Integrasi Pareto Fitness, Multiple-Population dan Temporary Population pada Algoritma Genetika untuk Pembangkitan Data Test pada Pengujian Perangkat Lunak

Mohammad Reza Maulana, Romi Satria Wahono dan Catur Supriyanto
Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
reza.stmikwp@gmail.com, romi@romisatriawahono.net, caturs@research.dinus.ac.id

Abstrak: Pengujian perangkat lunak memerlukan biaya yang mahal dan sering kali lebih dari 50% biaya keseluruhan dalam pengembangan perangkat lunak digunakan dalam tahapan ini. Untuk mengurangi biaya proses pengujian perangkat lunak secara otomatis dapat digunakan. Hal yang sangat penting dalam pengujian perangkat lunak secara otomatis adalah proses menghasilkan data tes. Pengujian secara otomatis yang paling efektif dalam menekan biaya adalah pengujian *branch coverage*. Salah satu metode yang banyak digunakan dan memiliki kinerja baik adalah algoritma genetika (AG). Salah satu permasalahan AG dalam menghasilkan data tes adalah ketiga target cabang dipilih memungkinkan tidak ada satupun individu yang memenuhi kriteria. Hal ini akan menyebabkan proses pencarian data tes memakan waktu lebih lama. Oleh karena itu di dalam penelitian ini diusulkan integrasi *pareto fitness*, *multiple-population* dan *temporary population* di dalam proses pencarian data tes dengan menggunakan AG (AG-PFMPTP). *Multiple-population* diusulkan untuk menghindari *premature convergence*. Kemudian *pareto fitness* dan *temporary population* digunakan untuk mencari beberapa data tes sekaligus, kemudian mengevaluasinya dan memasukkan ke dalam *archive temporary population*. Dari hasil pengujian yang telah dilakukan rata-rata generasi metode AG-PFMPTP secara signifikan lebih sedikit dalam menghasilkan data tes yang dibutuhkan dibandingkan metode AG standar ataupun AG dengan multiple-population (AG-MP) pada semua benchmark program yang digunakan. Hal tersebut menunjukkan metode yang diusulkan lebih cepat dalam mencari data tes yang dibutuhkan.

Kata Kunci: pengujian perangkat lunak, pembangkitan data tes, algoritma genetika

1 PENDAHULUAN

Pengujian perangkat lunak merupakan bagian yang tidak terpisahkan dari rekayasa perangkat lunak (Anand et al., 2015). Pengujian perangkat lunak merupakan tahap yang sangat penting dalam siklus pengembangan perangkat lunak yang bertujuan untuk memastikan tingkat kualitas perangkat lunak yang dihasilkan pada tingkatan tertentu (Ferrer, Kruse, Chicano, & Alba, 2015). Proses pengujian perangkat lunak memerlukan biaya yang mahal (Anand et al., 2015)(P McMinn, 2004)(Alakeel, 2014), sering kali lebih dari 50% biaya total proses pengembangan perangkat lunak digunakan dalam proses ini (Anand et al., 2015). Untuk mengatasi hal tersebut pengujian perangkat lunak secara otomatis dapat dilakukan untuk mengurangi biaya (Anand et al., 2015)(Ferrer et al., 2015)(Díaz, Tuya, & Blanco, 2003)(Hermadi, Lokan, & Sarker, 2014), dan mendapatkan hasil pengujian yang lebih meyakinkan (Díaz et al., 2003)(Hermadi et al., 2014).

Dalam pengujian perangkat lunak secara otomatis, pembangkitan data tes merupakan hal yang paling utama (P McMinn, 2004)(Lakhotia, McMinn, & Harman, 2009)(Mao, 2014)(Mao, Xiao, Yu, & Chen, 2015)(Ribeiro, 2008). Pembangkitan data tes adalah proses identifikasi serangkaian uji kasus untuk memenuhi kecukupan data tes yang dipilih (Pachauri & Srivastava, 2013). Pembangkitan data tes menguraikan proses pencarian data tes terbaik untuk kriteria tes tertentu.

Terdapat beberapa pendekatan yang digunakan dalam membangkitkan data tes pada pengujian perangkat lunak secara otomatis, diantaranya pengujian white-box (pengujian struktural) (Mao, 2014)(Mao et al., 2015)(Phil McMinn, Harman, Lakhotia, Hassoun, & Wegener, 2012) dan pengujian black-box (pengujian fungsional) (Ferrer et al., 2015)(Vos et al., 2013). Dibandingkan pengujian fungsional, pengujian struktural memegang peranan penting karena lebih efektif dalam mengurangi biaya dalam proses pengujian perangkat lunak (Mao et al., 2015).

Saat ini banyak metode yang digunakan untuk membangkitkan data tes. Metode pencarian berbasis Metaheuristik atau yang dikenal dengan Search-Based Software Testing (SBST) banyak digunakan dalam penelitian berdasarkan *survey* yang telah dilakukan oleh McMinn (Phil McMinn, 2011). Metode ini dapat membangkitkan data tes dengan hasil cakupan yang tinggi dan mempunyai kemampuan dalam memperlihatkan kesalahan program yang diuji (Mao et al., 2015). Selain itu berdasarkan *survey* yang dilakukan Ali et al. (Ali, Briand, Hemmati, & Panesar-Walawege, 2010) menyatakan bahwa SBST lebih unggul dalam aspek efektifitas biaya dibandingkan dengan pembangkitan data tes secara tradisional. Meskipun memiliki banyak keunggulan, SBST mempunyai tantangan dalam penentuan *fitness function* yang akan berpengaruh pada efek cakupan hasil dan pencarian yang lebih efisien (Mao et al., 2015). *fitness function* digunakan dalam menyesuaikan arah pencarian data tes dan membangun hasil cakupan.

