 ISSN:2302-8149
e-ISSN:2540-9719

Optimasi Kecepatan *Continuous Integration/ Continuous Delivery* pada Otomatisasi Jaringan berbasis *Grey Wolf Optimizer*

Continuous Integration/ Continuous Delivery Optimization on Network Automation using Gray Wolf Optimizer

Ronald Adrian*, Anni Karimatul Fauziyyah, Sahirul Alam
Teknologi Rekayasa Internet, Departemen Teknik Elektro dan Informatika, Sekolah Vokasi, Universitas Gadjah Mada
Sekip Unit III Blimbing Sari Caturtunggal Depok Sleman Yogyakarta 55281, Indonesia
*e-mail: ronald.adr@ugm.ac.id

(***received*:** 10 September 2022, ***revised*:** 22 September 2022, ***accepted*:** 22 September 2022)

Abstrak

Continuous Integration/ Continuous Delivery merupakan metode terkini yang dipergunakan dalam otomatisasi jaringan. Pemrograman dalam jaringan telah banyak membantu admin jaringan dalam mengelola semua perangkatnya. Salah satu kebutuhan jaringan yang bersifat *real-rime*, memaksa admin jaringan untuk dapat menyediakan data secara cepat. Kecepatan deployment dapat ditingkatkan dalam rangka untuk menyediakan data atau konfigurasi jaringan yang sesuai waktu terkini. Untuk mengatasi permasalahan tersebut, penelitian ini mengusulkan implementasi algoritme GWO dalam proses *Continuous Integration/Continuous Delivery*. Algoritme ini terbukti unggul dalam kecepatan pencarian nilai fungsi tujuan dibandingkan dengan algoritme lain yang sejenis. Hasil yang didapatkan menunjukkan bahwa waktu konvergensi lebih cepat sebesar 74%. Nilai ini berdampak pada peningkatan kecepatan *deployment* program sebesar 41,2%. Hasil ini menunjukkan bahwa algoritme GWO dapat menjadi alternatif dalam peningkatan kecepatan *Continuous Integration/Continuous Delivery*.

Kata kunci: algoritme, wolf, konvergensi, optimisasi, otomasi

Abstract

Continuous Integration/Continuous Delivery is the latest method used in network automation. In-network programming has helped network admins a lot in managing all their devices. One of the real-time networks needs to force network admins to be able to provide data quickly. Deployment speed can be increased to provide up-to-date data or network configuration. To tackle these problems, we propose implementing the GWO algorithm in the Continuous Integration/Continuous Delivery process. This algorithm is proven to be superior in the speed of finding the value of the objective function compared to other similar algorithms. The results obtained indicate that the convergence time is faster by 74%. This value has an impact on increasing program deployment speed by 41.2%. These results indicate that the GWO algorithm can be an alternative to increasing the speed of Continuous Integration/ Continuous Delivery.

Keywords: *algorithm, wolf, convergence, optimization, automation*

1 Pendahuluan

Pemrograman telah menjadi tren baru dalam pembuatan sistem otomatisasi pada jaringan internet [1]. Sistem otomatisasi ini akan mempermudah admin dalam melakukan proses konfigurasi dan pengawasan jaringan. Skema permrograman, penyebaran, dan pengujian aplikasi yang umum digunakan adalah CI/CD [2]. Proses *Continuous Integration* (CI) memberikan sejumlah manfaat. Hal ini dikarenakan setiap pengiriman program dapat memberikan peluang bagi sistem untuk melakukan proses tambahan lainnya. Beberapa proses tambahan yang dapat dilakukan antara lain adalah kompilasi

kode program, pengujian unit program, analisis kode program, dan pengujian integrasi. *Continuous Delivery* (CD) adalah proses pengembangan dalam rentang yang cukup pendek sehingga kode program selalu dalam kondisi *deployable*. *Continuous Integration* (CI), perubahan kecil pada program dapat terus diintegrasikan ke dalam kode program utama. *Continuous Delivery* (CD) berarti bahwa perubahan tersebut dapat direkayasa secara mandiri untuk mendapatkan fungsi tertentu pada waktu tertentu sesuai dengan keinginan admin [3].

CI/CD membantu *programmer* untuk melakukan otomatisasi pengembangan perangkat lunak pada jaringan. Proses otomatisasi ini memerlukan waktu yang tidak sedikit. Kecepatan pemrosesan pada CI/CD dapat ditingkatkan agar dapat mempersingkat waktu penyebaran aplikasi [4]. Penelitian ini berfokus pada optimasi kecepatan yang akan berdampak pada singkatnya waktu penyebaran aplikasi pada program.

