Available *online* at: https://journal.lppmunindra.ac.id/index.php/JOTI

Jurnal Optimasi Teknik Industri

| ISSN (Print) 2656-3789 | ISSN (Online)2657-0181 |

Optimasi *Open Location Routing Problem* Menggunakan Metode Metaheuristik *Simulated Annealing*, *Large Neighborhood Search*, dan *Adaptive Large Neighborhood Search*

Audi Ziyad Afkar Muhammad[1], Mochamad Egidio Pramudya Kafi[2], Narsico Rafael Hasibuan[3], Noor Athiea Rahmawatie[4], Achmad Pratama Rifai[5]*

[1,2,3,4,5]*Department of Mechanical and Industrial Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia*

*Corresponding author: achmad.p.rifai@ugm.ac.id

ARTICLE INFORMATION

Received : 3 Agustus 2024
Revised : 10 Februari 2025
Accepted : 26 Maret 2025
Available online : 30 Maret 2025

KATA KUNCI

Open-Location Routing Problem;
Simulated Annealing;
Large Neighborhood Search;
Adaptive Large Neighborhood Search;

ABSTRAK

Penelitian ini menyelesaikan permasalahan *Open-Location Routing Problem* (OLRP) yang merupakan variasi dari permasalahan *Capacitated-Location Routing Problem* (CLRP). OLRP ini berkembang dan menjadi relevan seiring dengan meningkatnya penggunaan jasa perusahaan logistik pihak ketiga. Perbedaan utama OLRP dengan CLRP adalah kendaraan tidak kembali ke pusat distribusi setelah melayani pelanggan dalam OLRP. OLRP dikembangkan dengan tujuan utama untuk mengurangi total biaya yang terdiri atas biaya operasional fasilitas, biaya tetap kendaraan, dan biaya perjalanan. Dalam penelitian ini dilakukan pengembangan dan perbandingan beberapa pendekatan metaheuristik berbasis *Simulated Annealing* (SA), *Large Neighborhood Search* (LNS), dan *Adaptive Large Neighborhood Search* (ALNS) untuk menyelesaikan OLRP. Model yang diuji memiliki 21 pelanggan dan 5 depot potensial. Hasil pengujian menunjukkan bahwa ketiga metode menyelesaikan OLRP dengan waktu komputasi dan hasil total biaya yang bervariasi. Simulated annealing menghasilkan total biaya yang minimal sebesar $321,0406 dengan waktu komputasi hanya 54 detik.

I. PENDAHULUAN

Open Location Routing Problem (OLRP) adalah sebuah masalah yang merupakan kombinasi dari penentuan lokasi fasilitas (*facility location problem*) dan penentuan rute kendaraan (*Vehicle Routing Problem*). Masalah ini akan muncul ketika sebuah perusahaan memilih untuk menggunakan perusahaan pihak ketiga dalam aktivitas logistiknya. Ketika sebuah perusahaan mengontrak aktivitas distribusinya kepada perusahaan pihak ketiga, kendaraan layanan dimiliki oleh perusahaan tersebut. Oleh karena itu, setelah melayani pelanggan, kendaraan ini kembali ke perusahaan logistik, bukan ke pusat distribusi atau depot perusahaan yang menyewanya. Oleh karena itu, kami mengusulkan masalah penentuan lokasi-rute terbuka (OLRP) untuk menangani situasi ini.

OLRP memiliki banyak aplikasi dalam praktiknya. Misalnya, perusahaan surat kabar dan perusahaan periklanan harus menentukan lokasi optimal dari pabrik cetak mereka untuk menutupi area layanan mereka dengan biaya minimal. Karena perusahaan ini biasanya tidak memiliki armada kendaraan sendiri untuk mendistribusikan surat kabar atau iklan cetak, menyewa perusahaan pihak ketiga untuk mendistribusikan produk mereka dari pabrik cetak adalah opsi yang hemat biaya dan efisien. Perbedaan antara nilai solusi CLRP dan OLRP dapat menjadi referensi yang baik bagi perusahaan yang mempertimbangkan apakah harus mengalihdayakan aktivitas logistiknya kepada perusahaan pihak ketiga

https://doi.org/10.30998/joti.v7i1.24873

atau tidak karena perbedaan ini mewakili potensi penghematan dari alih daya logistik, mereka juga dapat memfasilitasi negosiasi harga antara perusahaan dan penyedia layanan TPL.

Dalam permasalahan OLRP yang akan diselesaikan, terdapat 21 pelanggan dengan koordinat dan permintaan yang telah ditentukan ditunjukkan pada Tabel 1, serta 5 depot potensial dengan koordinat, kapasitas produksi, dan biaya pembukaan yang diketahui seperti yang ditampilkan dalam Tabel 2. Untuk melayani pelanggan dalam pengiriman produk dari depot, digunakan sebuah armada kendaraan homogen yang memiliki kapasitas sebesar 6000 produk. Untuk membuka armada kendaraan baru tidak membutuhkan biaya. Terdapat beberapa batasan tambahan yang harus dipenuhi oleh solusi OLRP. Total permintaan yang dilayani oleh sebuah kendaraan tidak boleh melebihi kapasitas kendaraan. Setiap pelanggan harus dikunjungi sekali dan hanya sekali oleh tepat satu kendaraan, dan permintaan setiap pelanggan harus dipenuhi. Total permintaan pelanggan yang ditugaskan ke sebuah depot tidak boleh melebihi kapasitas depot. Terakhir, setiap rute kendaraan harus dimulai dari depot dan berakhir pada pelanggan.

