Pembangkitan Data Uji Menggunakan Algoritma Genetika Multi-populasi Fuzzy Adaptif

Eka Prakarsa Mandyartha
Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Adhi Tama Surabaya
Email: ekaprakarsa@itats.ac.id

Abstract. Test Data Generation using Fuzzy Adaptive Multi-population Genetic Algorithm . *Test data generation techniques based on genetic algorithm has been widely applied. As consequences, the time required in the software testing process can be reduced. The test data is used to detect software defects. This study proposes a genetic algorithm for generating test data to execute all the branches in a program. Control flow graph generated from the program to illustrate the flow of the program code, which contains branches. Branch target selected from sub-populations. Fuzzy adaptive is employed to obtain genetic parameters dynamically based on search conditions. Experimental results show that the proposed method when applied to a set of program which has many branches, better than the multi-population genetic algorithm that genetic parameters are static, in terms of the number of executions and the computation time. If test data can be obtained quickly, then the software defects can be found early.*
Keywords: *multi-population genetic algorithm, adaptive fuzzy, software quality, data test generation, search-based testing*

Abstrak. *Teknik pembangkitan data uji berbasis algoritma genetika telah diaplikasikan secara luas agar waktu yang diperlukan dalam proses pengujian perangkat lunak dapat dikurangi. Data uji digunakan untuk mendeteksi adanya cacat perangkat lunak. Pada penelitian ini diusulkan algoritma genetika sebagai pembangkit data uji untuk mengeksekusi semua cabang dalam sebuah program. Control flow graph dibangkitkan dari sebuah kode program untuk menggambarkan aliran kode program, yang berisi cabang-cabang. Cabang target dipilih dari sub-sub populasi. Fuzzy adaptif digunakan untuk memperoleh parameter genetika secara dinamis berdasarkan kondisi pencarian. Pendekatan algoritma genetika yang diusulkan ini ketika diterapkan pada kumpulan program dengan jumlah cabang yang sangat banyak, telah ditunjukkan secara eksperimental bahwa lebih baik, dalam hal jumlah eksekusi dan waktu komputasi, dibandingkan dengan algoritma genetika multi-populasi yang parameter genetikanya bersifat statis. Dengan data uji yang dapat diperoleh secara cepat maka cacat perangkat lunak dapat ditemukan lebih dini.*
Kata Kunci: *algoritma genetika multi-populasi, fuzzy adaptif, kualitas perangkat lunak, pembangkitan data uji, pengujian berbasis pencarian*

1. Pendahuluan

Pengujian perangkat lunak adalah alat jamin yang pertama digunakan untuk mengkontrol kualitas dari produk perangkat lunak sebelum perangkat lunak itu diterima oleh pelanggan (Galin, 2004). Dalam pengembangan perangkat lunak, proses ini merupakan komponen yang mahal dan proses yang memakan waktu (Alshraideh, dkk, 2011) (Maragathavalli, dkk, 2012) (Doungsa-ard, dkk, 2007) (Edvardsson, 1999) (Chen, dkk, 2008). Dengan teknik pembangkitan data uji berbasis algoritmik, pengujian perangkat lunak dapat dilakukan lebih efisien yaitu waktu yang dibutuhkan untuk tugas ini dapat dikurangi, sehingga biaya juga dapat dikurangi dan meningkatkan kualitas produk akhir. Teknik pembangkitan data uji bertujuan mencari data masukan program yang akan memenuhi kondisi cabang target tertentu (Doungsa-ard, dkk, 2007). Data masukan yang diperoleh dapat digunakan untuk mengevaluasi cakupan cabang target tersebut yaitu sesuai dengan spesifikasi atau tidak. Dengan kata lain, cacat dalam kode sebuah program dapat ditemukan pula.

Ada tiga metode pendekatan pada pengujian perangkat lunak (Alshraideh, dkk, 2011) yaitu teknik *random* (data uji dibangkitkan secara acak hingga memenuhi domain variabel input secara keseluruhan), teknik statis (program yang diuji tidak dieksekusi), dan teknik dinamis (program yang

diuji dieksekusi). Teknik dinamis yang merupakan teknik pembangkitan data uji terbaru, untuk mencapai cabang target tertentu, mengikuti dua pendekatan. Pendekatan pertama yaitu memodelkan sistem dengan pertidaksamaan liner yang diselesaikan dengan teknik tertentu dan pendekatan kedua menggunakan teknik pencarian metaheuristik. Pencarian metaheuristik memodelkan masalah pembangkitan data uji sebagai masalah optimasi yaitu pencarian data uji yang optimal. Data uji yang optimal yaitu data uji yang mencakup semua cabang sehingga cabang target tereksekusi.

