Perbandingan Algoritma *Branch and Bound*, *Greedy*, dan *Nearest Neighbor* dalam Menentukan Rute Perjalanan

Pencarian Rute Wisata Jawa Tengah yang Optimal

Stefanus Agus Haryono (13514097)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514097@std.stei.itb.ac.id

***Abstract*—Rekreasi merupakan salah satu kebutuhan manusia yang penting untuk dipenuhi. Salah satu cara berekreasi adalah dengan melakukan perjalanan wisata, menikmati pesona keindahan alam dan kota yang jarang kita lihat dalam kehidupan sehari –hari. Melakukan perjalanan ini tentu membutuhkan waktu dan biaya. Efisiensi dalam melakukan perjalanan merupakan hal yang harus diperhatikan. Kesalahan dalam penentuan perjalanan dapat menyebabkan hilangnya waktu yang berharga dan tambahan pengeluaran. Makalah ini akan membahas bagaimana perbandingan dari penerapan algoritma *Branch and Bound* (BnB), *Greedy*, dan *Nearest Neighbor* dalam menentukan rute perjalanan paling optimal. Pada makalah ini, rute perjalanan akan dicontohkan dalam perjalanan berkeliling kota - kota di Jawa Tengah dan DIY. Dengan adanya rute perjalanan yang optimal, diharapkan rekreasi wisata dapat dinikmati semaksimal mungkin.**

***Kata Kunci—Branch and Bound, Greedy, Nearest Neighbor,* efisien, rute perjalanan, wisata**

I. PENDAHULUAN

Rekreasi dan berwisata merupakan kebutuhan sekunder manusia. Setiap manusia pada suatu masa pada hidupnya pasti ingin berwisata mengunjungi suatu tempat yang jarang atau belum pernah ia kunjungi sebelumnya. Berwisata merupakan hal yang umum dilakukan sehingga muncullah agen – agen wisata yang menyediakan jasa persiapan perjalanan wisata. Jasa tersebut tentu mempermudah kita karena kita tidak lagi perlu memikirkan transportasi, penginapan, dan hal – hal lain yang harus kita urus di tempat tujuan kita nanti.

Namun, salah satu pengalaman buruk yang penulis alami ketika berwisata adalah kehabisan waktu hingga tidak sempat mengunjungi tempat yang sebetulnya merupakan salah satu tujuan utama penulis. Yang lebih membuat kecewa lagi adalah, hal ini disebabkan kesalahan dari pihak agen wisata yang melakukan kesalahan dalam mengatur rangkaian acara sehingga waktu habis dalam perjalanan bolak – balik yang tidak efektif. Dari pengalaman tersebut, penulis terdorong untuk mencari tahu cara untuk menemukan cara yang paling mudah dalam menemukan rute perjalanan terbaik agar tidak terjadi kesalahan yang sama seperti yang telah penulis rasakan.

Jawa Tengah dan DIY merupakan Provinsi di Indonesia yang cocok untuk dijadikan sebuah tempat wisata. Kedua tempat wisata tersebut memiliki banyak tempat wisata unik yang sulit ditemui di tempat lain seperti berbagai Candi, Malioboro, Lawang Sewu, Sam Poo Kong, dan tempat – tempat lain. Selain itu, biaya hidup di kedua Provinsi tersebut juga terdorong lebih murah dibandingkan dengan Provinsi lain sehingga biaya berwisata menjadi lebih murah. Faktor – faktor tersebut menjadikan Jawa Tengah dan DIY sebagai tujuan wisata baik oleh wisatawan lokal, maupun wisatawan mancanegara. Akan tetapi, wisatawan sering bingung dalam memilih rute perjalanan yang tepat karena belum ada panduan pasti.

Dalam makalah ini, penulis akan mencari tahu rute perjalanan berkeliling Jawa Tengah dan DIY yang paling optimal dengan menerapkan beberapa algoritma yaitu *Branch and Bound*, *Greedy*, dan *Nearest Neighbor*. Pada makalah ini, diasumsikan wisatawan memulai perjalanan dari kota Yogyakarta, berkeliling ke kota – kota lain, dan kembali lagi ke kota Yogyakarta. Biaya perjalanan pada makalah ini dihitung berdasarkan jarak perjalanan dengan asumsi bahwa semakin jauh jarak, semakin tinggi biaya perjalanannya.