Peningkatan optimasi dapat dilakukan dengan cara menambah katalis untuk dapat mempercepat proses tersebut. Penelitian ini mengusulkan metode optimasi kecepatan CI/CD dengan menggunakan algoritme *Grey Wolf Optimizer* (GWO) sebagai katalis untuk mempercepat penyebaran aplikasi pada program. GWO merupakan salah satu algoritme metaheuristik terbaru selain algoritme *moth flame optimization* (MFO), dan BAT. Algoritme GWO memiliki keunggulan dalam penyelesaian masalah multi variable karena memiliki empat hirarki proses pencarian yang bekerja secara bersama-sama. Empat hirarki tersebut adalah *alfa*, *beta*, *gamma*, dan *omega* [5].

Tujuan penelitian ini adalah membantu admin dalam mempercepat proses penyebaran kode program dengan menggunakan metode terkini agar pekerjaannya menjadi lebih cepat. Pemilihan algoritme GWO ini juga telah melewati beberapa tahap pengujian sehingga diperoleh bahwa algoritme GWO memiliki kecepatan penemuan nilai fungsi tujuan yang terbaik. Pengujian pembandingan metode ini melibatkan algoritme metaheuristik lainnya seperti algoritme *Particle Swarm Optimization* (PSO) [6], BAT [7], *Moth Flame Optimization* (MFO) [8], dan *Modified Moth Flame Optimization* (MMFO) [9].

2 Tinjauan Literatur

Popularitas *continuous integration* (CI) meningkat sebagai solusi untuk mempermudah pembaharuan program. Kemampuan CI dalam memberikan kualitas dengan kecepatan bergantung pada otomatisasi pengujian yang andal [10]. Data survei [11] menyajikan studi empiris untuk mengamati pengaruh kematangan otomatisasi uji pada kualitas otomatisasi pengujian dan siklus rilis dalam konteks CI dari aplikasi *open-source*. Penelitian tersebut menekankan pada survei kematangan otomatisasi pengujian dari 37 proyek *open-source*. Hasil utama dari analisis regresi yang telah dilakukan mengungkapkan bahwa tingkat kematangan otomatisasi pengujian lebih tinggi secara positif sejalan dengan kualitas produk tinggi (nilai p=0,000624). Siklus rilis yang dilakukan menjadi lebih pendek (nilai p=0,01891). Penelitian tersebut memiliki kesimpulan bahwa manfaat potensial dari peningkatan kematangan otomatisasi pengujian (menggunakan praktik standar terbaik) adalah peningkatan kualitas produk dan percepatan siklus rilis dalam konteks CI pada aplikasi *open-source*. Penelitian ini membuka peluang dalam memperluas metode yang telah dipergunakan dengan menambahkan lebih banyak kumpulan data dalam bahasa pemrograman dan perangkat CI yang berbeda, aplikasi berbayar, dan proyek industri dalam skala besar. Rekomendasi yang diberikan adalah menggunakan praktik terbaik yang sesuai dengan standar untuk meningkatkan kematangan otomatisasi pengujian [11]

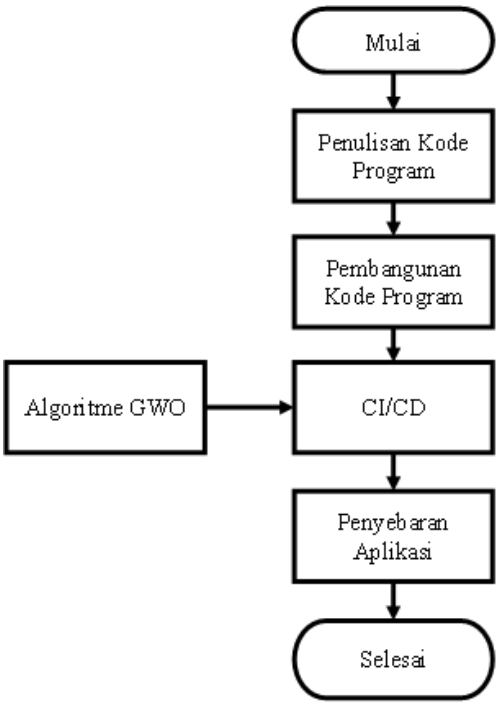
Gray Wolf Optimizer (GWO) merupakan algoritme yang terinspirasi oleh serigala abu-abu (*Canis lupus*) [15]. Algoritme GWO ini meniru hirarki kepemimpinan dan mekanisme berburu dari serigala abu-abu. Empat jenis serigala abu-abu yang memiliki identitas sebagai *alfa*, *beta*, *delta*, dan *omega* dipergunakan untuk mensimulasikan hirarki kepemimpinan. Selain itu, metode ini menerapkan tiga langkah utama berburu yaitu pencarian mangsa, pelingkatan mangsa, dan penyerangan mangsa. Algoritme tersebut telah dilakukan pengujian dengan menggunakan 29 fungsi pengujian yang terkenal. Hasil yang didapatkan telah diverifikasi melalui studi perbandingan dengan *Particle Swarm Optimization* (PSO), *Gravitational Search Algorithm* (GSA) [12], *Differential Evolution* (DE) [13], *Evolutionary Programming* (EP) [14], dan *Evolution Strategy* (ES). Perbandingan pengujian tersebut menunjukkan bahwa algoritme GWO mampu memberikan hasil yang sangat kompetitif dibandingkan dengan algortime metaheuristik lainnya. Algoritme ini terbukti dapat menyelesaikan permasalahan yang kompleks dengan sangat baik [15].

