# Implementasi Algoritma Greedy dalam Pembagian kerja pada Mesin yang Identik

William Sentosa / 13513026
*Program Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*williamsentosa@itb.ac.id*

***Abstract*—Proses optimasi adalah sebuah hal yang sangat dicari pada dunia industry. Selain menghemat waktu, proses ini juga dapat memberikan keuntungan yang maksimal bagi penggunanya. Salah satu persoalan yang cukup sering ditemui dalam dunia industri adalah pembagian pekerjaan terhadap mesin yang identik. Sering kali industri menginginkan pembagian kerja yang baik sehingga tidak perlu waktu lama untuk menyelesaikan seluruh pekerjaan. Algoritma greedy merupakan suatu algoritma yang terkenal dalam masalah optimas. Selain cukup mangkus, algoritma ini juga relatif mudah untuk diimplementasikan.**

***Keywords— brute force, exhaustive search, greedy*, mesin, pekerjaan.**

## I. PENDAHULUAN

Dalam kehidupan sehari-hari, proses yang efisien dalam hal waktu sangatlah penting. Waktu menjadi hal yang sangat berharga untuk sebagian besar orang sehingga banyak orang yang mencari keefisienan dalam suatu proses. Suatu lingkup yang sangat menekankan keefisienan dalam proses adalah dunia perindustrian.

Dalam dunia perindustrian, keefisienan dalam waktu sangatlah penting. Semakin efisien proses produksinya, semakin maksimal keuntungan yang didapatkan. Salah satu permasalahan yang sering ditemui adalah pembagian tugas pada mesin yang identik. Jenis tugas yang diberikan memiliki waktu proses yang berbeda-beda. Selain itu, setiap mesin dapat memproses seluruh jenis pekerjaan karena mesin tersebut identik satu sama lain.

Permasalahan ini dapat diselesaikan dengan menggunakan algoritma *greedy*, salah satu algoritma yang cukup umum dalam dunia Informatika. Algoritma ini akan menghasilkan pembagian kerja yang efisien dengan kompleksitas waktu yang jauh lebih baik dari algoritma *exhaustive search*. Makalah ini akan membahas implementasi algoritma *greedy* pada permasalahan pembagian kerja pada mesin yang identik.

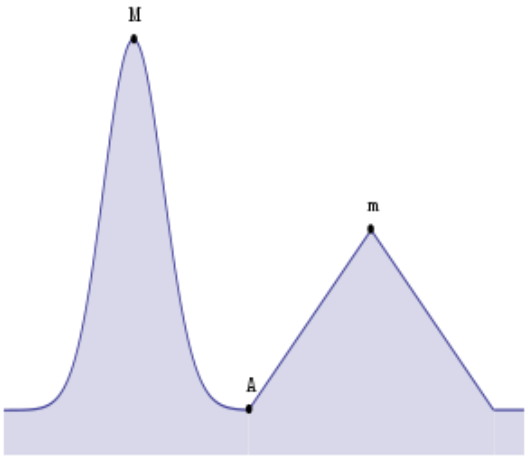
## II. DASAR TEORI

### 1. Algoritma Greedy

Algoritma *greedy* merupakan algoritma yang paling populer untuk memecahkan persoalan optimasi. Persoalan optimasi yang dimaksud adalah persoalan mencari solusi optimum, entah itu persoalan maksimasi (*maximization*) maupun minimasi (*minimization*).

Arti *greedy* bila diterjemahkan memiliki arti rakus, tamak, dll. Algoritma ini mencerminkan prinsip greedy yang rakus dalam pembentukan solusi langkah per langkah. Pada setiap langkah, terdapat banyak pilihan yang perlu dievaluasi, sehingga algoritma ini akan memilih langkah yang terbaik pada setiap langkah.

Pada setiap langkah, kita membuat pilihan optimum lokal dengan harapan langkah sisanya akan mengarah ke solusi optimum global. Namun, hasil yang diperoleh oleh algoritma ini tidak selalu menghasilkan solusi yang terbaik. Solusi yang didapat bisa berupa solusi *sub-optimum* atau *pseudo-optimum*.




**Gambar 1.** Grafik pencarian *greedy*

Melalui gambar diatas, dapat kita lihat bhawa mungkin saja algoritma greedy hanya menemukan solusi m yang merupakan solusi *sub-optimum* padahal solusi seharunya adalah M.

