DINAMIKA TEKNIK
Vol. II, No. 1
Januari 2008
14 - 23

PERANCANGAN DAN PEMBUATAN SISTEM INFORMASI PENJADWALAN PRODUKSI DENGAN METODE HEURISTIK GENERATE AND TEST

Antono Adhi, Zuliyati
Dosen Fakultas Teknik dan Dosen Ekonomi Universitas Stikubank Semarang

Abstract

Sequencing and scheduling are forms of decision-making which play a crucial role in manufacturing as well as in service industries. In the current competitive environment, effective sequencing and scheduling has become a necessity for survival in the market-place. Companies have to meet shipping dates committed to the customers, as failure to do so may result in a significant loss of good will. They also have to schedule activities in such a way as to use the resources available in an efficient manner. Scheduling concerns the allocation of limited resources to tasks or jobs over time. It is a decision-makin process that has as a goal the optimization of one or more objectives. Generate and Test Algorithm is used as a model for scheduling jobs to get information of optimum Make Span. Because of Generate and Test try all of possibility of sequence jobs, this information system is based on computer to process faster.

Keywords : Scheduling, Generate and Test Algorithm

A. PENDAHULUAN

Pada perusahaan-perusahaan dengan sistem *job shop* atau perusahaan yang berproduksi setiap kali ada pesanan pelanggan, dibutuhkan penjadwalan proses produksi, *job* atau pesanan yang mana yang akan diproduksi terlebih dahulu. Penjadwalan digunakan untuk mengalokasikan waktu proses mesin atau aktivitas untuk setiap *job*. Untuk setiap kemungkinan urutan *job* akan didapatkan Make Span yang berbeda-beda. Tujuan utama dalam penjadwalan produksi adalah mendapatkan Make Span (total waktu penyelesaian seluruh job) paling optimum sehingga akan memperkecil waktu menganggur sumber daya mesin dan tenaga kerja.

Penjadwalan yang tidak optimum berarti mengurutkan *job-job* mana yang akan dikerjakan terlebih dahulu dan menghasilkan Make Span yang tidak optimum. Hal ini akan menghasilkan alokasi *job-job* pada mesin atau aktivitas yang tidak efektif karena

menyebabkan waktu penyelesaian seluruh *job* akan lebih panjang dari semestinya dan sumber daya mesin serta tenaga kerja tidak menghasilkan produktivitas yang baik karena seringnya terjadi waktu menganggur.

B. PENJADWALAN PRODUKSI

Fungsi dalam penjadwalan produksi :

- *Loading* (pembebanan) untuk mengkompromikan kebutuhan yang diminta dengan kapasitas yang ada. Pembebanan diimplementasikan dengan menentukan berapa lama sebuah job diproses oleh sebuah mesin. Nilai ini bisa didapatkan dengan mengalikan waktu pengerjaan unit satuan *job* dengan besar *job*.
- *Sequencing* (penentuan urutan) untuk membuat prioritas pengerjaan dalam pemrosesan *job-job* yang masuk. Banyak metode yang digunakan untuk mendapatkan urutan pengerjaan *job-job* seperti Algoritma *Johnson* dan *Branch and Bound*. Penentuan urutan dilakukan untuk mendapatkan *Make Span* paling optimum. Metode *Generate and Test* dapat digunakan untuk mendapatkan *Make Span* paling optimum.
- *Dispatching* untuk memberikan perintah kerja ke tiap mesin atau fasilitas lainnya.
- Pengendalian kinerja penjadwalan dengan :
  - Memonitor perkembangan pencapaian pemenuhan order dalam semua sektor.
  - Merancang ulang sequencing bila ada kesalahan atau skala prioritas baru.
- *Updating schedules*

Hal-hal yang harus diperhatikan dalam penjadwalan produksi adalah parameter input penjadwalan produksi :

- Teknologi pemrosesan. Metode apa yang digunakan untuk mendapatkan jadwal yang optimum.
- Limit kapasitas. Berapa kapasitas mesin dan aktivitas.

