IMPLEMENTASI *TREE STRUCTURE MODELING* DAN ALGORITMA GENETIKA UNTUK MENYELESAIKAN *UNEQUAL-AREA FACILITY LAYOUT PROBLEM*

(Studi Kasus Desain Tata Letak Ruang dalam Rumah)

SKRIPSI

oleh:

SITI MAINUR FITRIANA
0610960058

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2012

UNIVERSITAS BRAWIJAYA

IMPLEMENTASI *TREE STRUCTURE MODELING* DAN ALGORITMA GENETIKA UNTUK MENYELESAIKAN *UNEQUAL-AREA FACILITY LAYOUT PROBLEM*

(Studi Kasus Desain Tata Letak Ruang dalam Rumah)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

oleh:

SITI MAINUR FITRIANA
0610960058-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2012


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

Implementasi *Tree Structure Modeling* dan Algoritma Genetika untuk Menyelesaikan *Unequal-Area Facility Layout Problem* (Studi Kasus Desain Tata Letak Ruang dalam Rumah)

Oleh :

SITI MAINUR FITRIANA
0610960058-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 25 Januari 2012

Dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Dewi Yanti L.,S.Kom,M.Kom	Drs. Achmad Ridok, M. Kom
NIP. 198111162005012004	NIP. 196808251994031002

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP.196908071994121001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	:	Siti Mainur Fitriana
NIM	:	0610960058-96
Jurusan	:	Matematika
Program Studi	:	Ilmu Komputer
Penulis skripsi berjudul	:	Implementasi *Tree Structure Modeling* dan Algoritma Genetika untuk Menyelesaikan *Unequal-Area Facility Layout Problem* (Studi Kasus Desain Tata Letak Ruang dalam Rumah)

Dengan ini menyatakan bahwa :

1. Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.
2. Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 25 Januari 2012
Yang menyatakan,

Siti Mainur Fitriana
NIM. 0610960058-96


UNIVERSITAS BRAWIJAYA

IMPLEMENTASI *TREE STRUCTURE MODELING* DAN ALGORITMA GENETIKA UNTUK MENYELESAIKAN *UNEQUAL-AREA FACILITY LAYOUT PROBLEM*
(Studi Kasus Desain Tata Letak Ruang dalam Rumah)

ABSTRAK

Dalam dunia arsitektur, desain tata letak ruang dalam rumah, yang biasa disebut dengan denah/*floorplan* merupakan dasar atau landasan saat merencanakan membangun rumah. Pada hakekatnya, merancang desain tata letak ruang dalam rumah harus sesuai dengan keinginan pemilik bangunan rumah. Banyak faktor yang sebaiknya diperhatikan dalam menentukan desain tersebut seperti kebutuhan ruang dan fungsi ruang.

Dengan memperhatikan dua faktor tersebut, permasalahan desain tata letak ruang dalam rumah bisa digolongkan ke dalam *unequal-area facility layout problem*, yaitu menentukan tata letak fasilitas yang paling efisien dari sekumpulan ruangan dalam suatu fasilitas tertentu dengan luas tak sama. Pada penelitian ini, *tree structure modeling* diusulkan untuk merepresentasikan solusi dari *unequal-area facility layout problem* tersebut. Dimana diasumsikan luas fasilitas yang tersedia mencukupi. Parameter yang digunakan, antara lain ketersediaan lahan (ukuran fasilitas), kebutuhan ruangan (jumlah dan ukuran ruangan), dan nilai kedekatan antar ruang tersebut. Sedangkan untuk menunjukkan pilihan desain tata letak ruang dalam rumah yang baik dengan sedikit karakteristik yang diketahui digunakan algoritma genetika untuk mendapatkan pilihan desain tata letak ruang dalam rumah yang mendekati optimal menurut sistem.

Hasil dari penelitian ini menunjukkan bahwa *tree structure modeling* dan algoritma genetika kurang tepat digunakan untuk kasus desain tata letak ruang dalam rumah. Berdasarkan uji coba yang telah dilakukan nilai *fitness* terbaik diperoleh dengan menggunakan ukuran populasi 35, maksimal generasi 500, probabilitas crossover 80 dan probabilitas mutasi 0.5.

Keyword : desain tata letak ruang dalam rumah, *unequal-area facility layout problem*, *tree structure modeling*, algoritma genetika

THE IMPLEMENTATION OF TREE STRUCTURE MODELLING AND GENETIC ALGORITHM TO SOLVE UNEQUAL-AREA FACILITY LAYOUT PROBLEM (Case Study of Rooms Layout Design in the House)

ABSTRACT

In architecture, rooms layout design in the house that is usually called as a site plan or a floor plan is the basic concept in building house plan. Normally, designing the layout in the house must be matched with the owner of the house desire. There are a lot of factors that should be recommended in deciding the design, like the room's need and room's function.

Concerning the two factors above, the problems faced in designing house layout can be classified into unequal-area facility layout problem, namely deciding facility layout that is most efficient among a group of rooms in certain facility with different range. In this study, tree structure modeling is suggested to represent the solution from that unequal-area facility layout problem, where the range of facility is assumed sufficiently. The parameter that is used, such as, land availability (facility measurements), room's needs (the sum and the size of the rooms), and approximation values between rooms. While, to show the good rooms layout design in the house preference with less known characteristics, genetic algorithm is used, to get the option of rooms layout design in the house which is closely optimal based on system.

The result of the study showed that tree structure modeling and genetic algorithm is not suitable to be used for the case of rooms layout design in the house. Based on the experiments have been done, the best fitness value is received by using population score 35, generation maximal 500, crossover probability 80 and mutation probability 0.5.

Keywords : rooms layout design in the house, unequal-area facility layout problem, tree structure modeling, genetic algorithm.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kepada Allah SWT atas segala limpahan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "Implementasi *Tree Structure Modeling* dan Algoritma Genetika untuk Menyelesaikan *Unequal-area Facility Layout Problem* dengan Studi Kasus Desain Tata Letak Ruang dalam Rumah" sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer.

Atas terselesainya skripsi ini, penulis mengucapkan terima kasih kepada :

1. Dewi Yanti Liliana, S.Kom, M.Kom, selaku pembimbing utama dalam penulisan skripsi.
2. Drs. Achmad Ridok, M.Kom, selaku pembimbing pendamping dalam penulisan skripsi.
3. Dr. Abdul Rouf Alghofari, M.Sc, selaku Ketua Jurusan Matematika FMIPA Universitas Brawijaya.
4. Drs. Marji, MT, selaku Ketua Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
5. Dany Primanita K., ST, selaku dosen pembimbing akademik.
6. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
7. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya.
8. Orang tua penulis, yang senantiasa mendoakan dan mendukung penulis agar segera menyelesaikan skripsi.
9. Saudara-saudari penulis, Kak Ida, Kak Ina, Kak Roni, Dina, dan Risa, yang tak pernah bosan memberikan semangat dan motivasi kepada penulis.
10. Kakak-kakak ipar penulis, Kak Azis, Kak Rena, dan Mas Dadang, atas dukungan, perhatian, dan motivasi yang diberikan.

11. Ranti, Nina, Iiz, Weni, serta semua rekan-rekan Takmir Masjid An-Nur SMA Negeri 1 Boyolangu Tulungagung, yang memberikan kenangan indah bagi penulis semasa SMA.
12. Mbak Indah, Dian, Putri, Mas Adib, Mas Data, Dik Pipit, Yuli, Lupi, Falen, Dwina, Granit, Nata, Yanuar, Riris, Yoga, Dinda, Ephy, Devi, Nita, Asri, serta semua rekan-rekan ilkomers FMIPA Universitas Brawijaya, yang telah memberikan banyak bantuan demi kelancaran pelaksanaan penyusunan skripsi ini.
13. Puput, Viroh, Dik Endah, Riska, Dik Novita, serta semua rekan-rekan di FMIPA Universitas Brawijaya.
14. Rekan-rekan kontrakan Baitul Muflihun SS55B, khususnya yang masih setia menemani penulis hingga menyelesaikan skripsinya.
15. Dan semua pihak yang telah membantu dalam penyusunan skripsi ini yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa dalam penyusunan skripsi ini banyak kekurangan, untuk itu saran dan kritik yang membangun demi kesempurnaan penulisan selanjutnya sangat penulis harapkan. Semoga skripsi ini dapat bermanfaat.

Malang, 25 Januari 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCECODE*

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Setiap keluarga pasti menginginkan sebuah rumah yang ideal untuk ditempati bersama keluarga. Rumah bukan hanya tempat untuk berteduh dari hujan dan terik matahari tetapi juga madrasah utama dan pertama untuk mencetak generasi bangsa sekaligus tempat berkumpul keluarga dan mempererat tali silaturrahmi. Oleh karena itu, rumah dijadikan sebagai kebutuhan primer manusia.

Sedikit pengetahuan tentang membangun rumah yang ideal, menjadi masalah tersendiri bagi seseorang yang akan membangun rumah. Untuk itu, dibutuhkan jasa arsitek yang biayanya relatif mahal dalam merancang desain rumah. Hal ini disebabkan rumah yang ideal akan bisa diwujudkan dari desain rumah yang tepat. Bagi orang yang awam dalam mendesain rumah dan keterbatasan dana untuk membayar jasa arsitek juga merupakan beban tersendiri (Liliana, 2006).

Saat ini perkembangan dunia komputer semakin pesat semakin memudahkan pekerjaan manusia dalam menyelesaikan permasalahan yang dihadapi manusia. Termasuk permasalahan di atas, yaitu merancang desain rumah. Dalam istilah arsitektur dan insinyur bangunan, desain rumah biasa disebut dengan *floorplan*. *Floorplan* adalah gambar yang berbentuk skala, gambar-gambar ruang (ruang tamu, kamar tidur, kamar mandi, taman, dll) yang menunjukkan hubungan antar ruang. Sebagian orang menyebut *floorplan* sebagai denah. Denah sendiri merupakan suatu dasar atau landasan saat merencanakan membangun sebuah rumah yang ideal bagi penghuni rumah (Amin dkk, 2010).

Permasalahan merancang desain rumah tersebut bisa digolongkan ke dalam *facility layout problem* (FLP). Menentukan tata letak yang paling efisien dari departemen-departemen dalam sebuah fasilitas disebut dengan *facility layout problem* (FLP) (Garey dan Johnson, 1973). Dalam permasalahan ini, departemen yang dimaksud adalah ruang dalam rumah dimana rumah sebagai fasilitas. FLP terbagi ke dalam dua kategori berdasarkan bentuk daerah fasilitas: *equal-area facility layout problem* dan *unequal-area facility layout problem*. *Equal-area facility layout problem* biasanya dapat

dimodelkan sebagai *quadratic assignment problem*. Namun dalam praktiknya permasalahan desain tata letak departemen hampir merupakan *unequal-area facility layout problem* (Honiden, 2004). Untuk menghasilkan desain tata letak ruang dalam rumah yang optimal, yang merupakan *unequal-area facility layout problem*, diperlukan metode yang tepat.

Dalam menyelesaikan *unequal-area facility layout problem*, banyak peneliti yang sudah melakukannya, antara lain Amour dan Buffa mengusulkan metode pertukaran kedekatan departemen pada tahun 1963. Bazaraa memperkenalkan pendekatan *branch and bound* pada tahun 1975 sedangkan van Camp dkk mengusulkan metode *nonlinier programming model and heuristic* pada tahun 1991. Tam mewakilkan tata letak sel pada lantai toko yang spesifik sebagai *slicing tree* dan mengembangkannya dengan *simulated annealing* pada tahun 1992. Saat ini, peneliti menggunakan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* telah meningkat (Honiden, 2004).

Beberapa peneliti tersebut antara lain Rojas dan Torres (2004) menggunakan algoritma genetika untuk mendapatkan desain tata letak yang optimal pada ladang peternakan susu kambing, Peres dkk (2004) menggunakan algoritma genetika untuk mendesain tata letak kantor-kantor bank, Honiden (2004) menggabungkan *tree structure modeling* dan algoritma genetika.

Algoritma genetika adalah teknik pencarian probabilistik dan optimasi yang mengoperasikan populasi dari kromosom-kromosom. Tiap-tiap populasi merepresentasikan kemungkinan solusi dari permasalahan yang diberikan, kemudian memelihara beberapa solusi berkualitas tinggi. Keuntungan dari algoritma genetika adalah pengoperasian yang sederhana namun dapat menunjukkan pilihan yang baik untuk menyelesaikan permasalahan dengan ruang pencarian yang luas, dimana hanya sedikit karakteristiknya yang diketahui. Saat ini algoritma genetika sudah diaplikasikan ke dalam banyak permasalahan antara lain bidang optimasi, perencanaan dan penjadwalan, serta desain dan lain-lain (Kubalik, dkk., 2002). Algoritma genetika juga dapat digunakan untuk memberikan alternatif tata letak ruang dalam sebuah lahan atau rumah (Liliana, 2006).

Terushige Honiden pada bulan Desember 2004 telah membuktikan bahwa *tree structure modeling* dan algoritma genetika dapat menyelesaikan *unequal-area facility layout problem* dengan menggunakan dua macam permasalahan tata letak dan tiga macam *crossover* (PMX, OX, CX) di dalam algoritma genetikanya, dimana *crossover* dengan PMX memiliki hasil yang lebih baik dibanding *crossover* lainnya. Karena itu, skripsi ini menggunakan *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* namun dengan penyesuaian studi kasus desain tata letak ruang dalam rumah. Dengan demikian skripsi ini berjudul "Implementasi *Tree Structure Modeling* dan Algoritma Genetika untuk Menyelesaikan *Unequal-area Facility Layout Problem*" dengan studi kasus desain tata letak ruang dalam rumah.

1.2 Rumusan Masalah

Berdasarkan penjelasan latar belakang sebelumnya, maka rumusan masalah yang digunakan dalam penelitian skripsi ini adalah:

- Bagaimana menentukan model genetika, mulai dari representasi kromosom, perhitungan *fitness*, seleksi, *crossover*, serta mutasi yang dapat digunakan untuk menyelesaikan permasalahan *unequal-area facility layout problem* dengan studi kasus desain tata letak ruang dalam rumah?
- Bagaimana pengimplementasian *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* agar menghasilkan desain tata letak ruang dalam rumah yang mendekati optimal?

1.3 Batasan Masalah

Adapun batasan masalah yang digunakan dalam penelitian skripsi ini adalah sebagai berikut:

1. Panjang dan lebar lahan berdasarkan masukan *user*, dengan syarat luas lahan yang dihasilkan berkisar 60-200 m^2.
2. Panjang dan lebar ruang berdasarkan masukan *user*, dengan syarat sesuai dengan spesifikasi ruangan yang telah ditentukan.
3. Diasumsikan luas lahan mencukupi untuk meletakan semua ruang.

4. Desain yang dihasilkan berupa gambar dua dimensi (2D), lahan dan ruang berbentuk empat persegi panjang yang beraturan,
5. Tidak memperhatikan adanya *accessway* (jalan masuk suatu ruang ke ruang lain), koridor, penempatan posisi pintu dan jendela, penempatan letak ruang depan, kanan, kiri, dan belakang.
6. Desain yang dihasilkan berdasarkan nilai kedekatan dan jarak antar ruang.
7. Tata letak ruang pada lahan berlantai satu.
8. Tidak ada suatu ruangan yang masuk ke dalam bagian ruangan lain.

1.4 Tujuan Penelitian

Tujuan penelitian dari skripsi ini adalah:

1. Menentukan model genetika, mulai dari representasi kromosom, perhitungan *fitness*, seleksi, *crossover*, serta mutasi yang dapat digunakan untuk menyelesaikan permasalahan desain tata letak ruang dalam rumah.
2. Membandingkan nilai *fitness* yang dihasilkan dari kombinasi ukuran populasi, maksimal generasi, probabilitas *crossover* dan probabilitas mutasi yang berbeda dalam menghasilkan desain tata letak ruang dalam rumah yang mendekati optimal.

1.5 Manfaat Penelitian

Manfaat penelitian dari skripsi ini adalah:

1. Tersedia sistem yang dapat membantu menyelesaikan *unequal-area facility layout problem* dengan studi kasus desain tata letak ruang dalam rumah.
2. Tersedia referensi terkait implementasi *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* dalam hal mendesain tata letak ruang dalam rumah bagi peneliti lain.

1.6 Metodologi Penelitian

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah:

1. Studi Literatur
   Mempelajari teori-teori yang berhubungan dengan desain tata letak ruang dalam rumah, *unequal-area facility layout problem*, *tree structure modeling*, algoritma genetika secara umum, serta pendekatan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* dari berbagai referensi.
2. Wawancara Langsung
   Wawancara yang dilakukan dengan cara mengkonsultasikan data-data yang diperoleh dari studi literatur kepada arsitek dan mahasiswa yang kuliah di jurusan arsitektur.
3. Pendefinisian dan Analisis Masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi optimal.
4. Perancangan dan Implementasi Program
   Membuat perancangan perangkat lunak dan mengimplementasikan hasil rancangan tersebut ke dalam sebuah program komputer.
5. Uji Coba dan Analisa Hasil Implementasi
   Menguji perangkat lunak dengan data yang dimiliki dan menganalisa hasil implementasi tersebut kemudian dievaluasi.

1.7 Sistematika Penulisan

Pembuatan skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Berisi penjelasan singkat tentang teori-teori yang berhubungan dengan desain tata letak ruang dalam rumah, *unequal-area facility layout problem*, *tree structure*

modeling, algoritma genetika secara umum, pendekatan algoritma genetika dalam menyelesaikan *unequal-area facility layout problem* serta hal-hal lain yang terkait.

3. BAB III METODOLOGI DAN PERANCANGAN
   Bab ini berisi metode-metode yang digunakan dalam menyelesaikan *unequal-area facility layout problem* dengan menggunakan *tree structure modeling* dan algoritma genetika dengan studi kasus desain tata letak dalam rumah.
4. BAB IV IMPLEMENTASI DAN PEMBAHASAN
   Bab ini berisi penjelasan tentang implementasi sistem dan hasil pengujian yang dilakukan.
5. BAB V KESIMPULAN DAN SARAN
   Bab ini berisi kesimpulan yang diperoleh dari hasil dan pembahasan serta saran-saran yang bisa dilakukan untuk pengembangan penelitian lebih lanjut.

BAB II
TINJAUAN PUSTAKA

2.1 Rumah

Dalam arti umum, rumah adalah bangunan buatan manusia yang dijadikan tempat tinggal selama periode waktu tertentu (Odop, 2010).

2.1.1 Ciri-Ciri Rumah yang Ideal

Rumah yang ideal biasanya memiliki beberapa hal seperti di bawah ini (Odop, 2010):

1. Fungsional
   Rumah harus memenuhi unsur utama, yakni berfungsi sebagai tempat tinggal manusia. Rumah harus mampu menganyomi penghuninya dari panas, hujan, dan pengaruh luar yang bisa mengancam keselamatan serta kenyamanan penghuni.
2. Indah
   Rumah yang indah akan membuat penghuni rumah merasa betah tinggal di dalamnya. Rumah indah tidak berarti harus mahal dan berbiaya tinggi. Rumah indah bisa dalam bentuk fisik dan nonfisik. Rumah indah secara fisik, yaitu tampilan bangunan itu sendiri, baik dari segi desainnya, penataan ruangan, besarnya bangunan, maupun pemilihan bahan dan teknologinya. Sedangkan, rumah indah secara nonfisik bisa karena keindahan dan kebahagiaan penghuninya. Jika kedua unsur ini bisa dipenuhi dalam sebuah rumah maka rumah itu benar-benar indah.
3. Kuat
   Sebagai tempat tinggal yang setiap saat didiami, rumah secara struktural harus kuat dan memberi keamanan bagi penghuninya. Rumah harus mampu berdiri kokoh sampai waktu yang tidak ditentukan. Konstruksinya harus mampu menompang seluruh elemen bangunan, kuat terhadap beban bangunan itu sendiri ataupun beban yang ditimpakan kepadanya. Bangunan harus kokoh dalam mendukung seluruh kegiatan penghuninya, dan sangat baik jika struktur

bangunannya mampu mendukung beban jika suatu saat ditumbuhkan atau dikembangkan lagi.

4. Nyaman
   Rumah yang nyaman tidak sekedar indah dan megah namun lebih kepada perasaan penghuninya ketika berada di dalamnya. Rumah yang nyaman akan membuat penghuninya betah di dalamnya.
5. Ekonomis
   Rumah juga harus ekonomis. Artinya, rumah harus memiliki ruang yang proporsional dengan bahan bangunan yang tahan iklim dalam waktu lama. Singkat kata, rumah yang bernilai ekonomis harus sesuai dengan kemampuan seseorang yang ingin membangun rumah. Hal ini tentu saja bisa disiasati dengan beberapa tips, semisal ukuran rumah yang disesuaikan dengan kebutuhan, pemilihan bahan bangunan dan model yang sederhana.

2.1.2 Pengenalan Bagian-Bagian Rumah

Rumah merupakan satu kesatuan dari beberapa unsur ruang yang membentuknya. Tanpa kelengkapan unsur ruang tersebut, maka rumah tidak akan menjadi lengkap bahkan tidak lagi dapat disebut rumah. Unsur-unsur pendukung dari rumah antara lain sebagai berikut (Kristiawan, 2010 dan Odop, 2010):

1. Teras depan
   Teras depan adalah ruang yang terletak di muka sebuah rumah tinggal. Ruang ini merupakan ruang umum yang menghubungkan antara tamu dan pemilik rumah. Ukuran yang biasanya dibutuhkan untuk teras adalah seluas 1,5 m x 1,5 m atau 2 x 3 m.
2. Ruang tamu
   Ruang tamu adalah ruang yang digunakan untuk menerima dan melayani tamu yang datang. Ruang ini biasanya terletak dekat dengan teras depan dan pintu depan, serta jauh dari kamar tidur, sehingga ketenangan dan privasi kamar-kamar tidur terjamin. Ruang ini biasanya didekatkan dengan dapur dan ruang makan untuk memudahkan pelayanan bagi para tamu yang datang. Ukurannya bisa 3 x 3 m, 3 x 4 m, 4 x 5 m atau lebih sesuai kebutuhan dan kemampuan pemilik rumah.

3. Dapur
   Dapur adalah ruang yang digunakan untuk mengolah dan menyiapkan makanan serta minuman bagi keluarga. Ruang dapur biasanya diletakkan dekat dengan ruang tamu dan ruang makan sehingga pelayanan bagi keluarga ataupun tamu menjadi lebih mudah. Ukuran untuk dapur standar 2 x 3 m namun bisa juga 3 x 3 m, 3 x 4 m atau bahkan lebih tergantung kebutuhan.
4. Ruang keluarga
   Ruang keluarga adalah ruang khusus untuk anggota keluarga. Biasanya, ruang keluarga ini digunakan oleh seluruh anggota keluarga untuk kegiatan-kegiatan yang sifatnya rekreasi/dan santai. Ruang keluarga kadang-kadang didesain lebih menarik dan mencakup kesenangan serta hobi dari seluruh anggota keluarga, maka ruang ini biasanya diletakkan dekat dengan ruang tamu, ruang makan, dan dapur. Ukuran standar ruang keluarga 4 x 4 m dan 4 x 5 m.
5. Ruang makan
   Ruang makan adalah ruang yang digunakan secara khusus untuk makan dan minum bagi anggota keluarga. Ruang makan biasanya menjadi satu dengan ruang keluarga. Ruang makan juga dapat diletakkan secara terpisah dengan ruang keluarga. Namun, ruang makan seyogianya diletakkan di dekat dapur atau bersebelahan dengan dapur dan dekat dengan ruang tamu sehingga pelayanan bagi keluarga ataupun tamu menjadi lebih praktis dan mudah. Ukuran standar ruang makan bisa 2 x 3 m.
6. Kamar tidur
   Kamar tidur adalah kamar yang digunakan untuk beristirahat/tidur yang sifatnya sangat pribadi. Oleh karena itu, kamar tidur harus diletakkan jauh dari keramaian dan kebisingan. Untuk menciptakan ketenangan dan kenyamanan waktu istirahat/tidur, maka letak kamar tidur sebaiknya dekat dengan kamar mandi dan jauh dari ruang tamu serta garasi. Ukuran yang biasa dibutuhkan kamar tidur adalah 3 x 3m, 3 x 4m bahkan lebih besar.

7. Kamar mandi
   Kamar mandi adalah kamar yang digunakan untuk mandi. Kamar mandi ini biasanya menjadi satu dengan tempat untuk buang air kecil/besar (*closet*). Fasilitas lain dalam kamar mandi adalah bak air, tempat handuk, tempat sabun, dsb. Kamar mandi biasanya diletakkan di dalam kamar tidur utama, di samping kamar tidur anak dan di dekat pintu belakang. Ruangan ini biasanya berukuran 1,5 x 1,5m, 1,75 x 2m dan 2 x 2m.
8. Gudang
   Gudang adalah ruang atau kamar yang disediakan secara khusus untuk menyimpan perabot rumah tangga. Gudang untuk rumah tangga sebaiknya diletakkan jauh dari ruang tamu, ruang keluarga, atau ruang makan, supaya tidak menimbulkan kesan kotor atau berantakkan.
9. Garasi (*Carport*)
   Garasi (*carport*) adalah ruang khusus untuk menyimpan alat-alat transportasi (misalnya: mobil, sepeda motor, dsb). Garasi biasanya diletakkan dekat dengan ruang tamu dan ruang makan atau ruang keluarga. Dari ruang tamu atau ruang makan menuju garasi sebaiknya disediakan sebuah pintu penghubung. Ukuran garasi untuk 1 mobil 3 x 5m.
10. Ruang/kamar tambahan
    Sebagian besar rumah-rumah besar sering menyediakan ruang-ruang atau kamar-kamar tambahan lain, misalnya ruang santai, ruang baca/belajar, ruang sembahyang, taman dalam, ruang bermain anak, dan sebagainya. Ruangan-ruangan tambahan ini biasanya disediakan menurut kesenangan atau hobi dari penghuni rumah tinggal tersebut.

Itulah keseluruhan dari unsur ruang yang membentuk suatu rumah tinggal. Unsur ruang tersebut tentu saling berhubungan dan tidak dapat dipisahkan.

Secara umum kebutuhan ruangan bagi sebuah keluarga sederhana dan yang baru memulai berumah tangga, biasanya hanya ruang-ruang utama sesuai fungsi pokoknya, semisal ruang tamu, kamar tidur 1-3 buah, ruang keluarga yang disatukan dengan ruang makan, teras, kamar mandi serta WC, dan mungkin sedikit taman.

Kebutuhan ruang dan tampilan bangunan disesuaikan dengan kebutuhan seseorang yang ingin membangun rumah dan dana yang tersedia (Odop, 2010).

2.1.3 Denah dan Prinsip Dasar Merancang Denah

Kata denah dalam Kamus Besar Bahasa Indonesia diartikan sebagai gambar rancangan bangunan. Artinya, denah merupakan suatu dasar atau landasan saat merencanakan membangun sebuah hunian tempat tinggal yang nyaman bagi keluarga (Amin dkk, 2010). Denah adalah gambar lokasi dan gambar ruang-ruang lengkap dengan ukuran-ukurannya yang akan dibuat untuk membentuk suatu bangunan rumah. Denah tersebut merupakan gambar bangunan rumah tinggal yang terpotong ± 1 m dari luas permukaan tanah (Kristiawan, 2010).

Dari denah inilah dapat dibaca dan dibayangkan model, bentuk, atau wujud bangunan rumah setelah terbangun nantinya. Dengan denah akan tampak letak dari ruang publik untuk menerima tamu, ruang semiprivat untuk bercengkeraman bersama seluruh keluarga, letak ruang privat yang akan digunakan untuk aktivitas pribadi, hingga letak ruang servis yang sejatinya diadakan untuk menunjang kebutuhan pokok sebagai penghuni rumah. Kesemua hal tersebut dapat ditentukan dan direncanakan melalui sebuah rencana yang bernama denah (Amin dkk, 2010).

