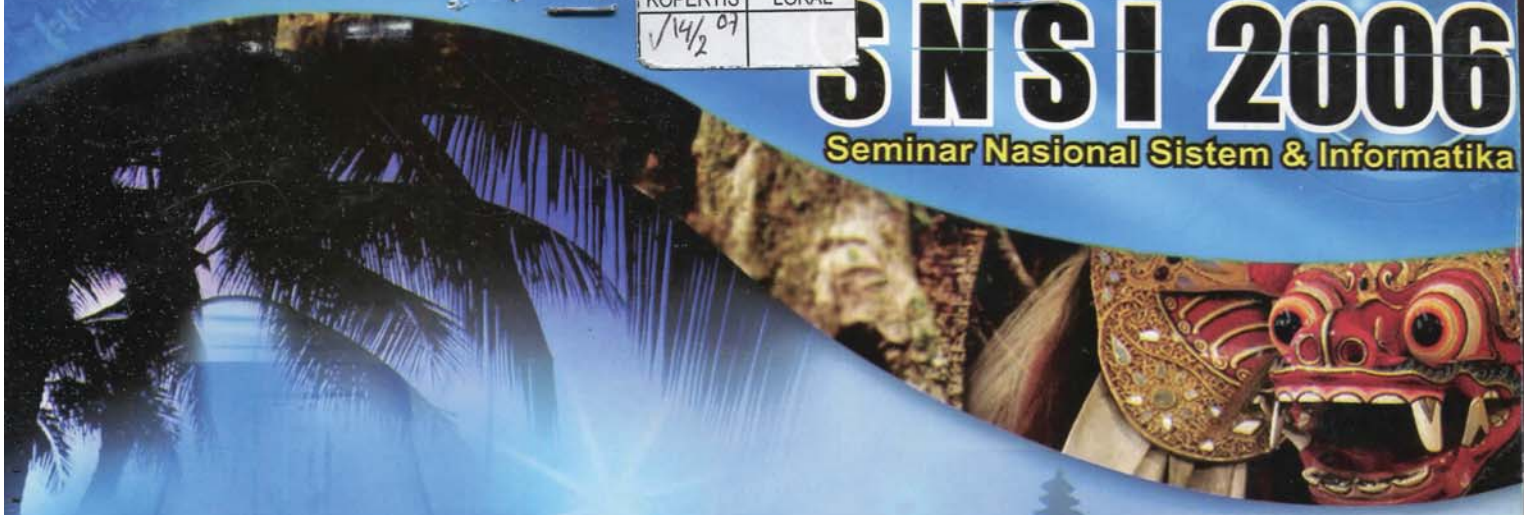
ISSN : 1907-8749

SNSI 2006

Seminar Nasional Sistem & Informatika



PROCEEDINGS

Seminar Nasional Sistem & Informatika 2006

Inna Kuta Beach Hotel

17 November 2006

Editor :

Yudi Agusta, PhD.

STIKOM BALI

DAFTAR ISI

IMPLEMENTASI ALGORITMA GENETIKA UNTUK DESAIN RUANG DALAM RUMAH

Liliana, S.T.
Fakultas Teknik Informatika, Universitas Surabaya
liliana@if.ubaya.ac.id

ABSTRACT
Most property agents sell houses with specific designs. People sometimes want to build houses with their specific design or belief. On the other hand, designer fees are usually too expensive, so people with a limited budget, can not afford to hire one. A solution offered here is to build an application which can show alternative home designs for users. The method used to build this application is Genetic Algorithm (GA), an optimation method for finding some alternative solutions for a specific problem that is being investigated. This method involves notions of some evolution processes in life including mutation, crossover, inheritance and natural selection.
Keywords: *Genethic Algorithm, Home Design.*

1. Pendahuluan

Pada saat membangun rumah, seringkali seseorang dihadapkan pada masalah keterbatasan lahan dan bagaimana desain rumah yang ideal sekaligus nyaman untuk dia dan keluarganya. Meskipun kebanyakan *real estate* saat ini sudah menyediakan rumah dengan desain tertentu, beberapa orang ingin memiliki rumah dengan tata letak ruang yang sesuai dengan kepribadiannya. Untuk mendesain rumah, dibutuhkan desain rumah yang harus dikerjakan oleh arsitek, dan biasanya membutuhkan biaya yang relatif tinggi. Bagi orang yang awam dengan desain rumah dan keterbatasan dana, membayar arsitek untuk desain tata letak ruang rumahnya menjadi beban tersendiri.