Tabel 1. Informasi Pelanggan

Node	Koordinat X	Koordinat Y	*Demand*
6	151	264	1100
7	159	261	700
8	130	254	800
9	128	252	1400
10	163	247	2100
11	146	246	400
12	161	242	800
13	142	239	100
14	163	236	500
15	148	232	600
16	128	231	1200
17	156	217	1300
18	129	214	1300
19	146	208	300
20	164	208	900
21	141	206	2100
22	147	193	1000
23	164	193	900
24	129	189	2500
25	155	185	1800
26	139	182	700

Tabel 2. Informasi Depot Potensial

Node	Koordinat X	Koordinat Y	Kapasitas	*Fixed Cost* Depot	*Fixed Cost* Armada
1	136	194	15000	50	0
2	143	237	15000	50	0
3	136	216	15000	50	0
4	137	204	15000	50	0
5	128	197	15000	50	0

Penelitian ini berfokus pada pengembangan solusi untuk masalah *Open-Location Routing Problem* (OLRP), sebuah variasi dari masalah *Capacitated-Location Routing Problem* (CLRP). OLRP adalah permasalahan optimasi dalam logistik yang bertujuan untuk menentukan lokasi depot serta rute kendaraan yang optimal untuk melayani pelanggan, dengan mempertimbangkan kapabilitas depot dan kendaraan.

Untuk menyelesaikan OLRP, penelitian ini mengembangkan dan membandingkan beberapa pendekatan metaheuristik berbasis *Simulated Annealing* (SA), *Large Neighborhood Search* (LNS), dan *Adaptive Large Neighborhood Search* (ALNS). Setiap pendekatan ini dirancang untuk menemukan solusi yang mendekati optimal dengan cara yang efisien, terutama untuk masalah-masalah yang kompleks dan berdimensi tinggi. Adapun asumsi dan batasan yang diterapkan dalam penelitian ini adalah sebagai berikut:

1. **Biaya Transportasi dari Titik Akhir Konsumen ke Depot:** Biaya transportasi dari titik akhir konsumen ke depot tidak dimasukkan dalam perhitungan nilai objektif. Asumsi ini dibuat untuk menyederhanakan model dan fokus pada optimasi rute.
2. **Kunjungan Konsumen oleh Kendaraan:** Setiap konsumen hanya diperbolehkan dikunjungi satu kali oleh satu kendaraan. Batasan ini memastikan bahwa tidak ada duplikasi layanan dan setiap konsumen dilayani secara efisien.
3. **Penyediaan Kendaraan oleh Pihak Ketiga:** Kendaraan disediakan oleh pihak ketiga (*third party*), sehingga tidak ada batasan pada jumlah kendaraan yang tersedia. Hal ini memungkinkan penelitian untuk fokus pada optimasi rute dan lokasi tanpa mempertimbangkan keterbatasan armada kendaraan.

II. METODE

1. *Simulated Annealing*

Simulated Annealing (SA) merupakan salah satu algoritma dalam metaheuristik yang diperkenalkan oleh [1] dengan menggunakan analogi antara *annealing* padatan dan penyelesaian masalah optimasi kombinatorial. Algoritma SA merupakan metaheuristik *stochastic* yang menggunakan metode

 https://doi.org/10.30998/joti.v7i1.24873

pencarian secara random serta penerimaan yang juga random. Meskipun begitu, algoritma SA ini dirancang agar proses pencarian tidak cepat terkunci pada titik minimum lokal sehingga selama proses pencarian solusi tersebut, SA tidak hanya akan menerima solusi yang lebih baik namun juga solusi yang lebih buruk dengan probabilitas yang akan menurun seiring waktu. Tujuan dari adanya solusi yang lebih buruk tersebut adalah untuk menghindari konvergensi pencarian pada titik minimum lokal. Probabilitas penerimaan solusi yang lebih buruk tersebut ditentukan berdasarkan dua parameter yakni temperature serta perbedaan nilai *objective function* dari solusi saat ini dengan solusi *neighbor* di mana ketika nilai suhu tinggi maka probabilitas untuk menerima solusi yang lebih buruk juga tinggi dan seiring dengan penurunan suhu tersebut, probabilitas untuk menerima solusi yang lebih buruk juga menurun [2].

Algoritma SA ini sudah banyak digunakan sejak saat itu dan terus berkembang hingga saat ini dikarenakan mampu memecahkan banyak masalah optimasi kombinatorial dalam kehidupan nyata. Contohnya adalah untuk menyelesaikan permasalahan penjadwalan [3], masalah perencanaan tata letak fasilitas [4], *vehicle routing problem* [5], *travel salesman problem* [2], dan banyak permasalahan lain yang juga dapat diselesaikan menggunakan algoritma SA.

2. *Large Neighborhood Search*

Large Neighborhood Search (LNS) adalah sebuah algoritma dalam metaheuristik yang digunakan untuk mengoptimalkan kembali sebuah solusi secara iteratif dengan menggunakan pendekatan penghancuran (*destroy*) dan perbaikan (*repair*) hingga kondisi penghentian tertentu tercapai [6]. LNS (*Large Neighborhood Search*) pertama kali dikembangkan oleh Shaw pada tahun 1998 dan telah menjadi pendekatan yang cukup banyak digunakan dalam menyelesaikan berbagai masalah optimasi kombinatorial [6]. Secara khusus, dalam pendekatan LNS langkah penghancuran merupakan memecah atau membongkar sebagian dari solusi yang sudah ada dengan tujuan untuk menciptakan variasi baru dalam solusi tersebut kemudian langkah perbaikan dilakukan untuk memperbaiki atau membangun kembali solusi yang rusak akibat dari penghancuran untuk menghasilkan solusi berikutnya. Dibandingkan dengan heuristik pencarian lokal biasa, LNS lebih efektif dalam menjelajahi ruang solusi karena adanya kedua operator tersebut dapat membuat lingkungan yang lebih besar pada setiap iterasi.

