**BAB II**

**TINJAUAN PUSTAKA DAN LANDASAN TEORI**

1. **Tinjauan Pustaka**

Penulis senantiasa mempelajari hasil penelitian terdahulu, sebagai acuan guna membangun wawasan berpikir dalam penelitian ini. Beberapa hasil penelitian terdahulu yang dijadikan referensi adalah sebagai berikut.

Penelitian yang dilakukan Hakim (2016), untuk menyelesaikan masalah penentuan komposisi pakan ayam petelur menggunakan Algoritma Memetika (AM). Tujuan dari penelitian ini adalah untuk menghasilkan sistem yang dapat dengan optimal menentukan komposisi asupan makanan yang tepat dikonsumsi untuk memenuhi kebutuhan nutrisi ayam petelur agar menghasilkan telur dengan kualitas yang baik. AM pada penelitian ini telah berhasil cukup efektif untuk menyelesaikan kasus penentuan komposisi asupan makanan ayam. Persamaan dengan penelitian yang dilakukan penulis yaitu algoritma memetika digunakan sebagai pendekatan untuk melakukan optimasi. Perbedaan dengan penelitian yang penulis lakukan yaitu menggunakan AM untuk melakukan optimasi penjadwalan tenaga kebidanan sedangan pada penelitian oleh Hakim yaitu menggunakan AM untuk melakukan optimasi komposisi asupan makanan ayam.

Penelitian yang dilakukan Altintas *et al.*  (2014), untuk menyelesaikan masalah penjadwalan ujian pada universitas Suleyman Demirel University (SDU) Turki, menggunakan AM. Pada penelitian ini AM dimodifikasi menjadi *Self-Generating Multimeme Algorithm* (*SGMA*) yang mengelola 6 mutasi, 2 *crossover,* dan 2 *hill climbing.* Pemilihan operator saat proses evolusi akan ditentukan menurut skor yang diperoleh melalui operasi *roulette wheel.* Hasil SGMA pada penelitian ini cukup baik untuk menyelesaikan masalah penjadwalan ujian. Persamaan dengan penelitian yang dilakukan penulis yaitu AM digunakan sebagai pendekatan untuk melakukan optimasi penjadwalan. Perbedaan pada penelitian yang penulis lakukan yaitu mengoptimasi penjadwalan tenaga kebidanan sedangkan pada penelitian oleh Altintas *et al.* yaitu mengoptimasi penjadwalan ujian perkuliahan.

6

Penelitian yang dilakukan Gonsalves dan Kuwata (2015), untuk menyelesaikan masalah *Nurse Scheduling Problem* menggunakan algoritma memetika. Penelitian ini berusaha memecahkan permasalahan *Nurse Scheduling Problem* menggunakan algoritma memetika dengan memakai data yang berasal di Queen’s Medical Centre (QMC), Nottingham, UK. Penelitian ini menggunakan 6 *hard constraint* dan 5 *soft constraint* sebagai syarat untuk mencapai hasil yang optimal. Solusi yang didapatkan mencapai hasil yang optimal dengan dapat meminimalkan *soft constraint* dan *hard constraint* dengan baik. Persamaan dengan penelitian yang dilakukan penulis yaitu algoritma memetika digunakan sebagai pendekatan untuk melakukan optimasi penjadwalan. Perbedaan pada penelitian yang penulis lakukan yaitu mengoptimasi penjadwalan tenaga kebidanan dan menggunakan aplikasi berbasis web sedangkan pada penelitian oleh Gonzalves dan Kuwata yaitu mengoptimasi penjadwalan suster dan menggunakan aplikasi berbasis desktop.