Gray Wolf Optimizer (GWO) adalah salah satu algoritme metaheuristik berbasis *Swarm Intelligence* yang mebggunakan perilaku serigala dengan mekanisme kepemimpinan dan strategi berburu yang terencana dengan baik. Adhikary *et al.* (2022) telah memperkenalkan varian baru GWO yang disebut sebagai *Randomized Balanced Grey Wolf Optimizer* (RBGWO). Metode ini membantu serigala menjelajahi ruang pencarian dengan cara yang efisien. Algoritme yang diusulkan dapat meningkatkan efisiensi secara keseluruhan dari proses pencarian dengan membangun keseimbangan antara eksploitasi dan kemampuan eksplorasi dengan menggabungkan tiga strategi peningkatan berturut-turut. Hal ini dilengkapi dengan mekanisme hierarki sosial dan cara pemilihan dari nomor yang terdistribusi secara acak. Varian RBGWO yang baru diusulkan ini telah mengungguli GWO dan varian sebelumnya (RW-GWO, EGWO+ dan EGWO∗). Keunggulan ini terbukti dari pengujian sebagian besar kasus pada fungsi *benchmark* CEC 2014 dengan skala yang berbeda. Hasil dari varian yang diusulkan juga telah diverifikasi dengan algoritme metaheuristik lainnya seperti GSA, CS, TPHS, CL-PSO, LX-BBO, B-BBO, SOS, DERand1Bin, Firefly, GWO, RW-GWO, EGWO+ dan EGWO pada fungsi *benchmark* CEC 2014. Analisis statistik hasil menyajikan efisiensi yang baik dari RBGWO yang diusulkan dalam kinerja secara keseluruhan [16].

Penelitian yang dilakukan adalah mengimplementasikan algoritme GWO kedalam proses CI/CD. Berdasarkan dari penelitian-penelitian sebelumnya, algoritme GWO belum pernah diujicobakan dalam implementasi CI/CD. Implementasi algoritme GWO ini diharapkan mampu membuka peluang pada masa depan bahwa proses CI/CD dapat dipercepat agar esensi kecepatan dalam otomatisasi jaringan tidak terabaikan.

3 Metode Penelitian

Metode penelitian ini memiliki rancangan diagram alir seperti yang ditampilkan pada Gambar 1. Tahapan penelitian ini dimulai dari penulisan kode program, pembangunan kode program, CI/CD dengan menggunakan algoritme *Golf Wave Optimizer* (GWO) dan diakhiri dengan penyebaran aplikasi yang telah lulus tes. GWO menjadi komponen utama dalam rangka mempercepat proses pengetesan otomatis.

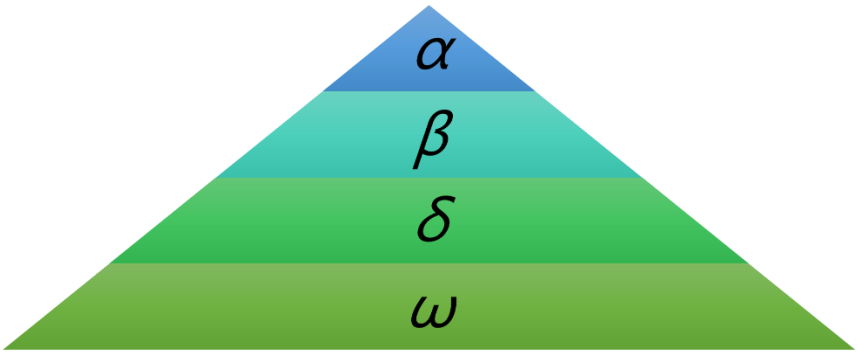


Gambar 1. Diagram Alir Impelementasi CI/CD berbasis GWO

Algoritme *Grey Wolf Optimizer* (GWO) memiliki tahapan sebagai berikut:

 ISSN:2302-8149
e-ISSN:2540-9719

1. Hirarki Sosial
   GWO memiliki hirarki sosial dengan analogi *n* untuk memodelkan hierarki sosial dari serigala secara matematis saat inisiasi GWO. Algoritme ini menganggap solusi yang paling cocok sebagai *alfa* (α). Akibatnya, solusi terbaik kedua dan ketiga masing-masing diberikan nama *beta* (β) dan *delta* (δ). Sisa dari kandidat solusi tersebut diasumsikan sebagai *omega* (ω). Dalam algoritme GWO, perburuan (optimasi) dipandu oleh α, β, dan δ. Kandidat solusi *omega* (ω) akan mengikuti pergerakan hasil dari ketiga kandidat tersebut seperti yang tertampil pada Gambar 2.



