BAB 2

LANDASAN TEORI

2.1 Pengurutan Pekerjaan (*Job Sequencing*)

2.1.1 Deskripsi Umum

Dalam industri manufaktur, tujuan penjadwalan ialah untuk meminimasikan waktu dan biaya produksi, dengan cara mengatur apa yang harus dibuat, kapan, dengan pekerja yang mana, dan menggunakan alat apa. Penjadwalan produksi memiliki objektif untuk memaksimalkan efisiensi operasional dan mengurangi biaya. Salah satu permasalahan yang menarik dalam penjadwalan ialah *Job Sequencing*.

Job Sequencing adalah suatu pengurutan pekerjaan, dimana kombinasi urutan-urutan tersebut diukur berdasarkan performanya. *Job Sequencing* ini merupakan salah satu dari sekian banyak permasalahan pada analisis produksi. Permasalahan ini sangatlah rumit dan biasanya jauh dari penyelesaian yang optimal, terutama untuk kasus dimana pekerjaan sangatlah banyak. Menurut Cox et al., *Job Sequencing* dapat dianalogikan seperti ini: Diberikan sejumlah n *Job* untuk diproses, dan tiap-tiap *Job* memiliki waktu mulai, waktu proses, dan waktu dimana tanggal pekerjaan tersebut harus selesai, dan setiap pekerjaan diproses oleh mesin, dan permasalahannya adalah bagaimana mengurutkan mesin-mesin tersebut agar optimal dalam

suatu kondisi kriteria tertentu. Hal ini menunjukkan kapan pekerjaan harus selesai bila pesanan manufaktur ingin diselesaikan tepat waktu, dimana hal ini dapat memberikan pengaruh yang besar bagi produktivitas sebuah proses.

Dua hal yang menjadi kunci dalam *Job Sequencing*, menurut Wight, adalah prioritas dan kapasitas. Prioritas berarti "Apa yang harus dikerjakan lebih dahulu?" dan kapasitas berarti "Siapa yang harus mengerjakannya?".

Beberapa daftar dari kriteria performa untuk dioptimalkan dalam *Job Sequencing* adalah sebagai berikut:

- Rata-rata waktu produksi.
- Waktu menganggur dari mesin.
- Rata-rata perbedaan antara waktu selesai dengan waktu pengiriman.
- Rata-rata pekerjaan yang selesai sebelum waktunya.
- Rata-rata pekerjaan yang selesai melewati waktunya.
- Rata-rata waktu menunggu.
- Rata-rata pekerjaan di dalam *system.*
- Persentase dari jumlah pekerjaan yang memiliki perbedaan antara waktu selesai dengan waktu pengiriman.

Beberapa faktor yang dideskripsikan dan diklarifikasikan sebagai permasalahan dalam penjadwalan:

- Jumlah pekerjaan yang harus dijadwalkan.
- Jumlah mesin.

- Tipe manufaktur (*Flow Shop* atau *Job Shop*).
- Aturan datangnya pekerjaan (Statis atau Dinamis).
- Kriteria dimana alternatif jadwal lain akan dievaluasi.

Faktor pertama menjelaskan jumlah pekerjaan yang akan diproses, waktu untuk memproses, dan tipe mesin yang diperlukan. Sementara faktor kedua menjelaskan men*gen*ai jumlah mesin yang diperlukan. Faktor ketiga menjelaskan men*gen*ai jenis dari aliran material, jika aliran dari material kontinyu dan pekerjaan tersebut membutuhkan urutan mesin yang sama, maka disebut *Flow Shop Pattern*. Jika dimana aliran material berbeda-beda tiap pekerjaan, urutan mesin pun berbeda, maka disebut *Job Shop Pattern*. Dudek et al. melaporkan bahwa sebanyak 57% permasalahan pada industri adalah permasalahan *Job Shop*, sementara 20% adalah permasalahan *Flow Shop*. Laporan tersebut menyatakan bahwa kebanyakan dari penelitian men*gen*ai *Job Sequencing* berkutat seputar permasalahan *Flow Shop* dengan kriteria tujuan meminimalkan total keseluruhan waktu proses. Sementara pada penelitian men*gen*ai *Job Shop* dihadapkan dengan permasalahan dimana kriteria tidak hanya satu saja, melainkan lebih dari satu.

Pola kedatangan pekerjaan dideskripsikan sebagai statis atau dinamis. Pada pola statis, ada sejumlah n pekerjaan, setiap pekerjaan harus diproses oleh beberapa mesin, semua pekerjaan dapat diproses pada inisialisasi awal periode penjadwalan, dan selama pemrosesan, tidak ada pekerjaan baru yang datang. Pada pola dinamis, pekerjaan datang bergantung dari proses stokastik.

2.1.2 *Job Shop Sequencing*

Telah dijelaskan sebelumnya bahwa pengurutan pekerjaan pada produksi sangatlah rumit berdasarkan kombinasi-kombinasi yang sulit dipecahkan. Walaupun beberapa proses telah dicoba untuk diterapkan pada permasalahan *Flow Shop*, pada permasalahan *Job Shop*, sangat sedikit perkembangan yang terjadi. Hal ini bergantung dari kompleksnya faktor-faktor yang ada pada permasalahan *Job Shop*, seperti permasalahan dimana setiap *Job* memiliki spesifikasi dan urutan proses yang berbeda-beda. Oleh karena itu, maka simulasi menjadi satu-satunya jalan yang digunakan sebagai alat dalam penelitian men*gen*ai *Job Shop*. Pada landasan teori ini, akan dipaparkan beberapa aturan-aturan men*gen*ai pengurutan pekerjaan, dengan permasalahan sebagai berikut:

Tabel 2.1 Contoh Soal *Job Sequencing*

Pekerjaan	Waktu Proses	Pengiriman
1	5	15
2	8	10
3	6	15
4	3	25
5	10	20
6	14	40
7	7	45
8	3	50

- *FCFS*: Memilih pekerjaan berdasarkan yang datang dahulu, itu yang diproses.

Tabel 2.2 Contoh Penyelesaian *FCFS*

Pekerjaan	Mulai	Proses	Selesai	Pengiriman	Keterlambatan
1	0	5	5	15	-10
2	5	8	13	10	3
3	13	6	19	15	4
4	19	3	22	25	-3
5	22	10	32	20	12
6	32	14	46	40	6
7	46	7	53	45	8
8	53	3	56	50	6
Total Keterlambatan					39
Rata-rata Keterlambatan					6.5
Keterlambatan Maksimum					12
Jumlah Pekerjaan Terlambat					6

- *DDATE*: Memilih pekerjaan berdasarkan tanggal pengiriman yang terdekat.