Beberapa algoritma SBST yang digunakan dalam membangkitkan data tes diantaranya simulated annealing (SA) (Cohen, Colbourn, & Ling, 2003)(Patil & Nikumbh, 2012), ant colony optimization (ACO) (Mao et al., 2015), dan algoritma genetika (AG) (Pachauri & Srivastava, 2013)(Alshraideh, Mahafzah, & Al-Sharaeh, 2011)(Praveen Ranjan Srivastava & Kim, 2009)(Miller, Reformat, & Zhang, 2006)(Aleti & Grunske, 2015)(Yao & Gong, 2014). Penggunaan algoritma SA mempunyai kelebihan dalam memecahkan optimasi pada masalah program yang kompleks (Mao et al., 2015). Walaupun SA memiliki kelebihan, algoritma tersebut memiliki kelemahan dalam hal kecepatan konvergensi (Mao, 2014). Srivastava et al. (P R Srivastava & Baby, 2010) mengadopsi algoritma ACO untuk menghasilkan urutan tes berdasarkan pengujian perangkat lunak, hal ini digunakan untuk memberikan hasil cakupan pengujian yang tinggi. Akan tetapi ACO memiliki kelemahan seperti pada algoritma SA, ACO memerlukan waktu yang lama dalam menghasilkan konvergensi. Algoritma genetika mempunyai kelebihan dalam menangani program dengan skala besar dan efektif dalam melakukan pencarian dibandingkan dengan metode pencarian

yang lain (Phil McMinn, 2011)(Harman & McMinn, 2010)(Yao & Gong, 2014).

Di dalam banyak penelitian yang dirangkum oleh Bueno (Bueno, Jino, & Wong, 2014), algoritma genetika digunakan untuk menghasilkan data tes untuk pengujian *statement coverage*, *path coverage*, *fault based coverage*, dan *branch coverage*. Diantara jenis pengujian tersebut, *branch coverage* merupakan pengujian yang paling efektif dalam menekan biaya dalam proses pengujian (Mao, 2014). *Branch coverage* merupakan pengujian yang dilakukan untuk menguji kondisi percabangan di dalam *source code* program. Salah satu permasalahan yang ada dalam menghasilkan data test untuk menguji setiap kondisi percabangan di dalam program dengan metode algoritma genetika adalah ketika cabang dipilih untuk target cakupan pengujian memungkinkan tidak ada satupun individu (individu mengkodekan data tes masukan) di dalam populasi yang memenuhi kriteria (Pachauri & Srivastava, 2013)(Alshraideh et al., 2011)(Miller et al., 2006). Kriteria dalam hal ini adalah hasil data tes yang didapatkan tidak dapat digunakan untuk mengeksekusi target cabang yang ditentukan. Apabila cabang tersebut tidak terlewati hal ini menyebabkan *source code* yang ada di dalam cabang tersebut tidak dapat dieksekusi untuk diuji.

Pada penelitian yang dilakukan (Alshraideh et al., 2011) mengusulkan model *multi-population* dalam melakukan pencarian beberapa target cakupan pengujian secara bersamaan. Dalam setiap tahap pencarian, tidak hanya satu kandidat target yang dicari, melainkan beberapa target sekaligus. Hal ini dilakukan untuk menghindari optimal lokal jika hanya satu target yang dicari. *Multi-population* juga akan mencegah terjadinya *premature convergence* yang akan mengakibatkan hasil pencarian tidak bisa beragam (Cantu-paz, 1997). Metode yang diimplementasikan menunjukan pencarian terhadap target menjadi lebih singkat, jumlah eksekusi yang diperlukan untuk cakupan target lebih banyak, dan lebih efektif dalam proses pencarian jika dibandingkan dengan yang hanya menggunakan *single-population* (Alshraideh et al., 2011). Meskipun demikian *multiple-population* belum banyak berkontribusi terhadap tercapainya pemenuhan target yang dapat tereksekusi.

Pada penelitian lain yang dilakukan (Pachauri & Srivastava, 2013) mengusulkan model perbaikan proses pencarian dengan cara mengurutkan target yang akan dieksekusi. Selain mengurutkan target, peneliti mengusulkan teknik penyimpanan beberapa individu terbaik di dalam *memory* yang pada iterasi berikutnya akan menggantikan individu yang paling buruk. Penggunaan elitism juga diimplementasikan oleh peneliti untuk membandingkan dan menggantikan sebagian atau keseluruhan individu di dalam populasi yang dihasilkan untuk mendapatkan populasi terbaik. Hasil dari metode yang diusulkan menunjukkan peningkatan pencarian target yang lebih baik. Walaupun metode yang diusulkan memberikan hasil yang baik, masih terdapat kemungkinan untuk diperluas dan ditingkatkan performa dalam menghasilkan data tes.

Seperti yang sudah dijelaskan sebelumnya *fitness function* memegang peranan penting dalam proses pencarian target (Mao et al., 2015). *Fitness function* yang lebih baik memungkinkan proses pencarian target akan semakin lebih baik. Terdapat konsep penghitungan nilai fitness pada kasus *multiobjective* di dalam proses pencarian di dalam algoritma genetika yang dinamakan *pareto fitness*. Hasil penelitian menunjukan perbaikan *fitness function* tersebut dapat meningkatkan kinerja algoritma genetika (Elaoud, Loukil, & Teghem, 2007)(Ferrer, Chicano, & Alba, 2012).

Berdasarkan hal tersebut skema penggabungan *multi-population* dan *pareto fitness* akan diusulkan pada penelitian ini. Di dalam penelitian ini peneliti juga mengusulkan metode *temporary population* yang nantinya digunakan dalam penyimpanan beberapa populasi terbaik untuk beberapa target cabang sekaligus. Diharapkan dengan penggabungan metode tersebut dapat meningkatkan kinerja algoritma genetika dalam proses pencarian dan pemenuhan target pengujian akan menjadi lebih baik.

Paper ini disusun dengan urutan sebagai berikut. Pada bagian 2, penelitian terkait akan dijelaskan. Pada bagian 3 metode yang diusulkan akan dipaparkan. Selanjutnya bagian 4 akan disajikan perbandingan hasil eksperimen metode yang diusulkan dengan metode yang lain. Kemudian pada bagian terakhir akan disampaikan kesimpulan dari penelitian yang dilakukan.

