Evalutioner Computation
Part-3

Pokok Bahasan

1. Travelling Salesman Problem (TSP)
2. Flow-Shop Scheduling Problem (FSP)
3. Two-Stage Assembly Flow-Shop Scheduling Problem
4. Job-Shop Scheduling Problem (JSP)
5. Transportation Problem
6. Flexible Job-Shop Scheduling Problem (FJSP)
7. Multi Travelling Salesman Problem (m-TSP)
8. Tugas

Pengantar

- Masalah kombinatorial adalah masalah yang mempunyai himpunan **solusi *feasible*** yang terhingga. Meskipun secara prinsip solusi masalah ini bisa dengan **enumerasi lengkap**, pada masalah kompleks **dibutuhkan waktu yang tidak bisa diterima secara praktis**.
- Contoh pada masalah *Travelling Salesperson Problem* (*TSP*) yang melibatkan pemilihan rute terbaik untuk mengunjungi *n* kota. Metode enumerasi lengkap harus menguji *n!* kemungkinan solusi. Jika *n*=20 maka ada lebih dari $2{,}4\times10^{18}$ kemungkinan solusi. *PC* mungkin memerlukan **waktu lebih dari 5 jam** untuk melakukan enumerasi lengkap, sebuah hal yang tidak bisa diterima secara praktis.
- **Algoritma genetika** telah **sukses diterapkan pada** berbagai **masalah** kombinatorial seperti **perencanaan** dan **penjadwalan produksi** pada industry manufaktur. Meskipun solusi optimum tidak diperoleh, tetapi solusi yang mendekati optimum bisa didapatkan dalam **waktu yang relatif cepat** dan bisa diterima secara praktis.

Travelling Salesman Problem (TSP)

- Pencarian rute terbaik merupakan salah satu permasalahan yang sering dihadapi dalam kehidupan sehari-hari.
- Salah satu contoh yaitu rute manakah yang **memiliki biaya paling murah (atau paling pendek)** untuk dilalui seorang salesman yang harus mengunjungi sejumlah daerah.
- Tiap daerah harus dikunjungi **tepat satu kali** kemudian kembali lagi ke tempat semula. Permasalahan ini dikenal sebagai Travelling Salesman Problem *(TSP)*. Jika ada lebih dari seorang *salesman* maka disebut *multi Travelling Salesman Problem (m-TSP)*.
- Secara matematis *TSP* bisa diformulasikan sebagai masalah minimasi biaya perjalanan sebagai berikut:

$$Z = \min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\, x_{ij}\right\}$$

  - n menyatakan banyaknya kota (selanjutnya disebut simpul/*node*).
  - C_{ij} menyatakan biaya (atau jarak, tergantung tujuan minimasi) dari simpul *i ke j*.

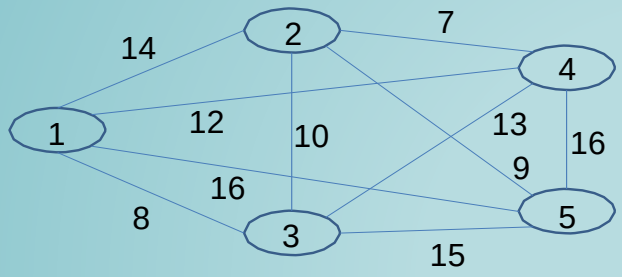
dengan kendala (*contraint*):

$$x_{ij} = \begin{cases} 1, \text{ jika ada perjalanan salesman dari simpul } i \text{ ke } j, \text{ untuk } i, j = 1,2,3,...,n-1 \\ 0, \text{ jika tidak ada perjalanan} \end{cases}$$

Travelling Salesman Problem (TSP)

1. Representasi Chromosome

- Perhatikan masalah TSP berikut :



Tabel jarak antar simpul

Node	1	2	3	4	5
1	-	14	8	12	16
2	14	-	10	7	9
3	8	10	-	13	15
4	12	7	13	-	16
5	16	9	15	16	-

- Representasi **permutasi** bisa digunakan untuk menyatakan sebuah solusi. Setiap gen pada *chromosome* berupa angka integer yang menyatakan nomer dari tiap simpul.
- Contoh *chromosome*, total jarak, dan *fitness*-nya

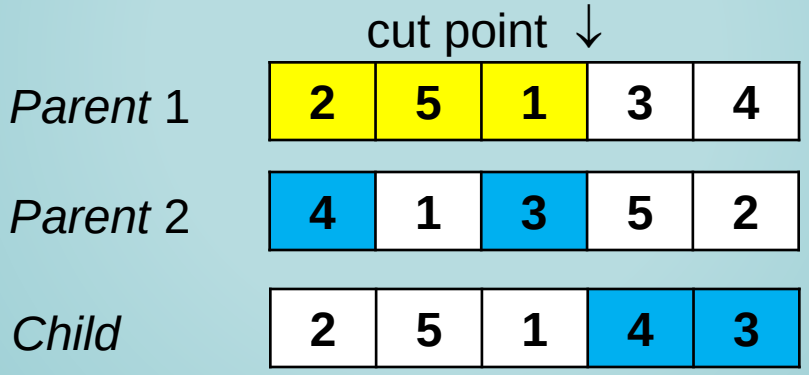
No	Chromosome	Total Jarak (C)	$fitness = \frac{100}{C}$
1	[1 2 3 4 5]	14+10+13+16+16=69	1,449
2	[2 3 4 1 5]	10+13+12+16+9=60	1,667
3	[4 1 2 5 3]	12+14+9+15+13=63	1,587

Sebuah chromosome [2 3 4 1 5] menyatakan perjalanan dimulai dari simpul 2 kemudian secara berurutan mengunjungi simpul 3, 4, 1, 5 dan kemudian kembali ke simpul 2.

Travelling Salesman Problem (TSP)

2. Crossover

- Metode crossover yang paling sederhana adalah dengan melakukan modifikasi ***one-cut-point crossover*** yang digunakan pada representasi biner. Perhatikan contoh pada Gambar 4.2. Segment kiri dari chromosome *child* didapatkan dari *parent 1* dan segmen kanan didapatkan dari urutan gen tersisa dari *parent 2*.



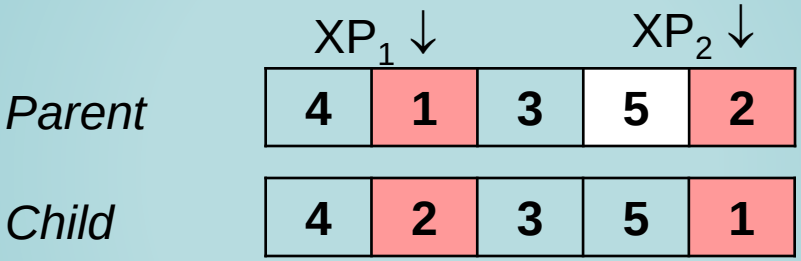
Gambar 4.2 Crossover pada representasi permutasi

- Beberapa metode lain yang digunakan pada representasi permutasi adalah ***partial- mapped crossover* (PMX), *order crossover* (OX), *cycle crossover* (CX), *position-based crossover*, *order-based crossover*, dan *heuristic crossover*.**

Travelling Salesman Problem (TSP)

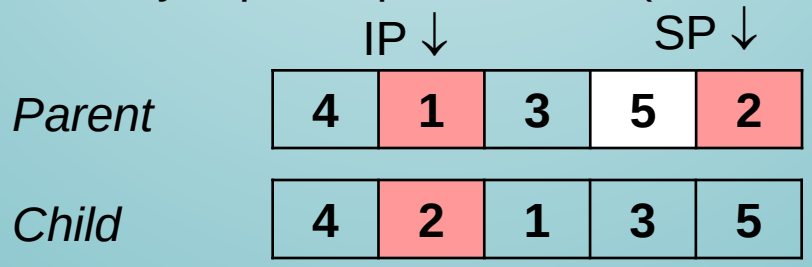
3. Mutasi

- Metode mutasi yang paling sederhana adalah ***reciprocal exchange mutation*.** Bekerja dengan memilih dua posisi (*exchange point* / XP) secara random kemudian menukarkan nilai pada posisi tersebut.