Tabel 2.3 Contoh Penyelesaian *DDATE*

Pekerjaan	Mulai	Proses	Selesai	Pengiriman	Keterlambatan
2	0	8	8	10	-2
1	8	5	13	15	-2
3	13	6	19	15	4
5	19	10	29	20	9
4	29	3	32	25	7
6	32	14	46	40	6
7	46	7	53	45	8
8	53	3	56	50	6
Total Keterlambatan					40
Rata-rata Keterlambatan					6.67
Keterlambatan Maksimum					9
Jumlah Pekerjaan Terlambat					6

- *SLACK*: Memilih pekerjaan berdasarkan selisih waktu terkecil antara tanggal pengiriman dan waktu proses.

Tabel 2.4 *SLACK* Tiap Pekerjaan

Pekerjaan	Waktu Proses	Pengiriman	Slack
1	5	15	10
2	8	10	2
3	6	15	9
4	3	25	22
5	10	20	10
6	14	40	26
7	7	45	38
8	3	50	47

Tabel 2.5 Contoh Penyelesaian SLACK

Pekerjaan	Mulai	Proses	Selesai	Pengiriman	Keterlambatan
2	0	8	8	10	-2
9	8	6	14	15	-1
1	14	5	19	15	4
5	19	10	29	20	9
4	29	3	32	25	7
6	32	14	46	40	6
7	46	7	53	45	8
8	53	3	56	50	6
Total Keterlambatan					40
Rata-rata Keterlambatan					6.67
Keterlambatan Maksimum					9
Jumlah Pekerjaan Terlambat					6

- *SPT*: Memilih pekerjaan berdasarkan waktu proses yang paling cepat.

Tabel 2.6 Contoh Penyelesaian *SPT*

Pekerjaan	Mulai	Proses	Selesai	Pengiriman	Keterlambatan
4	0	3	3	25	-22
8	3	3	6	50	-44
1	6	5	11	15	-4
3	11	6	17	15	2
7	17	7	24	45	-21
2	24	8	32	10	22
5	32	10	42	20	22
6	42	14	56	40	16
Total Keterlambatan					62
Rata-rata Keterlambatan					15.5
Keterlambatan Maksimum					22
Jumlah Pekerjaan Terlambat					4

- *Critical Ratio*: Memilih pekerjaan berdasarkan faktor kritis terkecil, yaitu perbandingan antara tanggal pengiriman dan waktu proses.

Tabel 2.7 *Critical Ratio*

Pekerjaan	**Waktu Proses**	**Pengiriman**	***CR***
1	5	15	3.00
2	8	10	1.25
3	6	15	2.50
4	3	25	8.33
5	10	20	2.00
6	14	40	2.86
7	7	45	6.43
8	3	50	16.67

Tabel 2.8 Contoh Penyelesaian *Critical Ratio*

Pekerjaan	**Mulai**	**Proses**	**Selesai**	**Pengiriman**	**Keterlambatan**
2	0	8	8	10	-2
5	8	10	18	20	-2
3	18	6	24	15	9
6	24	14	38	40	-2
1	38	5	43	15	28
7	43	7	50	45	5
4	50	3	53	25	28
8	53	3	56	50	6
Total Keterlambatan					76
Rata-rata Keterlambatan					15.2
Keterlambatan Maksimum					28
Jumlah Pekerjaan Terlambat					5

Selain aturan-aturan yang tertera diatas, terdapat juga beberapa algoritma yang dapat digunakan untuk permasalahan pengurutan pekerjaan. Salah satunya adalah algoritma Hodgson, yang memiliki tahapan sebagai berikut:

1. Urutkan semua pekerjaan dengan aturan *DDATE*, jika terdapat suatu pekerjaan dengan nilai keterlambatan nol atau nilai keterlambatan positif, hentikan. Jika selain itu, lanjutkan ke langkah 2
2. Dimulai dari mula aturan *DDATE*, temukan task yang terlambat pertama kali. Jika tidak ada task yang terlambat, lanjutkan ke langkah 4, jika ada, lanjutkan ke langkah 3
3. Anggaplah suatu pekerjaan yang telat ada pada urutan ke i pada urutan. Maka periksa semua pekerjaan sampai pada urutan ke i dan identifikasi yang memiliki waktu proses terlama. Ambil pekerjaan itu dan singkirkan. Revisi semua perhitungan keterlambatan, dan kembali ke langkah 2.
4. Semua pekerjaan yang telah disingkirkan dapat ditaruh ke urutan terakhir dalam pekerjaan.

Berikut adalah contoh penyelesaian Algoritma Hodgson:

Tabel 2.9 Contoh Penyelesaian Algoritma Hodgson Iterasi 1

i	2	1	3	5	4	6	7	8
ti	8	5	6	10	3	14	7	3
ci	8	13	19	29	32	46	53	56
di	10	15	15	20	25	40	45	50
Li	-2	-2	4	9	7	6	8	6
Urutan Baru : ……2								

Tabel 2.10 Contoh Penyelesaian Algoritma Hodgson Iterasi 2

i	1	3	5	4	6	7	8
ti	5	6	10	3	14	7	3
ci	5	11	21	24	38	45	48
di	15	15	20	25	40	45	50
Li	-10	-4	1	-1	-2	0	-2
Urutan Baru: ……..2 - 5							

Tabel 2.11 Contoh Penyelesaian Algoritma Hodgson Iterasi 3

i	1	3	4	6	7	8
ti	5	6	3	14	7	3
ci	5	11	14	28	35	38
di	15	15	25	40	45	50
Li	-10	-4	-11	-12	-10	-12
Urutan Baru: 1 – 3 – 4 – 6 – 7 – 8 – 2 – 5						

Pada iterasi pertama, dicari pekerjaan yang memiliki Li positif pertama kali, didapat terjadi pada pekerjaan no. 3. Setelah itu, dari pekerjaan pada urutan paling awal sampai pada urutan pekerjaan yang memiliki Li positif pertama kali, dicari ti paling besar, dan didapat pada pekerjaan no. 2,

sehingga no. 2 dihapus dari daftar, dan ditaruh paling belakang. Pada iterasi selanjutnya, pekerjaan no. 2 sudah tidak ada pada daftar, dan lakukan hal yang sama, hingga terdapat pekerjaan no. 5 yang harus dihapus dari daftar dan ditaruh di paling belakang. Sehingga sampai pada iterasi ke-3 dimana tidak ada lagi pekerjaan yang memiliki Li positif, sehingga urutannya adalah urutan pekerjaan pada iterasi ke-3 ditambah dengan urutan pekerjaan yang sudah dihapus pada iterasi-iterasi sebelumnya.

