***Jurnal Online tentang Ilmu Komputer dan Teknologi Informasi (OJCSIT) Vol.*** ***(2)*** ***-*** ***No. (1)***  ***Nomor Referensi: W11-0064 115*** **Desain Beberapa Jalur Uji Data Generator**  **Berdasarkan Meta-Heuristik**   
***Abstrak*** **- Kompleksitas sistem perangkat lunak telah meningkat secara dramatis dalam dekade terakhir.** **Pengujian perangkat lunak sebagai komponen padat karya menjadi semakin mahal.** **Secara khusus, biaya pengujian mencapai hingga 50% dari total biaya pengembangan perangkat lunak.**  **Mencari kesalahan perangkat lunak, dengan menggunakan metode berbasis evolusi seperti algoritma genetika dan meta-heuristik lainnya adalah salah satu upaya menuju tujuan ini.** **Bagian pengujian perangkat lunak yang paling sulit adalah generasi data uji.**  **Saat ini, proses ini terutama proses manual.** **Otomatisasi pembuatan data uji secara signifikan dapat mengurangi total biaya pengujian perangkat lunak dan siklus pengembangan perangkat lunak secara umum.** **Uji data generasi kira-kira dapat mencapai 40% dari total biaya pengujian perangkat lunak.** **Otomatisasi akan meningkatkan keandalan software** **yang akan meningkatkan pengembang dan pelanggan** **kepercayaan dalam sistem perangkat lunak yang dikembangkan.**  **Pendekatan yang ada dari data uji otomatis menghasilkan beberapa keberhasilan dengan menggunakan evolusi**  **Algoritma perhitungan** **Makalah ini menggunakan algoritma genetika untuk mengotomatisasi perolehan data uji yang diharapkan**  **Mencapai tujuan yang diinginkan secara lebih efisien.**  **Kata kunci - Algoritma Genetika, Generasi Uji Otomatis, Algoritma Genetika Operator, Perangkat Lunak**  **Pengujian.** I. PENDAHULUAN   
Komputer dan program telah diperluas di banyak bagian kehidupan kita. Pengujian perangkat lunak adalah cara untuk memverifikasi kebenaran dan kesesuaian sistem perangkat lunak untuk memastikan bahwa program memenuhi spesifikasi. Sementara pengujian perangkat lunak sangat signifikan, ini juga sangat mahal, kerja yang melelahkan dan menyita waktu. Tujuan pengujian perangkat lunak adalah untuk menghasilkan serangkaian jumlah kasus uji minimal sehingga mengungkapkan sebanyak mungkin cacat dengan memenuhi kriteria tertentu yang disebut kriteria kecukupan tes mis. Cakupan jalan Kriteria yang disebut kriteria uji kecukupan mis. Cakupan jalan Pengujian perangkat lunak otomatis dapat secara signifikan mengurangi biaya pengembangan perangkat lunak. Pengujian didefinisikan sebagai proses pelaksanaan program dengan tujuan untuk menemukan kesalahan. Kasus uji adalah sepasang input dan output yang diharapkan.