Evolutionary search merupakan teknik pencarian heuristik yang saat ini sedang diteliti untuk pembangkitan data uji. Penelitian oleh (Alshraideh, dkk, 2011), (Maragathavalli, dkk, 2012), (Doungsa-ard, dkk, 2007), (Andreou, dkk 2007), dan (Chen, dkk, 2008) menggunakan algoritma genetika, yang merupakan salah satu teknik *evolutionary search*, untuk membangkitkan data uji secara otomatis. Penelitian yang dilakukan oleh (Maragathavalli, dkk, 2012), (Doungsa-ard, dkk, 2007), dan (Andreou, dkk, 2007) menggunakan algoritma genetika *panmictic* atau biasa dikenal dengan algoritma genetika konvensional. Algoritma genetika konvensional memodelkan solusi sebagai populasi tunggal (*single population*). Sedangkan penelitian Alshraideh(Alshraideh, dkk, 2011), dan (Chen, dkk, 2008) menggunakan algoritma genetika dengan multi-populasi (populasi terdiri dari sub-sub populasi) sebagai sekumpulan solusinya. Algoritma genetika jenis tersebut dikenal dengan algoritma genetika paralel. Pembangkitan data uji berbasis algoritma genetika lebih baik dibanding dengan teknik random dan pencarian lokal (*local search*) (Chen, dkk, 2008). Namun pencarian menggunakan algoritma genetika dengan populasi tunggal menimbulkan masalah baru yaitu solusi yang terjebak pada lokal optimum (Alshraideh, dkk, 2011). Untuk mengatasi masalah tersebut, maka dikembangkan algoritma genetika paralel yaitu algoritma genetika yang membagi populasi menjadi sub-sub populasi. Hal ini disebabkan dengan sub-sub populasi yang masing-masing berevolusi, maka lebih banyak kombinasi solusi yang dihasilkan daripada hanya populasi tunggal.

Penelitian oleh (Maeda, dkk, 2006) menemukan bahwa penggunaan metode paralel algoritma genetika juga menimbukan beberapa masalah. Masalah yang paling penting yaitu hasil pencarian yang tidak selalu optimal. Hal ini disebabkan oleh parameter algoritma genetika yang tetap (*crossover rate, mutation rate*) juga *migration rate* yang konstan. Pengaturan parameter seperti *crossover rate* dan *mutation rate* mempengaruhi performa algoritma genetika.

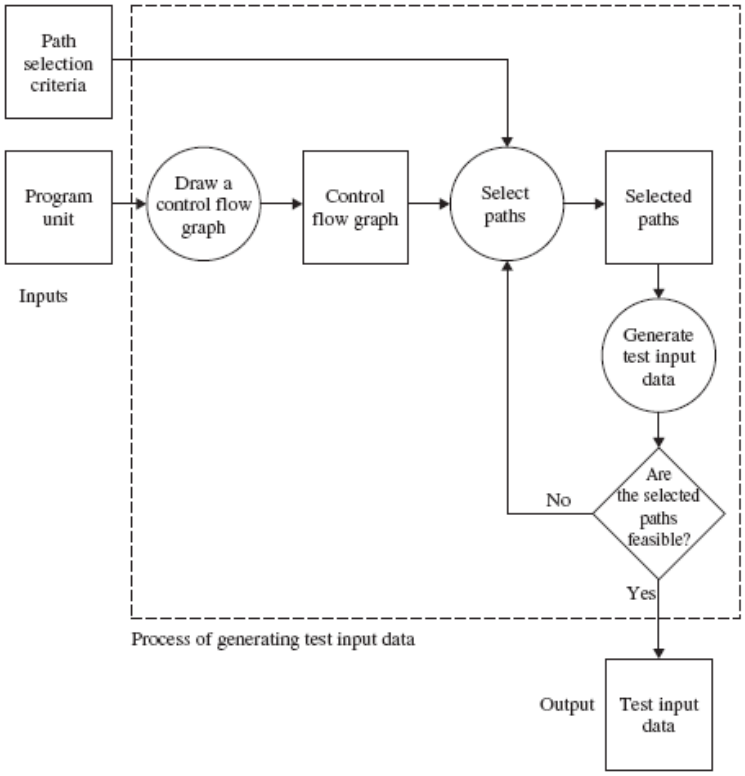
Tujuan penelitian ini adalah mencari data masukan program sehingga mengeksekusi cabang target tertentu yang didasarkan pada pencarian data masukan yang memenuhi kondisi *control dependency* cabang tertentu. Algoritma genetika multi-populasi digunakan dalam proses pencarian. Pada algoritma genetika multi-populasi dilakukan *tuning* terhadap parameter-parameter genetika seperti *crossover rate, mutation rate* serta *migration rate* menggunakan penalaran fuzzy agar dapat dihasilkan nilai parameter yang tepat, bersifat dinamis. Algoritma genetika multi-populasi fuzzy adaptif ini dapat disebut juga *fuzzy adaptive paralel genetic algorithm* atau FAPGA.