Selain algoritma Hodgson, terdapat juga algoritma Wilkerson-Irwin, yang memiliki tahapan sebagai berikut:

1. Urutkan pekerjaan dengan aturan *DDATE*. Bandingkan dua pekerjaan pertama pada urutan, anggap pekerjaan ini adalah a dan b. Jika *max* (t_a,t_b) ≤ dari *max* (d_a, d_b), masukkan task a ke kolom α dan yang lainnya ke kolom β. Jika tidak, maka masukkan pekerjaan dengan waktu pemrosesan terpendek ke kolom α dan yang lainnya ke kolom β .Pekerjaan ketiga di dalam urutan dimasukkan ke kolom γ.
2. Bandingkan β dan γ untuk melihat jika β akan dijadwalkan bersama dengan α pada urutan. Jika tβ ≤ tγ atau jika Fα + *max* (tβ, tγ) ≤ *max* (dβ, dγ), pindahkan pekerjaan di kolom β ke α dan pekerjaan di kolom γ ke β. Pekerjaan selanjutnya dalam urutan *DDATE* akan dimasukkan ke dalam kolom γ. Fα adalah nilai kumulatif dari waktu proses pekerjaan yang ada pada kolom α Jika tidak ada lagi pekerjaan dari urutan *DDATE*, masukkan

pekerjaan α dan β ke dalam urutan lalu hentikan. Selain itu, ulangi langkah 2. Jika dua kondisi diatas tidak terpenuhi, pergi ke langkah 3.

3. Taruh β ke dalam urutan teratas pada *DDATE* tersisa dan pindahkan pekerjaan di γ ke kolom β. Bandingkan α dan β untuk melihat apakah β akan dijadwalkan bersama dengan α pada urutan. Jika $t\alpha \leq t\beta$ atau jika $F\alpha - t\alpha + max\ (t\alpha, t\beta) \leq max\ (d\alpha, d\beta)$, pindahkan pekerjaan pada kolom β ke kolom α dan pilih dua pekerjaan berikutnya pada daftar *DDATE* untuk menjadi β dan γ yang baru. Kembali ke langkah 2. Jika kondisi tidak terpenuhi, pergi ke langkah 4.
4. Masukkan pekerjaan pada kolom α kembali ke daftar teratas *DDATE* dan masukkan pekerjaan terakhir ke kolom α. Kembali ke langkah 3. Jika tidak ada pekerjaan di dalam urutan, masukkan β ke urutan dan pindahkan dua pekerjaan pertama pada daftar *DDATE* menjadi β dan γ. Kembali ke langkah 2

Berikut adalah contoh penyelesaian Algoritma Wilkerson-Irwin:

Tabel 2.12 Contoh Penyelesaian Algoritma Wilkerson-Irwin

Step	**α**	**β**	**γ**	**Step 2: Calculation**	**TB**	**Step 3: Calc**	**TA**
				$F\alpha + max\ (t\beta, t\gamma) \leq max\ (d\beta, d\gamma)$	$t\beta \leq t\gamma$	$F\alpha - t\alpha + max\ (t\alpha, t\beta) \leq max\ (d\alpha, d\beta)$	$t\alpha \leq t\beta$
2	2	1	3	8+6≤15 Yes	5≤6 Yes		
2	1	3	5	13+10≤20 No	6≤10 Yes		
2	3	5	4	19+10≤25 No	10≤3 No		
3	3	4				19-6+6≤25 Yes	6≤3 No
2	4	5	6	22+14≤40 Yes	10≤14 Yes		
2	5	6	7	32+14≤45 No	14≤7 No		

Tabel 2.13 Contoh Penyelesaian Algoritma Wilkerson-Irwin (lanjutan)

Step	**α**	**β**	**γ**	**Step 2: Calculation**	**TB**	**Step 3: Calc**	**TA**
3	5	7				32-10+10≤45 Yes	10≤7 No
2	7	6	8	39+14≤50 No	14≤3 No		
3	7	8				39-7+7≤56 Yes	7≤3 No
2	8	6					
Urutan Baru: 2 – 1 – 3 – 4 – 5 – 7 – 8 – 6							

Pada awalnya, urutan adalah 2 1 dan 3, dimasukkan ke kolom α, β dan γ. Pada iterasi selanjutnya, pekerjaan di kolom α dimasukkan ke dalam urutan pekerjaan yang baru, hingga didapati untuk selanjutnya urutan kolom kolom α, β dan γ berdasarkan pekerjaan yang tersisa pada *DDATE*. Pada saat kedua syarat No, yaitu tidak terpenuhi, maka pekerjaan pada kolom β, yaitu pekerjaan no. 5, disimpan sementara ke dalam urutan *DDATE*, sehingga pada saat kembali ke step 2, urutan *DDATE* berikutnya lah yang memasuki kolom α, β dan γ, dimana ada pekerjaan no. 5 disitu, begitu seterusnya sampai tidak ada lagi pekerjaan yang tersisa pada *DDATE*.

Pada umumnya, beberapa aturan yang ada terbukti lebih baik daripada aturan yang lain. Akan tetapi, hal ini juga dipengaruhi oleh banyak faktor, seperti kriteria performa, pola kedatangan pekerjaan, dan lainnya.

Ada aspek dinamis dari sebuah permasalahan *Job Shop Sequencing*. Sebuah jadwal dari beberapa pekerjaan telah dibuat, dan sementara jadwal tersebut dilaksanakan, pekerjaan baru akan datang. Muhlemann et al., mengatakan bahwa ada dua kondisi ekstrim untuk menjadwalkan sebuah *Job*

Shop: yang pertama yaitu membuat ulang jadwal setiap kali ada pekerjaan baru, atau langsung menyelesaikan jadwal yang ada sebelum menyelesaikan pekerjaan yang baru datang. Pada kasus kedua, ini mengubah permasalahan *Job Shop* dari dinamis menjadi statis. Sementara itu, hal yang baik adalah menggunakan sebuah komputer yang senantiasa *online* dan cepat tanggap terhadap setiap perubahan yang terjadi. Sebuah program komputer yang baik akan menganalisa status aktual saat sebuah jadwal berlangsung, dan mengaplikasikan berbagai macam kriteria, hingga menyusun ulang jadwal yang sedang berlangsung. Akan tetapi, membutuhkan suatu biaya yang mahal untuk menyusun sebuah program komputer yang efektif, dan ini menjadi suatu hambatan bagi banyak perusahaan maupun peneliti pada masa ini

2.2 Algoritma Genetika

2.2.1 Deskripsi Umum Algoritma Genetika

Algoritma Genetika adalah suatu algoritma yang didasarkan pada konsep evolusi dan perubahan *gen* pada makhluk hidup. Algoritma Genetika (AG) diciptakan oleh John Holland dari Universitas Michigan pada tahun 1975. Algoritma Genetika adalah sebuah teknik yang bersifat stokastik dan berbasis pada ide-ide evolusi dari seleksi alam dan genetika.

