Penerapan Algoritma Greedy pada Penjadwalan Produksi Single-Stage dengan Parallel Machine di Industri Konveksi

Ahmad Juniar
Jurusan Sistem Informasi, Sekolah Tinggi Manajemen Industri – Kementerian Perindustrian
Jl. Letjen Suprapto 26, Cempaka Putih, Jakarta 10510, telp: (021) 42886064, fax: (021) 42888206
ahmadjuniar@gmail.com

Abstrak

Industri konveksi adalah perusahaan yang memproduksi berbagai macam pakaian jadi seperti kaos, kemeja, celana panjang. Dari sisi pembuatan produk, industri ini bertipe make to order (MTO), artinya produk akan dibuat jika ada pesanan pelanggan. Setiap pesanan yang diterima sangat bervariasi dalam hal jenis pakaian yang akan diproduksi serta jumlahnya sehingga perlu dilakukan penjadwalan produksi pada setiap pesanan. Dalam penyusunan jadwal produksi, perusahaan harus mampu mengalokasikan setiap pekerjaan yang beragam ke dalam stasiun kerja (mesin jahit dan operator) secara seimbang agar menghasilkan minimum makespan (total waktu penyelesaian pekerjaan yang minimal). Penelitian ini bertujuan untuk menyusun jadwal produksi single-stage pada mesin paralel agar menghasilkan makespan yang minimal menggunakan algoritma greedy. Dari hasil penelitian, algoritma greedy selalu menghasilkan solusi optimal untuk kasus ini. Selain itu, algoritma greedy selalu paling cepat dalam menghasilkan solusi dibandingkan algoritma exhaustive search.

Kata kunci*— penjadwalan produksi di mesin paralel, algoritma greedy, makespan*

Abstract

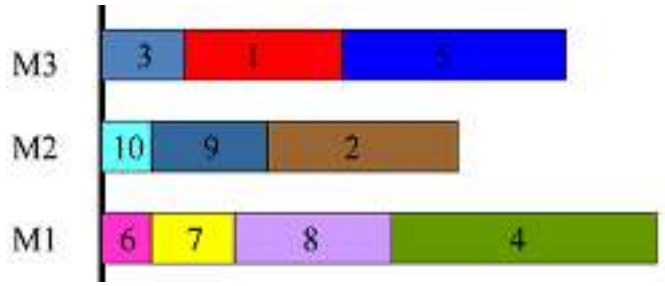
Garmen industry is a company that produces a wide range of apparel such as t-shirts, shirts, trousers. Based on manufacture of products, type of this industry is make to order (MTO), means that product will be made if there are customers order. Any orders received vary greatly in terms of the type of clothes and amount of clothes in every production scheduling. In preparation of production schedule, the company should be able to allocate varied jobs in several parallel working station (sewing machine and operator) in balance workload in order to produce a minimum makespan (total time of job completion is minimal). This study aims to develop a single-stage production schedule on a parallel machine that produces minimal makespan with greedy algorithm. From the research, greedy algorithm always produces an optimal solution for this case. In addition, the greedy algorithm is faster than exhaustive search algorithms in obtaining solution.

Keywords*— job scheduling in parallel machine, greedy algorithm, makespan*

1. PENDAHULUAN

Dalam dunia industri, keefisienan waktu sangatlah penting. Semakin efisien proses produksi, semakin maksimal jumlah produksi yang dihasilkan atau semakin minimum biaya yang harus dikeluarkan. Perusahaan konveksi adalah suatu perusahaan yang menghasilkan pakaian jadi, seperti kemeja, celana, kaos serta jaket. Setiap jenis pakaian memerlukan waktu penyelesaian yang berbeda-beda, misalnya; produksi kaos lebih cepat dibandingkan produksi jaket. Industri konveksi menggunakan operator dan mesin jahit untuk setiap stasiun kerjanya. Dari sisi pembuatan produknya, perusahaan konveksi termasuk dalam kategori *make to order* (MTO) karena produk akan dibuat jika ada pesanan dari pelanggan. Setiap pesanan dari pelanggan sangat bervariasi dari jenis pakaian yang dibuat maupun jumlahnya. Fokus penelitian ini adalah menyelesaikan permasalahan penjadwalan

produksi pakaian jadi di perusahaan konveksi dalam rangka meminimumkan *makespan* (total waktu penyelesaian pekerjaan) untuk setiap pesanan (*job order*) yang diterima dari pelanggan.