1.1. Metode

Desain algoritma untuk pembangkitan data uji secara umum dapat dilihat pada gambar 1. Pada desain algoritma terdapat tahapan-tahapan yang harus dilalui untuk menghasilkan data uji. Masukan (*input*) berupa fungsi dari program kemudian dianalisis menggunakan *control flow graph* (CFG) yang menggambarkan aliran (*path*) kode program. Setelah aliran kode program diketahui kemudian dibangkitkan data uji dengan metode algoritma genetika multi-populasi fuzzy adaptif. Jalur-jalur yang tergambar pada CFG dapat digunakan untuk menentukan jalur yang harus dipenuhi agar sebuah cabang target dapat dipenuhi (dieksekusi).

Unit Program

Pembangkitan data uji pada *white-box testing* (pengujian berbasis struktural atau berbasis kode program) merupakan proses untuk mendapatkan masukan program dimana elemen yang dipilih akan dieksekusi. Seperti contoh pada kode program berikut, data masukan (data uji) adalah variabel masukan fungsi yaitu x, y dan z. Untuk mengeksekusi target, nilai x, y, dan z yang dibangkitkan harus memenuhi cabang target "*if(y==z)*". Gambar 2 mengilustrasikan program yang berisi cabang.



Gambar 1. Desain metodologi pembangkitan data uji secara umum (Alshraideh, dkk, 2011)

```
void contoh(int x,int y,int z)
{
  if(x <= y)        (1)
  {
    if(y==z)        (2)
    {
      //Target
  }
}
```

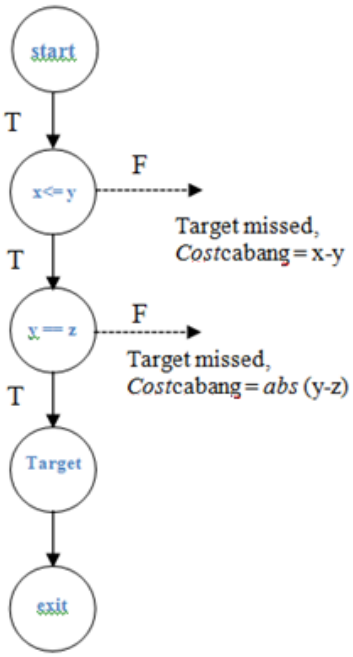
Gambar 2. Ilustrasi program yang berisi cabang

Control Flow Graph (CFG)

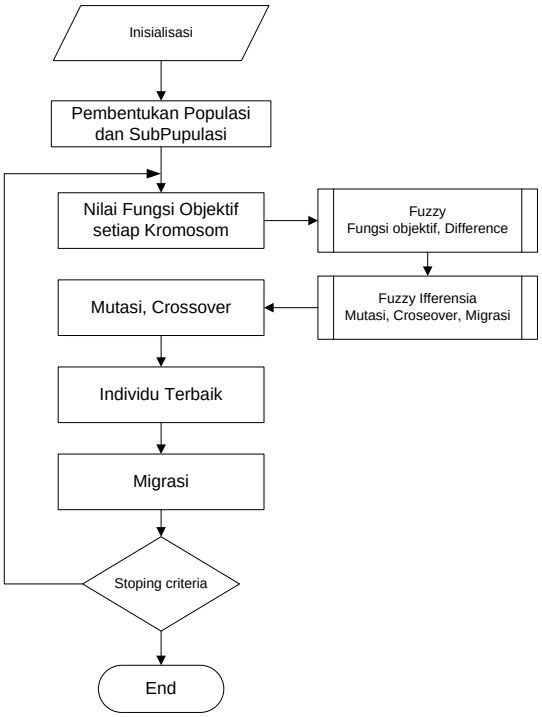
Fungsi utama dari CFG adalah mengambarkan aliran kode program mulai awal program hingga akhir program. CFG dari fungsi program pada gambar 2 digambarkan oleh gambar 3. Berdasarkan jalur yang tergambar pada CFG tersebut, untuk mengeksekusi cabang target yaitu "*if(y==z)*" maka cabang "*if(x<=y)*" harus dipenuhi, artinya cabang "*if(y==z)*" *control dependence* terhadap cabang "*if(x<=y)*". "y==z" dan "x<=y" disebut juga predikat cabang.

Pembangkitan Data Uji

Langkah-langkah pembangkitan data uji menggunakan algoritma genetika multi-populasi fuzzy adaptif digambarkan pada gambar 4.