3. *Adaptive Large Neighborhood Search*

ALNS (*Adaptive Large Neighborhood Search*) merupakan pengembangan dari metode LNS (*Large Neighborhood Search*). Metode ini dikembangkan karena sering ditemukan kasus di mana solusi hanya terjebak pada solusi optimum lokal dan gagal menemukan solusi optimum global pada metode LNS [7]. Sebagai pengembangan dari LNS klasik, ALNS terdiri dari penyematan operator penghancuran dan perbaikan dengan prosedur pemilihan adaptif [6].

Perbedaan utama dari penyesuaian adaptif dalam ALNS adalah adanya proses pencatatan dampak dari sebuah operator dengan mencatat informasi setiap kali sebuah operator terikat dalam proses LNS di tiap iterasi [8]. Dengan adanya proses tersebut, operator yang secara historis berkinerja baik akan memiliki probabilitas lebih tinggi untuk dipilih pada iterasi berikutnya.

Metode ALNS memperbaiki kelemahan LNS dengan cara lebih adaptif dan responsif terhadap performa masing-masing operator selama iterasi pencarian solusi. Setiap kali operator diterapkan, dampaknya terhadap solusi saat ini dinilai dan dicatat, memungkinkan algoritma untuk belajar dari iterasi sebelumnya. Hal ini memberikan fleksibilitas dalam memilih operator yang paling efektif berdasarkan performa historis, meningkatkan peluang menemukan solusi optimum global.

Penggunaan ALNS terbukti lebih efektif dalam berbagai masalah optimasi yang kompleks, termasuk dalam permasalahan *flow shop scheduling* [9], *drilling process parameter* [10], *vehicle routing problem* untuk distribusi beras [11], *sequence-dependent job sequencing and tool switching* [12, 13], dan *flying sidekick traveling salesman problem with multiple drops* [7]. ALNS mampu mengeksplorasi ruang solusi yang lebih luas dengan lebih efisien dibandingkan metode tradisional, karena pemilihan operator yang adaptif memungkinkan algoritma untuk menghindari perangkap optimum lokal.

Penelitian menunjukkan bahwa ALNS dapat memberikan hasil yang lebih baik dalam waktu yang lebih singkat dibandingkan dengan metode optimasi lainnya. Misalnya, dalam penelitian yang dilakukan oleh Pisinger dan Ropke [6], ALNS menunjukkan peningkatan kinerja signifikan dalam menyelesaikan masalah perutean kendaraan dibandingkan metode LNS klasik. Keunggulan ini terutama karena kemampuan ALNS untuk menyesuaikan strategi pencarian berdasarkan performa operator, memungkinkan eksplorasi yang lebih efektif dari ruang solusi.

III. HASIL DAN PEMBAHASAN

1. Model matematika

Tujuan utama dari *Open-Location Routing Problem* (OLRP) adalah meminimalkan total biaya sistem distribusi, yang mencakup biaya perjalanan

kendaraan, biaya pembukaan depot, dan biaya tetap kendaraan. *Objective function* digambarkan oleh *eq*. (1), di mana C_{ij} merupakan biaya perjalanan dari node i ke node j, X_{ij} adalah variabel biner yang bernilai 1 apabila rute dari node i ke j dilewati oleh kendaraan, dan 0 jika tidak. FD_k merupakan biaya pembukaan depot k, y_k adalah variabel biner yang bernilai 1 jika depot k dibuka, dan 0 jika tidak, FV merupakan biaya tetap kendaraan yang akan dikenakan untuk setiap kendaraan yang dioperasikan terlepas dari jarak atau jumlah pelanggan yang dilayani, dan x_{ki} adalah variabel biner yang bernilai 1 jika kendaraan melakukan perjalanan dari depot k ke *node* i. *Equation* (1) memformulasikan *objective function*, yaitu meminimasi total biaya yang terdiri dari biaya transportasi, biaya pembukaan depot, dan biaya tetap kendaraan. Model matematika dari OLRP yang diadaptasi dari Vincent dan Lin [14] difromulasikan sebagai berikut.

Objective function

$$\min z = \sum_{i\epsilon Nj}\sum_{i\epsilon Nc} c_{ij}x_{ij} + \sum_{k\epsilon N0} FD_k y_k + \sum_{k\epsilon N0} FV x_{ki} \quad (1)$$