Gambar 4.3 Reciprocal exchange mutation

- Metode lainnya, ***insertion mutation***. Bekerja dengan memilih satu posisi (*selected point* / *SP*) secara random kemudian mengambil dan menyisipkan nilainya pada posisi lain (*insertion point* / *IP*) secara random.



Gambar 4.4 Insertion mutation

Flow-Shop Scheduling Problem (FSP)

- FSP berkaitan dengan penjadwalan sejumlah ***j* job** pada sejumlah ***m* mesin**.
- Semua job mempunyai urutan pemrosesan (operasi) yang sama, mulai dari mesin ke-1, mesin ke-2, dan seterusnya sampai mesin ke-*m*.
- Waktu pemrosesan setiap operasi pada sebuah mesin mungkin berbeda dan diasumsikan telah diketahui sebelumnya.
- Kendala lengkap dari FSP bisa diringkas sebagai berikut:
  - Sebuah job hanya mengunjungi sebuah mesin tepat satu kali.
  - Tidak ada kendala yang diprioritaskan (*precedence*) di antara operasi-operasi dari job yang berbeda.
  - Operasi job pada mesin tidak bisa dinterupsi.
  - Sebuah mesin hanya bisa memproses satu operasi pada satu waktu.

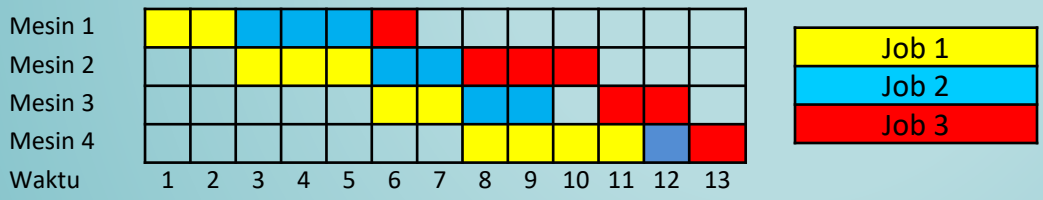
Flow-Shop Scheduling Problem (FSP)

- Perhatikan masalah penjadwalan 3 job (J1, J2, dan J3) pada 4 mesin yang mempunyai waktu pemrosesan sebagai berikut:

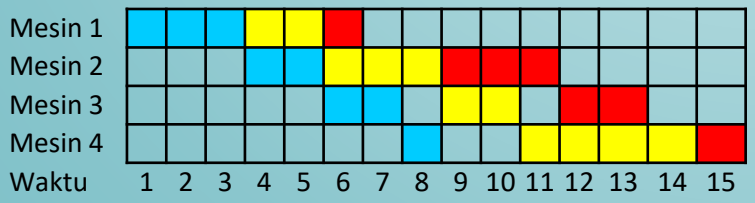
Job	Mesin			
	1	2	3	4
1	2	3	2	4
2	3	2	2	1
3	1	3	2	1

Urutan job yang harus diselesaikan menentukan waktu selesainya seluruh job (***makespan***). Proses optimasi dilakukan untuk menentukan urutan operasi yang menghasilkan nilai *makespan* minimum.

- Jika pemrosesan job dengan urutan J1 → J2 → J3 maka didapatkan *makespan* sebesar 13 pada Gantt-Chart berikut:



- Jika pemrosesan job dengan urutan J2 → J1 → J3 maka didapatkan *makespan* sebesar 15 pada Gantt-Chart berikut:



Representasi permutasi seperti yang ada pada TSP bisa diadopsi untuk masalah FSP. Setiap gen pada *chromosome* menyatakan nomer dari tiap job. Operator *crossover* dan mutasi yang sama pada TSP juga bisa digunakan.

Two-Stage Assembly Flow-Shop Scheduling Problem

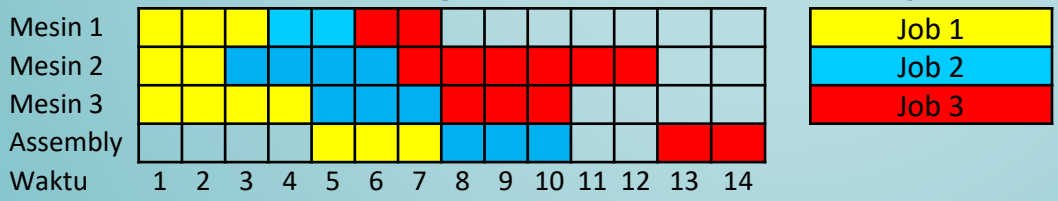
- *Two-stage assembly flowshop* merupakan variasi dari FSP. Pada permasalahan ini sebuah job memiliki *m* operasi yang bisa dikerjakan pada *m* mesin secara paralel.
- Pemrosesan tahap kedua (*assembly stage*) hanya bisa dilakukan setelah semua operasi pada tahap pertama telah diselesaikan.
- Ilustrasi sederhana dari masalah ini adalah pada proses pembuatan sebuah PC yang bisa dianggap sebagai sebuah job. Setelah sejumlah komponen seperti CPU, harddisk, memory dan lain-lain selesai dibuat pada tahap pertama pada tempat/mesin yang berbeda secara paralel maka komponen-komponen ini masuk ke *assembly-station* pada tahap kedua untuk dirakit sesuai spesifikasi yang dibutuhkan konsumen.
- Jika pada saat yang sama terdapat *n* pesanan PC dengan spesifikasi yang berbeda maka bisa dikatakan ada *n* job. Masalah yang timbul adalah bagaimana menentukan urutan pembuatan semua pesanan PC supaya didapatkan waktu penyelesaian semua PC dalam waktu yang paling singkat.

Two-Stage Assembly Flow-Shop Scheduling Problem

- Misalkan terdapat 3 job yang harus diproses pada 3 mesin dengan waktu operasi sebagai berikut

job	mesin			assembly
	1	2	3	
1	3	2	4	3
2	2	4	3	3
3	2	6	3	2

- Jika pemrosesan job ditentukan job 1 $\rightarrow$ job 2 $\rightarrow$ job 3. Pada job 1, operasi tahap kedua (*assembly*) bisa dilakukan pada waktu ke-4 setelah semua operasi tahap pertama diselesaikan. Berikut Gantt-chart dari urutan operasi tersebut dengan nilai *makespan* = 14.