Pada hakekatnya, merancang denah sebuah rumah harus sesuai keinginan penghuni atau pemilik bangunan rumah tinggal tersebut. Untuk itu, beberapa langkah berikut ini sebaiknya diikuti agar dalam perencanaannya tidak melenceng jauh dari apa yang dibayangkan (Amin dkk, 2010):

1. Jumlah penghuni rumah
   Menghitung terlebih dahulu anggota keluarga yang akan menempati rumah tersebut. Jumlah penghuni rumah ini akan menjadi dasar utama dalam mengalokasikan kebutuhan ruang.
2. Kebutuhan ruang penghuni rumah
   Setelah diketahui pengguna yang akan menempati rumah tersebut, mulai direka-reka kebutuhan luas ruang yang sesuai dengan kebutuhan penghuni ruangan tersebut.

3. Fungsi ruang
   Setelah kebutuhan ruang terdata, harus dipastikan kejelasan fungsi dan kegunaan ruang tersebut. Dengan demikian, dapat dipertimbangkan adanya kemungkinan menyatukan ruang-ruang yang fungsinya sama. Hal ini juga berguna sebagai bahan pertimbangan saat membuat alur sirkulasi ruang pada denah rumah.
4. Kenyamanan
   Setelah mengetahui fungsi dari masing-masing ruang, langkah selanjutnya adalah memikirkan kenyamanan penggunaan ruang tersebut. Hal ini dapat dilakukan antara lain dengan mengatur penempatan pintu dan jendela, serta tata letak perabotan di dalamnya.
5. Keamanan
   Keamanan merupakan hal utama yang harus dipertimbangkan setelah kenyamanan. Keamanan memiliki pengertian dan lingkup yang luas, di antaranya kekuatan struktur rumah, gangguan-gangguan yang tidak diinginkan, dan sebagainya.
6. Nilai estetika
   Nilai estetika merupakan unsur tambahan yang sebaiknya dialokasikan ke dalam ruang-ruang yang ada di rumah. Dengan demikian rumah tersebut akan nyaman dipergunakan, aman dari gangguan apapun, serta indah dan sedap dipandang.

Setelah semua hal mendasar di atas terakomodasi, akan diperoleh hasil yang maksimal dari sebuah perancangan denah rumah. Wujud bangunan pun akan sesuai dengan yang diharapkan sehingga di kemudian hari tidak terjadi penyesalan akibat salah merencanakan denah.

2.2 *Unequal-area Facility Layout Problem*

Facility Layout Problem (FLP) berkaitan dengan melokasikan sekumpulan departemen sesuai dengan besar luas lahan yang diberikan. Pemberian nilai pada sebuah departemen berdasarkan banyaknya keterkaitan, yang dapat direpresentasikan oleh matriks kedekatan. Banyaknya keterkaitan antar departemen dapat digunakan

untuk memutuskan sesuai keinginan dalam melokasikan departemen satu dengan departemen lainnya. Pemberian nilai tersebut sebanyak departemen yang diminta. Permasalahan untuk menemukan tata letak yang optimal dapat menggunakan fungsi yang berdasarkan penilaian kedekatan dan jarak masing-masing departemen. Ada 2 bagian permasalahan yang terlibat dalam menyelesaikan FLP: i) mendapatkan matriks kedekatan yang optimal untuk menentukan batasan-batasan dan ii) mengambil tata letak dari matriks kedekatan (Bhowmik, 2008).

Minimal parameter yang diperlukan untuk memecahkan masalah FLP adalah: jumlah departemen, ukuran kantor, ukuran departemen yang akan ditempatkan, aliran matriks antara departemen dari cabang dan unit *cost* transportasi matriks. Menurut Islier pada tahun 1998, biaya mutlak tidak diperlukan dalam rangka untuk membandingkan alternatif tata letak, sehingga semua unit biaya diasumsikan sama (bahkan semua sama dalam kesatuan) dalam praktek. Aliran matriks bisa diubah untuk persepsi matriks dimana kedekatan diinginkan antar departemen dapat diwakili (Rojas dan Torres, 2004).

FLP terbagi ke dalam dua kategori berdasarkan bentuk daerah fasilitas: *equal-area facility layout problem* (EA-FLP) dan *unequal-area facility layout problem* (UA-FLP). *Equal-area facility layout problem* biasanya dapat dimodelkan sebagai *quadratic assignment problem*. Namun dalam praktiknya permasalahan desain tata letak departemen hampir merupakan *unequal-area facility layout problem* (Honiden, 2004).

Permasalahan tata letak fasilitas dengan luas tak sama (UA-FLP) pertama kali diformulasikan oleh Armour dan Buffa pada tahun 1963. Armour dan Buffa mengasumsikan ada sebuah fasilitas berbentuk persegi panjang dengan dimensi yang tetap, panjang H dan lebar W. Kemudian ada sejumlah departemen harus disusun dan diletakkan ke dalam fasilitas tersebut. Banyaknya departemen, luas daerah masing-masing departemen, dan nilai aliran yang berhubungan dengan masing-masing sepasang departemen diasumsikan dengan perumusan sebagai berikut:

n = banyaknya departemen
f_{ij} = total aliran antara departemen i dan j, dimana $i,j = 1,2,\ldots,n$
dij = jarak antara departemen i dan j, dimana $i,j = 1,2,\ldots,n$

Fungsi objektif untuk meminimalkan total aliran produk yang melewati fasilitas dirumuskan sebagai berikut:

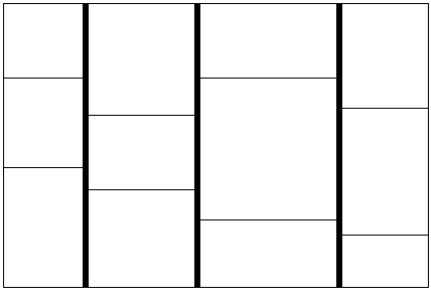
$$Total\ Cost = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} f_{ij}\, d_{ij} \qquad (2.1)$$

Tujuannya adalah untuk membagi daerah ke dalam sub daerah departemen sehingga meminimalkan total perpindahan material dan biaya terkait seluruh fasilitas. Kombinatorial secara alami dari perumusan ini dibuat dengan mempertimbangkan semua kemungkinan tata ruang yang hampir tidak mungkin (Shebanie, 2004).

Untuk menyederhanakan permasalahan ini, para peneliti mengajukan beberapa model. Model yang sering digunakan dalam algoritma metaheuristik adalah model struktur kolom fleksible (*Flexible Bay Structure*) dan struktur pohon pemotong (*Slicing Tree Structure*).

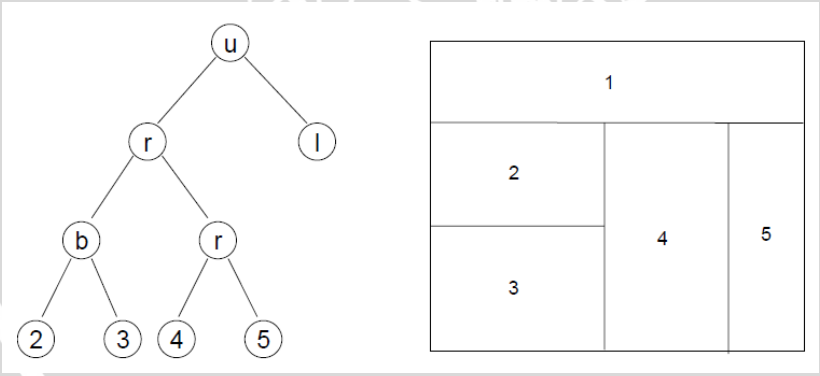
Pada model struktur kolom fleksible, peletakan departemen-departemen akan membentuk sebuah kolom atau baris. Tiap kolom tidak harus memiliki lebar yang sama dan tidak harus memiliki jumlah departemen yang sama. Lebar kolom secara otomatis ditentukan oleh departemen-departemen yang terkandung dalam kolom tersebut. Dengan demikian, permasalahan ini menjadi lebih sederhana untuk diselesaikan karena hanya dituntut untuk mengatur urutan departemen untuk diletakkan ke dalam kolom dan menentukan jumlah departemen yang akan dikandung tiap-tiap kolom. Gambar 2.1 menunjukkan sebuah solusi UA-FLP yang menggunakan model struktur kolom fleksibel (Shebanie, 2004).

.



Gambar 2.1 Model Struktur Pohon Fleksibel
(Shebanie, 2004)

Sedangkan pada model struktur pohon pemotong, solusi dari UA-FLP direpresentasikan oleh sebuah pohon. Representasi pohon tersebut berisi departemen sebagai node dan juga “u”, “b”, “r”, dan “l” sebagai penghubung node-node. Penghubung node memainkan peran sebagai operator pemotong (*slicing*) dalam pembentukan subfasilitas. Keempat penghubung node adalah “u” untuk potongan atas, “b” untuk potongan bawah, “r” untuk potongan kanan, dan “l” untuk potongan kiri. Node departemen dan penghubung node menentukan lokasi relative dan ketersinggungan dari departemen-departemen. Gambar 2.2 menunjukkan sebuah solusi UA-FLP yang menggunakan model struktur pohon pemotong (Shebanie, 2004).



Gambar 2.2 Model Struktur Pohon Pemotong
(Shebanie, 2004)

2.2.1 Formulasi Matematika

Formulasi matematika untuk menyelesaikan permasalahan tata letak fasilitas dengan luas tak sama (UA-FLP) berbeda-beda tergantung tujuan yang ingin dicapai.

Pada penelitian Honiden (2004), Fasilitas diasumsikan dengan bentuk empat persegi panjang dan ditetapkan oleh luas dan aspek rasio. Aspek rasio a_i dari fasilitas i yang ditetapkan sebagai berikut:

$$a_i = \frac{panjang\ fasilitas\ i}{lebar\ fasilitas\ i} = \frac{h_i}{w_i} \qquad (2.2)$$

Batasan pada bentuk fasilitas dapat ditentukan dengan membatasi aspek rasionya $[a_i^L, a_i^U]$, dimana a_i^L dan a_i^U adalah batas bawah dan batas atas yang membatasi a_i.

Tujuan pertama dalam FLP pada penelitian Honiden adalah untuk meminimalkan total biaya bagian yang berpindah di antara fasilitas. Mempertimbangkan luas lantai yang tidak ditentukan di dalam permasalahan ini dengan meminimalkan total luas tata ruang dimana semua fasilitas dan sekumpulan ruang yang sudah pasti dimasukkan sebagai tujuan kedua.

Karena itu, permasalahan tata letak fasilitas di sini didefinisikan dengan dua tujuan:

$$Min\ f = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} \cdot v_{ij} \cdot d_{ij} \qquad (2.3)$$

$$Min\ S = H.W \qquad (2.4)$$

Dengan syarat

$a_i^L \le a_i \le a_i^U, \quad i = 1,2,\dots,n$

$s_i = h_i \times w_i, \quad i = 1,2,\dots,n$

Dimana,

c_{ij} : biaya pemindahan antara fasilitas i dan j,

v_{ij} : banyaknya bagian yang berpindah antara fasilitas i dan j,

n : banyaknya fasilitas,

d_{ij} : jarak antara *centroid* fasilitas i dan j,
s_i : luas fasilitas i,
S : total luas yang menutupi semua fasilitas,
H : panjang total fasilitas,
W : lebar total fasilitas.

Penempatan fasilitas yang efektif dengan memfasilitasi perpindahan sumber daya, pada dasarnya juga mempertimbangkan interaksi antara fasilitas. Interaksi tersebut disebut sebagai hubungan kedekatan yang diinginkan antar fasilitas. Hubungan kedekatan menentukan fasilitas tersebut dekat atau terpisah satu sama lain. Model ini mengharuskan pengguna untuk menentukan bobot nilai kedekatan yang diinginkannya dengan penilaian hubungan antara setiap dua fasilitas. Jika dua fasilitas yang diperlukan dekat dengan satu sama lain, pengguna dapat memberi nilai bobot yang tinggi dan sebaliknya (Elbeltagi dan Hegazy, 2003).

Dalam literatur Francis dan White pada tahun 1974 ada enam nilai kedekatan yang biasanya ditetapkan di awal dan pengguna dapat memberikan bobot nilai yang diinginkan terkait setiap kategori. Dalam model ini, bobot nilai-nilai yang digunakan adalah ditampilkan pada Tabel 2.1, menyatakan hubungan eksponensial dengan kedekatan yang diinginkan. Nilai bobot satu berarti bahwa dua fasilitas tidak memiliki interaksi antara mereka dan jarak yang memisahkan mereka adalah tidak relevan (Elbeltagi dan Hegazy, 2003).

Tabel 2.1 Nilai Hubungan Kedekatan

Hubungan yang diinginkan antara fasilitas		Bobot kedekatan
Absolutely necessary	(A)	7500
Especially important	(E)	1500
Important	(I)	250
Ordinary closeness	(O)	50
Unimportant	(U)	10
Undesirable	(X)	1

Fungsi objektif yang digunakan (*total travel distance*) untuk menyelesaikan permasalahan tersebut dapat diformulasikan sebagai berikut (Elbeltagi dan Hegazy, 2003):

$$Min\ f = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} R_{ij} \qquad (2.5)$$

Dimana,

n : banyaknya jumlah fasilitas,
d_{ij} : jarak antara *centroid* fasilitas i dan j,
R_{ij} : nilai bobot kedekatan yang diinginkan antara fasilitas i dan j.

Metode pengukuran jarak antara dua fasilitas ada dua tipe, yaitu *rectangular* (Manhattan) dan *straight* (Euclidean). Dimana caranya adalah menjumlahkan jarak vertikal dan horizontal antara lokasi tersebut (Kado, 1995).

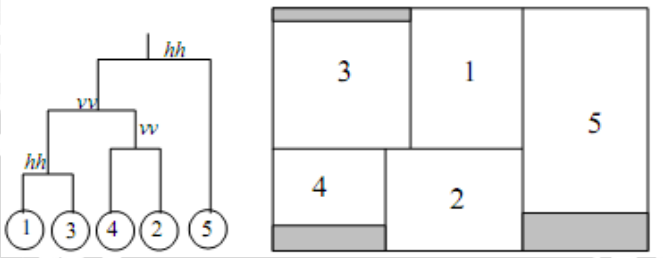
Jarak Manhattan $d = |X_i\text{-}X_j| + |Y_i\text{-}Y_j|$ (2.6)

Jarak Eulidean $d = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$ (2.7)

2.2.2 Model Struktur Pohon (*Tree Structure Modeling*)

Beberapa proses tata letak ruang dari fasilitas dengan luas tak sama dapat direpresentasikan dengan *tree structure model* yang ditunjukkan pada Gambar 2.3. Luas yang menaungi di dalam tata letak ruang diasumsikan mencukupi. Banyaknya angka di dalam *tree* mewakili banyaknya fasilitas dan huruf menunjukkan operasi yang menentukan orientasi fasilitas ketika meletakkan fasilitas berdekatan satu dengan yang lainnya. Orientasi dasar tiap fasilitas diberikan sebelumnya. Operasi "*h*" menyatakan bahwa fasilitas ditempatkan pada petunjuk yang sama seperti aslinya, dan operasi "*v*" menunjukkan rotasi fasilitas dalam 90°. Operasi tersebut dipilih untuk menghasilkan jarak minimum *centroid* untuk dua fasilitas yang berdekatan. Untuk menghasilkan variasi tata letak ruang dapat dilakukan dengan cara mengubah urutan fasilitas dan struktur pohon (*tree structure*). Untuk mengatasi UA-FLP ekuivalen dengan

penentuan struktur pohon (*tree structure*) sesuai dengan tata letak ruang dengan nilai fungsi objektif yang minimum (Honiden, 2004).



Gambar 2.3 Contoh Tata Letak dan Struktur Pohonnya (Honiden, 2004)

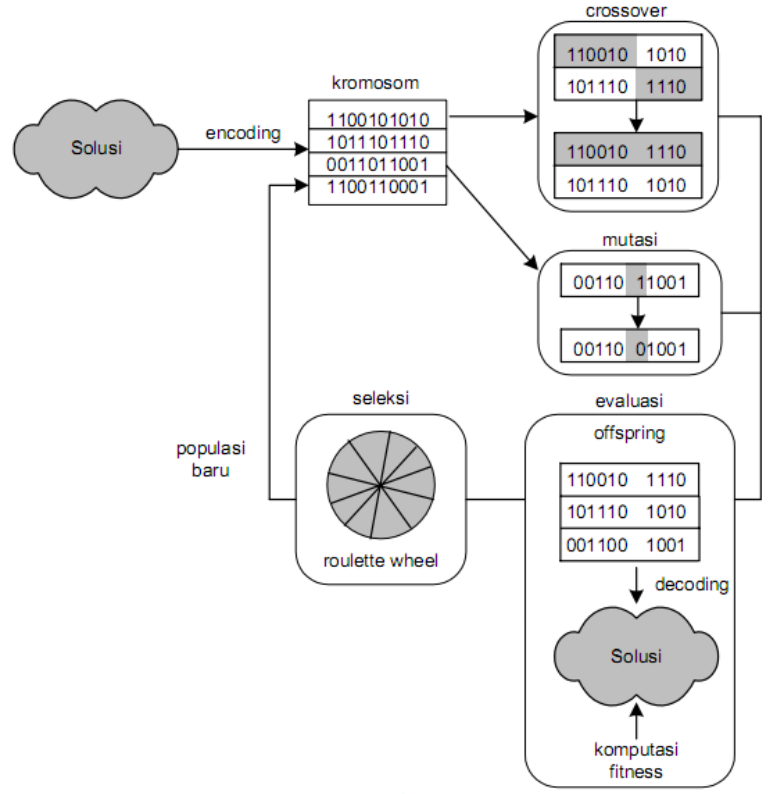
2.3 Algoritma Genetika

Algoritma genetika adalah salah satu model kecerdasan buatan (*artificial intelligence*). Algoritma genetika diperkenalkan pertama kali oleh John Holland dari Universitas Michigan pada tahun 1975. John Holland menyatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan ke dalam terminologi genetika. Kemudian, Goldberg pada tahun 1989 mendefinisikan algoritma genetika sebagai suatu pencarian algoritma berdasarkan pada mekanisme seleksi alam dan genetika alam. Bauer pada tahun 1993 mendefinisikan algoritma genetika sebagai perangkat lunak, prosedur yang dimodelkan setelah genetika dan evolusi (Desiani dan Arhami, 2006).

Berbeda dengan teknik pencarian konvensional, algortima genetika berangkat dari himpunan solusi yang dihasilkan secara acak. Himpunan ini disebut populasi. Sedangkan setiap individu dalam populasi disebut kromosom yang merupakan representasi dari solusi. Kromosom-kromosom berevolusi dalam suatu proses yang berkelanjutan yang disebut generasi. Pada setiap generasi, kromosom dievaluasi berdasarkan suatu fungsi evaluasi (Gen dan Cheng, 1997). Setelah beberapa generasi maka algoritma genetika akan konvergen pada kromosom terbaik, yang diharapkan merupakan solusi optimal (Goldberg, 1989).

2.3.1 Struktur Umum Algoritma Genetika

Struktur umum dari algoritma genetika dapat digambarkan pada Gambar 2.4 berikut:



Gambar 2.4 Diagram Struktur Umum Algoritma Genetika (Subakti, 2006)

Secara umum, algoritma genetika memiliki lima komponen dasar (Michalewicz, 1996):

1. Representasi genetika dari kemungkinan solusi permasalahan.
2. Cara untuk membentuk populasi awal dari kemungkinan solusi.

3. Fungsi evaluasi yang menilai solusi berdasarkan *fitness*-nya.
4. Operator genetika yang mengubah komposisi genetika dari kromosom anak selama proses reproduksi.
5. Nilai untuk parameter algoritma genetika.

Sebenarnya, hanya ada 2 jenis operasi dalam algoritma genetika, yaitu (Subakti, 2006):

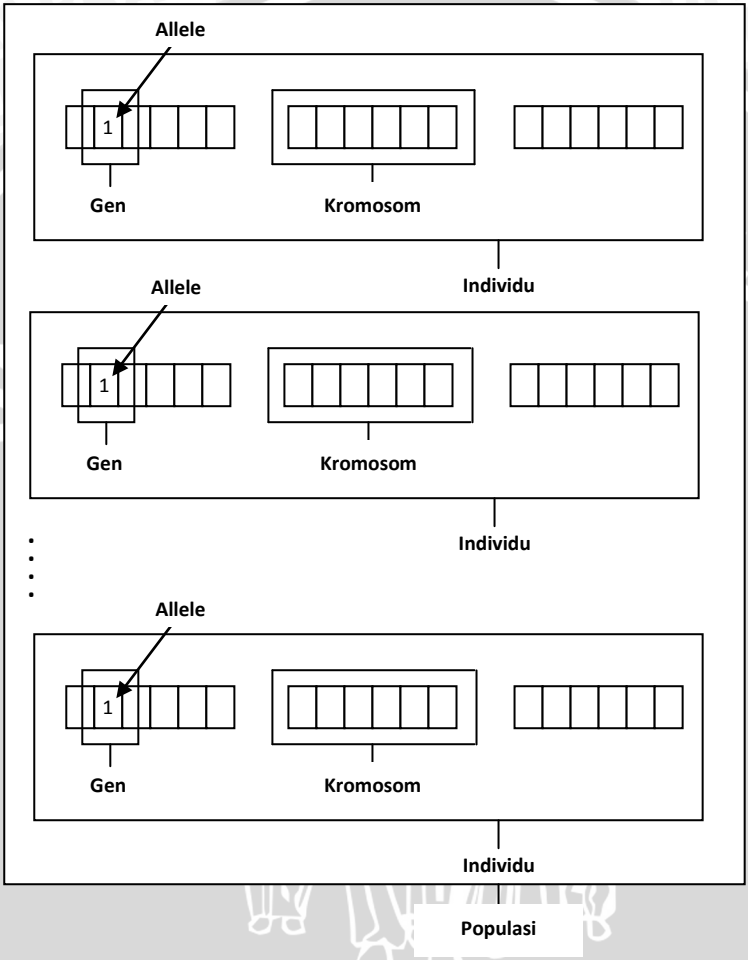
1. Operasi genetika: *crossover* (pindah silang) dan mutasi
2. Operasi evolusi: seleksi

2.3.2 Pengkodean

Pengkodean adalah suatu teknik untuk menyatakan populasi awal sebagai calon solusi suatu masalah ke dalam suatu kromosom (Gen dan Cheng, 2000). Beberapa definisi penting yang perlu diperhatikan dalam mendefinisikan individu/solusi (pengkodean) untuk membangun penyelesaian permasalahan dengan algoritma genetika adalah sebagai berikut (Hartanti, 2007):

- ***Genotype* (Gen)**, sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom.
- ***Allele***, nilai dari gen.
- **Kromosom**, gabungan gen-gen yang membentuk nilai tertentu.
- **Individu**, menyatakan suatu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
- **Populasi**, merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi.
- **Generasi**, menyatakan satu siklus proses evolusi atau satu iterasi di dalam algoritma genetika.

Dari ilustrasi perbedaan istilah-istilah di atas dapat dilihat pada Gambar 2.5 berikut:



Gambar 2.5 Representasi Penyelesaian Permasalahan dalam Algoritma Genetika

Beberapa pengkodean menurut Marek Obitko (1998) dapat dijelaskan sebagai berikut:

2.3.2.1 Pengkodean Biner

Pengkodean jenis ini sering digunakan. Kromosom dari pengkodean biner ini berupa kumpulan dari nilai biner 0 dan 1. Contoh pengkodean biner dapat dilihat pada Gambar 2.6.

Kromosom A	10110010110010101100101
Kromosom B	11111110000011000001111 1

Gambar 2.6 Pengkodean Biner

Dalam pengkodean biner memungkinkan didapatkan kromosom dengan nilai *allele* yang kecil, tetapi kekurangannya tidak dapat digunakan untuk beberapa permasalahan dan terkadang diperlukan adanya koreksi setelah proses *crossover* dan mutasi. Salah satu permasalahan yang menggunakan pengkodean ini adalah menghitung nilai maksimal dari suatu fungsi.

2.3.2.2 Pengkodean Permutasi

Kromosom dari pengkodean permutasi ini berupa kumpulan nilai integer yang mewakili suatu posisi dalam sebuah urutan. Biasanya digunakan pada permasalahan TSP (Travelling Salesman Problem). Contoh pengkodean permutasi dapat dilihat pada Gambar 2.7.

Kromosom A	1 5 3 2 6 4 7 9 8
Kromosom B	8 5 6 7 2 3 1 4 9

Gambar 2.7 Pengkodean Permutasi

2.3.2.3 Pengkodean Nilai

Kromosom dari pengkodean nilai berupa kumpulan dari suatu nilai, yang bisa berupa macam-macam nilai sesuai dengan permasalahan yang dihadapi, seperti bilangan real, char atau objek yang lain. Pengkodean ini merupakan pilihan yang bagus untuk beberapa permasalahan khusus, biasanya diperlukan metode khusus untuk memproses *crossover* dan mutasinya sesuai dengan

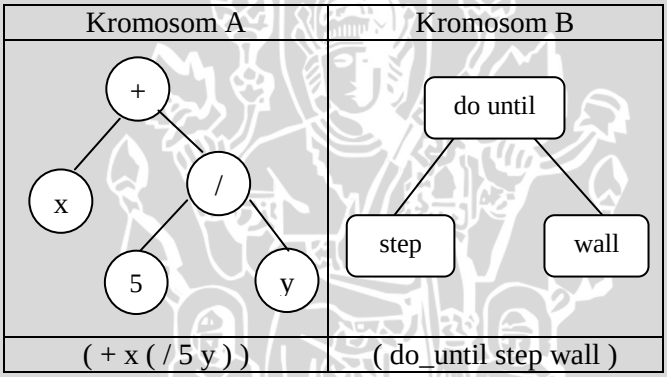
permasalahan yang dihadapi. Contoh pengkodean nilai sebagai berikut:

Kromosom A	1.2324 5.3243 0.4556 2.3293 2.4545
Kromosom B	ABDJEIFJDHDIERJFDLDFLFEGT
Kromosom C	(back), (back), (right), (forward), (left)

Gambar 2.8 Pengkodean Nilai

2.3.2.4 Pengkodean Pohon

Pengkodean pohon biasanya untuk program evolusi atau ekspresi, pada *genetic programming*. Kromosom yang digunakan berupa sebuah pohon dari beberapa objek, seperti fungsi atau perintah pada bahasa pemrograman.



Gambar 2.9 Pengkodean Pohon

2.3.3 Nilai *Fitness*

Nilai *Fitness* adalah nilai yang menyatakan baik tidaknya suatu solusi (individu) yang didapat. Dengan kata lain nilai *fitness* menyatakan nilai dari fungsi objektif. Nilai *fitness* ini yang dijadikan acuan dalam mencapai nilai optimal dalam algoritma genetika. Algoritma genetika bertujuan mencari individu dengan nilai *fitness* paling tinggi. Fungsi objektif/fungsi *fitness* ini merupakan ukuran kinerja suatu individu agar tetap bertahan hidup dalam lingkungannya (Suyanto, 2007).

Untuk permasalahan minimalisasi, nilai fitness adalah inverse dari nilai minimal yang diharapkan. Proses inverse dapat dilakukan dengan rumus berikut (Basuki, 2003):

$$Fitness = A - F(X) \quad (2.8)$$

atau:

$$Fitness = \frac{A}{F(X) + e} \quad (2.9)$$

dimana,

A = konstanta yang ditentukan
X = individu (kromosom)
e = bilangan kecil yang ditentukan untuk menghindari pembagi nol atau $F(X) = 0$

2.3.4 Seleksi

Seleksi digunakan untuk memilih individu-individu mana saja yang akan dipilih untuk proses *crossover* dan mutasi. Seleksi digunakan untuk mendapatkan calon induk yang baik. "Induk yang baik akan menghasilkan keturunan yang baik". Semakin tinggi nilai *fitness* suatu individu semakin besar kemungkinannya untuk terpilih (Kusumadewi, 2003).