Subject to:

$$\sum_{j\epsilon N} x_{ij} = 1, \quad \forall\, i \epsilon N_c \quad (2)$$

$$\sum_{j\epsilon N} x_{ji} = \sum_{j\epsilon N} x_{ij}, \quad \forall\, i \epsilon N \quad (3)$$

$$\sum_{j\epsilon N} U_{ji} - \sum_{j\epsilon N} U_{ij} = d_i, \quad \forall\, i \epsilon N_c \quad (4)$$

$$U_{ij} \leq CV x_{ij}, \quad \forall\, i,j \epsilon N, i \neq j \quad (5)$$

$$\sum_{j\epsilon N_c} U_{kj} = \sum_{j\epsilon N_c} z_{jk} \sum_{j\epsilon N_c} d_j\,, \qquad \forall\, k \epsilon N_0 \quad (6)$$

$$\sum_{j\epsilon N_c} U_{jk} = 0, \quad \forall\, k \epsilon N_0 \quad (7)$$

$$U_{ij} \leq (CV - d_i)x_{ij}, \quad \forall\, i \epsilon N_c, \forall\, j \epsilon N \quad (8)$$

$$U_{ij} \geq d_j x_{ij}, \quad \forall\, i \epsilon N, \forall\, j \epsilon N_c \quad (9)$$

$$\sum_{k\epsilon N0} z_{ik} = 1, \quad \forall\, i \epsilon N_c \quad (10)$$

$$\sum_{i\epsilon N_c} d_j z_{ik} \leq CD_k y_k, \forall\, k \epsilon N_0 \quad (11)$$

$$\sum_{i\epsilon N_c} d_j z_{ik} \leq CD_k y_k, \forall\, k \epsilon N_0 \quad (12)$$

$$x_{ik} \leq z_{ik}, \quad \forall\, i \epsilon N_c, \forall\, k \epsilon N_0 \quad (13)$$

$$x_{ki} \leq z_{ik}, \quad \forall\, i \epsilon N_c, \forall\, k \epsilon N_0 \quad (14)$$

$$x_{ij} + z_{ik} + \sum_{m\epsilon N_0, m\neq k} z_{jm} \leq 2, \quad \forall\, i, \forall\, j \epsilon N_c, i \neq j, \forall\, k \epsilon N_0 \quad (15)$$

Constraint (2) memastikan bahwa setiap pelanggan i dilayani tepat satu kali oleh kendaraan sehingga harus ada tepat satu kendaraan mengunjungi setiap pelanggan. *Equation* (3) memastikan bahwa jumlah kendaraan yang masuk ke *node* i sama dengan jumlah kendaraan yang keluar dari *node* i. *Equation* (4) memastikan *demand* yang masuk ke *node* i harus sama dengan *demand* yang keluar ditambah dengan *demand* di *node* tersebut. *Equation* (5) memastikan bahwa *demand* yang tersisa yang dibawa oleh kendaraan tidak boleh melebihi kapasitas kendaraan itu sendiri. *Equation* (6) memastikan bahwa total *demand* dari pelanggan yang ditugaskan kepada depot tertentu tidak boleh melebihi kapasitas dari depot tersebut. *Equation* (7) memastikan bahwa seluruh *demand* yang tersisa setelah kendaraan melayani pelanggan terakhir adalah 0 sehingga seluruh *demand* pelanggan harus dipenuhi sebelum kendaraan selesai dengan rutenya. *Equation* (8) memastikan kapasitas kendaraan tidak melebihi batasnya saat melakukan perjalan dari *node* i ke *node* j. *Equation* (9) mematikan bahwa sisa kapasitas kendaraan setelah perjalanan dari *node* i ke j tidak melebihi sisa kapasitas yang tersedia untuk memenuhi permintaan pelanggan i. *Equation* (10) memastikan bahwa setiap pelanggan i harus ditugaskan tepat di satu depot atau dilayani oleh kendaraan yang berasal dari satu depot tertentu. *Equation* (11) memastikan bahwa total demand pelanggan yang dilayani oleh satu depot tidak boleh melebihi kapasitas depot tersebut. *Equation* (12) memastikan bahwa jika depot k ditugaskan untuk melayani pelanggan i, maka harus ada kendaraan yang melakukan perjalanan dari *node* i ke k. *Equation* (13) memastikan bahwa depot hanya bisa ditugaskan untuk melayani pelanggan jika ada perjalanan yang dilakukan dari pelanggan ke depot. *Equation* (14) memastikan bahwa jumlah kombinasi x_{ij} dan z_{ik} yang aktif tidak melebihi m.

2. *Solution representation*

Representasi solusi akan berbentuk urutan depot dan pelanggan yang akan dikunjungi serta *dummy zero* (Ndummy). Depot (N0) dinyatakan sebagai {1, 2, ..., N0} dan pelanggan (Nc) dinyatakan sebagai {N0+1, N0+2, ..., N0+Nc}. Rute harus dimulai dari depot sehingga angka pertama dalam urutan akan merepresentasikan depot. Setiap depot akan melayani pelanggan-pelanggan di antara dirinya dan depot berikutnya. Pelanggan-pelanggan ditambahkan satu per satu, dari kiri ke kanan, ke dalam rute saat ini dari depot tersebut. Dalam proses pengisian rute tersebut

 https://doi.org/10.30998/joti.v7i1.24873

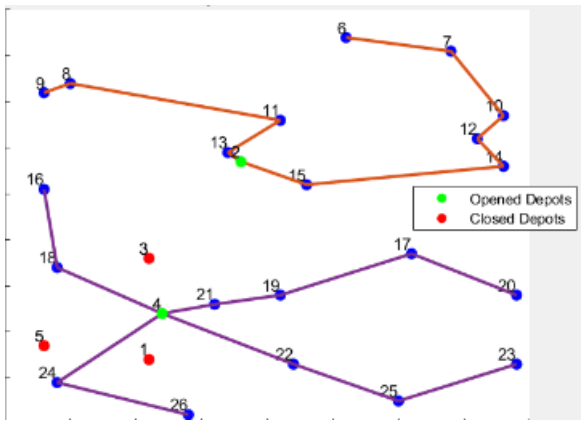
juga akan dipertimbangkan kapasitas kendaraan sehingga pelanggan akan terus ditambahkan asalkan batasan kapasitas kendaraan tidak dilanggar.

