APLIKASI ALGORITMA *GREEDY* UNTUK MENGOPTIMALKAN EFISIENSI PEKERJAAN

Tommy Hidayat Santoso - 13506071

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jalan Ganesha 10, Bandung
E-mail: if16071@students.if.itb.ac.id

ABSTRAK

Efisiensi pada zaman modern ini adalah sesuatu yang sangat krusial. Pekerjaan yang baik adalah pekerjaan yang dikerjakan dengan efisien sehingga menghasilkan keuntungan maksimal. Masalah yang dibahas dalam makalah ini adalah pengalokasian jumlah pegawai ke dalam beberapa pekerjaan yang berbeda sehingga tenaga yang digunakan efisien. Masalah yang dibahas dalam makalah ini hanyalah pengalokasian pegawai yang bersifat umum untuk suatu masalah. Untuk pengembangannya dapat dilakukan pengalokasian pegawai secara khusus sesuai spesialisasinya.

Kata kunci: *greedy, efisiensi*

1. PENDAHULUAN

Zaman yang semakin modern menuntut efisiensi setiap saat. Ini membuat penggunaan sumber daya manusia harus digunakan sebaik-baiknya untuk mencapai hasil yang terbaik. Sebagai manusia yang hidup di zaman modern ini, saya ingin mencoba mengimplementasikan ilmu yang sudah saya dapatkan ke dalam kehidupan nyata. Untuk itu saya mencoba menggunakan algoritma *greedy* ini untuk mengoptimalkan penggunaan sumber daya manusia dalam lingkungan kerja. Dengan kerja yang optimal maka hasil yang didapatkan juga akan maksimal.

2. ALGORITMA *GREEDY*

2.1. DEFINISI ALGORITMA *GREEDY*

Algoritma *greedy* adalah algoritma yang cara kerjanya mirip dengan salah satu sifat buruk manusia, rakus. Sesuai prinsip rakus, algoritma *greedy* ini akan mengambil keputusan yang dianggap terbaik hanya untuk saat itu saja yang diharapkan dapat memberikan solusi terbaik secara keseluruhan. Algoritma *greedy* ini praktis, ringkas dan fleksibel (dapat digunakan pada berbagai persoalan dengan hasil yang cukup memuaskan). Algoritma *Greedy* membentuk solusi langkah per langkah *(step by step).* Terdapat banyak pilihan yang perlu di eksplorasi pada setiap langkah solusi, karenanya pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.

Pada setiap langkah diperoleh optimum lokal. Bila algoritma berakhir, kita berharap optimum lokal menjadi optimum global.

2.2 SKEMA UMUM ALGORITMA

Algoritma *greedy* disusun oleh elemen-elemen berikut:

1. Himpunan kandidat.
Berisi elemen-elemen pembentuk solusi. Dalam kasus ini adalah banyaknya pegawai.

2. Himpunan solusi
Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan. Dalam kasus ini adalah himpunan jumlah pegawai yang ditugaskan pada suatu pekerjaan.

3. Fungsi seleksi (*selection function*)
Memilih kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan (*feasible*)
Memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala (*constraints*) yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.

5. Fungsi obyektif
Fungsi yang memaksimumkan atau meminimumkan nilai solusi (misalnya panjang lintasan, keuntungan, dan lain-lain). Dalam kasus ini adalah peningkatan efisiensi setiap ditambahkan pegawai.

3. METODE

Kita sekarang akan melihat bagaimana algoritma *greedy* menugaskan jumlah sumber daya manusia pada suatu pekerjaan sehingga menjadi efisien. Dalam hal ini dimisalkan kita mempunyai empat buah pekerjaan yang berbeda, kita beri nama A, B, C, D, sedangkan jumlah sumber daya manusia yang tersedia hanya ada delapan orang. Pada kasus ini kita harus menugaskan semua pegawai pada pekerjaan yang ada. Algoritma yang dipakai adalah *greedy by efficiency,* yaitu mencari berapa peningkatan efisiensi maksimumnya. Sebelum memulai ke dalam implementasi algoritma kita harus mengetahui terlebih dahulu berapa sebenarnya efisiensi maksimum dari suatu pekerjaan untuk semua kemungkinan. Karena jumlah pegawai yang dipunyai ada delapan orang, maka ada delapan kemungkinan untuk setiap pekerjaan. Tabel di bawah ini menunjukkan efisiensi yang mungkin dari ke-4 pekerjaan yang tersedia

SDM	0	1	2	3	4	5	6	7	8
A	0	60	75	85	85	70	60	50	30
B	0	70	70	70	70	90	90	90	100
C	0	80	90	90	85	80	75	75	75
D	0	90	85	80	75	50	40	30	20