Kasus uji yang baik adalah kasus yang memiliki probabilitas tinggi untuk mendeteksi kesalahan yang belum ditemukan (cakupan jalur). Periset telah menunjukkan kesesuaian menggunakan perhitungan evolusioner dalam pengujian perangkat lunak. Salah satu perhitungannya adalah algoritma genetika (GA). Generator data uji berbasis GA hanya dapat menghasilkan satu masukan uji kasus dalam satu waktu. Ini berarti bahwa setiap generator data uji yang ada harus digunakan lebih dari satu kali. Jadi, beberapa data uji yang diperlukan dapat segera tersedia saat mencoba menemukan data uji lainnya. Semua ini membuat data test generator yang ada tidak efisien. Dalam makalah ini, kami prihatin dengan masalah mengambil program yang sewenang-wenang dan secara otomatis menghasilkan data uji untuk mencapai tingkat cakupan program tertentu. Makalah ini menggunakan gagasan Genetic Algorithm (GA) untuk secara otomatis mengembangkan satu set data uji untuk mencapai tingkat cakupan. Tujuan makalah ini adalah untuk menyajikan generator data uji berbasis GA yang mampu menghasilkan banyak data uji untuk mencakup beberapa jalur target. Sisa kertas disusun sebagai berikut. Bagian 2 memberikan latar belakang pengujian perangkat lunak.   
Bagian 3 membahas generator data uji berbasis GA dengan menggunakan standar (GA). Bagian 4 membahas kumpulan data uji coba jalur berbasis GA kami. Bagian 5 membahas percobaan untuk memvalidasi pendekatan. Bagian 6 membahas perbandingan antara pekerjaan dan karya lainnya. Akhirnya, bagian 7 untuk referensi.   
II. PENGUJIAN PERANGKAT LUNAK   
Pengujian perangkat lunak adalah salah satu komponen penting sistem perangkat lunak dengan banyak kendala yang kompleks dan saling terkait. Kualitas sistem perangkat lunak terutama ditentukan oleh kualitas proses perangkat lunak yang dihasilkannya. Telah diperkirakan bahwa kesalahan yang tertangkap selama fase spesifikasi sistem mungkin sekitar 50 kali lebih murah untuk diperbaiki daripada kesalahan yang tidak terdeteksi sampai pada tahap pengujian sistem. Definisi pengujian menurut standar IEEE / ANSI adalah: "Proses mengoperasikan sistem atau komponen dalam kondisi tertentu yang mengamati atau merekam hasilnya, dan melakukan evaluasi terhadap beberapa aspek sistem atau komponen". Definisi pengujian perangkat lunak sesuai standar IEEE / ANSI adalah: "Proses menganalisis item perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dan yang dipersyaratkan dan mengevaluasi fitur dari item perangkat lunak".

Teknik pengujian perangkat lunak dikelompokkan menjadi dua kategori: analisis statis dan pengujian dinamis. Dalam analisis statis, pengkodean kode membaca kode sumber perangkat lunak pemrogram yang sedang diuji, pernyataan berdasarkan pernyataan, dan secara visual mengikuti aliran logika program dengan memberi makan masukan. Jenis pengujian sangat tergantung pada pengalaman pengulas. Teknik pengujian dinamis mengeksekusi program yang diuji pada input data uji dan amati output. Pengujian dinamis dapat dikelompokkan menjadi dua kategori: pengujian kotak hitam dan pengujian whitebox. Pengujian kotak putih berkaitan dengan sejauh mana uji coba kasus atau mencakup aliran logika program. Jenis pengujian ini juga dikenal sebagai pengujian cakupan logika atau pengujian struktural karena mempertimbangkan struktur program. Pengujian kotak hitam menguji fungsionalitas perangkat lunak terlepas dari strukturnya. Ini hanya tertarik untuk memverifikasi output sebagai respons terhadap input data yang diberikan. Kecukupan pengujian cakupan logika dapat dinilai dengan menggunakan kriteria yang berbeda: pernyataan, keputusan, kondisi, keputusan / kondisi, cakupan kondisi ganda dan cakupan jalur. Kriteria yang lebih kuat dikenal sebagai keputusan atau cakupan cabang. Hal ini mensyaratkan bahwa setiap keputusan memiliki hasil Benar dan Salah setidaknya satu kali. Kekuatan tipe ini adalah bahwa hal itu akan memakan semua kemungkinan hasil setidaknya satu kali. Kelemahan ini   
Tipe kondisi tertentu masker kondisi lainnya. Kriteria kelayakan uji cakupan jalan berkaitan dengan   
Pelaksanaan jalur (terpilih) dalam program.   
III. GENERATOR DATA UJI BERBASIS GA   
Berbagai metode untuk mengembangkan generator data uji telah diproses selama ini. Setiap metode dimaksudkan untuk memenuhi kriteria kecukupan uji tertentu, dan sesuai dengan tujuan pengujian yang diinginkan. Pada usia dini otomatisasi pengujian perangkat lunak, sebagian besar generator data uji menggunakan algoritma gradien keturunan. Algoritma ini tidak efisien dan timeconsuming, dan tidak dapat melepaskan diri dari optima lokal di ruang pencarian data masukan yang mungkin. Algoritma pencarian Meta-heuristik telah digunakan dalam generator data uji sebagai alternatif yang lebih baik. Sejumlah kecil teknik data uji telah menjadi generator data uji acak acak, struktural atau berorientasi trafo, berorientasi objek, dan berorientasi analisis.   
Keterbatasan pendekatan ini telah menghentikan penerimaan umum mereka. Generator acak membuat sejumlah besar data uji karena tidak ada informasi mengenai tujuan pengujian; Generator seringkali gagal menemukan data yang memenuhi tujuan pengujian yang dinyatakan. Generator struktural atau berorientasi jalur pertama-tama mengidentifikasi jalur dimana data uji dihasilkan. Jalan ini sering kali tidak memungkinkan menyebabkan generator gagal menemukan masukan yang akan melintasi jalan setapak.