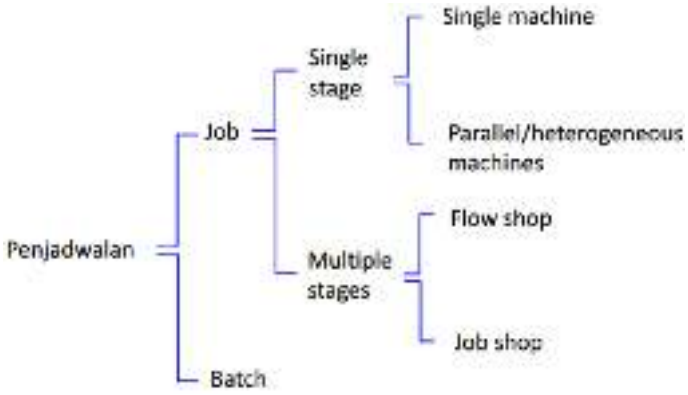


Gambar 1. Penjadwalan *single-stage* pada *parallel machine*

Berdasarkan pesanan, perusahaan konveksi harus menentukan penjadwalan produksi yaitu berupa urutan jenis produk apa saja dan berapa jumlah yang harus dikerjakan oleh setiap stasiun kerja agar setiap stasiun kerja seperti ditunjukan Gambar 1, memiliki beban kerja yang seimbang. Semakin seimbang pembagian beban kerja di setiap stasiun kerja, maka *minimum makespan* akan tercapai. Permasalahan ini dapat diselesaikan dengan menggunakan algoritma greedy, salah satu algoritma yang banyak dipakai dalam menyelesaikan persoalan optimasi. Algoritma ini akan menghasilkan pembagian kerja yang efisien dengan kompleksitas waktu yang jauh lebih baik dari algoritma *exhaustive search*.

2. TINJAUAN PUSTAKA

2.1 Penjadwalan Produksi



Gambar 2. Penjadwalan Produksi

Penjadwalan produksi adalah pengalokasian sumber daya yang terbatas untuk mengerjakan sejumlah pekerjaan [1]. Penjadwalan produksi terbagi atas beberapa kelompok seperti ditunjukkan pada Gambar 2 [2]. Penjadwalan *single-stage* adalah penjadwalan setiap pekerjaan (*job*) hanya melewati satu stasiun kerja saja untuk menghasilkan produk. Jika pekerjaan dilayani oleh lebih dari satu stasiun kerja maka disebut *single-stage* in *parallel machines,* contoh: antrian pelayanan di *teller* bank.

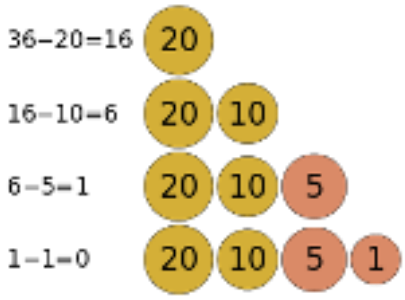
Penjadwalan *multi-stage* adalah penjadwalan setiap *job* yang harus melewati beberapa stasiun kerja untuk sebelum menghasilkan produk jadi. Penjadwalan *flow shop* adalah proses penjadwalan *job-job* yang memiliki urutan pengerjaan yang sama saat melewati beberapa stasiun kerja. Sedangkan penjadwalan *job shop* adalah proses penjadwalan *job-job* yang memiliki urutan pengerjaan yang tidak sama. Penjadwalan *job shop* biasanya digunakan untuk menjadwalkan pekerjaan yang beragam dengan menggunakan fasilitas yang sama.

Penjadwalan *job* adalah penjadwalan untuk memecahkan masalah urutan saja, karena ukuran *job* telah diketahui sedangkan penjadwalan *batch* adalah penjadwalan untuk memecahkan masalah penentuan ukuran *batch* dan masalah urutan secara simultan [2].