Gambar 3. CFG sebuah program



Gambar 4. Diagram alir pembangkitan data uji menggunakan algoritma genetika multi-populasi fuzzy adaptif

2.1. Inisialisasi

Pada tahap ini dilakukan inisialisasi parameter-paremeter yang akan digunakan dalam algoritma genetika, yaitu ukuran populasi, jumlah generasi, metode seleksi, ukuran sub populasi, *migration rate*, termasuk inisialisasi parameter genetika (*crossover rate* dan *mutation rate*). Ukuran populasi adalah jumlah individu (solusi) yang diinginkan; jumlah generasi adalah maksimum generasi (iterasi); metode seleksi adalah metode yang digunakan untuk memilih individu sebagai parent yang akan melakukan proses kawin-silang (*crossover*), misalnya metode

roulette-wheel; ukuran sub populasi adalah jumlah sub populasi (*island*); sedangkan *migration rate* menentukan peluang seberapa banyak individu dalam tiap sub populasi saling bertukar dengan sub populasi lainnya. Parameter genetika *crossover rate* menentukan peluang seberapa banyak *parent* untuk melakukan kawin-silang, sedangkan *mutation rate* menentukan peluang seberapa banyak *offspring* (individu hasil kawin-silang) bermutasi.

2.2. Pembentukan populasi dan sub populasi

Berbeda dengan algoritma genetika konvensional yang menggunakan populasi tunggal, algoritma genetika multi-populasi mendefinisikan populasi sebagai kumpulan dari sub-sub populasi. Sebagai contoh, misalkan jumlah populasi ditentukan 20 individu dan ada 8 sub populasi, berarti tiap sub populasi memiliki 20 individu sehingga total individu adalah 160 (20 dikali 8). Setiap sub populasi melakukan proses pembentukan *offspring* baru secara independen terhadap sub populasi lainnya. Oleh karena itu, dinamakan algoritma genetika paralel (*parallel genetic algorithm*). Meskipun proses algoritma genetika paralel dilakukan menggunakan pemrosesan komputasi paralel (melibatkan beberapa prosesor), namun pemrosesan komputasi paralel secara serial (hanya melibatkan satu prosesor) atau pseudo-paralel dapat pula dilakukan. Penelitian ini menggunakan metode pseudo-paralel. Kedua pemrosesan tersebut sama-sama memberikan solusi optimal dengan reduksi jumlah eksekusi yang dibutuhkan untuk mencapai solusi yang konvergen dan reduksi waktu komputasi dibandingkan dengan algoritma genetika populasi tunggal.

2.3. Fungsi objektif

Pada algoritma genetika untuk mendeskripsikan solusi yang dihasilkan baik atau jelek, digunakan nilai objektif. Masing-masing solusi dihitung nilai objektifnya. Nilai objektif merupakan keluaran dari fungsi objektif. Nilai objektif ini pula yang menggambarkan *fitness* sebuah solusi atau biasa disebut juga nilai *fitness*, sehingga fungsi objektif dapat disebut juga fungsi *fitness*.

Fungsi objektif yang digunakan pada kasus pembangkitan data uji ini adalah fungsi *cost* yang diusulkan oleh (Bottaci, 2003). Fungsi *cost* merupakan fungsi pinalti yang merepresentasikan jarak cabang (*branch distance*) yang akan dihitung bila predikat cabang tersebut tidak terpenuhi. Tabel 1 menunjukkan fungsi *cost* terhadap pemenuhan ekspresi predikat dengan *a*, *b* adalah bilangan riil dan *k* adalah bilangan riil positif. “Predikat A AND predikat B” “(A&&B)” dan “Predikat A OR predikat B” “(A||B)” adalah ekspresi predikat operasi logika.

Tabel 1. Fungsi Cost Ekspresi Predikat Cabang

Ekspresi Predikat	Cost bila tidak memenuhi Ekspresi Predikat
$a \leq b$	$a - b$
$a < b$	$a - b + k$
$a = b$	$abs\ (a-b)$
$a \neq b$	$k - abs\ (a-b)$
$a \geq b$	$b - a$
$a > b$	$b - a + k$
$A \&\& B$	$max(cost(A), cost(B))$
$A \| B$	$min(cost(A), cost(B))$

Ilustrasi penghitungan fungsi *cost* untuk kode program gambar 2 sebagai berikut: Misalkan algoritma genetika menghasilkan solusi x = 2, y = 3 dan z = 5. Untuk mengeksekusi target maka cabang target “*if(y==z)*” harus dipenuhi. Agar cabang target “*if(y==z)*” dieksekusi maka cabang target “*if(x<=y)*” harus dipenuhi. Karena ekspresi predikat “*x<=y*” terpenuhi, “2<=3” bernilai *true*, maka tidak dikenakan fungsi pinalti atau nilai fitnessnya 0. Pada ekspresi predikat “*y==z*”, “3==5” bernilai *false*, sehingga dihitung fungsi *cost*nya sebagai *abs*(3-5). Total nilai *fitness*nya adalah 0 + 2 = 2.

