JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 10
No. 1
April 2024

Metode Heuristik untuk Menentukan Jadwal Periodik Rute Kendaraan Distribusi Produk dengan Permintaan Stokastik

Mohamad Sofitra[#1], Dedi Wijayanto[#2], Noveicalistus H. Djanggu[#3]

[#]*Pusat Kajian Logistik & Rantai Pasok, Jurusan Teknik Industri, Universitas Tanjungpura*
Jl. Prof. Hadari Nawawi, Pontianak

[1]mohamad.sofitra@industrial.untan.ac.id
[2]dediwijayanto@industrial.untan.ac.id
[3]noveicalistus.djanggu@industrial.untan.ac.id

Abstrak— **Pada penelitian ini dirancang suatu metode solusi heuristik untuk menyelesaikan permasalahan *Periodic Vehicle Routing Problem with Stochastic Demand* pada kasus *rural logistics*. Tujuannya adalah memperoleh suatu set jadwal distribusi berupa *tour* apriori harian kendaraan ke sejumlah pengecer selama periode waktu *T* yang memberikan ekspektasi total waktu distribusi paling minimum. Algoritma solusi yang dirancang terdiri atas dua tahapan proses yaitu tahap konstruksi solusi dan tahap perbaikan solusi. Tahap konstruksi solusi dilakukan dengan menggunakan prinsip *cluster-first-route-second*. Sedangkan tahap perbaikan solusi digunakan strategi eksploratif dan strategi eksploitatif yang berfungsi mencari tetangga-tetangga solusi yang mungkin lebih baik dari solusi-solusi sementara yang telah berhasil ditemukan pada tahap konstruksi atau tahap perbaikan solusi pada iterasi sebelumnya. Strategi eksploratif terdiri atas dua metode yaitu metode *inter-tour circular exchange* dan *overtime cut-off*, sedangkan strategi eksploitatif menggunakan metode *intra-tour swap*. Pengujian kinerja algoritma solusi hasil rancangan dilakukan menggunakan 27 blok eksperimen hasil kombinasi tiga faktor yang diduga merupakan parameter-parameter yang berpengaruh pada kinerja algoritma solusi. Dengan menggunakan pengaturan parameter pencarian yang tepat, algoritma yang dirancang mampu menunjukkan kinerja yang memuaskan yaitu menghasilkan solusi yang berkualitas baik dengan waktu komputasi yang wajar.**

Kata kunci— ***vehicle routing problem, stochastic demand,* jadwal periodik*, rural logistics*, solusi heuristik**

I. PENDAHULUAN

Pada makalah ilmiah ini dikembangkan sebuah metode solusi heuristik untuk suatu permasalahan penentuan rute kendaraan periodik dengan permintaan stokastik (*periodic vehicle routing problem with stochastic demand*) atau disingkat PVRPSD. Berbeda dengan pemodelan permasalahan *vehicle routing problem* (VRP) deterministik yang bertujuan untuk mencari suatu jadwal distribusi produk dengan total ongkos terendah atau waktu/jarak rute *tour* terpendek, model PVRPSD dalam penelitian ini bertujuan untuk menyusun jadwal distribusi berbasis rute apriori yang memberikan ekspektasi total waktu distribusi paling minimal. Hasilnya berupa jadwal distribusi apriori untuk sejumlah hari dalam satu periode, untuk sejumlah kendaraan yang akan digunakan berikut jumlah produk yang akan didistribusikan oleh setiap kendaraan tersebut, serta rute *tour* apriori yang akan ditempuh oleh setiap kendaraan berupa urut-urutan kunjungan dari satu pengecer ke pengecer lain hingga kembali ke depot.

VRP adalah salah satu permasalahan penting dalam bidang distribusi dan transportasi. Permasalahan ini telah mengundang sangat banyak kontribusi dari para peneliti sejak pertama kali permasalahan ini diangkat oleh Dantzig dan Ramser di tahun 1959. Hingga kini, berbagai model dan metode solusi untuk VRP telah cukup banyak ditawarkan terutama untuk varian model VRP deterministik [1] [2] .

