau dari aspek efisiensi pendanaan dan efektifitas pemanfaatannya.

Di negara-negara maju, teknologi ini menjadi industri bagi end-user dalam waktu dekat. Pasar komersial drone telah siap diramaikan oleh para raksasa teknologi. Menurut Teal Group, perusahaan riset di bidang aerospace di Amerika Serikat, pasar drone di bidang militer dan sipil di dunia diperkirakan mencapai $89 miliar untuk satu dekade ke depan.

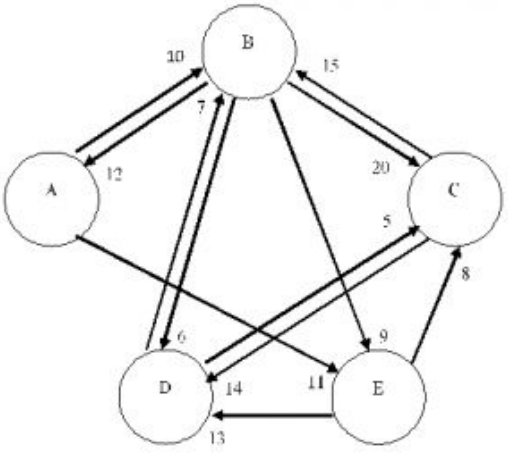
Di bulan April 2014, Google memulai memfokuskan diri dalam pengembangan industri drone. Google telah mengakuisisi produsen drone Titan Aerospace. Titan telah menguji berbagai aplikasi drone, termasuk di bidang pengiriman data, pemantauan sawah, serta bantuan pencarian dan penyelamatan. Drone keluaran Titan Aeorospace dapat mengudara hingga mencapai lima tahun tanpa perlu mendarat atau mengisi ulang bahan bakar. Kemampuan produk drone Titan Aerospace ini tepat dengan misi Google untuk menyebarkan layanan internet ke berbagai wilayah.

Teknologi drone sangat menarik perusahaan-perusahaan raksasa tersebut. Drone dapat meningkatkan produktivitas, menekan biaya operasional, dan meminimumkan resiko kecelakaan. Selain itu, drone juga membuka peluang bisnis baru yang bisa saja baru teridentifikasi di masa depan, seperti pengantaran barang.

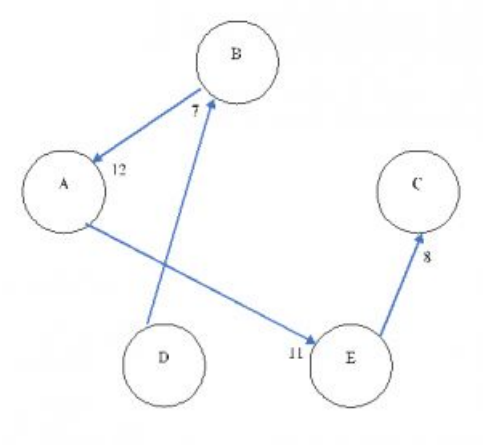
Bukan hanya pengiriman barang, drone juga dapat diterapkan dalam pengawasan infrastruktur fisik, pemadaman kebakaran hutan, dan eksplorasi lokasi tambang minyak/mineral.

B. *Algoritma Branch and Bound*

Algoritma *Branch and Bound* merupakan salah satu algoritma penelusuran jalur. Algoritma ini melakukan perhitungan suatu proses bernama *Branching,* yaitu perhitungan secara rekursif untuk memecah masalah kedalam masalah-masalah kecil, sambil tetap menghitung nilai terendah / terbaik. Selain *Branching,* untuk meningkatkan performa, algoritma ini akan melakukan pencatatan biaya minimum sebagai *Bound* dalam setiap perhitungan, sehingga untuk calon hasil jawaban yang diperkirakan yang melebihi bound akan dibuang karena tidak mungkin akan mencapai nilai terbaik



Gambar 2.1. Graph masukan.



