Jurnal Ilmiah Ilmu Komputer Vol. 6, No. 2, September 2020 (P) ISSN 2442-451X
Fakultas Ilmu Komputer (O) ISSN 2503-3832
Universitas AL Asyariah Mandar

Desain Antarmuka Pada *Vehicle Routing Problem* Untuk Manajemen Armada Multi-Drone

Setyawan Ajie Sukarno[1], Yuliadi Erdani[2]

[1,2]Teknik Otomasi Manufaktur dan Mekatronika, Politeknik Manufaktur Bandung, Bandung, Indonesia
[1]setyawan_as@polman-bandung.ac.id, [2] yul_erdani@polman-bandung.ac.id

ABSTRAK

Artikel ini membahas desain antarmuka pada *vehicle routing problem* (VRP) 3-dimensi untuk armada multi-drone. Armada ini melakukan perjalanan untuk mengunjungi serangkaian titik dengan memperhatikan batasan tertentu. Karena VRP diklasifikasikan sebagai masalah optimasi NP-hard, algoritma aproksimasi seperti Algoritma Genetika diterapkan untuk menemukan solusi terbaik untuk masalah optimisasi kombinatorial ini. Dalam merancang GUI ini, kami menggunakan Netlogo sebagai alat untuk membangun antarmuka, dan juga untuk eksperimen dan simulasi. Hasil penelitian ini menunjukkan bahwa dengan menggunakan Netlogo, kita dapat mendesain antarmuka untuk mensimulasikan algoritma aproksimasi dalam penyelesaian permasalahan optimasi kombinatorial, yang mudah dioperasikan oleh pengguna.

Kata Kunci : Antarmuka, Netlogo, VRP, Drone, Algoritma Genetika.

ABSTRACT

In this paper, we discuss the designing of a GUI for a 3-Dimensional Vehicle Routing Problem for a fleet of drones. The fleet performs routes to visit a set of points while respecting constraints. Since VRP is classified as an NP-hard optimization problem, an approximation algorithm i.e. Genetic Algorithm is applied to find a best solution to this combinatorial optimization problem. In designing this GUI, we implement Netlogo as the tool to build the interface, and also run the experiment and simulation as well. This research indicates that by using Netlogo, we can design an interface to simulate an approximation algorithm in solving combinatorial optimization problems, which is easy to operate by users.

Keywords: GUI, Netlogo, VRP, Drone, Genetic Algorithm.

(P) ISSN 2442-451X
(O) ISSN 2503-3832

1. PENDAHULUAN

Di era ini, drone memainkan peran penting dalam kehidupan kita, dan diprediksi akan semakin besar dan lebih penting di masa depan. Karena fleksibilitas dan harganya yang murah, penerapan drone menjadi lebih luas mencakup monitoring, pertanian, pertahanan, teknik sipil, logistik, SAR, studi ilmiah, dan lain lain.

Baru-baru ini, tren baru sedang bergerak ke arah mengelola armada multi-drone yang berkolaborasi untuk mencapai misi yang diberikan. Hal ini membuka banyak peluang penelitian untuk mengembangkan platform pada manajemen armada multi-drone. *Vehicle routing problem* (VRP) adalah studi yang tepat untuk menjawab tantangan ini, dalam menemukan jalur terbaik untuk setiap drone, dengan beberapa batasan yang harus dipertimbangkan.

Ada banyak metode untuk menyelesaikan VRP, dan dapat dikategorikan menjadi dua kelompok yaitu: metode eksak dan aproksimasi. Namun, berhubung VRP diklasifikasikan sebagai masalah optimasi *NP-hard*, metode aproksimasi menjadi lumrah untuk diterapkan. Salah satu metode aproksimasi yang paling banyak digunakan dalam berbagai jenis masalah transportasi seperti VRP adalah algoritma genetika (AG).

Artikel ini akan membahas desain antarmuka pada proses eksekusi algoritma genetika yang digunakan untuk menyelesaikan *vehicle routing problem* untuk manajemen armada multi-drone. Antarmuka ini akan didesain dengan menggunakan bahasa pemrograman Netlogo.

2. TINJAUAN PUSTAKA

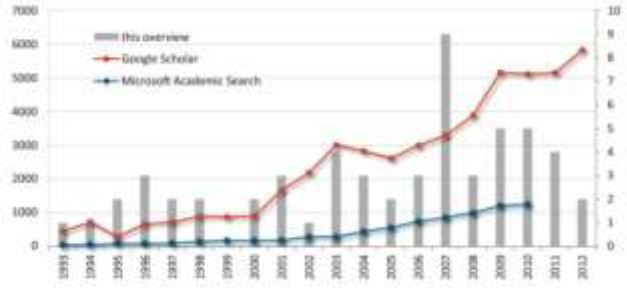
2.1. Vehicle Routing Problem

VRP adalah metode untuk merancang rute yang optimal dan paling murah untuk armada kendaraan dalam melayani satu set pelanggan, dengan beberapa batasan yang harus diperhatikan seperti misalnya waktu tempuh dan kapasitas armada [1], [2]. Rute dirancang sedemikian rupa sehingga setiap pelanggan dikunjungi hanya satu kali oleh satu kendaraan saja, semua kendaraan harus mulai dan selesai pada titik yang sama yang dikenal sebagai *depot*, dan memenuhi beberapa batasan yang diterapkan. Dalam beberapa dekade, banyak peneliti yang telah secara ekstensif mempelajari bidang studi ini, seperti yang disurvei pada [1].