II. DASAR TEORI

A. Algoritma *Greedy*

Algoritma *Greedy* merupakan algoritma yang sering digunakan dalam persoalan optimisasi. Prosedur umum dalam algoritma ini adalah dengan membuat pilihan optimum lokal pada setiap langkah. Prosedur ini dilakukan dengan harapan bahwa pengambilan langkah optimum lokal akan membawa kita menuju solusi optimum global (solusi terbaik).

Pendekatan *Greedy* dalam mencari rute perjalanan terbaik adalah sebagai berikut. Anggap kota – kota tujuan sebagai

himpunan simpul N dan jalan antar kota sebagai himpunan sisi E. Pendekatan *Greedy* akan memilih sisi paling pendek dari himpunan sisi E dengan syarat bahwa sisi yang dipilih tidak membuat sebuah siklus dan tidak membuat simpul yang terhubung dengan sisi tersebut memiliki derajat lebih dari 2. Langkah ini dilakukan terus menerus hingga didapat N buah sisi.

Pseudocode pencarian rute dengan *Greedy* adalah sebagai berikut:

```
function Greedy (input N: Himpunan Kota, E: Himpunan Sisi Terurut, a: Kota awal) → Himpunan Sisi Yang Diambil
Deklarasi
    S: Himpunan sisi yang diambil
    Counter: Mencatat sisi ke berapa yang di cek saat ini
Algoritma
    While size(S) !=size(N) {loop selama jumlah sisi < jumlah kota}
        if E[Counter] tidak membuat siklus dan tidak membuat simpul terhubung menjadi berderajat >2
            S.add (E[Counter])
        Else
            Counter++
        endif
    endwhile
    return S
```

Algoritma ini memiliki kompleksitas $O(n^2)$. Iterasi dilakukan paling banyak sejumlah jumlah sisi yang ada yaitu n(n-1) dengan n adalah jumlah kota.

B. Algoritma *Nearest Neighbor*

Algoritma ini menggunakan pendekatan yang hampir sama dengan algoritma *Greedy*. Prosedur pencarian rute pada algoritma ini adalah sebagai berikut. Anggap kota – kota tujuan sebagai himpunan simpul N dan jalan antar kota sebagai himpunan sisi E. Misalkan, kota kita saat ini adalah kota A. Carilah kota yang jaraknya paling dekat dari kota A dari himpunan kota N. Jarak antar kota dapat dilihat dari himpunan sisi E. Langkah ini dilakukan terus menerus hingga seluruh kota dikunjungi. Setelah seluruh kota dikunjungi, kembali ke kota awal.

Pseudocode pencarian rute dengan *Nearest Neighbor* adalah sebagai berikut:

```
function NearestNeighbor (input N: Himpunan Kota, E: Himpunan Sisi Terurut, A: Kota awal) → Himpunan Kota Terurut
Deklarasi
    K: Himpunan Kota yang Diambil
    B: Kota dengan jarak terpendek dari kota saat ini
    C: Kota tempat kita berada saat ini
Algoritma
    K.add(A)
    C=A
    for i=1 to size (N) do
        B=Kota Terdekat dari C dan belum ada di K
        K.add(B)
        C=B
    K.add(A)
    return K
```

Algoritma ini memiliki kompleksitas $O(n^2)$. Iterasi dilakukan untuk setiap kota yaitu sejumlah n kota. Untuk setiap kota, perlu dicari sisi yang paling pendek yang menghubungkan dengan kota yang belum dipilih, yaitu di antara n-1 sisi.

C. Algoritma *Branch and Bound*

Algoritma *Branch and Bound* (BnB) digunakan dalam persoalan optimisasi, yaitu meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batasan persoalan. Pada algoritma ini, setiap simpul diberi sebuah nilai *cost*. *Cost* ini merupakan nilai taksiran lintasan termurah ke simpul tujuan. Simpul yang akan diekspan pada algoritma ini tidak berdasarkan urutan, namun berdasarkan *cost*, yaitu *cost* terkecil untuk kasus minimasi dan sebaliknya.