Pada dasarnya ada 4 kondisi yang sangat mempengaruhi proses evaluasi, yaitu:

- Kemampuan organisme untuk melakukan reproduksi
- Keberadaaan populasi organisme yang bisa melakukan reproduksi
- Keberagaman organisme dalam suatu populasi
- Perbedaan kemampuan untuk *survive*

Sesuatu yang stokastik adalah sebuah kejadian yang bersifat acak, dimana munculnya suatu kejadian tidak dapat diramalkan, akan tetapi, jika diukur dari seluruh distribusi observasi, biasanya akan mengikuti sebuah pola.

Algoritma Genetika sangat tepat digunakan untuk berbagai macam permasalahan yang kompleks dan sulit diselesaikan dengan metode konvensional. Metode ini dikategorikan sebagai pencari solusi *global* secara heuristik. AG merupakan metode yang terinspirasi oleh biologi evolusioner seperti kawin silang atau rekombinasi, mutasi, dan seleksi, khususnya pada seleksi, sesuai dengan prinsip-prinsip yang dicetuskan oleh Charles Darwin, yaitu "*Survival of the fittest*". Dikarenakan di alam ini, kompetisi antar individu untuk sumber daya yang terbatas mengakibatkan individu yang kuatlah yang akan bertahan dan mendominasi yang lemah. Individu yang lebih kuat (*fit*) akan memiliki tingkat *survival* dan reproduksi yang lebih tinggi daripada individu yang kurang *fit*. Pada kurun waktu tertentu, populasi secara keseluruhan akan lebih banyak memuat organisme yang *fit*.

AG diimplementasikan dengan bantuan simulasi komputer dalam sebuah populasi, dimana sebuah solusi, yakni sebuah individu, diwakili secara abstrak oleh apa yang disebut kromosom. Beberapa cara untuk menginterpretasikan solusi ke dalam sebuah kromosom, salah satunya adalah secara *binary code*, yaitu sebuah kromosom yang terdiri dari angka 0 dan 1. Akan tetapi, pemberian kode yang lain juga memungkinkan.

Walaupun menggunakan angka acak, AG sama sekali tidak menghasilkan nilai yang acak, sebaliknya mereka menggunakan informasi historis untuk mengarahkan pencarian ke daerah dengan performa yang lebih baik.

Perincian proses *encoding* dan *decoding* (yaitu proses dimana sebuah solusi dikodekan menjadi sebuah kromosom, dan sebaliknya) tidak dipahami sepenuhnya, namun beberapa teori yang dicetuskan oleh John Holland adalah sebagai berikut:

- Evolusi merupakan sebuah proses yang beroperasi pada kromosom.
- Seleksi alamiah adalah hubungan antara kromosom dengan performa dari struktur yang dikodekan. Proses seleksi alami menyebabkan kromosom yang lebih baik untuk memiliki kemampuan reproduksi yang lebih baik daripada mereka yang kurang baik.
- Proses reproduksi ialah titik dimana evolusi berjalan. Mutasi menyebabkan kromosom-kromosom menjadi berbeda dari induknya,

sedangkan proses kawin silang atau biasa disebut rekombinasi menghasilkan kromosom yang berbeda namun masih membawa sifat dari induknya.

- Evolusi biologis tidak memiliki ingatan, apapun yang diketahui oleh suatu individu terdapat dalam *gen* dan kromosom yang dibawa oleh individu tersebut.

2.2.2 Terminologi Algoritma Genetika

Didalam AG, terdapat banyak sekali istilah yang perlu diketahui. Istilah-istilah ini diambil dari istilah biologis, mengingat kesamaan proses dari AG ini terhadap proses yang terjadi pada makhluk hidup. Adapun beberapa persamaan istilahnya adalah sebagai berikut:

Tabel 2.14 Terminologi Algoritma Genetika

Terminologi AG	Arti
Gen/Locust	*Bit* yang ada di dalam sebuah *string*
Chromosome/Phenotype	Susunan dari banyak *bit* yang membentuk suatu *string*
Genotype	Parameter atau vektor solusi dari *Phenotype*
Parent	Kromosom orang tua
Offspring	Kromosom anak yang dihasilkan oleh proses operasi AG
Crossover	Kawin silang antara kedua kromosom orang tua
Mutation	Mutasi dari sebuah kromosom hingga menghasilkan kromosom yang berbeda
Fitness	Nilai kesesuaian dari sebuah kromosom

2.2.3 Struktur Umum Algoritma Genetika

Pada algoritma ini, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin yang dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi dikenal dengan istilah kromosom. Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut dengan istilah generasi. Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut dengan fungsi *fitness*. Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut. Generasi berikutnya dikenal dengan istilah anak atau biasa disebut *offspring*, terbentuk dari gabungan 2 kromosom generasi sekarang yang bertindak sebagai induk atau biasa disebut parent dengan menggunakan teknik penyilangan atau disebut juga *crossover*. Selain *operator* penyilangan, suatu kromosom dapat juga dimodifikasi dengan menggunakan *operator* mutasi. Populasi generasi yang baru di*gen*buk dengan cara menyeleksi nilai *fitness* dari kromosom induk dan nilai *fitness* dari kromosom anak, serta menolak kromosom-kromosom yang lainnya hingga ukuran populasi akan konstan. Setelah melalui beberapa generasi, maka algoritma ini akan konver*gen* ke kromosom terbaik.

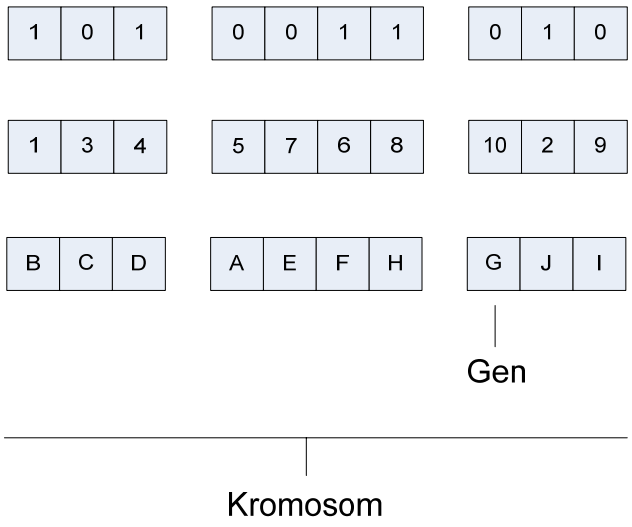
Adapun langkah-langkah dasar Algoritma Genetika adalah:

1. Membangkitkan inisialisasi awal dari populasi secara acak.
2. Mencari kesesuaian *fitness* dari populasi.
3. Pilih induk dari populasi.
4. Lakukan kawin silang antara induk yang menghasilkan populasi.
5. Lakukan mutasi pada individu dalam populasi.
6. Mencari kesesuaian *fitness* dari populasi.
7. Buang individu-individu yang kurang baik.
8. Kembali ke no. 3.