Ketika kapasitas kendaraan telah dilanggar, maka akan digunakan *dummy zeros* untuk mengakhiri sebuah rute dan memulai rute baru. *Dummy zeros* juga dapat digunakan meskipun *demand* yang terakumulasi belum melebihi kapasitas kendaraan. Penentuan *dummy zeros* ini didapatkan dari sebuah perhitungan dengan *Equation* (15) di mana CV mewakili kapasitas kendaraan, dan d_i adalah permintaan dari pelanggan i dengan ⌈x⌉ mewakili bilangan bulat terkecil yang lebih besar atau sama dengan x.

$$\sum_{i \epsilon N_c} \frac{d_i}{CV} \tag{15}$$

5	3	0	1	0	4	18	16	0	0	0	21	19	17	20	0	24	26	0	22	25	23	0	2	15	14	12	10	7	6	0	13	11	8	9

Gambar 1. *Solution Representation*



Gambar 2. Visualisasi *Routing*

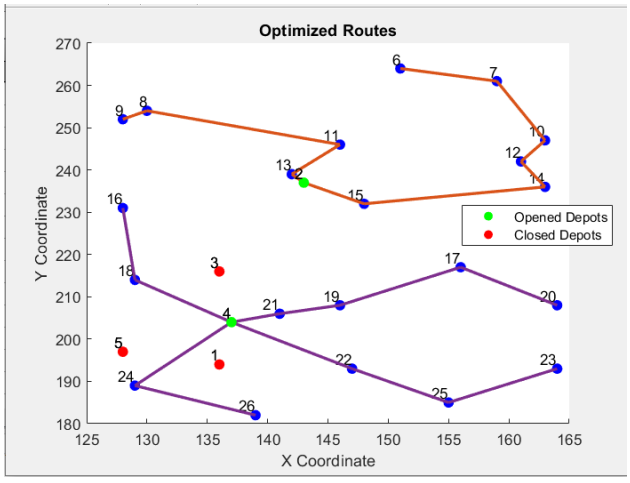
Dengan menggunakan *dummy zeros* dalam representasi solusi, pembuatan rute dapat dieksekusi secara acak sehingga menciptakan eksplorasi dan ruang pencarian menjadi lebih luas dan solusi-solusi yang lebih baik dapat ditemukan. Gambar 1 mengilustrasikan *solution representation* yang digunakan dalam penelitian ini, sedangkan Gambar 2 menunjukkan visualisasi *routing*.

3. Hasil *Simulated Annealing*

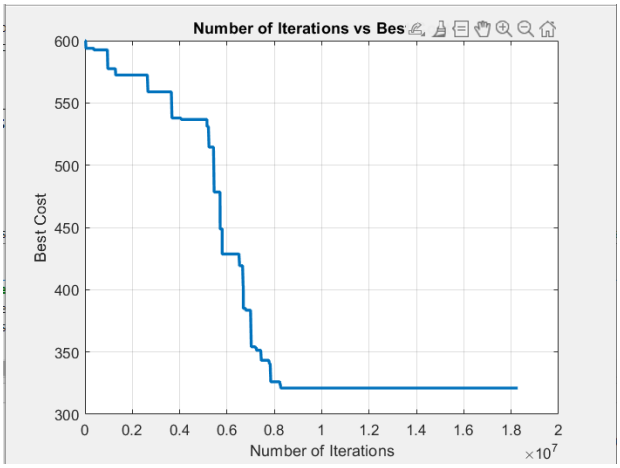
Penyelesaian *Open Location Routing Problem* (OLRP) menggunakan algoritma *Simulated Annealing* (SA) melibatkan beberapa tahapan yang terstruktur, dengan parameter algoritma yang telah ditetapkan. Suhu awal diatur sebesar 10.000, suhu akhir sangat rendah yaitu 1e-6, dan laju pendinginan sebesar 0,999. Setiap kali suhu diturunkan, proses tersebut mencakup iterasi internal dengan maksimum 1000 iterasi. Inisialisasi solusi awal menggunakan *Greedy Algorithm*. Operasi yang digunakan dalam proses optimasi ini mencakup tiga jenis operator utama untuk memodifikasi solusi, yaitu *swap operator*, *insert operator*, dan *two-opt move operator*. Dalam penyelesaian *Open Location Routing Problem* (OLRP) menggunakan metode *Simulated Annealing* (SA) dengan bantuan *software* MATLAB dan spesifikasi AMD Ryzen 7 4800H, kami berhasil mencapai solusi terbaik algoritma ini dengan nilai *total cost* yang tercatat sebesar $321,0406 dengan waktu komputasi 54 detik. Gambar 3 menunjukkan hasil komputasi SA, dimana rute yang didapatkan divisualisasikan pada Gambar 4.

