Perbandingan Algoritma Greedy dan Algoritma B&B dalam Pencarian Lintasan Terpendek

Agam Syauqi Lamaida[1], Dewangga Respati[2], Taufik Ramadhany[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if13056@students.if.itb.ac.id[1], if13120@students.if.itb.ac.id[2], if13112@students.if.itb.ac.id[3]

Abstrak

Berdasarkan pengalaman kami dalam mengerjakan Tugas Pemrograman 1 IF2251 (Penerapan Algoritma Dijkstra untuk Perutean Adaptif pada Jaringan Komputer), kami berpendapat bahwa algoritma dijkstra untuk pencarian lintasan terpendek yang tercantum dalam diktat tersebut kurang sempurna. Algoritma Dijkstra yang ada di diktat membutuhkan waktu pencarian menjadi lebih lama. Hal itu disebabkan karena pada algoritma dijkstra, prinsipnya adalah mencari lintasan yang memenuhi dari simpul awal ke simpul akhir. Lalu semua lintasan tersebut dibandingkan. Sehingga otomatis waktu penemuan solusi pun menjadi lebih lama. Kami mempunyai pikiran untuk menerapkan Algoritma Branch & Bound pada pencarian lintasan terpendek. Karena pada Algoritma Branch & Bound, prinsip pencarian solusi adalah mencari solusi optimal dengan memperhatikan solusi yang akan ditemukan selanjutnya. Dengan adanya prinsip tersebut, kami berpendapat bahwa solusi yang ditemukan adalah solusi yang benar untuk semua keadaan dengan tetap memperhatikan kemangkusan dan kecepatan algoritma.

***Kata kunci**: algoritma greedy, algoritma branch & bound, algoritma dijkstra*

1. Pendahuluan

Prinsip pencarian lintasan terpendek memegang peranan yang penting selama ini. algoritma yang selama ini sering digunakan untuk memecahkan masalah tersebut adalah algoritma dijkstra. Tapi algoritma dijkstra yang selama ini kami pelajari,membutuhkan waktu yang lama untuk menemukan solusinya. Hal ini dikarenakan pada algoritma dijkstra kita membandingkan semua lintasan yang mungkin dari simpul awal ke simpul akhir. Kami mempunyai pemikiran untuk menerapkan Algoritma Branch & Bound pada prinsip pencarian solusi. Dengan penerapan ini akan ditemukan solusi yang benar untuk semua kondisi tanpa mengurangi kemangkusan dari algoritma.

2. Algoritma Greedy

2.1. Definisi Algoritma Greedy

Algoritma Greedy membentuk solusi langkah per langkah. Pada setiap langkah tersebut akan dipilih keputusan yang paling optimal. Keputusan tersebut tidak perlu memperhatikan keputusan selanjutnya yang akan diambil, dan keputusan tersebut tidak dapat diubah lagi pada langkah – langkah selanjutnya. Prinsip utama algoritma greedy adalah prinsip *"take what you can get now!"*. Maksud dari prinsip tersebut adalah sebagai berikut; pada setiap langkah dalam algoritma greedy kita ambil keputusan yang paling optimal untuk langkah tersebut tanpa memperhatikan konsekuensi pada langkah – langkah selanjutnya, lalu kita namakan solusi tersebut dengan optimum lokal**.** Dengan cara pengambilan nilai optimum lokal pada setiap langkah diharapkan akan tercapainya optimum global**,** yaitu tercapainya solusi optimum yang melibatkan keseluruhan langkah dari awal sampai akhir. Dalam makalah ini, kami menggunakan algoritma Dijkstra untuk dibandingkan dengan algoritma B&B.

2.2. Skema Umum Algoritma Greedy

Elemen – elemen yang digunakan dalam penerapan algoritma greedy antara lain :

1. Himpunan Kandidat, C.
   Himpunan yang berisi elemen pembentuk solusi.
2. Himpunan Solusi, S.
   Himpunan yang terpilih sebagai solusi persoalan.
3. Fungsi Seleksi, SELEKSI.
   Fungsi yang memilih kandidat yang paling mungkin untuk mencapai solusi optimal.
4. Fungsi Kelayakan, LAYAK.
   Fungsi yang memeriksa apakah suatu kandidat yang dipilih dapat memberikan solusi yang layak. Maksudnya yaitu apakah

kandidat tersebut bersama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala yang ada.