Dantzig dan Ramser [3] adalah orang pertama yang mengatasi masalah perancangan rute kendaraan dalam literatur optimisasi, dengan penulisan makalah mereka dalam jurnal Management Science mengenai perancangan rute armada truk pengiriman bensin antara terminal curah dan sejumlah stasiun pemasok bahan bakar. Masalah yang dirumuskan dalam [3] bernama *truck dispatching problem*, tetapi beberapa literatur di VRP mengutip makalah mereka sebagai contoh pertama dari masalah ini.

2.2. Algoritma Genetika

Menurut [1], ada banyak metode untuk menyelesaikan VRP, dan dapat dikategorikan menjadi dua kelompok yaitu: metode eksak dan aproksimasi. Namun, berhubung VRP diklasifikasikan sebagai masalah optimasi *NP-hard*, metode aproksimasi menjadi lumrah untuk diterapkan. Salah satu metode aproksimasi yang paling banyak digunakan dalam berbagai jenis masalah transportasi seperti VRP adalah algoritma genetika (AG) [4]. Topik AG dalam menyelesaikan VRP telah diteliti secara masif oleh banyak makalah dalam beberapa dekade, seperti yang ditunjukkan pada Gambar 1.



Gambar 1. Jumlah karya ilmiah per tahun tentang penggunaan AG untuk menyelesaikan VRP [4].

Algoritma genetika adalah algoritma aproksimasi yang dirancang serta terinspirasi dari ide-ide evolusi seleksi genetik dan alam [5]. Algoritma ini bergantung pada prosedur bio-evolusi seperti persilangan (*crossover*), mutasi dan seleksi, untuk menghasilkan generasi yang lebih baik daripada yang sebelumnya.

Algoritma ini banyak digunakan pada beberapa masalah optimisasi kombinatorial seperti TSP (*traveling salesman problem*), VRP, dan beberapa masalah penjadwalan. Dalam algoritma genetika, langkah awal yang pertama-tama dilakukan adalah menghasilkan beberapa populasi secara acak, yang akan ditetapkan sebagai solusi awal.

Selanjutnya, beberapa prosedur evolusi dijalankan, seperti pemilihan kromosom, persilangan, dan mutasi, untuk mendapatkan generasi yang lebih baik dalam rangka menemukan solusi terbaik dalam menjawab masalah optimisasi yang dihadapi. Untuk menilai apakah generasi yang dihasilkan lebih baik dari generasi sebelumnya, diperlukan sebuah alat uji yang disebut fungsi *fitness*. Jika nilai *fitness* dari generasi sekarang lebih baik dari generasi sebelumnya, maka nilai *fitness* generasi sekarang akan menggantikan nilai *fitness* generasi sebelumnya sebagai nilai *fitness* terbaik.

Prosedur evolusi dijalankan hingga jumlah iterasi telah memenuhi batasan, dan nilai *fitness* terakhir yang disimpan sebagai nilai *fitness* terbaik akan ditetapkan sebagai solusi dari VRP, dan untaian kromosom menjadi urutan rute perjalanan armada drone

2.3. Netlogo

NetLogo adalah generasi baru dari bahasa pemodelan sistem multi-agen [6]. Netlogo ini dirancang oleh Uri Wilensky pada tahun 1999, dan sejak itu, telah

(P) ISSN 2442-451X
(O) ISSN 2503-3832

dikembangkan terus menerus di Northwestern's Center for Connected Learning dan Computer-Based Modeling (CCL).

Netlogo adalah alat yang dapat diprogram untuk mensimulasikan fenomena sosial dan alami, dan juga untuk memodelkan sistem yang kompleks. Pengguna dapat memberikan instruksi kepada ratusan atau ribuan agen untuk beroperasi secara independen dan bersamaan. Hal ini memungkinkan untuk mengeksplorasi hubungan antara perilaku tingkat mikro individu dan pola tingkat makro yang muncul dari interaksi mereka [6].

NetLogo memungkinkan pengguna untuk membuka simulasi dan "bermain" dengan mereka, menjelajahi perilaku mereka dalam berbagai kondisi. NetLogo juga merupakan bahasa pemodelan yang cukup sederhana, yang memungkinkan siswa dan peneliti untuk membuat model mereka sendiri, bahkan jika mereka bukan programmer profesional [6]. Hal tersebut membuat Netlogo sangat cocok diaplikasikan dalam pendidikan dan penelitian, dan sekarang digunakan oleh banyak peneliti di seluruh dunia [7].

3. METODE PENELITIAN

Pada penelitian ini, metode penelitian yang dilakukan meliputi studi pustaka, konsultasi dan perancangan jendela antarmuka.

Tahap pertama adalah memulai penelitian dengan studi pustaka yaitu pencarian program yang cocok untuk proyek ini. Proses ini dilakukan dengan cara mengumpulkan literatur yang ada kaitannya dengan penelitian. Dalam proses ini, kami menemukan sebuah bahasa pemrograman yang bernama Netlogo, yang merupakan generasi baru dari bahasa pemodelan sistem multi-agen [6].

Tahap kedua adalah melakukan konsultasi dengan pihak-pihak yang sebelumnya membuat penelitian yang serupa, atau pernah menggunakan Netlogo dalam proyek yang serupa, sehingga mendapatkan informasi yang dapat mendukung penelitian.

