GREEDY ALGORITHM TO SOLVE CARPOOL PROBLEM

Glen Christian

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jalan Ganesha no 10 Bandung
E-mail: if15098@students.if.itb.ac.id

ABSTRAK

Masalah tentang kenaikan bahan bakar sekarang ini, untuk kalangan bawah, kehidupan semakin sukar, saat mereka berjuang untuk tetap hidup ditengah-tengah harga yang terus melambung dan biaya transportasi umum yang terus menaik.

Saat pemakai transportasi umum mengeluh tentang biaya yang lebih tinggi, sama dengan pemilik mobil, mereka menghabiskan biaya untuk bahan bakar sebanyak dua kali dari biasanya. Indonesia tidak membutuhkan lebih banyak demonstran jalanan, debat politik, dan janji-janji palsu. Yang Indonesia butuhkan adalah lebih orang yang mau dan bisa untuk menyelesaikan masalah yang kita semua hadapi.

Salah satunya adalah Rudyanto yang karena masalah ini, memikirkan ide carpool. Carpool adalah saling meminjamkan kendaraan. Rudyanto berkata bahwa, dengan carpooling, orang dapat menghemat uang untuk biaya bahan bakar, sekaligus membantu mengurangi kepadatan kendaraan. Tetapi, bagaimanakah cara agar carpool problem ini dapat menjadi sesuatu yang adil bagi tiap penggunanya? Carpool Problem dapat diselesaikan dengan Algoritma Greedy.

Kata kunci: Carpool

1. PENDAHULUAN

Salah satu masalah yang marak sekarang ini dan banyak mengundang protes adalah masalah BBM (Bahan bakar minyak). Seperti yang kita ketahui sekarang harga premium adalah 4500 per liter. Sedangkan dulu, harganya hanya kira-kira 2200 per liter.

Bagi kalangan atas dari masyarakat, mungkin masalah ini tidak begitu menjadi sesuatu yang mereka cemaskan. Tetapi bagi kalangan menengah ke bawah, masalah ini membawa dampak yang sangat besar bagi kehidupan mereka. Dapat dibayangkan, untuk kalangan menengah, saat membeli BBM, dengan uang 100000 rupiah, mereka dapat membuat tangki mobil mereka penuh, sedangkan sekarang ini hal yang sama membutuhkan uang kira-kira 225000 rupiah. Karena masalah ini, maka timbulah ide tentang carpool. Carpool adalah saling meminjamkan kendaraan, sehingga dapat menghemat biaya BBM.

2. Carpool Problem

Masalah tentang carpool dapat dimodelkan sebagai berikut: kelompok yang terdiri dari N orang memutuskan untuk melakukan carpool untuk bekerja. Tiap hari adalah subset dari orang tersebut datang. Mereka harus melakukan formulasi untuk membentuk algoritma untuk memutuskan siapakah yang akan menyetir mobil pada hari itu.

Hal yang paling penting untuk dipertimbangkan adalah pemilihan dari pengguna mobil hari itu harus seadil-adilnya. Artinya untuk jangka waktu yang panjang, tidak boleh hanya satu supir saja yang menyetir lebih banyak dari yang lain. Algoritma yang mereka bentuk harus kuat, artinya apabila pada suatu hari jika satu orang yang tidak seharusnya menyetir, menyetir atau sebaliknya, algoritma tersebut seharusnya bisa memulihkan kasus pengecualian, dan masih bisa untuk membuat pemilihan yang adil pada tahap selanjutnya.

Untuk mendefinisikan keadilan dari algoritma, kita mempertimbangkan jumlah ideal pengendara/ supir dari peserta. Secara ideal, jika seseorang i datang sejumlah bk dari semua hari, saat semua pengendara sejumlah k datang maka i harus menyetir bk/k dari hari- hari tersebut. Ini adalah jumlah ideal dari seseorang i menyetir adalah $\sum_k bk/k$.

Sedangkan untuk error, formulasinya adalah $x = (x_i)$ dari formulasi di bawah ini

$$x' = x + \delta(s, i), \text{ dimana } \delta(s, i) = -e_i + \frac{1}{|s|} \sum_{P_j \in s}$$

Dimana e_j dan $|s|$ adalah jumlah kedatangan. Dalam kasus ini, algoritma tersebut adil apabila ada pembatas yaitu B yang menyebabkan

$\max_{i,t} |x_i(t)| \leq B$ independen.