Gambar 2.2. Jalur terpendek dari graph yang telah dicari dengan algoritma *Branch and Bound.*

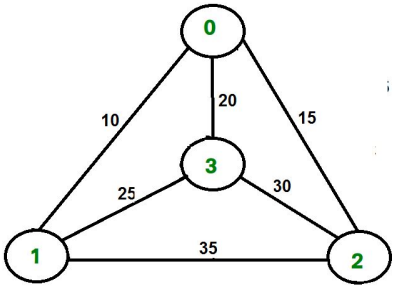
sumber :
https://piptools.net/algoritma-bb-branch-and-bound/

C. *Traveliing Salesman Problem*

Seperti yang telah ditulis di bagian awal tadi, masalah pengiriman barang dengan drone merupakan sebuah masalah yang disebut TSP (*Travelling Salesman Problem)*. TSP adalah suatu permasalahan di mana seseorang harus mengunjungi semua lokasi dimana tiap lokasi hanya dikunjungi sekali, dan dia harus mulai dari dan kembali ke lokasi asal. Tujuannya adalah menentukan rute dengan jarak total atau biaya yang paling minimum

III. Perancangan Algoritma

Langkah awal yang dilakukan dalam perancangan algoritma *Branch and Bound* adalah dengan membuat graf dari pemetaan lokasi pengiriman sebagai *node* dan jarak antar lokasi sebagai *edge* sebuah graf. Untuk setiap lokasi, harus mempunya jarak terhadap lokasi yang lain. Graf yang dibuat merupakan *undirected graph.*



Gambar 3.1. Contoh pemetaan graf.

sumber :
https://www.geeksforgeeks.org/traveling-salesman-problem-using-branch-and-bound-2/

Untuk setiap jalur yang dilewati, kita cari *cost* dari setiap pasangan *node.* Pencarian *cost tour* atau bobot tur dilakukan dengan rumus :

Bobot tur T : ½ (Jumlah bobot dari dua sisi yang berhubungan dengan u yang berada di tur T)

Di mana u ∈ V dan V adalah simpul yang ada pada graf.

A. Penyelesaian dengan Algoritma Branch and Bound

Dari gambar 3.1., misalkan *node* awal adalah 0, dengan *Brnach and Bound* untuk mencari bobot tur lengkap, didapatkan :

Node	Sisi dengan cost paling sedikit	Total cost
0	(0, 1), (0, 2)	25
1	(0, 1), (1, 3)	35
2	(0, 2), (2, 3)	45
3	(0, 3), (1, 3)	45

Bobot tur lengkap = 1/2(25 + 35 + 45 + 45) = 75

Bobot tur lengkap yang didapatkan merupakan fungsi pembatas *(bound)* untuk menghitung *cost* setiap simpul dari pohon.

Cost :

0 : 75

Path (0,i2) :

i2 = 1 :

cost :½ ((10+15)+(10+25)+(10+35)+(20+25)) = 75

i2 = 2 :

cost :½ ((10+15)+(10+25)+(10+35)+(20+25)) = 75

i3 = 3 :

cost :½ ((10+20)+(10+25)+(10+35)+(20+25)) = 77,5

Path(0,3) tidak digunakan karena *cost* lebih dari 75.

Path(0,1,i3) :

i3 = 2:

cost : ½ ((10+15)+(10+35)+(15+35)+(20+25)) = 82,5

i3 = 3:

cost : ½ ((10+20)+(10+25)+(10+35)+(20+25)) = 77,5

Path(0,1,2) tidak digunakan karena *cost* lebih dari 77,5.

Maka, *path* yang ditelusuri adalah 0 -> 1 -> 3 -> 2 -> 0.

Minimum cost yang didapatkan pada jalur tersebut adalah :