2.2.4 Komponen-Komponen Utama Algoritma Genetika

2.2.4.1 Teknik Penyandian

Teknik penyandian disini meliputi penyandian *gen* dari kromosom. *Gen* merupakan bagian dari kromosom. Satu *gen* biasanya akan mewakili satu variabel. *Gen* dapat direpresentasikan dalam bentuk: *string bit*, pohon, *array*, bilangan *real*, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk *operator* genetika.



Gambar 2.1 Contoh Struktur Data Algoritma Genetika

Demikian juga, kromosom dapat direpresentasikan dengan menggunakan:

- *String Bit* : 10011, 01101, 1101, dst
- Bilangan *real* : 65.65, -67.98, 562.88, dst
- Elemen permutasi : E2, E10, E5, dst
- Daftar aturan : R2, R3, R1, dst
- Elemen program : Pemrograman genetika
- Struktur lainnya

2.2.4.2 Inisialisasi

Ukuran populasi tergantung pada masalah apa yang akan dipecahkan dan jenis *operator* genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian harus dilakukan inisialisasi terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi kromosom dilakukan secara acak, namun demikian harus tetap memperhatikan *domain* solusi dan kendala permasalahan yang ada.

Penyelesaian menggunakan Algoritma Genetika membutuhkan dua hal yang harus di definisikan:

1. Sebuah representasi genetika dari solusi dari individu atau kromosom
2. Sebuah fungsi kesesuaian (*fitness function*) untuk mengevaluasi individu

Representasi standar dari solusi ialah sebuah *array of bit*s, atau serangkaian angka atau simbol yang mewakili sesuatu dalam permasalahannya. Sifat utama yang membuat representasi genetika ini mudah ialah karena bagian-bagian mereka mudah disusun karena ukurannya yang tetap, yang memungkinkan operasi persilangan sederhana. Representasi dengan ukuran yang variatif juga dapat digunakan, akan tetapi implementasi persilangan akan lebih sulit.

Fungsi kesesuaian didefinisikan untuk representasi genetika dan mengukur kualitas dari solusi yang diwakilkan. Fungsi kesesuaian selalu bergantung pada permasalahan. Contohnya, pada permasalahan sebuah tas, kami ingin memaksimalkan nilai total dari obyek yang dapat

dimasukkan ke dalam tas dengan sebuah kapasitas yang tetap. Representasi solusi dapat berupa *bit*s, dimana setiap *bit* mewakili sebuah objek, dan nilai dari setiap *bit* yaitu 0 atau 1 mengartikan apakah objek tersebut akan dimasukkan ke dalam tas atau tidak. Tidak setiap saat representasi seperti ini dapat digunakan, dimana ukuran dari objek dapat melebihi ukuran tas. Oleh karena itu diperlukan yang namanya kesesuaian dari solusi, yaitu ialah jumlah nilai dari seluruh objek di dalam tas fisibel, bila tidak maka tidak sesuai. Dalam beberapa permasalahan, bahkan sulit untuk mendefinisikan kesesuaian, dan dalam kasus-kasus seperti ini digunakan Algoritma Genetika interaktif.

Setelah mendefinisikan representasi genetika dan fungsi kesesuaian, AG berlanjut untuk menginisialisasi sebuah populasi. Pada awalnya beberapa solusi dari permasalahan yang terkait dikumpulkan secara acak, kelompok solusi ini akan membentuk sebuah populasi. Pada kasus pengurutan pekerjaan pada *Job Shop*, solusi berupa beberapa urutan kombinasi pekerjaan. Ukuran populasi bergantung pada sifat dari permasalahan, tetapi biasanya mengandung banyak sekali solusi yang mungkin diangkat. Secara tradisional, solusi dibangkitkan secara acak, menutupi seluruh jangkauan kemungkinan lingkup penyelesaian (*search space*). Namun terkadang, solusi permasalahan dapat dibibitkan pada bagian-bagian tertentu dimana solusi optimal dapat ditemukan.

2.2.4.3 Pemilihan *Operator*

Setelah sebuah populasi awal dibangkitkan, algoritma melewati beberapa *operator*, yakni:

- Seleksi, yang memberikan hasil kromosom terkuat
- Genetika, berupa kawin silang (*crossover*) dan mutasi (*mutation*)

2.2.4.4 *Operator* Seleksi

Operator ini memiliki beberapa sifat dasar yaitu:

1. Ide kunci: memberikan preference kepada solusi yang lebih baik, hingga memungkinkan mereka untuk menurunkan *gen-gen* mereka kepada generasi yang berikutnya
2. Tingkat kebaikan setiap individu bergantung pada kesesuaiannya
3. Kesesuaian dapat ditetapkan oleh sebuah fungsi obyektif atau sebuah penilaian subyektif

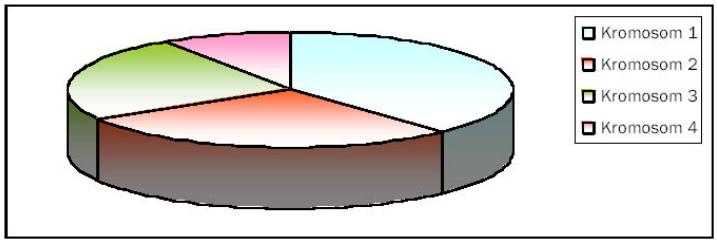
Pada setiap generasi, sebuah proporsi dari populasi yang ada dipilih untuk menurunkan generasi yang baru. Solusi individu dipilih melalui proses yang berbasis kesesuaian (*fitness-based*), dimana solusi yang lebih sesuai atau kuat diukur dengan fungsi kesesuaian cenderung lebih mudah untuk terpilih. Beberapa metode seleksi mengukur tingginya kesesuaian dari setiap solusi dan lebih memilih yang terbaik. Beberapa

metode lainnya hanya mengukur sampel acak dari populasi, karena proses ini dapat memakan waktu banyak.