Generator berorientasi analisis memiliki kemampuan untuk menghasilkan data uji kualitas tinggi, namun mengandalkan perancang mereka yang memiliki wawasan besar mengenai domain operasi, dan karenanya tidak mudah umum dapat menggunakan sistem perangkat lunak yang sewenang-wenang. Generator yang berorientasi pada tujuan memberikan solusi kekuatan industri.   
Pekerjaan awal dengan pendekatan berorientasi pada tujuan telah mencapai beberapa keberhasilan terbatas yang menghasilkan data uji kualitas tinggi untuk program kecil. Proses pembuatan data uji otomatis menggunakan GA memiliki dua langkah utama:   
1 - Instrumentasi: ini adalah proses memasukkan probe (tag) pada awal / akhir dari setiap blok kode misalnya pada awal setiap fungsi, sebelum dan sesudah hasil benar dan salah dari setiap kondisi. Tag ini digunakan untuk   
Monitor jalur yang dilalui dalam program saat sedang dijalankan dengan data masukan uji tertentu.   
2 - uji evolusi data: ini adalah lingkaran dimana program dijalankan dengan beberapa data masukan awal, umpan balik dikumpulkan dan data masukan disesuaikan sampai kriteria memuaskan tercapai.  
Umpan balik adalah semacam nilai kebugaran yang diberikan pada data masukan saat ini untuk mencerminkan kesesuaiannya sesuai dengan   
Kriteria tes yang diberikan Langkah dasar GA ditunjukkan pada Gambar 1. Sebelum GA dapat digunakan, ada empat hal yang bergantung pada domain: mendefinisikan representasi genetik dari solusi masalah, menentukan fungsi kebugaran, memilih metode seleksi kromosom, dan menentukan operator genetik [menyeberang , Mutasi]. Pei dkk. Pada tahun 1994 [4] mengamati bahwa sebagian besar generator data uji menggunakan evaluasi simbolis. Mereka mengembangkan generator data percobaan single-path-coverage yang menggunakan GA. Roper dkk. Pada tahun 1995 [8] mengembangkan generator data uji berbasis GA yang bertujuan untuk melintasi semua cabang dalam program target. Jones dkk. Pada tahun 1995 [3] mengembangkan generator data uji berbasis GA untuk mencapai cakupan cabang dengan menggunakan urutan string biner untuk representasi individual. Dalam makalah ini, pendekatan memenuhi cakupan jalur. Pekerjaan mencoba menghasilkan beberapa data uji untuk mencakup beberapa jalur target sekaligus.   
IV. GENERATOR UJI PATH GAJAH BERBASIS GA   
Banyak generator data uji berbasis GA mengadopsi cakupan pernyataan atau cabang. Namun, kriteria cakupan jalan mencakup kriteria cakupan pernyataan dan cakupan cabang. Pengujian struktur perangkat lunak yang lebih efektif harus memiliki cakupan jalur sebagai tujuannya. Pekerjaan ini berfokus meliputi serangkaian jalur target dalam satu rangkaian GA yang mencakup beberapa jalur dalam satu proses yang memerlukan penggabungan jalur ini dalam perhitungan kebugaran. Dengan demikian, mencoba untuk memenuhi banyak jalur sekaligus diharapkan dapat meningkatkan efektifitas dan efisiensi dari generator data uji. Pada bagian 4.1, kita membahas algoritma genetika. Pada bagian 4.2, kita membahas struktur masalah. Pada bagian 4.3, kita membahas representasi yang dipilih ke kromosom. Pada bagian 4.4, kita membahas desain fungsi fitness. Pada bagian 4.5, kita membahas kriteria seleksi. Pada bagian 4.6, kita membahas desain cross over operator. Pada bagian 4.7, kita membahas desain operator mutasi.  