Penelitian yang dilakukan Albar (2013), untuk menyelesaikan masalah penjadwalan kuliah menggunakan Algoritma Genetika Tabu Search dan AlgoritmaMemetika dengan membandingkan hasil serta performa kedua algoritma tersebut. Variabel yang di analisa antara lain *soft constraint, hard constraint, total soft constraint, total hard constraint,* dan waktu eksekusi yang di butuhkan aplikasi untuk mencapai hasil yang optimal. Hasil keluaran solusi Algoritma *Genetika Tabu Search* dan Algoritma *Memetika* dapat mencapai optimal yang membedakanya terdapat pada variabel pengujian waktu eksekusi pada Algoritma Genetika Tabu Search dua hingga tiga kali lebih lambat dibandingkan dengan Algoritma Memetika*.* Persamaan dengan penelitian yang dilakukan penulis yaitu algoritma memetika digunakan sebagai pendekatan untuk melakukan optimasi penjadwalan. Perbedaan pada penelitian yang penulis lakukan yaitu mengoptimasi penjadwalan tenaga kebidanan sedangkan pada penelitian Albar mengoptimasi penjadwalan kuliah.

1. **Landasan Teori**

**2.2.1. Sistem**

Menurut Jogiyanto dalam Andri *et al.* (2013) sistem adalah jaringan daripada elemen-elemen yang saling berhubungan, membentuk suatu kesatuan untuk melaksanakan suatu tujuan pokok dari sistem tersebut. Suatu sistem bisa dikatakan merupakan kombinasi atas beberapa komponen yang bekerja secara bersama-sama untuk melakukan suatu pekerjaan tertentu.

Menurut Jogiyanto dalam Andri *et al.* (2013) suatu sistem memiliki karakteristik atau sifat-sifat tertentu yaitu memiliki komponen, batas sistem, lingkungan luar sistem, penghubung, masukan, keluaran, dan tujuan. Adapun pengertian dari masing-masing karakteristik Sistem adalah sebagai berikut:

1. Komponen Sistem (*Component System*)

Komponen-komponen sistem atau elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

2. Batas Sistem (*Boundary System*)

Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainya atau dengan lingkungan lainya. Batas ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan.

3. Lingkungan Luar Sistem (*Environment System*)

Apapun siluar batas dari sistem yang mempengaruhi operasi sistem dinamakan lingkungan luar. Lingkungan luar sistem dapat bersifat menguntungkan dengan demikian harus tetap dijaga dan dipelihara.

4. Penghubung Sistem (*Interface System)*

Penghubung merupakan media penghubung antar subsistem. Melalui penghubung ini memungkinkan Sumber-Sumber daya mengalir dari satu subsistem ke subsistem lainya.

5. Masukan Sistem (*Input System*)

Masukan sistem adalah energi yang dimasukan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukan supaya sistem tersebut dapat beroperasi. *Signal input* adalah energi yang diproses untuk didapatkan keluaran.

6. Keluaran Sistem (*Output System*)

Keluaran adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. Misalnya untuk sistem komputer, panas yang dihasilkan adalah keluaran yang tidak berguna.

7. Pengolah Sistem (*Process System*)

Suatu sistem dapat mempunyai suatu bagian pengolahan yang akan merubah masukan menjadi keluaran.

8. Sasaran Sistem (*Objective System*)

Suatu sistem pasti mempunyai sasaran. Jika suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem.

**2.2.2. Penjadwalan**

Penjadwalan Menurut Yohanes (2013) penjadwalan secara umum dapat diartikan seperti: “*scheduling is the allocation of recource evertime to perform collection of risk*”,yang artinya penjadwalan adalah pengalokasian Sumber daya yang terbatas untuk mengerjakan sejumlah pekerjaan. Masalah penjadwalan memiliki banyak bentuk tergantung dari permasalahanya.

Batasan (*constraint*) pada penjadwalan merupakan suatu aturan-aturan yang jika tidak dipenuhi akan menurunkan kualitas kelayakan suatu hasil penjadwalan. Adapun batasan yang terdapat pada penjadwalan biasanya terbagi menjadi dua jenis yaitu batasan keras (*hard constraint*) dan batasan lunak (*soft constraint*). *Hard constraint* merupakan batasan yang harus dipenuhi untuk menghasilkan jadwal yang layak sedangkan *soft constraint* merupakan batasan tambahan yang masih bisa ditolerir artinya jika *soft constraint* terpenuhi maka akan menambah kualitas penjadwalan dan jika saja tidak terpenuhi maka jadwal masih dianggap layak.