Tabel 3.1 Efisiensi setiap pekerjaan dengan semua kemungkinan jumlah pegawai

Karena kita akan menugaskan pegawai ke dalam beberapa pekerjaan yang berbeda, ada baiknya kita menghitung terleih dahulu berapa efisiensi yang ditambahkan jika seorang pegawai ditugaskan ke dalam pekerjaan tertentu. Hal ini bukan perkara sulit karena kita tinggal menghitung perbedaan efisiensi sebelum pengalokasian pegawai dan setelah pengalokasian pegawai. Jika hanya ada satu pegawai, maka solusi optimalnya adalah jika kita menugaskan dia ke suatu pekerjaan dimana dia akan memberikan efisiensi tertinggi. Tetapi karena kita mempunyai lebih dari satu pegawai kita harus mengecek apakah dengan menugaskan mereka dengan cara biasa dapat menghasilkan solusi optimal global. Tabel di bawah ini menunjukkan perubahan efisiensi setiap pekerjaan untuk setiap pegawai tambahan:

SDM	1	2	3	4	5	6	7	8
A	60	15	10	0	-15	-10	-10	-20
B	70	0	0	0	20	0	0	10
C	80	10	0	-5	-5	-5	0	0
D	90	-5	-5	-5	-20	-10	-10	-10

Tabel 3.2 Perubahan efisiensi setiap pekerjaan setiap penambahan pegawai

Dengan melihat tabel 3.2 kita dapat melihat kemana pegawai pertama akan ditugaskan. Pekerjaan A memiliki efisiensi 60, pekerjaan B memiliki efisiensi 70, pekerjaan C memiliki efisiensi 80, dan pekerjaan D memiliki efisiensi 90. Dengan data ini kita dapat langsung menugaskan pegawai pertama ke dalam pekerjaan D yang merupakan optimum lokal untuk saat ini.

Lihat lagi tabel 3.2 untuk menentukan kemana pegawai kedua akan ditugaskan. Untuk saat ini pekerjaan yang memberikan pertambahan efisiensi paling besar adalah pekerjaan C sehingga pegawai 2 langsung kita alokasikan ke pekerjaan C.

Lihat lagi tabel 3.2 menambahkan pegawai ketiga ke dalam pekerjaan B akan memberikan nilai efisiensi 70. Sedangkan jika ditugaskan ke pekerjaan A hanya akan menghasilkan efisiensi 60. Ini berarti pegawai ketiga kita alokasikan ke dalam pekerjaan B.

Setelah tiga pegawai kita alokasikan kepada tiga pekerjaan yang berbeda pula, maka hanya ada satu pekerjaan yang tersisa yang belum mempunyai pegawai. Untuk menentukan nasib pekerjaan ini mari kita lihat lagi tabel 3.2. Untuk saat ini pekerjaan A lah yang memberikan pertambahan efisiensi tertinggi, kesanalah pegawai keempat ditugaskan.

Sekarang semua pekerjaan mempunyai pekerjanya masing-masing. Di sinilah saatnya menggunakan algoritma *greedy* untuk menentukan pengalokasian empat pegawai yang tersisa. Lihat lagi tabel 3.2 bandingkan berapa perubahan efisiensi yang diberikan untuk penambahan pegawai berikutnya. Terlihat bahwa pekerjaan A jika dikerjakan oleh dua orang efisiensinya akan bertambah paling besar, karena itu pegawai kelima akan ditugaskan ke pekerjaan tersebut.

Pegawai keenam mempunyai dua pilihan pekerjaan yang memberikan peningkatan efisisensi yang sama, yait pekerjaan A dan pekerjaan C. Dalam kasus kali ini yang dipilih adalah pekerjaan dengan indeks terkecil. Pegawai keenam ditugaskan ke pekerjaan A sehingga pekerjaan A sekarang dikerjakan tiga orang pegawai.

Lihat lagi tabel 3.2 pegawai ketujuh hanya akan memberikan efisiensi jiika dia ditugaskan ke dalam pekerjaan C dimana dia akan memberikan peningkatan efisiensi sebesar 10. Sedangkan jika dia ditugaskan ke pekerjaan yang lainnya tidak akan menambah efisiensi sama sekali atau malah mengurangi efisiensi. Jadi jelaslah kemana pegawai ketujuh.