Tahap ketiga adalah mendesain jendela antarmuka dengan menggunakan Netlogo. Jendela antarmuka didesain sebagai panel kontrol untuk memantau progres algoritma, yang terdiri dari panel untuk mengontrol parameter eksperimental dan simulasi, tombol untuk mengontrol proses, dan monitor untuk mengamati proses serta solusi rute.

Gambar 2 menunjukkan antarmuka standar saat kita memulai Netlogo. Di sana, kita dapat mendesain jendela antarmuka kita sendiri menggunakan beberapa jenis item antarmuka seperti yang ditunjukkan pada Gambar 3 dibawah ini.

Gambar 2. Antarmuka standar saat memulai Netlogo.

.

Gambar 3. Jenis parameter input yang tersedia di Netlogo.

Dalam desain antarmuka di proyek ini, jenis-jenis parameter input yang dipilih yaitu: saklar geser, tombol, dan monitor.

Saklar geser panel kontrol parameter didesain untuk memasukkan parameter inisial penting sebelum algoritma genetika berjalan, seperti: jumlah target yang perlu dikunjungi, jumlah drone yang tersedia, jumlah iterasi yang dijalankan, ukuran turnamen pemilihan kromosom, jumlah populasi, persentase probabilitas terjadinya mutasi dan batasan waktu penerbangan.

Input jenis tombol didesain untuk mengendalikan urutan algoritma, seperti: memuat lokasi yang harus disinggahi oleh drone, menghasilkan populasi awal, menjalankan algoritma, memuat rute, memeriksa persimpangan dan menjalankan simulasi.

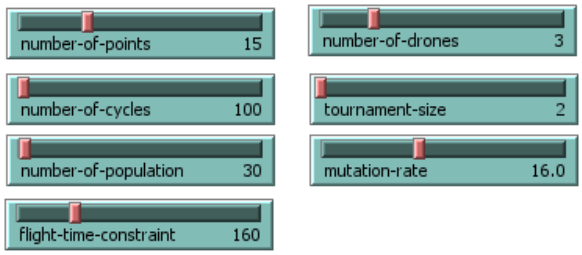
Monitor dan *plotter* didesain untuk mengamati kemajuan pada proses berjalannya algoritma.

3.1. Panel Kendali Parameter

Untuk mengatur parameter sebelum algoritma dijalankan, digunakan input tipe saklar geser seperti yang ditunjukkan pada Gambar 4. Parameter *number-of-points* adalah untuk menentukan jumlah target yang akan dikunjungi dalam misi. Parameter *number-of-drones* adalah untuk memastikan jumlah drone yang tersedia untuk melakukan misi. Parameter *number-of-cycles* adalah untuk menentukan jumlah iterasi yang akan dijalankan

(P) ISSN 2442-451X
(O) ISSN 2503-3832

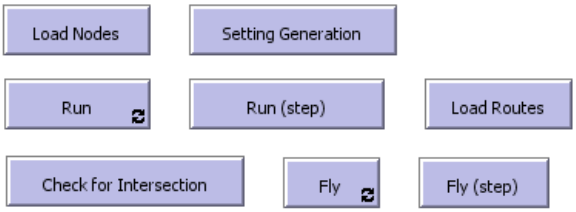
oleh algoritma. *Tournament-size* adalah parameter untuk menentukan jumlah kromosom pada generasi sebelumnya yang akan dipertandingkan dalam memilih orang tua untuk generasi baru. Parameter *number-of-population* adalah untuk menentukan jumlah populasi dalam algoritma genetika. Parameter *mutation-rate* adalah untuk menentukan nilai persentasi probabilitas terjadinya prosedur mutasi. Dan parameter *flight-time-constraint* adalah untuk mengatur waktu maksimum bagi setiap drone untuk menyelesaikan rute mereka.



Gambar 4. Saklar geser untuk menentukan parameter awal pada algoritma genetika.

3.2. Tombol Kendali Proses

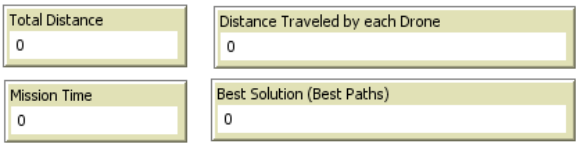
Untuk mengontrol urutan algoritma, digunakan input jenis tombol seperti yang ditunjukkan pada gambar.5. Tombol "*Load Nodes*" adalah untuk menginstal titik-titik yang harus dituju oleh drone pada bidang 3-dimensi yang terdapat pada Netlogo. Tombol "*Setting Generation*" adalah untuk menghasilkan populasi awal yang merupakan prosedur pembuka dalam algoritma genetika. Tombol "*Run*" adalah untuk menjalankan semua prosedur dalam algoritma. Tombol "*Load Routes*" adalah untuk mengalokasikan titik-titk dan misi untuk setiap drone. Tombol "*Check for Intersection*" dibuat untuk memeriksa apakah ada persimpangan yang berpotensi tabrakan antar drone atau tidak. Dan tombol "*Fly*" adalah untuk memulai dan menjalankan simulasi penerbangan.



Gambar 5. Tombol pengendalian proses pada AG.