Algoritma genetika akan mencari kandidat solusi yang memiliki nilai *fitness* minimum, karena bila semua cabang yang menyebabkan cabang target dieksekusi dan cabang target itu

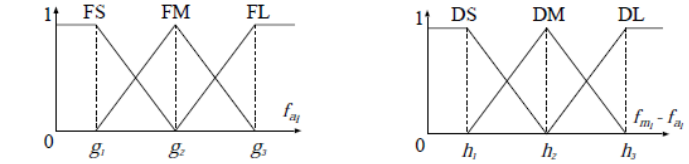
sendiri juga dieksekusi maka nilai *fitness*nya adalah 0, artinya nilai *fitness* minimum memberikan data uji yang mendekati eksekusi cabang target. Permasalahan ini merupakan permasalahan optimasi untuk kasus minimisasi.

f_{mi} - f_{ai} \ f_{ai}	FS	FM	FL
DS	CVS MVL EVL	CLS MLL ELL	CLL MLS ELS
DM	CS ML EL	CM MM EM	CL MS ES
DL	CLS MLL ELL	CLL MLS ELS	CVL MVS EVS

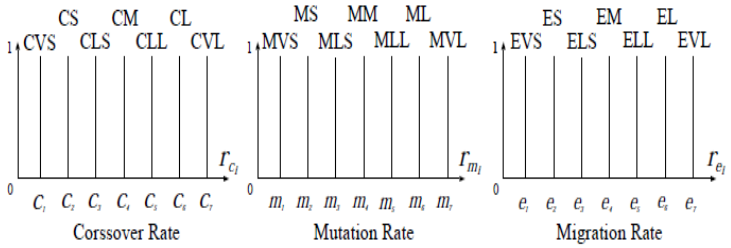
Gambar 5. Aturan fuzzy untuk menentukan nilai parameter probabilitas crossover, probabilitas mutasi dan migration rate (Maeda, dkk, 2006)

2.4. Fuzzy adaptif

Metode algoritma genetika multi-populasi fuzzy adaptif (Maeda, dkk, 2006) merupakan teknik adaptif, dengan parameter genetika, probabilitas *crossover* (*crossover rate*) dan mutasi (*mutation rate*), ditentukan berbasis penalaran fuzzy sesuai tahapan pencarian. Pada tahap awal, probabilitas *crossover* harus bernilai kecil sedangkan probabilitas mutasi harus bernilai besar sehingga menghasilkan solusi yang beragam. Namun, pada tahap akhir, probabilitas mutasi harus bernilai kecil untuk menghindari hilangnya individu unggul sedangkan probabilitas *crossover* harus bernilai besar untuk mendapatkan individu unggul secara cepat.



(a)



(b)

Gambar 6 (a) Fungsi keanggotaan fuzzy pada bagian anteseden (b) Fungsi keanggotaan fuzzy pada bagian implikasi (Maeda, dkk, 2006)

Parameter genetika pada metode algoritma genetika multi-populasi fuzzy adaptif yaitu probabilitas *crossover* (rc_i), probabilitas mutasi (rm_i) dan *migration rate* (re_i) ditentukan dari aturan fuzzy. Penalaran fuzzy berdasarkan dua variabel yaitu nilai fitness rata-rata (fa_i) dan selisih antara nilai *fitness* maksimum dengan nilai *fitness* rata-rata (fm_i-fa_i) pada masing-masing *island i* atau sub-populasi *i*. Berdasar pada dua variabel tersebut dapat diketahui bahwa keadaan masing-masing *island* berada pada tahap awal atau tahap akhir. Aturan fuzzy dengan tiga parameter, probabilitas *crossover*, probabilitas mutasi dan *migration rate* dibentuk berdasarkan nilai fa_i dan (fm_i - fa_i) yang ditunjukkan oleh gambar 5.

Pada gambar 5, FS (fa_i *small*) adalah nilai *fitness* rata-rata sebuah *island* yang bernilai kecil dan ini berarti *island* tersebut berada pada tahap awal proses pencarian. FL (fa_i *large*)

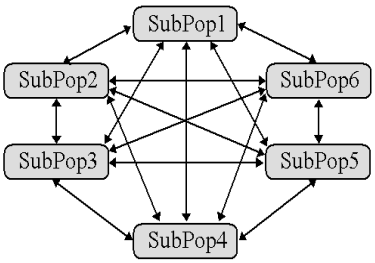
adalah nilai fitness rata-rata sebuah *island* yang bernilai besar dan mengandung arti bahwa *island* tersebut berada pada tahap akhir proses pencarian. Sedangkan DS (fm_i - fa_i *small*), adalah fm_i - fa_i pada *island*, yang bernilai kecil mempunyai implikasi bahwa individu-individu pada *island* tersebut agak kompak. DL merupakan kebalikannya. Berdasarkan perbedaan keadaaan FS, FM, FL DS dan DL, maka parameter-parameter dapat dibentuk misalnya CVS (rc_i *very small*), MVL (rm_i very large), dan EVL (re_i *very large*) adalah parameter yang dibentuk pada kondisi FS dan DS. Dengan metode ini, permasalahan, yaitu kemampuan pencarian pada algoritma genetika tidak selalu optimal terutama pada tahap awal dan akhir proses pencarian, dapat diselesaikan.