Gambar 2. Hirarki Algoritme GWO

2. Pengepungan Mangsa
   Seperti yang telah disebutkan sebelumnya, serigala abu-abu mengelilingi mangsanya selama melakukan perburuan. Untuk memodelkan perilaku pengepungan ini secara matematis, maka diusulkan pemodelan persamaan 1 dan 2 berikut:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \quad (1)$$

$$\vec{X}(t+1) = \left|\vec{X}_p(t) - \vec{A} \cdot \vec{D}\right| \quad (2)$$

   Variabel *t* menunjukkan iterasi saat ini. $\vec{A}$ dan $\vec{C}$ merupakan koefisien vektor. $\vec{X}_p$ merupakan vektor posisi dari target/mangsa. $\vec{X}$ merupakan vector posisi dari serigala abu-abu. Vektor $\vec{A}$ dan $\vec{C}$ dapat dihitung dengan menggunakan persamaan 3 dan 4 berikut:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \quad (3)$$

$$\vec{C} = 2 \cdot \vec{r}_2 \quad (4)$$

   Selama proses iterasi, komponen $\vec{a}$ akan menurun secara linier dari 2 ke 0. Komponen $\vec{r}_1$ dan $\vec{r}_2$ adalah vektor acak dalam rentang [0,1]. Proses pengepungan mangsa ini dilakukan pada area dua dimensi dengan koordinat *X* dan *Y*. Tidak menutup kemungkinan juga bahwa proses pencarian dapat dilakukan secara tiga dimensi atau multi dimensi untuk mempercepat proses pencariannya.

3. Perburuan Mangsa
   Serigala abu-abu memiliki kemampuan untuk mengenali lokasi mangsa dan mengepung mereka. Perburuan biasanya dipandu oleh komponen *alfa*. Komponen lainnya seperti *beta* dan *delta* sesekali berpartisipasi dalam proses perburuan ini. Akan tetapi, dalam ruang pencarian

 ISSN:2302-8149
e-ISSN:2540-9719

yang abstrak, serigala abu-abu tidak mengetahui tentang letak lokasi mangsa yang optimal. Untuk mensimulasikan perilaku berburu serigala abu-abu secara matematis, algoritme ini menganggap bahwa *alfa* merupakan solusi kandidat terbaik, sedangkan beta dan delta memiliki pengetahuan yang lebih baik tentang lokasi potensial mangsa. Dalam konteks ini, posisi mangsa bersifat dinamis. Oleh karena itu, algoritme ini dapat menyimpan tiga solusi terbaik pertama yang telah diperolehnya. Langkah selanjutnya adalah mewajibkan agen pencari lainnya (termasuk omega) untuk memperbarui posisi mereka sesuai dengan posisi agen pencarian terbaik. Analogi tersebut diusulkan dalam persamaan 5 dan 6.

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right| \quad (5)$$

$$\vec{X}_1 = \left|\vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha)\right|, \vec{X}_2 = \left|\vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta)\right|, \vec{X}_3 = \left|\vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta)\right| \quad (6)$$

Persamaan tersebut menunjukkan bagaimana agen pencarian memperbarui posisinya berdasarkan *alfa*, *beta*, dan *delta* dalam ruang pencarian dua dimensi. Dapat diamati bahwa posisi akhir dalam ruang pencarian akan berada di tempat acak dalam lingkaran yang ditentukan oleh posisi *alfa*, *beta*, dan *delta*. Dengan kata lain, *alfa*, *beta*, dan *delta* dapat memperkirakan posisi mangsa. Serigala lainnya dapat memperbarui posisi mereka secara acak di sekitar posisi mangsa.

4. Penyerangan (Eksploitasi)
   Seperti yang telah dijelaskan pada tahapan sebelumnya, serigala abu-abu menyelesaikan perburuan dengan menyerang mangsanya ketika berhenti bergerak. Untuk memodelkan secara matematis proses pendekatan ke mangsa, algoritme ini menurunkan nilai $\vec{a}$. Nilai ini juga akan mempengaruhi fluktuasi penguragan nilai $\vec{A}$. Dengan kata lain, nilai $\vec{A}$ dalam interval [-2*a*,2*a*] akan menurun dari 2 ke 0 selama iterasi. Ketika nilai $\vec{A}$ berada pada rentang [-1,1], posisi agen pencarian berikutnya dapat berada pada posisi mana saja antara posisinya saat ini dan posisi mangsa.Gbr. Dengan menggunakan komponen-komponen pencarian yang telah dibahas sebelumnya, algoritme GWO memungkinkan agen pencariannya memperbarui posisi mereka berdasarkan lokasi *alfa*, *beta*, dan *delta*. Akan tetapi, algoritme GWO juga memiliki kelemahan terhadap stagnasi dalam solusi lokal dengan menggunakan komponen-komponen tersebut. Kondisi ini diperkuat dengan adanya mekanisme melingkar pada GWO yang menunjukkan eksplorasi sampai batas tertentu. Oleh karena itu, GWO membutuhkan lebih banyak komponen-komponen pencarian tambahan untuk menekankan proses eksplorasi menjadi lebih baik.