2.2 Algoritma Greedy

Algoritma greedy merupakan salah satu algoritma untuk memecahkan persoalan optimasi. Persoalan optimasi yang dimaksud adalah persoalan mencari solusi optimum yaitu maksimasi (*maximization*) ataupun minimasi (*minimization*). Arti *greedy* bila diterjemahkan secara harafiah memiliki arti rakus atau tamak. Algoritma ini mencerminkan prinsip *greedy* 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 [3]. Prinsip greedy adalah "*take what you can get now!*" yaitu:

1. Pada setiap langkah, algoritma membuat pilihan optimal lokal.
2. Langkah tersebut dilakukan dengan harapan bahwa langkah sisanya mengarah ke solusi optimal global [4].



Gambar 3. Penerapan algoritma greedy pada penukaran uang logam

Salah satu contoh penerapan algoritma greedy adalah permasalahan meminimalkan jumlah uang recehan logam sebagai hasil penukaran terhadap uang kertas seperti ditunjukan oleh Gambar 3 Jika tersedia uang pecahan logam senilai 1, 5, 10 dan 20, maka uang kertas senilai 36 dapat ditukar dengan cara berikut:

Penetapan strategi greedy yaitu: pada setiap langkah, pilih koin dengan nilai terbesar dari himpunan koin yang tersisa, sehingga langkah yang diambil adalah sebagai berikut:

1. Langkah 1: pilih 1 buah koin bernilai 20 (Total = 20)
2. Langkah 2: pilih 1 buah koin bernilai 10 (Total = 20 + 10 = 30)
3. Langkah 3: pilih 1 buah koin bernilai 5 (Total = 20 + 10 + 5 = 35)
4. Langkah 4: pilih 1 buah koin bernilai 1 (Total = 20 + 10 + 5 + 1 = 36)

Solusi: Jumlah koin minimum = 4 (solusi optimal minimum)

Algoritma greedy disusun oleh elemen-elemen berikut [3]:

1. Himpunan kandidat berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi berisi kandidat-kandidat yang terpilih sebagai solusi dari permasalahan
3. Fungsi seleksi adalah pemilihan kandidat yang paling memungkinkan untuk mencapai solusi optimal. Kandidat yang sudah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.
4. Fungsi kelayakan adalah pemeriksaan apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi.
5. Fungsi obyektif merupakan fungsi untuk memaksimumkan atau meminimumkan nilai solusi (misalnya memaksimalkan keuntungan penjualan, meminimumkan biaya produksi dan lain-lain).

```
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 U {x}) then
         S←S U {x}
      endif
   endwhile
{SOLUSI(S) sudah diperoleh or C = {} }
```

Gambar 4. Algoritma greedy dalam bentuk *pseudo code*

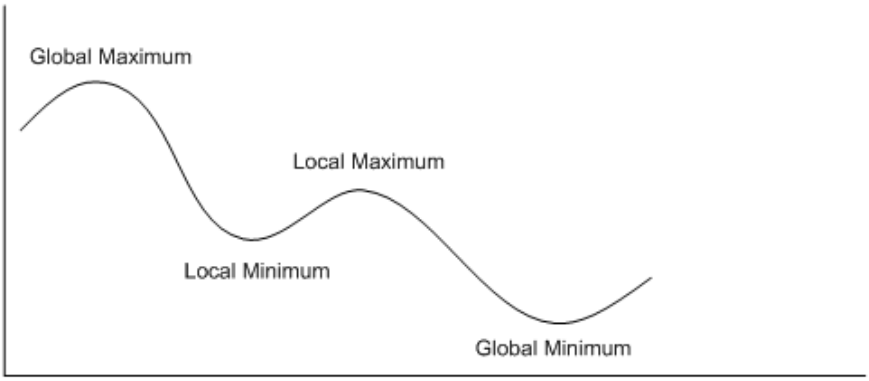
Algoritma greedy tidak selalu memberikan solusi optimal karena 2 hal berikut [5]:

1. Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada sebagaimana pada metode *exhaustive search*.
2. Adanya beberapa alternatif fungsi seleksi. jika menginginkan agar algoritma bekerja untuk menghasilkan solusi yang mendekati optimal, maka perlu memilih fungsi seleksi yang tepat.