3.3. Monitor Observasi dan Plotter Nilai Fitness

Untuk mengamati proses algoritma, digunakan input jenis monitor dan *plotter* seperti yang ditunjukkan pada Gambar 6 dan Gambar 7. Monitor dalam Gambar 6 adalah untuk menunjukkan kemajuan dalam mencari solusi terbaik di setiap iterasi, hingga iterasi akhir, sebagaimana dinyatakan dalam parameter *number-of-cycles*.



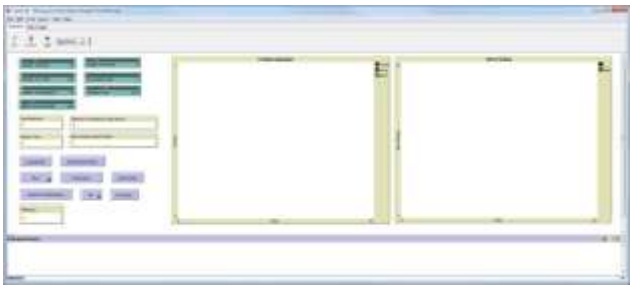
Gambar 6. Monitor observasi.

Plotter dalam Gambar 7 adalah untuk menampilkan kemajuan dalam mencari nilai *fitness* terbaik di setiap iterasi. Nilai *fitness* adalah persamaan paling penting dalam setiap masalah optimisasi kombinatorial, atau kadang-kadang dilambangkan sebagai fungsi objektif. Dalam kebanyakan kasus VRP, fungsi tujuannya adalah untuk meminimalkan jumlah ongkos implementasi.

Plotter di sisi kiri menampilkan nilai *fitness* di setiap iterasi, dan plotter di sisi kanan menampilkan nilai *fitness* terbaik, yang merupakan solusi terbaik yang diperoleh dalam kemajuan saat ini. Proses merencanakan akan dihentikan ketika iterasi telah mencapai nilai dalam parameter *number-of-cycles*.

Gambar 7. Layar *plotter* untuk memonitor fungsi *fitness* pada algoritma.

Setelah semua parameter input sudah ditentukan, jendela antarmuka dapat dirancang seperti yang ditunjukkan pada Gambar 8 di bawah ini.



Gambar 8. Rancangan jendela antarmuka.

4. HASIL PENELITIAN

Di bagian ini, dijelaskan detail tentang bagaimana GUI bekerja dan juga mendukung percobaan dan simulasi.

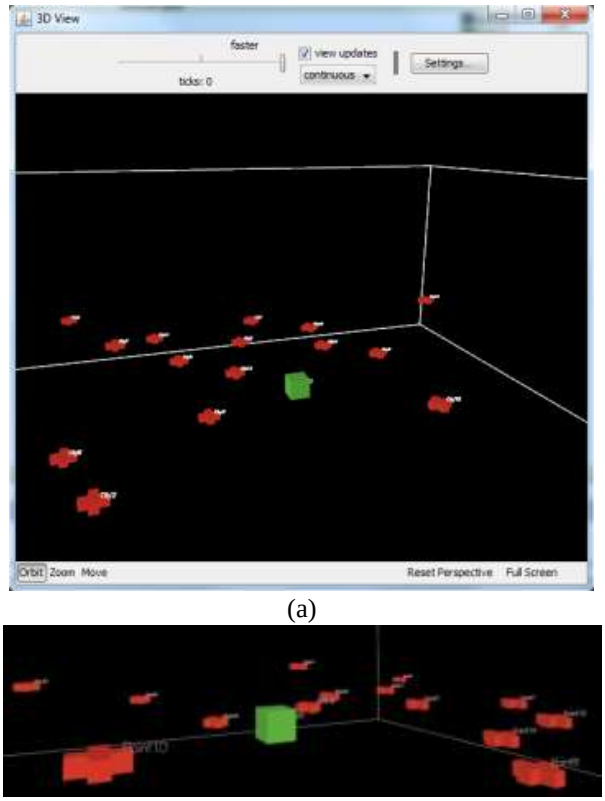
4.1. *Loading Nodes* dan *Setting Generation*

Tombol "*Load Nodes"* adalah tombol untuk memulai proses dan meletakkan sekelompok target di bidang 3-dimensi. Kemudian, Netlogo akan menunjukkan posisi target-target di bidang 3-dimensi seperti yang ditunjuk-kan

(P) ISSN 2442-451X
(O) ISSN 2503-3832

pada Gambar 9. Jumlah target ditentukan oleh saklar geser *number-of-points* pada panel kendali parameter.

Setelah menentukan posisi target-target pada bidang 3-dimensi, tahap selanjutnya adalah menghasilkan beberapa jumlah populasi, secara acak, yang akan menjadi generasi awal atau solusi pertama yang diketahui. Kemudian populasi pertama tadi dievaluasi dengan menghitung nilai *fitness* dari solusi tersebut dengan menerapkan fungsi *fitness*. Produksi populasi awal ini dilakukan dengan menekan tombol "*Setting Generation"*.



(a)

(b)

Gambar 9. Tampilan 3-dimensi dari lokasi target-target armada drone. (a) Dengan tampilan jendela. (b) Dengan tampilan yang diperbesar.