2 PENELITIAN TERKAIT

Penelitian pembangkitan data tes menggunakan metode algoritma genetika sudah banyak dilakukan. Banyak peneliti menggunakan metode algoritma genetika karena kemampuan dari metode ini lebih baik dibandingkan dengan metode lain. Pachauri & Srivastava (Pachauri & Srivastava, 2013) mengusulkan model pencarian data tes menggunakan algoritma genetika dengan memanfaatkan metode pengurutan *branch* yang akan dicari, kemudian mengimplementasikan *memory* dan *elitism*. Metode perbaikan yang diusulkan adalah dalam proses pencarian data tes, yaitu dengan cara mengurutkan target yang akan dieksekusi. Selain mengurutkan target, peneliti mengusulkan teknik penyimpanan beberapa individu terbaik di dalam *memory* yang pada iterasi berikutnya akan menggantikan individu yang paling buruk. Penggunaan elitism juga diimplementasikan oleh peneliti untuk membandingkan dan menggantikan sebagian atau keseluruhan individu di dalam populasi yang dihasilkan untuk mendapatkan populasi terbaik. Hasil dari metode yang diusulkan menunjukkan peningkatan pencarian target yang lebih baik. Walaupun metode yang diusulkan memberikan hasil yang baik, masih terdapat kemungkinan untuk diperluas dan ditingkatkan performa dalam menghasilkan data tes.

Alshraideh et al. mengusulkan proses pembangkitan data tes menggunakan *multiple-population* pada algoritma genetika (Alshraideh et al., 2011). Dalam setiap tahap pencarian, tidak hanya satu kandidat target yang dicari, melainkan beberapa target sekaligus. Hal ini dilakukan untuk menghindari optimal lokal jika hanya satu target yang dicari. *Multi-population* juga akan mencegah terjadinya *premature convergence* yang akan mengakibatkan hasil pencarian tidak bisa beragam. Metode yang diimplementasikan menunjukan pencarian terhadap target menjadi lebih singkat, jumlah eksekusi yang diperlukan untuk cakupan target lebih banyak, dan lebih efektif dalam proses pencarian jika dibandingkan dengan yang hanya menggunakan *single-population*. Begitupun Yao et al. (Yao & Gong, 2014) mengusulkan pembangkitan data tes menggunakan algoritma genetika dengan memanfaatkan *individual sharing* di dalam *multiple-population*. Metode yang diusulkan Yao et al. berbeda dengan *multiple-population* tradisional. Tujuan utama dari strategi yang diusulkan adalah untuk memperluas jangkauan pencarian setiap populasi dengan berbagi individu, sehingga dapat meningkatkan efisiensi algoritma.

Ferrer et al. mengusulkan proses pembangkitan data tes menggunakan konsep *Multiobjective Evolutionary based on Pareto Efficiency* dengan menggunakan metode *Multi-*

Objective Cellular algoritma genetika (MOCell) (Ferrer et al., 2012). Dalam pendekatan metode yang diusulkan, solusi untuk masalah ini adalah tes *suite*, yaitu, satu set data tes. Tes *suite* dievaluasi sesuai dengan dua tujuan. Tujuan pertama mengevaluasi cakupan yang diperlukan, secara umum, mengeksekusi tes *suite* kedalam program yang diuji. Tujuan kedua adalah evaluasi hitungan sederhana dari jumlah data tes di dalam satu set tes *suite*. Jumlah test data yang sedikit dan mencakup semua *test suite* yang dibutuhkan akan menjadikan pengujian lebih efektif.

Di dalam penelitian ini, peneliti akan mengintegrasikan konsep *pareto fitness*, *multiple population* dan *temporary population* untuk meningkatkan kinerja algoritma genetika dalam melakukan pencaraian data tes yang hasilnya akan digunakan untuk melakukan pengujian perangkat lunak.

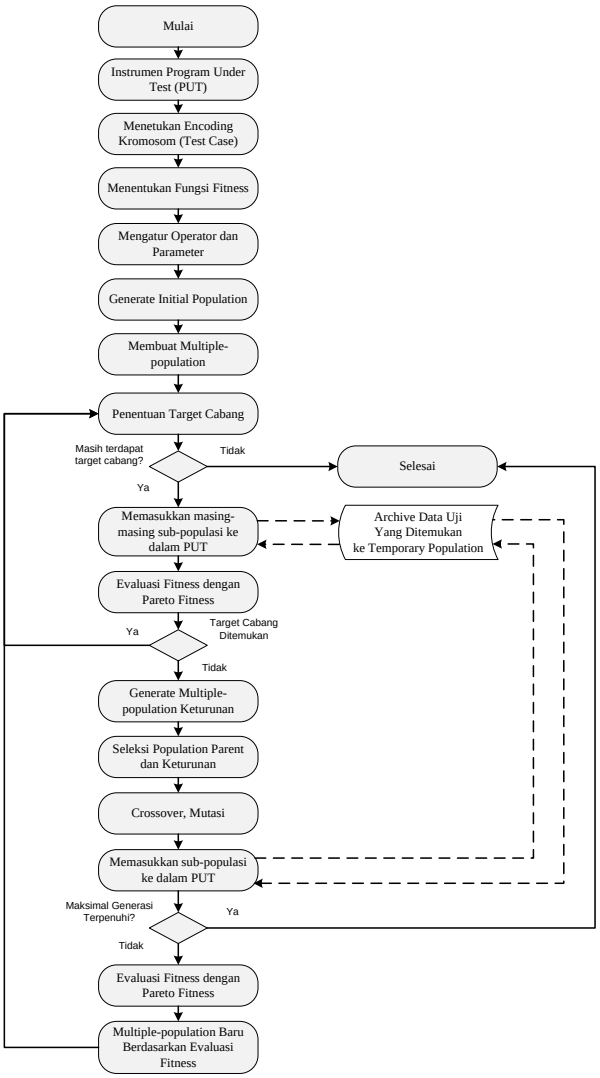
3 METODE YANG DIUSULKAN

Metode yang diusulkan yaitu integrasi *pareto fitness*, *multiple-population* dan *temporary population* di dalam algoritma genetika untuk pembangkitan data tes perangkat lunak. *Pareto fitness* digunakan untuk menghitung masing-masing individu yang memungkinkan dengan menggunakan fungsi fitness yang berbeda. Hal ini dapat mempercepat pencarian target data tes. *Temporary p*opulation digunakan untuk menyimpan individu yang sesuai dengan fungsi *fitness* yang memenuhi kriteria. *Temporary population* juga digunakan untuk membandingkan hasil individu sebelumnya dengan generasi yang sedang berjalan. Hal ini akan membantu pengurangan data tes yang sedang dicari, karena sebagian data tes sudah ditemukan. *Multiple-population* digunakan untuk menghindari *premature convergence* dan menjadikan individu lebih beragam.