Berikut merupakan skema umum algoritma *greedy*:

```
procedure greedy(input C: himpunan_kandidat;
output S : himpunan_solusi)
{ menentukan solusi optimum dari persoalan
optimasi dengan algoritma greedy
Masukan: himpunan kandidat C
Keluaran: himpunan solusi S
}
Deklarasi
   x : kandidat;
Algoritma:
   S←{} { inisialisasi S dengan kosong }
   while (belum SOLUSI(S)) and (C ≠ {} ) do
      x←SELEKSI(C); { pilih sebuah kandidat
                      dari C}
      C← C - {x} { elemen himpunan kandidat
                   berkurang satu }
      if LAYAK(S ∪ {x}) then
         S←S ∪ {x}
      endif
   endwhile
{SOLUSI(S) sudah diperoleh or C = {} }
```

Algoritma *greedy* disusun oleh elemen-elemen berikut:

1. **Himpunan kandidat.**
   Berisi elemen-elemen pembentuk solusi.

2. **Himpunan solusi**
   Berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.

3. **Fungsi seleksi**
   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**
   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 ini merupakan fungsi yang memaksimumkan atau meminimumkan nilai solusi (misalnya keuntungan, dan lain-lain).

Algoritma *greedy* tidak selalu memberikan solusi yang paling optimum karena 2 hal berikut :

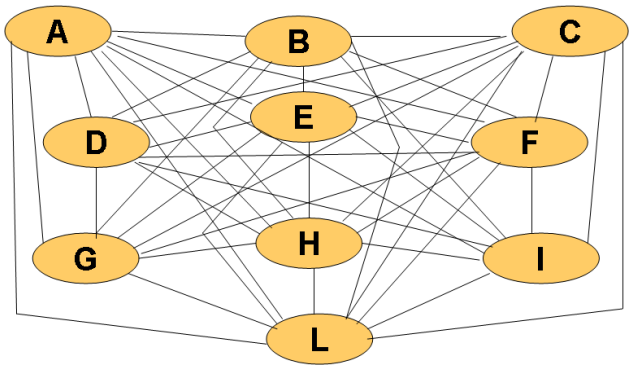
1. Algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada (sebagaimana pada metode *exhaustive search*).
2. Pemilihan Fungsi Seleksi
   Mungkin saja terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma bekerja dengan benar dan menghasilkan solusi yang benar-benar optimum.

**2. Algortima Exhaustive Search**

*Exhaustive search* adalah teknik pencarian solusi secara solusi *brute force* untuk masalah yang melibatkan pencarian elemen dengan sifat khusus, biasanya di antara objek-objek kombinatorik seperti permutasi, kombinasi, atau himpunan bagian dari sebuah himpunan.

Langkah-langkah metode *exhaustive search* :

1. Enumerasi (list) setiap solusi yang mungkin dengan cara yang sistematis.
2. Evaluasi setiap kemungkinan solusi satu per satu, mungkin saja beberapa kemungkinan solusi yang tidak layak dikeluarkan, dan simpan solusi terbaik yang ditemukan sampai sejauh ini (the best solution found so far).
3. Bila pencarian berakhir, umumkan solusi terbaik (the winner).




**Gambar 2.** *Exhaustive search* dalam penelusuran graf

Gambar diatas menunjukan bagaimana exhaustive ssearch bekerja. Exhaustive search mencacah semua kemungkinan yang ada lalu membandingkannya satu sama lain.

Algortima ini pasti akan selalu menghasilkan solusi karena algoritma ini mencacah seluruh kemungkinan solusi. Akan tetapi, waktu dan sumberdaya yang dibutuhkan oleh algoritma ini sangatlah besar sehingga untuk permasalahan yang besar algoritma ini jarang digunakan.

## III. RUMUSAN MASALAH

Permasalahan yang harus dipecahkan adalah persoalan optimasi. Ada **n buah jenis pekerjaan** yang harus dijadwalkan untuk dilakukan oleh **m buah mesin yang identik**. Hasil yang diharapkan adalah seluruh pekerjaan dikerjakan dan waktu untuk menyelesaikan seluruh pekerjaannya merupakan waktu yang paling **minimum**.

Berikut merupakan salah satu ilustrasi permasalahan :

Terdapat 5 jenis pekerjaan untuk membuat pakaian. Jenis pakaian yang harus dibuat adalah adalah kemeja, kaos tanpa kerah, kaos dengan kerah, jaket, dan celana panjang. Terdapat 3 mesin identik yang dapat memproses setiap jenis pakaian.