- Rencana agregat untuk :
  - Persediaan
  - Jumlah tenaga kerja
  - Batasan lembur, sub kontrak, dll.
- Kebutuhan pemeliharaan.
- Kelayakan dan jumlah persediaan antar tingkat.

Variabel keputusan penjadwalan produksi memuat :

- Kuantitas pasti tenaga kerja yang digunakan harian.
- Setting *adjustable* tingkat produksi aktual untuk *overtime* dan *undertime*.
- Alokasi spesifik *job* ke sumber daya.
- *Sequencing, time phasing* dari pesanan sampai unit produksi.

C. GENERATE AND TEST

Metode *Generate and Test* adalah metode yang membangkitkan (*generate*) semua kemungkinan urutan pengerjaan *job* yang ada. Untuk setiap kemungkinan urutan *job* dihitung *Make Span*-Nya. Jika hasilnya lebih kecil dari *Make Span* optimum sebelumnya, maka urutan *job* tersebut dipilih sebagai urutan yang paling optimum. Demikian hal ini dilakukan sampai seluruh kemungkinan urutan *job* dihitung. Algoritma *Generate and Test* adalah :

1. Bangkitkan suatu kemungkinan solusi (membangkitkan suatu titik tertentu atau lintasan tertentu dari keadaan awal).
2. Uji untuk melihat apakah node tersebut benar-benar merupakan solusinya dengan cara membandingkan node tersebut atau node akhir dari suatu lintasan yang dipilih dengan kumpulan tujuan yang diharapkan.
3. jika solusi ditemukan, keluar. Jika tidak, ulangi kembali langkah yang pertama.

Dengan membangkitkan seluruh kemungkinan urutan *job*, maka jumlah seluruh kemungkinan yang ada adalah N! (N factorial). Di mana N adalah jumlah *job* yang

ada. Contoh jika ada 3 *job* (*job* 1, 2 dan 3), maka ada 3! kemungkinan urutan *job* yaitu :

1. 1-2-3
2. 1-3-2
3. 2-1-3
4. 2-3-1
5. 3-1-2
6. 3-2-1

D. ALOKASI JOB PADA MESIN

Setiap *job* mempunyai waktu pengerjaan yang berbeda dibanding dengan *job* lain. Hal ini tergantung seberapa banyak dan seberapa rumit pengerjaan *job* pada suatu mesin atau aktivitas. Contohnya pekerjaan pemotongan, bubut, pengukiran dan finishing almari pabrik meubel untuk pesanan PT HIJ yang banyak dan rumit dengan ukiran mempunyai waktu pekerjaan lebih lama dibanding dengan pesanan PT XYZ yang selalu memesan jenis almari sederhana dan lebih sedikit. Alokasi *job* pada mesin ditampilkan tabel satuan waktu. Satuan waktu bisa dalam bentuk menit, jam, hari, bulan, dsb. Tabel 1 menunjukkan contoh alokasi waktu *job 1*, *job 2*, *job 3* pada mesin 1, mesin 2 dan mesin 3.

Tabel 1 Alokasi *job* pada mesin

MESIN	JOB		
	1	2	3
1	7.5	5.3	4.6
2	2.2	3.4	3.1
3	4.6	6.9	6.2

Dari tabel di atas terlihat bahwa waktu pengerjaan *job2* pada mesin 3 adalah 6.9 satuan waktu.

E. MAKE SPAN

Make Span adalah total waktu pengerjaan seluruh *job*. *Make Span* dapat diperoleh pada waktu pengerjaan *job* terakhir pada mesin atau aktivitas terakhir. Waktu diselesaikannya *job* ke i sampai pada mesin ke j adalah :