Upaya klasifikasi terhadap varian model VRP ini juga telah dilakukan, salah satunya adalah klasifikasi berdasarkan pada jenis keputusan yang dibuat oleh model [3]. Klasifikasi berdasarkan jenis keputusan ini meliputi: jenis VRP paling klasik yang selalu menjadi bagian inti dari varian model VRP turunannya yakni keputusan penugasan kendaraan dan pilihan rute yang harus ditempuh; kemudian model VRP *with profit* yaitu model turunan yang menambahkan keputusan tentang pelangan mana yang harus dilayani berdasarkan profit yang ditawarkannya; VRP *with split delivery* yaitu model turunan yang menambahkan keputusan tetang jumlah produk yang harus didistribusikan; selanjutnya, VRP *with multiple commodities* yaitu model turunan yang menambahkan keputusan tentang jenis-jenis produk yang harus didistribusikan sepanjang waktu. Terdapat pula varian model VRP yang terintegrasi dengan jenis keputusan lain dalam bidang logistik misalnya model *location-routing*

problem yang menggabungkan VRP dengan pemilihan lokasi untuk depot; *multi-echelon routing problem* yang menggabungkan keputusan logistik eselon jamak dengan model VRP; *inventory routing problem* yang menggabungkan VRP dengan keputusan dalam pengelolaan persediaan pada di sisi pelanggan.

Berbagai metode solusi heuristik/metaheuristik juga telah pernah dicoba untuk digunakan dalam pencarian solusi dari model-model VRP deterministik, seperti metaheuristik berbasis solusi tunggal misalnya *simulated annealing*, *large neighborhood search*, *tabu search*, dan *greedy randomized adaptive search procedure*; serta metaheuristik berbasis populasi solusi antara lain yang berbasis *evolutionary computation* dan *swarm intelligence* [4].

Model VRP deterministik mengasumsikan bahwa semua informasi yang diperlukan oleh model selalu tersedia dan dapat diketahui dengan pasti. Pada prakteknya asumsi deterministik ini sebagiannya tidak berlaku karena dalam kenyataannya terdapat faktor-faktor dari permasalahan yang menjadi parameter dari model bersifat tidak pasti. Ketidakpastian ini dapat bersumber dari adanya variasi dalam kejadian yang bersifat acak (*random*) misalnya adanya kejadian acak jumlah pengecer yang akan dilayani, jumlah *demand* dari pengecer tersebut, durasi tempuh kendaraan ataupun durasi pelayanan. Ketidakpastian ini hanya akan terungkap setelah berlalunya waktu sehingga keputusan yang telah terlanjur dibuat harus disesuaikan kembali sepanjang waktu. Penyesuaian atau koreksi atas keputusan yang telah dibuat ini berkonsekuensi pada bertambahnya ongkos total yang harus dikeluarkan.

Model VRP yang mempertimbangkan parameter yang bersifat acak baik probabilistik atau stokastik diistilahkan sebagai model PVRP atau SVRP. Model yang telah dikembangkan untuk PVRP atau SVRP hingga saat ini umumnya dapat dikelompokkan berdasarkan jenis parameter model yang bersifat probabilistik/stokastik. Misalnya model PVRP atau SVRP dengan parameter probabilistik/stokastik pada: *demand*, keberadaan pelanggan, waktu/ongkos perjalanan kendaraan, waktu/ongkos pelayanan pelanggan [5][6] ataupun destinasinya [7].

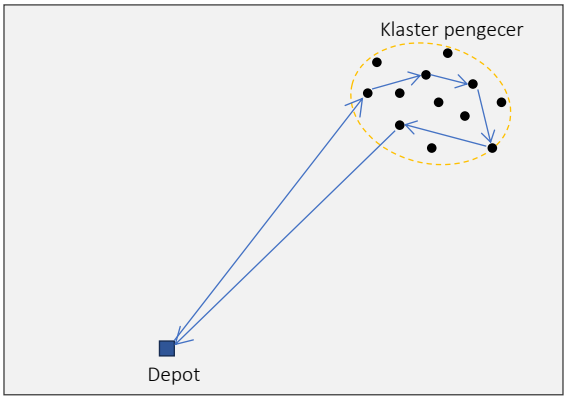
Sebagaimana model VRP deterministik, metode solusi yang telah coba dikembangkan untuk model PVRP atau SVRP secara umum terbagi menjadi dua yaitu: metode solusi eksak (antara lain: *integer L-shaped*, *branch and cut*, dan *branch and price*), dan metode heuristik atau metode sub optimal (antara lain: *adaptive large neighborhood search*, *rollout algorithms*, *progressive hedging*, dan *evolutionary algorithms*) [8], [9]. Namun demikian, usaha penelitian yang mengembangkan model dan metode solusi untuk PVRP atau SVRP ini dianggap masih lambat dan agak tertinggal dibandingkan varian model VRP deterministik [6][10].