Dengan menggunakan algoritma ini, ada dua cara untuk mencari rute terpendek. Cara pertama adalah dengan Matriks ongkos-tereduksi. Pada cara ini, setiap sisi digambarkan sebagai elemen matriks. Matriks kemudian diubah menjadi matriks tereduksi, yaitu matriks yang setiap kolom dan barisnya mengandung paling tidak satu buah angka nol. Batas yang digunakan pada cara ini adalah jumlah elemen pengurang dari semua baris dan kolom yang digunakan ketika mereduksi matriks.

Cara kedua adalah dengan Bobot Tur Lengkap. Pada cara ini, batas yang digunakan adalah bobot minimum tur lengkap yaitu ½ (jumlah dua buah sisi minimum dari setiap simpul). Simpul tujuan yang memiliki bobot paling kecil adalah simpul yang harus diambil pada langkah berikutnya. Hal ini dilakukan terus menerus hingga seluruh kota telah dikunjungi. Pada makalah ini, cara yang akan digunakan adalah Bobot Tur Lengkap. *Pseudocode* algoritma ini adalah sebagai berikut:

function BnB (**input** N: Himpunan Kota, E: Himpunan Sisi, a: Kota awal) → Himpunan Sisi Yang Diambil

Deklarasi

S: Himpunan kota yang diambil

B: Kota asal sekarang

M: Himpunan Bobot Tur Minimal yang Dihitung

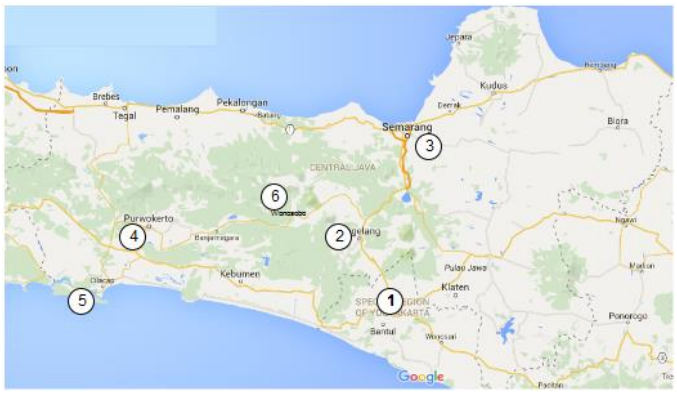
T: Kota Tujuan

Algoritma

While size(S)!=size(N) do
M.Initialize
T=M.HitungBobotMinimal(a) {Hitung dan cari kota dengan bobot minimal, masukkan ke variabel T}
S.add(T)
B=T
endwhile

III. Tujuan Wisata Jawa Tengah dan DIY

Jawa Tengah dan DIY memiliki beberapa kota tujuan wisata dengan keunikannya masing – masing. Berikut adalah gambar dan penjelasan dari beberapa kota tujuan wisata yang akan digunakan sebagai contoh pada makalah ini:



Keterangan:

1: Yogyakarta

2: Magelang

3: Semarang

4: Purwokerto

5: Cilacap

6: Wonosobo

Jarak dalam kilometer antar keenam kota tersebut adalah sebagai berikut (Jarak didapat dari Google Map):

Kota	1	2	3	4	5	6
1	0	45,4	129	176	186	96,1
2	45,4	0	80,7	143	171	57,3
3	129	80,7	0	203	237	117
4	176	143	203	0	51	90
5	186	171	237	51	0	124
6	96,1	57,3	117	90	124	0

1. Yogyakarta

Kota Yogyakarta merupakan bagian dari provinsi DIY. Kota ini memiliki beberapa tempat yang cocok sebagai tujuan wisata, antara lain:

- Jogja Bay Pirates Adventure Waterpark

  Sebuah *waterpark* terbesar, terlengkap, dan tercanggih di Indonesia. Tempat ini menyediakan wahana permainan air dan edukasi.

- De Mata Trick Eye Museum

  Sebuah museum yang berisi banyak gambar 3 dimensi. Museum ini berisi gambar – gambar yang dapat menipu mata Anda melalui ilusi optik 3 dimensi.

- Lava Tour Merapi

  Di tempat ini, Anda dapat menikmati rasanya menaiki mobil *offroad* untuk mengelilingi sebuah kawasan di sekitar gunung merapi

- Malioboro

  Sebuah jalan di kota Yogyakarta yang merupakan pusat wisata. Di tempat ini Anda dapat berbelanja, dan menikmati kuliner khas Yogyakarta.