Gambar 1 Menunjukkan alur dari metode yang diusulkan. langkah yang pertama adalah membuat instrumen program yang akan diuji (*Program Under Test* (PUT)). Instrumen tersebut diantaranya analisa kondisi yang ada di source code program tersebut. Instrumen ini nantinya akan berpengaruh pada penentuan nilai fitness. Setelah itu proses selanjutnya adalah menentukan proses *encoding* untuk mengkodekan kromosom yang akan dimasukkan di dalam populasi. Dalam penelitian ini digunakan *binary string encoding* untuk pengkodean kromosom. Proses penentuan fungsi *fitness* dilakukan setelahnya dengan melihat instrumen yang telah dianalisa pada tahap awal.

Parameter dan operator algoritma genetika diatur untuk inisialiasi kebutuhan saat proses algoritma genetika berlangsung. Contoh pengaturan parameter adalah mengatur panjang dari kromosom, ukuran populasi dan bermacam-macam nilai probabilitas. Sedangkan contoh mengatur operator seperti pemilihan proses seleksi, *crossover* dan mutasi.

Proses selanjutnya yaitu membangkitkan populasi awal secara acak. Untuk metode yang diusulkan, proses selanjutnya adalah pembuatan *multiple-population*. Kemudian pemilihan target cabang ditentukan, jika target cabang masih ada akan dilanjutkan proses pencarian, jika tidak maka pencarian selesai. Dari masing-masing sub-populasi yang dihasilkan akan dievaluasi menggunakan *pareto fitness*. Setelah dilakukan evaluasi *fitness* selanjutnya masing-masing kromosom yang terdapat di dalam sub-populasi dimasukkan ke dalam PUT. Di bagian ini akan dilakukan pengecekan apakah kromosom sesuai dengan data tes yang dibutuhkan. Di dalamnya terdapat kemungkinan tiga kemungkinan, pertama tidak ada satupun data tes yang diharapkan diciptakan. Kedua, hanya terdapat sebagian data tes yang memenuhi kriteria, kemudian dimasukkan ke dalam *archive* data tes untuk disimpan. Ketiga, semua data tes berhasil ditemukan, setelahnya dimasukkan ke dalam *archive* data tes untuk disimpan. Pada bagian ini *temporary population* mengecek berdasarkan evaluasi *fitness*. Walaupun bukan individu yang sesuai target cabang, akan tetapi apabila memenuhi kriteria yang lain maka akan dilakukan penyimpanan ke dalam *archive temporary population*. Proses selanjutnya mengecek apakah cabang yang sudah dipilih sebelumnya dieksekusi. Jika dieksekusi maka akan ke proses penentuan target cabang lagi. Apabila tidak maka akan ke proses selanjutnya, yaitu pembuatan *multiple-population keturunan*.



Gambar 1. Metode yang Diusulkan

Setelah populasi keturunan baru tercipta, selanjutnya akan dilakukan proses pemilihan populasi *parent* dan populasi keturunan yang nantinya akan dilakukan proses *crossover*. Proses mutasi akan dilanjutkan setelah proses *crossover* selesai dikerjakan. Setelah itu sub-populasi keturunan yang sudah melalui operasi genetik dimasukkan ke dalam PUT. Jika terdapat individu yang sesuai dengan kriteria maka akan dilakukan penyimpanan ke *archive temporary population*. Langkah berikutnya melakukan pengecekan maksimal generasi yang sudah ditentukan. Apabila generasi saat ini sama dengan batas maksimal generasi, maka pencarian akan dihentikan, jika tidak maka proses berikutnya akan dikerjakan. Selanjutnya adalah proses evaluasi *fitness* menggunakan

pareto fitness. Setelah itu sub-populasi baru akan diciptakan berdasarkan evaluasi *fitness* yang sudah didapatkan. Proses pemilihan target cabang kembali dilakukan, dan mengulangi proses selanjutnya sampai target cabang sudah tidak ada atau batas maksimal generasi sudah terpenuhi.

4 HASIL EKSPERIMEN

Eksperimen dilakukan pada komputer dengan spesifikasi processor intel intel core 2 duo, 2 GB RAM, dan Sistem Operasi Windows 8. Lingkungan pengembangan program menggunakan Netbeans IDE 8 dengan bahasa pemrograman Java.

Di dalam penelitian ini menggunakan *benchmark* data program publik yang juga digunakan oleh peneliti lain seperti oleh Pachauri & Srivastava (Pachauri & Srivastava, 2013) dan Ferrer et al (Ferrer et al., 2012). Penggunaan *benchmark* program publik memungkinkan orang lain dapat memvalidasi hasil penelitian yang dilakukan. Pada penelitian ini menggunakan tiga *benchmark* program yang banyak digunakan oleh peneliti, yaitu *Myers Triangle*, *Michael Triangle* dan *Sthamer Triangle* (Pachauri & Srivastava, 2013)(Ferrer et al., 2012). Berikut ini merupakan informasi *benchmark* program yang digunakan:

- *Myers Triangle.* Program ini menggolongkan segitiga atas dasar sisi input sebagai non-segitiga atau segitiga, yaitu, sama kaki, sama sisi atau sisi tak sama panjang. Diberikan tiga input nilai di dalam parameter yang semuanya mewakili sisi segitiga. *Control flow graph* yang dihasilkan memiliki 14 *node* dengan predikat *node* berjumlah 6. Memiliki kondisi kesetaraan dengan operator DAN, yang membuat cabang sulit untuk dilewati.
- *Michael Triangle.* Program ini juga menggolongkan segitiga atas dasar sisi input sebagai non-segitiga atau segitiga, yaitu, sama kaki, sama sisi atau sisi tak sama panjang. Dibutuhkan tiga input nilai dengan semua dari ketiganya mewakili sisi segitiga tetapi dengan kondisi predikat yang berbeda. *Control flow graph* yang dihasilkan memiliki 26 *node* dengan jumlah predikat *node* 11.
- *Sthamer Triangle.* Seperti benchmark program sebelumnya, program ini juga menggolongkan segitiga atas dasar sisi input sebagai non-segitiga atau segitiga, yaitu, sama kaki, sama sisi, segitiga sudut kanan atau sisi tak sama panjang. *Control flow graph* yang dihasilkan memiliki 29 *node* dengan jumlah predikat *node* 13. Program ini memiliki kondisi persamaan dengan operator DAN dan operator relasional yang kompleks