4.2. Run

Fungsi *fitness* yang disebutkan pada bagian sebelumnya, merupakan aspek penting dalam masalah-masalah optimasi, karena fungsi *fitness* merupakan satu-satunya faktor untuk memutuskan solusi mana yang dipilih sebagai solusi terbaik. Jika pada sebuah iterasi dihasilkan populasi dengan nilai *fitness* yang lebih baik daripada populasi sebelumnya, maka solusi terbaru akan ditasbihkan sebagai solusi terbaik sementara, seperti yang terlihat pada diagram alir di Gambar 10.

Perancangan fungsi *fitness* akan tergantung pada kendala-kendala yang diterapkan. Dalam proyek ini, yang menjadi fungsi *fitness* adalah total dari ongkos implementasi, yang merupakan fungsi dari waktu dan jarak. Semakin kecil waktu tempuh dan jarak perjalanan, akan menjadikan sebuah solusi dikatakan lebih baik dari solusi yang didapat sebelumnya.

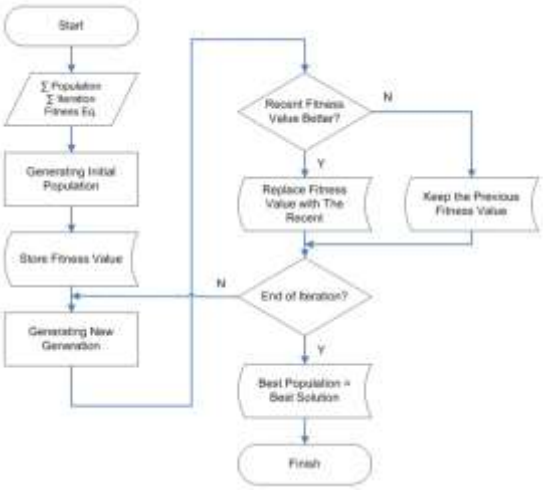
Fungsi *fitness* yang diterapkan dalam proyek ini dirumuskan seperti pada formula (1) berikut,

$$F = min \sum_{v=1}^{V} \sum_{i=0}^{N} \sum_{j=1}^{N+1} T_{ij} X_{ij}^{v} \quad (1)$$

sama seperti proyek [8], dimana T_{ij} adalah jarak yang ditempuh ketika drone bergerak dari titik-*i* ke titik-*j*. Kemudian, X^{v}_{ij} adalah variabel biner yang didefinisikan untuk setiap busur (i, j), yang bernilai sama dengan 1 jika dan hanya jika sebuah drone-*v* mendatangi titik-*i* dan kemudian berjalan langsung ke titik-*j*, dan bernilai 0 jika tidak. *N* merupakan variabel yang menunjukkan jumlah target yang terdapat dalam bidang 3-dimensi, dan *V* merupakan variabel yang menunjukkan jumlah drone dalam armada.

Pada Gambar 10 dibawah, setelah menghasilkan populasi awal, algoritma akan memproduksi populasi atau generasi yang baru, dengan harapan menemukan generasi dengan nilai *fitness* yang lebih baik dari generasi sebelumnya, sehingga dapat menghasilkan solusi yang lebih baik.

Untuk tujuan tersebut, beberapa proses harus dijalankan seperti seleksi kromosom, *crossover* dan mutasi, sebagai prosedur evolusi untuk menghasilkan generasi baru dengan nilai *fitness* yang lebih baik dalam menjawab masalah optimasi yang dihadapi. Untuk memulai prosedur evolusi, tombol "*Run*" pada panel kendali proses di jendela antarmuka ditekan terlebih dahulu.



Gambar 10. Diagram alir dari algoritma genetik.

Prosedur evolusi dimulai dengan seleksi kromosom, dengan memilih kromosom terbaik dari generasi yang sudah dihasilkan, untuk menjadi "orang tua" bagi generasi yang akan dihasilkan. Diharapkan, dengan kromosom yang terseleksi, generasi yang dihasilkan oleh algoritma

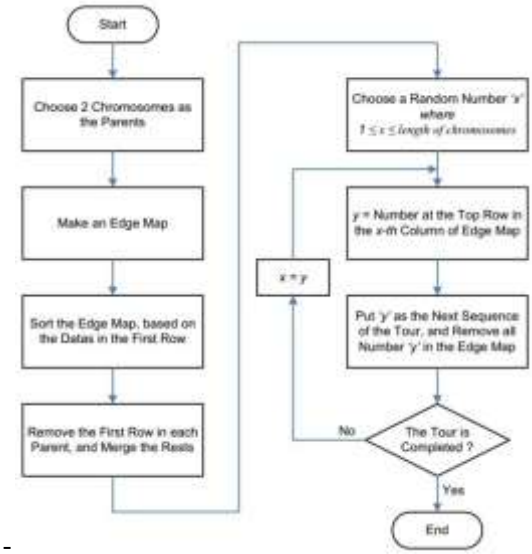
(P) ISSN 2442-451X
(O) ISSN 2503-3832

genetika dapat menjadi generasi yang lebih baik, yang dibuktikan dengan fungsi *fitness* yang diaplikasikan.

Setelah proses seleksi kromosom, prosedur selanjutnya adalah prosedur *crossover*, yang merupakan fase penting dalam algoritma genetika, karena dapat mempengaruhi kinerja algoritma. Umbarkar dan Sheth [9] menyatakan bahwa pemilihan operator *crossover* sangat berdampak pada kinerja algoritma genetika. Dengan kata lain, memilih teknik *crossover* yang tepat adalah penting untuk mendapatkan solusi yang lebih baik.