$$T_{i,j} = MAX\ (T_{i-1,j}\ ;\ T_{i,j-1}) + A_{i,j}$$

Di mana

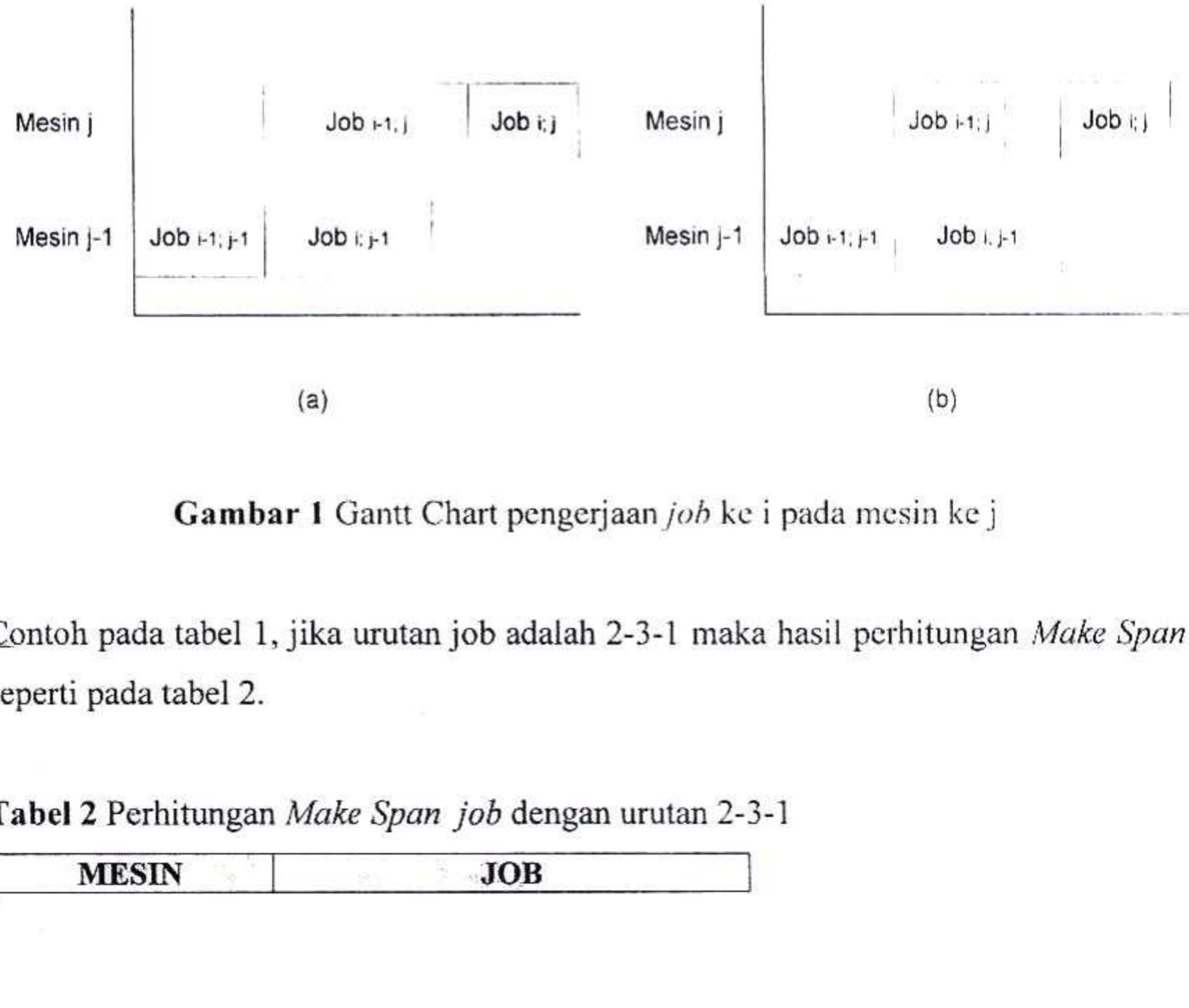
$T_{i,j}$ = waktu diselesaikannya *job* ke i sampai pada mesin ke j

$T_{i-1,j}$ = waktu diselesaikannya *job* ke i-1 sampai pada mesin ke j

$T_{i,j-1}$ = waktu diselesaikannya *job* ke i sampai pada mesin ke j-1

$A_{i,j}$ = alokasi waktu diselesaikannya *job* ke i pada mesin ke j

Parameter MAX ($T_{i-1,j}$; $T_{i,j-1}$) dimasukkan karena pengerjaan *job* ke i pada mesin ke j mulai dilakukan setelah pengerjaan *job* ke i-1 pada mesin j seperti pada gambar 1a atau setelah pengerjaan *job* ke i pada mesin ke j-1 seperti pada gambar 1b.