Untuk pegawai yang terakhir atau pegawai kedelapan, mari kita lihat lagi tabel 3.2 untuk terakhir kalinya. Sebenearnya jumlah pegawai yang sekarang ditugaskan sudah optimum sehingga penambahan jumlah pegawai sebanyak satu orang ke pekerjaan manapun tidak akan memberikan peningkatan efisiensi. Tetapi karena semua pegawai harus ditugaskan pada suatu pekerjaan maka pegawai terakhir ini bebas ditempatkan di pekerjaan manapun kecuali pekerjaan D karena ini satu-satunya pekerjaan yang memberikan pengurangan efisiensi untuk saat ini. Pada kasus ini dia ditugaskan ke pekerjaan dengan indeks terkecil yaitu A sehingga solusi akhir yang kita dapatkan untuk masalah ini adalah J = {4,1,2,1}. Dari contoh di atas memang terlihat bahwa solusi ini adalah solusi optimal tetapi apakah penggunaan algoritma *greedy* ini akan selalu berhasil?

Kasus: Kita mendapatkan efisiensi maksimum total ketika kita menugaskan pegawai ke dalam suatu pekerjaan dimana mereka dapat memberikan peningkatan efisiensi terbesar.

Bukti: Misalkan pekerjaan kita namai dengan A, B, C, D, lalu didefinisikan x1, x3, x3, x4 (dengan x adalah nama pekerjaan) sebagai berikut:

x1 – pertambahan efisiensi jika satu pegawai ditugaskan ke dalam pekerjaan x

x1 + x2 – pertambahan efisiensi jika dua pegawai ditugaskan ke dalam pekerjaan x

Dengan algoritma *greedy* yang kita gunakan, pegawai akan meberikan peningkatan efisiensi sebesar max(a1,b1,c1,d1) yang akan menghasilkan nilai d1 untuk pegawai pertama lalu max(a1,b1,c1,d2) untuk pegawai kedua. Peningkatan efisiensi untuk saat ini adalah d1+ max(a1,b1,c1,d2). Jika kita memilih pegawai kedia untuk langsung kembali ditempatkan ke dalam pekerjaan D, maka alternatif perubahan efisiensi untuk pegawai kedua adalah d1+ d2. Untuk kasus pertama efisiensi total yang dicapai adalah d1+c1 <= d1+ max(a1,b1,c1,d2). Sedangkan untuk kasus alternatif efisiensi total yang dicapai adalah d1+d2. **Perlu diketahui bahwa peningkatan efisiensi jika seorang pegawai ditugaskan untuk suatu pekerjaan yang belum dikerjakan akan lebih tinggi daripada menambahkan jumlah pegawai untuk membantu pekerjaan lainnya.**

Karena a1+a2 <= d1+a2 <= d1+c1 <= d1+ max(a1,b1,c1,d2) maka pemilihan pekerjaan dengan algoritma *greedy* adalah yang terbaik

Membuat algoritma ini tidaklah susah, tetapi satu yang menjadi masalah adalah membatasi masalah (semua pegawai harus ditugaskan). Maksimal terdapat delapan pegawai dalam suatu pekerjaan dan jika seorang pegawai dapat ditugaskan ke dalam dua pekerjaan yang akan memberikan peningkatan efisiensi yang sama, maka tugaskanlah dia ke pekerjaan yang memiliki indeks lebih kecil.

4. KESIMPULAN

Makalah ini menunjukkan penggunaan algoritma *greedy* untuk menugaskan jumlah orang yang tepat agar pekerjaan efisien dan memberikan keuntungan terbesar. Hal itu akan sering kita temui dalam kehidupan ini terutama di dunia bisnis.

Algoritma *greedy* adalah algoritma yang praktis, gampang diimplementasikan, dan berjalan dengan cepat walaupun tidak selalu memberikan hasil optimal. Pembuktian kebenarannya memerlukan bukti matematik yang kompleks dan panjang. Algoritma *greedy* sendiri dikenal sebagai algoritma yang rapuh, sedikit kesalahan bisa berakibat fatal, tetapi ini ditutupi dengan penerapan *greedy* yang bersifat universal.

Menggunakan algoritma *greedy* berarti berani menempuh resiko, walaupun dari luar terlihat tidak ada masalah bisa saja akan ada masalah yang muncul di saat tidak diperkirakan. Kelebihan lainnya, algoritma *greedy* ini sering menjadi pilihan jika masalah yang dihadapi itu besar dan kompleks yang membutuhkan waktu lama jika diselesaikan menggunakan algoritma lainnya, itulah yang membuat algoritma *greedy* ini semakin menarik. Makalah ini menunjukkan bahwa algoritma *greedy* adalah salah satu alternatif untuk memecahkan masalah dengan cepat. **Memang tidak ada algoritma yang sempurna.**

REFERENSI

[1] Munir, Rinaldi. (2007). Diktat Kuliah Strategi Algoritmik. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] url : www.topcoder.com
Waktu akses : 16 Mei 2008 16.34 WIB