Umbarkar dan Sheth [9] meninjau beberapa operator *crossover* yang diusulkan dan dicoba oleh berbagai peneliti. Salah satunya, yang disebut *edge recombination crossover* (ERX), adalah teknik yang diterapkan dalam proyek ini. Teknik ini juga digunakan oleh Hileman [10] dalam menyelesaikan kasus *traveling salesman problem* (TSP). Namun, karena VRP lebih rumit dari TSP, beberapa modifikasi dilakukan untuk proyek ini.

Gambar 11 memberikan gambaran mengenai proses berjalannya prosedur *crossover* dengan metode ERX. Namun artikel ini tidak akan membahas secara detail mengenai metoda ERX dalam prosedur *crossover*.



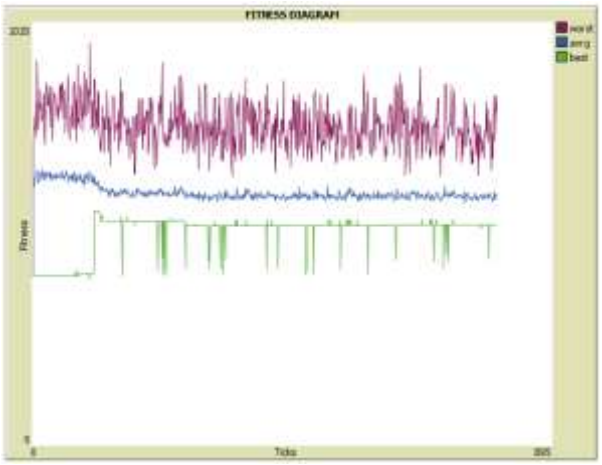
Gambar 11. Diagram alir dari prosedur *crossover* menggunakan metode ERX.

Setelah prosedur *crossover* selesai, proses evolusi belanjut ke prosedur mutasi. Prosedur mutasi dapat menghasilkan kromosom baru yang tidak berasal dari prosedur *crossover*. Dalam VRP, mutasi dihasilkan dari proses pertukaran posisi atau perubahan urutan dari solusi. Penjelasan sederhana tentang prosedur mutasi ditunjukkan pada Gambar 12.

Initial Result from Crossover Procedure												
1	15	13	6	7	8	5	3	14	4	12	10	11	9	2
Mutated Result												
1	15	13	12	7	8	5	3	14	4	6	10	11	9	2

Gambar 12. Contoh prosedur mutasi pada solusi VRP.

Pada saat algoritma genetika dijalankan dengan menekan tombol "*Run*" pada panel kendali proses, *plotter* di jendela antarmuka akan menampilkan kemajuan dari solusi yang dihasilkan pada setiap iterasi. Gambar 13 menunjukkan nilai *fitness* dari generasi yang dihasilkan pada setiap iterasi. Kurva merah menunjukkan nilai *fitness* dari kromosom terburuk dalam satu generasi hasil prosedur evolusi dalam satu iterasi. Sedangkan kurva hijau menunjukkan sebaliknya, yaitu nilai *fitness* dari kromosom terbaik. Dan kurva biru adalah nilai rata-ratanya.



Gambar 13. Kurva nilai *fitness* dari generasi yang dihasilkan pada setiap iterasi.

Jika sebuah iterasi menghasilkan kromosom dengan nilai *fitness* yang lebih baik dari nilai *fitness* yang sebelumnya dianggap terbaik, maka plotter kedua akan memberikan indikasi di monitor pada jendela antarmuka, seperti pada Gambar 14.

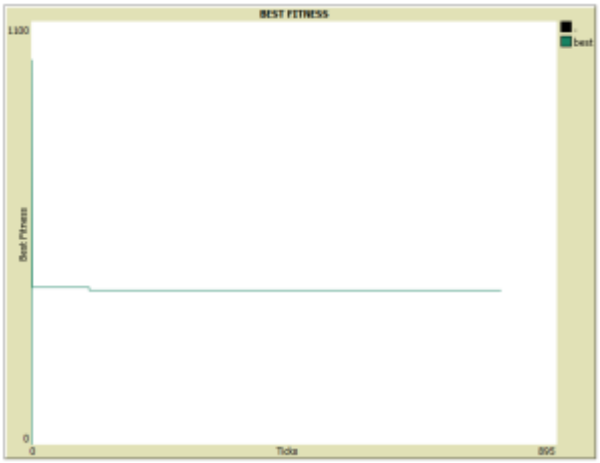
Saat prosedur evolusi selesai dijalani, dengan sejumlah iterasi yang sebelumnya ditentukan oleh parameter *number-of-cycles* di panel kendali parameter seperti pada Gambar 4 di sub-bab sebelumnya, monitor observasi akan menampilkan solusi terbaik dari prosedur evolusi, seperti pada Gambar 15, dan juga informasi lain.

"Total Distance" pada monitor observasi di Gambar 15 adalah data yang menunjukkan total jarak yang dijelajahi oleh semua drones. Angka yang muncul merupakan penjumlahan dari data pada monitor *"Distance Traveled by each Drone"* di Gambar 15. Dan "*Mission Time*" merupakan total waktu yang dibutuhkan oleh semua drones dalam menyelesaikan misi. Sedangkan "*Best Solution*" menunjukkan kromosom terbaik yang dihasilkan prosedur evolusi yang sudah dijalankan.