5	3	0	1	0	4	18	16	0	0	0	21	19	17	20	0	24	26	0	22	25	23	0	2	15	14	12	10	7	6	0	13	11	8	9

Gambar 3. Hasil komputasi *Simulated Annealing*



Gambar 4. Rute *Simulated Annealing*



Gambar 5. Grafik Konvergensi *Simulated Annealing*

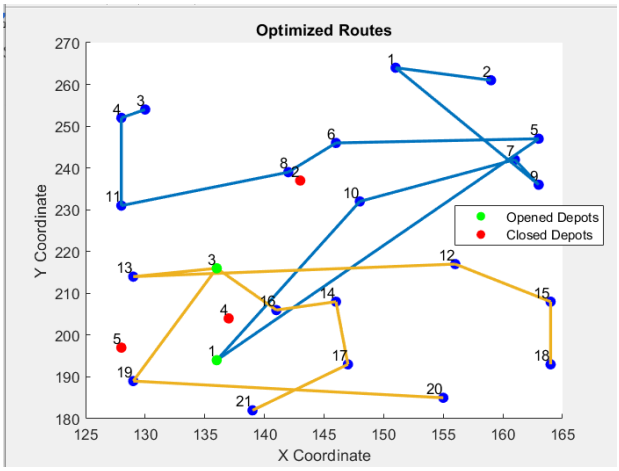
Dari Gambar 4. terlihat bahwa depot yang dipilih untuk melayani 21 *customer* adalah depot 2 dan depot 4. Di depot 2 terdapat dua armada kendaraan yang dibuka dengan rute yang berbeda. Kendaraan pertama memiliki rute 2-13-11-8-9 dan kendaraan kedua memiliki rute 2-15-14-12-10-7-6. Kemudian, pada depot 4 terdapat empat armada kendaraan dengan rute berturut-turut 4-18-16, 4-24-26, 4-21-19-17-20, dan 4-22-25-23. Pada Gambar 5, terlihat bahwa nilai objektif sudah stabil dari setengah total iterasi yang dibutuhkan. Hal ini membuktikan bahwa nilai objektif yang didapat dari algoritma ini sudah mendekati nilai optimal.

4. Hasil *Large Neighborhood Search*

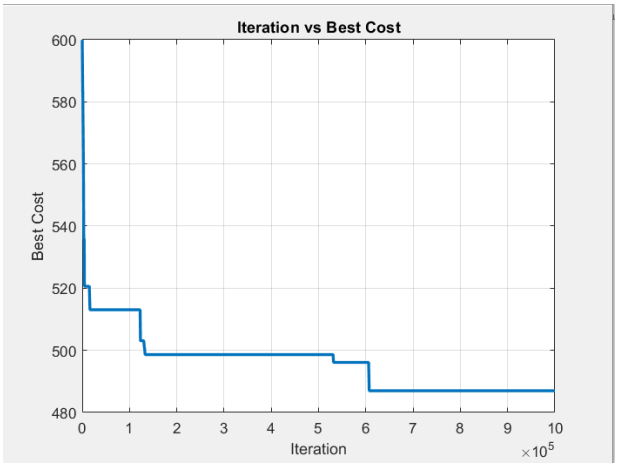
Penyelesaian *Open Location Routing Problem* (OLRP) menggunakan algoritma *Large Neighborhood Search* (LNS) melibatkan beberapa tahapan yang terstruktur, dengan parameter algoritma yang telah ditetapkan, seperti maksimum iterasi sebesar 1.000.000. Inisialisasi solusi awal menggunakan *Greedy Algorithm*. Operasi yang digunakan dalam proses optimasi ini mencakup empat jenis *destroy operator* dan *repair operator* untuk memodifikasi solusi, yaitu *random removal*, *worst removal*, *random repair*, dan *greedy repair*. Dalam penyelesaian *Open Location Routing Problem* (OLRP) menggunakan *Large Neighborhood Search* (LNS) dengan *software* MATLAB dengan spesifikasi AMD Ryzen 7 4800H, kami berhasil mencapai solusi terbaik algoritma ini dengan nilai *total cost* yang tercatat sebesar $486,9949 dengan waktu komputasi 66 detik. Gambar 6 menunjukkan hasil komputasi LNS, dimana rute yang didapatkan divisualisasikan pada Gambar 7.

1	10	11	13	16	9	8	0	15	12	14	6	7	0	3	18	17	20	23	0	21	19	22	26	0	24	25

Gambar 6. Hasil komputasi *Large Neighborhood Search*



Gambar 7. Rute *Large Neighborhood Search*



Gambar 8. Grafik Konvergensi *Large Neighborhood Search*

Dari Gambar 7. terlihat bahwa depot yang dipilih untuk melayani 21 *customer* adalah depot 1 dan depot 3. Di depot 1 terdapat dua armada kendaraan yang dibuka dengan rute yang berbeda. Kendaraan pertama memiliki rute 1-10-11-13-16-9-8 dan kendaraan kedua memiliki rute 1-15-12-14-6-7. Kemudian, pada depot 3 terdapat empat armada kendaraan dengan rute berturut-turut 3-18-17-20-23, 3-21-19-22-26, 3-24-25. Pada Gambar 8, terlihat bahwa nilai objektif sudah stabil dari 40% akhir dari total iterasi yang dibutuhkan. Hal ini membuktikan bahwa proses pencarian solusi menggunakan LNS telah mencapai *convergence*.