Gantt-chart untuk two-stage assembly flowshop

- Seperti halnya pada FSP, representasi permutasi seperti yang digunakan pada TSP bisa diadopsi untuk masalah *two-stage assembly flowshop*. Operator crossover dan mutasi yang sama pada TSP juga bisa digunakan.

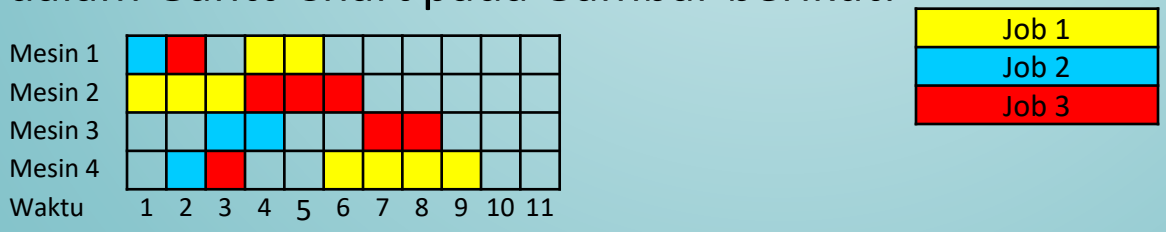
Job-Shop Scheduling Problem (JSP)

- JSP merupakan perluasan (bentuk umum) dari FSP. Pada masalah ini tiap job mungkin mempunyai **urutan operasi yang berbeda**. Pada beberapa kasus, **sebagian job** hanya **memerlukan sebagian mesin**. Perhatikan masalah JSP berikut:

Job	Waktu Operasi Pada Mesin				Urutan Operasi
	1	2	3	4	
1	2	3	-	4	$2 \rightarrow 1 \rightarrow 4$
2	1	-	2	1	$1 \rightarrow 4 \rightarrow 3$
3	1	3	2	1	$1 \rightarrow 4 \rightarrow 2 \rightarrow 3$

Pada kasus ini, job 1 tidak memerlukan mesin 3 dan urutan operasinya dimulai dari mesin 2, kemudian ke mesin 1 dan 4.

- Misalkan $O_{i,j}$ menyatakan **operasi ke-*j* dari job *i***. Jika pemrosesan job dilakukan dengan urutan $O_{1,1} \rightarrow O_{2,1} \rightarrow O_{1,2} \rightarrow O_{3,1} \rightarrow O_{3,2} \rightarrow O_{1,3} \rightarrow O_{2,2} \rightarrow O_{3,3} \rightarrow O_{3,4} \rightarrow O_{2,3}$ maka didapatkan makespan sebesar 9 yang ditunjukan dalam Gantt-Chart pada Gambar berikut:



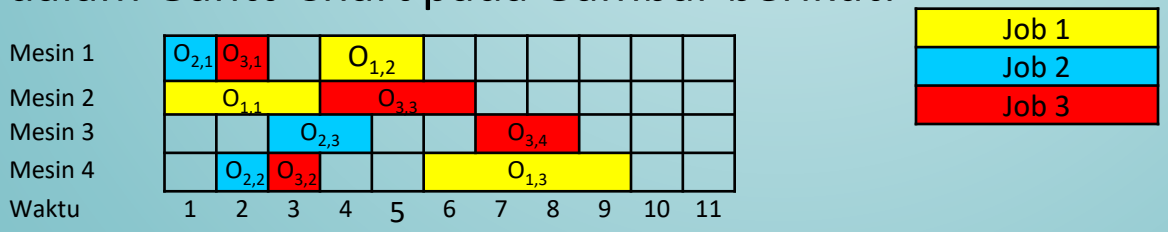
Job-Shop Scheduling Problem (JSP)

- JSP merupakan perluasan (bentuk umum) dari FSP. Pada masalah ini tiap job mungkin mempunyai **urutan operasi yang berbeda**. Pada beberapa kasus, **sebagian job** hanya **memerlukan sebagian mesin**. Perhatikan masalah JSP berikut:

Job	Waktu Operasi Pada Mesin				Urutan Operasi
	1	2	3	4	
1	2	3	-	4	$2 \rightarrow 1 \rightarrow 4$
2	1	-	2	1	$1 \rightarrow 4 \rightarrow 3$
3	1	3	2	1	$1 \rightarrow 4 \rightarrow 2 \rightarrow 3$

Pada kasus ini, job 1 tidak memerlukan mesin 3 dan urutan operasinya dimulai dari mesin 2, kemudian ke mesin 1 dan 4.

- Misalkan $O_{i,j}$ menyatakan **operasi ke-*j* dari job *i***. Jika pemrosesan job dilakukan dengan urutan $O_{1,1} \rightarrow O_{2,1} \rightarrow O_{1,2} \rightarrow O_{3,1} \rightarrow O_{3,2} \rightarrow O_{1,3} \rightarrow O_{2,2} \rightarrow O_{3,3} \rightarrow O_{3,4} \rightarrow O_{2,3}$ maka didapatkan makespan sebesar 9 yang ditunjukan dalam Gantt-Chart pada Gambar berikut:



Job-Shop Scheduling Problem (JSP)

1. Representasi Chromosome

Job	Waktu Operasi Pada Mesin				Urutan Operasi
	1	2	3	4	
1	2	3	-	4	$2 \rightarrow 1 \rightarrow 4$
2	1	-	2	1	$1 \rightarrow 4 \rightarrow 3$
3	1	3	2	1	$1 \rightarrow 4 \rightarrow 2 \rightarrow 3$

- Representasi integer yang memuat nomer job (*job-based representation*) bisa digunakan untuk masalah JSP. Misalkan untuk solusi di atas maka bisa dinyatakan sebagai:

  Job : [1 2 1 3 3 1 2 3 3 2]
  Op : [1 1 2 1 2 3 2 3 4 3]

  dan pemrosesan job dilakukan dengan urutan $O_{job,op}$:

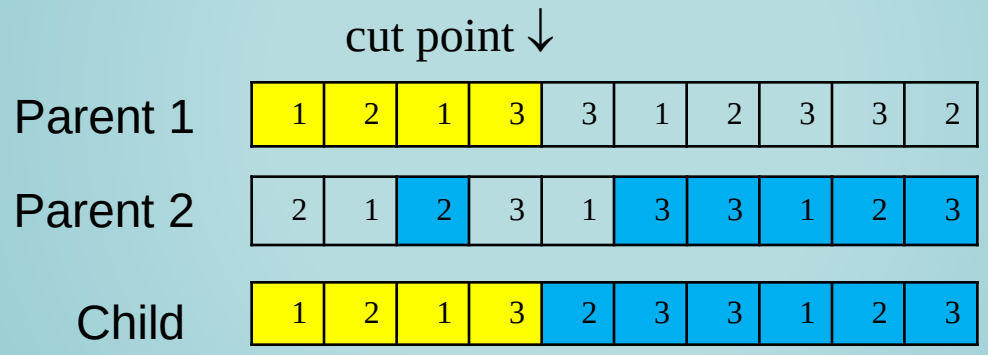
  $O_{1,1} \rightarrow O_{2,1} \rightarrow O_{1,2} \rightarrow O_{3,1} \rightarrow O_{3,2} \rightarrow O_{1,3} \rightarrow O_{2,2} \rightarrow O_{3,3} \rightarrow O_{3,4} \rightarrow O_{2,3}$

- Perhatikan pada representasi ini angka 1 muncul sebanyak 3 kali yang menyatakan bahwa job 1 mempunyai 3 operasi. Hal serupa terjadi pada angka 3 yang muncul sebanyak 4 kali yang menyatakan bahwa job 3 mempunyai 4 operasi.