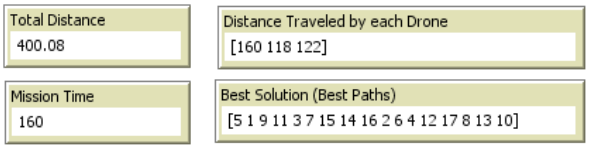
Selain menampilkan data di monitor observasi, pada saat iterasi berakhir, bidang 3-dimensi juga akan menampilkan perencanaan rute yang merupakan hasil dari iterasi yang sudah dilakukan, seperti yang ditunjukkan pada Gambar 16. Gambar 16 (a) merupakan tampilan standar dari bidang 3-dimensi Netlogo, yang merupakan

(P) ISSN 2442-451X
(O) ISSN 2503-3832

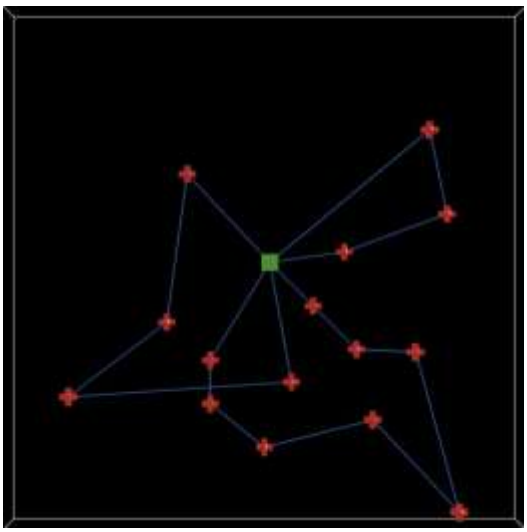
penampakan tegak lurus bidang-z, dan Gambar 16 (b) merupakan tampilan yang sudah diperbesar dan berbeda orientasinya.



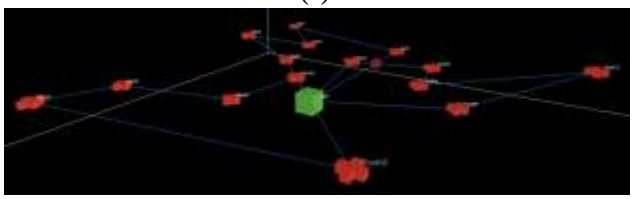
Gambar 14. Kurva monitoring nilai *fitness* terbaik dari proses evolusi.



Gambar 15. Hasil proses evolusi yang direkam oleh monitor observasi.



(a)



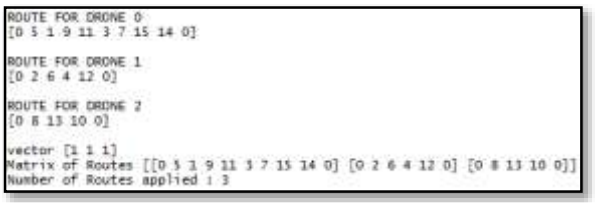
(b)

Gambar 16. Tampilan 3-dimensi dari perencanaan rute.

Dari hasil yang tampak pada Gambar 16 (a), kita dapat menyimpulkan bahwa hasil prosedur evolusi yang sudah dijalankan bukan merupakan hasil yang optimal. Hal ini terlihat jelas karena adanya interseksi pada perencanaan rute yang ditampilkan. Secara sederhana, solusi optimal untuk VRP dengan fungsi *fitness* seperti pada rumus 1 seharusnya tidak terdapat interseksi. Namun hal ini bukanlah sebuah kesalahan, karena dalam metode aproksimasi seperti algoritma genetika, solusi optimal bukanlah tujuan utama [11].

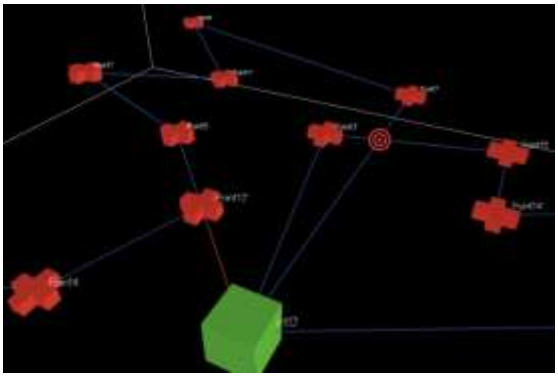
4.3. Perencanaan Rute dan Pengecekan Interseksi

Pada saat prosedur evolusi sudah selesai dijalankan melalui serangkaian iterasi, langkah selanjutnya dalah menekan tombol "*Load Routes*" pada panel kendali proses. Ini merupakan proses untuk mengisi matriks rute pada setiap drone, seperti tampak pada Gambar 17 di bawah ini. Gambar 17 ini merupakan tampilan pada command center di jendela antarmuka. Dapat kita lihat perencanaan rute yang akan dijalani oleh setiap drone.



Gambar 17. Tampilan tangkapan layar pada *command center* di jendela antarmuka setelah tombol "*Load Route*" diklik.

Setelah perencanaan rute sudah diproses, langkah selanjutnya adalah melakukan pengecekan interseksi dengan menekan tombol "*Check for Intersection*". Pada tahap ini, jika terdapat interseksi pada hasil perencanaan rute seperti pada Gambar 16 (a), maka program akan memastikan tidak akan terjadi tabrakan antar drone, meski terdapat interseksi.