Pilihan untuk mengembangkan metode solusi heuristik untuk permasalahan PVRPSD ini tidak terhindarkan mengingat permasalahan VRP deterministik merupakan permasalahan kombinatorial yang bersifat NP-*hard* dimana pencarian solusi bagi permasalah ini jika menggunakan metode optimal akan membutuhkan waktu komputasi yang meningkat secara eksponensial seiring dengan ukuran permasalahan yang semakin besar [11]. Sehingga penggunaan metode solusi optimal dalam menyelesaikan permasalahan VRP nyata yang berukuran besar menjadi tidak praktis. PVRPSD yang memiliki faktor permasalahan yang lebih banyak dari VRP dengan sendirinya juga merupakan permasalahan NP-*hard* karena kompleksitas yang semakin tinggi dengan adanya tambahan faktor permasalahan berupa penjadwalan periodik dan parameter *demand* yang bersifat *random* dalam permasalahan tersebut. Meskipun metode solusi heuristik tidak dapat menjamin diperolehnya solusi berupa jadwal distribusi yang optimal (ongkos terendah/rute terpendek) namun solusi heuristik masih dapat menghasilkan solusi dengan kualitas yang sangat baik dengan waktu komputasi yang wajar.

Adanya parameter stokastik serta sifat alami kombinatorial pada permasalahan PVRPSD mengakibatkan proses menghitung nilai ekspektasi total ongkos atau panjang rute dari suatu jadwal distribusi apriori menggunakan formulasi klasik seperti yang diusulkan Bertsimas [12] menjadi sangat menyulitkan. Untuk ukuran permasalahan realistis yang cukup besar, jumlah permutasi kejadian *demand* untuk seluruh pengecer menjadi sangat besar dan akan meningkat secara eksponensial mengikuti ukuran permasalahannya.

Terdapat sejumlah pendekatan yang dapat digunakan untuk mengatasi tantangan komputasi dalam menghitung ekspektasi total biaya atau ekspektasi total jarak dari sebuah *tour* apriori pada model PVRPSD antara lain menggunakan metode berbasis simulasi, seperti simulasi Monte Carlo atau *discrete-event simulation*, untuk memperkirakan ekspektasi total biaya atau jarak dari *tour* apriori yang diharapkan seperti pada artikel ilmiah [13] dan [14]. Dalam metode ini, sejumlah besar skenario acak dibuat berdasarkan parameter stokastik permasalahan, dan ekspektasi total biaya atau ekspektasi jarak dari *tour* apriori diestimasikan dari rata-rata yang dihasilkan oleh skenario-skenario ini. Pendekatan lainnya adalah dengan menggunakan *approximation algorithm* [15], seperti programma dinamis atau heuristik berbasis programma linier, untuk menemukan solusi yang baik yang mendekati solusi optimal dalam waktu yang wajar. Algoritma ini dapat mengeksploitasi struktur dan properti masalah untuk menemukan solusi yang baik tanpa harus mencari semua skenario permintaan yang ada.

Selain dua pendekatan di atas juga dapat digunakan pendekatan yang menyederhanakan asumsi atau relaksasi dari masalah asli, seperti mengasumsikan independensi kejadian permintaan atau hanya mempertimbangkan sebagian skenario permintaan. Meskipun asumsi-asumsi yang disederhanakan pada pendekatan terakhir ini mungkin tidak dapat menangkap seluruh kompleksitas masalah, asumsi-asumsi ini dapat memberikan kerangka kerja yang mudah digunakan untuk menemukan solusi yang baik. Pendekatan yang terakhir ini lah yang akan digunakan pada makalah ini.