2. Magelang

Magelang merupakan sebuah kota kecil di tengah - tengah Jawa Tengah. Beberapa tujuan wisata di kota ini adalah sebagai berikut:

- Candi Borobudur

  Candi yang terkenal hingga mancanegara ini berada di kota Magelang. Candi ini merupakan candi Budha terbesar yang ada di Indonesia dan telah diakui oleh UNESCO.

- Punthuk Setumbu

  Punthuk Setumbu adalah sebuah bukit yang berada di kabupaten Magelang. Di tempat ini, Anda dapat melihat matahari terbit yang sangat indah.

- Sungai Progo

  Sungai yang berada di Magelang ini dapat digunakan untuk melakukan arung jeram pada saat musim hujan. Tempat ini tentu cocok bagi penggemar olahraga air arung jeram.

3. Semarang

Semarang merupakan Ibukota Jawa Tengah. Tujuan wisata di Semarang antara lain:

- Lawang Sewu

  Bangunan kuno ini berada di pusat kota Semarang. Bangunan dengan gaya unik ini memiliki ratusan pintu dan

foto – foto bangunan tempo dulu, serta sebuah penjara bawah tanah.

- Klenteng Sam Po Kong

  Klenteng ini merupakan sebuah bangunan yang memiliki keindahan arsitektur Tiong Hoa. Di Klenteng ini juga terdapat Patung Cheng Ho setinggi 10 meter yang merupakan tertinggi di Asia Tenggara.

- Candi Gedong Songo

  Candi ini merupakan peninggalan budaya Hindu. Candi ini terbagi dalam beberapa lokasi yang tersebar di lereng Gunung Ungaran.

4. Purwokerto

   Purwokerto mungkin memang jarang didengar oleh orang. Namun, kota ini sebenarnya juga memiliki tujuan wisata:

- Baturraden

  Baturraden merupakan sebuah tempat wisata yang memiliki keindahan alam di dataran tinggi. Baturraden terletak di lereng gunung Slamet. Tempat ini memiliki berbagai fasilitas seperti kolam renang, sepeda air, hingga pemandian air panas.

- Pemandian Air Panas Pancuran 7

  Pemandian yang berada di dekat Baturraden ini merupakan sebuah pemandian air panas yang mengandung belerang.

- Telaga Sunyi

  Telaga ini merupakan area wisata alam dengan telaga yang alami. Di tempat ini Anda dapat melihat keindahan alam yang jarang ditemui.

5. Cilacap

   Cilacap merupakan daerah di Jawa Tengah yang berbatasan dengan Jawa Barat. Tempat wisata di daerah ini antara lain:

- Pulau Nusakambangan

  Pulau yang merupakan tempat perasingan narapidana ini dapat dicapai dengan menaiki kapal dalam waktu 10 menit dari Teluk Penyu. Di pulai ini, Anda dapat menikmati keindahan alam dan pantai.

- Teluk Penyu

  Teluk Penyu merupakan sebuah pantai yang menjorok ke Samudra Hindia. Pantai ini hanya berjarak 2 kilometer dari kota Cilacap.

- Kampung Laut

  Tempat ini cocok bagi pecinta kuliner laut. Tempat ini menyediakan kuliner laut, dan wisata hutan bakau. Kita juga dapat melihat kehidupan para nelayan di tempat ini.

6. Wonosobo

   Wonosobo merupakan sebuah kota yang terletak di dataran tinggi dan diapit leh Gunung Sindoro dan Gunung Sumbing. Tempat wisata di daerah ini:

- Telaga Warna

  Telaga ini dapat berubah warna dari biru laut, menjadi hijau hingga putih kekuningan. Telaga ini terletak di desa Dieng.

- Kawah Sikidang

  Kawasan Dieng merupakan sebuah gunung api yang masih cukup aktif. Salah satu tanda aktivitas vulkanik ini dapat dilihat di kawasan kawah Sikidang ini.