Gambar 18. Pengecekan interseksi pada bidang 3-dimensi Netlogo.

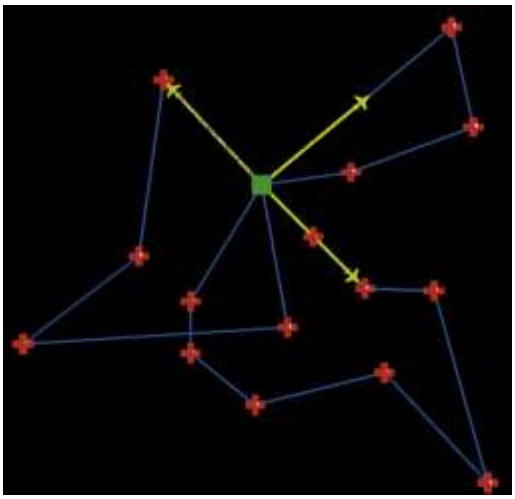
Pada Gambar 18 diatas, titik interseksi ditandai dengan bulatan merah. Jika perhitungan memprediksi akan

(P) ISSN 2442-451X
(O) ISSN 2503-3832

terjadi tabrakan di lokasi itu, maka drone dengan jarak tempuh terkecil harus mengalah dengan berhenti sejenak di lokasi target sebelum titik interseksi tersebut.

4.4. Simulasi Armada

Langkah terakhir dari proyek ini adalah menjalankan simulasi armada drone dalam bidang 3-dimensi di Netlogo. Setelah tombol "*Fly*" ditekan, maka jendela bidang 3-dimensi akan menampilkan simulasi armada drone seperti pada Gambar 19, sedari posisi awal (*depot*) hingga seluruh drone menyelesaikan misinya dalam mengunjungi target-target yang sudah ditetapkan pada tahap sebelumnya, dan kembali ke titik awal.



Gambar 19. Simulasi armada pada bidang 3-dimensi di Netlogo.

Saat seluruh armada drone sudah kembali ketitik awal keberangkatan mereka, maka simulasi ini dinyatakan selesai.

5. KESIMPULAN

Netlogo, yang didesain sebagai bahasa pemodelan sistem multi-agen, sangat sesuai untuk menjadi alat simulasi dan eksperimen dalam masalah optimasi seperti *vehicle routing problem*, *traveling salesman problem*, dan lainnya, terutama yang menggunakan algoritma *meta-heuristic* seperti algoritma genetika, yang diaplikasikan dalam proyek ini.

Meskipun sudah dikaji selama beberapa dekade, algoritma genetika masih membuka peluang untuk peningkatan kinerja dan eksplorasi. Salah satu item terpenting yang perlu dipertimbangkan agar algoritma bekerja seefektif mungkin adalah operator *crossover* dalam prosedur evolusi. Operator ini adalah tulang punggung algoritma genetika. Dalam proyek berikutnya, kami akan melakukan pengujian terhadap operator-operator *crossover*, untuk mencari metode yang paling cocok dalam menyelesaikan kasus VRP, dengan menggunakan Netlogo sebagai alat simulasi dan eksperimen. Kami percaya, dengan menemukan operator *crossover* terbaik, kinerja algoritma genetika akan ditingkatkan, terutama kecepatannya.

Daftar Pustaka

[1] Kumar, S.N. and Panneerselvam, R., 2012, A Survey on the Vehicle Routing Problem and Its Variants, Intelligent Information Management, Vol. 4, pp. 66-74.

[2] Laporte, G., 1992, The Vehicle Routing Problem: An overview of exact and approximate algorithms, European Journal of Operational Research, Vol. 59, pp. 345-358.

[3] Dantzig, G. B. and Ramser, J. H., 1959, The Truck Dispatching Problem, Management Science, INFORMS, Vol. 6, No. 1, pp. 80-91.

[4] Karakatic, K. and Podgorelec, V., 2015, A survey of genetic algorithms for solving multi depot vehicle routing problem, Applied Soft Computing, 27, 519–532.

[5] Goldberg, D.E., 1989, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, New York.

[6] Tisue, S. and Wilensky, U., 2004, Netlogo: A Simple Environment for Modelling Complexity, International Conference on Complex Systems (ICCS), Boston, May 16 - 21.

[7] Netlogo Home Page. [Online]. Tersedia di : https://ccl.northwestern.edu/netlogo/references.shtml. [Diakses pada Januari 2018]

[8] Sukarno, S.A., Atitallah, R.B. and Djemai, M., 2018, Approximation Algorithm for 3-Dimensional Vehicle Routing Problem for Fleet of Multi-Agents. 2018 6th International Conference on Control Engineering & Information Technology (CEIT), Istanbul, Turkey, pp. 1-6. doi: https://doi.org/10.1109/CEIT.2018.8751928

[9] Umbarkar, A.J. and Sheth, P.D., 2015, Crossover Operators in Genetic Algorithms: A Review, ICTACT Journal on Soft Computing, Vol. 06, Issue: 01.

[10] Hileman, W., Netlogo User Community Models, Classic Travelling Salesman. [Online]. Tersedia di: http://ccl.northwestern.edu/netlogo/models/community. [Diakses pada Januari 2018]

[11] Sukarno, S.A., 2019. Approximation Methods to Vehicle Routing Problem for a Drone Fleet Management. Ph.D. Valenciennes, France: Université Polytechnique Hauts de France.