A. Algoritma Genetika Desain   
Algoritma genetika diterapkan pada masalah yang melibatkan pencarian dan pengoptimalan dalam domain AI. Prinsip GA adalah bahwa mereka menciptakan dan memelihara populasi individu yang diwakili oleh kromosom. Kromosom ini biasanya dikodekan solusi untuk masalah.  
Kromosom ini diproses sesuai aturan seleksi, cross over dan mutasi. Setiap kromosom dalam populasi memiliki ukuran kebugarannya pada populasi. Nilai fitness ini menunjukkan seberapa sukses kromosom tersebut sebagai solusi terhadap masalah. Proses seleksi memilih kromosom dengan nilai fitness tinggi. Populasi baru berasal dari kromosom terpilih ini setelah diproses dengan proses crossover dan mutasi. Proses cross over analog dengan proses reproduksi seksual yang melibatkan dua kromosom yang menukar potongan data (informasi genetik). Proses mutasi memperkenalkan sedikit perubahan ke dalam proporsi kecil dalam populasi. Kode pseudo untuk GA sederhana ada di Fig2. Algoritma akan **beriterasi sampai populasi berkembang dari solusi ke masalah, atau sampai jumlah iterasi maksimum telah terjadi.**  
B. Struktur Masalah Masalah digambarkan sebagai pohon. Setiap simpul di pohon mewakili kondisi yang memiliki dua nilai (True, False). Dari Fig3. Untuk menghitung jumlah jalur dalam program ini, kita menggunakan teknik edge-matrix [m, m] dimana m adalah jumlah node dalam program sumber (kondisi) .Karena jalur dalam urutan, kita ganti 1 sampai 2. Setelah itu kita mengalikan diagonal. Dalam contoh ini, jumlah path adalah 8, Fig5. Dari Fig4. Pada matriks tepi, yang pertama di baris kedua menunjukkan bahwa kondisi kedua bergantung pada kondisi pertama. Dalam kasus node dependen, kita mengganti 1 dengan 3 di matriks tepi, Fig5.  
C. Representasi Kromosom Representasi kromosom (individu) dapat mempengaruhi kinerja generator data uji berbasis GA. Representasi yang paling umum adalah string biner yang mewakili simpul yang dikunjungi oleh kromosom ini. Kromosom diwakili sebagai masukan tes dan gennya mewakili simpul dan jika simpul tersebut dikunjungi atau tidak (satu kromosom sama dengan satu jalur). Jumlah gen sama dengan jumlah node. Dalam karya ini, representasi kromosom dipilih sebagai rangkaian data uji dimana masing-masing merepresentasikan satu jalur dan jumlah data uji sama dengan jumlah jalur yang dihasilkan. Cara representasi ini meningkatkan jangkauan pengujian yang akibatnya akan meningkatkan reliabilitas dan kepercayaan pada data generator uji. Salah satu sifat representasi baru ini adalah topeng kromosom. Topeng ini mewakili jalan yang diliputi oleh kromosom ini. Misalnya, jika kita memiliki dua variabel input x dan y dan satu kondisi pada sebuah node adalah (x <y). Tabel1 menunjukkan kromosom dengan kondisi sebelumnya. Nilai 1 menunjukkan bahwa jalur pertama ditutup. Dua nilai 0 menunjukkan bahwa jalur kedua dan ketiga tidak tercakup dalam kromosom ini. Target kami dalam pekerjaan ini adalah membuat topeng kromosom yang menunjukkan bahwa semua jalur tertutup. Dalam hal ini, kromosom ini merupakan solusi yang baik (wajib) terhadap masalah.