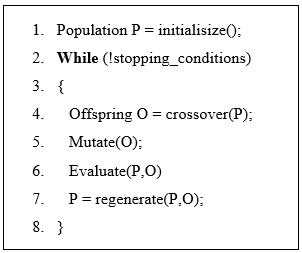
**2.2.3. Penjadwalan Tenaga Kebidanan**

Penjadwalan tenaga kebidanan merupakan penjadwalan yang masuk dalam kategori Penjadwalan Tenaga Kerja (*Manpower Scheduling*). Kualitas penjadwalan secara langsung akan mempengaruhi kualitas pelayanan kebidanan. Jika kualitas penjadwalan baik maka kualitas pelayanan dapat maksimal dilakukan. Penjadwalan tenaga kebidanan adalah pembuatan jadwal mingguan atau bulanan untuk sejumlah bidan dengan memberikan pergeseran (*shift*) yang mungkin dilakukan untuk setiap bidan. Jadwal harus memenuhi persyaratan dan ketentuan instansi untuk penjadwalan tenaga kebidanan dan memenuhi tuntutan jumlah bidan yang berbeda untuk ditugaskan pada setiap *shift* sementara juga harus memperhitungkan keinginan dan kebutuhan setiap bidan sebaik mungkin.

**2.2.4.** **Algoritma Memetika (AM)**

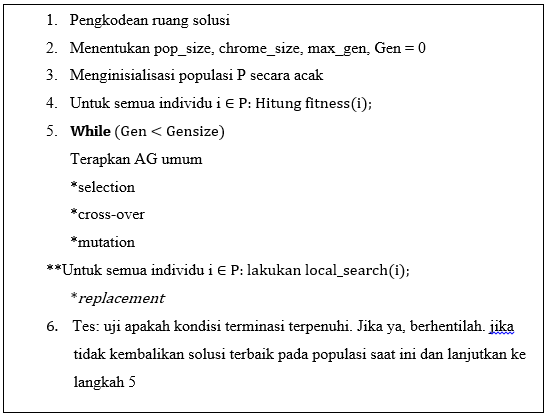
AM menggabungkan AE dan PL untuk mendapatkan solusi terbaik dengan lebih efektif dan efisien dikarenakan solusi yang buruk atau tidak layak dapat berubah menjadi solusi yang baik melalui proses peningkatan pada PL. AE pada AM pada umumnya merujuk pada Algoritma Genetika (AG) sehingga karakteristik AM identik dengan karakteristik dalam AG dengan penambahan operator pencarian lokal. Sehingga AM bisa dikatakan adalah perluasan dari AG.

AG bekerja dalam ruang pencarian yang menggunakan banyak individu sekaligus, sehingga kemungkinan AG untuk terjebak pada ekstrim lokal lebih kecil dibandingkan metode lain. Namun telah diketahui bahwa AG tidak cukup baik untuk mencari solusi yang sangat dekat dengan solusi optimal. Kekurangan tersebut dapat diakomodasi dengan cara menambahkan metode PL pada AG. Hasil penggabungan ini dikenal dengan sebutan AM. Pada AM, PL bertujuan untuk melakukan perbaikan lokal yang dapat diterapkan sebelum dan atau sesudah proses seleksi, *crossover,* dan mutasi. PL juga dapat berguna untuk mengontrol besarnya ruang pencarian solusi. AM dapat memberikan hasil yang lebih baik daripada AG, namun memerlukan waktu komputasi yang lebih lama dibanding AG. (Albar, 2013). Perbedaan proses pada AG dan AM dapat dilihat masing-masing pada gambar 2.1 dan gambar 2.2.



Gambar 2.1. *Pseudocode* Algoritma Genetika (AG)

(Sumber: Otani *et al.*, 2013)



Gambar 2.2. *Pseudocode* Algoritma Memetika (AM)

(Sumber: Anu, 2013)

**2.2.4.1. Algoritma Evolusi (AE)**

Menurut Suyanto (2014), Algoritma AE merupakan algoritma-algoritma optimasi yang berbasis “teori genetika” dari “teori evolusi” yang disampaikan oleh Charles Darwin. AE merepresentasikan suatu kandidat solusi sebagai satu individu dan sekumpulan kandidat solusi sebagai populasi. AE bekerja dengan cara membangkitkan, menguji, dan berusaha memperbaiki sekumpulan kandidat solusi sampai ditemukanya satu solusi yang bisa diterima. Rekombinasi dan mutasi, yang bersifat stokastik, meningkatkan keberagaman individu. Sebaliknya, seleksi menurunkan keberagaman individu dan berperan sebagai suatu kekuatan yang memaksa individu-individu untuk meningkatkan kualitasnya. Adapun skema umum AE dapat dilihat pada gambar 2.3.