Berikut merupakan waktu yang diperlukan untuk membuat setiap jenis pakaian :

- Untuk membuat 1 potong kemeja diperlukan waktu 3 jam
- Untuk membuat 1 potong kaos tanpa kerah diperlukan waktu 1 jam
- Untuk membuat 1 potong kaos dengan kerah diperlukan waktu 2 jam
- Untuk membuat 1 potong jaket diperlukan waktu 5 jam
- Untuk membuat 1 potong celana panjang diperlukan waktu 4 jam

Berikut merupakan jumlah dari setiap jenis pakaian yang harus dibuat :

| Jenis Pakaian | Jumlah |
|---|---|
| Kemeja | 4 |
| Kaos tanpa kerah | 2 |
| Kaos dengan kerah | 5 |
| Jaket | 1 |
| Celana panjang | 2 |

Untuk permasalahan diatas, kita perlu mencari bagaimana pembagian kerja terhadap 3 mesin tersebut agar seluruh pekerjaan tersebut selesai dalam waktu yang paling minimum. Waktu penyelesaian seluruh pekerjaan merupakan **waktu paling lama yang dibutuhkan oleh ketiga mesin dalam mengerjakan pekerjaannya**.

Berikut merupakan contoh dari solusi yang mungkin dihasilkan :

Solusi Pertama :

| | Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|---|
| **Kemeja** | 2 | 1 | 1 |
| **Kaos tanpa kerah** | 1 | 1 | 0 |
| **Kaos dengan kerah** | 2 | 2 | 1 |
| **Jaket** | 1 | 0 | 0 |
| **Celana panjang** | 1 | 1 | 0 |
| **Total Waktu** | 20 jam | 12 jam | 5 jam |

Solusi kedua :

| | Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|---|
| **Kemeja** | 2 | 1 | 1 |
| **Kaos tanpa kerah** | 0 | 1 | 1 |
| **Kaos dengan kerah** | 2 | 2 | 1 |
| **Jaket** | 0 | 0 | 1 |
| **Celana panjang** | 1 | 1 | 0 |
| **Total Waktu** | 14 jam | 12 jam | 11 jam |

Waktu yang diperlukan solusi 1 untuk mengerjakan seluruh perkerjaan adalah max(20,12,5) yaitu 20 jam.

Waktu yang diperlukan solusi 2 untuk mengerjakan seluruh perkerjaan adalah max(14,12,11) yaitu 14 jam.

Berdasarkan 2 solusi yang telah dijabarkan, solusi yang terbaik adalah Solusi 2. Tujuan dari permasalahan ini adalah pencarian pembagian kerja terhadap mesin sehingga waktu yang diperlukan untuk mengerjakan seluruh pekerjaan menjadi minimum.

## IV. PENYELESAIAN PERSOALAN

Permasalahan optimasi ini secara umum memiliki struktur persoalan seperti berikut :

R : { R1, R2, R3, … , RN }
T : { T1, T2, T3, … , TN }
X : { X1, X2, X3, … , XN }

R : himpunan jenis pekerjaan dengan jumlah N
T : himpunan waktu proses dari pekerjaan R dengan jumlah N
X : himpunan jumlah dari pekerjaan R yang harus diselesaikan.

S : { B1, B2, B3, … , BM }

S : himpunan mesin pemroses yang identik dengan jumlah M.

Untuk persoalan yang telah dijabarkan pada penjelasan sebelumnya, struktur persoalannya adalah

R : { Kemeja, Kaos tanpa kerah, Kaos dengan kerah, Jaket, Celana Panjang }
T : { 3, 1, 2, 5, 4 }
X : { 4, 2, 5, 1, 2 }
S : { Mesin 1, Mesin 2, Mesin 3 }

Ada berbagai alternatif dalam menyelesaikan masalah ini, salah satunya menggunkan algoritma *brute force* (*exhaustive search*) dan algoritma *greedy*.

Dengan menggunakan algoritma brute force, kita mencacah seluruh kemungkinan solusi yang ada. Dari seluruh kemungkinan solusi tersebut, kita memilih solusi yang memiliki waktu pengerjaan pekerjaan yang paling minimum.