D. Desain Fungsi Kebugaran   
Kebugaran mengungkapkan betapa baiknya solusi kromosom dalam kaitannya dengan optimal global. Semacam ini   
Nilai akan dihitung untuk setiap kromosom di setiap populasi dengan fungsi kebugaran. Nilai fitness dulu   
Bandingkan individu dan untuk membedakan penampilan mereka. Seseorang yang mendekati solusi optimal mendapatkan nilai fitness lebih tinggi daripada individu yang jauh. Masalah yang sulit dalam menggunakan GA sering kali dilakukan Untuk menemukan fungsi kebugaran yang sesuai yang menghitung nilai fitness dan mengungkapkan masalahnya sebaik mungkin. Dalam pekerjaan ini, kita memilih fungsi fitness sebagai berikut

Nilai fitness (kromosom) = jumlah simpul yang dikunjungi + PU (kromosom) (1)   
PU (kromosom) = Σ (Probability (Gene) + Pemanfaatan (Gene)) (2)   
Probabilitas (Gene) = jumlah node yang dikunjungi / jumlah semua node (3)   
Pemanfaatan (Gen) = jumlah kejadian jalur ini / jumlah semua tes (4)   
Fungsi fitness terdiri dari dua bagian: bagian pertama adalah jumlah node yang dikunjungi oleh kromosom ini (jumlah Simpul yang dikunjungi oleh semua jalur dalam kromosom ini tanpa redundansi), bagian kedua adalah jumlah probabilitas gen dan pemanfaatan gen untuk semua jalur. Probabilitas gen dihitung dari pemberian setiap gen sebuah nilai yang mewakili Kepentingan yang sesuai dengan yang lain. Pemanfaatan gen dihitung dari rasio penggunaan gen ini (path) pada siklus uji.   
E. Desain Operator Seleksi   
Operator seleksi memilih dua individu dari satu generasi untuk menjadi orang tua untuk proses rekombinasi (cross over dan mutasi). Ada berbagai metode untuk memilih individu, mis. Secara acak atau berkenaan dengan nilai fitness mereka. Pemilihan acak memilih kedua orang tua secara acak dari populasi untuk menghasilkan populasi baru tanpa mengetahui informasi tentang kebugaran orang tua ini dan jika itu adalah solusi yang baik atau tidak. Metode ini memberi kesempatan yang sama pada semua kromosom. Metode ini akan menjamin keragaman dan kawin yang sehat. Jika individu dipilih sehubungan dengan nilai fitness mereka, ini menjamin bahwa kromosom dengan nilai fitness yang lebih tinggi memiliki kemungkinan lebih tinggi untuk dipilih; Yang lain cenderung dibuang. Dengan menggunakan metode ini, kromosom dengan nilai fitness tertinggi harus dipilih. Metode seleksi yang digunakan dalam pekerjaan ini adalah linear rank. Untuk pemilihan peringkat linier, individu diurutkan berdasarkan nilai fitness mereka dan pangkat N ditugaskan ke individu terbaik dan peringkat 1 ditugaskan ke individu terburuk. Probabilitas seleksi secara linear diberikan kepada individu menurut pangkat mereka:  
Pi = (1 / N) (η- + (η + -η-) ((i-1) / (N-1))); I € {1, 2, ..., N} (5)  
Di sini η- / N adalah probabilitas individu terburuk yang dipilih dan η | + / N adalah probabilitas individu terbaik untuk dipilih. Karena ukuran populasi tetap konstan, kondisi η | + = 2- η | - dan η | -> = 0 harus dipenuhi. Perhatikan bahwa semua individu mendapatkan peringkat yang berbeda (probabilitas seleksi berbeda), bahkan jika mereka memiliki nilai fitness yang sama. Probabilitas ditentukan gradien fungsi linier.  
Η- = 2 / rm + 1 dan η + = 2rm / rm + 1. Algoritma pemilihan peringkat linier ditunjukkan pada Gambar 6. Gradien dari fungsi linier rm memiliki nilai 1,7 karena harus memenuhi 1 <= rm <= 2 dan ini adalah nilai bagus yang digunakan dalam penentuan linier sampai sekarang.