10 + 25 +30 + 15 = 80

B. Penyelesaian dengan program Java

```
import java.util.*;

class BNB
{

    static int N = 4;
    static int final_path[] = new int[N +
1];
    static boolean visited[] = new
boolean[N];
    static int final_res =
Integer.MAX_VALUE;
    static void copyToFinal(int
curr_path[])
    {
        for (int i = 0; i < N; i++)
            final_path[i] = curr_path[i];
        final_path[N] = curr_path[0];
    }
    static int firstMin(int adj[][], int i)
    {
        int min = Integer.MAX_VALUE;
        for (int k = 0; k < N; k++)
            if (adj[i][k] < min && i != k)
                min = adj[i][k];
        return min;
    }
    static int secondMin(int adj[][], int
i)
    {
        int first = Integer.MAX_VALUE,
second = Integer.MAX_VALUE;
        for (int j=0; j<N; j++)
        {
            if (i == j)
                continue;

            if (adj[i][j] <= first)
            {
                second = first;
                first = adj[i][j];
            }
```

```
            else if (adj[i][j] <= second &&
                    adj[i][j] != first)
                second = adj[i][j];
        }
        return second;
    }
    static void TSPRec(int adj[][], int
curr_bound, int curr_weight,
                int level, int curr_path[])
    {
        if (level == N)
        {
            if (adj[curr_path[level -
1]][curr_path[0]] != 0)
            {
                int curr_res = curr_weight
+
adj[curr_path[level-1]][curr_path[0]];
                if (curr_res < final_res)
                {
                    copyToFinal(curr_path);
                    final_res = curr_res;
                }
            }
            return;
        }
        for (int i = 0; i < N; i++)
        {
            if (adj[curr_path[level-1]][i]
!= 0 &&
                    visited[i] == false)
            {
                int temp = curr_bound;
                curr_weight +=
adj[curr_path[level - 1]][i];
                if (level==1)
                curr_bound -=
((firstMin(adj, curr_path[level - 1]) +

firstMin(adj, i))/2);
                else
                curr_bound -=
((secondMin(adj, curr_path[level - 1]) +

firstMin(adj, i))/2);
                if (curr_bound +
curr_weight < final_res)
                {
                    curr_path[level] = i;
                    visited[i] = true;
                    TSPRec(adj, curr_bound,
curr_weight, level + 1,
                            curr_path);
                }
                curr_weight -=
adj[curr_path[level-1]][i];
                curr_bound = temp;
                Arrays.fill(visited,false);
                for (int j = 0; j <= level
- 1; j++)
                    visited[curr_path[j]] =
true;
            }
        }
    }
    static void TSP(int adj[][])
    {
        int curr_path[] = new int[N + 1];
        int curr_bound = 0;
        Arrays.fill(curr_path, -1);
        Arrays.fill(visited, false);
        for (int i = 0; i < N; i++)
            curr_bound += (firstMin(adj, i)
+
                        secondMin(adj, i));
        curr_bound = (curr_bound==1)?
curr_bound/2 + 1 :

curr_bound/2;
        visited[0] = true;
        curr_path[0] = 0;
        TSPRec(adj, curr_bound, 0, 1,
curr_path);
    }
 }
```

Class BNB tersebut dijalankan dengan driver :

```
public static void main(String[] args)
    {
        int adj[][] = {{0, 10, 15, 20},
                       {10, 0, 35, 25},
                       {15, 35, 0, 30},
                       {20, 25, 30, 0} };

        TSP(adj);

        System.out.printf("Minimum cost :
%d\n", final_res);
        System.out.printf("Path Taken : ");
        for (int i = 0; i <= N; i++)
        {
            System.out.printf("%d ",
final_path[i]);
        }
    }
```

Sumber program :
https://www.geeksforgeeks.org/traveling-salesman-problem-using-branch-and-bound-2/

Program akan menghasilkan keluaran :

Minimum cost : 80
Path Taken : 0 1 3 2 0

IV. KESIMPULAN

Dengan menggunakan drone yang diprogram dengan algoritma *Branch and Bound* dan pendekatan masalah *Travelling Salesman Problem,* kita dapat menemukan jarak terpendek dari suatu sirkuit yang melewati seluruh simpul untuk mengantarkan barang dari suatu lokasi ke lokasi lain sampai drone kembali ke lokasi awal.

REFERENSI

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/Makalah/MakalahStmik30.pdf

[3] https://piptools.net/algoritma-bb-branch-and-bound/

[4] https://www.geeksforgeeks.org/traveling-salesman-problem-using-branch-and-bound-2/

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, 26 April 2019

Raihan Luthfi Haryawan
13517016