Keterangan :
A : Kemeja
B : Kaos tanpa kerah
C : Kaos dengan kerah
D : Jaket
E : Celana Panjang

<table>
<tr><th>Solusi</th><th>Waktu pengerjaan</th></tr>
<tr><td>Solusi ke-1<br><table><tr><th></th><th>M 1</th><th>M 2</th><th>M 3</th></tr><tr><td>A</td><td>4</td><td>0</td><td>0</td></tr><tr><td>B</td><td>2</td><td>0</td><td>0</td></tr><tr><td>C</td><td>5</td><td>0</td><td>0</td></tr><tr><td>D</td><td>1</td><td>0</td><td>0</td></tr><tr><td>E</td><td>1</td><td>0</td><td>0</td></tr><tr><td>Total</td><td>37</td><td>0</td><td>0</td></tr></table></td><td>37 jam</td></tr>
<tr><td>Solusi ke-2<br><table><tr><th></th><th>M 1</th><th>M 2</th><th>M 3</th></tr><tr><td>A</td><td>4</td><td>0</td><td>0</td></tr><tr><td>B</td><td>2</td><td>0</td><td>0</td></tr><tr><td>C</td><td>5</td><td>0</td><td>0</td></tr><tr><td>D</td><td>1</td><td>0</td><td>0</td></tr><tr><td>E</td><td>0</td><td>1</td><td>0</td></tr><tr><td>Total</td><td>33</td><td>4</td><td>0</td></tr></table></td><td>33 jam</td></tr>
<tr><td>Solusi ke-3<br><table><tr><th></th><th>M 1</th><th>M 2</th><th>M 3</th></tr><tr><td>A</td><td>4</td><td>0</td><td>0</td></tr><tr><td>B</td><td>2</td><td>0</td><td>0</td></tr><tr><td>C</td><td>5</td><td>0</td><td>0</td></tr><tr><td>D</td><td>1</td><td>0</td><td>0</td></tr><tr><td>E</td><td>0</td><td>0</td><td>1</td></tr><tr><td>Total</td><td>33</td><td>0</td><td>4</td></tr></table></td><td>33 jam</td></tr>
<tr><td>……Solusi lainnya ……..</td><td>….. Waktu lainya …..</td></tr>
<tr><td>Solusi ke- n-1<br><table><tr><th></th><th>M 1</th><th>M 2</th><th>M 3</th></tr><tr><td>A</td><td>1</td><td>1</td><td>1</td></tr><tr><td>B</td><td>1</td><td>1</td><td>0</td></tr><tr><td>C</td><td>1</td><td>2</td><td>2</td></tr><tr><td>D</td><td>1</td><td>0</td><td>0</td></tr><tr><td>E</td><td>0</td><td>1</td><td>1</td></tr><tr><td>Total</td><td>11</td><td>12</td><td>11</td></tr></table></td><td>12 jam ( Solusi optimum )</td></tr>
<tr><td>Solusi ke-n<br><table><tr><th></th><th>M 1</th><th>M 2</th><th>M 3</th></tr><tr><td>A</td><td>0</td><td>0</td><td>12</td></tr><tr><td>B</td><td>2</td><td>0</td><td>0</td></tr><tr><td>C</td><td>5</td><td>0</td><td>0</td></tr><tr><td>D</td><td>1</td><td>0</td><td>0</td></tr><tr><td>E</td><td>1</td><td>0</td><td>0</td></tr><tr><td>Total</td><td>21</td><td>0</td><td>0</td></tr></table></td><td>21 jam</td></tr>
</table>

Bila terdapat n jenis pekerjaan dan setiap jenis pekerjaan ada n pekerjaan yang harus diselesaikan dan terdapat n mesin, maka kompleksitas algoritmanya adalah $O(n^n)$. Cara ini sangat tidak efisien untuk jumlah n yang besar. Dengan algoritma greedy, masalah ini dapat dipecahkan dengan cara yang efisien.

Untuk menyelesaikan persoalan ini, perlu ditentukan elemen-elemen dari algoritma *greedy*

1. Himpunan kandidat : himpunan kombinasi pekerjaan yang mungkin terjadi
2. Himpunan solusi : kombinasi pembagian pekerjaan dengan syarat seluruh pekerjaan harus dikerjakan.
3. Fungsi seleksi : masukkan pekerjaan yang paling lama kedalam mesin yang paling cepat menyelesaikan pekerjaannya.
4. Fungsi layak : memeriksa apakah total pekerjaan seluruh mesin sama dengan total pekerjaan yang ada.
5. Fungsi obyektif : waktu minimum yang digunakan untuk menyelesaikan pekerjaan

Algoritma ini bekerja dengan **menempatkan pekerjaan yang membutuhkan waktu paling lama terhadap mesin yang paling cepat menyelesaikan pekerjaannya**. Untuk itu, diperlukan himpunan pekerjaan P yang telah **terurut mengecil berdasarkan lama prosesnya**.

Berikut adalah Skema umum untuk persoalan ini :

1. Pilih pekerjaan yang prosesnya paling lama
2. Cari mesin yang waktu total pengerjaan pekerjaannya paling cepat
3. Tambahkan pekerjaan yang telah didapat kedalam mesin teersebut.
4. Ulangi langkah 1 sampai pekerjaannya telah habis dibagi.