Langkah pertama dalam seleksi ini adalah pencarian nilai *fitness*. Masing-masing individu dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam wadah seleksi tersebut. Nilai *fitness* ini yang nantinya digunakan pada tahap-tahap seleksi berikutnya (Kusumadewi, 2003).

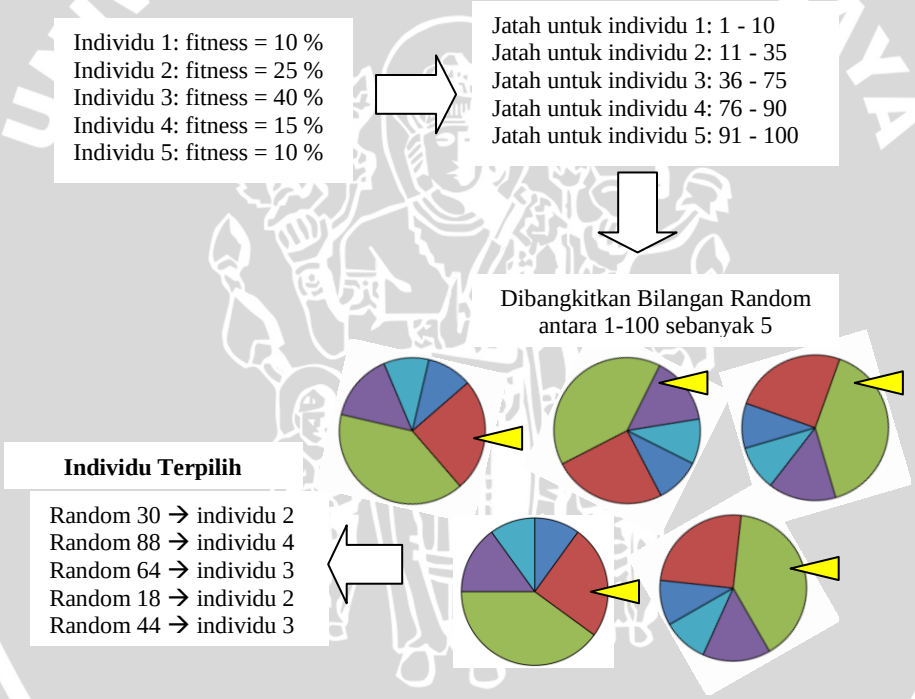
Terdapat beberapa metode seleksi, antara lain: seleksi dengan metode mesin *roulette* dan seleksi dengan turnamen (Hartanti, 2007).

2.3.4.1 Seleksi dengan Mesin *Roulette*

Metode seleksi dengan mesin *roulette* ini merupakan metode yang paling sederhana dan sering dikenal dengan nama *stochastic*

sampling with replacement. Cara kerja metode ini adalah sebagai berikut (Hartanti, 2007):

1. Menghitung nilai *fitness* dari masing-masing individu (f_i, dimana *i* adalah individu ke-1 s/d ke-n).
2. Menghitung total *fitness* semua individu.
3. Menghitung probabilitas masing-masing individu.
4. Dari probabilitas tersebut, menghitung jatah masing-masing individu pada angka 1 sampai 100.
5. Membangkitkan bilangan random antara 1 sampai 100.
6. Dari bilangan random yang dihasilkan, menentukan individu mana yang terpilih dalam proses seleksi.



Gambar 2.10 Seleksi dengan Mesin *Roulette*

2.3.4.2 Seleksi dengan Turnamen

Pada metode seleksi dengan turnamen, ditetapkan suatu *tour* untuk individu-individu yang dipilih secara random dari suatu populasi. Individu-individu yang terbaik dalam kelompok ini akan diseleksi sebagai induk. Parameter yang dugunakan pada metode ini adalah ukuran *tour* yang bernilai 2 sampai N (jumlah individu dalam suatu populasi). Kebanyakan metode seleksi ini digunakan pada *binary*, dimana hanya dua individu yang dipilih (Hartanti, 2007).

2.3.5 *Crossover*

Crossover (pindah silang) adalah operator dari algoritma genetika yang melibatkan dua induk untuk membentuk kromosom baru. *Crossover* menghasilkan titik baru dalam ruang pencarian yang siap untuk diuji. Operasi ini tidak selalu dilakukan pada semua individu yang ada. Individu dipilih secara acak untuk dilakukan *crossing* dengan P_c antara 0,6 s/d 0,95. Jika pindah silang tidak dilakukan, maka nilai dari induk akan diturunkan kepada keturunan (Desiani dan Arhami, 2006).

Prinsip dari *crossover* adalah melakukan operasi (pertukaran, aritmatika) pada gen-gen yang bersesuaian dari dua induk untuk menghasilkan individu baru. Proses *crossover* dilakukan pada setiap individu dengan probabilitas *crossover* yang ditentukan. Ada beberapa cara yang bisa digunakan untuk melakukan *crossover*, sesuai dengan pengkodeannya yang dijelaskan sebagai berikut (Obitko, 1998 dan Hartanti, 2007):

2.3.5.1 *Crossover* untuk Pengkodean Biner

Pada pengkodean biner terdapat 4 metode *crossover* yang sering digunakan antara lain:

a. *Crossover* Satu Titik

   Memilih satu titik tertentu, selanjutnya nilai biner sampai titik *crossover*-nya dari induk pertama digunakan dan sisanya dilanjutkan dengan nilai biner dari induk kedua.

   Contoh:

   11001011 + 11011**111** = **11001111**

b. *Crossover* Dua Titik
Memilih dua titik tertentu, lalu nilai biner sampai titik *crossover* pertama pada induk pertama digunakan, dilanjutkan dengan nilai biner dari titik pertama sampai titik kedua, kemudian sisanya melanjutkan nilai biner dari titik kedua pada induk pertama lagi.
Contoh:

11001**01** + **11**011**11** = **11011111**

c. *Crossover Uniform*
Nilai biner yang digunakan dipilih secara random dari kedua induk.
Contoh:

1**10**01**01** + **1**10**111**01 = 11011111

d. *Crossover* Aritmatik
Suatu operasi aritmetika digunakan untuk menghasilkan *offspring* yang baru.
Contoh:

11001011 + 11011111 = 11001001 (AND)

2.3.5.2 *Crossover* untuk Pengkodean Permutasi

Memilih satu titik tertentu, nilai permutasi sampai titik *crossover* pada induk pertama digunakan, lalu sisanya dilakukan *scan* terlebih dahulu, jika nilai permutasi pada induk kedua belum ada pada *offspring*, nilai tersebut ditambahkan.

Contoh:

(123456789) + (**45368**9721) = (**12345**689)

Beberapa metode operator *crossover* diciptakan untuk representasi permutasi, seperti *Partial-Mapped Crossover*, *Order Crossover*, dan *Cycle Crossover*, akan dijelaskan berikut (Hartanti, 2007).

Partial-Mapped Crossover (PMX). PMX diciptakan oleh Goldberg dan Lingle. PMX merupakan rumusan modifikasi dari pindah silang dua-poin. Hal yang penting dari PMX adalah pindah silang dua-poin ditambah dengan beberapa prosedur tambahan. PMX mempunyai langkah kerja sebagai berikut:

1. Menentukan dua posisi pada kromosom dengan aturan acak. Substring yang berada dalam dua posisi ini dinamakan daerah pemetaan.
2. Menukar dua substring antar induk untuk menghasilkan *proto-child*.
3. Menentukan hubungan pemetaan di antara dua daerah pemetaan.
4. Menentukan kromosom keturunan mengacu pada hubungan pemetaan.

Order Crossoveer (OX). OX diciptakan oleh Davis. Metode ini merupakan variasi dari PMX dengan prosedur tambahan. OX bekerja sebagai berikut:

1. Memilih substring dari sebuah induk secara acak.
2. Membangkitkan sebuah *proto-child* dengan mengkosongkan tempat substring induk 2 pada induk 1.
3. SHR *allele* dari substring pada tempat yang bersesuaian.
4. Menukar substring antara 2 induk.

Cycle Crossover (CX). CX diciptakan oleh Oliver, Smith dan Holland. Metode ini mengkopi gen-gen dari satu induk dan memilih gen-gen yang lain dari induk yang lain, dengan mengingat dan pola *cycle*. Cara kerja CX adalah sbb:

1. Menemukan *cycle* yang didefinisikan dari relasi posisi gen-gen antara induk
2. Menyalin gen-gen dalam *cycle* pada *proto-child* dengan relasi posisi dari sebuah induk
3. Menentukan gen-gen diingat yang berasal dari induk lain
4. Mengisi keturunan dengan gen-gen yang diingat tadi.

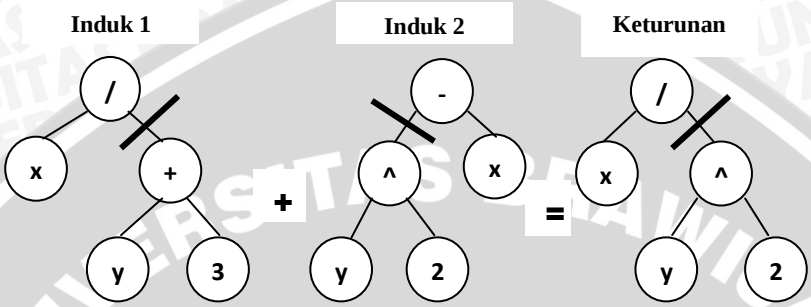
2.3.5.3 *Crossover* untuk Pengkodean Nilai

Semua metode *crossover* pada pengkodean biner bisa digunakan pada *crossover* untuk pengkodean nilai.

2.3.5.4 *Crossover* untuk Pengkodean Pohon

Memilih satu titik tertentu dari tiap induk, dan menggabungkan pohon di bawah titik pada induk pertama dan pohon

di bawah titik pada induk kedua. Contoh crossover untuk pengkodean pohon bisa dilihat pada Gambar 2.11 berikut:



Gambar 2.11 *Crossover* untuk Pengkodean Pohon

2.3.6 Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Mutasi diperlukan untuk mengembalikan informasi bit yang hilang akibat *crossover*. Banyaknya individu yang mengalami mutasi ditentukan oleh besarnya probabilitas mutasi.

Metode dan tipe mutasi yang dilakukan tergantung pada pengkodean dan permasalahan yang diangkat. Berdasarkan pengkodean, terdapat beberapa macam diantaranya (Obitko, 1998):

8.3.6.1 Mutasi untuk Pengkodean Biner

Pada pengkodean biner, mutasi yang digunakan adalah *bit inversion* yaitu, melakukan *inverse* pada bit yang terpilih, 0 menjadi 1 dan sebaliknya 1 menjadi 0.
Contoh:

1**1**001001 => 1**0**001001

8.3.6.2 Mutasi untuk Pengkodean Permutasi

Pada pengkodean permutasi, mutasi yang digunakan adalah *order changing* yaitu, memilih dua nilai dari gen dan menukarnya.

Contoh:

(1 **2** 3 4 5 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)

Beberapa operator mutasi telah diciptakan untuk pengkodean permutasi, seperti metode *Inversion*, *Insertion*, *Displacement*, dan *Reciprocal Exchange Mutation*.

Inversion Mutation memilih dua posisi dalam sebuah kromosom dengan cara acak dan kemudian menginversikan substring di antara dua posisi tersebut. *Insertion Mutation* memilih sebuah gen dengan cara acak dan memasukkan ke dalam kromosom dengan cara acak pula. *Displacement Mutation* memilih sebuah sub/sekelompok gen dengan cara acak kemudian memasukkan ke dalam kromosom dengan cara acak. Sedangkan *Reciprocal Exchange Mutation* (REM) memilik dua posisi acak, kemudian menukar dua gen dalam posisi tersebut.

8.3.6.3 Mutasi untuk Pengkodean Nilai

Pada pengkodean nilai dilakukan penentuan sebuah nilai kecil yang akan ditambahkan atau dikurangkan pada salah satu gen dalam kromosom.

Contoh:

(1.29 5.68 **2.86 4.11** 5.55) => (1.29 5.68 **2.73 4.22** 5.55)

8.3.6.4 Mutasi untuk Pengkodean Pohon

Pada pengkodean pohon mutasi yang dilakukan adalah *changing operator* dimana node yang terpilih akan diubah.

2.3.7 Parameter Genetika

Pemilihan parameter genetika menentukan penampilan kinerja algoritma genetika dalam memecahkan masalah. Ada dua parameter dasar dari algoritma genetika, yaitu probabilitas *crossover* (Pc) dan probabilitas mutasi (Pm). Parameter lain yang juga ikut menentukan efisiensi kinerja algoritma genetika adalah ukuran populasi (Desiani dan Arhami, 2006 dan Suyanto, 2007).

2.3.7.1 Probabilitas *Crossover*

Probabilitas *crossover* menyatakan seberapa sering proses *crossover* akan terjadi antara dua kromosom orang tua. Jika tidak terjadi *crossover,* satu orang tua dipilih secara random dengan probabilitas yang sama dan diduplikasi menjadi anak. Jika terjadi *crossover,* keturunan dibuat dari bagian-bagian kromosom orang tua. Jika probabilitas *crossover* 100% maka keseluruhan keturunan dibuat dengan *crossover*. Jika *crossover* 0% maka seluruh generasi baru dibuat dari salinan kromosom-kromosom dari populasi lama yang belum tentu menghasilkan populasi yang sama dengan populasi sebelumnya karena adanya penekanan selektif.

Hasil penelitian yang sudah pernah dilakukan oleh praktisi algoritma genetika menunjukkan bahwa angka probabilitas *crossover* sebaiknya cukup tinggi, yaitu antara 80% sampai 95% untuk memberikan hasil yang baik. Untuk beberapa masalah tertentu probabilitas *crossover* 60% memberikan hasil yang baik.

2.3.7.2 Probabilitas Mutasi

Probabilitas mutasi menyatakan seberapa sering bagian-bagian kromosom akan dimutasi. Jika tidak ada mutasi, keturunan diambil/disalin langsung setelah *crossover* tanpa perubahan. Jika mutasi dilakukan, bagian-bagian kromosom diubah. Jika probabilitas mutasi 100%, keseluruhan kromosom diubah. Jika probabilitas mutasi 0%, tidak ada yang diubah. Probabilitas mutasi dalam algoritma genetika seharusnya diberi nilai yang kecil. Umumnya, probabilitas mutasi diset untuk mendapatkan rata-rata satu mutasi per kromosom, yaitu angka/*allele* = 1/(panjang kromosom). Dari hasil yang sudah pernah dicoba menunjukkan bahwa angka probabilitas terbaik adalah antara 0,5% sampai 1%.

2.3.7.3 Ukuran Populasi

Ukuran populasi adalah banyaknya kromosom dalam satu populasi. Jika terlalu sedikit kromosom dalam populasi, algoritma genetika mempunyai kemungkinan yang sedikit untuk melakukan *crossover* dan hanya sebagian kecil dari ruang yang dieksplorasi. Sebaliknya, jika terlalu banyak jumlah kromosom, algoritma genetika cenderung lambat dalam menentukan solusi. Ukuran

populasi yang sering digunakan oleh peneliti yang sudah ada adalah antara 20 sampai 30, tetapi terkadang ukuran 50 sampai 100 dilaporkan baik.

2.3.8 Keuntungan Menggunakan Algoritma Genetika

Ada 3 keuntungan utama dalam mengaplikasikan algoritma genetika pada masalah-masalah optimasi (Subakti, 2006):

1. Algoritma genetika tak banyak memerlukan kebutuhan matematis mengenai masalah optimasi
2. Kemudahan dan kenyamanan dan pada operator-operator evolusi membuat algoritma genetika sangat efektif dalam melakukan pencarian global (dalam probabilitas)
3. Algoritma genetika menyediakan banyak fleksibelitas untuk digabungkan dengan metode *heuristic* yang tergantung domain, untuk membuat implementasi yang efisien pada masalah-masalah khusus.

2.4 Pendekatan Algoritma Genetika Untuk Menyelesaikan UA-FLP

Honiden (2004) menggunakan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* yang dapat menemukan struktur pohon (*tree structure*) yang mendekati optimal.

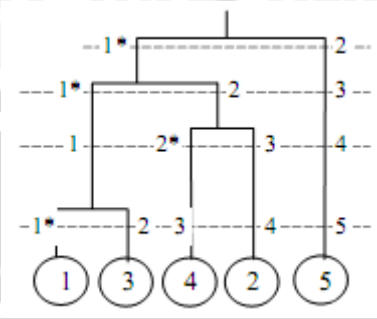
3.4.1 Inisialisasi Kromosom

Populasi awal kromosom dibangkitkan secara random. Struktur pohon (*tree structure*) dapat direpresentasikan dengan kromosom sebagai berikut:

$$(a_1\ a_2\ a_3\ \ldots\ a_n : b_1\ b_2\ b_3\ \ldots\ b_{n-2}) \qquad (2.10)$$

Kromosom ini berisi dua bagian string numerik, yaitu a_i, b_i dimana string a_i menunjukkan urutan fasilitas dalam struktur pohon (*tree structure*), string b_j menunjukkan urutan pengelompokan fasilitas. Karakter ":" adalah sebuah pembatas. Misalnya kromosom yang sesuai dengan struktur pohon (*tree structure*) pada Gambar 2.12 adalah sebagai berikut

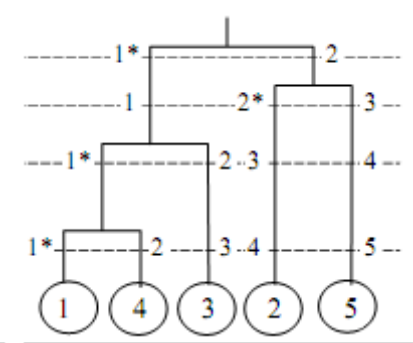
(1 3 4 2 5:1 2 1)



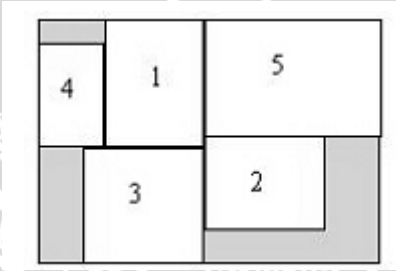
Gambar 2.12 Struktur Pohon
(Honiden, 2004)

Urutan gen "1 3 4 2 5" sama dengan urutan fasilitas yang diminta pada Gambar 2.11. Gen yang belakang "1 2 1" sesuai angka numerik dengan "*" pada struktur pohon (*tree structure*) tersebut. Angka "1*" menunjukkan bahwa fasilitas pertama "1" dan fasilitas berikutnya "3" dikelompokkan pertama kali. Kemudian angka "2*" menunjukkan bahwa fasilitas kedua "4" dan fasilitas berikutnya "2" dikelompokkan pada prosedur pengelompokan kedua. Pada prosedur pengelompokan ketiga, kelompok fasilitas pertama "1 3" dan kelompok fasilitas berikutnya "4 2" dikelompokkan. Sehingga dua kelompok terakhir fasilitas digabungkan, yang terakhir angka "1*" diabaikan dalam kromosom. Banyaknya jumlah gen dalam string b_j adalah (n-2). Pada Gambar 3.1 nilai b_j dibatasi $1 \leq b_j \leq n\text{-}j$, $j = 1, 2, \ldots, n\text{-}2$.

Jika dua ruangan bertukar dalam pohon, sepasang fasilitas gabungan yang berdekatan akan berubah. Jika struktur pohon (*tree structure*) berubah, proses pengelompokan fasilitas juga berubah. Misalnya, ketika struktur pohon (*tree structure*) pada Gambar 2.11 diubah menjadi Gambar 2.13, kromosom dapat direpresentasikan dengan (1 4 3 2 5: 1 1 2). Dan tata letak fasilitas yang sesuai dengan modifikasi struktur pohon (*tree structure*) ditunjukkan pada Gambar 2.14. Untuk bisa menghasilkan variasi tata letak dapat dilakukan dengan mengubah gen di dalam struktur pohon (*tree structure*) kromosom.



Gambar 2.13 Struktur Pohon yang Dimodifikasi
(Honiden, 2004)



Gambar 2.14 Tata Letak Fasilitas yang Dimodifikasi
(Honiden, 2004)

3.4.2 Fungsi *Fitness*

Facility Layout Problem (FLP) didefinisikan pada subbab 2.2.1 dengan rumus 2.3 dan 2.4. Sehingga fungsi *fitness* yang digunakan sebagai berikut:

$$z_i = \frac{f_i}{f_{max}} \cos\theta + \frac{S_i}{S_{max}} \sin\theta \qquad (2.11)$$

Di mana,

f_i : total biaya bagian yang dipindahkan kromosom *i*,
S_i : total luas kromosom *i*,
f_{max} : nilai maksimum f_i,
S_{max} : nilai maksimum S_i,
θ : nilai random antara 0 dan $\pi/2$

Fungsi *fitness* untuk solusi yang sesuai dengan kromosom *i* didefinisikan sebagai berikut:

$$F_i = z_{max} - z_i \quad (2.12)$$

Dimana, $z_{max} = cos\theta + sin\theta$. z_{max} adalah nilai z_i ketika $f_i = f_{max}$, $s_i = s_{max}$. Fungsi *fitness* digunakan pada prosedur seleksi.

Ketika memilih kromosom untuk generasi selanjutnya kromosom non-dominasi akan dipilih pertama. Jika jumlah kromosom non-dominasi kurang dari ukuran populasi, kromosom dominasi ditambahkan ke dalam populasi.

3.4.3 Crossover

Pada gen pertama dari kromosom bisa dilakukan *crossover* PMX, OX, dan CX. *Crossover* tersebut dilakukan agar tidak menghasilkan keturunan yang tidak layak. Berdasarkan penelitian Honiden, 2004, *crossover* PMX adalah operator *crossover* yang lebih efektif untuk mengatasi solusi non-dominasi di antara operator lainnya. *Crossover* satu titik digunakan pada gen kedua berdasarkan *crossover* gen pertama.

3.4.4 Mutasi

Mutasi ini dilakukan dengan mempertukarkan dua gen yang dipilih di bagian bekas keturunan secara acak. Mutasi untuk bagian akhir adalah nilai satu gen b_j yang dipilih secara random diubah berdasarkan $1 \leq b_j \leq n\text{-}j$.

3.4.5 Seleksi

Sepasang kromosom diambil dari populasi menggunakan seleksi dengan mesin *roulette*. Probabilitas pemilihan kromosom *i* dihitung sebagai berikut:

$$P_i = \frac{F_i}{\sum_{j=1}^{PS} F_j} \quad (2.13)$$

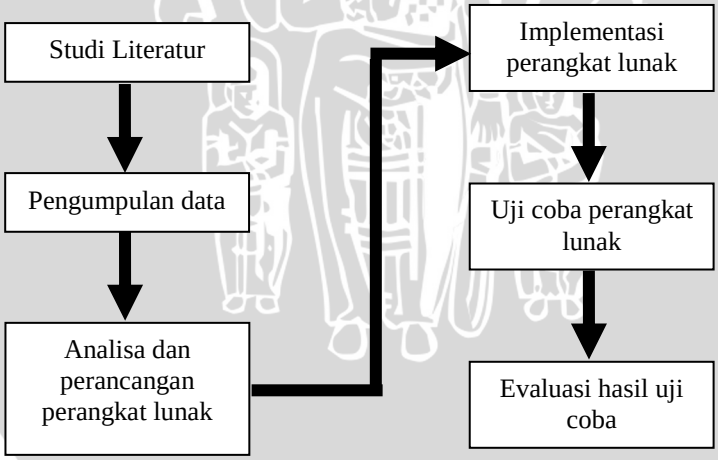
PS adalah ukuran populasi. Ketika menghitung z_i di F_i pada setiap generasi, diberikan secara acak. Kemudian dapat dipilih berbagai kromosom non-dominasi.


UNIVERSITAS BRAWIJAYA

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan ini akan dibahas langkah-langkah yang digunakan dalam penelitian tentang *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* dengan studi kasus desain tata letak ruang dalam rumah. Penelitian dilakukan dengan tahapan-tahapan sebagai berikut:

1. Mempelajari literatur yang berhubungan dengan desain tata letak ruang dalam rumah dan metode yang digunakan, yaitu *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem*.
2. Mengumpulkan data-data desain tata letak ruang dalam rumah dari literatur dan wawancara langsung.
3. Menganalisa dan merancang perangkat lunak.
4. Membuat perangkat lunak berdasarkan analisis dan perancangan yang telah dilakukan (implementasi).
5. Uji coba perangkat lunak.
6. Evaluasi hasil uji coba yang sudah dilakukan oleh sistem.



Gambar 3.1 Langkah-Langkah Pembuatan Perangkat Lunak

3.1 Analisis Sistem

3.1.1 Deskripsi Sistem

Sistem yang akan dibangun merupakan sistem yang dibuat sebagai implementasi teori-teori yang didapat dari berbagai literatur mengenai permasalahan desain tata letak ruang dalam rumah. *Tree structure modeling* dan algoritma genetika digunakan untuk menyelesaikan *unequal-area facility layout problem* untuk mengetahui bagaimana metode-metode tersebut dalam menyelesaikan permasalahan desain tata letak ruang dalam rumah tersebut. Sistem yang dibangun akan memberikan desain tata letak ruang dalam rumah yang paling mendekati optimal kepada *user*.

3.1.2 Batasan Sistem

Sistem ini dibangun untuk menyelesaikan permasalahan desain tata letak ruang dalam rumah yang sudah disebutkan pada bab 1. Metode yang digunakan adalah *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* dengan studi kasus desain tata letak ruang dalam rumah. Luas lahan yang digunakan berkisar antara 60 m^2 sampai dengan 200 m^2. Luas bangunan (total luas keseluruhan ruangan yang dibutuhkan) tidak boleh melebihi 60% dari luas lahan yang tersedia (Amin, dkk, 2010). Jumlah kebutuhan ruangan yang dimasukkan hanya dibatasi 15 ruangan, dimana ruangan tersebut harus sesuai spesifikasi yang ditentukan. Nilai kedekatan antar ruang yang digunakan sesuai Tabel 2.1 Nilai hubungan kedekatan.

3.1.3 Data yang Digunakan

Pada penelitian ini, data yang digunakan dalam pengujian sistem adalah data-data yang diambil dari contoh permasalahan nyata yang diambil dari internet dan buku "101 Denah Rumah" dalam hal mendesain denah rumah dengan penyesuaian batasan masalah pada bab 1 dan batasan sistem pada subbab 3.1.2.

3.2 Perancangan Sistem

Sistem yang akan dibangun menggunakan *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area*

facility layout problem. Perancangan dari sistem ini dibangun mulai dari beberapa tahapan yaitu inputan yang dimasukan *user* sampai dihasilkan desain tata letak ruang dalam rumah yang memiliki nilai *fitness* paling tinggi. Serta melakukan uji keefektifan algoritma genetika dalam menyelesaikan studi kasus desain tata letak ruang dalam rumah.

3.2.1 Perancangan Penyelesaian Desain Tata Letak Ruang dalam Rumah

Proses desain tata letak ruang dalam rumah dengan *tree structure modeling* dan algoritma genetika dimulai dengan menentukan besarnya parameter genetika yang terdiri dari ukuran populasi, maksimal generasi, peluang *crossover*, dan peluang mutasi. Selanjutnya menginputkan ketersediaan lahan, kebutuhan ruangan, dan nilai kedekatan antar ruang sebagai parameter desain. Setelah itu dilakukan inisialisasi kromosom dari *tree structure modeling* yang terbentuk, perhitungan nilai *fitness*, seleksi, *crossover*, sampai dengan mutasi dan seterusnya ke generasi berikutnya.