Salah satu solusi yang dapat ditawarkan dari permasalahan diatas adalah dengan membuat aplikasi komputer yang dapat menampilkan pilihan solusi desain tata letak ruang dalam suatu lahan dengan ukuran tertentu. Metode yang digunakan untuk membuat aplikasi tersebut adalah menggunakan sistem algoritma genetika.

Algoritma genetika merupakan sebuah metode optimasi yang dapat memberikan alternatif solusi untuk suatu masalah. Metode ini didasari oleh proses evolusi biologi dalam kehidupan, antar lain *mutation*, *crossover*, *inheritance* dan *natural selection*. Menggunakan konsep *survival of the fittest*, Darwin menyatakan bahwa evolusi makhluk hidup terjadi karena adanya seleksi dari alam. Semakin adaptif sifatnya, individu tersebut semakin dapat bertahan dan menghasilkan keturunan. Keturunan yang bertahan dalam proses evolusi memiliki sifat dari induknya, akan tetapi perkawinan antar dua induk dan proses mutasi dapat mengakibatkan perubahan sifat pada keturunan tersebut. Perubahan-perubahan ini diharapkan dapat menghasilkan keturunan yang prima dan dapat bertahan. Dalam penyelesaian masalah, pilihan solusi-solusi diwujudkan dengan keturunan baru dari individu hasil perkawinan dan mutasi. Bila keturunan yang prima dapat dikembangkan, maka solusi yang baik juga dapat diperoleh.

Algoritma Genetika banyak digunakan dalam berbagai bidang bisnis, teknik maupun penelitian. Permasalahan yang dapat diatasi oleh algoritma ini biasanya merupakan masalah yang tidak dapat atau sukar diatasi dengan rumus matematika atau cara konvensional. Dalam penelitian ini, algoritma genetika digunakan untuk memberikan alternatif tata letak ruang dalam sebuah bidang/lahan atau rumah.

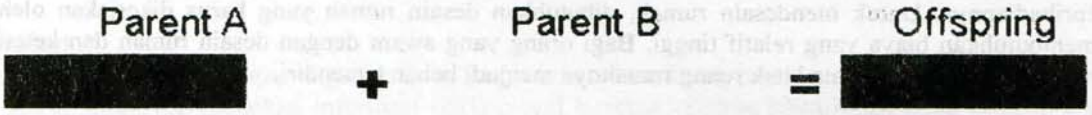
2. Dasar Teori

Ide awal algoritma genetika berasal dari teori Charles Darwin tentang evolusi yang berbasis pada konsep "*survival of the fittest*" yang menyatakan bahwa evolusi jenis-jenis spesies makhluk hidup dan ekosistemnya terjadi karena seleksi alam[2]. Semakin tinggi kemampuan individu untuk beradaptasi, maka semakin tinggi kemungkinan individu tersebut dapat bertahan dan memiliki keturunan. Keturunan dari individu-individu tersebut akan mewarisi sifat-sifat induknya, dimana sifat-sifat tersebut dapat mengalami perubahan yang disebabkan oleh pencampuran sifat kedua induk maupun proses mutasi.

Algoritma Genetika ditemukan pertama kali pada tahun 1960. Algoritma Genetika merupakan salah satu algoritma pemodelan evolusi (*evolutionary modelling*) yang dikembangkan oleh John Holland pada dekade 1960 dan 1970-an dengan tujuan memodelkan perkembangan kemampuan adaptasi sebuah sistem[2]. Algoritma genetika diimplementasikan sebagai simulasi yang berawal dari sebuah populasi yang dihasilkan secara random dan terdiri dari kromosom-kromosom, seperti halnya anggota tubuh makhluk hidup dan merepresentasikan solusi dari masalah. Populasi tersebut akan menghasilkan keturunan populasi yang baru dan diharapkan lebih baik dari populasi sebelumnya. Semakin baik kondisi suatu populasi, semakin besar kemungkinan populasi itu untuk dikembangkan menjadi populasi selanjutnya. Kondisi ini diulangi sampai mendapatkan kondisi yang diharapkan, dengan kata lain solusi terbaik sudah diperoleh[1].