Gambar. 1 Klaster pengecer yang lokasinya terpisah cukup jauh dari depot merupakan salah satu ciri khas permasalahan distribusi pada daerah pedesaan.

II. METODOLOGI DAN PERANCANGAN ALGORITMA SOLUSI PVRPSD

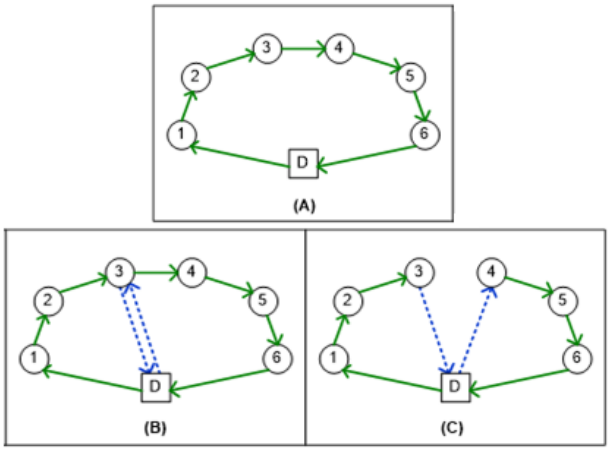
A. *Deskripsi Permasalahan*

Permasalahan yang diangkat dalam makalah ini adalah permasalahan distribusi barang pada tahap akhir atau *last mile delivery* di daerah pedalaman. Terdapat hanya satu buah kendaraan truk dengan kapasitas Q yang harus mendistribusikan barang ke suatu klaster pengecer yang lokasinya terpisah cukup jauh jaraknya dari depot atau pusat distribusinya seperti terlihat pada Gambar. 1. Situasi seperti ini adalah situasi tipikal yang dihadapi oleh suatu pusat distribusi barang di daerah pedalaman dengan kepadatan penduduk rendah. Umumnya pusat distribusi berada di ibu kota provinsi atau kabupaten, sedangkan klaster-klaster pengecer yang harus dilayani berada di desa-desa di kabupaten/kecamatan yang berjarak cukup jauh serta tersebar pada suatu area geografis yang cukup luas.

Karena truk yang dialokasikan oleh perusahaan hanya satu untuk melayani satu klaster pengecer, maka diperlukan horison perencanaan $T = \{1, 2, \dots, l\}$ yang terdiri atas l hari dengan waktu kerja sebesar h jam/hari untuk memenuhi seluruh *demand* pengecer dari klaster tersebut.

Dalam kasus ini, kejadian besaran *demand* dari setiap pengecer bersifat acak dengan hanya dua jenis kejadian acak yaitu: kejadian *demand* sejumlah tertentu barang dan kejadian *demand* sejumlah nol barang. Peluang kejadian *demand* D dari pengecer i sejumlah k adalah sebesar $\Pr\{D_i = k\} = p_i(k)$; sedangkan $\Pr\{D_i = 0\} = 1 - p_i(k)$. Jumlah permintaan terbesar dalam satu periode yang mungkin dari suatu pengecer pada klaster tersebut adalah sebesar K, dimana $K < Q$. Jumlah *demand* yang sesungguhnya dari suatu pengecer hanya dapat diketahui setelah truk benar-benar tiba di lokasi pengecer tersebut.

Dengan *demand* yang bersifat acak ini akan dicari rute-rute keliling (*tour*) yaitu τ dari truk distribusi selama T yang memberikan total ekspektasi waktu distribusi $E[L_\tau^T]$ paling minimal. Untuk itu mula-mula ditentukan terlebih dahulu rute-rute keliling apriori untuk truk setiap harinya misalnya $\tau = (0, 1, \dots, n, n+1 \triangleq 0)$ yaitu mulai dari depot $(i = 0)$ lalu ke pengecer pertama $(i = 1)$ dan seterusnya hingga pengecer ke-n lalu kembali lagi ke depot.