Pada contoh permasalahan penukaran uang kertas ke uang recehan logam, solusi tidak optimal dapat dilihat [5]. Jika terdapat uang recehan koin senilai 10, 7, 1 dan uang kertas yang akan ditukar adalah 15, maka:

- Solusi greedy: 15 = 10 + 1 + 1 + 1 + 1 + 1 (6 koin)
- Sedangkan solusi optimal: 15 = 7 + 7 + 1 (hanya 3 koin)

Hal tersebut terjadi karena pada fungsi seleksi, algoritma greedy akan mengambil nilai uang koin terbesar terlebih dahulu yaitu uang koin bernilai 10 sebelum mengambil uang koin lainnya. Solusi yang didapat bisa berupa solusi *local-optimum* atau mendekati optimal seperti ditunjukkan pada Gambar 5.



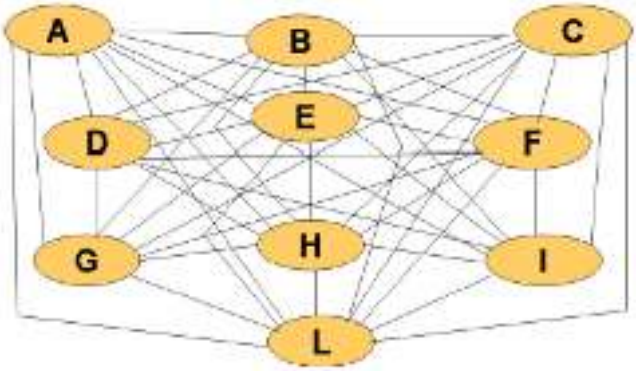
Gambar 5. *Local optimum* dan *global optimum*

2.3 Algoritma Exhaustive search

Exhaustive search adalah teknik pencarian solusi secara *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 [6].

Langkah-langkah metode *exhaustive search* adalah sebagai berikut [6]:

1. Enumerasi 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.



Gambar 6. Representasi *exhaustive search* dalam bentuk *graph*

Gambar 6 menunjukkan cara *exhaustive search* bekerja. *Exhaustive search* mencacah semua kemungkinan yang ada, lalu membandingkannya satu sama lain. Algortima ini pasti akan selalu menghasilkan solusi optimal karena algoritma ini mencacah seluruh kemungkinan solusi. Namun, waktu dan sumberdaya yang dibutuhkan oleh algoritma ini sangat besar sehingga untuk permasalahan yang besar algoritma ini jarang digunakan.

3. METODOLOGI PENELITIAN

Tahapan penelitian ini ditunjukkan oleh Gambar 7 dengan penjelasan sebagai berikut:

1. Identifikasi permasalahan: mendefinisikan permasalahan yang dihadapi industri konveksi. Salah satu permasalahan pada industri konveksi adalah masalah penjadwalan setiap pesanan (*job order*) dari pelanggan. Setiap pelanggan biasanya memesan jenis pakaian dan jumlah yang beragam sehingga perlu dilakukan penjadwalan untuk setiap pesanan (*job order*) agar pengalokasian beban kerja menjadi seimbang pada setiap stasiun kerja (mesin jahit dan operator).
2. Tujuan penelitian: menjadwalkan setiap pesanan yang diterima dalam rangka meminimalkan *makespan* (total waktu penyelesaian pekerjaan) agar tercipta efisiensi waktu produksi.
3. Analisis dan pembahasan: menganalisis permasalahan penjadwalan *single-stage* pada mesin parallel dengan algoritma greedy dan membandingkannya dengan algoritma *exhaustive search* dalam menghasilkan solusi optimal dan efisiensi algoritmanya. Data yang digunakan dalam penelitian ini adalah data *dummy* namun dapat mewakili kasus pada dunia nyata.
4. Implementasi: mengimplementasikan algoritma greedy dalam penjadwalan *single-stage* pada mesin paralel dengan bahasa pemrograman Java.
5. Kesimpulan: menetapkan kesimpulan dari hasil penelitian.