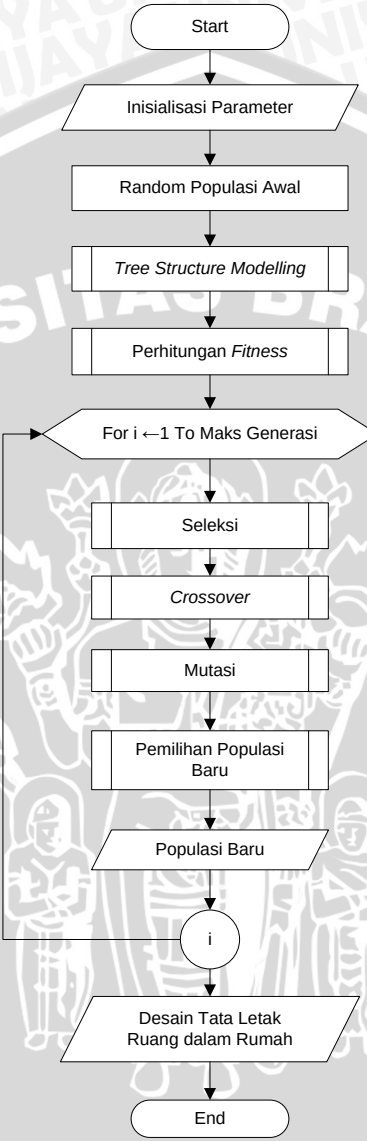
3.2.2 Diagram Alir Desain Tata Letak Ruang dalam Rumah

Secara keseluruhan implementasi *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* dengan studi kasus desain tata letak ruang dalam rumah adalah sebagai berikut:

1. Menginisialisasi semua parameter yang dibutuhkan pada desain tata letak ruang dalam rumah.
   a. Parameter genetika, antara lain:
      - Ukuran populasi.
      - Maksimal generasi.
      - Probabilitas *crossover* (Pc).
      - Probabilitas mutasi (Pm).
   b. Parameter desain, antara lain:
      - Ketersediaan lahan, meliputi panjang, dan lebar lahan.
      - Kebutuhan ruangan, meliputi jenis, panjang dan lebar ruang yang dibutuhkan.
      - Nilai kedekatan antar ruang yang dibutuhkan.

2. Membangkitkan populasi awal secara random sebanyak ukuran populasi yang sudah ditentukan.
3. Melakukan proses desain tata letak setiap kromosom pada populasi berdasarkan *tree structure* yang terbentuk untuk mendapatkan titik *centroid* tiap ruang (gen pertama). Sehingga diperoleh jarak antar ruang.
4. Evaluasi setiap kromosom pada populasi dengan menghitung *total travel distance* setiap kromosom untuk mendapatkan nilai *fitness*-nya. *Total travel distance* diperoleh dari jarak antar ruang (hasil langkah 3) dan nilai kedekatan antar ruang yang sudah ditentukan.
5. Membentuk populasi baru dengan melakukan langkah-langkah sebagai berikut:
   a. Melakukan proses seleksi pada populasi dengan menggunakan metode seleksi dengan mesin *roulette* untuk menentukan kromosom yang layak menjadi induk pada proses *crossover*.
   b. Melakukan *crossover* antara kedua induk hasil seleksi dengan menggunakan metode *Partial-Mapped Crossover* (PMX) pada gen pertama dan metode *crossover* satu titik pada gen kedua sesuai dengan probabilitas *crossover* yang sudah ditentukan.
   c. Melakukan mutasi pada kromosom, baik dari populasi awal ataupun dari keturunan yang dihasilkan, dengan menggunakan metode *Reciprocal Exchange Mutation* (REM) pada gen pertama dan metode mutasi random pada gen kedua sesuai dengan probabilitas mutasi yang sudah ditentukan.
   d. Memilih kromosom dengan nilai *fitness* tertinggi sebanyak ukuran populasi dari populasi sementara (populasi awal atau populasi sebelumnya, total kromosom keturunan, dan total kromosom mutasi) untuk dimasukkan ke dalam populasi baru.
6. Menggunakan populasi baru untuk melakukan kembali langkah 5.
7. Proses-proses tersebut dilakukan sesuai dengan maksimal generasi yang sudah ditentukan.

Langkah-langkah dari seluruh proses di atas dapat digambarkan dengan *flowchart* pada Gambar 3.2.



Gambar 3.2 *Flowchart* Proses *Tree Structure Modeling* dan Algoritma Genetika

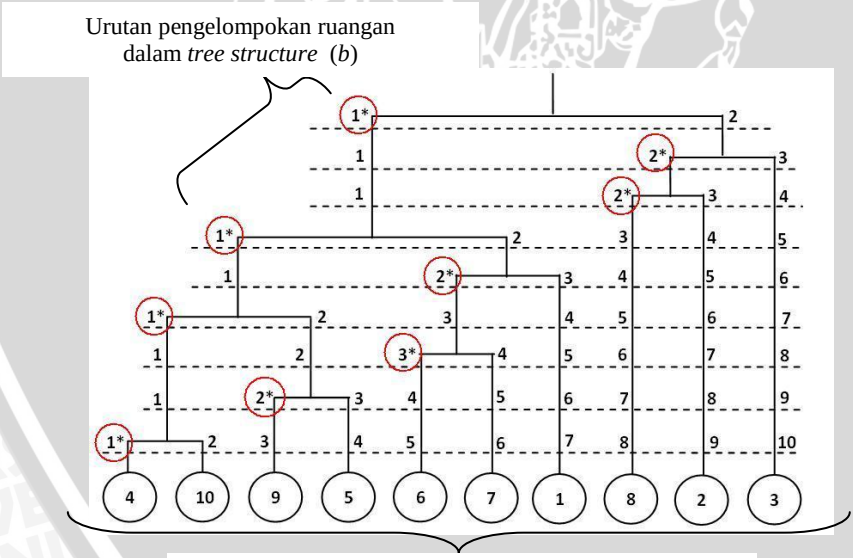
3.2.3 Representasi Kromosom

Pada penelitian ini representasi kromosom dilakukan berdasarkan *tree structure* yang terbentuk. Terdapat dua pengkodean yang digunakan, yaitu pengkodean permutasi pada gen pertama dan pengkodean nilai pada gen kedua. Pengkodean permutasi menunjukkan urutan ruangan yang jumlahnya sesuai kebutuhan ruangan yang dimasukkan sedangkan pengkodean nilai menunjukkan urutan pengelompokkan ruangan tersebut. Representasi kromosom dapat dirumuskan sebagai berikut:

$$(a_1\ a_2\ a_3\ \ldots\ldots\ a_n : b_1\ b_2\ b_3\ \ldots\ldots\ b_{n-2}) \qquad (3.1)$$

Kromosom ini berisi dua bagian string numerik, yaitu a_i, b_i dimana string a_i menunjukkan urutan ruangan dalam *tree structure*, string b_j menunjukkan urutan pengelompokan ruangan. Karakter ":" adalah sebuah pembatas. Nilai b_j dibatasi $1 \leq b_j \leq n\text{-}j$, $j = 1, 2, \ldots, n\text{-}2$.

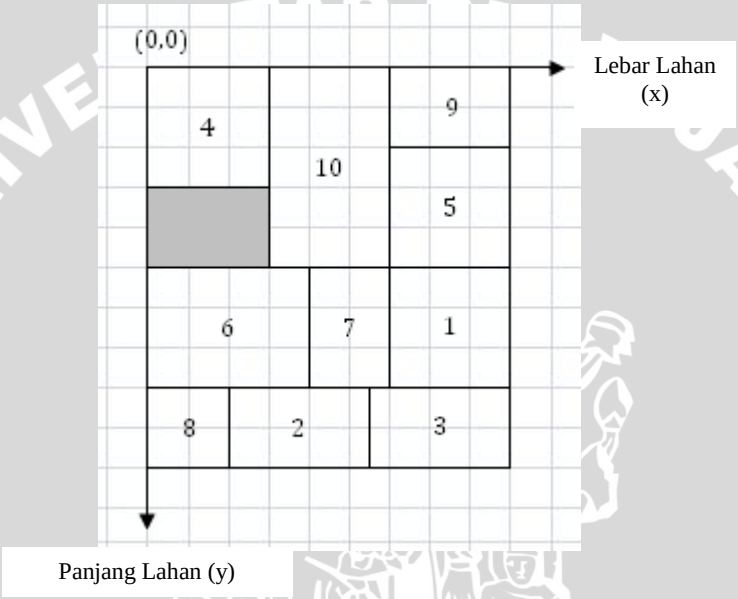
Misalkan terdapat representasi kromosom berikut: 4 10 9 5 6 7 1 8 2 3 : 1 2 3 1 2 1 2 2 yang diambil dari *tree structure* pada Gambar 3.3.



Gambar 3.3 Contoh Struktur Pohon Ruang dalam Rumah

Urutan gen pertama “4 10 9 5 6 7 1 8 2 3” sesuai urutan ruangan pada Gambar 3.3. Gen kedua “1 2 3 1 2 1 2 2” sesuai dengan angka yang terdapat tanda asterik (*), dimana menunjukkan urutan pengelompokan ruangan. Sedangkan pengelompokkan terakhir 1* tidak dimasukkan ke dalam gen kedua dari kromosom.

Dari Gambar 3.3, dapat dipetakan ke dalam denah atau tata letak ruang dalam rumah seperti Gambar 3.4 berikut:



Gambar 3.4 Contoh Tata Letak Ruang dalam Rumah

Pada Gambar 3.4 diasumsikan bahwa satu kotak mewakili satu meter ruangan. Lebar dan panjang kotak bernomor pada Gambar 3.4 sama dengan lebar dan panjang ruangan yang diinputkan oleh *user*. Penomoran diambil sesuai urutan ruangan yang diminta yang kemudian diacak pada penyusunan struktur pohon (*tree structure*) kromosom. Lebar dan panjang yang diinputkan *user* berdasarkan jenis ruangan yang telah diiputkan sebelumnya harus memenuhi spesifikasi ruangan sesuai Tabel 3.1 berikut:

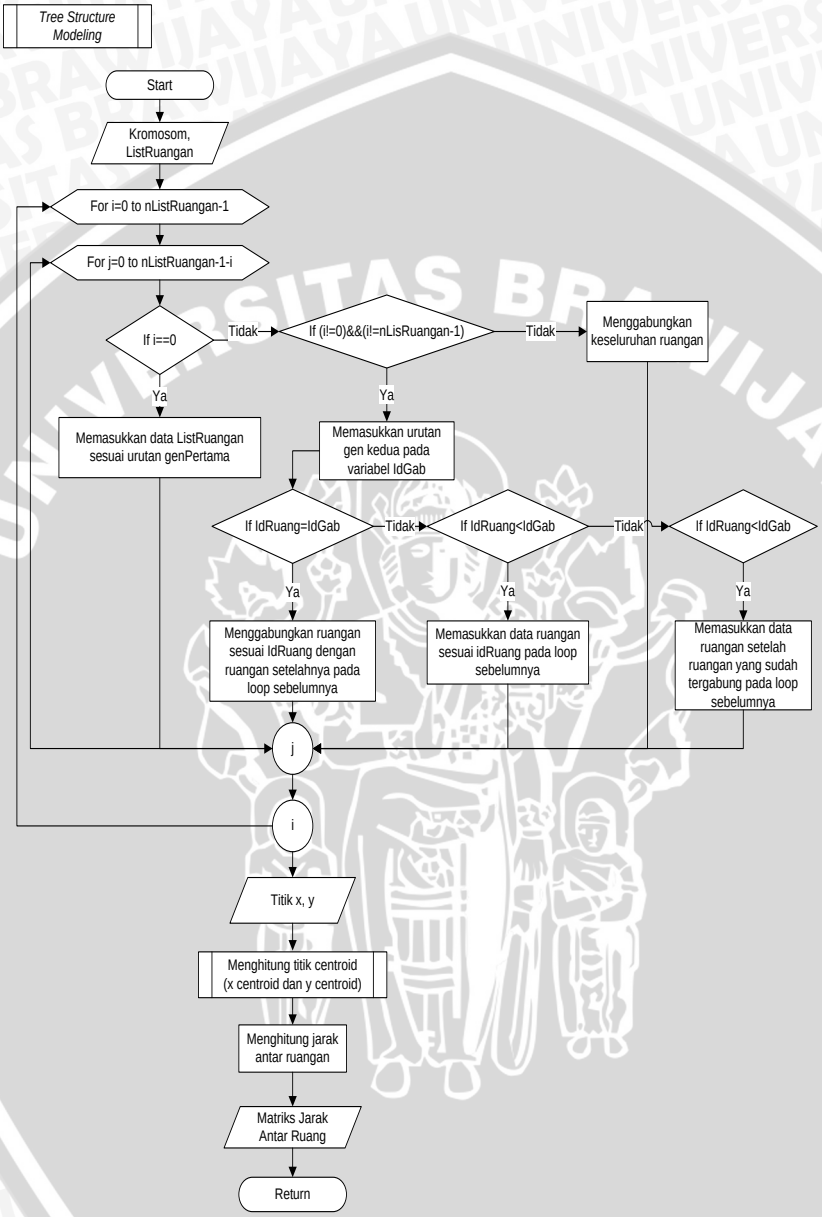
Tabel 3.1 Spesifikasi Ruangan

No.	Jenis Ruangan	Min. Area (m^2)	Maks. Area (m^2)	Min. Lebar (m)	Maks. Lebar (m)	Min. Panjang (m)	Maks. Panjang (m)
1.	Teras	2	9	1	3	1.5	3
2.	Ruang tamu	4	10.5	2	3	2	3.5
3.	Dapur	3	9	1	3	2	3
4.	Ruang keluarga	7.5	21	2.5	4	2.5	7
5.	Ruang makan	4.5	10.5	1.5	3	2.5	3.5
6.	Kamar tidur	7	18	2	3	3	6
7.	Kamar mandi	2,25	4.5	1,5	2	1.5	3
8.	Garasi	12	15	3	3	4	5
9.	Mushola	5	12	2	3	2	4
10.	Gudang	2,25	9	1,5	2	1.5	3

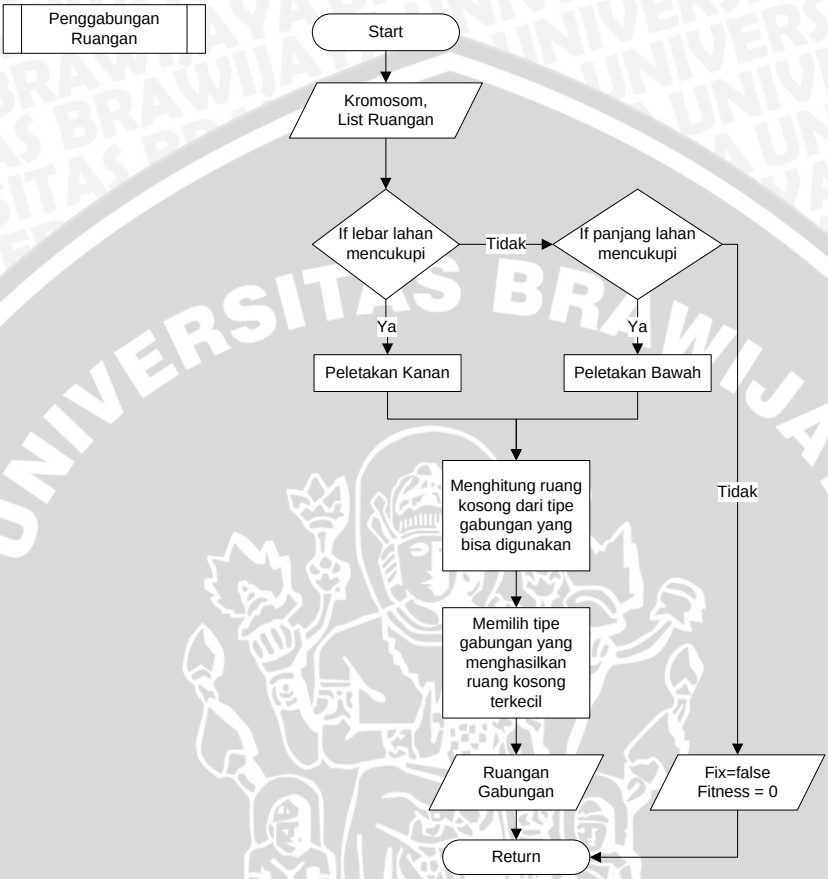
Pada Tabel 3.1, diasumsikan garasi untuk 1 mobil, ruang/kamar tambahan berupa tempat bersembahyang yaitu mushola. Spesifikasi tiap ruangan dalam rumah berbeda-beda tergantung pada kebutuhan pemilik dan dana yang tersedia. Tabel 3.1 di atas adalah ukuran standar yang biasa digunakan untuk rumah minimalis/rumah sederhana dengan luas lahan berkisar antara 60 m^2 sampai dengan 200 m^2.

3.2.4 *Tree Structure Modeling*

Hasil representasi kromosom dari *tree structure* yang telah terbentuk selanjutnya dipetakan ke dalam denah/tata letak ruang dalam rumah untuk mengetahui titik *centroid* tiap ruang. Dari titik *centroid* tersebut, dapat diperoleh jarak antar ruang yang digunakan dalam perhitungan *fitness*. Langkah-langkah proses *tree structure modeling* dapat dilihat pada Gambar 3.5 dan Gambar 3.6.



Gambar 3.5 *Flowchart* Proses *Tree Structure Modeling*



Gambar 3.6 *Flowchart* Proses Penggabungan Ruangan

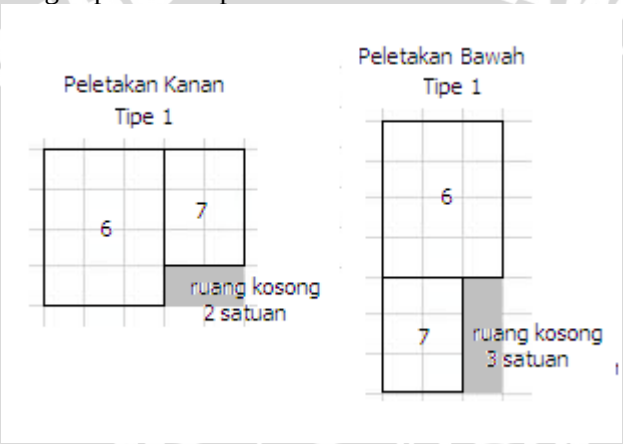
Tipe gabungan digunakan untuk menentukan kedua ruangan yang akan digabungkan mengalami rotasi atau tidak mengalami rotasi. Tipe gabungan yang digunakan dapat dilihat pada Tabel 3.2.

Tabel 3.2 Tipe Gabungan untuk Peletakan Ruangan

Tipe Gabungan	Ruangan1	Ruangan2
Tipe 1	Tidak mengalami rotasi	Tidak mengalami rotasi
Tipe 2	Tidak mengalami rotasi	Mengalami rotasi

Tipe 3	Mengalami rotasi	Tidak mengalami rotasi
Tipe 4	Mengalami rotasi	Mengalami rotasi

Dari keempat tipe tersebut dapat digunakan untuk menghitung ruang kosong. Ruang kosong adalah lahan yang tersisa dari proses penggabungan. Tipe penggabungan yang memiliki ruang kosong yang terkecil akan digunakan untuk proses penggabungan ruangan. Perhitungan ruang kosong tergantung pada peletakan yang dilakukan peletakan kanan atau peletakan bawah. Proses perhitungan ruang kosong dapat dilihat pada Gambar 3.7.



Gambar 3.7 Ilustrasi Perhitungan Ruang Kosong

3.2.5 Perhitungan *Fitness*

Facility Layout Problem (FLP) didefinisikan dalam bab dua sebelumnya mempunyai fungsi objektif yang berbeda tergantung tujuan mana yang ingin dicapai. Pada studi kasus desain tata letak ruang dalam rumah menggunakan rumus *total travel distance* (2.5) sebagai fungsi objektif dikarenakan dalam desain tata letak ruang dalam rumah menggunakan hubungan kedekatan antar ruangan bukan perpindahan aliran. Sehingga fungsi *fitness* yang digunakan sebagai berikut:

$$Fitness = \frac{A}{F(X) + e} \quad (3.2)$$

Dimana,

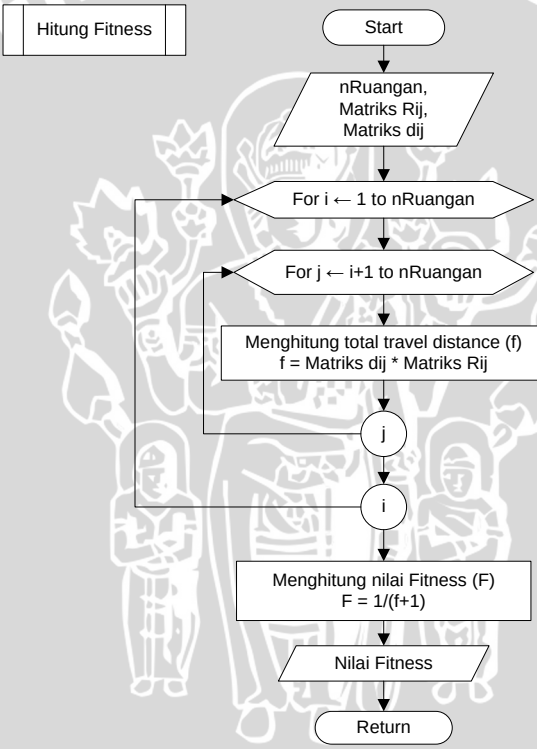
A = konstanta yang ditentukan, yaitu 1.

X = individu (kromosom).

e = bilangan kecil yang ditentukan untuk menghindari pembagi nol atau *F(X)* = 0, yaitu 1.

F(X) = fungsi objektif yang digunakan, yaitu: rumus *total travel distance*.

Flowchart proses hitung *fitness* ditunjukkan pada Gambar 3.8 berikut:



Gambar 3.8 *Flowchart* Proses Hitung *Fitness*

3.2.6 Seleksi

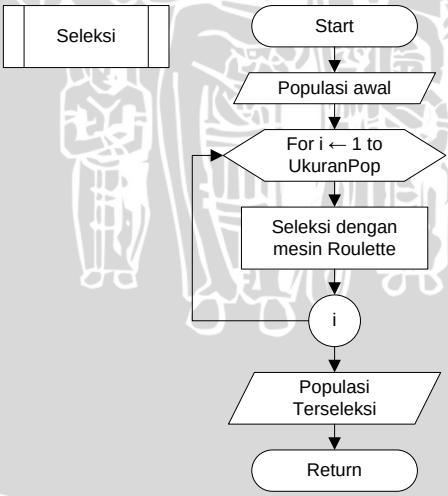
Sepasang kromosom diambil dari populasi menggunakan seleksi dengan mesin *roulette*. Probabilitas pemilihan kromosom *i* dihitung sebagai berikut:

$$P_i = \frac{F_i}{\sum_{j=1}^{PS} F_j} \quad (3.3)$$

PS adalah ukuran populasi. Sedangkan F_i adalah nilai fitness kromosom *i* dan F_j adalah total nilai fitness semua kromosom. *Flowchart* proses seleksi ditunjukkan pada Gambar 3.9.

Langkah-langkah seleksi dengan mesin *roulette* adalah sebagai berikut:

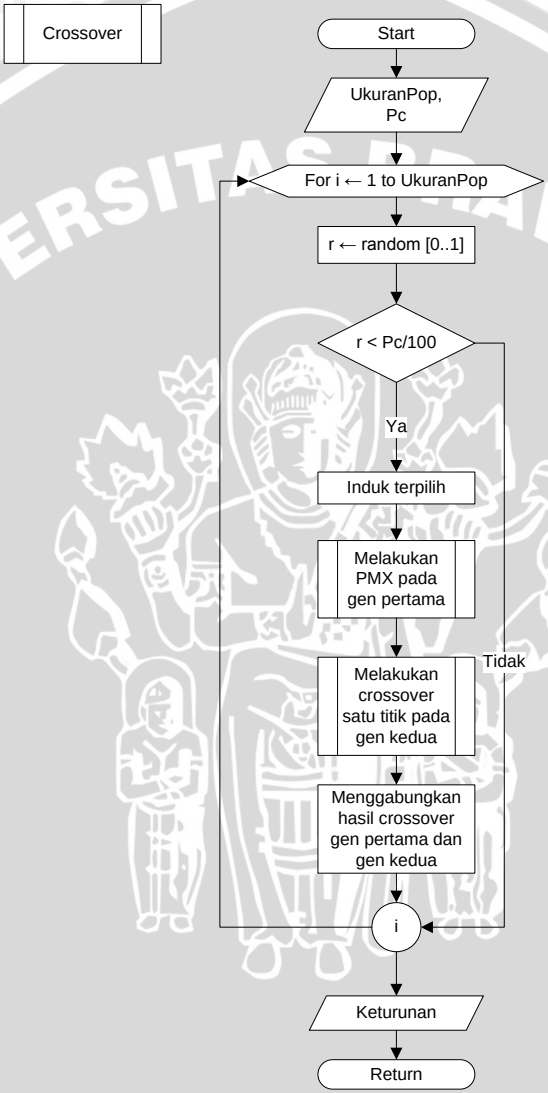
1. Menghitung nilai *fitness* dari masing-masing kromosom.
2. Menghitung total nilai *fitness* semua kromosom.
3. Menghitung probabilitas tiap kromosom sesuai rumus 3.3.
4. Menghitung probabilitas kumulatif masing-masing kromosom.
5. Menghitung range masing-masing kromosom pada angka 0 sampai 1 dari probabilitas yang dihasilkan.
6. Membangkitkan bilangan random antara 0 sampai 1.
7. Menentukan kromosom mana yang terpilih dalam proses seleksi dari bilangan random yang diperoleh.



Gambar 3.9 *Flowchart* Proses Seleksi

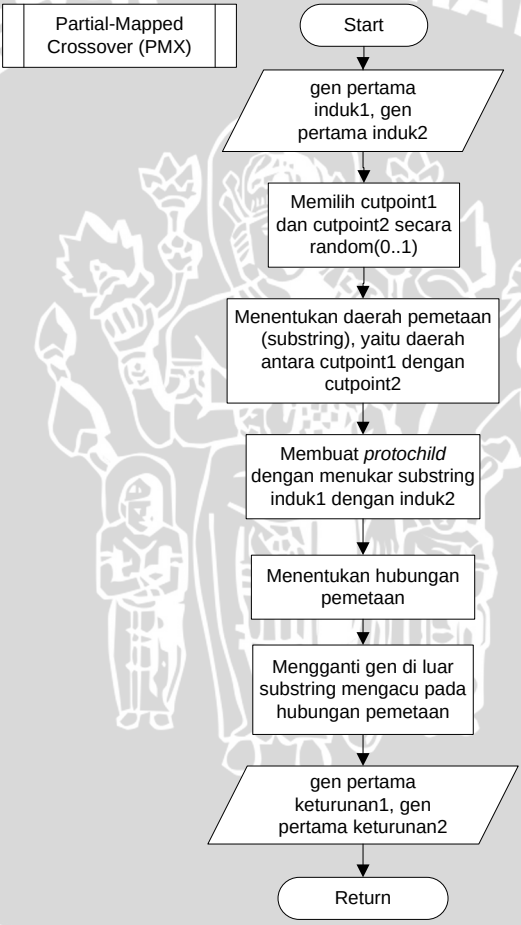
3.2.7 *Crossover*

Flowchart proses *crossover* ditunjukkan pada Gambar 3.10 berikut:

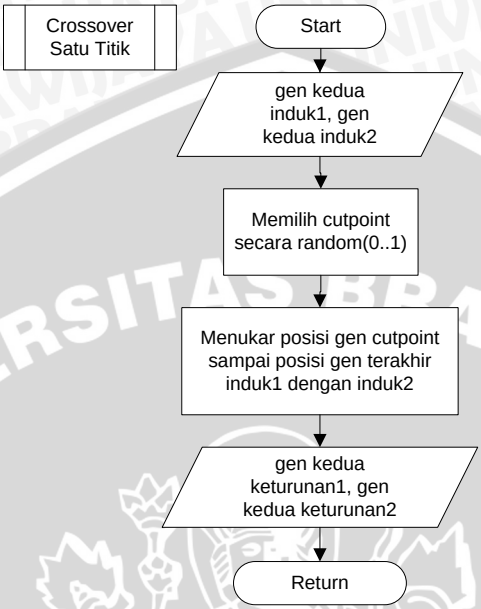


Gambar 3.9 *Flowchart* Proses *Crossover*

Pada gen pertama dari kromosom dilakukan *crossover* dengan metode *Partial-Mapped Crossover* (PMX). *Crossover* ini dilakukan agar tidak menghasilkan keturunan yang tidak layak. *Crossover* satu titik digunakan pada gen kedua berdasarkan *crossover* gen pertama. Tinggi peluang *crossover* dalam suatu populasi akan ditentukan oleh nilai probabilitas *crossover* (Pc). *Flowchart* proses *Partial-Mapped Crossover* (PMX) bisa dilihat pada Gambar 3.11. Sedangkan *flowchart* proses *crossover* satu titik bisa dilihat pada Gambar 3.12.