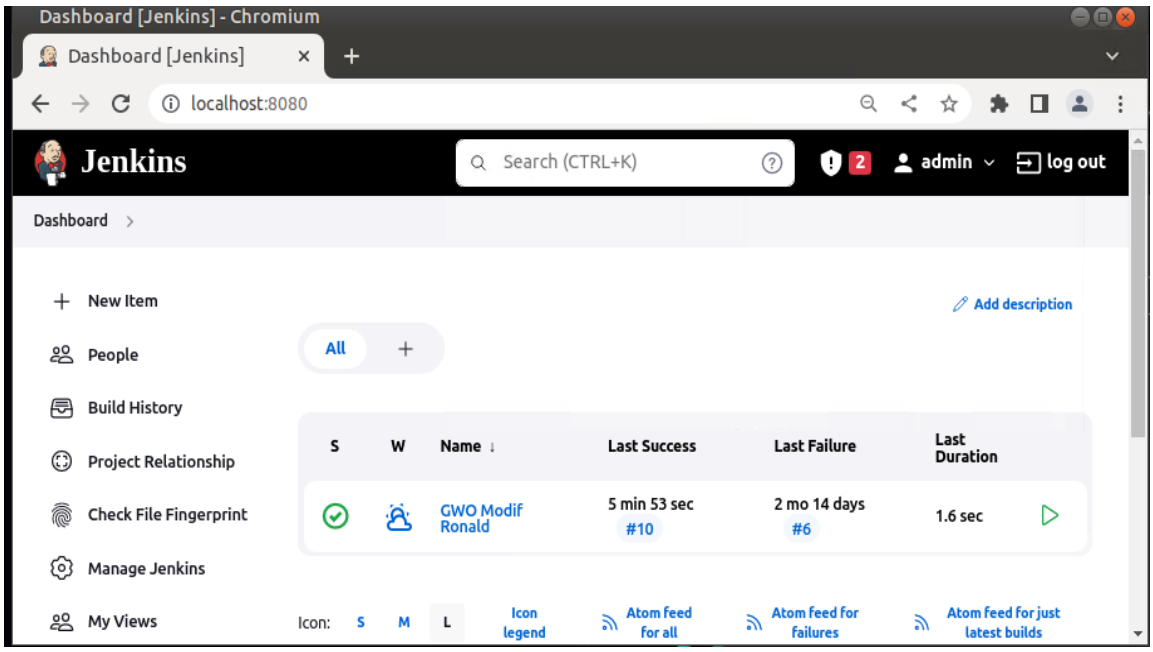
5. Pencarian (Eksplorasi)
   Proses pencarian yang dilakukan oleh serigala abu-abu menggunakan posisi *alfa*, *beta*, dan *delta*. Semua komponen tersebut menyebar untuk mencari mangsa dan melakukan pengepungan dengan berkumpul untuk menyerang mangsa. Untuk memodelkan secara matematis, algoritme ini menggunakan $\vec{A}$ dengan nilai acak lebih besar dari 1 atau kurang dari -1 untuk mewajibkan agen pencari memiliki keleluasaan bergerak dari mangsanya. Kondisi ini menekankan pada eksplorasi dan memungkinkan algoritme GWO untuk dapat bekerja dalam pencarian secara global. Proses seperti ini akan terus berulang dalam pencarian mangsa terbaik.

   Komponen lain dari GWO yang mendukung eksplorasi adalah $\vec{C}$ yang berisi nilai acak dalam rentang [0,2]. Komponen ini memberikan bobot acak kepada mangsa yang dijadikan sebagai target utama, dimana dapat dipergunakan untuk memprioritaskan (C>1) atau mengurangi prioritasnya (C<1) secara stokastik. Ini membantu GWO untuk menunjukkan perilaku yang lebih acak selama pengoptimalan, mendukung eksplorasi, dan penghindaran nilai optimal local karena yang diprioritaskan adalah optimal global.