Menemukan algoritma yang memerlukan batas bawah B mempunyai paling sedikit dua keuntungan. Pertama, saat kita ingin memodelkan keadilan, nilai B yang paling kecil akan menjamin deviasi terkecil dari kondisi ideal. Kedua, untuk program pen-generate jadwal otomatis kita secara pasti mempunyai keinginan untuk mengurangi jarak dari nilai yang mungkin untuk tiap variable yang kita harus pantau.

Untuk masalah secara global algoritma yang optimal adalah Algoritma Greedy. Dalam kesadaran bahwa batas B adalah yang terkecil, diketahui batas untuk Algoritma Greedy memenuhi persamaan

$B < \frac{n-1}{2}$ dimana $\frac{n-1}{2}$ adalah batas terkecil yang mungkin untuk semua algoritma. Untuk Carpool problem Coppersmith et al. telah menunjukkan bahwa semua algoritma memenuhi

$B \geq \frac{3}{8}(n-1) - \frac{1}{4}$

Tujuan kita adalah mengecilkan jarak dari kedua batas ini

2.1 Algoritma

Algoritma greedy adil dalam carpool problem ini. Algoritmanya adalah:

1. Assign initial values E = (0, . . . , 0) to the n people
2. If there are k people appear on that day, than pick the person who has the highest value to drive and update E:

   Add $(-1+\frac{1}{k})$ to driver's value

   Add $\frac{1}{k}$ to each passenger

Note : At each time t, $E_i = x_i - y_i$, and $\sum_i E_i = 0$.

An online algorithm A depends only on the past and present schedules of arrivals.

Start from E(0) = (0, ..., 0). At each time t, the cumulative error vector is
$E(t) = (E_1(t), ...,E_n(t))$, t € N
Define the error bound of carpool problem as
B = min BA, where BA = max schedule(max i,t |Ei (t)|)
If BA is finite, then the algorithm A is fair.

Jika n > 2, maka

$$B \leq Bgreedy \leq \frac{n-1}{2} - \frac{1}{lcm(2,3,...,n)}$$

Dan untuk tiap n,

$$B > \frac{3(n-1)}{8} - \frac{1}{4}$$

Algoritma Greedy optimal dari semua algoritma, batas bawah = batas atas.

2.2 Daftar Kedatangan

Pada lampiran ini ditunjukkan strategi kemenangan untuk mengeluarkan beberapa gaps. Tiap node menunjukkan error vector. Pada tiap state, terjadi pemilihan kedatangan. Tiap pilihan dari pengendara menuju ke error state yang baru. Daripada mencari semua kemungkinan yang mungkin, lebih baik memilih beberapa persen dari daftar kedatangan secara random. Biarpun lebih memakan waktu. Mulai dari asal, algoritma greedy diterapkan untuk kedatangan random untuk mengecek batas atas. Untuk n = 6 dan 40 milyar kedatangan, nilai terbesar dari Ei yang dipunyai adalah $\frac{88}{60}$ dimana batas atasnya adalah $\frac{149}{60}$

Algoritma diatas sukses untuk mengeluarkan lebih banyak gaps dari algoritma sebelumnya.
Tetapi, untuk membuat lower bound lebih baik, masih diperlukan untuk mengeluarkan paling sedikit tiga kasus berikutnya,

$(\frac{8}{12},\frac{10}{12},\frac{9}{12}), (\frac{7}{12},\frac{11}{12},\frac{9}{12}), (\frac{9}{12},\frac{9}{12},\frac{9}{12})$

3. Algoritma Greedy

Algoritma greedy adalah algoritma yang mengarah kepada problem solving yang bersifat metaheuristik yang melakukan optimasi secara lokal pada tiap tahap dengan harapan mencapai solusi secara global.

Sebagai contoh, mengaplikasikan strategi greedy pada traveling salesman problem dapat diartikan menjadi "Pada tiap tahap, kunjungi tiap kota yang paling dekat dari tempat kamu berada".