F. Perancangan Cross Over Operator   
Cross over beroperasi pada tingkat individu. Kedua orang tua (kromosom) menukar informasi substring pada posisi acak di kromosom untuk menghasilkan dua senar baru yang mewakili keturunan. Operator crossover mencari lebih baik dalam materi genetik. Tujuannya adalah untuk menciptakan individu yang lebih baik dan sebagai hasilnya menciptakan populasi yang lebih baik. Penyeberangan operator yang digunakan dalam pekerjaan ini bergantung pada gagasan baru.   
Setelah memilih dua orang tua, yang terbaik dari kedua individu tersebut akan bertahan. Yang terbaik ditentukan oleh jumlah jalur yang diliputi oleh kromosom ini. Yang lainnya menjadi hasil proses OR pada topeng kedua orang tua dan kemudian menyusun ulang jalur di kromosom ini. Sebagai contoh, ini adalah topeng dua kromosom (yang dipilih sebagai orang tua) seperti yang ditunjukkan pada Tabel 2. Seperti yang ditunjukkan pada contoh sebelumnya, kromosom kedua bertahan karena mencakup tiga jalur tapi yang pertama hanya mencakup dua jalur saja.   
Kromosom pertama yang baru adalah hasil proses (11000 OR 01101) adalah (11101) yang mencakup empat jalur.   
G. Desain Operator Mutasi   
Mutasi adalah perubahan acak sesekali dari sedikit nilai yang mengubah beberapa fitur dengan konsekuensi yang tidak dapat diprediksi. Mutasi digunakan untuk mempertahankan keragaman populasi dan untuk menjaga agar populasi tidak berkumpul dalam satu solusi. Operator mutasi yang digunakan dalam pekerjaan ini bergantung pada gagasan baru.   
Setelah menyeberang, kromosom baru bermutasi dengan memilih jalur yang tidak tercakup dalam dua kromosom dan mencoba empat atau lima kali untuk dicakup. Hal ini dilakukan dengan memberi masukan data baru dan menguji kembali jalurnya. Jika kita menemukan jalan yang tertutup, ganti dengan yang sudah tua. Jika jalur coved tidak ditemukan, tetap tidak ditutupi. Sebagai contoh, pada contoh sebelumnya, kedua kromosom setelah crossover menjadi seperti ditunjukkan pada Tabel3. Jalur pertama di kromosom kedua dan jalur keempat di dua kromosom tidak tertutup. Jika kita menemukan jalan keempat yang tertutup, kedua kromosom tersebut menjadi seperti yang ditunjukkan pada Tabel4. Ini berarti kita menemukan solusi untuk masalah yang diwakili oleh kromosom pertama. Tes sto ps dalam kasus ini dan mencetak solusi. Tetapi jika kita tidak menemukan jalan keempat tertutup, kita akan mencoba lagi. Jika kita menemukan jalan yang tertutup pertama, kedua kromosom akan menjadi seperti   
Ditunjukkan pada Tabel5.   
V. EKSPERIMEN   
Pada bagian ini, kami menyajikan dan menilai kinerja pekerjaan kami. Kami menyajikan hasil dan analisis eksperimental.   
A. Desain Percobaan Kami memiliki empat percobaan yang berbeda. Setiap percobaan terdiri dari serangkaian tiga putaran. Kami menilai kinerja seleksi peringkat linier, cross over operator baru dan operator mutasi baru. Kami menggunakan tiga program yang sesuai untuk masalah ini seperti jumlah maksimum, pengukuran suhu dan program uji kecil yang digunakan untuk menunjukkan semua langkah dari generator data uji ganda.   
B. Pengaturan Parameter GA   
Nilai parameter GA ditunjukkan pada Tabel 6. Parameter ini bervariasi dari satu percobaan ke percobaan lainnya.   
C. Extended CFG (Context Free Grammar)   
Bahasa yang digunakan dalam program sumber memiliki aturan pada Fig7.   
D. Standar GA   
Pada awalnya, kami menguji aturan GA standar pada ketiga program tersebut. Standar GA menggunakan pilihan acak, satu titik   
Mutasi silang dan acak. Hasilnya ditunjukkan pada Tabel7.   
E. Metode Seleksi Baru   
Hasil pemilihan linier bergilir dan bukan seleksi acak ditunjukkan pada Tabel 8.   
F. Metode Cross Baru   
Hasil penggunaan metode cross over baru kami dan bukan cross cross satu poin ditunjukkan pada Tabel 9.   
G. Metode Mutasi Baru   
Hasil penggunaan metode mutasi baru dan mutasi acak ditunjukkan pada Tabel 10.   
H. Eksperimen Akhir   
Akhirnya, kita meningkatkan kesempatan menggunakan peluang mutasi sampai 10 untuk mendapatkan jalan baru yang tertutup. Ini berarti kita melipatgandakan peluang metode ini. Hasil penggunaan mutasi baru ditunjukkan pada Tabel11. Pada program pertama, kami menemukan solusi untuk masalah karena semua jalur tertutup. Pada program kedua, jumlah individu yang mencapai rasio pertambahan ini meningkat. Pada program ketiga, jumlah jalan yang tercakup adalah, seperti yang ditunjukkan pada semua tabel sebelumnya, hasil terbaik sama sekali.   
I. Angka dan Tabel