Gambar 3.11 *Flowchart* Proses PMX

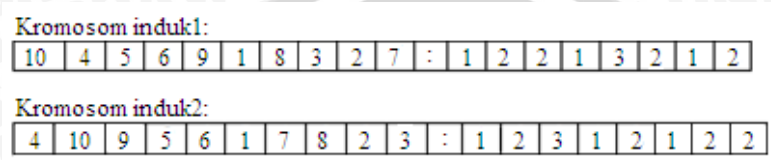


Gambar 3.12 *Flowchart* Proses *Crossover* Satu Titik

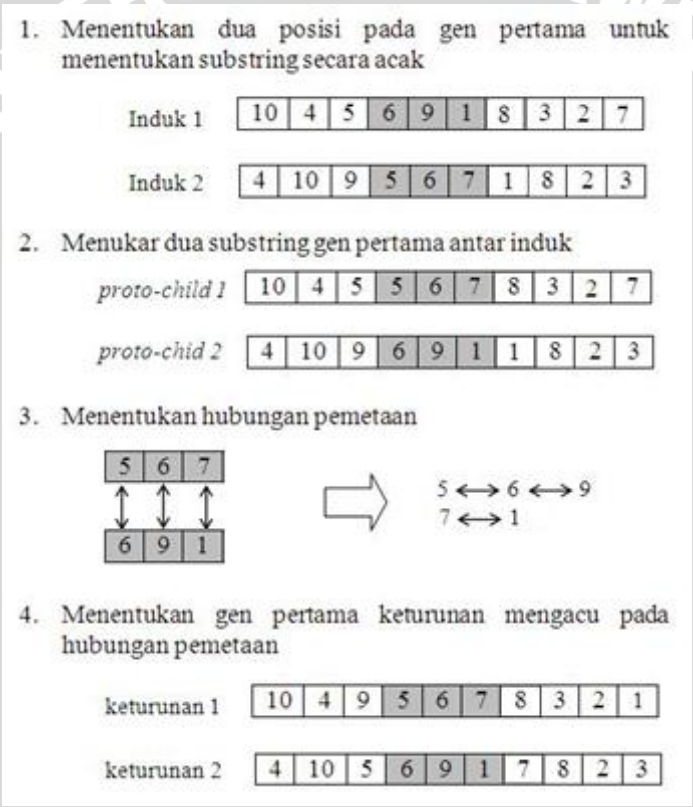
Keseluruhan langkah-langkah proses *crossover* dapat dijelaskan sebagai berikut:

1. Menentukan dua posisi pada gen pertama dari kromosom secara acak. Substring yang berada dalam dua posisi ini dinamakan daerah pemetaan.
2. Menukar dua substring gen pertama dari kromosom antar induk untuk menghasilkan *proto-child*.
3. Menentukan daerah pemetaan di antara dua daerah pemetaan pada gen pertama dari kromosom.
4. Menentukan kromosom keturunan mengacu pada hubungan pemetaan pada gen pertama dari kromosom.
5. Memilih satu titik tertentu sesuai kromosom yang mengalami perubahan gen pertama pada gen kedua dari kromosom tersebut, selanjutnya nilai sampai titik *crossover*-nya dari induk pertama digunakan dan sisanya dilanjutkan dengan nilai dari induk kedua dan sebaliknya.
6. Menggabungkan hasil kedua *crossover* tersebut.

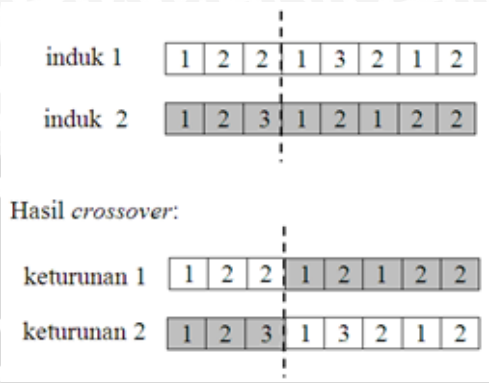
Contoh proses *crossover* dapat dilihat pada Gambar 3.13 sampai dengan Gambar 3.16 berikut:



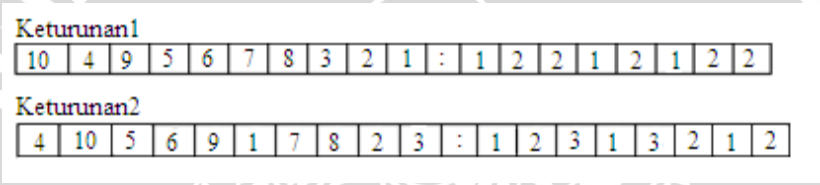
Gambar 3.13 Kromosom Sebelum Di-*crossover*



Gambar 3.14 Ilustrasi PMX pada Gen Pertama



Gambar 3.15 Ilustrasi *Crossover* Satu Titik pada Gen Kedua

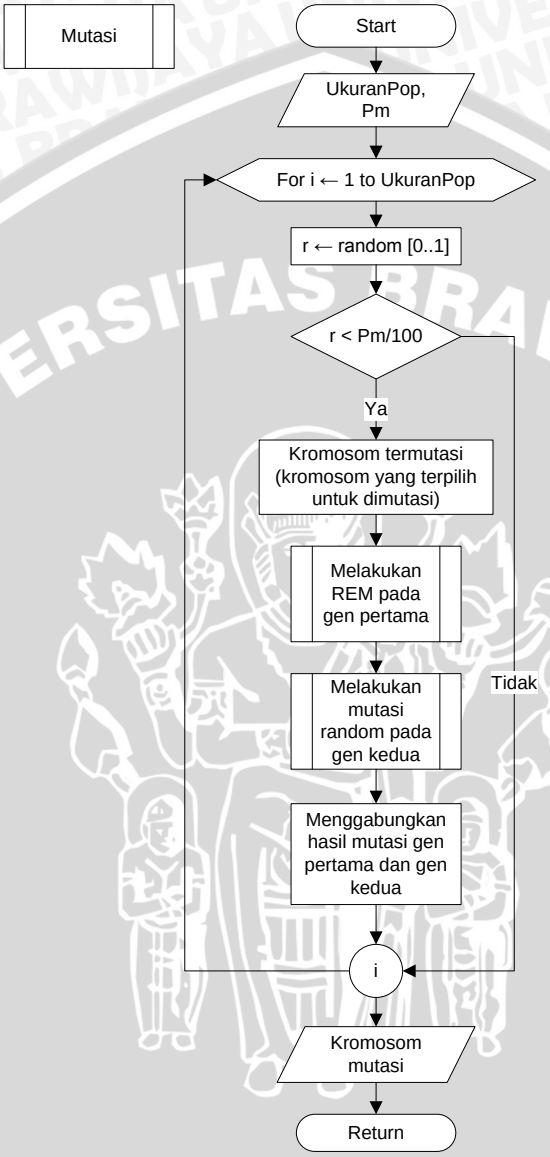


Gambar 3.16 Hasil Kromosom setelah Di-*crossover* dari Gen Pertama dan Gen Kedua

3.2.8 Mutasi

Mutasi ini dilakukan dengan mempertukarkan dua gen yang dipilih di bagian bekas keturunan secara acak, yaitu metode *Reciprocal Exchange Mutation* (REM). Mutasi untuk bagian akhir adalah nilai satu gen b_j yang dipilih secara random diubah berdasarkan $1 \leq b_j \leq n\text{-}j$. Tinggi peluang mutasi dalam suatu populasi akan ditentukan oleh nilai probabilitas mutasi (Pm).

Flowchart proses mutasi ditunjukkan pada Gambar 3.17 dan ilustrasi proses mutasi bisa dilihat pada Gambar 3.18.



Gambar 3.17 *Flowchart* Proses Mutasi

Kromosom sebelum dimutasi

4	10	9	5	6	7	1	8	2	3	:	1	2	3	1	2	1	2	2

Kromosom gen pertama setelah dimutasi

4	10	9	5	6	7	1	8	2	3	:	1	2	3	1	2	1	2	2

Jika urutan gen pertama berubah maka urutan gen kedua juga berubah

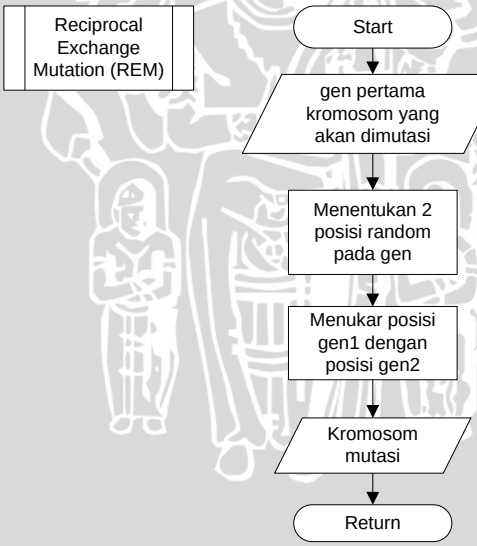
4	10	1	5	6	7	9	8	2	3	:	1	2	2	1	2	1	2	2

Kromosom setelah dimutasi

4	10	1	5	6	7	9	8	2	3	:	1	2	2	1	2	1	2	2

Gambar 3.18 Ilustrasi Proses Mutasi

Adapun *flowchart* proses *Reciprocal Exchange Mutation* (REM) pada Gambar 3.19 dan *flowchart* proses mutasi random pada Gambar 3.20.



Gambar 3.19 *Flowchart* Proses REM

Gambar 3.20 *Flowchart* Proses Mutasi Random

3.3 Perhitungan Manual

Contoh permasalahan yang digunakan pada desain tata letak ruang dalam rumah adalah sebagai berikut:

Seorang calon pemilik rumah ingin merencanakan tata letak ruang dalam rumah yang akan dibangun. Lahan yang dimiliki seluas 156 m^2 (lebar 9 m x panjang 17.3 m). Rencana rumah yang akan dibangun seluas 99 m^2 (lebar 9 m x panjang 11 m) dan sisanya adalah kebun di belakang dan sedikit taman di depan rumah. Adapun kebutuhan ruangan dalam rumah adalah sebagai berikut:

1. Kamar tidur 1 dengan lebar 3 m x panjang 3 m
2. Kamar tidur 2 dengan lebar 2 m x panjang 3.5m
3. Kamar tidur 3 dengan lebar 2 m x panjang 3.5m
4. Ruang tamu dengan lebar 3 m x panjang 3 m
5. Ruang makan dengan lebar 3 m x panjang 3 m
6. Ruang keluarga dengan lebar 3 m x panjang 4 m
7. Mushola dengan lebar 2 m x panjang 3 m
8. Kamar mandi dengan lebar 2 m x panjang 2 m
9. Dapur dengan lebar 2 m x panjang 3 m
10. Garasi dengan lebar 3 m x panjang 5 m

Sehingga jumlah ruangan yang akan dibangun ada 10 ruangan dengan tambahan kebun dan taman (diabaikan karena tidak termasuk ruang dalam rumah). Sedangkan nilai kedekatan antar ruang yang diberikan berdasarkan Tabel 2.1 misalkan seperti Tabel 3.3 berikut:

Tabel 3.3 Nilai Kedekatan Antar Ruang

Kebutuhan Ruangan	i / j	1	2	3	4	5	6	7	8	9	10
Kamar tidur 1	1	-									
Kamar tidur 2	2	O	-								
Kamar tidur 3	3	U	U	-							
Ruang tamu	4	X	X	U	-						
Ruang makan	5	U	U	O	O	-					
R. keluarga	6	I	I	O	I	E	-				
Mushola	7	U	U	U	U	I	I	-			
Kamar mandi	8	E	I	I	O	X	O	I	-		
Dapur	9	U	U	O	U	A	I	I	U	-	
Garasi	10	X	X	U	A	U	U	X	U	I	-

Keterangan:

A = Absolutely necessary, mutlak perlu ruangan-ruangan tersebut berdekatkan (berhampiran satu sama lain).
E = Especially important, sangat penting ruangan-ruangan tersebut berdekatan.
I = Important, penting ruangan-ruangan tersebut berdekatan.
O = Ordinary closeness, biasanya berdekatan, namun jika tidak berdekatan tidak masalah.
U = Unimportant, tidak perlu adanya keterkaitan kedekatan apapun.
X = Undesirable, tidak diinginkan ruangan-ruangan tersebut berdekatan.

3.3.1 Representasi Kromosom

Teknik pengkodean kromosom menggunakan pengkodean permutasi pada gen pertama dan pengkodean nilai pada gen kedua untuk menentukan struktur pohon (*tree structure*) yang terbentuk. Contoh representasi kromosom adalah sebagai berikut:

Kromosom 1 = 4 10 9 5 6 7 1 8 2 3 : 1 2 3 1 2 1 2 2
Kromosom 2 = 10 5 9 6 8 2 3 1 4 7 : 5 5 2 5 1 4 1 2

Kromosom 3 = 10 4 5 6 9 1 8 3 2 7 : 1 2 2 1 3 2 1 2
Kromosom 4 = 4 9 6 10 8 2 3 5 7 1 : 1 2 1 2 4 4 2 1

3.3.2 *Tree Structure Modeling*

Dari representasi kromosom, selanjutnya dilakukan proses *tree structure modeling*. Misalkan untuk kromosom 1 dapat dibentuk *tree structure modeling* seperti Gambar 3.3. Dengan langkah-langkah penggabungan ruangan sebagai berikut:

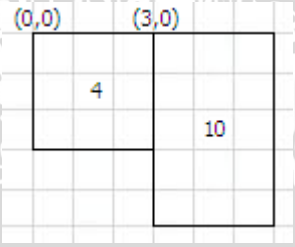
1. Memasukkan semua data kebutuhan ruangan sesuai urutan gen pertama, yaitu ruangan 4, ruangan 10, dan seterusnya hingga ruangan 1 dengan koordinat (0,0).
   Gen Pertama

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

   Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

2. Mengambil nilai gen kedua urutan pertama, yaitu 1, maka dilakukan proses penggabungan gen pertama dengan urutan ke-1 (sesuai isi gen kedua), yaitu ruangan 4 kemudian digabungkan dengan urutan setelahnya (urutan 2), yaitu ruangan 10 sehingga dihasilkan peletakan sebagai berikut:



Gambar 3.21 Tata Letak 1

Peletakan ini dipilih karena memiliki ruang kosong paling kecil dari tipe yang tersedia dengan peletakan kanan. Dilakukan peletakan kanan dikarenakan lebar lahan masih

mencukupi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 6 satuan yang merupakan hasil penjumlahan lebar ruangan 4 dan lebar ruangan 10, serta panjang 5 satuan yang merupakan nilai maksimal dari panjang ruangan 4 dan panjang ruangan 10 untuk peletakan kanan. Dari penggabungan tersebut diperoleh ruangan 4 dengan koordinat (0,0) dan ruangan 10 dengan koordinat (3,0).

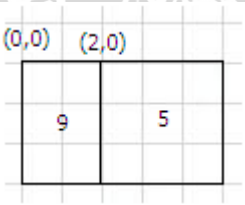
Gen Pertama

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

Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

3. Nilai gen kedua urutan selanjutnya, yaitu 2. Sehingga dilakukan penggabungan gen pertama dengan urutan ke-2, yaitu ruangan 9 digabungkan dengan urutan setelahnya, yaitu ruangan 5, sehingga dihasilkan peletakan sebagai berikut:



Gambar 3.22 Tata Letak 2

Peletakan ini dipilih karena nilai ruang kosong bernilai 0 dengan peletakan kanan. Dilakukan peletakan kanan dikarenakan lebar lahan masih mencukupi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 5 satuan yang merupakan hasil penjumlahan lebar ruangan 9 dan lebar ruangan 5, serta panjang 3 satuan yang merupakan nilai maksimal dari panjang ruangan 9 dan panjang ruangan 5 untuk peletakan kanan. Dari

penggabungan tersebut diperoleh ruangan 9 dengan koordinat (0,0) dan ruangan 5 dengan koordinat (2,0).

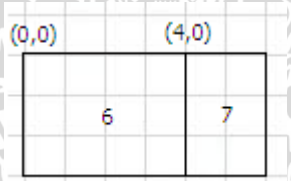
Gen Pertama

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

Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

4. Nilai gen kedua urutan selanjutnya, yaitu 3. Sehingga dilakukan penggabungan gen pertama dengan urutan ke-3, yaitu ruangan 6 digabungkan dengan urutan setelahnya, yaitu ruangan 7, sehingga dihasilkan peletakan sebagai berikut:



Gambar 3.23 Tata Letak 3

Peletakan ini dipilih karena nilai ruang kosong bernilai 0 dengan peletakan kanan. Dilakukan peletakan kanan dikarenakan lebar lahan masih mencukupi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 6 satuan yang merupakan hasil penjumlahan lebar ruangan 6 (lebar hasil rotasi) dan lebar ruangan 7, serta panjang 3 satuan yang merupakan nilai maksimal dari panjang ruangan 6 (panjang hasil rotasi) dan panjang ruangan 7 untuk peletakan kanan. Ruangan 6 mengalami rotasi sehingga lebar dan panjang ruangan ditukar. Dari penggabungan tersebut diperoleh ruangan 6 dengan koordinat (0,0) dan ruangan 7 dengan koordinat (4,0).

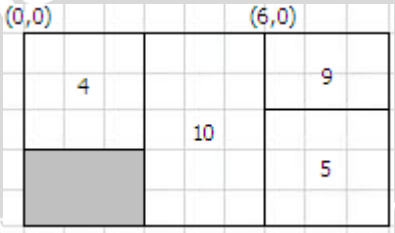
Gen Pertama

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

Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

5. Nilai gen kedua urutan selanjutnya, yaitu 1. Sehingga dilakukan penggabungan gen pertama dengan urutan ke-1, yaitu ruangan [4,10] digabungkan dengan urutan setelahnya, yaitu ruangan [9,5], sehingga dihasilkan peletakan sebagai berikut:



Gambar 3.24 Tata Letak 4

Peletakan ini dipilih karena nilai ruang kosong bernilai 0 dengan peletakan kanan. Dilakukan peletakan kanan dikarenakan lebar lahan masih mencukupi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 9 satuan yang merupakan hasil penjumlahan lebar ruangan [4,10] dan lebar ruangan [9,5] (lebar hasil rotasi), serta panjang 5 satuan yang merupakan nilai maksimal dari panjang ruangan [4,10] dan panjang ruangan [9,5] (panjang hasil rotasi) untuk peletakan kanan. Dari penggabungan tersebut diperoleh ruangan [4,10] dengan koordinat (0,0) dan ruangan [9,5] dengan koordinat (6,0).

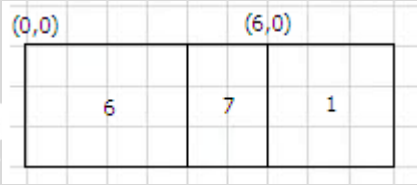
Gen Pertama

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

Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

6. Nilai gen kedua urutan selanjutnya, yaitu 2. Sehingga dilakukan penggabungan gen pertama dengan urutan ke-2, yaitu ruangan [6,7] digabungkan dengan urutan setelahnya, yaitu ruangan 1, sehingga dihasilkan peletakan sebagai berikut:



Gambar 3.25 Tata Letak 5

Peletakan ini dipilih karena nilai ruang kosong bernilai 0 dengan peletakan kanan. Dilakukan peletakan kanan dikarenakan lebar lahan masih mencukupi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 9 satuan yang merupakan hasil penjumlahan lebar ruangan [6,7] dan lebar ruangan 1, serta panjang 3 satuan yang merupakan nilai maksimal dari panjang ruangan [6,7] dan panjang ruangan 1 untuk peletakan kanan. Dari penggabungan tersebut diperoleh ruangan [6,7] dengan koordinat (0,0) dan ruangan 1 dengan koordinat (6,0).

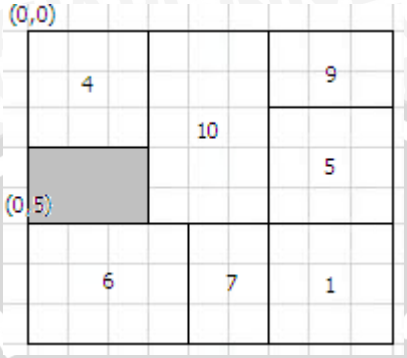
Gen Pertama

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

Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

7. Nilai gen kedua urutan selanjutnya, yaitu 1. Sehingga dilakukan penggabungan gen pertama dengan urutan ke-1, yaitu ruangan [[4,10],[9,5]] digabungkan dengan urutan setelahnya, yaitu ruangan [[6,7],1], sehingga dihasilkan peletakan sebagai berikut:



Gambar 3.26 Tata Letak 6

Peletakan ini dipilih karena nilai ruang kosong bernilai 0 dengan peletakan bawah. Dilakukan peletakan bawah dikarenakan lebar lahan tidak mencukupi. Ruangan [[4,10],[9,5]] tidak dapat mengalami rotasi karena terdapat ruangan dengan jenis ruangan garasi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 9 satuan yang merupakan nilai maksimal dari lebar ruangan [[4,10],[9,5]] dan lebar ruangan [[6,7],1], serta panjang 8 satuan yang merupakan hasil penjumlahan panjang ruangan [[4,10],[9,5]] dan panjang ruangan [[6,7],1] untuk peletakan bawah. Dari penggabungan tersebut diperoleh ruangan [[4,10],[9,5]] dengan koordinat (0,0) dan ruangan [[6,7],1] dengan koordinat (0,5).

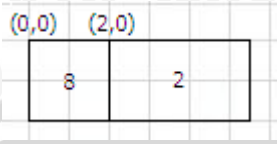
Gen Pertama

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

Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

8. Nilai gen kedua urutan selanjutnya, yaitu 2. Sehingga dilakukan penggabungan gen pertama dengan urutan ke-2, yaitu ruangan 8 digabungkan dengan urutan setelahnya, yaitu ruangan 2, sehingga dihasilkan peletakan sebagai berikut:



Gambar 3.27 Tata Letak 7

Peletakan ini dipilih karena nilai ruang kosong bernilai 0 dengan peletakan kanan. Dilakukan peletakan kanan dikarenakan lebar lahan masih mencukupi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 5,5 satuan yang merupakan hasil penjumlahan lebar ruangan 8 dan lebar ruangan 2 (lebar hasil rotasi), serta panjang 2 satuan yang merupakan nilai maksimal dari panjang ruangan 8 dan panjang ruangan 2 (panjang hasil rotasi) untuk peletakan kanan. Dari penggabungan tersebut diperoleh ruangan 8 dengan koordinat (0,0) dan ruangan 2 dengan koordinat (2,0).

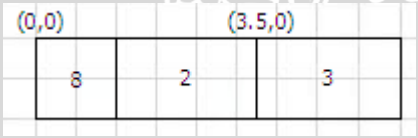
Gen Pertama

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

Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

9. Nilai gen kedua urutan selanjutnya, yaitu 2. Sehingga dilakukan penggabungan gen pertama dengan urutan ke-2, yaitu ruangan [8,2] digabungkan dengan urutan setelahnya, yaitu ruangan 3, sehingga dihasilkan peletakan sebagai berikut:



Gambar 3.28 Tata Letak 8

Peletakan ini dipilih karena nilai ruang kosong bernilai 0 dengan peletakan kanan. Dilakukan peletakan kanan dikarenakan lebar lahan masih mencukupi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 9 satuan yang merupakan hasil penjumlahan lebar ruangan [8,2] dan lebar ruangan 3 (lebar hasil rotasi), serta panjang 2 satuan yang merupakan nilai maksimal dari panjang ruangan [8,2] dan panjang ruangan 3 (panjang hasil rotasi) untuk peletakan kanan. Dari penggabungan tersebut diperoleh ruangan [8,2] dengan koordinat (0,0) dan ruangan 3 dengan koordinat (3.5,0).

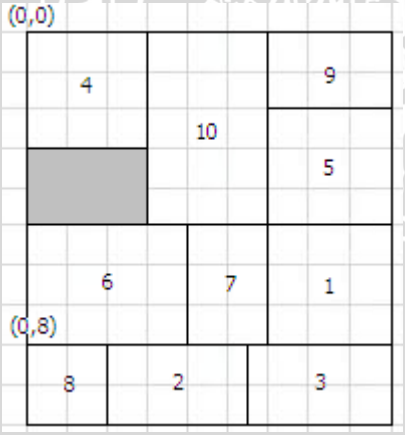
Gen Pertama

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

Gen Kedua

1	2	3	4	5	6	7	8
1	2	3	1	2	1	2	2

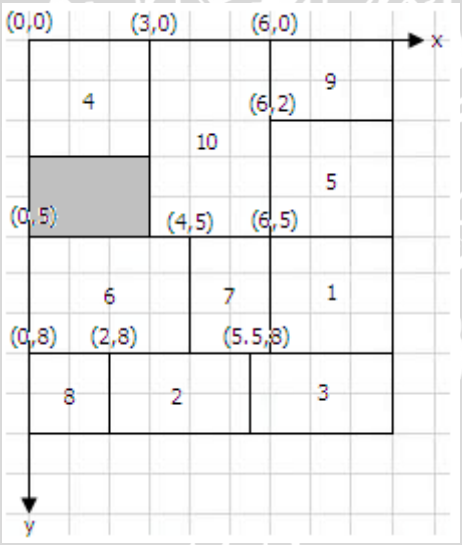
10. Menggabungkan keseluruhan ruangan, yaitu ruangan [[[4,10],[9,5]], [[6,7],1]] dan ruangan [[8,2],3]. Peletakan yang dihasilkan sebagai berikut:



Gambar 3.29 Tata Letak 9

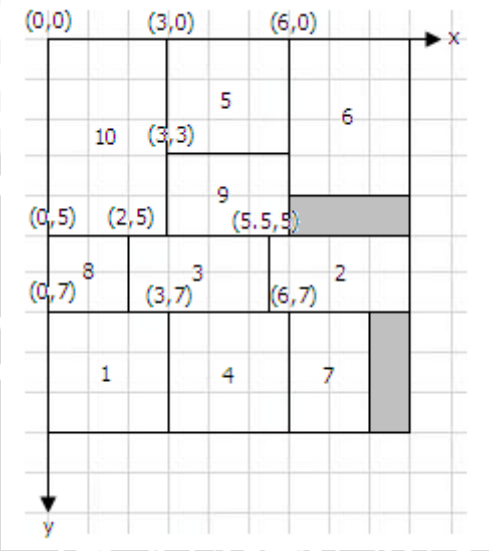
Peletakan ini dipilih karena nilai ruang kosong bernilai 0 dengan peletakan bawah. Dilakukan peletakan kanan dikarenakan lebar lahan tidak mencukupi dan panjang lahan masih mencukupi. Ruangan hasil penggabungan selalu diletakkan pada koordinat (0,0) dengan lebar 9 satuan yang merupakan nilai maksimal dari lebar ruangan [[[4,10],[9,5]], [[6,7],1]] dan lebar ruangan [[8,2],3], serta panjang 10 satuan yang merupakan penjumlahan dari panjang ruangan [[[4,10],[9,5]], [[6,7],1]] dan panjang ruangan [[8,2],3] untuk peletakan bawah. Dari penggabungan tersebut diperoleh ruangan [[[4,10],[9,5]], [[6,7],1]] dengan koordinat (0,0) dan ruangan [[8,2],3] dengan koordinat (0,8).