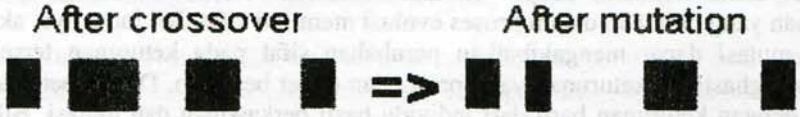
Tahapan Algoritma genetika :

1. Inisialisasi
   Tujuan dari tahap ini adalah untuk menghasilkan suatu populasi awal yang terbentuk dari angka acak.
2. Fitness
   Tahap ini bertujuan untuk menentukan kelayakan tiap kromosom dalam populasi, sesuai atau tidaknya deng keadaan yang diharapkan. Contoh fitness, untuk alasan sanitasi, maka dapur tidak boleh bersebelahan dengan kam mandi, garasi selalu berada di sebelah kanan atau kiri rumah, dan lain sebagainya.
3. New Population
   Membentuk populasi baru yang terbentuk dari populasi sebelumnya, disertai dengan proses-proses genetik samp mendapatkan solusi yang terbaik. Pada proses pembentukan populasi baru dapat terjadi hal-hal seperti dibawah ini:
   a. *Selection*
      Memilih populasi induk yang terbaik untuk dikembangkan menjadi populasi yang baru. Semakin baik kualit populasi, maka semakin besar kemungkinan populasi ini terpilih sebagai induk. Kualitas suatu popula ditentukan dengan sejauh mana populasi tersebut memenuhi kriteria fitness yang telah ditentukan sebelumnya.
   b. *Crossover*
      Melakukan persilangan antar induk untuk mendapatkan keturunan yang baru. Jika tidak ada persilangan indu maka keturunan yang dihasilkan akan memiliki sifat yang sama persis dengan induknya. Contoh dari crossov dapat dilihat di Gambar 2.1[1], dimana Parent A dan Parent B merupakan induk dan Offspring merupakan has perkawinan dua induk, dimana terjadi persilangan antara kedua induk dan menghasilkan warna biru dari Pare A dan warna merah muda dari Parent B.



Gambar 2.1. Contoh Crossover

   c. *Mutation*
      Kemungkinan perubahan sifat diluar sifat induk, pada keturunan yang dihasilkan. Contoh dari mutation dap dilihat di Gambar 2.2[1]. Keturunan yang muncul mengalami mutation, yaitu warna hijau kedua dari ki melebar. Mutation dapat berakibat perubahan menjadi sifat yang lebih baik dari induknya dan sebaliknya, bis berubah menjadi lebih jelek dari induknya, tergantung dari fitness yang ada.



Gambar 2.2. Contoh Mutation

   d. *Accepting*
      Menggantikan populasi induk dengan populasi keturunan baru yang dihasilkan.
4. Test
   Jika kondisi terbaik sudah didapatkan, atau dengan kata lain sudah ada populasi yang sesuai dengan kebutuhan maka proses dihentikan dan solusi terbaik sudah didapatkan. Jika solusi terbaik belum didapatkan, maka proses diulangi tahap menentukan *fitness* (no 2)

3. Analisa

Terdapat beberapa hal yang menjadi pertimbangan dalam mengatur letak ruang dalam rumah. Dari proses analisa, dapa diambil beberapa hal yang sebaiknya dihindari dalam mengatur tata letak ruang dalam rumah, antara lain:

- Kamar mandi sebaiknya tidak diletakkan di bagian depan rumah.
- Dapur tidak terletak di bagian depan atau bagian tengah rumah.
- Halaman tidak terletak di bagian tengah, sebaiknya ada di bagian paling depan atau paling belakang rumah.
- Garasi tidak terletak di bagian tengah atau belakang rumah.
- Kamar mandi tidak terletak di dekat dapur dan ruang makan.
- Ruang makan tidak terletak di dekat kamar mandi.
- Dapur tidak terletak di dekat kamar mandi.

4. Implementasi

4.1. Pembentukan populasi baru

NewPopulation, berisi angka random 0 – 9 dan bertujuan untuk mencari populasi baru yang akan dibentuk dari populasi di generasi sebelumnya dan digunakan untuk menentukan posisi ruangan sesuai dengan permintaan *user*.