Pada algoritma genetika multi-populasi fuzzy adaptif, rc_i, rm_i dan re_i tidak bernilai tetap. Nilai parameter tersebut pada masing-masing *island* ditentukan menggunakan inferensi fuzzy. Fungsi keanggotaan fuzzy pada anteseden (*IF part*) disusun dari fa_i dan (fm_i - fa_i). Algoritma genetika multi-populasi fuzzy adaptif menambahkan parameter yang disebut *migration rate* pada bagian implikasi (*THEN part*). Fungsi keanggotaan anteseden dan implikasi ditunjukkan gambar 6.

Karena proses fuzzifikasi nilai *fitness* (F) dan selisih nilai *fitness* maksimum dan nilai *fitness* rata-rata (D) bersifat dinamis, parameter g_1, g_2, g_3, h_1, h_2, h_3 pada fungsi keanggotaan fuzzy berbeda tiap generasi, maka nilai parameter tersebut ditentukan menggunakan variable statistik, kuartil Q1 (25% persentil), kuartil Q2 (50% persentil), dan kuartil Q3 (75% persentil) dari nilai *fitness* suatu generasi. Nilai g_1 dan h_1 menggunakan nilai Q1. Nilai g_2 dan h_2 menggunakan nilai Q2. Nilai g_3 dan h_3 menggunakan Q3.

2.5. Migrasi

Pada migrasi terjadi pertukaran individu antar sub populasi yang bertujuan agar keberagaman genetik terjadi. Pemilihan individu yang akan bermigrasi dapat dilakukan secara acak atau berdasarkan nilai *fitness*nya (individu dengan nilai *fitness* terbaik dipilih). Skema topologi migrasi paling banyak digunakan adalah topologi *complete net*. Pada topologi *complete net*, setiap individu dapat bermigrasi ke sub populasi lainnya secara bebas (*unrestricted migration*). Gambar 7 menunjukkan skema topologi *unrestricted migration*.



Gambar 7. Skema topologi migrasi complete net (*unrestricted migration*) (Alshraideh, dkk, 2011)

3. Hasil Penelitian dan Pembahasan

3.1. Hasil Penelitian

Pengujian dilakukan dengan membangkitan data uji pada tiga program yang berbeda (tabel 2) menggunakan algoritma genetika multipopulasi dan algoritma genetika multi-populasi fuzzy adaptif. Parameter pengaturan algoritma genetika ditunjukkan tabel 3. Jumlah eksekusi yang dibutuhkan untuk mencapai cabang target, dan waktu eksekusi dievaluasi pada tiga program tersebut.

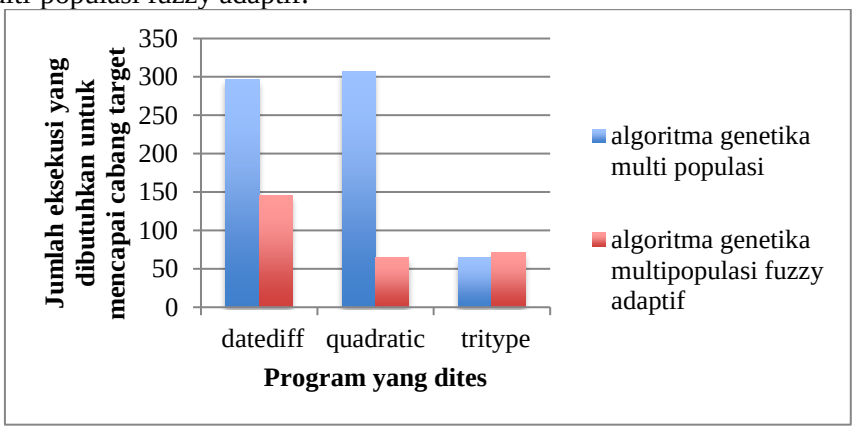
Tabel 2. Program Untuk Pembangkitan Data Uji

Nama Program	Jumlah Baris Kode	Total Cabang	Deskripsi Program
datediff	225	16	Program yang menentukan jumlah hari di antara dua tanggal
quadratic	195	11	Program yang menentukan akar dari persamaan kuadrat
tritype	88	10	Diberikan tiga variabel yang merupakan panjang sebuah segitiga, program menentukan tipe segitiga tersebut