Gambar 1 Gantt Chart pengerjaan *job* ke i pada mesin ke j

Contoh pada tabel 1, jika urutan job adalah 2-3-1 maka hasil perhitungan *Make Span* seperti pada tabel 2.

Tabel 2 Perhitungan *Make Span job* dengan urutan 2-3-1

MESIN	JOB

	2	**3**	**1**
1	5.3	9.9	17.4
2	8.7	13	19.6
3	15.6	21.8	26.4

Dari tabel 2 terlihat bahwa total waktu pengerjaan seluruh *job* dengan urutan 2-3-1 adalah 26.4 satuan waktu. Keterangan : angka 13 pada pengerjaan *job* 3 pada mesin 2 didapat dari MAX(8.7 ; 9.9) + 3.1 di mana 8.7 adalah waktu pengerjaan *job* 2 pada mesin 2. 9.9 adalah waktu pengerjaan *job* 3 sampai pada mesin 1 dan 3.1 adalah alokasi waktu pengerjaan *job* 3 pada mesin 2 (lihat tabel 1).

F. SISTEM INFORMASI PENJADWALAN

Pencarian jadwal produksi paling optimum dengan menggunakan metode *Generate and Test* membutuhkan jumlah pencarian yang tidak sedikit. Hal ini sangat mustahil dikerjakan secara manual dapat dilakukan dengan cepat. Oleh sebab itu sistem informasi penjadwalan yang dapat mendukung keputusan untuk menentukan jadwal urutan *job* paling optimum dilakukan dengan menggunakan bantuan perhitungan komputer.

Langkah-langkah sistem informasi mengeluarkan jadwal produksi *job-job* yang optimum adalah :

1. Menyediakan tabel alokasi mesin untuk setiap *job*.
2. Membangkitkan kemungkinan urutan *job-job*.
3. Menghitung *Make Span* setiap kemungkinan.
4. Menampilkan kemungkinan hasil urutan dan urutan paling optimum.

1. Menyediakan Tabel Alokasi Mesin Untuk Setiap *Job*

Tabel alokasi mesin untuk setiap *job* harus disediakan oleh *user*, berapa jumlah waktu akses mesin untuk masing-masing *job*. Tabel ini dapat dimasukkan langsung ke dalam *grid* atau diimport dari file *Excel*. Gambar 2 menunjukkan *user* dapat mengentri dari *grid* atau mengimport dari file Excel *yang di*-browse dari direktori

E. MAKE SPAN

Make Span adalah total waktu pengerjaan seluruh *job*. *Make Span* dapat diperoleh pada waktu pengerjaan *job* terakhir pada mesin atau aktivitas terakhir. Waktu diselesaikannya *job* ke i sampai pada mesin ke j adalah :