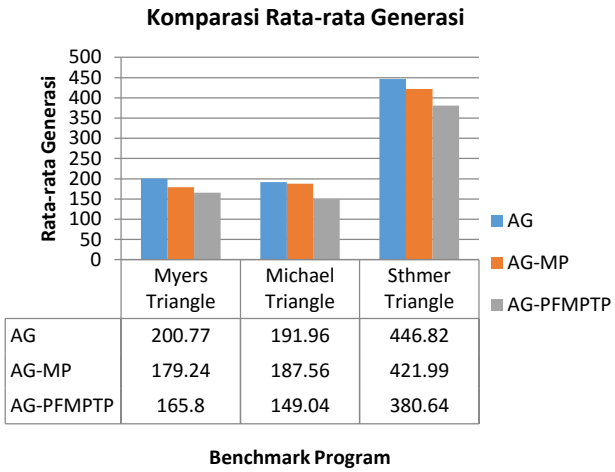
Pengujian akan dilakukan dengan masing-masing benchmark program yang digunakan. Penentuan parameter dan operator berdasarkan hasil beberapa percobaan yang dilakukan oleh peneliti. Dari beberapa percobaan didapatkan beberapa pengaturan yang cukup memadai untuk digunakan di dalam pengujian. Tabel 1 menunjukkan pengaturan parameter dan operator yang digunakan. Hal ini juga dilakukan oleh peneliti lain yaitu Yao & Gong (Yao & Gong, 2014).

Pengujian yang dilakukan bertujuan untuk mengetahui jumlah rata-rata generasi dan rata-rata *coverage*. Pengujian dilakukan sebanyak seratus kali untuk rata-rata generasi dan tiga puluh kali untuk rata-rata *coverage*.

Tabel 1. Penentuan Parameter dan Operator

Parameter/ Operator	Benchmark Program		
	Myers	Michaels	Sthmer
Jumlah kromosom	60	60	60
Panjang bit	6	6	6
Range nilai data tes	0-64	0-64	0-64
Seleksi	Random	Random	Random
Crossover	Single	Single	Single
Survive probability	0.5	0.5	0.5
Mutation probability	0.15	0.15	0.15
Jumlah subpopulasi	6	6	6
Maks generasi	1000	1000	1000
Jumlah Generasi	200	200	500

Dari Gambar 2 dapat dilihat bahwa metode AG-MP memiliki generasi lebih sedikit dibandingkan metode AG dari pengujian yang dilakukan pada semua benchmark program. Pada benchmark program *Myers Triangle* rata-rata generasi berkurang sebesar 21,53. Sedangkan pada pengujian yang dilakukan menggunakan benchmark program *Michael Triangle* metode AG dan AG-MP tidak terlalu jauh berbeda nilai dari rata-rata generasinya yang hanya berkurang 4,4. Kemudian untuk benchmark program *Sthmer Triangle* nilai rata-rata generasi berkurang sebesar 24,83.

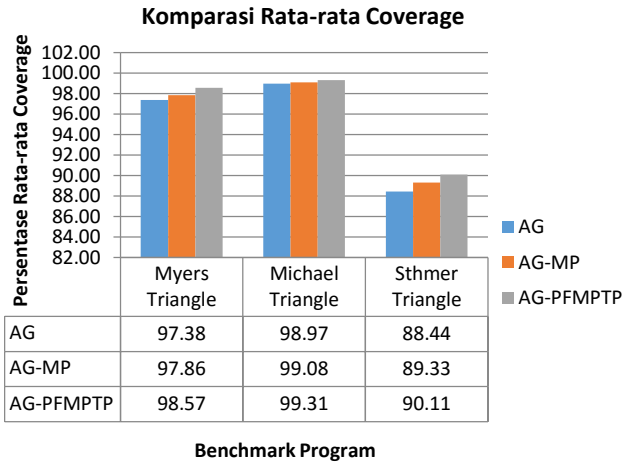


Gambar 2. Komparasi Rata-rata Generasi

Kemudian secara konsisten metode yang diusulkan yaitu metode AG-PFMPTP memiliki generasi yang paling sedikit dibandingkan dengan metode AG ataupun AG-MP pada semua benchmark program. Untuk benchmark program *Myers Triangle* metode AG-PFMPTP berkurang 34,97 untuk rata-rata generasinya dari metode AG. Sedangkan jika dibandingkan dengan metode AG-MP, rata-rata generasi metode AG-PFMPTP berkurang 13,44. Kemudian benchmark program *Michael Triangle* rata-rata generasi metode AG-PFMPTP berkurang 42,92 dari metode AG dan 38,52 dari metode AG-MP. Sedangkan untuk benchmark program *Sthmer Triangle* rata-rata generasi metode AG-PFMPTP berkurang 66,18 dari metode AG dan 41,35 dari metode AG-MP. Dari hasil komparasi tersebut, dapat disimpulkan bahwa metode AG-PFMPTP lebih efisien dalam mendapatkan data tes dibandingkan dengan kedua metode yang lainnya.

Dari Gambar 3 menunjukkan bahwa metode AG-MP memiliki nilai persentase *coverage* lebih besar dibandingkan metode AG dengan selisih 0,48% pada benchmark program *Myers Triangle*, selisih 0,11% pada benchmark program *Michael Triangle* dan selisih 0,89% pada benchmark program

Sthmer Triangle. Selisih terkecil terdapat pada benchmark program *Michael Triangle*. Kemudian metode AG-PFMPTP memiliki nilai persentase *coverage* lebih besar dibandingkan dengan metode AG. Metode AG-PFMPTP memiliki selisih persentase dari metode AG sebesar 1,19% untuk benchmark program *Myers Triangle*, 0,34% untuk benchmark program *Michael Triangle* dan 1,67% untuk benchmark program *Sthmer Triangle*. Begitupun jika dibandingkan dengan metode AG-MP, metode AG-PFMPTP lebih besar nilai persentase rata-rata *coverage* yang dihasilkan. Metode AG-PFMPTP memiliki selisih persentase dari metode AG sebesar 0,71% untuk benchmark program *Myers Triangle*, 0,23% untuk benchmark program *Michael Triangle* dan 0,78% untuk benchmark program *Sthmer Triangle*.