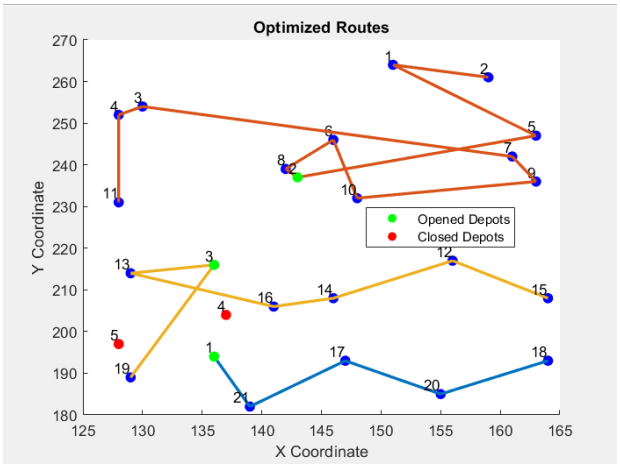
5. Hasil *Adaptive Large Neighborhood Search*

Penyelesaian *Open Location Routing Problem* (OLRP) menggunakan algoritma *Adaptive Large Neighborhood Search* (ALNS) melibatkan beberapa tahapan yang terstruktur, dengan parameter algoritma yang telah ditetapkan, seperti maksimum iterasi sebesar 1.000.000, *weight alpha* (*decay factor*) sebesar 0,9, *weight beta* sebesar 0,8 dan 0,1, dan *diversification rate* sebesar 0,1. Inisialisasi solusi awal menggunakan *Greedy Algorithm*. Operasi yang digunakan dalam proses optimasi ini mencakup empat jenis *destroy operator* dan *repair operator* untuk memodifikasi solusi, yaitu *random removal*, *worst removal*, *random repair*, dan *greedy repair*. Dalam penyelesaian *Open Location Routing Problem* (OLRP) menggunakan *Adaptive Large Neighborhood Search* (ALNS) dengan *software* MATLAB dan spesifikasi AMD Ryzen 7 4800H, kami berhasil mencapai solusi terbaik algoritma ini dengan nilai *total cost* yang tercatat sebesar $434,8739 dengan waktu komputasi 59,3 detik. Gambar 9 menunjukkan hasil komputasi LNS, dimana rute yang didapatkan divisualisasikan pada Gambar 10.

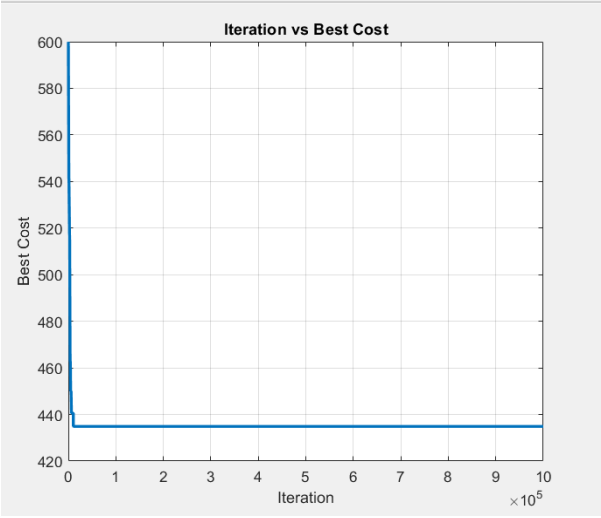
https://doi.org/10.30998/joti.v7i1.24873

1	26	22	25	23	0	2	13	11	15	14	12	8	9	16	0	10	6	7	0	3	18	21	19	17	20	0	24

Gambar 9. Hasil komputasi *Adaptive Large Neighborhood Search*



Gambar 10. Rute *Adaptive Large Neighborhood Search*



Gambar 10. Grafik Konvergensi *Adaptive Large Neighborhood Search*

Dari Gambar 10. terlihat bahwa depot yang dipilih untuk melayani 21 *customer* adalah depot 1, depot 2, dan depot 3. Di depot 1 terdapat satu armada kendaraan yang dibuka dengan rute 1-21-17-20-18. Di depot 2 terdapat dua armada kendaraan yang dibuka dengan rute yang berbeda. Kendaraan pertama memiliki rute 2-13-11-15-14-12-8-9-16 dan kendaraan kedua memiliki rute 2-10-6-7. Kemudian, pada depot 3 terdapat dua armada kendaraan dengan rute berturut-turut 3-18-21-19-17-20 dan 3-24. Pada Gambar 8, terlihat bahwa nilai objektif sudah stabil dari awal, yang menunjukkan proses pencarian solusi yang cepat.

IV. SIMPULAN

Dalam penyelesaian *Open Location Routing Problem* (OLRP) 5 depot dan 21 pelanggan dengan menggunakan ketiga metode algoritma optimasi ini, *Simulated Annealing* (SA) menunjukkan hasil yang paling efektif dan efisien dengan total biaya terendah dan waktu komputasi tercepat. Total biaya yang didapat sebesar $321,0406 dengan waktu komputasi 54 detik. *Adaptive Large Neighborhood Search* (ALNS) memberikan hasil lebih baik dibandingkan dengan *Large Neighborhood Search* (LNS) dalam hal total biaya dan waktu komputasi. Waktu komputasi *Adaptive Large Neighborhood Search* (ALNS) lebih cepat dibandingkan dengan *Large Neighborhood Search* (LNS) karena pemilihan *operator* ALNS menggunakan *weight destroy* dan *repair*, sedangkan LNS hanya menggunakan *random number* sehingga lebih lama waktu komputasinya. Perbandingan ketiga metode tertera pada Tabel 3.