Job-Shop Scheduling Problem (JSP)

2. Crossover

- Modifikasi *one-cut-point crossover* bisa diterapkan pada *job-based representation*. Perhatikan contoh pada Gambar berikut. Segment kiri dari chromosome *child* didapatkan dari *parent 1* dan segmen kanan didapatkan dari urutan gen tersisa dari *parent 2*.



3. Mutasi

- Metode *reciprocal exchange mutation* dan *insertion mutation* yang digunakan pada representasi permutasi bisa dengan mudah diterapkan untuk *job-based representation*.

Job-Shop Scheduling Problem (JSP)

4. Representasi Permutasi Untuk JSP

- Dengan penanganan khusus, representasi permutasi bisa diterapkan untuk JSP. Untuk permasalahan JSP yang telah diuraikan sebelumnya, diperlukan peta sebagai berikut:

Angka permutasi	1	2	3	4	5	6	7	8	9	10
Job	1	1	1	2	2	2	3	3	3	3

Gambar 4.10 Representasi permutasi untuk JSP

- Peta pada Gambar 4.10 menunjukkan aturan konversi dari angka permutasi yang muncul pada chromosome ke indeks dari tiap job. Misalkan didapatkan chromosome dengan representasi permutasi sebagai berikut:

[7 1 8 4 10 2 6 9 3 5]

- Maka bisa dikonversi menjadi *job-based representation* sebagai berikut:

[3 1 3 2 3 1 2 3 1 2]

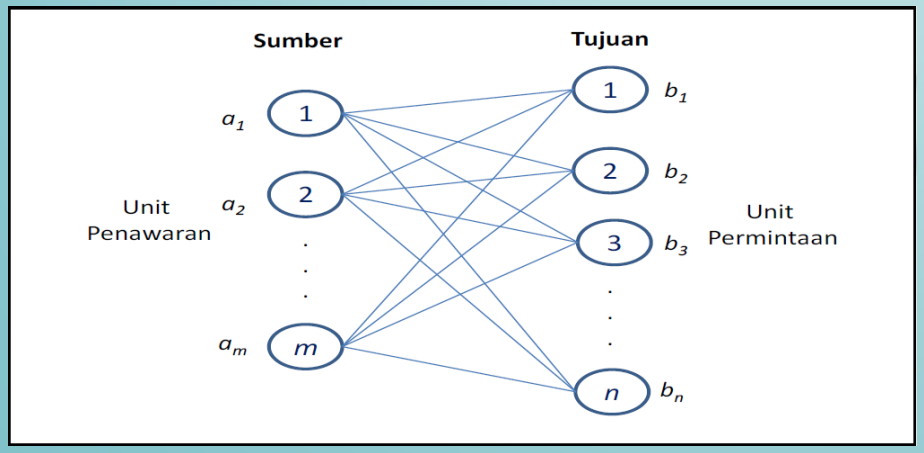
- Dengan representasi permutasi ini, operator crossover dan mutasi yang sama pada TSP juga bisa digunakan.

Transportation Problem

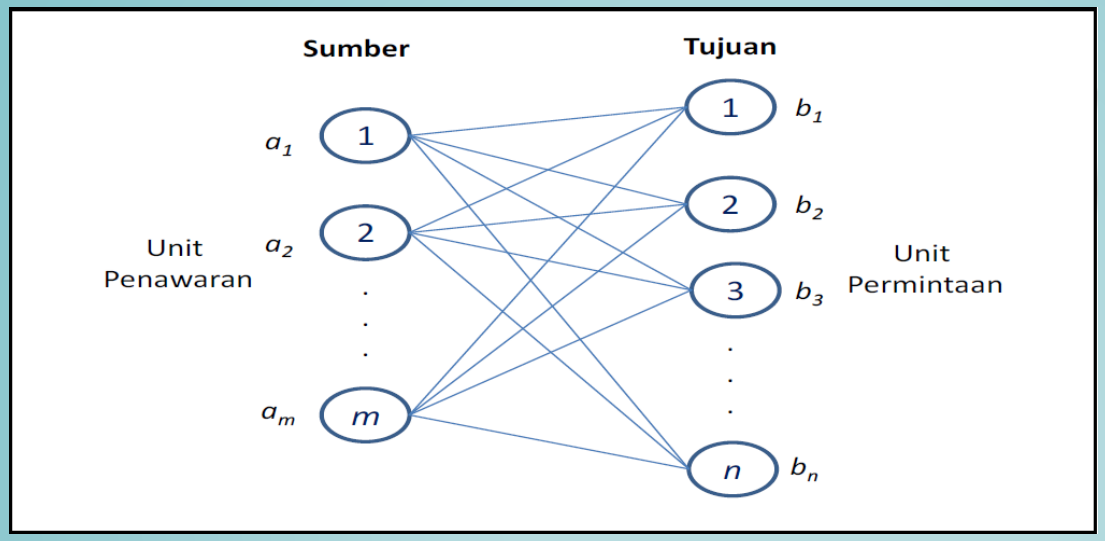
- Persoalan transportasi berkaitan dengan pendistribusian suatu komoditas atau produk dari sejumlah sumber (*supply*) kepada sejumlah tujuan (*destination*, *demand*) dengan tujuan meminimumkan ongkos pengangkutan yang terjadi.
- Persoalan ini mempunyai beberapa karakteristik yang bisa diringkas sebagai berikut (Taha 2011):
  1. Terdapat sejumlah sumber dan sejumlah tujuan tertentu.
  2. Kuantitas komoditas yang didistribusikan dari setiap sumber dan yang diminta oleh setiap tujuan besarnya tertentu.
  3. Komoditas yang dikirim atau diangkut dari suatu sumber ke suatu tujuan besarnya sesuai dengan permintaan dan atau kapasitas sumber.
  4. Ongkos pengangkutan komoditas dari suatu sumber ke suatu tujuan besarnya tertentu.

Transportation Problem

- Persoalan transportasi merupakan kasus khusus dari model pemrograman linier (*linear programming*). Persoalan ini membutuhkan pembatas (*constrains*) dan variabel yang sangat banyak sehingga penggunaan komputer dalam penyelesaiannya dengan model matematis (misalnya dengan metode simplex) memerlukan perhitungan yang panjang dan tidak praktis.
- GAs terbukti efektif untuk mendapatkan solusi yang mendekati optimum dalam waktu yang relatif cepat. Model transportasi sederhana dari sebuah jaringan dengan *m* sumber dan *n* tujuan digambarkan sebagai berikut:



Transportation Problem



- Gambar diatas menyatakan bahwa sumber i (i = 1, 2, …, m) mempunyai persediaan a_i unit untuk didistribusikan ke tujuan-tujuan, dan tujuan j (j = 1, 2, …, n) mempunyai permintaan b_j unit untuk diterima dari sumber-sumber.
- Variabel c_{ij} (i = 1, 2, …, m; j = 1, 2, …, n) adalah biaya pendistribusian yang dibutuhkan dari sumber i ke tujuan j. x_{ij} (i = 1, 2, …, m; j = 1, 2, …, n) adalah variabel keputusan yang menyatakan banyaknya produk yang harus dikirimkan dari sumber i ke tujuan j.