	Myers Triangle	Michael Triangle	Sthmer Triangle
AG	97.38	98.97	88.44
AG-MP	97.86	99.08	89.33
AG-PFMPTP	98.57	99.31	90.11

Gambar 3. Komparasi Rata-rata Coverage

Dari ketiga metode tersebut bisa dilihat tidak terlalu jauh peningkatan nilai persentase *coverage* yang dihasilkan. Jika dilihat pada hasil komparasi rata-rata *coverage* untuk benchmark program *Michael Triangle* sangat tipis sekali perbedaan nilai persentase dari ketiga metode. Walaupun demikian metode yang diusulkan, yaitu AG-PFMPTP memiliki nilai persentase *coverage* yang paling besar diantara dua metode yang lain. Hal tersebut menandakan metode AG-PFMPTP lebih banyak mencakup data tes yang dibutuhkan.

Untuk memvalidasi hasil evaluasi hasil pengujian metode AG, AG-MP dan AG-PFMPTP yang telah dilakukan signifikan berbeda atau tidak, maka akan dilakukan pengujian statistik uji beda. Sebelum dilakukan penentuan penggunaan metode uji beda yang digunakan, terlebih dahulu dilakukan uji normalitas data. Karena data yang didapatkan memiliki distribusi normal maka dilakukan pengujian parametrik dengan metode t-test. Uji normalitas data yang digunakan menggunakan metode Shapiro-Wilk (Razali & Wah, 2011).

Uji beda untuk rata-rata generasi bertujuan untuk mengetahui apakah ada perbedaan yang signifikan antara metode AG, AG-MP dan AG-PFMPTP dalam kecepatan proses pencarian data tes. Tabel 2 merupakan hasil t-test rata-rata generasi untuk masing-masing perbandingan metode. Masing-masing metode akan dibandingkan dengan metode lain. Metode AG akan dibandingkan dengan AG-MP, metode AG dengan AG-PFMPTP dan metode AG-MP dengan metode AG-PFMPTP.

Dari Tabel 2 dapat dilihat hasil Sig.(2-tailed) menghasilkan nilai yang berbeda untuk masing-masing perbandingan nilai. P-value dari masing-masing perbandingan dihitung dari hasil Sig.(2-tailed) dibagi dua. Dari hasil tersebut dapat disimpulkan:

1. Hasil uji beda rata-rata generasi metode AG dengan AG-MP menghasilkan nilai sig.(2-tailed) 0.116, oleh karena itu P-value yang dihasilkan adalah 0.056. Dari nilai tersebut dapat disimpulkan bahwa H0 diterima karena 0.056 > 0.5 yang artinya tidak terdapat perbedaan yang signifikan antara metode AG dan AG-MP.
2. Hasil uji beda rata-rata generasi metode AG dengan AG-PFMPTP menghasilkan nilai sig.(2-tailed) 0.036, oleh karena itu P-value yang dihasilkan adalah 0.018. Dari nilai tersebut dapat disimpulkan bahwa H0 ditolak karena 0.018 < 0.05 yang artinya terdapat perbedaan yang signifikan antara metode AG dan AG-PFMPTP.
3. Hasil uji beda rata-rata generasi metode AG-MP dengan AG-PFMPTP menghasilkan nilai sig.(2-tailed) 0.073, oleh karena itu P-value yang dihasilkan adalah 0.037. Dari nilai tersebut dapat disimpulkan bahwa H0 ditolak karena 0.037 < 0.05 yang artinya terdapat perbedaan yang signifikan antara metode AG-MP dan AG-PFMPTP.

Tabel 2. Uji Beda Rata-rata Generasi

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	AG - AG_MP	16.92000	10.96747	6.33207	-10.32469	44.16469	2.672	2	.116
Pair 2	AG - AG_PFMPTP	48.02333	16.21879	9.36392	7.73363	88.31303	5.129	2	.036
Pair 3	AG_MP - AG_PFMPTP	31.10333	15.36220	8.86937	-7.05849	69.26516	3.507	2	.073

Untuk hasil uji beda rata-rata generasi dapat dilihat bahwa untuk metode AG dan AG-MP tidak terdapat perbedaan yang signifikan. Dari penelitian Yao & Gong (Yao & Gong, 2014) menyatakan bahwa AG dengan menggunakan *multiple-population* signifikan berbeda dengan AG standar. Hal yang menyebabkan perbedaan ini dapat timbul dari pengaturan parameter & operator, sistem perangkat lunak, konversi program yang berbeda dan tujuan pengujian (Yao & Gong, 2014). Sedangkan hasil uji beda rata-rata generasi untuk metode AG-PFMPTP terdapat perbedaan yang signifikan baik dibandingkan dengan AG maupun AG-MP. Dengan melihat hasil uji beda tersebut dan komparasi metode yang telah dilakukan, maka metode AG-PFMPTP yang merupakan integrasi *pareto fitness*, *multiple-population* dan *temporary population* dapat meningkatkan kinerja dari algoritma genetika secara signifikan dalam melakukan proses pencarian data tes untuk pengujian perangkat lunak.

Selanjutnya untuk uji beda rata-rata *coverage* bertujuan untuk mengetahui apakah ada perbedaan yang signifikan antara metode AG, AG-MP dan AG-PFMPTP dalam mencakup data tes yang didapatkan berdasarkan batasan generasi tertentu. Tabel 3 merupakan hasil t-test rata-rata generasi untuk masing-masing perbandingan metode. Masing-masing metode akan dibandingkan dengan metode lain.

Metode AG akan dibandingkan dengan AG-MP, metode AG dengan AG-PFMPTP dan metode AG-MP dengan metode AG-PFMPTP.