4. HASIL DAN PEMBAHASAN

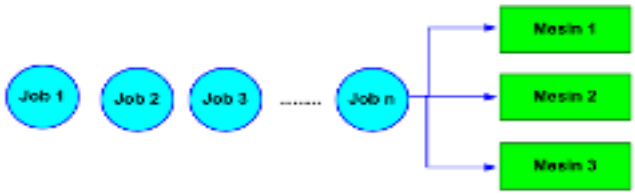
4.1 Kasus Penjadwalan Produksi

Berikut ini ilustrasi permasalahan pada penjadwalan produksi *single-stage* pada mesin parallel. Misalkan terdapat 5 (lima) jenis pekerjaan untuk membuat pakaian, jumlahnya serta waktu penyelesaian untuk setiap *job* seperti ditunjukkan oleh Tabel 1.

Tabel 1. Tabel Pekerjaan pada Penjadwalan Produksi Industri Konveksi

Jenis Job (*Pekerjaan*)	Kode *Job*	Jumlah	Waktu penyelesaian
Membuat kemeja	KM	4	3
Membuat topi	TP	2	1
Membuat kaos	KS	5	2
Membuat celana panjang	CP	2	4
Membuat jaket	JK	1	5

Pekerjaan dijadwalkan pada 3 (tiga) stasiun kerja (mesin jahit dan operator) seperti ditunjukkan pada Gambar 8. Penelitian ini mengasumsikan bahwa setiap operator penjahit memiliki *skill* yang sama sehingga setiap jenis pekerjaan akan diselesaikan dengan waktu yang sama.



Gambar 7. Ilustrasi penjadwalan *single-stage* pada mesin paralel

4.2 Penyelesaian Permasalahan Penjadwalan Produksi dengan Exhaustive Search

Algoritma *exhaustive search* akan mencacah seluruh kemungkinan solusi yang ada, kemudian memilih solusi yang memiliki waktu pengerjaan paling minimum. Berikut ini contoh penyelesaian dengan algoritma *exhaustive search*.

Tabel 2. Solusi ke 1

Job	Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
Membuat kemeja	4 buah	-	-
Membuat topi	2 buah	-	-
Membuat kaos	5 buah	-	-
Membuat celana panjang	2 buah	-	-
Membuat jaket	1 buah	-	-
Total	37 jam	0 jam	0 jam

Tabel 3. Solusi ke 2

Job	Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
Membuat kemeja	3 buah	1 buah	-
Membuat topi	2 buah	-	-
Membuat kaos	5 buah	-	-
Membuat celana panjang	2 buah	-	-
Membuat jaket	1 buah	-	-
Total	34 jam	3 jam	0 jam

Tabel 4. Solusi ke 3

Job	Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
Membuat kemeja	2 buah	1 buah	1 buah
Membuat topi	2 buah	-	-
Membuat kaos	5 buah	-	-
Membuat celana panjang	2 buah	-	-
Membuat jaket	1 buah	-	-
Total	31 jam	3 jam	3 jam

Tabel 5. Solusi ke x (optimal)

Job	Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
Membuat kemeja	1 buah	2 buah	1 buah
Membuat topi	1 buah	-	1 buah
Membuat kaos	2 buah	1 buah	2 buah
Membuat celana panjang	-	2 buah	1 buah
Membuat jaket	1 buah	-	-
Total	13 jam	12 jam	12 jam

Tabel 6. Solusi ke z-1

Job	Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
Membuat kemeja	-	1 buah	3 buah
Membuat topi	-	-	2 buah
Membuat kaos	-	-	5 buah
Membuat celana panjang	-	-	2 buah
Membuat jaket	-	-	1 buah
Total	0 jam	3 jam	34 jam

Tabel 7. Solusi ke z

Job	Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
Membuat kemeja	-	-	4 buah
Membuat topi	-	-	2 buah
Membuat kaos	-	-	5 buah
Membuat celana panjang	-	-	2 buah
Membuat jaket	-	-	1 buah
Total	0 jam	0 jam	37 jam

Bila terdapat total n pekerjaan yang dijadwalkan pada m mesin, maka kompleksitas algoritmanya adalah $O(n.2^n)$. Cara ini sangat tidak efisien untuk jumlah n yang besar. Dengan algoritma greedy, masalah ini dapat dipecahkan dengan cara yang lebih efisien.