Kebanyakan fungsi yang digunakan bersifat stokastik dan dirancang sehingga proporsi kecil dari mereka yang kurang sesuai tetap dapat kemungkinan untuk terpilih. Hal ini dilakukan untuk menjaga supaya perbedaan populasi tetap besar, yang menghindari konvergensi premature pada solusi yang buruk. Beberapa metode seleksi adalah:

1. *Roulette-Wheel Selection*

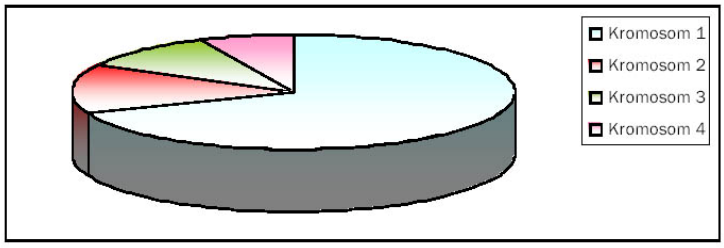
   Induk dipilih menurut kesesuaiannya. Semakin baik kromosom, semakin tinggi kesempatannya untuk terpilih. Bila semua kromosom dari populasi diletakkan pada sebuah roda rolet, setiap kromosom memiliki luas bagian yang berdasarkan fungsi kesesuaiannya, seperti pada gambar dibawah. Angka acak dibangkitkan untuk memilih kromosom. Kromosom yang memiliki kesesuaian lebih besar memiliki probabilitas terpilih yang lebih besar. Kekurangan metode ini ialah bila kromosom terbaik memiliki kesesuaian yang sangat besar dari seluruh roda, maka kromosom lainnya akan memiliki kesempatan yang sangat kecil untuk terpilih.



Gambar 2.2 *Roulette Wheel Selection*

2. *Rank-based fitness assignment*

Pertama populasi diurutkan, lalu setiap kromosom mendapatkan ranking dari urutan kesesuaiannya. Makin besar kesesuaiannya, maka rankingnya makin baik. Setelah itu dimasukkan ke dalam roda rolet. Bedanya dengan metode roda rolet diatas adalah jika metode roda rolet sebelumnya tidak diurutkan, maka metode rank ini mengurutkan, sehingga roda rolet yang terdapat akan teratur urutannya. Namun metode ini akan menuntun ke konvergensi yang lebih lambat, karena kromosom yang terbaik tidak jauh berbeda dengan yang lain.



Gambar 2.3 Rank-Based *Fitness* Assignment

3. *Steady-state selection*

Bukan merupakan metode seleksi, tetapi tujuan utama dari metode ini ialah sebagian besar dari kromosom harus bertahan ke generasi berikutnya. Dalam setiap generasi, beberapa kromosom dengan kesesuaian yang tinggi dipilih untuk menghasilkan turunan baru. Lalu beberapa kromosom dengan kesesuaian yang rendah dikeluarkan dan turunan-turunan baru menggantikannya. Sisa dari populasi bertahan ke generasi berikutnya.

4. *The Elitism*

Bila membuat populasi baru dengan persilangan dan mutasi, kita memiliki kesempatan yang besar untuk kehilangan kromosom yang terbaik. Metode ini menyalin beberapa kromosom terbaik ke populasi baru, sisanya dilakukan dengan cara biasa. *The Elitism* dapat meningkatkan kinerja AG karena menghindari hilangnya solusi terbaik.

5. Tournament Selection

Metode ini menjalankan sebuah persaingan diantara beberapa individu yang dipilih secara acak dari populasi dan memilih pemenangnya untuk disilangkan.

Tekanan persaingan dapat dirubah dengan mengubah ukuran persaingan. Jika lebih besar, maka individu yang lebih lemah memiliki kesempatan yang lebih kecil untuk terpilih. Kelebihan metode ini adalah bekekerja pada arsitektur yang berbeda-beda dan mengizinkan tekanan persaingan untuk mudah diubah.

2.2.4.5 *Operator* Genetika

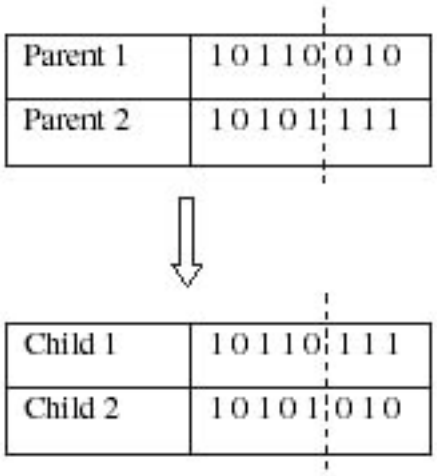
Langkah berikutnya ialah membangkitkan populasi solusi generasi kedua dari mereka yang terpilih dengan menggunakan *operator* genetika, yaitu persilangan dan mutasi. Untuk setiap solusi yang dihasilkan, sepasang solusi induk dipilih untuk member keturunan dari kumpulan yang dipilih sebelumnya. Dengan menghasilkan sebuah solusi turunan menggunakan persilangan dan mutasi, sebuah solusi baru akan muncul yang memiliki sifat-sifat dari induknya. Induk-induk baru dipilih untuk setiap turunan, dan proses ini berlanjut hingga sebuah solusi populasi dengan ukuran yang sesuai telah didapatkan. Proses ini akhirnya akan menghasilkan populasi kromosom generasi berikutnya yang berbeda dengan generasi awal. Secara umum, rata-rata kesesuaian populasi akan meningkat dengan prosedur ini, karena hanya individu terbaik dari generasi pertama telah dipilih untuk member keturunan, bersama dengan mereka yang kurang baik. Beberapa parameter Algoritma Genetika berdasarkan para ahli:

- Menurut De Jong, untuk permasalahan dengan solusi besar, digunakan Pop; P_c; P_m : 50; 0.6; 0.001
- Menurut Grefenstette, untuk jika fitness dari individu yang terbaik yang diambil, digunakan Pop; P_c; P_m : 80; 0.45; 0.01
- Jika rata-rata fitness tiap generasi digunakan sebagai indikator, digunakan Pop; P_c; P_m : 30; 0.95; 0.01

Beberapa dari sekian banyak *operator* genetika akan dijelaskan di bawah ini:

1. *Crossover*
   a. *One Point Crossover* (Rekombinasi satu titik)

      Penyilangan satu titik adalah penyilangan yang terjadi pada *gen* antara dua invididu kromosom induk setelah satu titik dari kromosom mereka.