 ISSN:2302-8149
e-ISSN:2540-9719

Komponen $\vec{C}$ tidak menurun secara linier seperti yang dialamai oleh komponen *A*. Komponen $\vec{C}$ ini diharuskan untuk memberikan nilai acak setiap saat dalam rangka untuk menekankan eksplorasi tidak hanya selama iterasi awal tetapi juga iterasi akhir. Komponen ini sangat membantu jika terjadi stagnasi optimal lokak, terutama pada iterasi akhir. Nilai dari komponen tersebut juga dapat dianggap sebagai hambatan untuk mendekati mangsa yang ada di alam. Secara umum, rintangan alami muncul di jalur berburu serigala dan sebenarnya dipergunakan untuk mencegah mereka mendekati mangsa dengan cepat dan nyaman. Proses inilah yang dilakukan oleh komponen $\vec{C}$. Sebenarnya semua ini tergantung pada posisi serigala, komponen $\vec{C}$ dapat secara acak memberi mangsa bobot dan membuatnya menjadi lebih sulit dan lebih jauh untuk dapat diterkam oleh serigala, atau sebaliknya.
Singkatnya dalam algoritme GWO, proses pencarian mangsa dimulai dengan membuat populasi secara acak dari serigala abu-abu (kandidat solusi). Selama proses iterasi, serigala *alfa*, *beta*, dan *delta* akan memperkirakan kemungkinan posisi mangsa. Setiap solusi kandidat akan memperbarui jaraknya dari mangsanya dan membuatnya menjadi lebih dekat. Komponen-komponen tersebut diturunkan dari 2 ke 0 untuk memprioritaskan proses eksplorasi dan eksploitasi. Solusi kandidat yang muncul akan cenderung menyimpang dari mangsa saat nilai $\vec{A}$>1 dan menyatu ke arah mangsa saat $\vec{A}$<1. Akhir prosesnya, algoritme GWO diakhiri dengan nilai akhir yang terbaik yang mendekat ke mangsa.

Proses selanjutnya adaah mengintegrasikan algoritme GWO ke dalam sistem CI/CD dengan menggunakan Jenkins seperti terlihat pada Gambar 3. Proses integrasi dilakukan secara tersetruktur dengan menggunakan bahasa pemrograman *python* yang dapat dieksekusi secara otomatis oleh Jenkins ini. Waktu eksekusi proses ini akan dibandingkan dengan waktu eksekusi program dari algoritme metaheuristik lainnya untuk melihat performa yang terbaik.



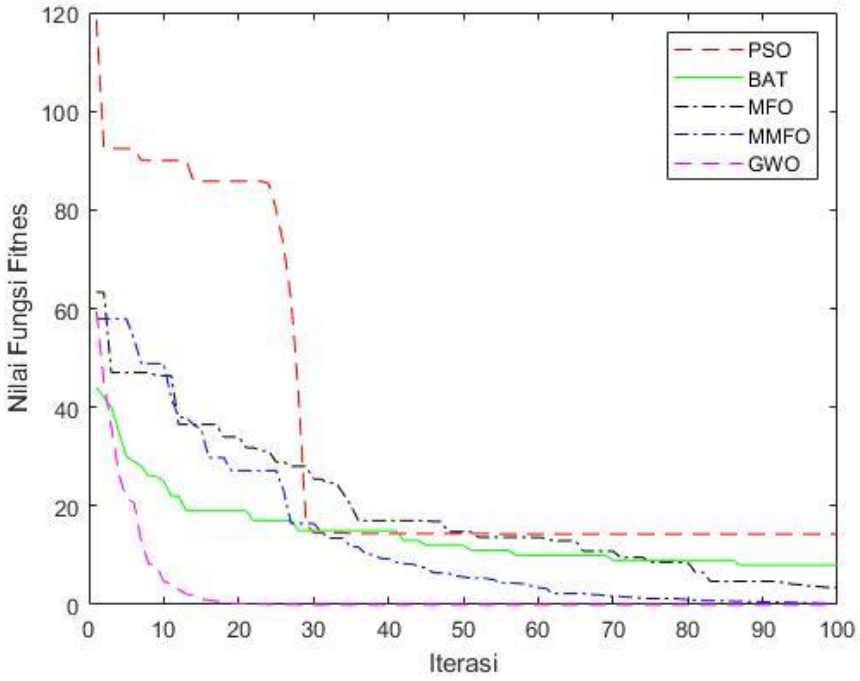
Gambar 3. Tampilan Jenkins

Program yang diintegrasikan ini digunakan untuk sistem otomatisasi jaringan. Seorang admin jaringan diharapkan tidak perlu untuk banyak melakukan pemrograman yang berulang-ulang karena dapat dikerjakan secara otomatis dengan menggunakan Jenkins dan algoritme katalisnya, yang dalam penelitian ini menggunakan algoritme GWO.