IV. Perbandingan Algoritma dalam Menentukan Rute Terbaik

A. *Greedy*

Penerapan algoritma *Greedy* pada kota – kota tujuan adalah sebagai berikut:

1. Ambil sisi Magelang – Yogyakarta (1-2)
2. Ambil sisi Purwokerto – Cilacap (4-5)
3. Ambil sisi Magelang – Wonosobo (2-6)
4. Sisi Magelang – Semarang (2-3) tidak diambil karena akan membuat simpul berderajat > 2
5. Ambil sisi Purwokerto – Wonosobo (4-6)
6. Sisi {(1-6),(3-6),(5-6)} tidak diambil karena akan membuat sebuah simpul berderajat > 2.
7. Ambil sisi Yogyakarta-Semarang (1-3)
8. Sisi {(2-4),(1-4),(1-5),(3-4)(2-5)} tidak diambil karena akan membuat simpul berderajat > 2.
9. Ambil sisi Semarang – Cilacap (3-5).

Jumlah pengecekan yang dilakukan dengan algoritma ini adalah sebanyak jumlah sisi yang ada yaitu 15. Rute yang didapat dengan algoritma ini adalah
Yogyakarta – Magelang – Wonosobo – Purwokerto – Cilacap – Semarang – Yogyakarta.
Total jarak yang didapat adalah: 45,4+57,3+90+51+237+129=609,7 km.
Perhitungan dengan rute dengan cara ini dapat dilakukan dengan cepat. Namun, pada umumnya pencarian rute dengan *Greedy* tidak memberikan rute yang paling optimal. Namun, jika tidak dibutuhkan hasil yang paling baik dan hanya diperlukan hasil yang cukup efektif, teknik *Greedy* cukup baik untuk diterapkan.

B. *Nearest Neighbor*

Penerapan algoritma ini pada list kota yang akan dikunjungi adalah sebagai berikut:

1. Kota asal: Yogyakarta
   Kota terdekat dilihat dari sisi: Magelang (1-2)
2. Kota asal: Magelang
   Kota terdekat dilihat dari sisi: Wonosobo (2-6)
   Yogyakarta tidak dipilih karena telah dipilih sebelumnya

3. Kota asal: Wonosobo
   Kota terdekat dilihat dari sisi: Purwokerto (6-4)
   Magelang tidak dipilih karena telah dipilih sebelumnya
4. Kota asal: Purwokerto
   Kota terdekat dilihat dari sisi: Cilacap(4-5)
5. Kota asal: Cilacap
   Kota terdekat dilihat dari sisi: Semarang(5-3)
6. Kembali ke kota awal (Yogyakarta) (3-1)
   Pencarian rute selesai.
   Jumlah pengecekan yang terjadi pada algoritma ini sama dengan pada algoritma *Greedy* yaitu 15 kali. Pada setiap simpul (kota) yang dikunjungi, kita perlu mengecek setiap sisi yang terhubung dan memilih sisi yang memenuhi syarat.
   Rute yang didapat adalah: Yogyakarta – Magelang – Wonosobo – Purwokerto – Cilacap – Semarang – Yogyakarta.
   Total jaraknya adalah:
   45,4+57,3+90+51+237+129=609,7
   Dapat dilihat dari hasil bahwa untuk kasus ini, pendekatan dengan algoritma *Nearest Neighbor* memberikan hasil yang sama dengan pendekatan *Greedy*. Dari kedua hasil yang sama tersebut, kita mungkin merasa bahwa hasil yang didapat merupakan hasil optimal yang sudah cukup baik. Namun, algoritma ini dapat memberikan hasil yang buruk jika kota yang terpilih belakangan berjarak sangat jauh dari kota asal (contohnya jika semakin lama kota semakin menjauh).

C. Algoritma *Branch and Bound*

Untuk mencari rute dengan BnB, pertama kita perlu menghitung *Cost* simpul akar.
Cost simpul akar= ½ [(45,4+96,1) + (45,4+57,3) + (80,7+117) + (51+90) + (51+124) + (57,3+90)] = 452,6.
Cost simpul akar ini merupakan batas bawah dari nilai rute yang bisa didapat. Pada umumnya, rute hasil yang didapat lebih besar dari *cost* simpul akar.
Langkah selanjutnya adalah memilih salah satu kota dari himpunan lima kota yang belum dipilih. Untuk setiap kota yang dipilih, sisi yang menghubungkan antar kota yang telah dipilih sebelumnya dengan kota yang telah akan dipilih harus dimasukkan ke dalam himpunan sisi dalam perhitungan *Cost.* Pemilihan kota tujuan pertama adalah sebagai berikut:
Yogyakarta – Magelang: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+117) + (51+90) + (51+124) + (57,3+90)] = 452,6.
Yogyakarta – Semarang: ½ [(45,4+129) + (45,4+57,3) + (80,7+129) + (51+90) + (51+124) + (57,3+90)] = 475,05.
Yogyakarta – Purwokerto: ½ [(45,4+176) + (45,4+57,3) + (80,7+117) + (51+176) + (51+124) + (57,3+90)] = 535,55.
Yogyakarta – Cilacap: ½ [(45,4+186) + (45,4+57,3) + (80,7+117) + (51+90) + (51+186) + (57,3+90)] = 528,55.
Yogyakarta – Wonosobo: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+117) + (51+90) + (51+124) + (57,3+90)] = 452,6.