$$T_{i,j} = MAX\ (T_{i-1,j}\ ;\ T_{i,j-1}) + A_{i,j}$$

Di mana

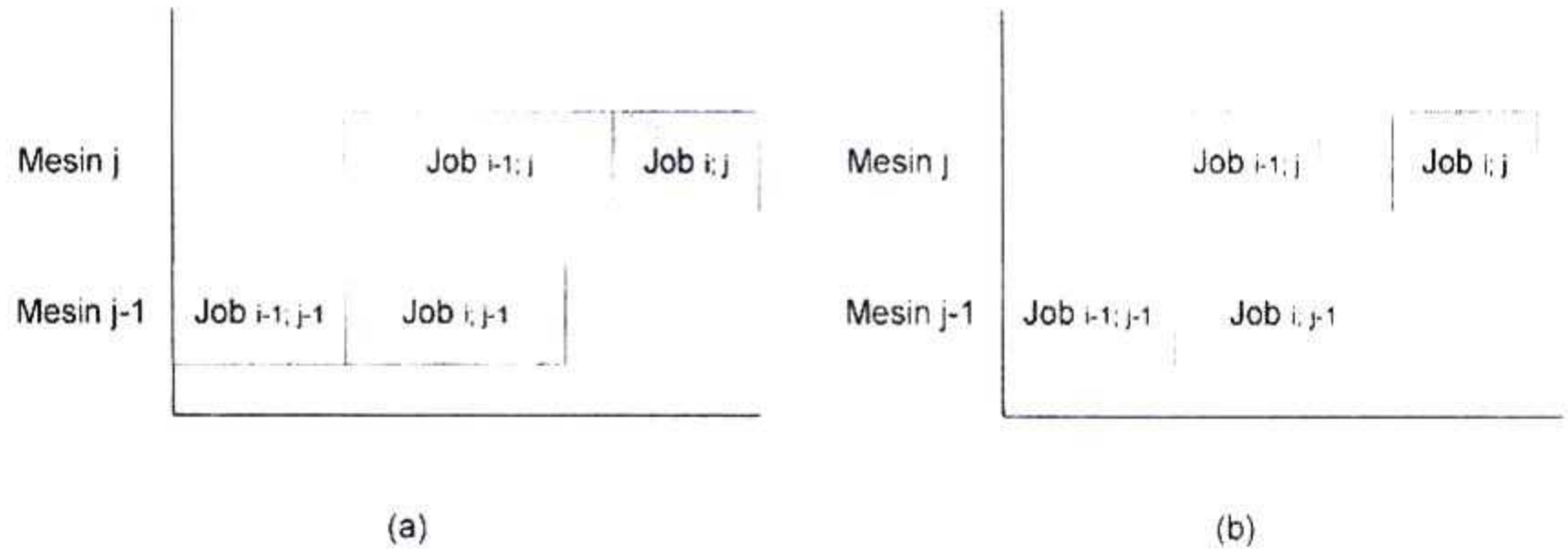
$T_{i,j}$ = waktu diselesaikannya *job* ke i sampai pada mesin ke j

$T_{i-1,j}$ = waktu diselesaikannya *job* ke i-1 sampai pada mesin ke j

$T_{i,j-1}$ = waktu diselesaikannya *job* ke i sampai pada mesin ke j-1

$A_{i,j}$ = alokasi waktu diselesaikannya *job* ke i pada mesin ke j

Parameter MAX ($T_{i-1,j}$; $T_{i,j-1}$) dimasukkan karena pengerjaan *job* ke i pada mesin ke j mulai dilakukan setelah pengerjaan *job* ke i-1 pada mesin j seperti pada gambar 1a atau setelah pengerjaan *job* ke i pada mesin ke j-1 seperti pada gambar 1b.