ISSN:2302-8149
e-ISSN:2540-9719

4 Hasil dan Pembahasan

Performa Algoritme GWO yang terlihat pada Gambar 4 terbukti yang paling cepat dalam penemuan nilai fungsi tujuannya (fitnes). Ada kenaikan kecepatan penemuan nilai fungsi tujuan terbaik yang didapatkan oleh algoritme GWO. Hal ini disebabkan oleh banyaknya dimensi pencarian yang dilakukan oleh GWO. Komponen-komponen pencarian seperti *alfa*, *beta*, dan *delta* yang bergerak untuk mencari mangsa dari koordinat yang berbeda-beda membuat pengepungan nilai tujuan menjadi lebih terarah. Semakin banyak dimensi yang digunakan maka akan semakin cepat juga dalam pencarian nilai terbaiknya. Nilai fungsi tujuan ini dipergunakan sebagai penentu kecepatan penyebaran aplikasi pada CI/CD. Kecepatan pengujian ini akan dibandingkan dengan metode-metode lainnya yang akan dipergunakan sebagai *benchmark* hasil akhir.

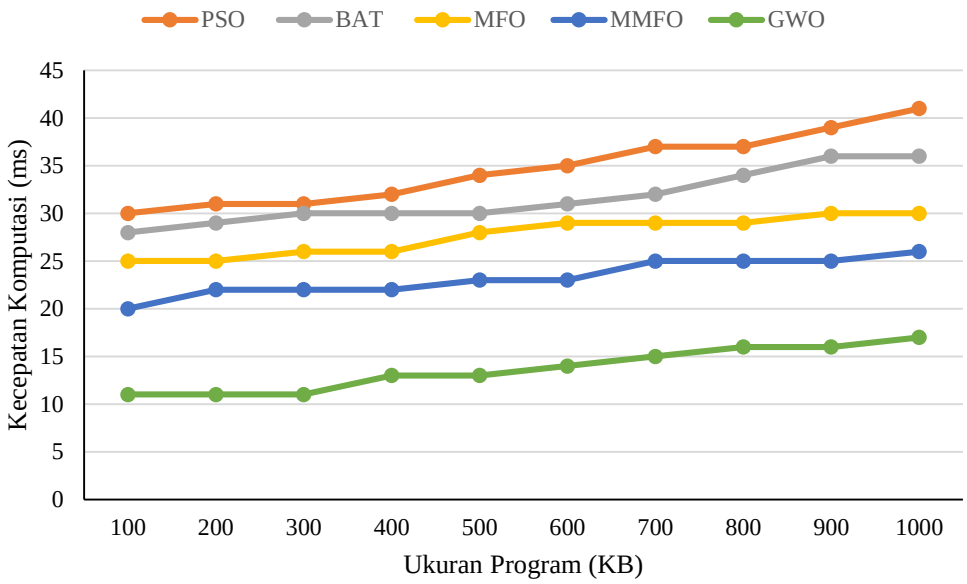


Gambar 4. Perbandingan Waktu Konvergensi

Sebelum dilakukan penentuan algoritme terbaik yang akan diimplementasikan pada CI/CD maka dilakukan terlebih dahulu berkaitan dengan perbandingan performa algoritme heuristik yang ada. Algoritme-algoritme heuristik yang dipergunakan dalam pengujian ini antara lain *particle swarm optimization* (PSO), *Bat*, *Moth Flame Optimization* (MFO), *Modified Moth Flame Optimization* (MMFO), dan *Grey Wolf Optimizer* (GWO). Pada gambar 4 terlihat bahwa GWO jauh lebih unggul dalam kasus penggunaan di CI/CD. Waktu konvergensinya unggul 74,1% dibandingkan dengan algoritme metaheuristik lainnya.

Implementasi algoritme GWO dalam CI/CD terbukti memiliki peningkatan kecepatan sebesar 41,2% dibandingkan dengan rata-rata waktu yang dimiliki oleh algoritme lainnya seperti yang ditampilkan pada Gambar 5. Peningkatan ini disebabkan oleh kecepatan waktu konvergensi yang unggul dari GWO dibandingkan dengan algoritme lainnya. Hal ini berdampak langsung dalam kecepatan komputasi dalam penyebaran aplikasi pada Jenkins. Secara umum, proses pengujian yang dilakukan pada penelitian ini menggunakan beberapa ukuran program yang bertahap mulai dari 100 KB sampai 1000 KB.