5. Fungsi SOLUSI
   Fungsi yang mereturn *boolean.* True jika himpunan solusi yang sudah tebentuk merupakan solusi yang lengkap; False jika himpunan solusi belum lengkap.
6. Fungsi Objektif
   Fungsi yang mengoptimalkan solusi.

Skema umum dari algoritma greedy dapat kita tuliskan sebagai berikut :

1. Inisialisasi S dengan kosong.
2. Pilih sebuah kandidat dari C (dengan SELEKSI).
3. Kurangi C dengan kandidat yang telah terpilih di atas.
4. Periksa apakah kandidat yang dipilih tersebut bersama – sama dengan himpunan solusi membentuk solusi yang layak (dengan LAYAK). Jika ya, masukkan kandidat ke himpunan solusi; jika tidak buang kandidat tersebut dan tidak perlu ditelaah lagi.
5. Periksa apakah himpunan solusi yang sudah terbentuk telah memberikan solusi yang lengkap (dengan SOLUSI). Jika ya, berhenti; jika tidak, ulangi dari langkah 2.

2.3. Algoritma Dijkstra

2.3.1. Skema Umum Algoritma Dijkstra

Algoritma Dijkstra menggunakan strategi greedy sebagai berikut :

> Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih.

Beberapa elemen yang kita gunakan dalam penerapan algoritma dijkstra :

1. Graf berbobot dengan n buah simpul kita representasikan dalam matriks M. Elemen matriks M yang dinyatakan dengan m_{ij}, yang dalam hal ini :
   m_{ij} = bobot sisi (i,j)
   $m_{ii} = 0$
   $m_{ij} = \infty$, jika tidak ada sisi dari simpul i ke simpul j.
2. Tabel S = [s_i] yang dalam hal ini,
   $s_i = 1$, jika simpul i termasuk ke dalam lintasan terpendek.
   $s_i = 0$, jika simpul i tidak termasuk ke dalam lintasan terpendek.
3. Dan hasil output dari dijkstra ini. elemen ini tergantung pada kebutuhan kita. Misal kita ingin mengoutput hanya panjang lintasan terpendek sajadari simpul asal ke setiap simpul, maka kita gunakan tabel yang elemennya sebanyak anggota simpul dan didalamnya menyimpan nilai jarak dari simpul asal ke semua simpul yang ada. Pada bagian 3.2 di bawah ini, kami mencotohkan hasil output yang berupa panjang lintasan terpendeknya saja dari simpul asal ke setiap simpul yang ada.

2.4. Prinsip Pencarian Lintasan Terpendek dengan Algoritma Dijkstra

Pada algoritma Dijkstra [1], semua elemen matriks M diisi dengan jarak antara simpul awal dengan simpul lainnya jika ada sisi yang menghubungkan kedua simpul tersebut. Jika tidak ada, elemen matriks diisi dengan ∞.

Selanjutnya, dijalankan proses iteratif yang akan memeriksa tiap simpul kecuali simpul awal. Dalam proses ini, dicari terlebih dahulu simpul yang memiliki jarak terpendek dengan simpul yang sebelumnya (untuk pertama kali adalah simpul awal), dan nilai S dari simpul tersebut diisi dengan nilai 1. Simpul ini untuk selanjutnya disebut simpul antara. Dari simpul antara tersebut, jarak antara simpul awal dengan simpul lain diperiksa. Jika jarak antara simpul awal dengan sebuah simpul lebih besar dari jarak simpul awal dengan simpul antara + jarak simpul antara dengan simpul tujuan tersebut, maka jarak antara simpul awal diisi dengan simpul tujuan. Proses ini diulangi sebanyak n -1 kali, dengan n adalah jumlah simpul dari graf.