Dari perhitungan *Cost* iterasi pertama, terdapat dua pasangan yang memenuhi yaitu Yogyakarta – Magelang dan Yogyakarta – Wonosobo.
Pada langkah berikutnya, kita memilih salah satu kota yang *Cost* nya *feasible*, misalnya Magelang. Pada iterasi kedua, kita menghitung kembali *Cost* dari setiap simpul tujuan yang tersisa 4 buah kota yang belum dikunjungi. Dalam iterasi kedua ini, perhitungan *Cost* harus memasukkan sisi Yogyakarta – Magelang. Jika *Cost* yang didapat tidak lebih baik dari *Cost* pada langkah sebelumnya, kita harus kembali dan mengecek pasangan dengan *Cost* yang lebih baik.
Iterasi kedua:
Magelang – Semarang: ½ [(45,4+96,1) + (45,4+80,7) + (80,7+117) + (51+90) + (51+124) + (57,3+90)] = 464,3.
Magelang – Purwokerto: ½ [(45,4+96,1) + (45,4+143) + (80,7+117) + (51+143) + (51+124) + (57,3+90)] = 521,95
Magelang – Cilacap: ½ [(45,4+96,1) + (45,4+171) + (80,7+117) + (51+90) + (51+171) + (57,3+90)] = 532,95.
Magelang – Wonosobo: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+117) + (51+90) + (51+124) + (57,3+90)] = 452,6.

Hasil dari iterasi kedua terlihat bahwa masih ada pasangan kota yang memenuhi syarat batas, yaitu Magelang – Wonosobo. Kota Wonosobo akan menjadi kota asal dalam iterasi berikutnya.
Iterasi ketiga:
Wonosobo – Semarang: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+117) + (51+90) + (51+124) + (57,3+90)] = 466,1.
Wonosobo – Purwokerto: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+117) + (51+90) + (51+124) + (57,3+90)] = 452,6.
Wonosobo – Cilacap: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+117) + (51+90) + (51+124) + (57,3+124)] = 469,6.

Iterasi ketiga juga memberikan pasangan kota yang *feasible* yaitu Wonosobo – Purwokerto. Kota asal pada iterasi berikutnya adalah Purwokerto.

Iterasi keempat:
Purwokerto – Semarang: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+203) + (203+90) + (51+124) + (57,3+90)] = 571,6
Purwokerto – Cilacap: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+117) + (51+90) + (51+124) + (57,3+90)] = 452,6.

Pada Iterasi keempat ini, pemilihan pasangan kota Purwokerto – Cilacap juga masih memenuhi syarat. Iterasi kelima dilakukan dengan Cilacap sebagai kota asal.

Iterasi kelima:
Cilacap – Semarang: ½ [(45,4+96,1) + (45,4+57,3) + (80,7+237) + (51+90) + (51+237) + (57,3+90)] = 569,1.

Namun, hasil iterasi kelima memberikan hasil yang tidak lebih baik dari hasil yang telah ditemukan sebelumnya. Jika ditemukan kasus seperti ini pada algoritma BnB, kita harus kembali ke simpul – simpul yang telah dibuat sebelumnya dan mencari alternatif rute lain. Dalam kasus

ini, kita kembali ke simpul yang saat ini memiliki nilai terbaik yaitu Yogyakarta – Wonosobo. Pembangkitan simpulpun diulang kembali terus menerus hingga didapat hasil akhir yang paling baik pada simpul daun. Paling baik di sini maksudnya adalah jika tidak ada simpul dengan *Cost* lebih baik dari *Cost* pada suatu simpul daun.