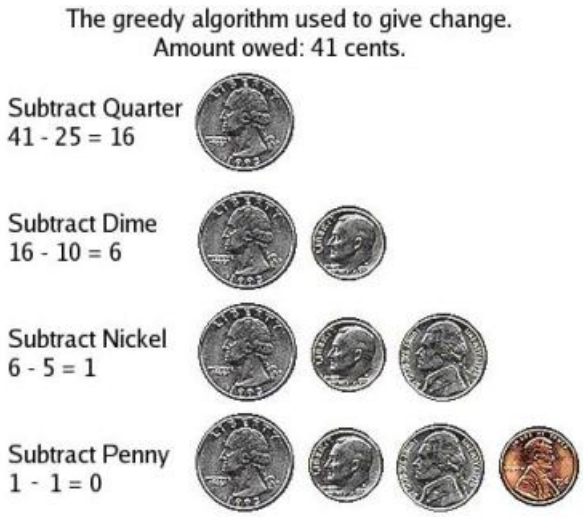
Yang terjadi:

1. Kandidat dari solusi tercipta
2. Ada fungsi seleksi, yang akan memilih kandidat terbaik untuk ditambahkan pada himpunan solusi
3. Fungsi kelayakan, untuk memutuskan bila suatu kandidat solusi dapat digunakan untuk kontribusi pada sebuah solusi.
4. Fungsi objektif, yang memberi nilai pada suatu solusi, atau solusi parsial, dan
5. Fungsi solusi, yang akan mengindikasi bahwa kita sudah menemukan solusi komplit.

Algoritma greedy menghasilkan solusi yang baik pada beberapa problem matematika, tetapi tidak pada yang lain. Kebanyakan problem yang dapat diselesaikan oleh Greedy mempunyai dua property yaitu:

Greedy Choice Property yaitu properti dimana kita dapat memilih pilihan apa saja yang akan menyelesaikan subproblem yang akan muncul nanti. Pilihan yang akan dibuat algoritma greedy akan bergantung pada pilihan yang dibuat sampai saat ini, tetapi bukan pilihan yang akan datang, atau semua solusi subproblem.

Secara iteratif akan mengambil pilihan berikutnya, mengurangi tiap problem menjadi sekup yang lebih kecil. Dengan kata lain, algoritma greedy tidak pernah memikirkan kembali pilihan yang sudah diambil. Ini adalah perbedaan yang paling mendasar dari program dinamis, yang menjamin untuk menemukan solusi. Pada tingkat yang berbeda, program dinamis mencari keputusan selanjutnya berdasarkan keputusan yang telah diambil sebelumnya.



Gambar 1. Contoh Masalah Yang Dapat Diselesaikan Algoritma Greedy

3.1 Aplikasi Greedy

Untuk kebanyakan masalah, algoritma greedy secara umum (tapi tidak selalu) gagal untuk mencari solusi optimal, karena algoritma ini biasanya tidak beroperasi pada semua data. Mereka memilih suatu pilihan dengan terlalu terburu-buru yang mencegahnya untuk mencapai solusi terbaik.

Sebagai contoh: greedy untuk graph coloring problem dan NP-complete problem tidak konsisten menemukan solusi optimum. Lebih jauh, mereka sebenarnya berguna, karena mereka memilih secara cepat dan sering memberi pendekatan yang baik ke solusi optimum.

Apabila algoritma greedy dapat dibuktikan untuk menghasilkan optimum global untuk problem yang diberikan, umumnya algoritma ini akan dipilih karena lebih cepat dari metode optimasi yang lain seperti program dinamis.

Contoh: beberapa algoritma greedy adalah algoritma Kruskal dan Prim untuk menemukan spanning trees, Algoritma Djikstra untuk memperoleh lintasan terpendek.

3.2 Problem Lain

Banyak problem lain yang dapat diselesaikan oleh Algoritma Greedy. Contohnya adalah Minimasi Waktu di dalam Sistem, Knapsack Problem: Fractional Knapsack, Penjadwalan Job dengan Tenggat Waktu, Pohon Merentang Minimum, Shortest Path, Travelling Salesperson Problem, dll.