Transportation Problem

- Dari gambaran yang telah diberikan, permasalahan transportasi dapat dinyatakan secara matematis sebagai berikut:

$$Z = \min\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}\right\}$$

dengan kendala (*contraint*):

$$\sum_{j=1}^{n} x_{ij} = a_i \quad \text{untuk } i = 1,2,3,....,m \qquad (4.4)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad \text{untuk } j = 1,2,3,....,n \qquad (4.5)$$

$$x_{ij} \quad \text{untuk semua } i \text{ dan } j$$

- Pers. (4.4) menyatakan jumlah barang yang dikirim dari setiap sumber sama dengan ketersediaan barang pada sumber tersebut.
- Pers. (4.5) menyatakan jumlah barang yang dikirim ke setiap tujuan sama dengan permintaan barang pada tujuan tersebut.
- Kendala-kendala di atas menyatakan total penawaran sama dengan total permintaan yang biasa disebut model transportasi seimbang.

Transportation Problem

1. Representasi Chromosome

- Misalkan diberikan permasalahan transportasi dengan 3 sumber dan 4 tujuan. Persediaan di sumber adalah 10, 15 dan 5. Permintaan di tujuan adalah 10, 5, 5 dan 10. Matriks biaya adalah sebagai berikut:

$$c = \begin{bmatrix} 4 & 5 & 1 & 2 \\ 3 & 2 & 4 & 5 \\ 5 & 4 & 3 & 1 \end{bmatrix}$$

- Karena solusi masalah transportasi adalah matriks yang menyatakan banyaknya produk yang harus dikirim dari sumber *i* ke tujuan *j* maka chromosome dapat dinyatakan sebagai matriks dengan contoh berikut:

					a_i
	0	0	5	5	**10**
$P =$	10	5	0	0	**15**
	0	0	0	5	**5**
b_j	**10**	**5**	**5**	**10**	

- Perhatikan total elemen setiap baris sama dengan banyaknya persediaan (a_i) di tiap sumber dan total elemen setiap kolom sama dengan banyaknya pemintaan (b_j) di tiap tujuan. Total biaya dari solusi di atas adalah 60.

Transportation Problem

2. Crossover

- Crossover dilakukan dengan melakukan perhitungan rata-rata tiap elemen dari dua chromosome (P_1 dan P_2) untuk menghasilkan anak (C) seperti contoh berikut:

$$P_1 = \begin{bmatrix} 0 & 0 & 5 & 5 \\ 10 & 5 & 0 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \qquad P_2 = \begin{bmatrix} 5 & 0 & 0 & 5 \\ 0 & 5 & 5 & 5 \\ 5 & 0 & 0 & 0 \end{bmatrix}$$

					a_i
C =	3	0	3	5	11
	5	5	3	3	16
	3	0	0	3	6
b_j	11	5	6	11	

- Karena ada proses pembulatan ke atas maka dihasilkan chromosome yang ***infeasible***. Perhatikan total nilai tiap baris dan kolom yang tidak sama dengan persediaan dan permintaan. Mekanisme perbaikan (*repairing*) diperlukan untuk menghasilkan chromosome yang ***feasible*** sebagai berikut:

					a_i
C' =	2	0	3	5	10
	5	5	2	3	15
	3	0	0	2	5
b_j	10	5	5	10	

Perbaikan chromosome di samping dilakukan dengan melakukan perubahan pada tiap sel yang total nilai baris dan kolomnya tidak sesuai.

Transportation Problem

3. Mutasi

- Metode mutasi sederhana bekerja dengan memilih 4 titik secara acak sehingga membentuk loop tertutup. Alokasi barang pada tiap titik sudut loop diubah sedemikian rupa sehingga total tiap baris dan total tiap kolom tidak berubah.

$P =$

				a_i
0	0	5	5	**10**
10	5	0	0	**15**
0	0	0	5	**5**
10	**5**	**5**	**10**	

(b_j)

$C =$

				a_i
0	0	5	5	**10**
5	5	0	5	**15**
5	0	0	0	**5**
10	**5**	**5**	**10**	

(b_j)

Transportation Problem

4. Representasi Permutasi

- Representasi permutasi bisa diadopsi untuk permasalahan transportasi. Misalkan untuk contoh kasus pada bagian ke-1 (**Slide 20**), setiap sel pada matriks alokasi diberi nomer urut seperti contoh berikut:

1	2	3	4
5	6	7	8
9	10	11	12

- Setiap gen pada chromosome menyatakan prioritas alokasi. Untuk contoh chromosome:

[5 8 3 1 9 12 10 11 6 2 7 4]

- maka bisa diuraikan menjadi sebuah solusi dengan langkah-langkah berikut:

1. Alokasikan unit maksimum pada sel dengan nomor urut 5. Pada sel ini unit maksimum yang bisa dialokasikan sebesar 10. ***a'i*** dan ***b'j*** menunjukkan total baris dan kolom sementara.

Transportation Problem

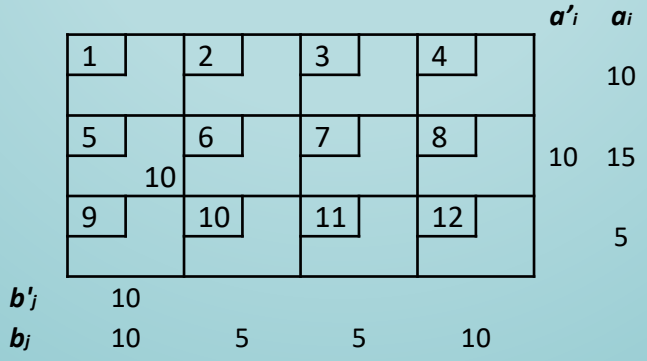
4. Representasi Permutasi

- Setiap gen pada chromosome menyatakan prioritas alokasi. Untuk contoh chromosome:

[5 8 3 1 9 12 10 11 6 2 7 4]

- Maka bisa diuraikan menjadi sebuah solusi dengan langkah-langkah berikut:

1. Alokasikan unit maksimum pada sel dengan nomor urut 5. Pada sel ini unit maksimum yang bisa dialokasikan sebesar 10. a'_i dan b'_j menunjukkan total baris dan kolom sementara.