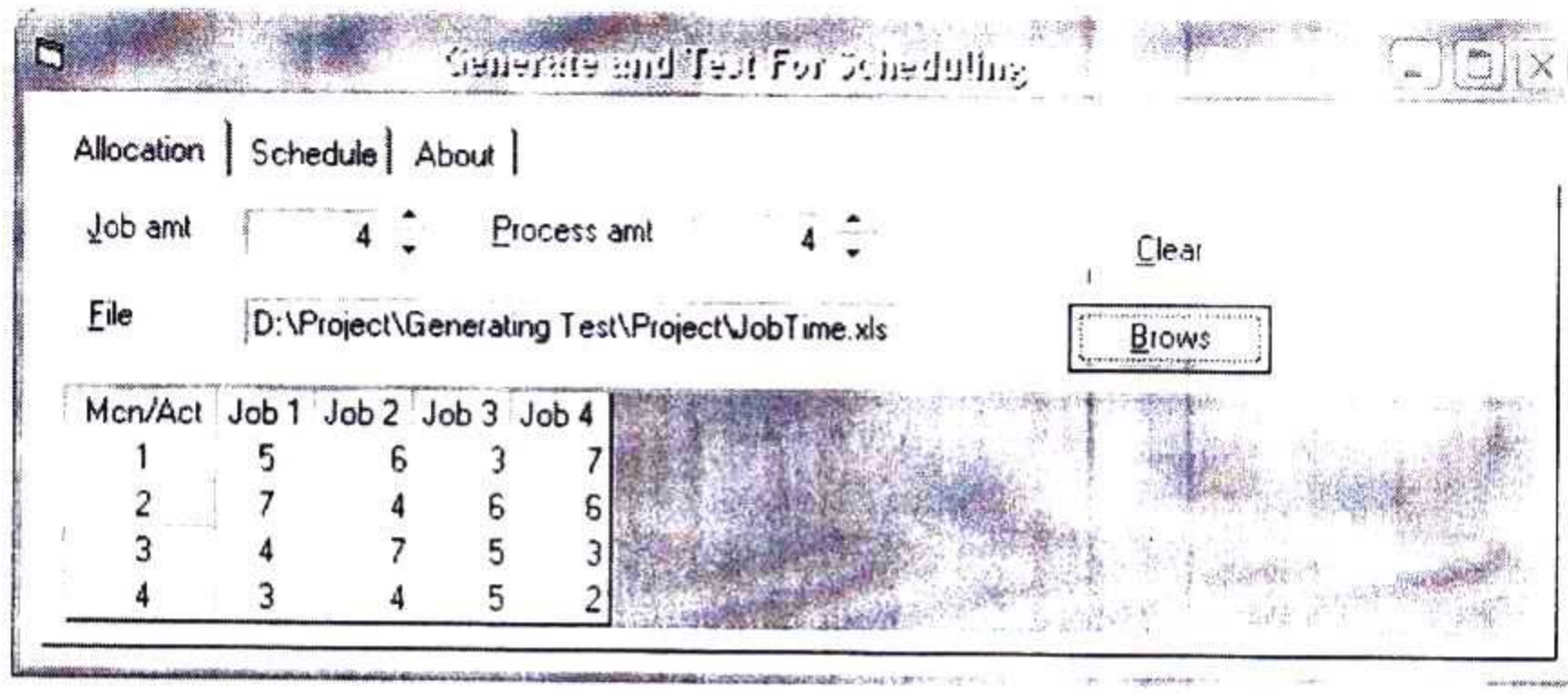
Gambar 1 Gantt Chart pengerjaan *job* ke i pada mesin ke j

Contoh pada tabel 1, jika urutan job adalah 2-3-1 maka hasil perhitungan *Make Span* seperti pada tabel 2.

Tabel 2 Perhitungan *Make Span job* dengan urutan 2-3-1

MESIN	JOB

penyimpanan. Sebelumnya jumlah *job* dan jumlah mesin ditentukan terlebih dahulu. *Setting* ini akan menentukan besar *grid*.



Gambar 2 *Grid* untuk memasukkan alokasi mesin pada *job*

2. Membangkitkan Kemungkinan Urutan *Job-job*

Kombinasi kemungkinan urutan *job* yang muncul dibangkitkan dari jumlah *job* yang telah ditentukan pada *setting* jumlah *job*. Setiap kombinasi urutan akan dihitung *Make* Span dan dibandingkan dengan urutan optimum sebelumnya. Jika lebih optimum, maka ganti urutan optimum dengan kombinasi yang baru dibangkitkan. Algoritma untuk membangkitkan kombinasi urutan *job-job* adalah :

1. Deklarasi ScheduledJob[1..JobAmt] as double sebagai array urutan optimum
2. Deklarasi ScheduledComb[1.. JobAmt] as double sebagai array kombinasi urutan *job*
3. Set ScheduledJob pertama kali dengan urutan awal : 1, 2, 3, …, JobAmt
4. Set OptimumMakeSpan = Maximum Double
5. WHILE MasihCounter DO
6. IF ScheduledJob Valid, merupakan kombinasi
7. IF MakeSpan (ScheduledJob) < OptimumMakeSpan
8. ScheduledJob = ScheduledComb
9. OptimumMakeSpan = MakeSpan (ScheduledJob)

```
10.
11.     Counter ScheduledJob = urutan job berikutnya
12.
13.      IF Counter Selesai
14.              MasihCounter = FALSE
15. END WHILE
```