ISSN:2302-8149
e-ISSN:2540-9719



Gambar 5. Perbandingan Kecepatan Komputasi

5 Kesimpulan

Algoritme heuristik memiliki kemampuan yang baik dalam melakukan komputasi yang sangat kompleks dengan cepat. Kemampuan ini sering dipergunakan untuk melakukan optimasi pada berbagai bidang. Sistem optimasi pada penelitian ini menekankan pada penyebaran aplikasi melalui CI/CD berbasis algoritme GWO. Algoritme ini terbukti mampu meningkatkan kecepatan penyebaran aplikasi sebesar 41,2%. Peningkatan kecepatan ini disebabkan oleh kemampuan algoritme GWO dalam mempercepat proses penemuan nilai tujuan. Penelitian ini membuka peluang pengembangan metode pada masa depan tentang otomatisasi di pemrograman jaringan. Tidak menutup kemungkinan juga untuk dapat dikembangkan lebih lanjut pada bidang lainnya. Pengembangan yang sangat terbuka pada optimasi penggunaan CI/CD dengan berbagai algoritme metaheuristik lainnya. Optimasi ini memiliki dampak yang sangat besar terutama dalam hal peningkatan kecepatan dan akurasi penyebaran aplikasi.

Ucapan Terima Kasih

Penelitian ini dibiayai oleh Hibah Penelitian Dana Masyarakat Sekolah Vokasi Universitas Gadjah Mada Tahun 2022.

Referensi

[1] O. Arouk and N. Nikaein, "5G Cloud-Native: Network Management & Automation," *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2020: Management in the Age of Softwarization and Artificial Intelligence, NOMS 2020*, Apr. 2020, doi: 10.1109/NOMS47738.2020.9110392.

[2] X. Jin and F. Servant, "CIBench: A Dataset and Collection of Techniques for Build and Test Selection and Prioritization in Continuous Integration," *Proceedings - International Conference on Software Engineering*, pp. 166–167, May 2021, doi: 10.1109/ICSE-COMPANION52605.2021.00070.

[3] A. Poth, M. Werner, and X. Lei, "How to Deliver Faster with CI/CD Integrated Testing Services?," *Communications in Computer and Information Science*, vol. 896, pp. 401–409, 2018, doi: 10.1007/978-3-319-97925-0_33/COVER.

[4] A. Alnafessah, A. U. Gias, R. Wang, L. Zhu, G. Casale, and A. Filieri, "Quality-Aware DevOps Research: Where Do We Stand?," *IEEE Access*, vol. 9, pp. 44476–44489, 2021, doi: 10.1109/ACCESS.2021.3064867.

[5] M. H. Nadimi-Shahraki, S. Taghian, S. Mirjalili, H. Zamani, and A. Bahreininejad, "GGWO: Gaze Cues Learning-Based Grey Wolf Optimizer and Its Applications for Solving Engineering

Problems," *Journal of Computational Science*, vol. 61, p. 101636, May 2022, doi: 10.1016/j.jocs.2022.101636.

[6] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Nov. 1995, vol. 4, no. 6, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.

[7] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," in *Studies in Computational Intelligence*, vol. 284, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 65–74. doi: 10.1007/978-3-642-12538-6_6.

[8] S. Mirjalili, "Moth-flame optimization algorithm: A novel Nature-inspired Heuristic Paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015, doi: 10.1016/j.knosys.2015.07.006.

[9] R. Adrian, S. Sulistyo, I. W. Mustika, and S. Alam, "A Controllable RSU and Vampire Moth to Support the Cluster Stability in VANET," *International Journal of Computer Networks & Communications*, vol. 13, no. 3, pp. 79–95, May 2021, doi: 10.5121/ijcnc.2021.13305.

[10] M. R. Pratama and D. Sulistiyo Kusumo, "Implementation of Continuous Integration and Continuous Delivery (CI/CD) on Automatic Performance Testing," *2021 9th International Conference on Information and Communication Technology, ICoICT 2021*, pp. 230–235, Aug. 2021, doi: 10.1109/ICOICT52021.2021.9527496.

[11] Y. Wang, M. v. Mäntylä, Z. Liu, and J. Markkula, "Test Automation Maturity Improves Product Quality—Quantitative Study of Open Source Projects using Continuous Integration," *Journal of Systems and Software*, vol. 188, p. 111259, Jun. 2022, doi: 10.1016/j.jss.2022.111259.

[12] W. Zhang, X. Fei, and B. Wang, "Research on Specific Harmonic Elimination Technology Based on Improved Gravity Search Algorithm," pp. 1499–1504, Aug. 2022, doi: 10.1109/ICMA54519.2022.9855983.

[13] W. W. Koczkodaj *et al.*, "Combating Harmful Internet Use with Peer Assessment and Differential Evolution," *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pp. 1–6, Jul. 2022, doi: 10.1109/ICECET55527.2022.9873437.

[14] D. Sobania, D. Schweim, and F. Rothlauf, "A Comprehensive Survey on Program Synthesis with Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, Mar. 2022, doi: 10.1109/TEVC.2022.3162324.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[16] J. Adhikary and S. Acharyya, "Randomized Balanced Grey Wolf Optimizer (RBGWO) for Solving Real Life Optimization Problems," *Applied Soft Computing*, vol. 117, p. 108429, Mar. 2022, doi: 10.1016/j.asoc.2022.108429.