Transportation Problem

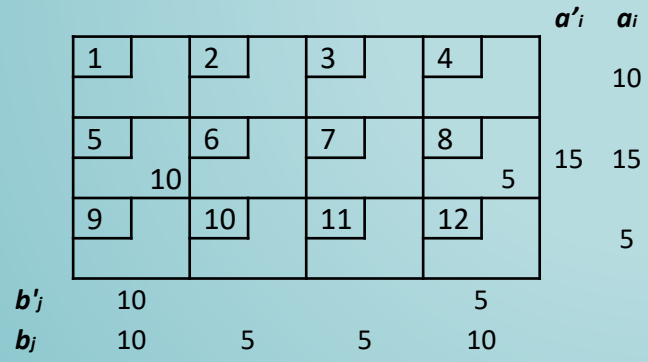
4. Representasi Permutasi

- Setiap gen pada chromosome menyatakan prioritas alokasi. Untuk contoh chromosome:

[5 8 3 1 9 12 10 11 6 2 7 4]

- Maka bisa diuraikan menjadi sebuah solusi dengan langkah-langkah berikut:

2. Isi sel 8 dengan unit maksimum sebesar 5.



Transportation Problem

4. Representasi Permutasi

- Setiap gen pada chromosome menyatakan prioritas alokasi. Untuk contoh chromosome:

[5 8 3 1 9 12 10 11 6 2 7 4]

- Maka bisa diuraikan menjadi sebuah solusi dengan langkah-langkah berikut:

3. Isi sel 3 dengan unit maksimum sebesar 5.

					a'_i	a_i
	1	2	3	4	5	10
			5			
	5	6	7	8	15	15
	10			5		
	9	10	11	12		5
b'_j	10		5	5		
b_j	10	5	5	10		

4. Sesuai urutan gen pada chromosome, isi sel 1. Tetapi sel ini tidak bisa disi karena kolom 1 sudah penuh ($b'_1 = b_1$). Hal yang sama juga terjadi pada sel 9 karena kolom yang bersesuaian sudah penuh.

Transportation Problem

4. Representasi Permutasi

- Setiap gen pada chromosome menyatakan prioritas alokasi. Untuk contoh chromosome:

  [5 8 3 1 9 12 10 11 6 2 7 4]

- Maka bisa diuraikan menjadi sebuah solusi dengan langkah-langkah berikut:

5. Isi sel 12 dengan unit maksimum sebesar 5.

					a'_i	a_i
	1	2	3	4		
			5		5	10
	5	6	7	8		
	10			5	15	15
	9	10	11	12		
				5	5	5
b'_j	10		5	10		
b_j	10	5	5	10		

6. Sel 10 dan 11 tidak bisa disi karena baris 3 sudah penuh ($a'_3 = a_3$). Hal yang sama juga terjadi pada sel 6 karena baris 2 sudah penuh.

Transportation Problem

4. Representasi Permutasi

- Setiap gen pada chromosome menyatakan prioritas alokasi. Untuk contoh chromosome:

[5 8 3 1 9 12 10 11 6 2 7 4]

- Maka bisa diuraikan menjadi sebuah solusi dengan langkah-langkah berikut:

7. Isi sel 2 dengan unit maksimum sebesar 5. Setelah sel ini disi maka alokasi sudah komplit ($a'_i = a_i$ dan $b'_j = b_j$) sehingga pengecekan gen setelahnya tidak perlu dilakukan.

					a'_i	a_i
	1	2	3	4		
		5	5		10	10
	5	6	7	8		
	10			5	15	15
	9	10	11	12		
				5	5	5
b'_j	10	5	5	10		
b_j	10	5	5	10		

Flexible Job-Shop Scheduling Problem (FJSP)

- Flexible job-shop problem (FJSP) merupakan bentuk umum (*generalized form*) dari JSP klasik.
- Pada permasalahan FSP, sebuah job memiliki beberapa operasi. Sebuah operasi bisa dikerjakan pada beberapa pilihan mesin. Skenario ini lebih dekat dengan kasus nyata yang ditemui pada industri manufaktur (Zhang et al. 2009). Keberadaan mesin alternatif ini membuat FJSP lebih sulit diselesaikan dibandingkan JSP.
- Ada dua keputusan yang harus dibuat pada FJPS :
  - Keputusan yang pertama adalah penentuan mesin untuk tiap operasi (*routing problem*).
  - Keputusan yang kedua adalah menentukan urutan operasi-operasi tersebut (*scheduling problem*). Seperti halnya dengan JSP, tujuan utama dari FJSP adalah meminimumkan waktu selesainya seluruh job (*makespan*).

Flexible Job-Shop Scheduling Problem (FJSP)

- Karena ada dua keputusan yang harus dibuat, maka pendekatan untuk menyelesaikan FJPS bisa diklasifikasikan dalam dua kategori dasar, yaitu pendekatan hirarki (*hierarchical approaches*) dan pendekatan terintegrasi (*integrated approaches*).
- Pendekatan hirarki digunakan untuk mengurangi kompleksitas permasalahan dengan menguraikan FJSP ke dalam dua sub permasalahan.
  - Sub permasalahan yang pertama adalah penentuan mesin menggunakan metode heuristik atau *dispatching rules*.
  - Sub permasalahan yang kedua adalah menentukan urutan operasi-operasi tersebut yang menghasilkan nilai *makespan* minimum. Pendekatan terintegrasi menyelesaikan kedua sub permasalahan tersebut secara bersamaan/simultan (Al-Hinai & ElMekkawy 2011; Pezzella, Morganti & Ciaschetti 2008; Yazdani, Amiri & Zandieh 2010). Meskipun pendekatan terintegrasi lebih sulit dan membutuhkan waktu komputasi yang lebih tinggi, sejumlah penelitian membuktikan bahwa pendekatan ini mampu memberikan hasil yang lebih baik.

Flexible Job-Shop Scheduling Problem (FJSP)

1. Representasi Chromosome *Task Sequencing List*

- Contoh satu kasus FJSP disajikan pada Tabel berikut:

job	operasi	machine	time
1	1	1	5
		2	6
	2	3	4
2	1	2	7
		3	8
	2	1	6
		3	4
	3	2	3
3	1	1	4
	2	2	4
4	1	1	5
	2	3	4
	3	1	6

Pada **Tabel** terdapat 4 job. Operasi 1 dari job 1 bisa dikerjakan di dua pilihan mesin. Jika dikerjakan di mesin 1, butuh waktu 5 unit. Jika dikerjakan di mesin 2, butuh waktu 6 unit.

Satu representasi yang dinamakan *task sequencing list representation* diusulkan oleh Kacem et al. (2002). Representasi ini memuat tiga bilangan bulat, yaitu: job, operasi dan mesin, seperti pada **Gambar 4.12** yang menunjukan prioritas penjadwalan yang pertama adalah memproses operasi 1 dari job 2 pada mesin 2, diikuti dengan memproses operasi 1 dari job 4 pada mesin 1, dan seterusnya.
Representasi ini mensyaratkan proses reproduksi crossover dan mutasi harus dilakukan secara hati-hati untuk menghasilkan *offspring* yang feasible.

job	2	4	1	2	3	1	3	4	2	4
operasi	1	1	1	2	1	2	2	2	3	3
mesin	2	1	2	1	1	3	2	3	2	1
urutan	1	2	3	4	5	6	7	8	9	10

Gambar 4.12. Contoh chromosome dari *task sequencing list representation*

Flexible Job-Shop Scheduling Problem (FJSP)

2. Representasi Chromosome Bilangan Pecahan

- Mahmudy, Marian and Luong (2013f) mengajukan penggunaan vector bilangan real sebagai representasi chromosome untuk FJSP. Keunggulan representasi ini adalah berbagai jenis metode crossover dan mutasi bisa diterapkan.
- Representasi ini juga selalu menghasilkan solusi feasible. Properti ini sangat berguna untuk menghemat waktu komputasi yang dibutuhkan untuk proses perbaikan (*repairing*) chromosome infeasible.
- Untuk proses decoding chromosome untuk kasus pada **Slide 31**, dibutuhan tabel yang memetakan indeks urutan kemunculan gen dengan nomer job sebagai berikut:

index	1	2	3	4	5	6	7	8	9	10
job	1	1	2	2	2	3	3	4	4	4

Flexible Job-Shop Scheduling Problem (FJSP)