Setiap proses *counter* akan menghasilkan urutan *job*, tetapi melalui proses validasi, hanya *counter* yang tidak terdapat *job* yang sama yang akan diproses. Alagoritma validasi ScheduledJob adalah :

```
1.  Valid = true
2.  FOR i  = 1 TO NumberJob
3.      numjob = 0
4.      FOR j = 1 TO NumberJob
5.          IF numjob = 1
6.                  EXIT FOR
7.                  Valid = False
8.
9.              numjob = numjob + 1
10.     NEXT j
11. NEXT i
```

3. Menghitung *Make Span* Setiap Kemungkinan

Setiap kemungkinan urutan *job* akan dihitung *Make Span*-nya. Algoritma untuk menghitung *Make Span* adalah :

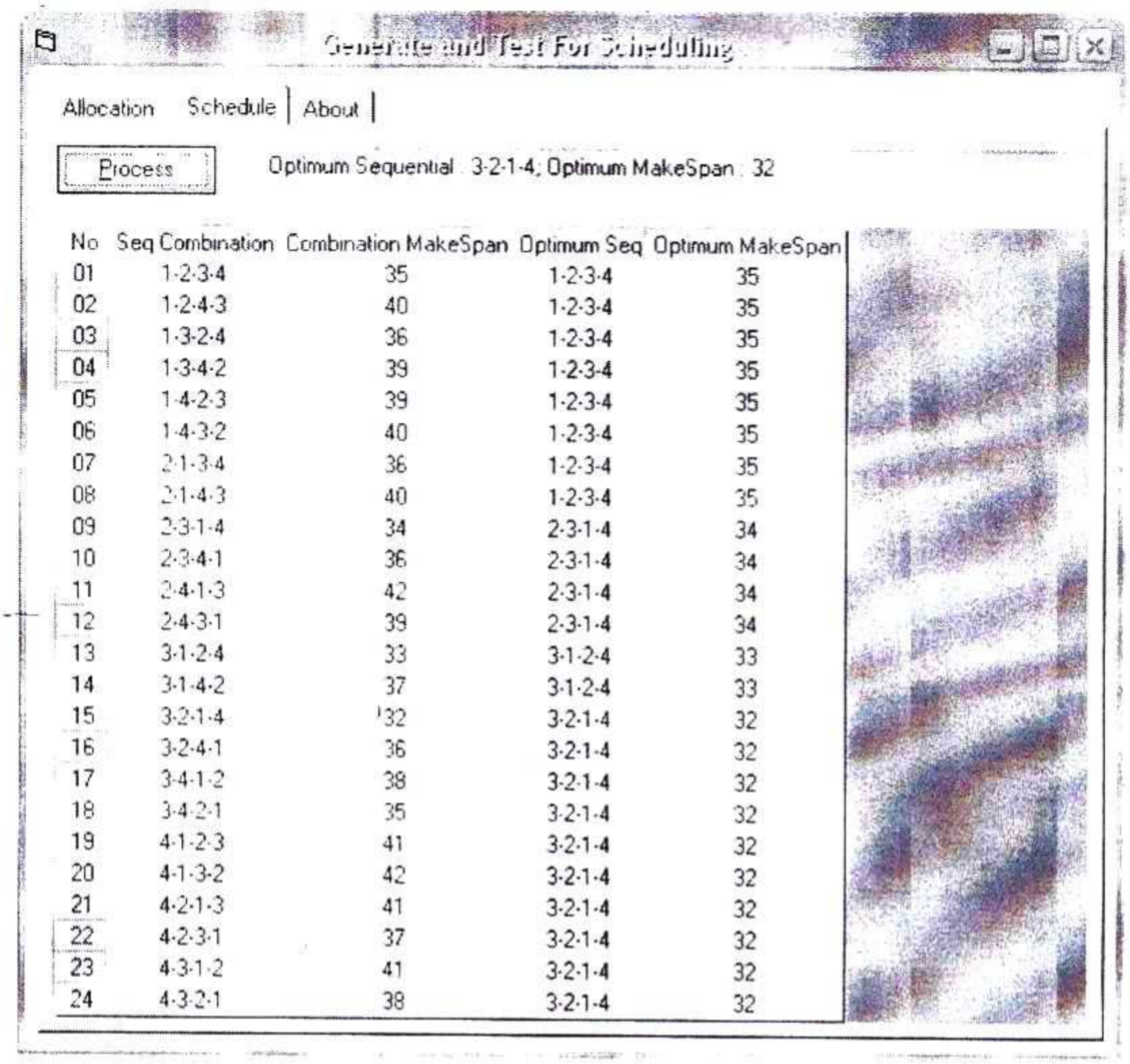
1. Deklarasi MakeSpan[1.. NumberJob] as double sebagai array urutan perhitungan Make Span mesin sebelumnya.
2. Deklarasi SchedComb [1.. NumberJob] as double sebagai array urutan *job* yang akan dicari Make Span-nya.