Algoritma 4.1.1 Pembentukan NewPopulation

Diulang sebanyak 5 kali (jumlah populasi) (k)
Diulang sebanyak panjang ruangan lantai yang di-generate (i)
Diulang sebanyak lebar ruangan yang di-*generate* (j)
unsur (i,j) pada populasi k = *crossover* dari populasi ke x unsur (a,b) dan ke y - unsur ke (c,d) dari populasi sebelumnya
dimana:
a = Acak angka dari satu sampai panjang ruang
b = Acak angka dari satu sampai lebar ruang
c = Acak angka dari satu sampai panjang ruang
d = Acak angka dari satu sampai lebar ruang
x = Acak angka dari satu sampai empat
y = Acak angka dari satu sampai empat

4.2. Pemeriksaan kelayakan populasi berdasarkan *fitness*

Generate Fitness, bertujuan untuk mencari ruangan yang sesuai dengan permintaan user dan menyesuaikan dengan batasan-batasan yang telah ditentukan, seperti kamar mandi tidak berada di dekat dapur dan ruang makan. Arti kata dekat adalah posisi ruangan-ruangan yang bersangkutan tidak saling berhimpitan satu sama lain. Nilai yang dikirim ke fungsi *Generate Fitness* adalah *Lnt*, yaitu lantai yang akan di-generate. Fungsi ini dipanggil sebanyak jumlah lantai. Berikut ini adalah pengecekan untuk setiap *fitness* yang ada dalam aplikasi.

Algoritma 4.2.1 Pemeriksaan untuk letak kamar mandi

Jika posisi dari batas depan rumah kurang dari 6 kotak/meter, dan hasil *crossover* populasi saat ini = kodeKamarMandi maka ruangan dianggap belum ditemukan

Algoritma 4.2.2 Pemeriksaan untuk letak dapur

Jika posisi dari batas depan rumah kurang dari 9 kotak/meter, dan hasil *crossover* populasi saat ini = kodeDapur maka ruangan dianggap belum ditemukan

Algoritma 4.2.3 Pemeriksaan untuk letak halaman

Jika posisi dari batas depan rumah lebih dari 5 kotak/meter atau dari batas belakang rumah lebih dari 5 kotak/meter, dan hasil *crossover* populasi saat ini = kodeHalaman maka ruangan dianggap belum ditemukan

Algoritma 4.2.4 Pemeriksaan untuk letak garasi

Jika posisi dari batas depan rumah lebih dari 8 kotak/meter dan dari batas kiri lebih dari 8 kotak/meter dan dari batas kanan lebih dari 8 kotak/mter dan, dan hasil *crossover* populasi saat ini = kodeGarasi maka ruangan dianggap belum ditemukan

Algoritma 4.2.5 Pemeriksaan untuk letak kamar mandi, dapur dan ruang makan

Jika hasil *crossover* populasi saat ini = kodeKamarMandi maka
Diperiksa untuk sisi kanan/kiri dari posisi kamar mandi (seluas ukuran yang diminta), jika kanan/kiri = kodeRuangMakan atau kodeDapur, maka ruangan dianggap belum ditemukan

Diperiksa untuk sisi atas/bawah dari posisi kamar mandi (seluas ukuran yang diminta), jika kanan/kiri = kodeRuangMakan atau kodeDapur, maka ruangan dianggap belum ditemukan

Algoritma 4.2.6 Pemeriksaan untuk letak ruang makan dan kamar mandi

Jika hasil *crossover* populasi saat ini = kodeRuangMakan maka
Diperiksa untuk sisi kanan/kiri dari posisi ruang makan (seluas ukuran yang diminta), jika kanan/kiri = kodeKamarMandi, maka ruangan dianggap belum ditemukan

Diperiksa untuk sisi atas/bawah dari posisi ruang makan (seluas ukuran yang diminta), jika kanan/kiri = kodeKamarMandi, maka ruangan dianggap belum ditemukan

Algoritma 4.2.7 Pemeriksaan untuk letak dapur dan kamar mandi

Jika hasil *crossover* populasi saat ini = kodeDapur maka
Diperiksa untuk sisi kanan/kiri dari posisi dapur (seluas ukuran yang diminta), jika kanan/kiri = kodeKamarMandi, maka ruangan dianggap belum ditemukan

Diperiksa untuk sisi atas/bawah dari posisi dapur (seluas ukuran yang diminta), jika kanan/kiri kodeKamarMandi, maka ruangan dianggap belum ditemukan

Jika ruangan ditemukan, proses akan memeriksa apakah dari titik tersebut sampai luas sesuai dengan ukuran ya diminta oleh user tersedia atau tidak. Jika tidak, maka ruangan dianggap tidak ditemukan.