3. Algoritma Branch & Bound

3.1. Definisi Algoritma Branch & Bound

Algoritma Branch & Bound merupakan algoritma yang menerapkan proses pencarian solusi secara skema BFS. Dalam pencarian solusi, setiap simpul diberi suatu atribut *cost*. Cost ini digunakan untuk mengenali simpul berikutnya yang akan dibangkitkan. Jadi dalam pembangkitan simpul selanjutnya kita tidak membangkitkannya berdasar urutan pembangkitan, tapi simpul dengan nilai cost terkecil yang akan diekspansi untuk menghasilkan solusi akhir. Nilai *cost* pada setiap simpul merepresentasikan nilai taksiran ($\hat{c}$) ongkos termurah dari simpul awal ke simpul solusi.

Dapat kita simpulkan bahwa nilai taksiran tersebut adalah merupakan batas bawah sehingga dapat diartikan sebagai fungsi pembatas. Fungsi pembatas ini digunakan untuk membatasi pembangkitan simpul yang tidak mengarah ke penemuan solusi.

3.2. Skema Umum Algoritma B&B

Pada penggunaan algoritma B&B ini kita membutuhkan antrian Q. Aturan pemasukan simpul ke dalam antrian adalah sebagai berikut, simpul yang baru dibangkitkan diletakkan di belakang antrian; dan simpul berikutnya yang akan diekspansi adalah simpul yang ada di depan antrian. Jadi dapat disimpulkan algoritma ini menerapkan skema *FIFO (First In First Out).*
Rumus yang digunakan untuk menghitung nilai taksiran (ĉ) adalah :

$$\hat{C}(i) = \hat{g}(i) + \hat{s}(i)$$

$\hat{C}(i)$ = ongkos untuk simpul i
$\hat{s}(i)$ = ongkos untuk mencapai simpul tujuan dari simpul i
$\hat{g}(i)$ = ongkos untuk mencapai simpul i dari simpul awal

Algoritma B&B secara umum adalah sebagai berikut

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi, maka solusi telah ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. Stop.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang memiliki ĉ paling kecil.
4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan, stop. Jika simpul i bukan merupakan simpul solusi maka bangkitkan semua anak – anaknya. Jika simpul i tidak mempunyai anak maka kembali ke langkah 2.
5. Simpul i dihapus dari Q dan dimasukkan ke himpunan solusi S.
6. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak – anak tersebut ke dalam antrian Q.
7. Kembali ke langkah 2.

3.3. Prinsip Pencarian Lintasan Terpendek dengan Algoritma B&B

Kita anggap antrian Q dan pohon ruang status sudah terbentuk. Langkah pertama adalah kita masukkan simpul asal ke dalam antrian Q. Karena pada antrian Q baru ada satu simpul yaitu simpul asal, maka kita pilih simpul asal untuk dibangkitkan. Jika simpul asal merupakan simpul solusi, maka solusi telah ditemukan; proses dihentikan. Jika bukan, maka bangkitkan semua anak simpul asal tersebut. Untuk setiap simpul anak hitung ĉ dari simpul anak dan masukkan ke antrian dengan skema *FIFO.*
Untuk langkah selanjutnya, kita ambil elemen pada antrian yang memiliki nilai ĉ yang paling kecil dan kita lakukan lagi proses seperti di atas. Mulai dari proses pembangkitan anak hingga proses pemasukan elemen anak tersebut ke dalam antrian.
Untuk pemrosesan traversal ke masing – masing elemen anak, kita dapat memilih menggunakan proses iteratif atau proses rekursif. Tapi karena Algoritma B&B menggunakan skema pencarian secara BFS, maka akan lebih natural apabila kita menggunakan proses iteratif.