2. Representasi Chromosome Bilangan Pecahan

- Untuk proses decoding chromosome untuk kasus pada **Slide 31**, dibutuhan tabel yang memetakan indeks urutan kemunculan gen dengan nomer job sebagai berikut:

Tabel 1 Memetakan indeks dengan urutan kemunculan gen

index	1	2	3	4	5	6	7	8	9	10
job	1	1	2	2	2	3	3	4	4	4

- Berdasarkan tabel di atas maka sebuah chromosome P=[143, 209, 115,173, 75, 179, 193, 96, 83, 144] bisa diuraikan menjadi sebuah solusi seperti ditunjukkan pada tabel berikut:

x	operasi	sorted *x*	operasi'	job,op,mac
143	1	75	5	2, 1, 3
209	2	83	9	4, 1, 1
115	3	96	8	4, 2, 3
173	4	115	3	2, 2, 3
75	5	143	1	1, 1, 2
179	6	144	10	4, 3, 1
193	7	173	4	2, 3, 2
96	8	179	6	3, 1, 1
83	9	193	7	3, 2, 2
144	10	209	2	1, 2, 3

Cara konversi chromosome menjadi solusi :
Mengurutkan ***x*** (bersama-sama ***operasi***) secara *ascending* sehingga dihasilkan ***sorted x*** dan ***operasi'***. Dari ***operasi'*** bisa didapatkan nomer ***job*** pada kolom terakhir dengan menggunakan **Tabel 1**. Misal operasi' 5 menunjukkan job 2, operasi' 9 menunjukkan job 4, dan seterusnya. Operasi (***op***) bisa didapatkan dengan memberi nomor urut kemunculan ***job*** yang sama.

Flexible Job-Shop Scheduling Problem (FJSP)

2. Representasi Chromosome Bilangan Pecahan

- Berdasarkan tabel di atas maka sebuah chromosome P=[143, 209, 115,173, 75, 179, 193, 96, 83, 144] bisa diuraikan menjadi sebuah solusi seperti ditunjukkan pada tabel berikut:

x	operasi	sorted *x*	operasi'	job,op,mac
143	1	75	5	2, 1, 3
209	2	83	9	4, 1, 1
115	3	96	8	4, 2, 3
173	4	115	3	2, 2, 3
75	5	143	1	1, 1, 2
179	6	144	10	4, 3, 1
193	7	173	4	2, 3, 2
96	8	179	6	3, 1, 1
83	9	193	7	3, 2, 2
144	10	209	2	1, 2, 3

Untuk menentukan mesin yang dipakai (***mac***) maka ***x*** dikonversi ke bilangan biner. Misalkan x_1 =75 dikonversi ke $(1001011)_2$. 2 bit paling kanan $(11)_2$=3 dipakai untuk menentukan indeks mesin. Karena ada 2 alternatif mesin (*n*=2) yang bisa dipakai untuk operasi pertama dari job 2, maka digunakan rumus sebagai berikut:

indeks mesin = 3 MOD *n* + 1 = 3 MOD 2 +1 = 2

MOD merupakan operator untuk menghitung sisa pembagian.

Dari hasil ini maka operasi pertama dari job 2 dilakukan pada alternatif mesin yang ke-2, yaitu mesin 3. Pada proses kenversi ini digunakan 2 bit paling kanan karena maksimum banyaknya alternatif mesin untuk setiap operasi adalah sebesar 2 yang hanya memerlukan 2 bit pada representasi biner. Untuk kasus lain dengan banyak alternatif mesin sebanyak 10 maka diperlukan 4 bit.

Flexible Job-Shop Scheduling Problem (FJSP)

2. Representasi Chromosome Bilangan Pecahan

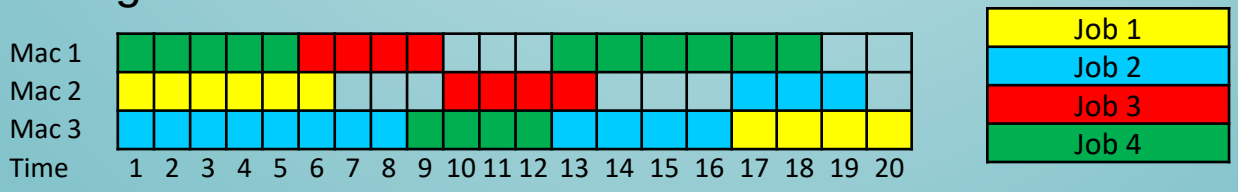
- Berdasarkan tabel di atas maka sebuah chromosome P=[143, 209, 115,173, 75, 179, 193, 96, 83, 144] bisa diuraikan menjadi sebuah solusi seperti ditunjukkan pada tabel berikut:

job	operasi	machine	time
1	1	1	5
		2	6
	2	3	4
2	1	2	7
		3	8
	2	1	6
		3	4
	3	2	3
3	1	1	4
	2	2	4
4	1	1	5
	2	3	4
	3	1	6

index	1	2	3	4	5	6	7	8	9	10
job	1	1	2	2	2	3	3	4	4	4

x	operasi	sorted *x*	operasi'	job,op,mac
143	1	75	5	2, 1, 3
209	2	83	9	4, 1, 1
115	3	96	8	4, 2, 3
173	4	115	3	2, 2, 3
75	5	143	1	1, 1, 2
179	6	144	10	4, 3, 1
193	7	173	4	2, 3, 2
96	8	179	6	3, 1, 1
83	9	193	7	3, 2, 2
144	10	209	2	1, 2, 3

- Dari solusi ini bisa dihasilkan Gantt-chart untuk menghitung *makespan* sebagai berikut:



Flexible Job-Shop Scheduling Problem (FJSP)

2. Representasi Chromosome Bilangan Pecahan

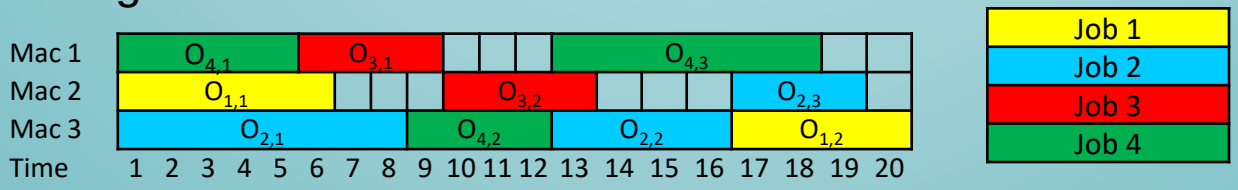
- Berdasarkan tabel di atas maka sebuah chromosome P=[143, 209, 115,173, 75, 179, 193, 96, 83, 144] bisa diuraikan menjadi sebuah solusi seperti ditunjukkan pada tabel berikut:

job	operasi	machine	time
1	1	1	5
		2	6
	2	3	4
2	1	2	7
		3	8
	2	1	6
		3	4
	3	2	3
3	1	1	4
	2	2	4
4	1	1	5
	2	3	4
	3	1	6

index	1	2	3	4	5	6	7	8	9	10
job	1	1	2	2	2	3	3	4	4	4

x	operasi	sorted *x*	operasi'	job,op,mac
143	1	75	5	2, 1, 3
209	2	83	9	4, 1, 1
115	3	96	8	4, 2, 3
173	4	115	3	2, 2, 3
75	5	143	1	1, 1, 2
179	6	144	10	4, 3, 1
193	7	173	4	2, 3, 2
96	8	179	6	3, 1, 1
83	9	193	7	3, 2, 2
144	10	209	2	1, 2, 3