Gambar. 2 Contoh *a priori route* dengan panah warna hijau di gambar (A), serta dua alternatif *classic recourse* ketika terjadi *route failure* ditandai dengan tambahan panah warna biru: kejadian *normal-stockout recourse* pada gambar (B) dan *exact stockout recourse* pada gambar (C).

Ketika truk melaksanakan aktivitas distribusi harian mengikuti rute keliling (*tour*) τ untuk melayani permintaan pengecer yang bersifat probabilistik ini, maka akan terjadi dua kejadian berikut: pertama, truk sukses menjalankan tugasnya memenuhi seluruh permintaan pengecer pada rute *tour* tersebut; kedua, truk mengalami *route failure* yaitu kejadian dimana truk kehabisan barang sebelum berhasil menyelesaikan rute *tour* τ. Ketika terjadi *route failure*, maka truk harus melakukan *recourse* yaitu perjalanan kembali ke depot untuk mengisi ulang muatan kendaraan sebelum melanjutkan aktivitas distribusi hingga seluruh permintaan pengecer pada rute *tour* τ terpenuhi (lihat Gambar. 2).

Peluang kejadian γ_i adalah peluang kejadian *exact stockout* dimana truk kehabisan barang bawaan persis setelah memenuhi permintaan pengecer i sehingga harus kembali ke depot untuk mengisi ulang muatan sebelum dapat melanjutkan aktivitas distribusi ke pengecer $i + 1$ dan seluruh *demand* pengecer pada rute τ selesai dilayani. Sedangkan peluang kejadian δ_i adalah peluang kejadian *normal stockout* dimana sisa barang bawaan truk tidak cukup untuk memenuhi permintaan pengecer i sehingga setelah mengisi ulang muatan di depot, truk harus kembali lagi ke pengecer i untuk melengkapi kekurangan pasokan sebelum kemudian melanjutkan aktivitas distribusi hingga *demand* seluruh pengecer pada rute τ selesai dilayani.

Dalam permasalahan ini, karena jarak depot ke klaster pengecer cukup jauh, maka dengan pertimbangan batas jam kerja/hari maka diasumsikan dalam satu hari, truk mungkin hanya boleh mengalami satu hingga maksimum dua kali *route failure* atau mungkin hanya mempunyai satu atau dua kali kesempatan untuk melakukan *recourse* kembali ke depot untuk mengisi ulang muatan.

Dengan demikian solusi atau fungsi tujuan yang ingin diperoleh dari pemecahan permasalahan ini adalah suatu set jadwal distribusi berupa *tour* harian kendaraan ke sejumlah pengecer selama periode waktu *T* yang memberikan ekspektasi total waktu distribusi paling minimum.

B. Pendekatan dan Metode yang Digunakan

Permasalahan *vehicle routing problem* yang merupakan permasalahan optimasi kombinatorial dapat dikategorikan ke dalam jenis NP-*hard* yang berarti tidak tersedia metode eksak dengan waktu komputasi yang polinomial untuk memperoleh solusi optimal [11]. Dengan sendirinya permasalah PVRPSD yang akan dipecahkan dalam penelitian ini juga masuk dalam kategori permasalahan NP-*hard* dimana solusi optimal untuk permasalahan yang berukuran cukup besar tidak mudah untuk diperoleh dalam waktu komputasi yang singkat. Dengan demikian dalam penelitian ini akan digunakan pendekatan heuristik dalam menemukan solusi bagi permasalahan PVRPSD.

Metode heuristik yang akan dikembangkan bagi permasalahan PVRPSD ini adalah dengan mengadopsi pendekatan *a priori paradigm*. Pada pendekatan ini, solusi atau jadwal distribusi yang disusun sebelumnya *(a priori planning*) akan mengalami penyesuaian (*recourse*) sepanjang waktu akibat terjadinya *route failure* [6]. Pada penelitian ini diadopsi model perhitungan ekspektasi jarak rute *tour* apriori klasik yang dikembangkan oleh Bertsimas [12] yang dimodifikasi menjadi ekspektasi waktu distribusi sebagai fungsi tujuan yang akan diminimasi dan menjadi ukuran kinerja dari metode solusi yang diusulkan.