4.3 Penyelesaian Permasalahan Penjadwalan Produksi dengan Algoritma Greedy

Berikut ini adalah penyelesaian permasalahan penjadwalan produksi dengan algoritma greedy.
Kondisi saat inisialisasi
C = {JK, CP, CP, KM, KM, KM, KM, KS, KS, KS, KS, KS, TP, TP} // Himpunan kandidat
S = { } // Himpunan solusi

Langkah ke-1: Hasil himpunan solusi sementara

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	-	-
Kumulatif beban stasiun kerja 1, 2, 3 = 5, 0, 0		

C = { CP, CP, KM, KM, KM, KM, KS, KS, KS, KS, KS, TP, TP}

Langkah ke-2: Hasil himpunan solusi sementara

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	CP	-
Kumulatif beban stasiun kerja 1, 2, 3 = 5, 4, 0		

C = { CP, KM, KM, KM, KM, KS, KS, KS, KS, KS, TP, TP}

Langkah ke-3: Hasil himpunan solusi sementara

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	CP	CP
Kumulatif beban stasiun kerja 1, 2, 3 = 5, 4, 4		

C = { KM, KM, KM, KM, KS, KS, KS, KS, KS, TP, TP}

Langkah ke-4: Hasil himpunan solusi sementara

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	CP	CP
-	KM	-
Kumulatif beban stasiun kerja 1, 2, 3 = 5, 7, 4		

C = { KM, KM, KM, KS, KS, KS, KS, KS, TP, TP}

Langkah ke-5: Hasil himpunan solusi sementara

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	CP	CP
-	KM	KM
Kumulatif beban stasiun kerja 1, 2, 3 = 5, 7, 7		

C = { KM, KM, KS, KS, KS, KS, KS, TP, TP}

Langkah ke-6: Hasil himpunan solusi sementara

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	CP	CP
KM	KM	KM
Kumulatif beban stasiun kerja 1, 2, 3 = 8, 7, 7		

C = { KM, KS, KS, KS, KS, KS, TP, TP}

Langkah ke-7: Hasil himpunan solusi sementara

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	CP	CP
KM	KM	KM
-	KM	-
Kumulatif beban stasiun kerja 1, 2, 3 = 8, 10, 7		

C = { KS, KS, KS, KS, KS, TP, TP}

Langkah ke-8: Hasil himpunan solusi sementara

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	CP	CP
KM	KM	KM
-	KM	KS
Kumulatif beban stasiun kerja 1, 2, 3 = 8, 10, 9		

C = { KS, KS, KS, KS, TP, TP}

Langkah ke-n: Hasil himpunan solusi akhir (n = 14)

Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
JK	CP	CP
KM	KM	KM
KS	KM	KS
KS	KS	KS
TP	-	TP
Kumulatif beban stasiun kerja 1, 2, 3 = 13, 12, 12		