- Dari solusi ini bisa dihasilkan Gantt-chart untuk menghitung *makespan* sebagai berikut:



Multi Travelling Salesman Problem (m-TSP)

- *Multi Travelling Salesman Problem (m-TSP)* merupakan pengembangan dari TSP. Pada permasalahan ini terdapat lebih dari seorang *salesman.* Representasi permutasi yang digunakan untuk TSP bisa dimodifikasi sehingga bisa digunakan untuk mTSP.
- Modifikasi bisa dilakukan dengan menambahkan segmen untuk menunjukkan banyaknya daerah yang dikunjungi oleh tiap salesman. Misalkan terdapat 10 daerah yang harus dikunjungi (m=10) dan 3 orang salesman (n=3), maka sebuah chromosome bisa ditunjukkan seperti pada Gambar berikut.

posisi	1	2	3	4	5	6	7	8	9	10
gen	2	5	1	7	6	10	4	9	8	3

segmen 1

11	12	13
3	4	3

segmen 2

- Gen-gen pada segmen 1 (posisi 1 sampai 10) menunjukkan urutan daerah yang dikunjungi, segmen 2 (posisi 11 sampai 13) menunjukkan banyaknya daerah yang dikunjungi tiap salesman. Dengan asumsi bahwa tiap salesman berangkat dari kantor pusat (KP) dan harus kembali juga ke kantor pusat.

Multi Travelling Salesman Problem (m-TSP)

posisi	1	2	3	4	5	6	7	8	9	10
gen	2	5	1	7	6	10	4	9	8	3

segmen 1

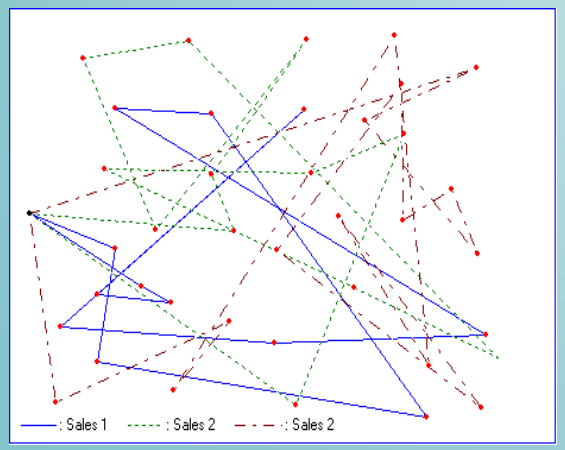
11	12	13
3	4	3

segmen 2

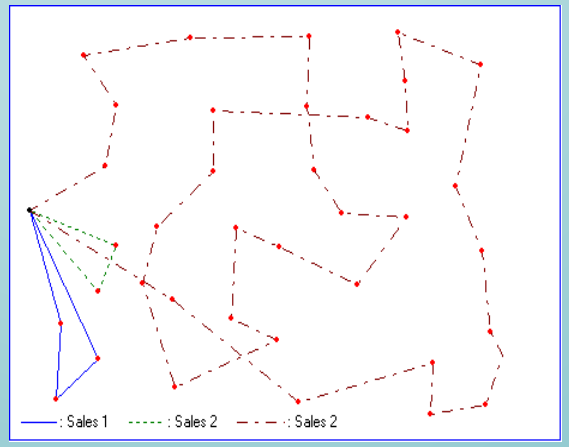
- Maka dari Gambar di atas dihasilkan rute tiap salesman sebagai berikut:

  Salesman 1 : KP → daerah 2 → daerah 5 → daerah 1 → KP
  Salesman 2 : KP → daerah 7→ daerah 6 → daerah 10 → daerah 4 → KP
  Salesman 3 : KP → daerah 9 → daerah 8 → daerah 3 → KP

- Untuk kasus dengan 40 node, Mahmudy (2008a) menunjukkan solusi yang dihasilkan oleh GAs seperti pada Gambar berikut.



Generasi 1 Total jarak = 6884,3



Generasi 50000 Total jarak = 2703

Tugas Kelompok

1. Jelaskan karakteristik dari masalah kombinatorial?
2. Untuk studi kasus TSP pada **Slide 5**, misal terdapat chromosome *P*=[3 4 2 1 5]. Hitung nilai total jarak dan fitnessnya!
3. Tentukan chromosome child untuk crossover pada representasi permutasi berikut!

cut point ↓

Parent 1	**2**	**1**	**5**	**4**	**3**
Parent 2	**3**	**5**	**1**	**4**	**2**
Child					

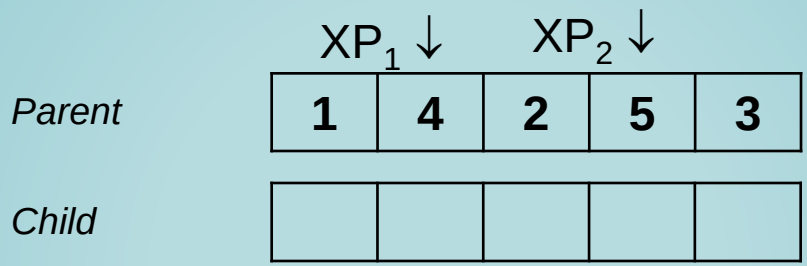
4. Tentukan chromosome child untuk *insertion mutation* pada representasi permutasi berikut!

IP ↓ SP ↓

Parent	**1**	**4**	**2**	**5**	**3**
Child					

Tugas Kelompok

5. Tentukan chromosome child untuk *reciprocal exchange mutation* pada representasi permutasi berikut!



6. Untuk dua individu pada permasalahan transportasi berikut tentukan *offspring* yang terbentuk dari proses *crossover*!

$$P_1 = \begin{bmatrix} 0 & 0 & 0 & 10 \\ 10 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \end{bmatrix} \qquad P_2 = \begin{bmatrix} 5 & 0 & 0 & 5 \\ 0 & 5 & 5 & 5 \\ 5 & 0 & 0 & 0 \end{bmatrix}$$

7. Untuk individu pada permasalahan transportasi berikut tentukan *offspring* yang terbentuk dari proses mutasi dengan menggunakan titik sudut yang diberi warna kuning!

$$P = \begin{bmatrix} 0 & 0 & 0 & 10 \\ 10 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Tugas Kelompok

8. Konversikan representasi permutasi *P*=[2 1 5 12 7 9 4 10 8 3 6 11] untuk menjadi solusi permasalah transportasi!
9. Hitung makespan untuk solusi FJSP dari representasi real *P*=[57 142 78 152 121 241 17 79 213 7] seperti pada contoh kasus di Slide 29 - 36!
10. Hitung makespan untuk solusi FJSP dari representasi real *P*=[129 114 147 186 220 58 159 11 31 235] seperti pada contoh kasus di Slide 29 - 36!