Tabel 3. Pengaturan Parameter Algoritma Genetika

Parameter	Algoritma Genetika Multipopulasi	Algoritma Genetika Multipopulasi Fuzzy Adaptif
Ukuran populasi	20	20
Jumlah generasi	1200	1200
Metode seleksi	Roulette Wheel	Roulette Wheel
Probabilitas crossover (crossover rate)	-	ditentukan dengan inferensi fuzzy
Probabilitas mutasi (mutation rate)	1/*L*	ditentukan dengan inferensi fuzzy
Ukuran island (island size)	8	8
Skema migrasi	Complete net	Complete net
Migration rate	0.2	ditentukan dengan inferensi fuzzy

Perbandingan jumlah eksekusi yang dibutuhkan untuk mencari data uji sehingga mencapai cabang target menggunakan algoritma genetika multi-populasi dan algoritma genetika fuzzy adaptif ditunjukkan pada gambar 8. Gambar 8 menggambarkan bahwa algoritma genetika multi-populasi fuzzy adaptif lebih baik dibandingkan algoritma genetika multi-populasi. Algoritma genetika multipopulasi membutuhkan 296 eksekusi untuk mencapai cabang target saat eksekusi program *datediff*, sedangkan 145 eksekusi diperlukan bila menggunakan algoritma genetika multi-populasi fuzzy adaptif.



Gambar 8. Perbandingan jumlah eksekusi yang dibutuhkan untuk mencapai cabang target pada program yang dites dengan algoritma genetika multi-populasi dan algoritma genetika multi-populasi fuzzy adaptif

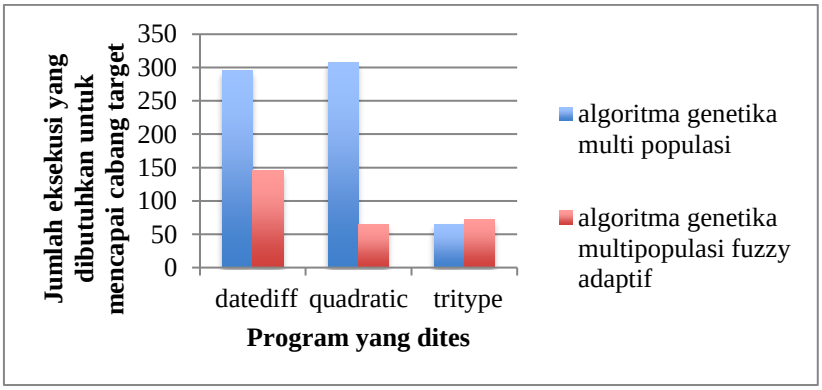
Waktu eksekusi yang diperlukan oleh program yang berbeda (tabel 2) menggunakan algoritma genetika multi-populasi fuzzy adaptif lebih sedikit dibandingkan algoritma genetika multi-populasi, yaitu rata-rata 0.24 lebih cepat. Hal ini disebabkan algoritma genetika multi-populasi fuzzy adaptif memerlukan jumlah eksekusi yang lebih sedikit untuk mencapai cabang target.

Tabel 4. Waktu Uji

	Human Time (manual)	Machine Time (algoritma genetika)
datediff	1503 menit	8.483 detik
quadratic	70 menit	7.005 detik
tritype	609 menit	6.764 detik

Tabel 4 menunjukkan waktu yang diperlukan oleh manusia (*human time*) (DeMillo, dkk, 1993) dan algoritma genetika (*machine time*) untuk membangkitkan data uji. Kolom

human time menunjukkan waktu (dalam menit) yang diperlukan penguji tenaga manusia (*human tester*) untuk membuat data uji, sedangkan *machine time* adalah waktu (dalam detik) yang diperlukan oleh algoritma genetika untuk mengeksekusi semua kasus uji.



Gambar 9 Perbandingan waktu eksekusi yang dibutuhkan untuk pembangkitan data uji pada program yang dites dengan algoritma genetika multi-populasi dan algoritma genetika multi-populasi fuzzy adaptif

Berdasarkan tabel 4, perbedaan waktu eksekusi antara *human time* dan *machine time* sangat dramatis. Waktu yang diperlukan algoritma genetika untuk pembangkitan data uji relatif konstan, sedangkan waktu yang dibutuhkan manusia relatif meningkat sejalan dengan kompleksitas kode program.

3.2. Jaminan Kualitas Perangkat Lunak dan Kecepatan Pengembangan

Bila cacat perangkat lunak dapat ditemukan lebih dini, akan bermanfaat secara signifikan pada jadwal pengembangan perangkat lunak. Hasil penelitian menemukan bahwa pengerjaan ulang spesifikasi, desain, dan kode perangkat lunak memerlukan biaya 40 hingga 50 persen dari total biaya pengembangan perangkat lunak (Jones, 1986). Secara praktis, setiap jam yang digunakan untuk mencegah adanya cacat perangkat lunak dapat mengurangi waktu perbaikan sekitar tiga hingga sepuluh jam. Potensi penghematan biaya dari deteksi cacat perangkat lunak lebih dini sangat besar, 60 persen dari cacat perangkat lunak biasanya terdapat pada desain (Tom, dkk, 1988).