Gambar 2.3. Skema Umum Algoritma Evolusi (AE)

(Sumber: Suyanto, 2014)

1. Inisialisasi

Pembentukan populasi awal dengan cara membangkitkan individu secara acak, tetapi pada kasus tertentu pembangkitan bisa dilakukan dengan fungsi heuristik untuk mendapatkan solusi yang lebih cepat.

2. Populasi

Populasi adalah kumpulan solusi yang mungkin (calon solusi). Suatu populasi biasanya dirancang memiliki individu dengan jumlah yang tetap pada setiap generasi.

3. Orangtua

Orangtua merupakan individu yang terpilih melalui proses seleksi yang selanjutnya akan melalui proses rekombinasi.

4. Anak

Anak (*offspring)* merupakan hasil dari proses rekombinasi orang tua dan akan mewarisi sifat-sifat orangtua mereka.

5. Rekombinasi

Rekombinasi (*crossover)* merupakan proses penggabungan dua orang tua yang akan menghasilkan individu anak.

6. Mutasi

Mutasi melakukan perubahan gen pada kromosom yang dipilih secara acak.. Proses mutasi bisa menghasilkan individu baru yang berkualitas lebih buruk, sama, atau lebih baik dari individu lama.

7. Terminasi

Merupakan kondisi yang dapat menghentikan proses evolusi seperti terdapat individu yang telah memiliki nilai *fitness*  yang diharapakan, evolusi telah mencapai jumlah generasi maksimum yang diizinkan, keberagaman populasi telah mencapai tingkat minimum yang diizinkan dan tidak ada lagi peningkatan nilai *fitness* yang diharapkan.

**2.2.4.2. Pencarian Lokal (PL)**

Menurut Hakim *et al.* (2016), PL merupakan metode heuristik yang bermula dari sebuah solusi awal yang dibentuk secara acak. Kemudian dibentuk sebuah lingkungan (*neighbourhood*) dari solusi awal tersebut dengan tujuan untuk mencari solusi yang lebih baik di dalam lingkungan tersebut. Setelah solusi terbaik yang ada di dalam lingkungan tersebut didapat, maka langkah selanjutnya adalah membuat lingkungan yang baru dari solusi tersebut. Langkah ini kemudian terus diulangi sampai tidak ditemukan lagi solusi yang lebih baik dari suatu lingkungan.

Metode PL yang umum digunakan pada AM adalah algoritma *Hill Climbing* (HC)*.* HCmenerima solusi awal dan kemudian solusinya ditingkatkan secara berulang-ulang hingga mencapai pada suatu kondisi dimana solusi tersebut sudah tidak dapat ditingkatkan atau mencapai maksimal. HC yang cukup sering digunakan adalah *Simple Hill Climbing* (SHC). Menurut Dangkua *et al.* (2017) proses algoritma SHC adalah sebagai berikut:

1. Evaluasi state awal, jika state awal sama dengan tujuan, maka proses berhenti. Jika tidak sama dengan tujuan maka lanjutkan proses dengan membuat state awal sebagai state sekarang.
2. Mengerjakan langkah berikut sampai solusi ditemukan atau sampai tidak ada lagi operator baru yang dapat digunakan dalam *state* sekarang:
3. Mencari sebuah operator yang belum pernah digunakan dalam *state* sekarang dan gunakan operator tersebut untuk membentuk *state* baru.
4. Evaluasi *state* baru*.*
5. Jika *state* baru adalah tujuan*,* maka proses berhenti.
6. Jika *state* baru tersebut bukan tujuan tetapi *state* baru lebih baik daripada *state* sekarang*,* maka buat *state* baru menjadi *state* sekarang.
7. Jika *state* baru ini tidak lebih baik daripada *state* sekarang*,* maka lanjutkan ke langkah 2.