C. Prosedur Perhitungan Ekspektasi Waktu Distribusi

Total ekspektasi waktu yang diperlukan untuk distribusi selama periode *T* dihitung menggunakan formulasi pada persamaan (1) berikut:

$$\boldsymbol{E[L_\tau^T] = \sum_{p=1}^{l} E\left[L_\tau^p\right]} \ldots (1)$$

$$\boldsymbol{E\left[L_\tau^p\right] = \sum_{i=0}^{n} d(i,i+1) + t_0 + \sum_{i=1}^{n}\left\{\delta_i\left[s(i,i) + t_{\delta_0} + t\left(\omega_{\delta_i}\right)\right] + \gamma_i\left[s(i,i+1) + t_{\gamma_0} + t\left(\omega_{\gamma_{i+1}}\right)\right]\right\}} \ldots (2)$$

Dimana:

$$\boldsymbol{s(i,j) = d(i,0) + d(0,j) - d(i,j)} \ldots (3)$$

dan

$$\boldsymbol{\gamma_i} = \begin{cases} \sum_{q=1}^{\left|\frac{iK}{Q}\right|}\left\{\sum_{k=1}^{K} p_i(k) f(i-1, qQ-k)\right\}, & \text{untuk } 2 \le i \le n \\ 0, & \text{untuk selainnya} \end{cases} \ldots (4)$$

$$\boldsymbol{\delta_i} = \begin{cases} \sum_{q=1}^{\left|\frac{iK}{Q}\right|}\left\{\sum_{k=1}^{K-1}\left(\sum_{r=k+1}^{K} p_i(r)\right) f(i-1, qQ-k)\right\}, & \text{untuk } 2 \le i \le n \\ 0, & \text{untuk selainnya} \end{cases} \ldots (5)$$

Sedangkan peluang total demand pengecer $\boldsymbol{1, \ldots, m}$ sebesar $\boldsymbol{r}$ yaitu $\boldsymbol{f(m,r) = \Pr\{\sum_{i=1}^{m} D_i = r\}}$ dapat dihitung dengan cara rekursif sebagai berikut:

$$\boldsymbol{f(m,r) = \sum_{k=0}^{\min[K,r]} p_m(k) f(m-1, r-k)}$$

untuk $\boldsymbol{m = 2, \ldots, n}$; dan $\boldsymbol{r = 0, \ldots, Km;}$ dengan kondisi awal sebagai berikut:

$\boldsymbol{f(m,r) = 0}$ untuk $\boldsymbol{r > Km}$

$$\boldsymbol{f(1,r)} = \begin{cases} \boldsymbol{p_i(r)}, & \text{untuk } \mathbf{0} \le \boldsymbol{r} \le \boldsymbol{K} \\ \mathbf{0}, & \text{untuk selainnya} \end{cases}$$

Persamaan (2) dapat dijelaskan sebagai berikut: suku pertama adalah waktu tempuh *tour* apriori, suku kedua adalah waktu memuat produk ke truk di depot, suku ketiga terdiri atas dua sub suku yang menghitung dua jenis ekspektasi waktu rute distribusi akibat terjadinya dua jenis *recourses.* Pada jenis *recourse* sub suku pertama, nilai ekspektasinya ditentukan oleh besar peluang terjadinya *normal-stockout* ketika truk berada di pengecer *i* yang disebut δ_i. Untuk jenis *recourse* sub suku kedua, nilai ekspektasinya ditentukan oleh γ_i yaitu peluang terjadinya *exact-stockout* tepat ketika truk selesai melayani demand pengecer i. Dalam tiap sub suku tersebut nilai ekspektasi waktu *tour* distribusinya dihitung atas komponen-komponen: waktu yang diperlukan untuk *recourse* kembali ke depot, waktu untuk pengisian ulang produk ke truk di depot saat *recourse*, dan waktu bongkar produk di setiap pengecer jika terjadi *recourse*. Dalam kejadian *normal-stockout recourse*, $t\left(\omega_{\delta_i}\right)$ adalah waktu bongkar dan pelayanan mulai dari pengecer pertama hingga pengecer *i* sesuai urutan *tour* apriori dan t_{δ_0} adalah waktu pengisian ulang muatan truk di depot. Sedangkan dalam kejadian *exact-stockout recourse*, $t\left(\omega_{\gamma_{i+1}}\right)$ adalah waktu bongkar dan pelayanan mulai dari pengecer pertama hingga pengecer $i+1$ sesuai urutan *tour* apriori dan t_{γ_0} adalah waktu pengisian ulang muatan truk di depot.