Tabel 3. Perbandingan Hasil Ketiga Metode

Metode	Waktu Komputasi (detik)	Total Biaya ($)
Simulated Annealing	54	321,0406
Large Neighborhood Search	66	486,9949
Adaptive Large Neighborhood Search	59,3	434,8739

Metode *Large Neighborhood Search* (LNS) dan *Adaptive Large Neighborhood Search* (ALNS) menunjukkan nilai objektif yang lebih rendah dibandingkan dengan metode *Simulated Annealing*. Hal ini disebabkan oleh keterbatasan operator *destroy* dan *repair* yang dipilih dalam metode tersebut. Secara khusus, pada operator *repair* yaitu *greedy repair* eksploitasi solusi yang dilakukan masih kurang optimal. Dalam *greedy repair*, hanya dipertimbangkan jarak terendah dari titik sebelumnya ke titik berikutnya. Ini berarti bahwa algoritma hanya fokus pada mengurangi jarak dalam langkah perbaikan tanpa memperhatikan faktor-faktor lain yang mungkin meningkatkan efisiensi keseluruhan solusi.Meskipun biaya transportasi dalam kasus ini bernilai nol, strategi *repair* yang hanya berfokus pada jarak terendah gagal untuk memanfaatkan keuntungan ini, sehingga tidak mengoptimalkan solusi secara keseluruhan.

Disarankan agar metode *Large Neighborhood Search* dan *Adaptive Large Neighborhood Search* mengadopsi operator repair alternatif yang mampu mengeksploitasi solusi objektif secara lebih efektif. Penggunaan operator repair yang lebih canggih dapat

membantu dalam mengeksplorasi ruang solusi dengan lebih baik. Selain itu, disarankan juga untuk mengembangkan kasus dengan meningkatkan biaya transportasi. Hal ini membantu mengevaluasi variasi objektif berdasarkan berbagai metode dan skenario dari skema biaya transportasi yang berbeda, dapat diperoleh pemahaman yang lebih komprehensif mengenai kinerja metode yang digunakan.

ACKNOWLEDGEMENT

Penelitian ini didanai oleh Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia, dengan nomor hibah 048/E5/PG.02.00.PL/2024 dan 2674/UN1/DITLIT/PT.01.03/2024.

REFERENSI

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983, doi: https://doi.org/10.1126/science.220.4598.671.

[2] H. Bayram and R. Şahin, "A New Simulated Annealing Approach for Travelling Salesman Problem," *Mathematical and Computational Applications*, vol. 18, no. 3, pp. 313–322, Dec. 2013, doi: https://doi.org/10.3390/mca18030313.

[3] R. Hanafi and E. Kozan, "A hybrid constructive heuristic and simulated annealing for railway crew scheduling," *Computers & Industrial Engineering*, vol. 70, pp. 11–19, Apr. 2014, doi: https://doi.org/10.1016/j.cie.2014.01.002.

[4] N. Moradi and Sh. Shadrokh, "A simulated annealing optimization algorithm for equal and un-equal area construction site layout problem," *DOAJ (DOAJ: Directory of Open Access Journals)*, Jun. 2019, doi: https://doi.org/10.22105/riej.2019.169867.1073.

[5] Nur, V. F. Yu, C. Bachtiyar, and Sukoyo, "A Simulated Annealing Heuristic for the Capacitated Green Vehicle Routing Problem," *Mathematical Problems in Engineering*, vol. 2019, pp. 1–18, Jan. 2019, doi: https://doi.org/10.1155/2019/2358258.

[6] D. Pisinger and S. Ropke, "Large neighborhood search," in *Handbook of Metaheuristics*, pp. 99–127.

[7] S. T. Windras Mara, R. Norcahyo, P. Jodiawan, L. Lusiantoro, and A. P. Rifai, "A survey of adaptive large neighborhood search algorithms and applications," *Computers & Operations Research*, vol. 146, p. 105903, Oct. 2022, doi: https://doi.org/10.1016/j.cor.2022.105903.

[8] Jean-Baptiste Mairy, P. Schaus, and Y. Deville, "Generic adaptive heuristics for large neighborhood search," *Principles and Practice of Constraint Programming*, Jan. 2010.

[9] A. Rifai, Huu Phuc Nguyen, and Siti Zawiah Md Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," *Applied Soft Computing*, vol. 40, pp. 42–57, Mar. 2016, doi: https://doi.org/10.1016/j.asoc.2015.11.034.

[10] Rachmadi Norcahyo, Achmad Pratama Rifai, Muslim Mahardika, Gesang Nugroho, and Bobby, "Minimization of hole entry and exit surface delamination on carbon fiber reinforced polymer (CFRP) drilling process using BPNN-ALNS," *AIP conference proceedings*, Jan. 2024, doi: https://doi.org/10.1063/5.0179672.

[11] Fadhil Wina Ramadhani, Wangi Pandan Sari, and Achmad Pratama Rifai, "Multi-Vehicle Capacitated Vehicle Routing Problem For Rice Commodities In Indonesia Considering The Factors Of Weather-Induced Damages And Carbon Emissions," *ASEAN Engineering Journal*, vol. 14, no. 2, pp. 195–207, May 2024, doi: https://doi.org/10.11113/aej.v14.21096.

[12] Achmad Pratama Rifai, Edi Sutoyo, Mara, and Siti Zawiah Md Dawal, "Multiobjective Sequence-Dependent Job Sequencing and Tool Switching Problem," *IEEE Systems Journal*, vol. 17, no. 1, pp. 1395–1406, Mar. 2023, doi: https://doi.org/10.1109/jsyst.2022.3213767.

[13] A. P. Rifai, S. T. Windras Mara, and R. Norcahyo, "A two-stage heuristic for the sequence-dependent job sequencing and tool switching problem," *Computers & Industrial Engineering*, vol. 163, p. 107813, Jan. 2022, doi: https://doi.org/10.1016/j.cie.2021.107813.

[14] V. F. Yu and S.-W. Lin, "A simulated annealing heuristic for the open location-routing problem," *Computers & Operations Research*, vol. 62, pp. 184–196, Oct. 2015, doi: https://doi.org/10.1016/j.cor.2014.10.009.