TABEL 1. Contoh kromosom

|  |  |  |  |
| --- | --- | --- | --- |
| ***Kromosom*** | ***X = 10***  ***Y = 11*** | ***X = 15***  ***Y = 9*** | ***X = 13***  ***Y = 1*** |
| ***Topeng*** | 1 | 0 | 0 |

Tabel 2. Kedua orang tua

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Kromosom1*** | ***1*** | ***1*** | ***0*** | ***0*** | ***0*** |
| ***Kromosom2*** | 0 | 1 | 1 | 0 | 1 |

Tabel 3. Dua Kromosom Setelah Cross Over

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Kromosom1*** | ***1*** | ***1*** | ***1*** | ***0*** | ***1*** |
| ***Kromosom2*** | 0 | 1 | 1 | 0 | 1 |

Tabel 4. Dua Kromosom jika jalur keempat diliputi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Kromosom1*** | ***1*** | ***1*** | ***1*** | ***1*** | ***1*** |
| ***Kromosom2*** | 0 | 1 | 1 | 1 | 1 |

Tabel 5. Dua Kromosom jika jalan pertama diliputi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Kromosom1*** | ***1*** | ***1*** | ***1*** | ***0*** | ***1*** |
| ***Kromosom2*** | 1 | 1 | 1 | 0 | 1 |

Tabel 6. Setup Parameter GA

|  |  |  |
| --- | --- | --- |
| ***Jumlah*** | ***parameter*** | ***Nilai*** |
| ***123456*** | Ukuran populasi  Jumlah Generasi  Metode Seleksi  Ukuran kromosom  Gradien Linear Function  Jumlah Tes | 6  10  Peringkat linier  Jumlah Jalan  1.7  5 |

Tabel 7. Standar GA

|  |  |  |
| --- | --- | --- |
| ***Program*** | ***Jumlah***  ***Jalan yang tertutup*** | ***Jumlah semua***  ***Jalan*** |
| ***Program Jumlah Maksimum***  ***Program Ukur Suhu***  ***Program Uji Sendiri*** | 213 | 44  16 |

Tabel 8. Seleksi Linear Ranking

|  |  |  |
| --- | --- | --- |
| ***Program*** | ***Jumlah***  ***Jalan yang tertutup*** | ***Jumlah semua***  ***Jalan*** |
| ***Program Jumlah Maksimum***  ***Program Ukur Suhu***  ***Program Uji Sendiri*** | 323 | 44  16 |

Tabel 9. Metode Cross Cross Baru

|  |  |  |
| --- | --- | --- |
| ***Program*** | ***Jumlah***  ***Jalan yang tertutup*** | ***Jumlah semua***  ***Jalan*** |
| ***Program Jumlah Maksimum***  ***Program Ukur Suhu***  ***Program Uji Sendiri*** | 325 | 44  16 |

Tabel 10. Metode mutasi baru

|  |  |  |
| --- | --- | --- |
| ***Program*** | ***Jumlah***  ***Jalan yang tertutup*** | ***Jumlah semua***  ***Jalan*** |
| ***Program Jumlah Maksimum***  ***Program Ukur Suhu***  ***Program Uji Sendiri*** | 339 | 44  16 |

Tabel 11. Eksperimen Akhir

|  |  |  |
| --- | --- | --- |
| ***Program*** | ***Jumlah***  ***Jalan yang tertutup*** | ***Jumlah semua***  ***Jalan*** |
| ***Program Jumlah Maksimum***  ***Program Ukur Suhu***  ***Program Uji Sendiri*** | 43  11 | 44  16 |

Tabel 12. Kompilasi antara pekerjaan kita dan karya Moataz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Program*** | ***Jumlah***  ***tertutupi***  ***Jalan***  ***(pekerjaan kita)*** | ***Jumlah***  ***Tak terlihat***  ***Jalan***  ***(pekerjaan kita)*** | ***Jumlah***  ***tertutupi***  ***Jalan***  ***(Moataz's***  ***kerja)*** | ***Jumlah***  ***Tak terlihat***  ***Jalan***  ***(Moataz's***  ***kerja)*** |
| ***Pencarian biner***  ***Semacam gelembung*** | 7  25 | 1  20 | 7  14 | 0  11 |

VI. PERBANDINGAN DENGAN KARYA LAIN   
Akhirnya, setelah menguji algoritma kami pada beberapa program, kami akan membandingkan hasilnya dengan hasil karya Moataz [1]. Hasilnya ditunjukkan pada Tabel 12.