Karena penjelasan perhitungan dengan BnB dapat memakan banyak tempat dan waktu, untuk pemrosesan selanjutnya penulis menggunakan program yang telah dibuat. Dari program yang mengimplementasikan algoritma BnB tersebut, penulis mendapatkan salah satu rute yang paling optimal untuk dilalui yaitu:
Yogyakarta – Magelang – Semarang – Wonosobo – Purwokerto – Cilacap – Yogyakarta dengan total jarak 570.1.

D. Perbandingan hasil

Dari ketiga rute hasil yang didapat, dapat dilihat bahwa hasil perhitungan dengan Algoritma BnB lah yang paling optimal. Namun, biaya dan waktu perhitungan dengan BnB tidak murah dan tidak mudah dilakukan, terutama jika jumlah kota yang akan dikunjungi semakin banyak.
Perbedaan rute yang dihasilkan oleh BnB dan kedua algoritma lainnya juga tidak terlalu signifikan, yaitu 39,6 km. Besar perbedaan ini < 10% dari solusi optimal yang ada.
Berdasarkan hasil, untuk melakukan pembuatan rute yang cepat dan cukup optimal, dapat kita gunakan algoritma *Greedy* ataupun *Nearest Neighbor*. Namun, jika keefektifan adalah hal yang paling utama dan kita memiliki waktu yang cukup banyak, algoritma BnB lah cara yang tepat untuk mencari rute.
Perlu diingat juga bahwa pencarian rute yang dibahas pada makalah ini hanyalah berdasarkan jarak. Pada kehidupan nyata, terdapat faktor – faktor lain yang perlu kita perhatikan juga seperti cuaca, kondisi jalan, ketersediaan transportasi, waktu, serta faktor – faktor lain. Namun, dengan adanya rute yang cukup optimal, diharapkan rute perjalanan yang dilalui adalah rute yang cukup baik sehingga kegiatan berwisata dapat dinikmati semaksimal mungkin.

V. Kesimpulan

Berbagai algoritma dapat diterapkan dalam mencari rute yang paling efektif dalam berwisata. Perencanaan rute dengan menggunakan algoritma – algoritma tersebut diharapkan dapat memberikan rute yang efisien sehingga wisatawan tidak lagi mengeluarkan biaya dan waktu tambahan akibat kesalahan rute. Salah satu pengaplikasian ini adalah pembentukan rute perjalanan wisata sekitar Jawa Tengah dengan total jarak optimal 570,1 km.

Penggunaan algoritma *Greedy* dan *Nearest Neighbor* cocok digunakan jika kita ingin menghitung rute dengan cepat. Perhitungan dengan kedua algoritma ini dapat dilakukan dengan cepat tapi tidak selalu memberikan hasil yang paling baik. Penggunaan algoritma *Branch and Bound* cocok jika kita benar – benar membutuhkan rute yang paling baik dari seluruh rute yang ada. Namun perlu diingat bahwa penggunaan algoritma ini memerlukan perhitungan yang cukup lama dan tidak mudah.

VI. Ucapan Terima Kasih

Pertama penulis ingin mengucapkan rasa syukur kepada Tuhan Yang Maha Esa karena atas berkat dan pertolongan-Nya penulis dapat menyelesaikan makalah ini. Penulis juga berterima kasih kepada orang tua yang selalu mendukung dan menolong penulis. Penulis juga ingin mengucapkan terima kasih kepada Dosen pembimbing kuliah Strategi Algoritma yaitu, Dr. Ir. Rinaldi Munir, M.T. dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc atas ilmu, bimbingan, dan pengajaran selama kuliah sehingga makalah ini dapat penulis selesaikan.

Referensi

[1] https://www.google.com/maps/place/Central+Java,+Indonesia, diakses pada 7 Mei 2016
[2] Heuristicswiki.wikispaces.com, diakses pada 6 Mei 2016
[3] http://lcm.csa.iisc.ernet.in/dsa/node186.html, diakses pada 6 Mei 2016
[4] https://lemon.cs.elte.hu/pub/doc/latest/a00612.html, diakses pada 7 Mei 2016
[5] Munir, Rinaldi, Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Program Studi Teknik Informatika Intitut Teknologi Bandung, 2009.
[6] www.goindonesia.com, diakses pada 7 Mei 2016
[7] www.tempatwisataseru.com, diakses pada 7 Mei 2016.

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, 8 Mei 2016

Stefanus Agus Haryono