C = { } // Himpunan kandidat telah kosong, iterasi berakhir

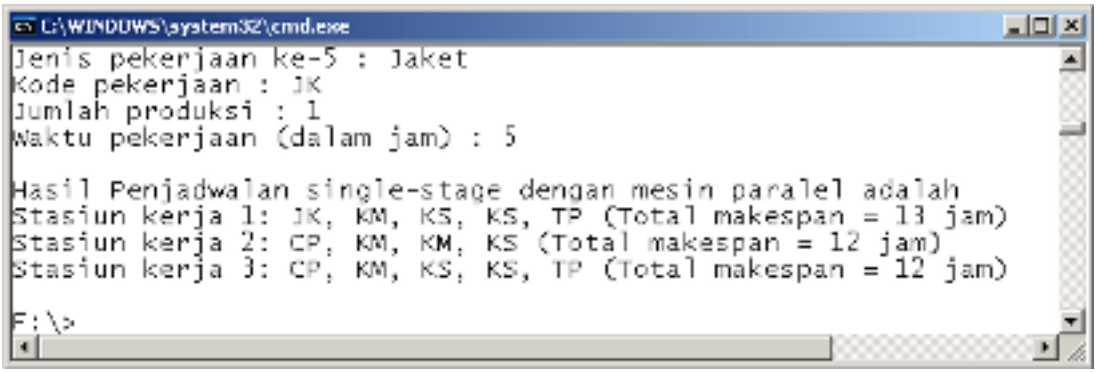
Dengan algoritma greedy maka hasil penjadwalan produksi dihasilkan dengan total waktu penyelesaian pekerjaan = max(13, 12, 12) = 13 jam. Algoritma greedy untuk kasus ini mempunyai kompleksitas waktu O(n). Berdasarkan pengujian dengan metode *exhaustive search*, algoritma greedy selalu menghasilkan solusi optimal minimum dan waktu penyelesaiaan yang lebih cepat dibanding metode *exhaustive search*. Hasil akhir algoritma greedy ditunjukkan oleh Table 8.

Tabel 8. Solusi algoritma greedy

Job	Stasiun kerja 1	Stasiun kerja 2	Stasiun kerja 3
Membuat kemeja	1 buah	2 buah	1 buah
Membuat topi	1 buah	-	1 buah
Membuat kaos	2 buah	1 buah	2 buah
Membuat celana panjang	-	2 buah	1 buah
Membuat jaket	1 buah	-	-
Total	13 jam	12 jam	12 jam

4.4 Implementasi Penjadwalan Produksi dengan Bahasa Pemrograman Java

Penjadwalan produksi *single-stage* pada mesin paralel menggunakan algoritma greedy diimplementasi dalam bahasa pemrograman java dengan editor Oracle Netbeans versi 7.1. *Output* program dapat dilihat pada pada Gambar 8.



Gambar 8. Impementasi Penjadwalan Produksi Dengan Bahasa Pemrograman Java

4. KESIMPULAN

Dari hasil pembahasan sebelumnya, maka dapat disimpulkan bahwa:

1. Algoritma greedy dapat menyelesaikan permasalahan penjadwalan produksi *single-stage* pada mesin paralel.
2. Untuk setiap kasus penjadwalan produksi *single-stage* di mesin paralel, algoritma greedy selalu menghasilkan solusi optimal minimum dengan waktu proses lebih cepat dibandingkan metode *exhaustive search*.

5. SARAN

Dari hasil penelitian ini, dapat dikembangkan penelitian lanjutkan, antara lain:

1. Penyelesaian persoalan penjadwalan produksi *multi-stage* baik berjenis *flowshop* ataupun *jobshop* dengan algoritma greedy.
2. Implementasi algoritma greedy dalam bentuk aplikasi java berbasis *graphical user interface* dengan tambahan grafik *gantt chart* agar *output* aplikasi yang dihasilkan menjadi lebih menarik.

DAFTAR PUSTAKA

[1] Sipper, D. and Bulfin Jr., *Production Planning, Control, and Integrations*, 1997, McGraw Hill, New York
[2] Bedworth D.D. and Bailey J.E., *Integrated Production Control System*, 1987, John Wiley & Sons, New York.
[3] Cormen, T.H, Leiserson, C.E, Rivest, R.L, and Stein, C, 2001, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts.
[4] Purnomo, A.C., Yuliana, M., Prasetyaningrum, I., 2014, Implementasi algoritma Greedy pada layanan call center taksi wisata berbasis web, EEPIS-Online Politeknik Elektronika Negeri, Surabaya: http://www.pens.ac.id/post/20130813144934-1292
[5] Munir, R, 2009, *Diklat Kuliah: Strategi Algoritma*, Penerbit ITB, Bandung.
[6] Christof, P., Pelzl, J., Preneel B., 2010, *Understanding Cryptography: A Textbook for Students and Practitioners*, Ed. 1, Springer, New York.