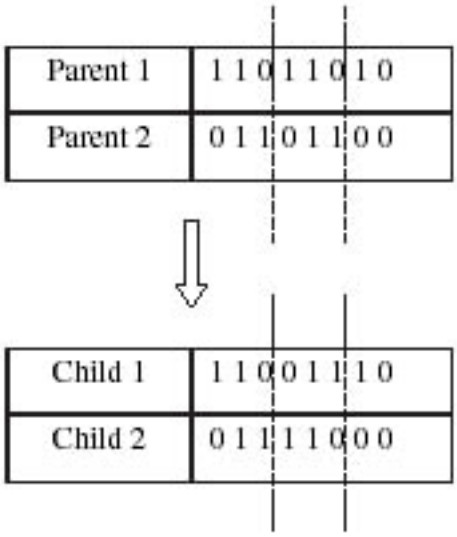


Gambar 2.4 *One Point Crossover*

Pada gambar diatas, terlihat bahwa kawin silang terjadi setelah *bit* ke-5 pada kromosom. Dimana *bit* ke 6, 7, 8 dari kromosom induk saling ditukar, sehingga menghasilkan kromosom anak.

b. *Two Point Crossover* (Rekombinasi dua titik)

Rekombinasi dua titik terjadi pada dua titik pada kromosom. Sehingga yang ditukar hanyalah *gen-gen* yang ada diantara titik-titik tersebut.



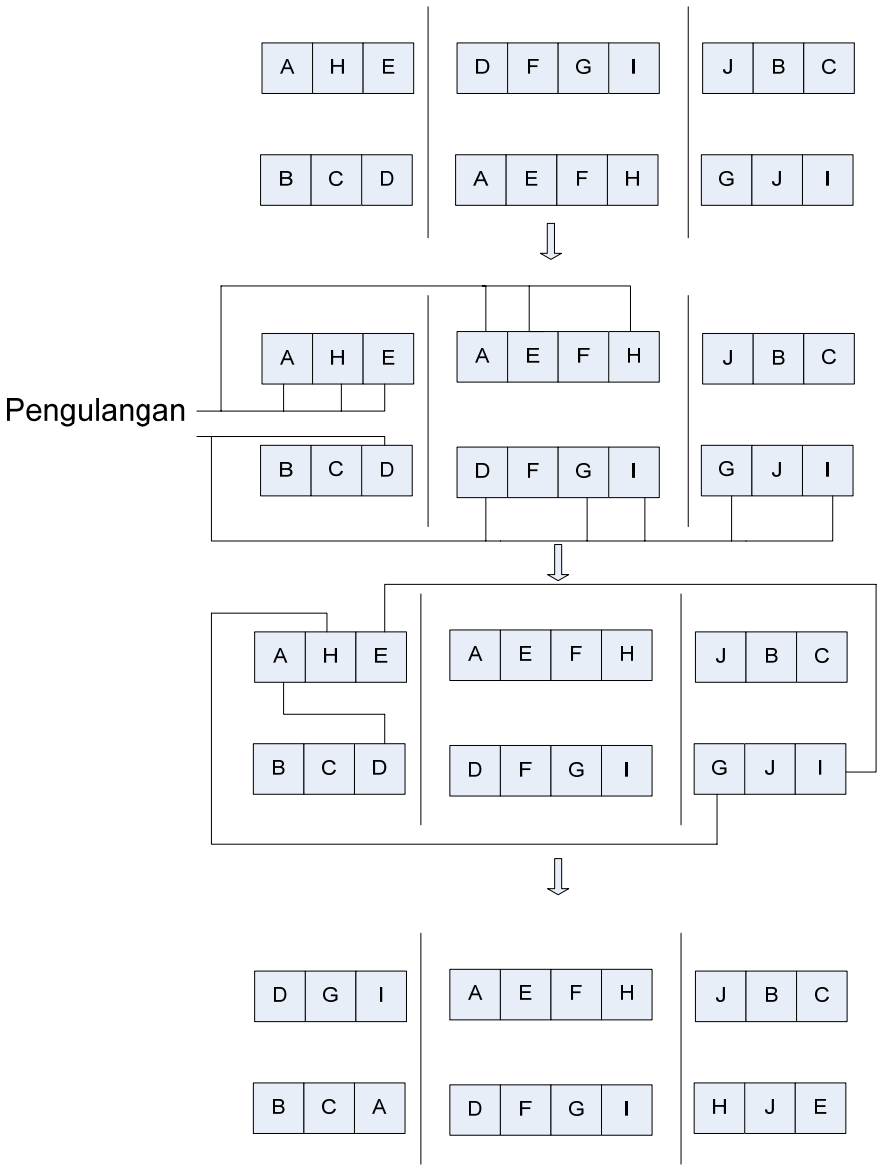
Gambar 2.5 *Two Point Crossover*

Pada gambar diatas dapat dilihat bahwa rekombinasi terjadi setelah *bit* ke – 3 dan sebelum *bit* ke -7, sehingga yang ditukar hanyalah *gen* ke 4, 5, dan 6.

c. Partially Matched Crossover

Pada beberapa kasus, jenis-jenis rekombinasi diatas tidak dapat diaplikasikan, karena kromosom akan menjadi suatu individu yang tidak fisibel. Sebagai contoh pada kasus *TSP* (*Travelling Salesman Problem*) dan pada kasus *Job Sequencing*.

Jika menggunakan metode diatas, maka akan terjadi pengulangan pada kota-kota atau *Job* yang diurutkan, sehingga banyak terjadi kromosom yang tidak fisibel. Oleh karena itu, ada suatu metode yang namanya *PMX* (*Partially Matched Crossover*).



Gambar 2.6 *Partially Matched Crossover*

Pada *PMX*, pertama-tama dilakukan *Two Point Crossover*. Setelah itu, setiap *gen* yang ada diluar zona *Two Point Crossover* yang memiliki kembaran di dalam area *Two Point Crossover*, saling ditukarkan antara kromosom orang tua masing masing.

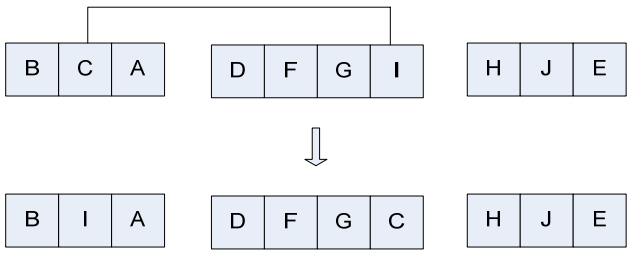
Ciri-ciri *operator* persilangan adalah:

- Faktor cirri utama dari AG dibandingkan dengan teknik optimasi lainnya.
- Untuk menentukan seberapa banyak dari populasi yang akan disilangkan, ditetapkan sebuah peluang persilangan (*crossover probability*).
- Dua individu diambil dari populasi menggunakan *operator* seleksi untuk menjadi induk dan dikawinkan.
- Sebuah atau beberapa titik pada kromosom dipilih secara acak, atau menurut aturan.
- Dua turunan yang dihasilkan dari perkawinan ini dimasukkan ke dalam populasi dari generasi berikutnya.
- Dengan menggabungkan beberapa bagian dari individu yang baik, proses ini cenderung akan membuat individu yang lebih baik lagi.