Dengan teknik pembangkitan data uji lebih cepat, maka cacat perangkat lunak dapat dideteksi lebih cepat pula. Sebagai contoh, pembangkitan data uji bertujuan untuk mencari data masukan program sehingga target dieksekusi. Setelah data uji diperoleh, maka pengembang perangkat lunak dapat mengetahui bahwa alur kode program yang diimplementasikan sesuai target atau tidak, dengan kata lain dapat diketahui cacat kode perangkat lunak. Cacat kode perangkat lunak disebabkan oleh berbagai hal yaitu kesalahan penulisan sintaks, kesalahan logika, dan juga skenario terburuk adalah kesalahan pengembang perangkat lunak dalam menginterpretasikan spesifikasi dan desain. Tidak tertutup kemungkinan, terdapat cacat pula pada desain dan spesifikasi perangkat lunak, sehingga hasil keluaran implementasi kode program tidak seperti yang diharapkan.

4. Kesimpulan

Pendekatan algoritma genetika multi-populasi fuzzy adaptif diusulkan untuk menghindari lokal optimum dan konvergensi solusi yang lebih cepat pada pencarian data uji. Pembangkitan data uji didasarkan pada cakupan cabang target sehingga data masukan harus mencakup cabang target tersebut.

Tiga buah program sampel, menggunakan algoritma genetika multipopulasi dan algoritma genetika multi-populasi fuzzy adaptif, telah dibandingkan dan dievaluasi dengan metrik, jumlah eksekusi yang dibutuhkan untuk mencakup cabang target dan waktu eksekusi. Hasil eksperimen menunjukkan bahwa algoritma genetika multi-populasi fuzzy adaptif lebih

baik dibandingkan algoritma genetika multi-populasi dalam pembangkitan data uji. Sebagai contoh, peningkatan waktu eksekusi sebesar 0.24 kali lebih cepat dibandingkan algoritma genetika multipopulasi.

Performa pembangkitan data uji yang dilakukan manusia dan algoritma dibandingkan pula. Hasil menunjukkan bahwa waktu yang dibutuhkan manusia (*human time*) jauh lebih lama (menit) untuk memilih data uji yang tepat dibandingkan waktu yang dibutuhkan algoritma (*machine time*), yaitu hanya beberapa detik saja. Peningkatan waktu ini berakibat pada rendahnya biaya yang dibutuhkan dalam proses pengujian perangkat lunak sekaligus penemuan cacat perangkat lunak yang lebih cepat pula sehingga jadwal pengembangan perangkat lunak dapat dilaksanakan tepat waktu namun kualitasnya juga tetap terjamin.

Referensi

Alshraideh, M., Mahafzah, B. A., & Al-Sharaeh, S. (2011). A multiple-population genetic algorithm for branch coverage test data generation. *Software Quality Journal*, *19*(3), 489-513.

Andreou, A. S., Economides, K. A., & Sofokleous, A. A. (2007, October). An automatic software test-data generation scheme based on data flow criteria and genetic algorithms. In *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on* (pp. 867-872). IEEE.

Bottaci, L. (2003, July). Predicate expression cost functions to guide evolutionary search for test data. In *Genetic and Evolutionary Computation Conference* (pp. 2455-2464). Springer Berlin Heidelberg.

Chen, Y., & Zhong, Y. (2008, October). Automatic path-oriented test data generation using a multi-population genetic algorithm. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on* (Vol. 1, pp. 566-570). IEEE.

DeMillo, R. A., & Offutt, A. J. (1993). Experimental results from an automatic test case generator. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *2*(2), 109-127.

Doungsa-ard, C., Dahal, K., Hossain, A., & Suwannasart, T. (2007, August). Test data generation from UML state machine diagrams using gas. In *Software Engineering Advances, 2007. ICSEA 2007. International Conference on* (pp. 47-47). IEEE.

Edvardsson, J. (1999, October). A survey on automatic test data generation. In *Proceedings of the 2nd Conference on Computer Science and Engineering* (pp. 21-28).

Galin, D. (2004). *Software quality assurance: from theory to implementation*. Pearson Education India.

Jones, C. (1986). *Tutorial programming productivity: issues for the eighties*. IEEE Computer Society Press.

Maeda, Y., Ishita, M., & Li, Q. (2006). Fuzzy adaptive search method for parallel genetic algorithm with island combination process. *International Journal of Approximate Reasoning*, *41*(1), 59-73.

Maragathavalli, P., Kanmani, S., Kirubakar, J. S., Sriraghavendrar, P., & Prasad, A. S. (2012, March). Automatic program instrumentation in generation of test data using genetic algorithm for multiple paths coverage. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on* (pp. 349-353). IEEE.

Tom, G., & Finzi, S. (1988). Principles of software engineering management. *Workingham, England: Addison-Wesley*.