Berdasarkan peletakan terakhir diperoleh pemetaan sebagai berikut:

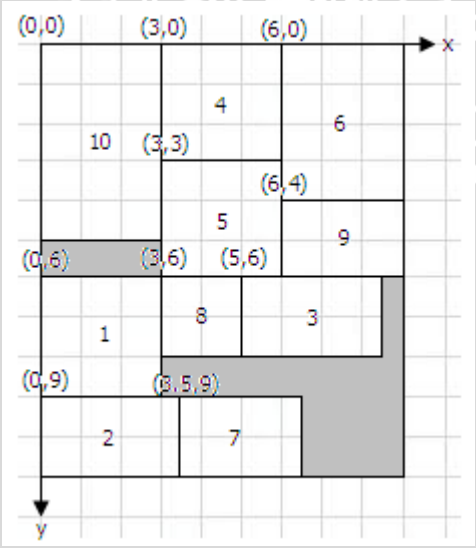


Gambar 3.30 Tata Letak Ruang Kromosom 1

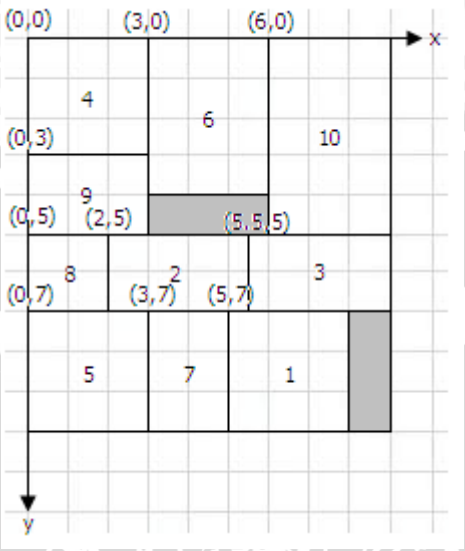
Dengan cara yang sama, diperoleh pemetaan Kromosom2, Kromosom3, dan Kromosom4 sebagai berikut:



Gambar 3.31 Tata Letak Ruang Kromosom 2



Gambar 3.32 Tata Letak Kromosom 3



Gambar 3.33 Tata Letak Kromosom 4

Pada Kromosom1, diketahui x-centroid, y-centroid masing-masing ruangan sebagai berikut: Ruangan 1 (7.5, 6.5), Ruangan 2 (3.75, 9), Ruangan 3 (7.25, 9), Ruangan 4 (1.5, 1.5), Ruangan 5 (7.5, 3.5), Ruangan 6 (2, 6.5), Ruangan 7 (5, 6.5), Ruangan 8 (1, 9), Ruangan 9 (7.5, 1), dan Ruangan 10 (4.5, 2.5).

Untuk menghitung jarak antar ruangan digunakan rumus 2.7 sehingga didapat jarak sebagai berikut:

$$d_{1,2} = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

$$= \sqrt{(7.5 - 3.75)^2 + (6.5 - 9)^2} = 4.507$$

Dengan cara yang sama diperoleh jarak antar ruangan yang lainnya sebagai berikut:

Tabel 3.4 Matriks Jarak Antar Ruang Kromosom1

i / j	1	2	3	4	5	6	7	8	9	10
1	-									
2	4.507	-								
3	2.512	3.5	-							
4	7.81	7.83	9.451	-						
5	3	6.657	5.506	6.325	-					
6	5.5	3.052	5.815	5.025	6.265	-				
7	2.5	2.795	3.363	6.103	3.905	3	-			
8	6.964	2.75	6.25	7.517	8.515	2.693	4.717	-		
9	5.5	8.835	8.004	6.021	2.5	7.778	6.042	10.31	-	
10	5	6.543	7.058	3.162	3.162	4.717	4.031	7.382	3.354	-

Pada Kromosom2, diketahui x-centroid, y-centroid masing-masing ruangan berdasarkan *tree structure modeling* yang terbentuk sebagai berikut: Ruangan 1 (1.5, 8.5), Ruangan 2 (7.25, 6.5), Ruangan 3 (3.75, 6), Ruangan 4 (4.5, 8.5), Ruangan 5 (4.5, 1.5), Ruangan 6 (7.5, 2), Ruangan 7 (7, 8.5), Ruangan 8 (1, 6), Ruangan 9 (4.5, 4), dan Ruangan 10 (1.5, 2.5). Dengan cara yang sama, sehingga didapat matriks jarak antar ruangan sebagai berikut:

Tabel 3.5 Matriks Jarak Antar Ruang Kromosom 2

i / j	1	2	3	4	5	6	7	8	9	10
1	-									
2	6.088	-								
3	3.363	3.536	-							
4	3	3.4	2.61	-						
5	7.616	5.706	4.562	7	-					
6	8.846	4.507	5.483	7.159	3.041	-				
7	5.5	2.016	4.1	2.5	7.433	6.519	-			
8	2.55	6.27	2.75	4.301	5.701	7.632	6.5	-		
9	5.408	3.717	2.136	4.5	2.5	3.606	5.148	4.031	-	
10	6	7.004	4.161	6.708	3.162	6.021	8.139	3.536	3.354	-

Pada Kromosom3, diketahui x-centroid, y-centroid masing-masing ruangan sebagai berikut: Ruangan 1 (1.5, 7.5), Ruangan 2 (1.75, 10), Ruangan 3 (6.75, 7), Ruangan 4 (4.5, 1.5), Ruangan 5 (4.5, 4.5), Ruangan 6 (7.5, 2), Ruangan 7 (5, 10), Ruangan 8 (4, 7), Ruangan 9 (7.5, 5), dan Ruangan 10 (1.5, 2.5). Dengan cara yang sama, sehingga didapat matriks jarak antar ruangan sebagai berikut:

Tabel 3.6 Matriks Jarak Antar Ruang Kromosom 3

i / j	1	2	3	4	5	6	7	8	9	10
1	-									
2	2.512	-								
3	5.274	5.831	-							
4	6.708	8.934	5.942	-						
5	4.243	6.149	3.363	3	-					
6	8.139	9.852	5.056	3.041	3.905	-				
7	4.301	3.25	3.473	8.515	5.523	8.382	-			
8	2.55	3.75	2.75	5.523	2.55	6.103	3.162	-		
9	6.5	7.62	2.136	4.61	3.041	3	5.59	4.031	-	
10	5	7.504	6.915	3.162	3.606	6.021	8.276	5.148	6.5	-

Pada Kromosom4, diketahui x-centroid, y-centroid masing-masing ruangan sebagai berikut: Ruangan 1 (6.5, 8.5), Ruangan 2 (3.75, 6), Ruangan 3 (7.25, 6.5), Ruangan 4 (1.5, 1.5), Ruangan 5 (1.5, 8.5), Ruangan 6 (4.5, 2), Ruangan 7 (4, 8.5), Ruangan 8 (1, 6), Ruangan 9 (1.5, 4), dan Ruangan 10 (7.5, 2.5). Dengan cara yang sama, sehingga didapat matriks jarak antar ruangan sebagai berikut:

Tabel 3.7 Matriks Jarak Antar Ruang Kromosom 4

i / j	1	2	3	4	5	6	7	8	9	10
1	-									
2	3.717	-								
3	2.136	3.536	-							
4	8.602	5.031	7.62	-						
5	5	3.363	6.088	7	-					

6	6.801	4.07	5.274	3.041	7.159	-				
7	2.5	2.512	3.816	7.433	2.5	6.519	-			
8	6.042	2.75	6.27	4.528	2.55	5.315	3.905	-		
9	6.727	3.01	6.27	2.5	4.5	3.606	5.148	2.062	-	
10	6.083	5.13	4.008	6.083	8.485	3.041	6.946	7.382	6.185	-

3.3.3 Nilai *Fitness*

Dalam menghitung nilai *fitness* tiap kromosom ada beberapa langkah sebagai berikut:

1. Menghitung total *travel distance* (*f*).
2. Menghitung nilai *fitness* (*F*).

Misalkan menghitung nilai *fitness* kromosom 1 sampai dengan kromosom 4 adalah sebagai berikut:

Langkah 1 - menghitung total *travel distance*

Untuk menghitung total *travel distance* digunakan rumus 2.5, yaitu total penjumlahan dari matriks jarak antar ruangan diperoleh dari proses *tree structure modeling* dikali dengan matriks kedekatan antar ruang yang diberikan, sehingga didapatkan total *travel distance* berikut:

$$
\begin{aligned}
f_{kromosom1} = {} & (d_{1,2} * R_{1,2}) + (d_{1,3} * R_{1,3}) + (d_{1,4} * R_{1,4}) + (d_{1,5} * R_{1,5}) + \\
& (d_{1,6} * R_{1,6}) + (d_{1,7} * R_{1,7}) + (d_{1,8} * R_{1,8}) + (d_{1,9} * R_{1,9}) + \\
& (d_{1,10} * R_{1,10}) + (d_{2,3} * R_{2,3}) + (d_{2,4} * R_{2,4}) + (d_{2,5} * R_{2,5}) + \\
& (d_{2,6} * R_{2,6}) + (d_{2,7} * R_{2,7}) + (d_{2,8} * R_{2,8}) + (d_{2,9} * R_{2,9}) + \\
& (d_{2,10} * R_{2,10}) + (d_{3,4} * R_{3,4}) + (d_{3,5} * R_{3,5}) + (d_{3,6} * R_{3,6}) + \\
& (d_{3,7} * R_{3,7}) + (d_{3,8} * R_{3,8}) + (d_{3,9} * R_{3,9}) + (d_{3,10} * R_{3,10}) + \\
& (d_{4,5} * R_{4,5}) + (d_{4,6} * R_{4,6}) + (d_{4,7} * R_{4,7}) + (d_{4,8} * R_{4,8}) + \\
& (d_{4,9} * R_{4,9}) + (d_{4,10} * R_{4,10}) + (d_{5,6} * R_{5,6}) + (d_{5,7} * R_{5,7}) + \\
& (d_{5,8} * R_{5,8}) + (d_{5,9} * R_{5,9}) + (d_{5,10} * R_{5,10}) + (d_{6,7} * R_{6,7}) + \\
& (d_{6,8} * R_{6,8}) + (d_{6,9} * R_{6,9}) + (d_{6,10} * R_{6,10}) + (d_{7,8} * R_{7,8}) + \\
& (d_{7,9} * R_{7,9}) + (d_{7,10} * R_{7,10}) + (d_{8,9} * R_{8,9}) + (d_{8,10} * R_{8,10}) + \\
& (d_{9,10} * R_{9,10}) \\
= {} & 78{,}140.61
\end{aligned}
$$

Dengan rumus yang sama diperoleh total *travel distance* dari kromosom yang lainnya sebagai berikut:

$f_{kromosom2}$ = 73,469.7
$f_{kromosom3}$ = 114,565.11
$f_{kromosom4}$ = 95,578.9

Langkah 2 - menghitung nilai *fitness*

Untuk menghitung nilai *fitness* digunakan rumus 3.2 sehingga didapatkan nilai *fitness* berikut:

$$F_{kromosom1} = \frac{1}{78{,}140.61+1} = 1.27973 \times 10^{-5} = 0.0000128$$

Dengan rumus yang sama diperoleh nilai *fitness* dari kromosom yang lainnya sebagai berikut:

$F_{kromosom2}$ = 1.36109×10^{-5} = 0.0000136
$F_{kromosom3}$ = 8.72858×10^{-6} = 0.0000087
$F_{kromosom4}$ = 1.04625×10^{-5} = 0.0000105

3.3.4 Seleksi

Kromosom yang terkumpul dan terbentuk dari populasi awal akan diseleksi dengan menggunakan metode mesin *roulette*. Dengan seleksi ini, maka kromosom/individu yang memiliki nilai *fitness* yang besar akan mempunyai peluang yang lebih besar untuk terpilih.

Tabel 3.8 Nilai *Fitness* Populasi Awal

Kromosom	Fitness	Probabilitas	Probabilitas Kumulatif	Range
Kromosom1	0.0000128	0.280647131	0.281	0-0.281
Kromosom2	0.0000136	0.298489298	0.579	0.282-0.579
Kromosom3	0.0000087	0.191419758	0.771	0.580-0.771
Kromosom4	0.0000105	0.229443813	1	0.772-1

Langkah berikutnya adalah membangkitkan bilangan random dari 0 sampai dengan 1 sebanyak jumlah populasi awal yaitu 4 bilangan. Bilangan random yang diperoleh adalah: 0.58, 0.45, 0.08, dan 0.75 maka individu-individu yang terpilih untuk melanjutkan

proses selanjutnya adalah kromosom 3, kromosom 2, kromosom 1, dan kromosom 3. Akhirnya diperoleh sebagai berikut:
Kromosom 1' = 10 4 5 6 9 1 8 3 2 7 : 1 2 2 1 3 2 1 2 (Kromosom 3)
Kromosom 2' = 10 5 9 6 8 2 3 1 4 7 : 5 5 2 5 1 4 1 2 (Kromosom 2)
Kromosom 3' = 4 10 9 5 6 7 1 8 2 3 : 1 2 3 1 2 1 2 2 (Kromosom 1)
Kromosom 4' = 10 4 5 6 9 1 8 3 2 7 : 1 2 2 1 3 2 1 2 (Kromosom 3)

3.3.5 Crossover

Proses selanjutnya adalah *crossover*. Metode yang digunakan adalah *Partial-Mapped Crossover* (PMX) pada gen pertama dan *crossover* satu titik pada gen kedua. Proses *crossover* tergantung pada probabilitas *crossover*. Misalkan probabilitas *crossover* adalah 50%, artinya diharapkan 50% kromosom yang akan mengalami *crossover*.

Langkah pertama yang harus dilakukan adalah membangkitkan nilai random antara 0 sampai 1. Jika nilai random < 0,5 maka kromosom tersebut yang akan dikenai proses crossover. Misalkan nilai random yang didapat adalah 0.44, 0.60 , 0.19, dan 0.93. Dari nilai random yang didapat, maka diperoleh 2 kromosom yang akan mengalami proses crossover, yaitu kromosom 1' dan kromosom 3'.

Proses *crossover* yang terjadi seperti pada Gambar 3.11 sampai dengan Gambar 3.14 pada subbab sebelumnya dengan mengambil kromosom 1' dan kromosom 3' (dari hasil seleksi) berturut-turut sebagai induk 1 dan induk 2. Kromosom keturunan yang dihasilkan misalkan sebagai berikut:
Kromosom keturunan 1= 4 10 5 6 9 1 2 8 7 3 : 1 2 3 1 3 2 1 2
Kromosom keturunan 2= 10 4 9 5 6 7 1 3 8 2 : 1 2 2 1 2 1 2 2

Pada kromosom keturunan 1 dan kromosom keturunan 2, saat dilakukan pemetaan ternyata melebihi batas ketersediaan lahan sehingga nilai *fitness* dianggap 0.

3.3.6 Mutasi

Proses selanjutnya adalah mutasi. Metode mutasi yang digunakan *Reciprocal Exchange Mutation* (REM). Mutasi pada gen kedua adalah mutasi random dinama nilai gen yang diubah

berdasarkan $1 \leq b_j \leq n\text{-}j$. Mutasi dilakukan berdasarkan pada probabilitas mutasi yang telah ditentukan. Misalkan probabilitas mutasi adalah 25%, maka diharapkan 25% kromosom yang akan mengalami mutasi.

Langkah pertama yang harus dilakukan adalah membangkitkan nilai random antara 0 sampai 1. Jika nilai random < 0,25 maka kromosom tersebut yang akan dikenai proses mutasi. Misalkan nilai random yang didapat adalah 0.44, 0.92 , 0.19, dan 0.63. Dari nilai random yang didapat, maka diperoleh 1 kromosom yang akan mengalami proses mutasi, yaitu kromosom 3'.

Proses mutasi dapat dilihat pada Gambar 3.16 pada subbab sebelumnya. Kromosom mutasi yang dihasilkan berasal dari kromosom 3' yang diambil secara acak, sebagai berikut:
Kromosom mutasi = 4 10 1 5 6 7 9 8 2 3 : 1 2 2 1 2 1 2 2

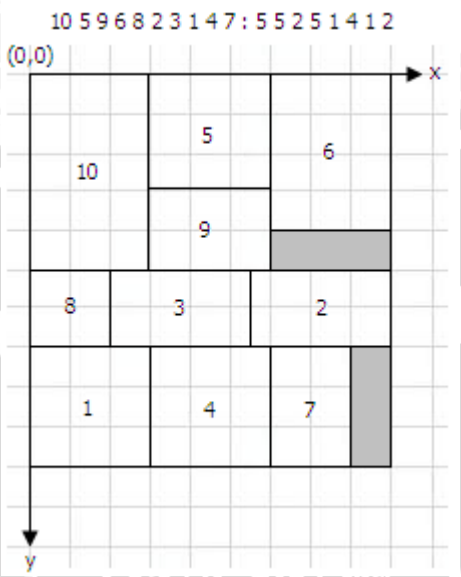
Pada kromosom mutasi, saat dilakukan pemetaan ternyata melebihi batas ketersediaan lahan sehingga nilai *fitness*-nya dianggap 0.

Selanjutnya dilakukan pengurutan kromosom berdasarkan nilai *fitness*-nya. Kromosom yang diambil untuk generasi selanjutnya sebanyak ukuran populasi, dimulai dari nilai *fitness* yang terbesar. Sehingga nilai *fitness* terkecil akan tereliminasi. Berikut tabel nilai *fitness* untuk generasi ke-1:

Tabel 3.9 Nilai *Fitness* Generasi ke-1

Kromosom	*Fitness*
Kromosom 2'	0.0000136
Kromosom 3'	0.0000128
Kromosom 1'	0.0000087
Kromosom 4'	0.0000087

Desain yang ditampilkan adalah desain yang memiliki nilai *fitness* terbesar. Berikut adalah desain tata letak yang akan ditampilkan berdasarkan nilai *fitness* terbesar pada Tabel 3.9. Jika maksimal generasi 1 iterasi, maka kromosom yang terpilih adalah kromosom yang memiliki nilai *fitness* terbesar pada generasi tersebut, yaitu kromosom 2'.



Gambar 3.34 Desain Tata Letak Ruang yang Terpilih

3.4 Perancangan Uji Coba

Perancangan uji coba ini dilakukan untuk mengetahui seberapa efektifnya *tree structure modeling* dan algoritma genetika dalam menyelesaikan permasalahan *unequal facility layout problem* dengan studi kasus desain tata letak ruang dalam rumah. Berikut adalah beberapa parameter untuk menguji keefektifan *tree structure modeling* dan algoritma genetika tersebut:

- Ukuran populasi : 20, 25, 30, 35 dan 40.
- Maksimal generasi : 10, 50, 100, 500, dan 1000.
- Probabilitas *crossover* (Pc) : 60, 80, 85, 90, dan 95 %.
- Probabilitas mutasi (Pm) : 0.5 – 1 %.

Uji coba dilakukan masing-masing 5 kali percobaan untuk setiap nilai parameter yang diuji. Hasil dari uji coba perubahan nilai ukuran populasi (dengan nilai maksimal generasi 1000, probabilitas *crossover* 60, dan probabilitas mutasi 0.5) akan disimpan seperti Tabel 3.10. Hasil dari uji coba perubahan nilai maksimal generasi (dengan nilai ukuran populasi 40, probabilitas *crossover* 60, dan

probabilitas mutasi 0.5) akan disimpan seperti Tabel 3.11. Hasil dari uji coba perubahan nilai probabilitas *crossover* (dengan nilai ukuran populasi 40, maksimal generasi 1000, dan probabilitas mutasi 0) akan disimpan seperti Tabel 3.12. Sedangkan hasil dari uji coba perubahan nilai probabilitas mutasi (dengan nilai ukuran populasi 40, maksimal generasi 1000, dan probabilitas *crossover* 0) akan disimpan seperti Tabel 3.13.

Tabel 3.10 Rancangan Uji Ukuran Populasi Berbeda

Ukuran Populasi	***Fitness* Rata-rata**	***Fitness* Maksimal**

Tabel 3.11 Rancangan Uji Maksimal Generasi Berbeda

Maksimal Generasi	***Fitness* Rata-rata**	***Fitness* Maksimal**

Tabel 3.12 Rancangan Uji Probabilitas *Crossover* Berbeda

Probabilitas *Crossover*	***Fitness* Rata-rata**	***Fitness* Maksimal**

Tabel 3.13 Rancangan Uji Probabilitas Mutasi Berbeda

Probabilitas Mutasi	***Fitness* Rata-rata**	***Fitness* Maksimal**

Nilai *fitness* digunakan untuk mengetahui tingkat efektifitas dari parameter genetika yang digunakan. Semakin tinggi nilai *fitness* yang dihasilkan maka semakin baik tingkat efektifitas dari nilai parameter yang diuji.


UNIVERSITAS BRAWIJAYA

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dibahas terkait implementasi sistem dan hasil pengujian yang dilakukan berdasarkan rancangan pada bab 3. Pembahasan tersebut antara lain lingkungan implementasi, struktur data, proses-proses yang dilakukan, serta analisa terhadap data yang dihasilkan sistem sesuai masukan dari *user*.

4.1 Lingkungan Implementasi

Implementasi merupakan proses perubahan dari representasi rancangan yang sudah dibuat ke dalam bahasa pemrograman yang dapat dimengerti oleh komputer. Lingkungan implementasi meliputi lingkungan perangkat keras serta perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan implementasi *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* adalah sebagai berikut:

1. *Processor* Intel® Core™ Duo CPU T2400 @ 1.83GHz.
2. Memori 1 GB.
3. *HardDisk* 120 GB.

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pembuatan implementasi *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* adalah sebagai berikut:

1. Sistem operasi Microsoft Windows XP Home Edition Version 2002 Service Pack 3.
2. Microsoft Visual Studio 2005.

4.2 Struktur Data

Struktur data yang digunakan dalam pembuatan implementasi *tree structure modeling* dan algoritma genetika untuk menyelesaikan

unequal-area facility layout problem dapat dilihat pada *Sourcecode* 4.1.

```
1   int ukuranPopulasi;
2   int maxGenerasi;
3   double pc;
4   double pm;
5
6   double lbrLahan;
7   double pjgLahan;
8   ArrayList listRuang;
9   int[][] bobotKedekatan;
10
11  DateTime awal, akhir;
12  TimeSpan selisihWaktu;
13
14  Random random;
15
16  ArrayList populasiInti;
17  ArrayList populasiTerseleksi;
18  ArrayList populasiBaru;
19  ArrayList indukTerpilih;
20  ArrayList kromTermutasi;
21
22  ArrayList log;
```

Sourcecode 4.1 Struktur Data

4.3 Implementasi Program

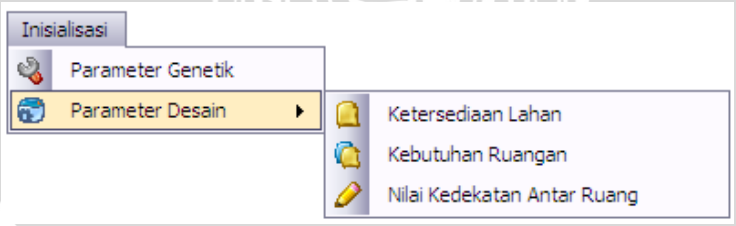
Tampilan utama dari program implementasi *tree structure modeling* dan algoritma genetika untuk menyelesaikan *unequal-area facility layout problem* dapat dilihat pada Gambar 4.1.

Gambar 4.1 Tampilan Utama Program

Program tersebut terdiri dari 4 menu. Masing-masing menu mempunyai fungsi sebagai berikut:

1. Menu Inisialisasi

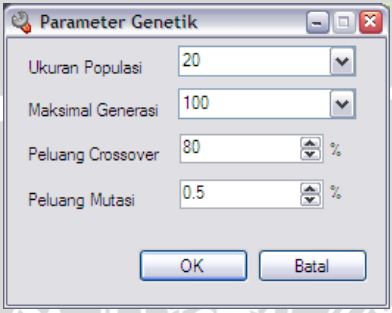
   Menu ini digunakan untuk menginputkan inisialisasi parameter yang akan diproses, yang terbagi menjadi 2 submenu, yaitu parameter genetika dan parameter desain. Sedangkan pada parameter desain terdiri dari 3 langkah, yaitu ketersediaan lahan, kebutuhan ruangan, dan nilai kedekatan antar ruang.



Gambar 4.2 Submenu Inisialisasi

Berikut ini adalah langkah-langkah inisialisasi pada menu Inisialisasi:

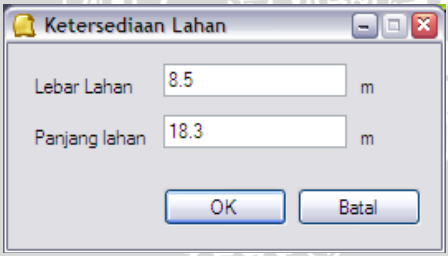
a) Menentukan parameter genetika yang akan digunakan, antara lain: ukuran populasi, maksimal generasi, peluang crossover, dan peluang mutasi. Gambar dialog parameter genetik bisa dilihat pada Gambar 4.3.



Gambar 4.3 Dialog Parameter Genetika

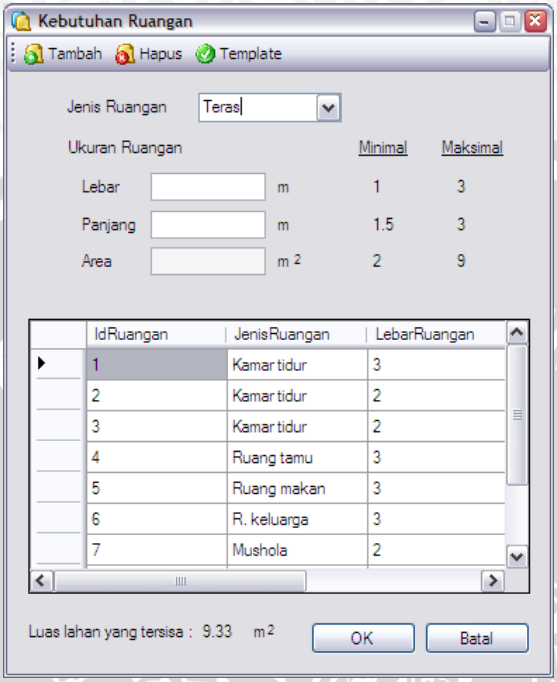
b) Menentukan parameter desain yang diinginkan. Terbagi dalam 3 tahap yang berurutan seperti berikut:

- Pertama, memasukkan data ketersediaan lahan yang akan dibangun sebuah rumah, antara lain: lebar lahan dan panjang lahan.



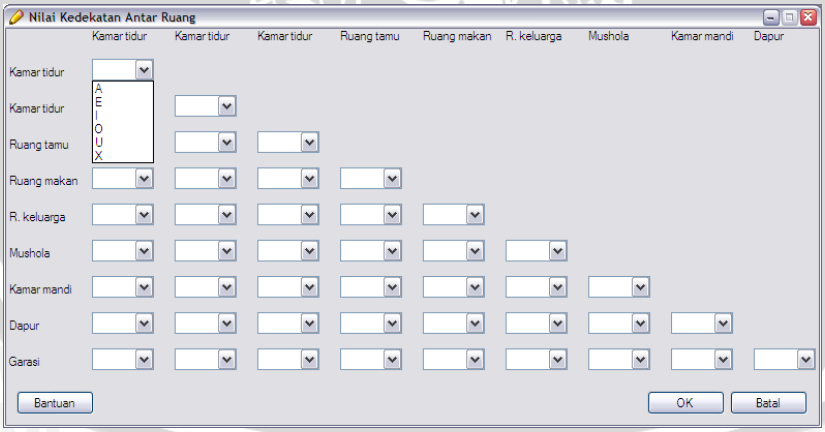
Gambar 4.4 Dialog Ketersediaan Lahan

- Setelah ketersediaan lahan diketahui, maka dapat diinputkan data kebutuhan ruangan sesuai ketersediaan lahan yang telah ditentukan. Data kebutuhan ruangan, antara lain: jenis ruangan, lebar ruangan dan panjang ruangan.