**2.2.5.** **Proses Algoritma Memetika**

Sesuai dengan *Pseudocode* pada gambar 2.2 maka proses utama AM bisa diuraikan menjadi 7 (tujuh) langkah, sebagai berikut:

1. Pengkodean

Pengkodean (*Encoding*) merupakan proses merepresentasikan masalah ke dalam individu. Represesntasi individu memiliki banyak bentuk yang paling umum digunakan dalam pengkodean antara lain *character encoding, binary encoding, discrete decimal encoding* dan bentuk lain tergantung dari jenis permasalahan yang ada. Contoh umum pengkodean dapat dilihat pada gambar 2.4.

1. *Character encoding*

M1

M2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P | M | M | O | S | P | M | M | O | P |

gen 10

gen 6

gen 5

gen 1

2. *Binary encoding*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

gen 10

gen 2

gen 1

3. *Discrete decimal encoding*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 6 | 4 | 3 | 8 | 9 | 0 |

gen 10

gen 2

gen 1

Gambar 2.4. Contoh umum pengkodean

2. Populasi Awal

Proses membangkitkan sejumlah individu secara acak. Jumlah individu yang dibangkitkan secara acak tergantung jumlah populasi yang ditentukan. Misal pada setiap generasi ditentukan populasi berjumlah 2 yang individunya dikodekan menggunakan metode *character encoding*. Contoh pembangkitan individu secara acak dapat dilihat pada gambar 2.5.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| GENERASI | INDIVIDU | Gen 1 | Gen 2 | Gen 3 | Gen 4 | Gen 5 |
| 1 | Individu 1 | P | P | M | M | S |
| M | M | S | M | P |
| S | S | S | M | M |
| Individu 2 | P | P | P | M | M |
| S | M | P | S | S |
| P | P | M | M | S |

Gambar 2.5. Contoh pembangkitan individu secara acak

2. Evaluasi *Fitness*

Melakukan evaluasi *fitness* terhadap setiap individu pada populasi. Nilai fungsi *fitness* bisa berupa maksimasi atau minimasi tergantung dari jenis permasalahan yang ada. Menurut Suyanto (2014), jika solusi yang dicari adalah memaksimalkan sebuah fungsi *h* (masalah maksimasi) maka nilai *fitness* yang digunakan adalah nilai dari fungsi *h* tersebut, yaitu:

*f = h* …………………………………………………………………… (1)

keterangan:

*f* : Nilai *fitness*

*h*: Maksimasi

sedangkan jika yang dicari adalah meminimalkan sebuah fungsi (masalah minimasi) maka fungsi *h*  tidak bisa langsung digunakan dan harus disesuaikan karna berlaku aturan bahwa individu yang lebih baik merupakan individu yang memiliki nilai *fitness* yang lebih baik. Oleh karena itu nilai *fitness* yang biasa digunakan adalah semakin kecil nilai *h* maka semakin besar nilai *f,* yaitu:

.………………………………………………………… (2)

keterangan:

*f* : Nilai *fitness*

*h*: Minimasi

a: Bilangan sangat kecil bervariasi sesuai dengan masalah yang ada

3. Seleksi (*Selection)*

Melakukan pemilihan dua buah individu sebagai induk, yang akan dipindahsilangkan, secara proposional sesuai dengan nilai *fitness-*nya. Metode *roulette-wheel* ini meniru permainan *roulette-wheel*  dimana masing-masing kromosom menempati potongan lingkaran pada roda roulette secara proposional sesuai dengan nilai *fitness*. Untuk mengimplementasikan metode *roulette-wheel selection* diimplementasikan dengan cara membuat interval [0,1] nilai komulatif dari nilai *fitness* masing-masing kromosom dibagi total nilai *fitness* dari semua kromosom. Gambar 2.6 menggambarkan contoh penggunaan metode *roulette-wheel selection.*

|  |  |
| --- | --- |
| Individu | Nilai *fitness* |
| 1 | 0,5 |
| 2 | 0,5 |
| 3 | 2 |
| Total *fitness* | 3 |