Algoritma 4.2.8 Pemeriksaan untuk luas ruangan

Jika semua fitness terpenuhi maka

Diulang sebanyak posisi panjang dan lebar ruangan

Diperiksa untuk setiap posisi yang sudah terisi, apakah posisi tersebut mencukupi ukura permintaan user. Jika tidak, ruangan dianggap belum ditemukan

Jika ruangan yang diminta tersedia, maka proses mengganti isi temporary sesuai dengan kode ruangan yang dimint seluas permintaan user. Status permintaan tersebut diubah menjadi telah terpenuhi. Jika semua ruangan untuk ko tersebut telah terpenuhi maka proses akan memeriksa apakah untuk kode ruangan yang sama masih terdapat permintaa yang belum terpenuhi, jika tidak proses akan melanjutkan ke kode ruangan yang lain, jika ada, proses akan berpindah k ruangan yang lain dengan kode yang sama.

Algoritma 4.2.9 Pemeriksaan saat ruangan sudah tersedia

Jika ruangan sudah dipenuhi maka

Diulang dari posisi saat ini, ke kanan (untuk lebar ruang) dan ke bawah (untuk panjang ruang) sesua input ukuran ruang, isi semua posisi yang ada didalamnya diganti menjadi kodeRuangYangDiminta

Jika ruangan yang diminta tidak tersedia, maka proses akan memeriksa apakah untuk kode ruangan yang sama masi terdapat permintaan yang belum terpenuhi, jika tidak proses akan melanjutkan ke kode ruangan yang lain, jika ada proses akan berpindah ke ruangan yang lain dengan kode yang sama. Jika ruangan telah terpakai, maka dilakukan proses keluar dari looping terakhir dan pemeriksaan dilanjutkan ke ruangan (p,l) berikutnya.

Proses diatas diulang selama jumlah solusi yang diminta belum dipenuhi, ukuran fitness dianggap belum mencukup ukuran panjang lantai, ukuran lebar lantai dan setiap kodeRuang. Perulangan pertama sebanyak 5 kali, bertujuan untu mencari 5 solusi dari lantai tertentu. Jumlah yang digunakan sesuai dengan permintaan user. Perulangan kedua digunaka sebagai penentuan, nilai *Fitness* dibawah ukuran tertentu dianggap sudah memenuhi syarat dan layak digunakan sebaga solusi. Perulangan ketiga dan keempat digunakan untuk menentukan isi tiap daerah dalam lahan yang tersedia Perulangan terakhir digunakan untuk menelusuri tiap jenis ruangan.

Algoritma 4.2.10 Pemeriksaan untuk tiap bagian dalam ruang

Selama solusi < 5

Selama Fitness > jumlahTertentu

Selama P < PanjangRuang

Selama L < LebarRuang

Selama semua ruang belum dicari

Proses pada algoritma 4.2.10, dilakukan untuk tiap generasi. Pada akhir proses semua solusi yang dihasilkan dimasukka ke *storage* sebagai populasi baru yang akan dikenai *crossover* di proses *NewPopulation*. Kemudian proses diulangi lag untuk solusi berikutnya. Selama proses menentukan letak tiap ruang diatas, dilakukan pencarian solusi yang terbaik untu generasi tersebut, yaitu solusi yang memiliki nilai fitness terkecil. n (sesuai inputan user) solusi yang memiliki fitness yang paling kecil yang akan diberikan pada user.

5. Kesimpulan

Kesimpulan yang didapat dari penelitian ini adalah bahwa algoritma genetika dapat digunakan untuk mencari solusi optimal tanpa harus menggunakan dan mengetahui metode penyelesainnya sendiri, dimana populasi awal yang digunakan terdiri dari sejumlah individu, sedangkan populasi selanjutnya didapatkan dari seleksi, crossover dan mutasi. Hal ini dapat menghindari terbentuknya nilai yang berada di daerah yang tidak optimal.

Daftar Pustaka

[1] Obitko M. (1998) **Genetic Algorithms** http://cs.felk.cvut.cz/~xobitko/ga/ diakses terakhir tanggal 30 Agustus 2006

[2] Situngkir H. (2003) **Algoritma Genetika** http://compsoc.bandungfe.net/intro/part12.html diakses terakhir tanggal 30 Agustus 2006