Masalah yang terdapat pada Minimasi Waktu dalam sistem adalah sebagai berikut: Sebuah server mempunyai n pelanggan yang harus dilayani Waktu pelayanan untuk tiap pelanggan sudah ditetapkan sebelumnya. Kita harus meminimumkan waktu dalam sistem. Masalah ini dapat diselesaikan dengan algoritma Greedy.

Masalah kedua adalah Knapsack Problem. Dengan Algoritma Greedy dapat dipecahkan sebagai berikut: masalah dipecahkan dengan memasukkan objek satu persatu ke dalam knapsack. Objek yang telah dimasukkan tidak dapat dikembalikan lagi. Dalam masalah ini greedy dapat menyelesaikan dengan beberapa alternative yaitu: Greedy by profit, Greedy by weight, Greedy by density, dll.

Masalah berikutnya adalah Job Schedulling. Algoritma Greedy dengan baik dapat menyelesaikan persoalan ini. Masalah yang kita hadapi adalah: Misalkan kita mempunyai n buah job yang akan dikerjakan oleh sebuah mesin. Tiap job diproses oleh mesin selama satu satuan waktu dan tenggat waktu - yaitu batas waktu job tersebut harus sudah selesai diproses.

Shortest Path, contoh masalahnya adalah sebagai berikut: misalkan simpul pada graf dapat merupakan terminal komputer atau simpul komunikasi dalam suatu jaringan, sedangkan sisi menyatakan saluran komunikasi yang menghubungkan dua bua terminal. Bobot pada graf dapat menyatakan biaya pemakaian saluran komunikasi antara dua buah terminal, jarak antara dua buah terminal, atau waktu pengiriman pesan(message) antara dua buah terminal. Persoalan lintasan terpendek di sini adalah menentukan jalur komunikasi terpendek antara dua buah terminal komputer. Lintasan terpendek akan menghemat waktu pengiriman pesan dan biaya komunikasi.

Yang terakhir, contoh masalahnya adalah: Diberikan n buah kota serta diketahui jarak antara setiap kota satu sama lain. Temukan perjalanan terpendek yang melalui

setiap kota lainnya hanya sekali dan kembali lagi ke kota asal keberangkatan. Pada masalah ini strategi greedy untuk memilih kota selanjutnya yang dikunjungi adalah: Pada setiap langkah, pilih kota yang belum pernah dikunjungi yang mempunyai jarak terdekat.

Sebenarnya Algoritma Greedy cukup powerful untuk menyelesaikan masalah. Contohnya adalah masalah-masalah di atas dapat diselesaikan dengan optimal oleh algoritma ini. Tetapi dengan pengamatan dapat dilihat, algoritma greedy hanya dapat mengatasi masalah-masalah bertipe tertentu.

IV. KESIMPULAN

Untuk kebanyakan masalah, algoritma greedy secara umum (tapi tidak selalu) gagal untuk mencari solusi optimal, karena algoritma ini biasanya tidak beroperasi pada semua data. Algoritma greedy berprinsip "Take what you can get". Pilihlah pilihan yang maksimum/ minimum pada tiap keadaan. Tetapi sebenarnya prinsip ini tidak selalu menjadi yang terbaik untuk mencapai solusi optimum secara global.

REFERENSI

[1]. R.Fagin and J.H.Williams, A fair carpool scheduling algorithm, IBM Journal of Research and development **27**(2) (1983), 133–139.

[2]. M.Ajtai, J.Aspnes, M.Naor, Y.Rabani, L.J.Schulman and O.Waarts, Fairness in Scheduling, Journal of Algorithms **29**(2) (1998), 306–357.

[3]. Moni Naor, On fairness in the carpool problem, Journal of Algorithms **55**(1) (2005), 93–98. Don Coppersmith, Tomasz J.Nowicki, GuiseppeA.Paleologo, Charles Tresser, Chai Wah Wu, The Optimality of the On-line greedy algorithm in carpool and chairman assignment problems , IBM Research Report, (2005)

[4]. S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, Proportionate progress: A notion of fairness in resource allocation, Algorithmica, 15, 600-625, 1996.

[5]. Munir, Rinaldi. 2006. *Diktat Kuliah IF2251 Strategi Algoritmik*. Penerbit ITB

[6]. http://www.nebeng.com/dokumentasi-detail.php?id=3

[7]. http://en.wikipedia.org/wiki