**N3**

**N2**

**N1**

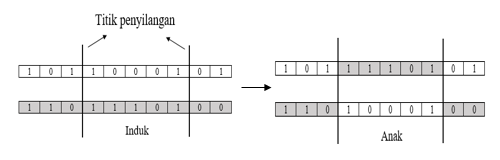
Gambar 2.6. Contoh pengunaan *roulette-wheel selection*

4. Rekombinasi (*Crossover*)

Setelah induk telah dipilih pada proses selanjutnya prosesrekombinasi antara kedua induk untuk menghasilkan anak (*offspring*). Dua anak akan tercipta dari dua induk. Rekombinasi bisa berakibat gen-gen pada individu yang mengarah ke solusi akan menyebar ke kromosom-kromosom lainya mengakibatkan konvergensi *premature.* untuk itu dibuat suatu aturan bahwa rekombinasi hanya bisa dilakukan hanya jika bilangan acak dibangkitkan dalam rentan [0,1] lebih kecil dari probabilitas rekombinasi *Pr*. Seperti pada teori seleksi alam, kemungkinan individu untuk melakukan rekombinasi atau perkawinan lumayan besar maka *Pr* biasanya di atur mendekati 1 misalnya 0,9. Menurut Zukhri (2013), beberapa metode yang biasanya digunakan didalam proses rekombinasi adalah sebagai berikut:

a. Rekombinasi *N-*titik (*N-Point Crossover*)

Nilai *N* dapat ditentukan terlebih dahulu atau ditentukan terlebih dahulu atau ditentukan secara acak. Dalam metode ini setiap kromosom induk dipotong menjadi *N*+1 bagian. Kromosom anak yang pertama dihasilkan dengan mewariskan bagian potongan dalam urutan ganjil dari induk yang pertama dan mewariskan bagian potongan dalam urutan genap dari induk yang kedua. Demikian juga dengan kromosom anak yang kedua dihasilkan dengan cara yang sama dengan menggabungkan bagian yang tersisa dari kedua kromosom induknya. Ilustrasi rekombinasi *N-*titikdapat dilihat pada gambar 2.7.

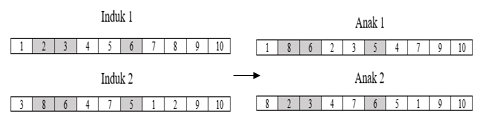
**

Gambar 2.7. Ilustrasi rekombinasi *N-*titik

(Sumber: Zukhri, 2013)

b. Rekombinasi Berbasis Posisi (*Position Based Crossover*)

Dalam metode ini dipilih sejumlah posisi gen secara acak, kemudian gen-gen pada posisi terpilih pada induk yang pertama diwariskan pada kromosom anak yang kedua, sedangkan gen-gen lainya dari kromosom anak yang kedua diambil daripada gen-gen induk yang kedua dengan urutan yang sama. Ilustrasi rekombinasi berbasis posisi dapat dilihat pada gambar 2.8.

**

Gambar 2.8. Ilustrasi rekombinasi berbasis posisi

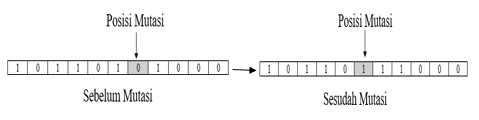
(Sumber: Zukhri, 2013)

5. Mutasi (*Mutation*)

Setelah rekombinasi dilakukan, prosedur mutasi akan diterapkan untuk semua individu pada populasi termasuk individu anak dengan syarat bilangan acak yang dibangkitkan dalam rentan [0,1] kurang dari probabilitas mutasi *Pm.* Seperti pada teori seleksi alam, kemungkinan individu untuk mengalami mutasi sangat kecil sehingga probabilitas mutasi biasanya diatur sangat kecil juga, misalnya 0,1. Menurut Zukhri (2013), beberapa metode yang biasanya digunakan didalam proses mutasi adalah sebagai berikut:

a. Mutasi Kode Biner

Mutasi represensati kromosom dengan kode biner dilakukan dengan mengubah nilai gen dengan nilai kebalikanya pada posisi tertentu yang dipilih secara acak. Untuk lebih jelasnya dapat dilihat pada gambar 2.9 yang menggambarkan ilustrasi mutasi kode biner.