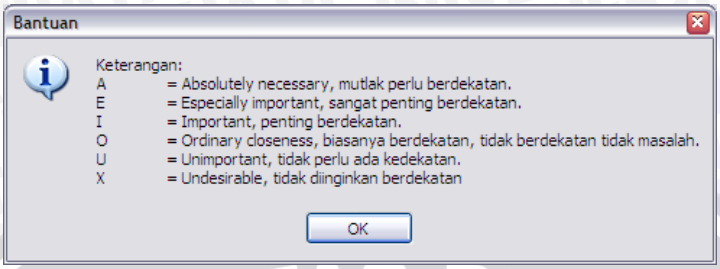


Gambar 4.5 Dialog Kebutuhan Ruangan

- Terakhir, menentukan nilai kedekatan antar ruang sesuai banyaknya kebutuhan ruangan.

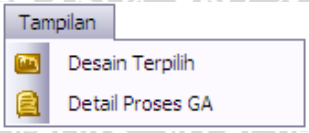


Gambar 4.6 Dialog Nilai Kedekatan Antar Ruang



Gambar 4.7 *Messagebox* Keterangan Nilai Kedekatan Antar Ruang

2. Menu Proses
   Menu untuk memproses data-data yang telah diinputkan pada menu Inisialisasi.
3. Menu Tampilan
   Menu untuk menampilkan hasil dari proses yang dilakukan. Berisi 2 submenu, yaitu desain terpilih dan detail proses GA.



Gambar 4.8 Submenu Tampilan

4. Info
   Menu ini berisi informasi tentang pembuat program.

4.3.1 Inisialisasi Kromosom

Pada proses inisialisasi kromosom dilakukan pembangkitan populasi awal secara acak. Panjang gen pertama sesuai banyaknya ruangan. Sedangkan panjang gen kedua didapatkan dari panjang gen pertama dikurangi 2. Urutan gen pertama menunjukkan ruangan yang diletakan terlebih dahulu. Urutan gen kedua menunjukkan penggabungan ruangan yang dilakukan. Proses inisialisasi kromosom dapat dilihat pada *Sourcecode* 4.2,

```
1  private ArrayList randomGenPertama(int _nGen1)
2  {
3     ArrayList genPertama = new ArrayList();
4     int id = random.Next(1, _nGen1 + 1);
```

```
5       genPertama.Add(id);
6
7       int jumlah = 0;
8       do
9       {
10        id = random.Next(1, _nGen1 + 1);
11        if (!genPertama.Contains(id))
12        {
13          genPertama.Add(id);
14        }
15        jumlah++;
16      } while (jumlah <= 10 * _nGen1);
17      return genPertama;
18  }
19
20  private ArrayList randomGenKedua(int _nGen2)
21  {
22      ArrayList genKedua = new ArrayList();
23
24      for (int jumlah = 0; jumlah < _nGen2; jumlah++)
25      {
26        int kel = random.Next(1, _nGen2 + 1 - jumlah);
27        genKedua.Add(kel);
28      }
29      return genKedua;
30  }
```

Sourcecode 4.2 Proses Inisialisasi Kromosom

4.3.2 *Tree Structure Modeling*

Pembentukan *tree structure modeling* sesuai dengan inisialisasi kromosom yang terbentuk. Proses penggabungan sesuai dengan gen kedua dan urutan ruangan yang akan digabungkan sesuai dengan gen pertama. Proses *tree structure modeling* dapat dilihat pada *Sourcecode* 4.3.

```
1   private void treeStructureModelling()
2   {
3       desainFix = true;
4
5       letakRuangan = new
    Peletakan[listRuangan.Count][];
6       for (int i = 0; i < listRuangan.Count; i++)
7       {
8           letakRuangan[i] = new
    Peletakan[listRuangan.Count];
```

```
9           for (int j = 0; j < listRuangan.Count - i;
    j++)
10          {
11              if (i == 0)
12              {
13                  int id = (int)genPertama[j];
14                  Ruangan ruang =
    (Ruangan)listRuangan[id - 1];
15                  letakRuangan[i][j] = new
    Peletakan(id.ToString(), "[" + ruang.JenisRuangan +
    ruang.IdRuangan.ToString() + "]",
    (int)(ruang.LebarRuangan * 10),
    (int)(ruang.PanjangRuangan * 10), 0, 0, null,
    null);
16              }
17              else if ((i != 0) && (i !=
    listRuangan.Count - 1))
18              {
19                  int idGab = (int)genKedua[i - 1];
20                  int idRuang = j + 1;
21                  if (idRuang == idGab)
22                  {
23                      if (((letakRuangan[i -
    1][j].Lebar + letakRuangan[i - 1][j + 1].Lebar <=
    lLahan) && (Math.Max(letakRuangan[i -
    1][j].Panjang, letakRuangan[i - 1][j + 1].Panjang)
    <= pLahan)) || ((letakRuangan[i - 1][j].Lebar +
    letakRuangan[i - 1][j + 1].Panjang <= lLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Panjang,
    letakRuangan[i - 1][j + 1].Lebar) <= pLahan)) ||
    ((letakRuangan[i - 1][j].Panjang + letakRuangan[i -
    1][j + 1].Lebar <= lLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Lebar,
    letakRuangan[i - 1][j + 1].Panjang) <= pLahan)) ||
    ((letakRuangan[i - 1][j].Panjang + letakRuangan[i -
    1][j + 1].Panjang <= lLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Lebar,
    letakRuangan[i - 1][j + 1].Lebar) <= pLahan)))
24                      {
25                          peletakanKanan(letakRuangan,
    i, j);
26                      }
27                      else if (((letakRuangan[i -
    1][j].Panjang + letakRuangan[i - 1][j + 1].Panjang
    <= pLahan) && (Math.Max(letakRuangan[i -
    1][j].Lebar, letakRuangan[i - 1][j + 1].Lebar) <=
    lLahan)) || ((letakRuangan[i - 1][j].Panjang +
    letakRuangan[i - 1][j + 1].Lebar <= pLahan) &&
```

```
    (Math.Max(letakRuangan[i - 1][j].Lebar,
    letakRuangan[i - 1][j + 1].Panjang) <= lLahan)) ||
     ((letakRuangan[i - 1][j].Lebar + letakRuangan[i -
    1][j + 1].Panjang <= pLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Panjang,
    letakRuangan[i - 1][j + 1].Lebar) <= lLahan)) ||
     ((letakRuangan[i - 1][j].Lebar + letakRuangan[i -
    1][j + 1].Lebar <= pLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Panjang,
    letakRuangan[i - 1][j + 1].Panjang) <= lLahan)))
28                      {
29                          peletakanBawah(letakRuangan,
    i, j);
30                      }
31                      else
32                      {
33                          desainFix = false;
34                          fitness = 0;
35                          break;
36                      }
37                  }
38                  else if (idRuang > idGab)
39                  {
40                      letakRuangan[i][j] =
    letakRuangan[i - 1][j + 1];
41                  }
42                  else if (idRuang < idGab)
43                  {
44                      letakRuangan[i][j] =
    letakRuangan[i - 1][j];
45                  }
46              }
47              else //if (i == listRuangan.Count - 1)
48              {
49                  //mengabungan keseluruhan ruangan
50                  if (((letakRuangan[i - 1][j].Lebar
    + letakRuangan[i - 1][j + 1].Lebar <= lLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Panjang,
    letakRuangan[i - 1][j + 1].Panjang) <= pLahan)) ||
     ((letakRuangan[i - 1][j].Lebar + letakRuangan[i -
    1][j + 1].Panjang <= lLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Panjang,
    letakRuangan[i - 1][j + 1].Lebar) <= pLahan)) ||
     ((letakRuangan[i - 1][j].Panjang + letakRuangan[i
    - 1][j + 1].Lebar <= lLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Lebar,
    letakRuangan[i - 1][j + 1].Panjang) <= pLahan)) ||
    ((letakRuangan[i - 1][j].Panjang + letakRuangan[i -
```

```
    1][j + 1].Panjang <= lLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Lebar,
    letakRuangan[i - 1][j + 1].Lebar) <= pLahan)))
51                  {
52                      peletakanKanan(letakRuangan, i,
    j);
53                  }
54                  else if (((letakRuangan[i -
    1][j].Panjang + letakRuangan[i - 1][j + 1].Panjang
    <= pLahan) && (Math.Max(letakRuangan[i -
    1][j].Lebar, letakRuangan[i - 1][j + 1].Lebar) <=
    lLahan)) || ((letakRuangan[i - 1][j].Panjang +
    letakRuangan[i - 1][j + 1].Lebar <= pLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Lebar,
    letakRuangan[i - 1][j + 1].Panjang) <= lLahan)) ||
     ((letakRuangan[i - 1][j].Lebar + letakRuangan[i -
    1][j + 1].Panjang <= pLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Panjang,
    letakRuangan[i - 1][j + 1].Lebar) <= lLahan)) ||
     ((letakRuangan[i - 1][j].Lebar + letakRuangan[i -
    1][j + 1].Lebar <= pLahan) &&
    (Math.Max(letakRuangan[i - 1][j].Panjang,
    letakRuangan[i - 1][j + 1].Panjang) <= lLahan)))
55                  {
56                      peletakanBawah(letakRuangan,
    i, j);
57                  }
58                  else
59                  {
60                      desainFix = false;
61                      fitness = 0;
62                      break;
63                  }
64              }
65          }
66          if (!desainFix) break;
67      }
68
69      if (desainFix)
70      {
71          Peletakan letak =
    letakRuangan[listRuangan.Count - 1][0];
72      letak.aturLetak(letakRuangan[listRuangan.Count -
    1][0]);
73          Peletakan[] letakOk = new
    Peletakan[listRuangan.Count];
74          listNamaRuang = new
```

```
    string[listRuangan.Count];
75          string[] listIdRuang = new
    string[listRuangan.Count];
76          ruangDalamRumah = new
    Rectangle[listRuangan.Count];
77
78          for (int i = 0; i < listRuangan.Count; i++)
79          {
80              Ruangan ruang =
    (Ruangan)listRuangan[i];
81              listNamaRuang[i] = ruang.JenisRuangan;
82              listIdRuang[i] =
    ruang.IdRuangan.ToString();
83              letakOk[i] = letak.cariLetakRuang("[" +
    listNamaRuang[i] + listIdRuang[i] + "]",
    letakRuangan[listRuangan.Count - 1][0]);
84              ruangDalamRumah[i] = new
    Rectangle(letakOk[i].X, letakOk[i].Y,
    letakOk[i].Lebar, letakOk[i].Panjang);
85          }
86
87          ArrayList ruangan_list = new ArrayList();
88          for (int i = 0; i < ruangDalamRumah.Length;
    i++)
89          {
90              ruangan_list.Add(ruangDalamRumah[i]);
91          }
92          Rectangle[] ruangan_array =
    (Rectangle[])ruangan_list.ToArray(typeof(Rectangle)
    );
93          jarakRuangan = new
    double[listRuangan.Count][];
94          for (int i = 0; i < listRuangan.Count; i++)
95          {
96              jarakRuangan[i] = new
    double[listRuangan.Count];
97              for (int j = i + 1; j <
    listRuangan.Count; j++)
98              {
99                  jarakRuangan[i][j] =
    hitungJarak(ruangan_array[i], ruangan_array[j]);
100             }
101         }
102     }
103 }
```

Sourcecode 4.3 Proses *Tree Structure Modeling*

4.3.3 Perhitungan *Fitness*

Perhitungan *fitness* dilakukan pada masing-masing kromosom. Perhitungan *fitness* diperoleh dari *inverse* penjumlahan *total travel distance* ditambah 1. Adapun *total travel distance* diperoleh dari penjumlahan matriks jarak antar ruang yang dikalikan dengan matriks kedekatan antar ruang. Proses perhitungan *fitness* dapat dilihat pada *Sourcecode* 4.4.

```
1   public void hitungFitness()
2   {
3     double totalTravelDistance = 0;
4
5     for (int i = 0; i < listRuangan.Count; i++)
6     {
7       for (int j = i + 1; j < listRuangan.Count; j++)
8       {
9         totalTravelDistance += jarakRuangan[i][j] *
          bKedekatan[i][j];
10      }
11    }
12    fitness = 1 / (totalTravelDistance + 1);
13  }
```

Sourcecode 4.4 Proses Perhitungan *Fitness*

4.3.4 Seleksi

Proses seleksi yang digunakan adalah seleksi dengan mesin *roulette*. Proses ini dimulai dengan menghitung total nilai *fitness* semua kromosom pada populasi. Kemudian memilih kromosom-kromosom pada populasi secara random berdasarkan probabilitas kumulatif yang dimiliki. Proses seleksi dapat dilihat pada *Sourcecode* 4.5.

```
1   private void seleksi()
2   {
3     double totFitness = 0;
4
5     foreach (Kromosom krom in populasiInti)
6     {
7       totFitness += krom.Fitness;
8     }
9
10    Dictionary<Kromosom, double> kromProbs = new
```

```
    Dictionary<Kromosom, double>(ukuranPopulasi);
11
12    double probKum = 0;
13
14    foreach (Kromosom krom in populasiInti)
15    {
16      probKum += (krom.Fitness / totFitness);
17      kromProbs.Add(krom, probKum);
18    }
19
20    for (int i = 0; i < ukuranPopulasi; i++)
21    {
22      double r = 0;
23      Kromosom kromTerpilih = null;
24      do
25      {
26         r = random.NextDouble();
27
28         foreach (KeyValuePair<Kromosom, double>
           range in kromProbs)
29         {
30           if (range.Value > r)
31           {
32             kromTerpilih = range.Key;
33             populasiTerseleksi.Add(kromTerpilih);
34             break;
35           }
36         }
37       }
38       while
          (Array.IndexOf(populasiTerseleksi.ToArray(),
          kromTerpilih) == -1);
39    }
40  }
```

Sourcecode 4.5 Proses Seleksi

4.3.5 *Crossover*

Metode *crossover* yang digunakan adalah *Partially Mapped Crossover* (PMX) pada gen pertama dan *crossover* satu titik pada gen kedua. Sebelum dilakukan proses *crossover*, dilakukan pemilihan sepasang kromosom induk yang akan mengalami *crossover* sesuai dengan nilai probabilitas *crossover* (Pc) yang telah ditentukan. Proses pemilihan kromosom yang akan di-*crossover* tersebut dapat dilihat pada *Sourcecode* 4.6.

```
1   private void crossover()
2   {
3     double r;
4     Kromosom induk1;
5     Kromosom induk2;
6
7     for (int i = 0; i < ukuranPopulasi; i++)
8     {
9       r = random.NextDouble();//nilai acak 0..1
10      if (r < (pc / 100))
11      {
12        indukTerpilih.Add((Kromosom)
          populasiTerseleksi[i]);
13      }
14      if (indukTerpilih.Count >= (int)ukuranPopulasi
          * (pc / 100))
15      break;
16    }
17
18    for (int j = 0; j < indukTerpilih.Count - 1; j +=
          2)
19    {
20      if((indukTerpilih[j] == null ||indukTerpilih[j
            + 1] == null))
21      break;
22      induk1 = (Kromosom)indukTerpilih[j];
23      induk2 = (Kromosom)indukTerpilih[j + 1];
24
25      doCrossover(induk1, induk2);
26    }
27  }
```

Sourcecode 4.6 Proses Pemilihan Dua Induk *Crossover*

Langkah selanjutnya adalah proses *crossover* yang terdiri dari *Partially Mapped Crossover* (PMX) pada gen pertama dan *crossover* satu titik pada gen kedua. Pada proses PMX, dibangkitkan dua bilangan random dengan nilai 1 sampai dengan banyaknya ruangan dikurangi 1. Bilangan random pertama dan kedua tidak boleh sama. Dua bilangan tersebut akan dijadikan *cutpoint* untuk menentukan dua posisi pada gen pertama untuk medapatkan substring gen tersebut. Setelah substring didapatkan, maka substring dari induk pertama ditukar dengan substring induk kedua.Kemudian menentukan hubungan pemetaan pada substring. Hubungan

pemetaan ini dijadikan acuan untuk menentukan gen pertama keturunan baru apabila terdapat nilai gen yang sama.

Sedangkan pada *crossover* satu titik, hanya dibangkitkan satu bilangan random dengan nilai 1 sampai dengan banyaknya ruangan dikurangi 1 sebagai *cutpoint*. Bilangan ini akan dijadikan acuan perubahan gen kedua dimana nilai gen pada *cutpoint* dan setelahnya pada induk pertama dan induk kedua ditukar. Proses *crossover* tersebut dapat dilihat pada *Sourcecode* 4.7.

```
1   private void doCrossover(Kromosom _induk1, Kromosom
    _induk2)
2   {
3     Kromosom keturunan1;
4     Kromosom keturunan2;
5
6     //PMX (Partially Mapped Crossover)
7     int cutpoint1;
8     int cutpoint2;
9     ArrayList keturunan1GenPertama = new
       ArrayList(listRuang.Count);
10    ArrayList keturunan2GenPertama = new
       ArrayList(listRuang.Count);
11    cutpoint1 = random.Next(1, listRuang.Count - 1);
12    cutpoint2 = random.Next(1, listRuang.Count - 1);
13    while (cutpoint2 == cutpoint1)
14     cutpoint2 = random.Next(1, listRuang.Count - 1);
15
16    if (cutpoint1 > cutpoint2)
17    {
18      int tukar;
19      tukar = cutpoint1;
20      cutpoint1 = cutpoint2;
21      cutpoint2 = tukar;
22    }
23
24    int[] pengganti1 = new int[listRuang.Count];
25    int[] pengganti2 = new int[listRuang.Count];
26    for (int i = 0; i < listRuang.Count; i++)
27    {
28      pengganti1[i] = pengganti2[i] = -1;
29      keturunan1GenPertama.Add(-1);
30      keturunan2GenPertama.Add(-1);
31    }
32    for (int i = cutpoint1; i < cutpoint2; i++)
33    {
```

```
34      keturunan1GenPertama[i] =
        _induk2.GenPertama[i];
35      keturunan2GenPertama[i] =
        _induk1.GenPertama[i];
36
37      pengganti1[(int)_induk2.GenPertama[i] - 1] =
         (int)_induk2.GenPertama[i];
38
39      pengganti2[(int)_induk1.GenPertama[i] - 1] =
         (int)_induk1.GenPertama[i];
40    }
41
42    for (int i = 0; i < listRuang.Count; i++)
43    {
44      if ((i >= cutpoint1) && (i < cutpoint2))
45        continue;
46
47      int n1 = (int)_induk1.GenPertama[i];
48      int m1 = pengganti1[n1 - 1];
49      int n2 = (int)_induk2.GenPertama[i];
50      int m2 = pengganti2[n2 - 1];
51
52      while (m1 != -1)
53      {
54        n1 =
          (int)_induk1.GenPertama[(int)_induk2.GenPertam
          a.IndexOf(m1)];
55        m1 = pengganti1[n1 - 1];
56      }
57      while (m2 != -1)
58      {
59        n2 =
          (int)_induk2.GenPertama[(int)_induk1.GenPertam
          a.IndexOf(m2)];
60        m2 = pengganti2[n2 - 1];
61      }
62      keturunan1GenPertama[i] = n1;
63      keturunan2GenPertama[i] = n2;
64    }
65
66    //crossover satu titik
67    int cutpoint = random.Next(1, listRuang.Count -
68  3);
69    ArrayList keturunan1GenKedua = new
       ArrayList(listRuang.Count - 3);
70    ArrayList keturunan2GenKedua = new
       ArrayList(listRuang.Count - 3);
71    for (int j = 0; j < listRuang.Count - 2; j++)
```

```
72      {
73        keturunan1GenKedua.Add(-1);
74        keturunan2GenKedua.Add(-1);
75      }
76
77      for (int j = 0; j < listRuang.Count - 2; j++)
78      {
79        if (j <= cutpoint)
80        {
81          keturunan1GenKedua[j] = _induk1.GenKedua[j];
82          keturunan2GenKedua[j] = _induk2.GenKedua[j];
83        }
84        else
85        {
86          keturunan1GenKedua[j] = _induk2.GenKedua[j];
87          keturunan2GenKedua[j] = _induk1.GenKedua[j];
88        }
89      }
90
91      keturunan1 = new Kromosom(keturunan1GenPertama,
          keturunan1GenKedua, lbrLahan, pjgLahan,
          listRuang, bobotKedekatan);
92      keturunan2 = new Kromosom(keturunan2GenPertama,
          keturunan2GenKedua, lbrLahan, pjgLahan,
          listRuang, bobotKedekatan);
93
94      populasiBaru.Add(keturunan1);
95      populasiBaru.Add(keturunan2);
96    }
```

Sourcecode 4.7 Proses *Crossover*

4.3.6 Mutasi

Metode mutasi yang digunakan adalah *Reciprocal Exchange Mutation* (REM) pada gen pertama dan mutasi random pada gen kedua berdasarkan$1 \leq b_j \leq n\text{-}j$ dimana b_j adalah nilai bilangan random yang dipilih, *n* adalah banyaknya ruangan, sedangkan *j* adalah indeks bilangan random tersebut. Langkah awal proses mutasi hampir sama dengan proses *crossover*, yaitu dilakukan pemilihan kromosom yang akan mengalami mutasi sesuai dengan probabilitas mutasi yang telah ditentukan. Proses pemilihan kromosom yang akan mengalami mutasi dapat dilihat pada *Sourcecode* 4.8.

```
1   private void mutasi()
2   {
```

```
3     double r;
4     Kromosom mutasi;
5     for (int i = 0; i < ukuranPopulasi; i++)
6     {
7       r = random.NextDouble();//nilai acak 0..1
8       if (r < (pm / 100))
9       {
10        kromTermutasi.Add((Kromosom)populasiBaru[i]);
11      }
12      if (kromTermutasi.Count == (int)ukuranPopulasi
          * (pm / 100))
13        break;
14    }
15
16    for (int j = 0; j < kromTermutasi.Count; j++)
17    {
18      mutasi = (Kromosom)kromTermutasi[j];
19      doMutasi(mutasi);
20    }
21  }
```

Sourcecode 4.8 Proses Pemilihan Kromosom Mutasi

Langkah selanjutnya adalah proses mutasi yang terdiri dari proses REM pada gen pertama dan mutasi random pada gen kedua. Pada REM, dibangkitkan dua bilangan random dengan nilai antara 0 sampai dengan banyaknya ruangan dikurangi 1. Kemudian nilai gen dengan indeks kedua bilangan tersebut ditukar. Sedangkan pada proses mutasi random, dibangkitkan satu bilangan random dengan nilai antara 0 sampai dengan banyaknya ruangan dikurangi 3. Nilai gen dengan indeks bilangan random tersebut diganti dengan bilangan random lain dengan nilai antara 1 sampai banyaknya ruangan dikurangi bilangan random pertama. Proses mutasi dapat dilihat pada *Sourcecode* 4.9.

```
1   private void doMutasi(Kromosom _mutasi)
2   {
3     //REM (Reciprocal Exchange Mutation)
4     int index1, index2, temp;
5
6     index1 = random.Next(0, listRuang.Count - 1);
7     index2 = random.Next(0, listRuang.Count - 1);
8     while (index2 == index1)
9       index2 = random.Next(0, listRuang.Count - 1);
10
```

```
11      temp = (int) _mutasi.GenPertama[index1];
12      _mutasi.GenPertama[index1] =
          _mutasi.GenPertama[index2];
13      _mutasi.GenPertama[index2] = temp;
14
15      //mutasi
16      int index = random.Next(0, listRuang.Count - 3);
17      _mutasi.GenKedua[index] = random.Next(1,
          listRuang.Count - index);
18
19      populasiBaru.Add(_mutasi);
20  }
```

Sourcecode 4.9 Proses Mutasi

4.4 Penerapan Aplikasi

Penerapan aplikasi dilakukan dengan memasukkan data-data inisialisasi yang diperlukan. Data-data tersebut antara lain parameter genetik dan parameter desain. Parameter genetik yang digunakan adalah sebagai berikut:

- Ukuran populasi : 20
- Maksimal generasi : 100
- Probabilitas *crossover* : 80 %
- Probabilitas mutasi : 0,5 %

Sedangkan parameter desain yang digunakan adalah salah satu contoh permasalahan nyata yang diambil dari internet (http://eramuslim.com/konsultasi.arsitektur/desai-rumah-islami-85-90-m2.htm) dengan penyesuaian batasan masalah pada bab 1 dan batasan sistem pada subbab 3.1.2. Berikut adalah parameter desain yang digunakan:

- Ketersediaan lahan
  - Lebar lahan : 8.5 m
  - Panjang lahan : 17.3 m
- Kebutuhan ruangan

Tabel 4.1 Data Parameter Kebutuhan Ruangan

Id	Jenis Ruangan	Lebar (m)	Panjang (m)
1.	Kamar tidur	3	3
2.	Kamar tidur	2	3.5
3.	Kamar tidur	2	3.5

4.	Ruang tamu	3	3
5.	Ruang makan	3	3
6.	Ruang keluarga	3	4
7.	Mushola	2	3
8.	Kamar mandi	2	2
9.	Dapur	2	3
10	Garasi	3	5

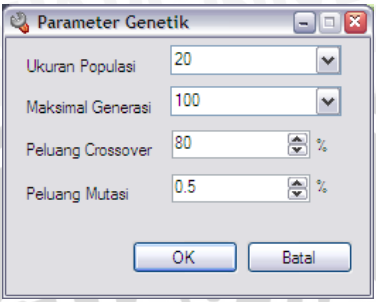
- Nilai kedekatan antar ruang yang digunakan dapat dilihat pada Tabel 4.2.

Tabel 4.2 Data Parameter Nilai Kedekatan Antar Ruang

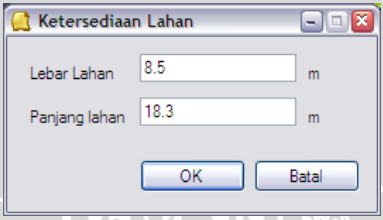
Kebutuhan Ruangan	i / j	1	2	3	4	5	6	7	8	9	10
Kamar tidur 1	1	-									
Kamar tidur 2	2	O	-								
Kamar tidur 3	3	U	U	-							
Ruang tamu	4	X	X	U	-						
Ruang makan	5	U	U	O	O	-					
R. keluarga	6	I	I	O	I	E	-				
Mushola	7	U	U	U	U	I	I	-			
Kamar mandi	8	E	I	I	O	X	O	I	-		
Dapur	9	U	U	O	U	A	I	I	U	-	
Garasi	10	X	X	U	A	U	U	X	U	I	-

Pada Tabel 4.1, lebar dan panjang ruangan tidak boleh melebihi batas minimal dan maksimal yang telah ditentukan pada Tabel 3.1 Spesifikasi Ruangan dimana disesuaikan dengan jenis ruangan tersebut. Sedangkan pada Tabel 4.2, pemberian nilai kedekatan disesuaikan keinginan *user*. Huruf AEIOUX menunjukkan kedekatan yang diinginkan. Bobot kedekatan tersebut dapat dilihat pada Tabel 2.1 Nilai Hubungan Kedekatan.