3. Deklarasi JobTime [1.. NumberMachine, 1..NumberJob] as double sebagai matrix alokasi mesin pada *job*
4. Deklarasi jobtimebefore as double
5. FOR i = 1 TO NumberJob
6. MakeSpan(i) = 0
7.
8. FOR i = 1 TO NumberJob
9. jobtimebefore = 0
10. FOR j = 1 TO NumberJob
11. IF MakeSpan(j) > jobtimebefore
12. jobtimebefore = MakeSpan(j) + JobTime(i, SchedComb(j))
13. ELSE
14. jobtimebefore = jobtimebefore + JobTime(i, SchedComb(j))
15. END IF
16. MakeSpan(j) = jobtimebefore
17. NEXT j
18. NEXT i
19.
20. MakeSpan = jobtimebefore

4. Menampilkan Kemungkinan Hasil Urutan dan Urutan Paling Optimum

Sistem informasi menampilkan keseluruhan proses pencarian *Make Span* optimum dari setiap kombinasi urutan. Setiap ada *Make Span* optimum baru, urutan kombinasi dicatat sebagai urutan optimum. Sampai kombinasi terakhir akan didapatkan urutan *job* optimum dan *Make Span* optimum.

Gambar 3 menunjukkan proses penjadwalan produksi berdasarkan data alokasi mesin pada *job* dengan data seperti pada gambar 2.



No	Seq Combination	Combination MakeSpan	Optimum Seq	Optimum MakeSpan
01	1-2-3-4	35	1-2-3-4	35
02	1-2-4-3	40	1-2-3-4	35
03	1-3-2-4	36	1-2-3-4	35
04	1-3-4-2	39	1-2-3-4	35
05	1-4-2-3	39	1-2-3-4	35
06	1-4-3-2	40	1-2-3-4	35
07	2-1-3-4	36	1-2-3-4	35
08	2-1-4-3	40	1-2-3-4	35
09	2-3-1-4	34	2-3-1-4	34
10	2-3-4-1	36	2-3-1-4	34
11	2-4-1-3	42	2-3-1-4	34
12	2-4-3-1	39	2-3-1-4	34
13	3-1-2-4	33	3-1-2-4	33
14	3-1-4-2	37	3-1-2-4	33
15	3-2-1-4	32	3-2-1-4	32
16	3-2-4-1	36	3-2-1-4	32
17	3-4-1-2	38	3-2-1-4	32
18	3-4-2-1	35	3-2-1-4	32
19	4-1-2-3	41	3-2-1-4	32
20	4-1-3-2	42	3-2-1-4	32
21	4-2-1-3	41	3-2-1-4	32
22	4-2-3-1	37	3-2-1-4	32
23	4-3-1-2	41	3-2-1-4	32
24	4-3-2-1	38	3-2-1-4	32

Gambar 3 *Grid* hasil perhitungan *Make Span* urutan *job*

G. DAFTAR PUSTAKA

Baker, Kenneth R, (1974), *Introduction to Sequencing and Scheduling*, John Wiley and Sons, New York.

Kusumadewi, Sri dan Purnomo, Hari, (2005), *Penyelesaian Masalah Optimasi dengan Teknik-teknik Heuristik*, Graha Ilmu, Yogyakarta.

Pinedo, Michael, (1995), *Scheduling : Theory, Algorithms and Systems*, Prentice Hall, New Jersey.