2. *Mutation*

Untuk operasi mutasi, hanya satu kromosom yang diseleksi. Penelitian ini menggunakan *operator* mutasi bernama *Order-Based Mutation*. Dimana *gen-gen* didalam kromosom hanya akan ditukar posisinya.

Tujuan dari mutasi adalah untuk menjaga keanekaragaman dalam populasi, dan menghindari terjadinya konvergensi premature, atau dikenal juga dengan *local optimum*.



Gambar 2.7 *Order-Based Mutation*

Menggunakan *operator-operator* genetika akan memiliki beberapa efek, yakni:

- Menggunakan seleksi saja akan cenderung mengisi populasi dengan salinan dari individu yang terbaik dalam populasi.
- Menggunakan *operator* seleksi dan persilangan saja akan cenderung mengakibatkan algoritma untuk berkonver*gen*si pada solusi yang baik namun sub-optimal.
- Menggunakan mutasi saja akan memberikan sifat acak dalam *search space.*
- Menggunakan seleksi dan seleksi dan mutasi akan menghasilkan sebuah algoritma yang bersifat *parallel*, dan *hill climbing.*

2.2.4.6 Terminasi

Proses generasional ini diulangi sampai sebuah kondisi terminasi sudah tercapai. Kondisi terminasi yang umum adalah:

- Solusi yang memuaskan kriteria sudah ditemukan.
- Jumlah generasi tetap sudah tercapai.
- Anggaran yang dialokasikan (waktu/uang komputasi) sudah tercapai.
- Kesesuaian solusi yang terbaik sedang mencapai atau telah mencapai sebuah plateau, atau dataran tinggi ynag rata, sehingga iterasi-iterasi berikutnya tidak lagi menghasilkan nilai yang lebih baik.
- Inspeksi manual.
- Kombinasi dari yang diatas.

2.2.5 Kelebihan dan Kekurangan Algoritma Genetika

Beberapa kelebihan dari metode ini adalah:

- Proposal atau solusi yang buruk tidak mempengaruhi solusi akhir karena langsung dibuang.
- Sifat induktif dari AG berarti metode ini tidak perlu mengetahui peraturan apapun dari permasalahan, ia bekerja berdasarkan peraturan internalnya sendiri.
- Dapat memberikan informasi mengenai stabilitas dari solusi.

- Dapat digunakan untuk memberikan solusi optimasi yang multidimensional.

Beberapa kekurangan dari metode ini adalah:

- Tidak selalu menemukan *global optimum* yang pasti.
- Karena proses evolusi induktif, pada alam hidup tidak berputar menuju solusi yang baik, tetapi berputar menjauhi yang buruk. Ini dapat mengakibatkan sebuah spesies berputar menuju kebuntuan evolusioner.
- Membutuhkan jumlah evaluasi yang besar dari fungsi kesesuaian.
- Memerlukan komputer untuk penanganannya.

2.2.6 Simulasi Sederhana Permasalahan Algoritma Genetika

Sebuah contoh permasalahan. Carilah sebuah nilai variabel x dan y dengan fungsi x + y = 15. Dengan beberapa parameter AG sebagai berikut:

- Ukuran populasi = 4
- *Crossover probability* (Pc) = 0.7
- *Mutation* probability (Pm) = 0.1
- *Max Generation* = 3
- Struktur data biner, dengan perincian sebagai berikut:
  - **0**: **0000**
  - **1**: **0001**
  - **2**: **0010**

- **3**: **0011**
- **4**: **0100**
- **5**: **0101**
- **6**: **0110**
- **7**: **0111**
- **8**: **1000**
- **9**: **1001**

- Fungsi kesesuaian

$$f = \frac{Hasil}{TargetNumber}$$

Selanjutnya, tahapan tahapan dibagi dalam beberapa iterasi / generasi:

1. Generasi 0 (Inisialisasi awal)
   - Bangkitkan populasi acak

Tabel 2.15 Populasi Kromosom pada Generasi 0

Kromosom	**Genotip**	**Nilai *Fitness***
10000101	4 5	9/15
00000001	0 1	1/15
00110101	3 5	8/15
10000000	8 0	8/15

   - Seleksi dan kawin silang

P1: 10000000

P2: 00110101

One Point Crossover setelah *bit* ke 3

C1: 10010101

C2: 00100000

- Seleksi dan mutasi

  P: 00000001

  Flip Mutation pada *bit* ke 4

  C: 00010001

- Populasi baru bertambah

  Tabel 2.16 Total Populasi Kromosom pada Generasi 0

Kromosom	**Genotip**	**Nilai *Fitness***
10000101	4 5	9/15
00000001	0 1	1/15
00110101	3 5	8/15
10000000	8 0	8/15
10010101	9 5	14/15
00100000	4 0	4/15
00010001	1 1	2/15

- Kromosom-kromosom yang nilai *fitness*nya rendah dibuang

  Tabel 2.17 Populasi Baru Kromosom pada Generasi 0

Kromosom	**Genotip**	**Nilai *Fitness***
10000101	4 5	9/15
00110101	3 5	8/15
10000000	8 0	8/15
10010101	9 5	14/15

2. Generasi 1

   - Populasi berdasarkan generasi sebelumnya

     Tabel 2.18 Populasi Kromosom pada Generasi 1

Kromosom	**Genotip**	**Nilai *Fitness***
10000101	4 5	9/15
00110101	3 5	8/15
10000000	8 0	8/15
10010101	9 5	14/15

   - Seleksi dan kawin silang

     P1: 00110101

     P2: 10000000

     One Point Crossover setelah *bit* ke 4

     C1: 00110000

     C2: 10000101

   - Seleksi dan mutasi

     P: 10000101

     Flip Mutation pada *bit* ke 7

     C: 10000111

- Populasi baru bertambah

Tabel 2.19 Total Populasi Kromosom pada Generasi 1

Kromosom	**Genotip**	**Nilai *Fitness***
10000101	4 5	9/15
00110101	3 5	8/15
10000000	8 0	8/15
10010101	9 5	14/15
00110000	3 0	3/15
10000101	8 5	13/15
10000111	8 7	15/15

- Kromosom-kromosom yang nilai *fitness*nya rendah dibuang

Tabel 2.20 Populasi Baru Kromosom pada Generasi 1

Kromosom	**Genotip**	**Nilai *Fitness***
10000101	4 5	9/15
10010101	9 5	14/15
10000101	8 5	13/15
10000111	8 7	15/15

- Didapatkan bahwa kromosom yang memiliki nilai kesesuaian paling tinggi adalah kromosom 1000 0111, dengan *geno*tip 8 dan 7. Hal ini berarti nilai X = 8 dan Nilai Y = 7