4. Analisis Hasil Algoritma Dijkstra dan Algoritma B&B

Pada proses analisis ini kami lebih menekankan kepada aspek kemangkusan dan kompleksitas algoritma. Tapi selain itu kami juga akan membahas aspek – aspek lain yang bersangkutan.
Dari hasil penjabaran masalah pencarian lintasan terpendek dengan kedua algoritma ini, kami akan menelaah beberapa hal, antara lain :

1. Masalah waktu yang dibutuhkan
2. Masalah memori yang dihabiskan
3. Masalah keefektifan

Pada algoritma Dijkstra. Dari bagian 2.4, kami melihat bahwa prinsip utama algoritma dijkstra adalah mencari semua lintasan dari simpul asal ke suatu simpul tujuan dan kemudian membandingkan setiap lintasan tersebut. Hal ini dapat kami ilustrasikan sebagai berikut, misal kita akan mencari panjang terpendek dari simpul 1 ke simpul 4. Dan lintasan yang tersedia adalah lintasan 1-4, 1-2-3-4, 1-3-4. Maka untuk hal seperti ini pada algoritma dijkstra akan membandingkan ketiga lintasan tersebut. Lintasan yang memiliki jarak terpendek akan dihasilkan sebagai solusi. Dan apabila hal itu kita lakukan unutk semua simpul, maka dapat kita bayangkan berapa banyak proses perbandingan dan penghitungan yang terjadi. Karena hal ini maka otomatis waktu yang dibutuhkan akan lebih lama dan terlihat jelas bahwa memori yang dibutuhkan juga tidak sedikit. Dari dua hal tersebut di atas keefektifan dari algoritma dijkstra juga kurang sempurna.
Pada algoritma B&B. Dari bagian 3.3, kami menganalisis bahwa algoritma B&B akan mengaktifkan semua anak simpul. Dan simpul yang tidak mengacu ke penyelesaian yang benar akan "dibunuh". Cara mengecek apakah simpul tersebut akan dibunuh atau tidak yaitu dengan nilai ĉ. Simpul yang memiliki nilai ĉ paling kecil akan terus dibangkitkan sementara simpul lainnya akan "dibunuh". Pembangkitan tersebut akan berlangsung terus sampai akhirnya ditemukan simpul akhir sebagai simpul solusi. Dengan melihat ilustrasi yang di atas. Misal kita akan mencari lintasan terpendek dari simpul 1 ke simpul 4, maka kita tidak perlu mengecek semua lintasan seperti yang dilakukan di atas. Misalkan pada lintasan 1-2-3-4, bila pada simpul 2 kita temukan bahwa nilai ĉ lebih besar dari yang lain maka otomatis lintasan tersebut tidak akan dibangkitkan. Karena pada setiap pencarian solusi kita hanya membangitkan satu lintasan tanpa

membandingkan lintasan, maka dapat disimpulkan bahwa waktu pencarian akan lebih singkat, memori yang digunakan hanya sedikit, dan algoritma menjadi lebih efektif.
Jadi ada perbedaan yang mencolok antara kedua algoritma di atas. Pada algoritma dijkstra kita membangkitkan semua lintasan yang mungkin kemudian membandingkannya. Hal itu yang menyebabkan membesarnya waktu dan memori yang dibutuhkan sehingga menjadi kurang efektif. Sedangkan pada algoritma B&B, pada akhir pencarian solusi kita pasti langsung menemukan satu solusi yang pasti tanpa perlu membandingkannya lagi. Sehingga otomatis waktu dan memori yang dibutuhkan lebih sedikit dan algoritma menjadi lebih efektif.

5. Kesimpulan

Masalah pencarian lintasan terpendek dapat diselesaikan dengan berbagai cara antara lain dengan algoritma dijkstra dan dengan algoritma B&B.

Dari hasil analisis di atas, kami menyimpulkan bahwa algoritma B&B memiliki kemampuan yang lebih baik dibanding dijkstra dalam hal waktu yang dibutuhkan, pengelolaan memori dan yang terutama adalah masalah efektifitas.

6. Daftar Pustaka

1. Munir, Rinaldi. 2004. “Diktat Kuliah IF2251 Strategi Algoritmik”
2. Cormen. Thomas H., Leiserson, Charles E., Rivest, Ronald L. 1990. “Introduction to Algorithms”. MIT Press
3. Goodrich, Tamassia, Mount. 2004. “Data Structures and Algorithms in C++ ”. John Wiley & Sons, Inc.