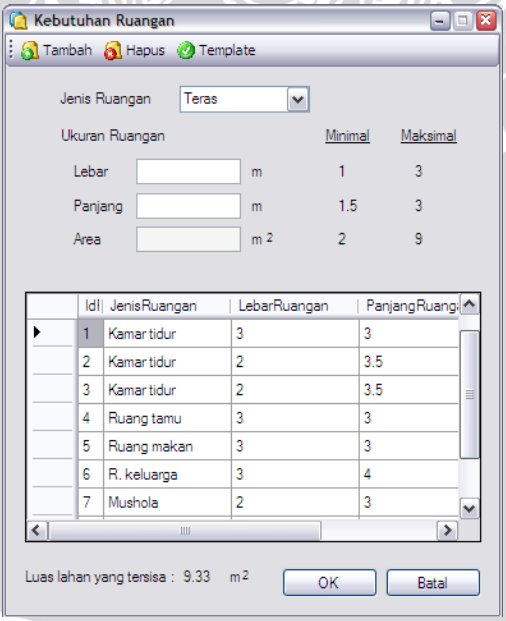
Proses inisialisasi parameter dalam implementasi program dapat dilihat pada Gambar 4.9 sampai dengan Gambar 4.12:



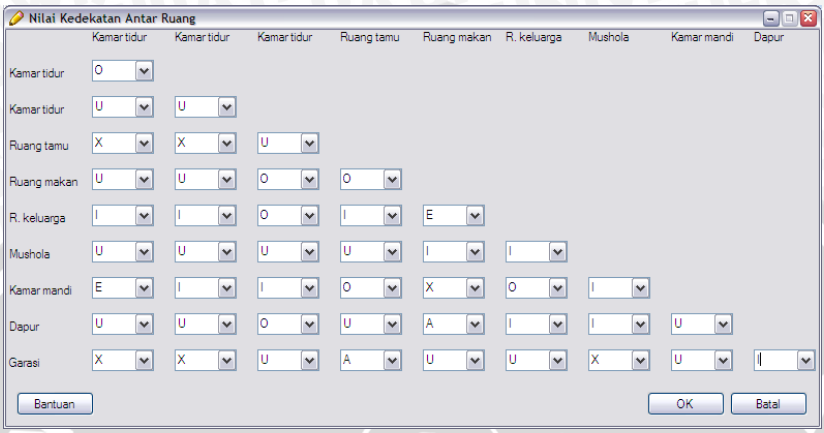
Gambar 4.9 Inisialisasi Parameter Genetika



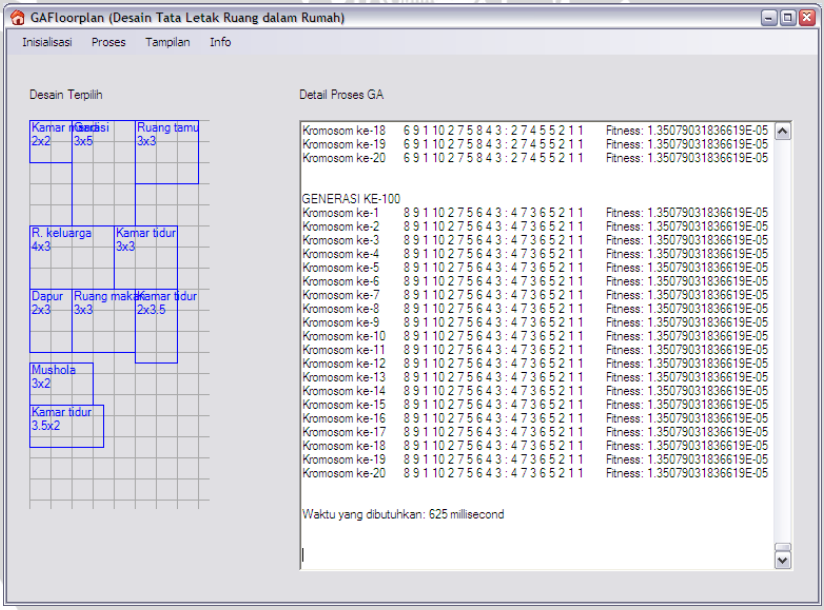
Gambar 4.10 Inisialisasi Ketersediaan Lahan



Gambar 4.11 Inisialisasi Kebutuhan Ruangan



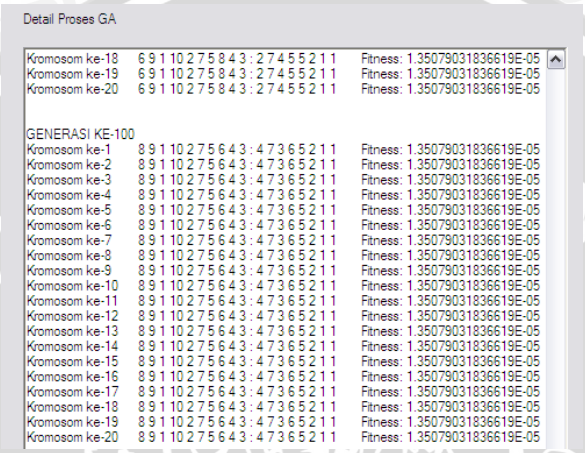
Gambar 4.12 Inisialisasi Nilai Kedekatan Antar Ruang



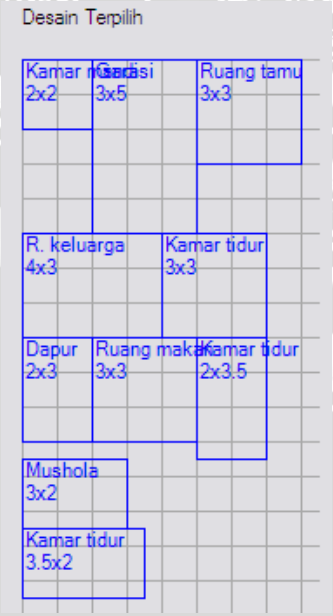
Gambar 4.13 Implementasi Program

Setelah dilakukan proses genetika dan *tree structure modeling*, diperoleh solusi yang terbaik pada generasi terakhir. Solusi tersebut merupakan kromosom yang paling optimal, yaitu dengan *nilai fitness*

1.83567924685061 x 10^{-5} yang dapat dilihat pada kromosom ke-20 pada Gambar 4.14. Dari kromosom tersebut dapat dihasilkan desain tata letak ruang dalam rumah seperti pada Gambar 4.15.



Gambar 4.14 Detail Proses Algoritma Genetika



Gambar 4.15 Desain Terpilih

4.5 Implementasi Uji Coba

4.5.1 Skenario Uji Coba

Pada tahap ini, uji coba yang digunakan sama dengan pembahasan sub bab 3.4, dimana pengujian akan dilakukan dengan parameter genetika yang berbeda.

Dari data-data inisialisasi parameter desain tersebut dilakukan pengujian keefektifan *tree structure modeling* dan algoritma genetika dalam implementasi dengan beberapa variasi parameter genetika sebagai berikut:

- Ukuran populasi : 20, 25, 30, 35 dan 40.
- Maksimal generasi : 10, 50, 100, 500, dan 1000.
- Probabilitas *crossover* (Pc) : 60, 80, 85, 90, dan 95 %.
- Probabilitas mutasi (Pm) : 0.5 – 1 %.

Uji coba dilakukan masing-masing 5 kali percobaan untuk setiap nilai parameter yang diuji. Uji coba pertama dilakukan dengan perubahan nilai ukuran populasi (dengan maksimal generasi 1000, probabilitas *crossover* 60, dan probabilitas mutasi 0.5. Uji coba kedua dengan perubahan nilai maksimal generasi (dengan ukuran populasi 40, probabilitas *crossover* 60, dan probabilitas mutasi 0.5). Uji coba ketiga dengan perubahan nilai probabilitas *crossover* (dengan ukuran populasi 40, maksimal generasi 1000, dan probabilitas mutasi 0). Sedangkan uji coba terakhir dengan perubahan nilai probabilitas mutasi (dengan ukuran populasi 40, maksimal generasi 1000, dan probabilitas *crossover* 0).

Data-data parameter desain untuk tiap pengujian sama. Hal ini dilakukan untuk mengetahui adanya perubahan nilai *fitness*. Berikut adalah parameter desain yang digunakan:

- Ketersediaan lahan
  - Lebar lahan : 8.5 m
  - Panjang lahan : 17.3 m
- Kebutuhan ruangan

Tabel 4.3 Data Uji Coba Parameter Kebutuhan Ruangan

Id	Jenis Ruangan	Lebar (m)	Panjang (m)
1.	Kamar tidur	3	3

2.	Kamar tidur	2	3.5
3.	Kamar tidur	2	3.5
4.	Ruang tamu	3	3
5.	Ruang makan	3	3
6.	Ruang keluarga	3	4
7.	Mushola	2	3
8.	Kamar mandi	2	2
9.	Dapur	2	3
10	Garasi	3	5

- Nilai kedekatan antar ruang yang digunakan dapat dilihat pada Tabel 4.4.

Tabel 4.4 Data Uji Coba Parameter Nilai Kedekatan Antar Ruang

Kebutuhan Ruangan	i / j	1	2	3	4	5	6	7	8	9	10
Kamar tidur 1	1	-									
Kamar tidur 2	2	O	-								
Kamar tidur 3	3	U	U	-							
Ruang tamu	4	X	X	U	-						
Ruang makan	5	U	U	O	O	-					
R. keluarga	6	I	I	O	I	E	-				
Mushola	7	U	U	U	U	I	I	-			
Kamar mandi	8	E	I	I	O	X	O	I	-		
Dapur	9	U	U	O	U	A	I	I	U	-	
Garasi	10	X	X	U	A	U	U	X	U	I	-

4.5.2 Hasil Uji Coba

Dari skenario uji coba, program dijalankan sesuai dengan inisialisasi parameter yang sudah ditentukan. Ada 4 macam uji coba yang dilakukan untuk perubahan nilai *fitness* pada ukuran populasi, maksimal generasi, probabilitas *crossover*, dan probabilitas mutasi yang berbeda sebanyak 5 kali untuk masing-masing parameter yang diuji.

Pada Tabel 4.5 dapat diketahui *fitness* maksimal dan *fitness* rata-rata pada setiap kali percobaan. Dari hasil uji coba dapat dilihat bahwa nilai *fitness* maksimal tertinggi diperoleh pada ukuran populasi 40 namun nilai *fitness* rata-rata tertinggi diperoleh pada ukuran populasi 35. Sedangkan nilai *fitness* maksimal dan nilai

fitness rata-rata terendah diperoleh pada ukuran populasi 25. Pengaruh perubahan ukuran populasi terhadap nilai *fitness* rata-rata dan nilai *fitness* maksimal dapat dilihat pada Gambar 4.16.

Tabel 4.5 Hasil Uji pada Ukuran Populasi Berbeda

Ukuran Populasi	*Fitness* Rata-rata	*Fitness* Maksimal
20	1.41E-05	1.52E-05
25	1.39E-05	1.46E-05
30	1.40E-05	1.47E-05
35	1.46E-05	1.55E-05
40	1.44E-05	1.57E-05

Pada Tabel 4.6 dapat diketahui *fitness* maksimal dan *fitness* rata-rata pada setiap kali percobaan. Dari hasil uji coba dapat dilihat bahwa nilai *fitness* maksimal tertinggi diperoleh pada maksimal generasi 1000 namun nilai *fitness* rata-rata tertinggi diperoleh sama pada maksimal generasi 500 dan 1000. Sedangkan nilai *fitness* maksimal dan nilai *fitness* rata-rata terendah diperoleh pada maksimal generasi 10. Pengaruh perubahan maksimal generasi terhadap nilai *fitness* rata-rata dan nilai *fitness* maksimal dapat dilihat pada Gambar 4.17.

Tabel 4.6 Hasil Uji pada Maksimal Generasi Berbeda

Maksimal Generasi	*Fitness* Rata-rata	*Fitness* Maksimal
10	1.23E-05	1.36E-05
50	1.36E-05	1.46E-05
100	1.39E-05	1.45E-05
500	1.47E-05	1.55E-05
1000	1.47E-05	1.56E-05

Pada Tabel 4.7 dapat diketahui *fitness* maksimal dan *fitness* rata-rata pada setiap kali percobaan. Dari hasil uji coba dapat dilihat bahwa nilai *fitness* maksimal dan *fitness* rata-rata tertinggi diperoleh pada probabilitas *crossover* 80. Sedangkan nilai *fitness* maksimal

terendah diperoleh pada probabilitas *crossover* 85 dan nilai *fitness* rata-rata terendah diperoleh pada probabilitas *crossover* 60. Pengaruh perubahan pada probabilitas *crossover* terhadap nilai *fitness* rata-rata dan nilai *fitness* maksimal dapat dilihat pada Gambar 4.18.

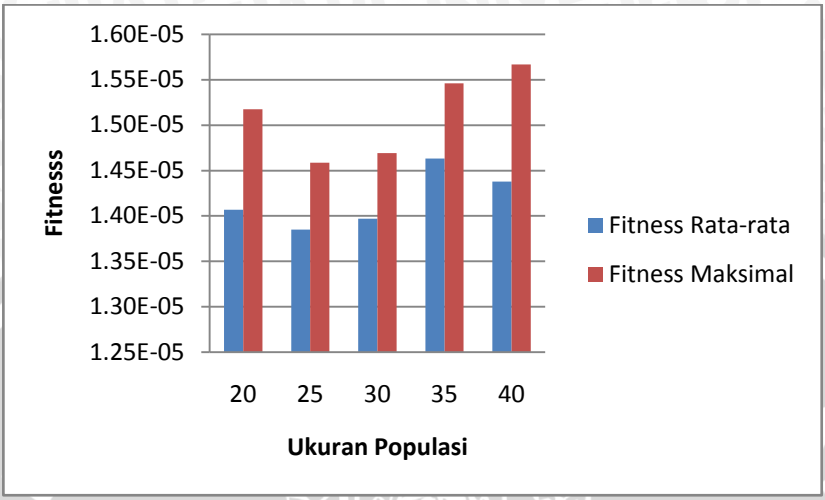
Tabel 4.7 Hasil Uji pada Probabilitas *Crossover* Berbeda

Probabilitas *Crossover*	*Fitness* Rata-rata	*Fitness* Maksimal
60	1.40E-05	1.54E-05
80	1.50E-05	1.56E-05
85	1.43E-05	1.49E-05
90	1.42E-05	1.50E-05
95	1.46E-05	1.50E-05

Pada Tabel 4.8 dapat diketahui *fitness* maksimal dan *fitness* rata-rata pada setiap kali percobaan. Dari hasil uji coba dapat dilihat bahwa nilai *fitness* maksimal dan *fitness* rata-rata tertinggi diperoleh pada probabilitas mutasi 0.5. Sedangkan nilai *fitness* maksimal dan *fitness* rata-rata terendah diperoleh pada probabilitas mutasi 0.6. Pengaruh perubahan pada probabilitas mutasi terhadap nilai *fitness* rata-rata dan nilai *fitness* maksimal dapat dilihat pada Gambar 4.19.

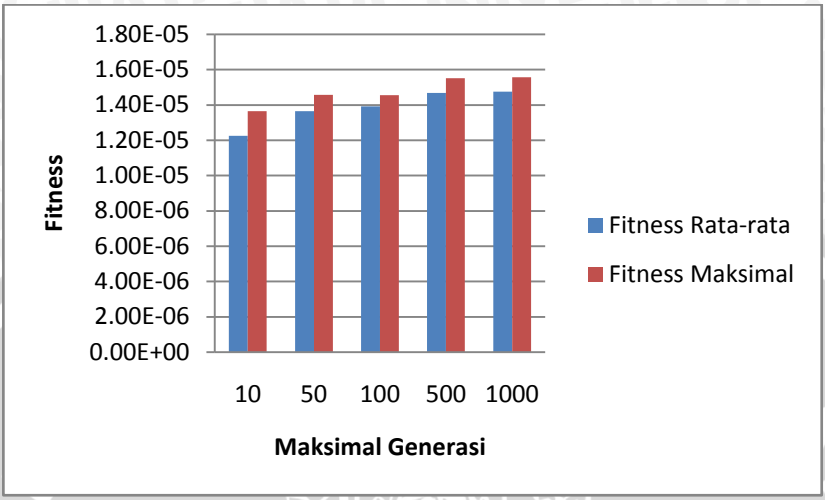
Tabel 4.8 Hasil Uji pada Probabilitas Mutasi Berbeda

Probabilitas Mutasi	Fitness Rata-rata	Fitness Maksimal
0.5	1.27E-05	1.58E-05
0.6	1.12E-05	1.23E-05
0.7	1.16E-05	1.32E-05
0.8	1.15E-05	1.37E-05
0.9	1.15E-05	1.29E-05
1	1.17E-05	1.30E-05



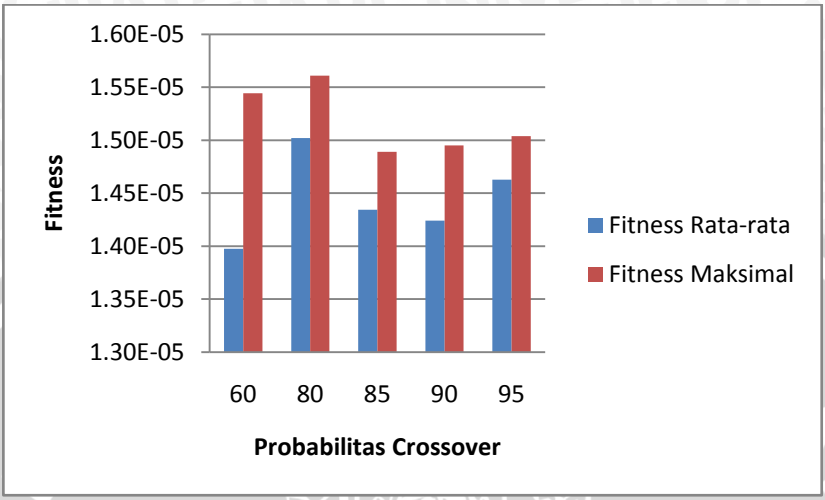
Gambar 4.16 Grafik Pengaruh Perubahan Ukuran Populasi

Pada Grafik 4.16 menunjukkan pengaruh perubahan ukuran populasi terhadap nilai *fitness* rata-rata dan nilai *fitness* maksimal pada algoritma genetika.Walaupun nilai *fitness* maksimal tertinggi diperoleh pada ukuran populasi 40. Nilai *fitness* terbaik stabil pada ukuran populasi 35 dengan ditunjukkan nilai *fitness* rata-rata tertinggi diperoleh pada ukuran populasi tersebut. Selain itu, semakin kecilnya ukuran populasi semakin cepat waktu yang dibutuhkan dalam proses genetika.



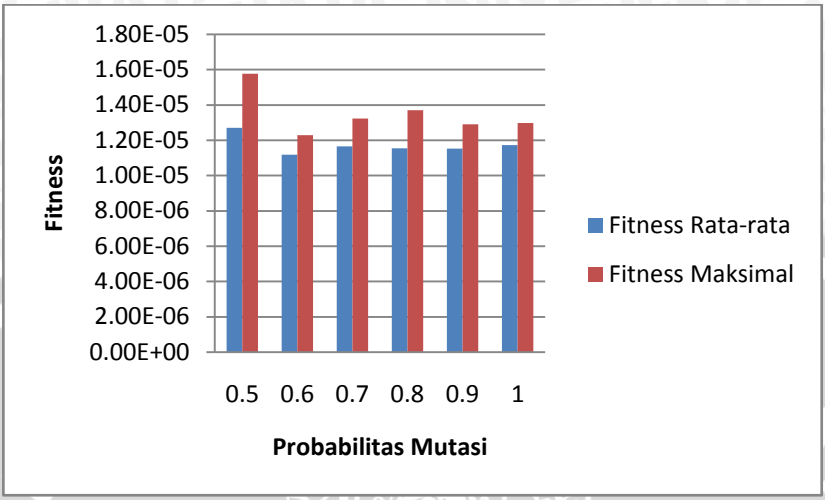
Gambar 4.17 Grafik Pengaruh Perubahan Maksimal Generasi

Pada Grafik 4.17 menunjukkan pengaruh perubahan maksimal generasi terhadap nilai *fitness* rata-rata dan nilai *fitness* maksimal pada algoritma genetika. Nilai *fitness* rata-rata dan *fitness* maksimal yang diperoleh pada maksimal generasi 500 dan 1000 hampir sama. Nilai *fitness* terbaik stabil pada ukuran populasi 500 dan 1000 dengan ditunjukkan nilai *fitness* rata-rata tertinggi diperoleh pada kedua maksimal generasi tersebut. Akan tetapi, semakin besar maksimal generasi yang diberikan maka semakin lama waktu yang dibutuhkan untuk proses genetika. Untuk mengefisienkan waktu maka maksimal generasi yang dipilih untuk menghasilkan nilai *fitness* terbaik pada maksimal generasi 500.



Gambar 4.18 Grafik Pengaruh Perubahan Probabilitas *Crossover*

Pada Grafik 4.18 menunjukkan pengaruh perubahan probabilitas *crossover* terhadap nilai *fitness* rata-rata dan nilai *fitness* maksimal pada algoritma genetika. Nilai *fitness* rata-rata dan *fitness* maksimal yang diperoleh pada probabilitas *crossover* 80. Nilai *fitness* terbaik stabil pada probabilitas *crossover* 80 dengan ditunjukkan nilai *fitness* rata-rata tertinggi diperoleh pada probabilitas *crossover* tersebut.



Gambar 4.19 Grafik Pengaruh Perubahan Probabilitas Mutasi

Pada Grafik 4.18 menunjukkan pengaruh perubahan probabilitas mutasi terhadap nilai *fitness* rata-rata dan nilai *fitness* maksimal pada algoritma genetika. Nilai *fitness* rata-rata dan *fitness* maksimal yang diperoleh pada probabilitas mutasi 0.5. Nilai *fitness* terbaik stabil pada probabilitas mutasi 0.5 dengan ditunjukkan nilai *fitness* rata-rata tertinggi diperoleh pada probabilitas *crossover* tersebut.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penelitian yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. *Tree structure modeling* dan algoritma genetika kurang tepat digunakan untuk menyelesaikan kasus desain tata letak ruang dalam rumah yang kompleks.
2. Penggunaan *tree structure modeling* pada kromosom membatasi solusi yang dihasilkan. Dengan menentukan panjang dan lebar ruangan secara spefisik menyebabkan terdapat ruang kosong yang dihasilkan dari proses penggabungan ruangan. Sehingga menyebabkan ketidakefisiensian pada desain tata letak yang dihasilkan.
3. Untuk menghasilkan desain tata letak ruang dalam rumah yang baik, banyak faktor yang perlu diperhatikan seperti *accessway*, koridor, penempatan posisi pintu dan jendela, penempatan letak ruang depan, kanan, kiri, dan belakang di samping kedekatan antar ruang. Sehingga tidak cukup jika hanya memperhatikan kebutuhan ruang dan fungsi ruang saja.
4. Penggunaan parameter genetika yang berbeda menimbulkan perubahan nilai *fitness* yang dihasilkan. Parameter tersebut antara lain ukuran populasi, maksimal generasi, probabilitas *crossover*, dan probabilitas mutasi. Pada hasil percobaan yang telah dilakukan, diperoleh nilai *fitness* terbaik pada ukuran populasi 35, maksimal generasi 500, probabilitas *crossover* 80, dan probabilitas mutasi 0.5.

5.2 Saran

Aplikasi yang dibangun masih jauh dari sempurna. Hal yang dapat bermanfaat untuk mengembangkan aplikasi ini adalah:

1. Disarankan untuk menggunakan metode algoritma lain dalam penyelesaian permasalahan ini seperti algoritma *tabu search*, algoritma semut, atau dengan metode *fuzzy rule* untuk menghasilkan desain tata letak ruang dalam rumah yang lebih baik.

2. *Tree structure modeling* dan algoritma genetika dapat digunakan untuk menyelesaikan kasus lain yang lebih sederhana seperti desain tata letak lahan pertanian atau pengaturan letak mesin.
3. Penambahan metode *compactness*-x dan *compactness*-y pada *tree structure modeling* dapat mengurangi ruang kosong yang dihasilkan.

DAFTAR PUSTAKA

Amin, C., Arifianti F. dan Putri G.P.S., 2010. *101 Denah Rumah*. Penebar Swadaya. Jakarta.

Basuki , Achmad. 2003. *Strategi Menggunakan Algoritma Genetika*. http://www.eepis-its.edu/~basuki/lecture/AlgoritmaGenetika.pdf. Tanggal akses 21 Juli 2010.

Bhowmik, Rekha. 2008. *An Approach to the Facility Layout Design Optimization*. IJCSNS International Journal of Computer Science and Network Security, Vol. 8 No. 4.

Desiani, Anita dan Arhami M., 2006. *Konsep Kecerdasan Buatan*. Penerbit ANDI. Yogyakarta.

Elbeltagi, E., dan Hegazy T., 2003. *Optimum Layout Planning For Irregular Construnction Sites*. Construction Specially Conference of the Canadian Society for Civil Engineering. Kanada.

Garey, M.R. dan Johnson D.S., 1973. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. WH Freeman. New York.

Gen, Mitsuo dan Runwei Cheng. 1997. *Genetic Algorithms and Engineering Design*. John Wiley and Sons. New York.

Gen, Mitsuo dan Runwei Cheng. 2000. *Genetic Algorithms and Engineering Optimization*. John Wiley and Sons. New York.

Goldberg, David Edward. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. John Wiley and Sons. New York.

Hartanti, Murtias Desi. 2007. *Reduksi Data Menggunakan Algoritma Genetika*. Proyek Akhir Jurusan Teknologi Informasi Politeknik Elektronika Negeri Surabaya. Surabaya

Honiden, Terushige. 2004. *Tree Structure Modeling and Genetic Algorithm-based Approach to Unequal-area Facility Layout Problem*. IEMS Vol. 3, No. 2, pp. 123-128.

Kado, Kazuhiro. 1995. *An Investigation of Genetic Algorithms for Facility Layout Problems*. Tesis. Universitas Edinburgh.

Kristiawan, Agung. 2010. *Rumah Idaman 3: Desain dan Bentuk Rumah Minimalis Satu Lantai dan Dua Lantai Lahan Sempit*. Seri Griya. Jakarta.

Kubalik, Jiri dkk. 2002. *Layout Problem Optimization Using Genetic Algorithms*. IFIP Conference Proceedings; Vol. 229. ISBN: -4020-7211-2.

Kusumadewi, Sri. 2003. *Artificial Intelligence Teknik dan Aplikasinya*. Graha Ilmu. Yogyakarta.

Liliana. 2006. *Implementasi Algoritma Genetika untuk Desain Ruang dalam Rumah*. Jurnal Seminar Nasional Sistem dan Informasi. SNSI06-012.

Michalewicz, Zbigniew. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag. New York.

Odop, Nistain. 2010. *22 Desain Rumah Modern Minimalis*. Penerbit Media Pressindo. Yogyakarta.

Obitko, Marek. 1998. *Introduction to Genetic Algorithm*. http://www.obitko.com/tutorials/genetic-algorithms/.Tanggal akses:19 Oktober 2010.

Peres, Jose, dkk. 2004. *Optimal Layout Design for Milk Liveztock Farms Using Genetic Algorithms*. Agricultural Engineering International: the CIGR Journal of Scientific Research and Development. Manuscript BC 03 019. Vol VI. 2004.

Rojas, G.S. dan Torres, J.F., 2004. *Genetic Algorithms for Designing Bank Offices Layouts*. 19th International Conference on Production Research.

Shebanie, Charles Raoul. 2004. An Integrated, Evolutionary Approach to Facility Layout and Detailed Design. Tesis. University of Pittsburgh.

Subakti, Irfan. 2006. *Inteligensia Mesin*. http://is.its-sby.edu/subjects/kbs/Irfan_Sistem_Berbasis_Pengetahuan.pdf. Tanggal akses: 19 Oktober 2010.

Suyanto. 2007. *Artificial Intelligence Searching, Reasoning, Planning dan Learning*. Penerbit INFORMATIKA Bandung.

UNIVERSITAS BRAWIJAYA