Berikut merupakan Algoritma umum untuk persoalan ini :

```
{ Array P sudah terurut mengecil berdasarkan
  lama proses pekerjaan }
function greedy(P : array of pekerjaan)
: array of Mesin

Kamus
  M : array of mesin
  MesinMin : pekerjaan
  i : integer

Algoritma
  { Mesin belum punya pekerjaan }
  InisialisasiKonsisiMesin()
  for (i = 0; i < n; i++)
      { Cari mesin yang paling cepat
        menyelesaikan pekerjaan }
      { Tambahkan pekerjaan }
      add(min(M), P[i])
  endfor
  return M
```

Himpunan pekerjaan P adalah

P : { D, E, E, A, A, A, A, C, C, C, C, C, B, B }

Pengisian pekerjaan :

P : { D, E, E, A, A, A, A, C, C, C, C, C, B, B }
Langkah – 1 :

| Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|
| D | | |
| Mesin 1, 2, 3 : 5, 0, 0 | | |

P : { E, E, A, A, A, A, C, C, C, C, C, B, B }
Langkah – 2 :

| Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|
| D | E | |
| Mesin 1, 2, 3 : 5, 4, 0 | | |

P : { E, A, A, A, A, C, C, C, C, C, B, B }
Langkah – 3 :

| Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|
| D | E | E |
| Mesin 1, 2, 3 : 5, 4, 4 | | |

P : { A, A, A, A, C, C, C, C, C, B, B }
Langkah – 4 :

| Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|
| D | E | E |
| | A | |
| Mesin 1, 2, 3 : 5, 7, 4 | | |

P : { A, A, C, C, C, C, C, B, B }
Langkah – 5 :

| Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|
| D | E | E |
| | A | A |
| Mesin 1, 2, 3 : 5, 7, 7 | | |

P : { A, C, C, C, C, C, B, B }
Langkah – 6 :

| Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|
| D | E | E |
| A | A | A |
| Mesin 1, 2, 3 : 8 7, 7 | | |

.....
.....

P : { B }
Langkah ke –n :

| Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|
| D | E | E |
| A | A | A |
| C | C | C |
| B | C | C |
| | B | |

Berdasarkan algoritma tersebut, didapatkan hasil pembagian pekerjaan sebagai berikut

| | Mesin 1 | Mesin 2 | Mesin 3 |
|---|---|---|---|
| **Kemeja** | 1 | 1 | 1 |
| **Kaos tanpa kerah** | 1 | 1 | 0 |
| **Kaos dengan kerah** | 1 | 2 | 2 |
| **Jaket** | 1 | 0 | 0 |
| **Celana panjang** | 0 | 1 | 1 |
| **Total Waktu** | 11 jam | 12 jam | 11 jam |

Ternyata waktu minimum yang diperlukan agar semua baju dapat selesai adalah 12 jam.

Algoritma greedy ini akan selalu memberikan solusi yang paling efektif dalam persoalan ini, meskipun alaupun algoritma ini tidak dapat menemukan semua kemungkinan solusi efektif. Kompleksitas waktu algoritma greedy adalah $O(n^2)$ sehingga algoritma ini jauh lebih mangkus disbanding pencarian dengan *brute force* pada kasus ini.

## V. KESIMPULAN

Melalui penjelasan diatas, dapat disimpulkan bahwa algoritma greedy sangat cocok untuk diterapkan pada persoalan pembagian kerja pada mesin identik. Hal ini dikarenakan kompleksitas waktu *greedy* jauh lebih baik dibandingkan kompleksitas waktu dari algoritma *exhaustive search*. Dengan adanya algoritma ini, industri dapat mengoptimalkan pembagian kerja pada mesin sehingga keuntungan yang didapat pun makin besar.

## VII. ACKNOWLEDGMENT

Pertama-tama saya ingin berterima kasih kepada Tuhan karena melalui anugrah-Nya saya bisa menyelesaikan makalah ini. Saya juga sangat berterima kasih kepada dosen saya Dr. Ir. Rinaldi Munir dan Dr.Nur Ulfa Maulidevi, karena telah memberikan inspirasi bagi saya. Saya juga berterima kasih kepada Institut Teknologi Bandung, tempat dimana saya menuntut ilmu ketika saya menyelesaikan makalah ini.

## REFERENCES

[1] Munir, Rinaldi, Diktat Kuliah Strategi Algoritma 3 rd, Bandung: Institut Teknologi Bandung.

## 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 2015

William Sentosa / 13513026