Tabel 3. Uji Beda Rata-rata Coverage

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	GA - GA_MP	-.49334	.38726	.22358	-1.45534	.46866	-2.207	2	.158
Pair 2	GA - GA_PFMPTP	-1.06732	.66947	.38652	-2.73038	.59573	-2.761	2	.110
Pair 3	GA_MP - GA_PFMPTP	-.57398	.29968	.17302	-1.31844	.17047	-3.317	2	.080

Dari Tabel 3 dapat dilihat hasil Sig.(2-tailed) menghasilkan nilai yang berbeda untuk masing-masing perbandingan nilai. *P-value* dari masing-masing perbandingan dihitung dari hasil Sig.(2-tailed) dibagi dua seperti yang dilakukan pada uji beda rata-rata generasi. Dari hasil tersebut dapat disimpulkan:

1. Hasil uji beda rata-rata *coverage* metode AG dengan AG-MP menghasilkan nilai sig.(2-tailed) 0.158, oleh karena itu *P-value* yang dihasilkan adalah 0.079. Dari nilai tersebut dapat disimpulkan bahwa H0 diterima karena 0.079 > 0.05 yang artinya tidak terdapat perbedaan yang signifikan antara metode AG dan AG-MP.
2. Hasil uji beda rata-rata *coverage* metode AG dengan AG-PFMPTP menghasilkan nilai sig.(2-tailed) 0.110, oleh karena itu *P-value* yang dihasilkan adalah 0.055. Dari nilai tersebut dapat disimpulkan bahwa H0 diterima karena 0.055 > 0.05 yang artinya tidak terdapat perbedaan yang signifikan antara metode AG dan AG-PFMPTP.
3. Hasil uji beda rata-rata *coverage* metode AG-MP dengan AG-PFMPTP menghasilkan nilai sig.(2-tailed) 0.080, oleh karena itu *P-value* yang dihasilkan adalah 0.040. Dari nilai tersebut dapat disimpulkan bahwa H0 ditolak karena 0.040 < 0.05 yang artinya terdapat perbedaan yang signifikan antara metode AG-MP dan AG-PFMPTP.

Dari hasil uji beda rata-rata *coverage* dapat dilihat bahwa untuk metode AG dan AG-MP tidak terdapat perbedaan yang signifikan. Kemudian hasil uji beda rata-rata *coverage* untuk metode AG dan AG-PFMPTP tidak terdapat perbedaan yang signifikan pula. Sedangkan metode AG-MP dan AG-PFMPTP terdapat perbedaan yang signifikan. Dengan melihat hasil uji beda dan komparasi metode yang telah dilakukan, metode AG-PFMPTP yang merupakan integrasi *pareto fitness*, *multiple-population* dan *temporary population* secara signifikan lebih baik dibandingkan metode AG-MP dalam mencakup data tes yang dibutuhkan untuk pengujian perangkat lunak.

5 KESIMPULAN

Dari hasil penelitian yang sudah dilakukan dari tahap analisa permasalahan dan *literature riview* sampai tahap validasi hasil evaluasi pengujian, telah diketahui hasil peningkatan kinerja dari metode yang diusulkan. Integrasi *pareto fitness*, *multiple-population* dan *temporary population* (AG-PFMPTP) dapat meningkatkan kinerja algoritma genetika dalam pencarian data tes yang digunakan untuk pengujian perangkat lunak dibandingkan agoritma genetika standar (AG) dan algortima genetika dengan menggunakan multiple-population (AG-MP).

Dari hasil pengujian rata-rata generasi secara konsisten metode yang diusulkan dapat memperkecil nilai rata-rata generasi dalam proses menghasilkan data tes pada semua benchmark program yang digunakan. Hal ini menunjukkan metode AG-PFMPTP lebih cepat dalam menemukan data tes dibandingkan metode AG ataupun AG-MP. Kemudian untuk hasil pengujian rata-rata *coverage* metode AG-PFMPTP mencakup lebih banyak data tes yang dibutuhkan dibandingkan metode AG dan AG-MP. Meskipun kenaikan nilai rata-rata *coverage* dari metode AG-PFMPTP tidak terlalu besar.

Pada proses validasi hasil menggunakan t-test untuk uji beda rata-rata generasi, metode AG-PFMPTP secara signifikan terdapat perbedaan jika dibandingkan dengan metode AG dan AG-MP. Hal ini menunjukkan metode AG-PFMPTP lebih cepat dalam mendapatkan data tes yang dibutuhkan dibandingkan dengan metode AG dan AG-MP. Kemudian untuk validasi uji beda rata-rata *coverage* didapatkan hasil bahwa metode AG-PFMPTP tidak terdapat perbedaan yang signifikan dalam mencakup data tes yang didapatkan jika dibandingkan dengan metode AG. Akan tetapi terdapat perbedaan yang signifikan apabila metode AG-PFMPTP dibandingkan dengan metode AG-MP.

REFERENSI

Alakeel, A. (2014). Using Fuzzy Logic Techniques for Assertion-Based Software Testing Metrics. *The Scientific World Journal*. Retrieved from http://www.hindawi.com/journals/tswj/aa/629430/

Aleti, A., & Grunske, L. (2015). Test data generation with a Kalman filter-based adaptive genetic algorithm. *Journal of Systems and Software*, *103*, 343–352. http://doi.org/10.1016/j.jss.2014.11.035

Ali, S., Briand, L. C., Hemmati, H., & Panesar-Walawege, R. K. (2010). A systematic review of the application and empirical investigation of search-based test case generation. *IEEE Transactions on Software Engineering*, *36*(6), 742–762. http://doi.org/10.1109/TSE.2009.52

Alshraideh, M., Mahafzah, B. a., & Al-Sharaeh, S. (2011). A multiple-population genetic algorithm for branch coverage test data generation. *Software Quality Journal*, *19*, 489–513. http://doi.org/10.1007/s11219-010-9117-4

Anand, S., Burke, E. K., Yueh, T., Clark, J., Cohen, M. B., Grieskamp, W., … Zhu, H. (2015). The Journal of Systems and Software An orchestrated survey of methodologies for automated software test case generation Orchestrators and Editors , *86*(2013), 1978–2001. http://doi.org/10.1016/j.jss.2013.02.061

Bueno, P. M. S., Jino, M., & Wong, W. E. (2014). Diversity oriented test data generation using metaheuristic search techniques. *Information Sciences*, *259*, 490–509. http://doi.org/10.1016/j.ins.2011.01.025

Cantu-paz, E. (1997). A Survey of Parallel Genetic Algorithms. *Department of Computer Science and Illinois Genetic Algorithms Laboratory University of Illinois at Urbana-Champaign*, *10*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.106.