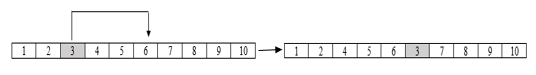


Gambar 2.9. Ilustrasi mutasi kode biner

(Sumber: Zukhri, 2013)

b. Mutasi Berbasis Posisi (*Position Based Mutation*)

Dalam metode ini, proses mutasi dilakukan dengan cara memilih posisi sebuah gen secara acak, kemudian gen tersebut akan dipindahkan pada posisi acak lainya. Ilustrasi mutasi berbasis posisi dapat dilihat pada gambar 2.10.

**

Gambar 2.10. Ilustrasi mutasi berbasis posisi

(Sumber: Zukhri, 2013)

6. Pencarian lokal (*Local Search*)

Pencarian Lokal (PL) dilakukan setelah proses rekombinasi dan mutasi, individu *improvement* yaitu peningkatan *fitness* individu melalui proses pencarian lokal akan dilakukan pada setiap individu dalam populasi termasuk anakdan orangtua untuk meningkatkan nilai *fitness-*nya menjadi lebih baik dan layak. Jika suatu individu berhasil ditingkatkan menjadi lebih baik maka individu tersebut akan menggantikan (*replacement*) individu yang sebelumnya jika tidak maka individu sebelumnya yang akan tetap digunakan. Setelah proses pencarian lokal dilakukan individu terbaik dari populasi sekarang akan disimpan kedalam elit individu dan disalin kedalam populasi untuk generasi selanjutnya. Elit individu bertujuan agar individu terbaik tidak akan rusak karena proses rekombinasi, mutasi dan tetap bisa digunakan pada generasi selanjutnya.

**2.2.6.** **PYTHON**

Berdasarkan *official web python* dalam Aisa *et al.* (2016), Python adalah bahasa pemrograman interpretatif multiguna dengan perancangan berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan siktaksis kode yang sangat jelas, dilengkapi fungsionalitas pustaka standar yang besar serta komprehensif. Beberapa kelebihan dari bahasa pemrograman python adalah sebagai berikut:

1. Memiliki kepustakaan yang luas. Dalam distribusi Python disediakan modul-modul yang dapat dipakai untuk berbagai keperluan.
2. Memiliki tata bahasa yang jernih dan mudah dipelajari.
3. Memiliki aturan layout kode sumber yang memudahkan pengecekan, pembacaan kembali, dan penulisan kode sumber.
4. Merupakan bahasa yang telah berorientasi objek.
5. Memiliki sistem pengolahan memori otomatis.

**2.2.7.** **FLASK**

Flask adalah *web framework* untuk bahasa pemrograman python yang menyediakan *tools, libraries,* dan teknologi yang memungkinkan untuk membuat aplikasi *web*. Flask pertama kali dikembangkan oleh Armin Ronacher di kelompokinternasional pencinta python yang disebut Pocoo tim. Flask adalah bagian dari kategori *micro-framework. Micro-framework* merupakan *framework* seperti pada umumnya dengan sedikit atau tanpa *dependencies* (ketergantungan) ke *external libraries.* Flask memiliki 2 *dependencies* utama yaitu Werkzeug sebagai WSGI *utility library* dan jinja2 sebagai *template engine*.

**2.2.8.** **SQLite**

Menurut Setiyadi *et al.* (2015), SQLite adalah suatu *library* yang menerapkan mesin *database-self-contained, server-less, zero-configuration, dan transactional. Self-contained* berarti SQLite membutuhkan sedikit sekali dukungan dari *library eksternal* atau dari sistem operasi. *Server-less* berati SQLite dalam mengakses database baik itu read atau write dapat secara langsung dari file database tanpa melalui proses server dan tidak mendukung pengaksesan secara remote (artinya database SQLite bisa dikendalikan dari jarak jauh dengan adanya jaringan komputer (“Computer Network”), baik melalui jaringan lokal (intranet) atau internet), dimana kebanyakan mesin SQL database diterapkan sebagai proses server yang terpisah. *Zero–configuration* menunjukkan SQLite tidak membutuhkan instalasi sebelum penggunaannya.