389

Cohen, M. B., Colbourn, C. J., & Ling, a. C. H. (2003). Augmenting simulated annealing to build interaction test suites. *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.* http://doi.org/10.1109/ISSRE.2003.1251061

Díaz, E., Tuya, J., & Blanco, R. (2003). Automated software testing using a metaheuristic technique based on tabu search. *Automated Software Engineering, …*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1240327

Elaoud, S., Loukil, T., & Teghem, J. (2007). The Pareto fitness genetic algorithm: Test function study. *European Journal of Operational Research*, *177*(3), 1703–1719. http://doi.org/10.1016/j.ejor.2005.10.018

Ferrer, J., Chicano, F., & Alba, E. (2012). Evolutionary algorithms for the multi-objective test data generation problem. *Software: Practice and Experience*, *42*(11), 1331–1362. http://doi.org/10.1002/spe.1135

Ferrer, J., Kruse, P. M., Chicano, F., & Alba, E. (2015). Search based algorithms for test sequence generation in functional testing. *Information and Software Technology*, *58*, 419–432. http://doi.org/10.1016/j.infsof.2014.07.014

Harman, M., & McMinn, P. (2010). A Theoretical and Empirical Study of Search Based Testing: Local, Global and Hybrid Search. *IEEE Transactions on Software Engineering*, *36*(2), 226–247. http://doi.org/http://dx.doi.org/10.1109/TSE.2009.71

Hermadi, I., Lokan, C., & Sarker, R. (2014). Dynamic stopping criteria for search-based test data generation for path testing. *Information and Software Technology*, *56*(4), 395–407. http://doi.org/10.1016/j.infsof.2014.01.001

Lakhotia, K., McMinn, P., & Harman, M. (2009). Automated test data generation for coverage: Haven't we solved this problem yet? *TAIC PART 2009 - Testing: Academic and Industrial Conference - Practice and Research Techniques*, 95–104. http://doi.org/10.1109/TAICPART.2009.15

Mao, C. (2014). Generating Test Data for Software Structural Testing Based on Particle Swarm Optimization. *Arabian Journal for Science and Engineering*, *39*, 4593–4607. http://doi.org/10.1007/s13369-014-1074-y

Mao, C., Xiao, L., Yu, X., & Chen, J. (2015). Adapting ant colony optimization to generate test data for software structural testing. *Swarm and Evolutionary Computation*, *20*, 23–36. http://doi.org/10.1016/j.swevo.2014.10.003

McMinn, P. (2004). Search-based software test data generation: a survey. *Software Testing, Verification and Reliability*, 1–58. http://doi.org/10.1002/stvr.v14:2

McMinn, P. (2011). Search-Based Software Testing: Past, Present and Future. *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*. http://doi.org/10.1109/ICSTW.2011.100

McMinn, P., Harman, M., Lakhotia, K., Hassoun, Y., & Wegener, J. (2012). Input Domain Reduction through Irrelevant Variable Removal and Its Effect on Local, Global, and Hybrid Search-Based Structural Test Data Generation. *IEEE Transactions on Software Engineering*, *38*(2), 453–477. http://doi.org/10.1109/TSE.2011.18

Miller, J., Reformat, M., & Zhang, H. (2006). Automatic test data generation using genetic algorithm and program dependence graphs. *Information and Software Technology*, *48*, 586–605. http://doi.org/10.1016/j.infsof.2005.06.006

Pachauri, A., & Srivastava, G. (2013). Automated test data generation for branch testing using genetic algorithm: An improved approach using branch ordering, memory and elitism. *Journal of Systems and Software*, *86*(5), 1191–1208. http://doi.org/10.1016/j.jss.2012.11.045

Patil, M., & Nikumbh, P. J. (2012). Pair-wise Testing Using Simulated Annealing. *Procedia Technology*, *4*, 778–782. http://doi.org/10.1016/j.protcy.2012.05.127

Razali, N. M., & Wah, Y. B. (2011). Power comparisons of Shapiro-Wilk , Kolmogorov-Smirnov , Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, *2*(1), 21–33.

Ribeiro, J. C. B. (2008). Search-based test case generation for object-oriented java software using strongly-typed genetic programming. *Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation - GECCO '08*, 1819. http://doi.org/10.1145/1388969.1388979

Srivastava, P. R., & Baby, K. (2010). Automated Software Testing Using Metahurestic Technique Based on an Ant Colony Optimization. *2010 International Symposium on Electronic System Design*, 235–240. http://doi.org/10.1109/ISED.2010.52

Srivastava, P. R., & Kim, T. (2009). Application of Genetic Algorithm in Software Testing. *Intenational Journal of Software Engineering and Its Applications*, *3*(4), 87–96. Retrieved from http://www.sersc.org/journals/IJSEIA/vol3_no4_2009/6.pdf

Vos, T. E. J., Lindlar, F. F., Wilmes, B., Windisch, A., Baars, A. I., Kruse, P. M., … Wegener, J. (2013). Evolutionary functional black-box testing in an industrial setting. *Software Quality Journal*, *21*, 259–288. http://doi.org/10.1007/s11219-012-9174-y

Yao, X., & Gong, D. (2014). Genetic Algorithm-Based Test Data Generation for Multiple Paths via Individual Sharing. *Computational Intelligence and Neuroscience*, *2014*, 1–12. http://doi.org/10.1155/2014/591294

BIOGRAFI PENULIS

Mohammad Reza Maulana. Memperoleh gelar S.Kom pada bidang ilmu komputer di Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Widya Pratama dan M.Kom pada bidang ilmu komputer di Universitas Dian Nuswantoro. Saat ini menjadi pengajar di STMIK Widya Pratama Pekalongan. Minat penelitian pada bidang software engineering.

Romi Satria Wahono. Memperoleh gelar B.Eng dan M.Eng pada bidang ilmu komputer di Saitama University, Japan, dan Ph.D pada bidang software engineering di Universiti Teknikal Malaysia Melaka. Menjadi pengajar dan peneliti di Fakultas Ilmu Komputer, Universitas Dian Nuswantoro. Merupakan pendiri dan CEO PT Brainmatics, sebuah perusahaan yang bergerak di bidang pengembangan software. Minat penelitian pada bidang software engineering dan machine learning. Profesional member dari asosiasi ilmiah ACM, PMI dan IEEE Computer Society.

Catur Supriyanto. Dosen di Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Semarang, Indonesia. Menerima gelar master dari Universiti Teknikal Malaysia Melaka (UTEM), Malaysia. Minat penelitiannya adalah *information retrieval, machine learning, soft computing* dan *intelligent system.*