TABEL I
CONTOH SEBAGIAN PERMUTASI KEJADIAN DEMAND DARI SUATU APRIORI TOUR DENGAN LIMA PENGECER

No	Apriori *Tour*	5	9	13	2	8	Jenis peluang *recourse*
	k	25	50	75	50	50	
	$\Pr\{D_i = k\}$	0,339	0,889	0,73	0,445	0,602	
1	Permut_1	1	0	1	1	1	
	k	25	0	75	50	50	gamma
	$f(m,r)$	0,339	0,038	0,027	0,012	-	
2	Permut_2	0	1	1	1	1	
	k	0	50	75	50	50	delta
	$f(m,r)$	0,661	0,588	0,43	0,191	-	
3	Permut_3	1	0	1	1	0	Tidak
	k	25	0	75	50	0	terjadi
	$f(m,r)$	-	-	-	-	-	*recourse*
4	Permut_4	0	1	1	1	0	
	k	0	50	75	50	0	delta
	$f(m,r)$	0,661	0,588	0,43	0,19	-	(*redundant*)

Persamaan (3) menjelaskan cara menghitung waktu tempuh rute *recourse* yaitu misalnya untuk kejadian *exact-stockout recourse*: waktu tempuh rute dari pengecer *i* kembali ke depot ($i = 0$) lalu lanjut ke pengecer *j*.

Konsekuensi dengan adanya rute *exact-stockout recourse* $i - 0 - j$ tambahan ini maka rute *tour* apriori dari pengecer i ke j yaitu $d(i,j)$ harus dihapuskan karena telah tergantikan oleh rute $d(i,0) + d(0,j)$.

Pada paragraf-paragraf selanjutnya akan diberikan ilustrasi mengenai prinsip-prinsip yang digunakan pada proses perhitungan gamma (γ_i) dan delta (δ_i) seperti ditunjukkan oleh persamaan (4) dan (5), yang akan direalisasikan pada rancangan algoritma solusi.

Misalkan suatu *a priori tour* terdiri atas kunjungan ke lima pengecer dengan urutan berikut: $5 - 9 - 13 - 2 - 8$. Kapasitas truk $Q = 150$, *demand* (k) dari pengecer yang akan dikunjungi tersebut dan peluang kejadiaanya $(\Pr\{D_i = k\})$ dapat dilihat pada TABEL I. Maka untuk menghitung ekspektasi gamma atau delta mula-mula dibangkitkan permutasi biner $(0,1)$ dengan jumlah digit sesuai dengan jumlah pengecer pada *a priori tour* seperti yang ditunjukkan oleh tabel tersebut.

Urut-urutan digit yang ada pada permutasi-permutasi biner tersebut menunjukkan semua kemungkinan kejadian *demand* yang mungkin terjadi pada suatu *tour* apriori. Setiap digit '0' pada permutasi menunjukkan tidak ada kejadian *demand* pada pengecer yang berkesesuaian, sedangkan digit '1' pada permutasi menunjukkan terjadinya *demand* pada pengecer terkait. Sehingga dalam contoh permutasi Permut_1 pada TABEL Iterdapat kejadian *demand* sejumlah k pada pengecer 5, 13, 2, dan 8 dengan masing-masing peluang kejadian $\Pr\{D_i = p_i(k)\}$, sedangkan pada pengecer 9 tidak ada kejadian *demand* $(k = 0)$ maka peluang kejadiannya sebesar $\Pr\{D_i = 0\} = (1 - p_i(k))$.

Dengan kapasitas truk sebanyak 150, maka sesuai kejadian *demand* Permut_1 terjadi peristiwa *exact stockout* pada saat truk berada pada pengecer 2. Peristiwa ini mengharuskan dilakukannya *recourse* ke depot untuk mengisi ulang muatan truk sebelum melanjutkan memenuhi permintaan pengecer 8. Karena hanya terjadi peristiwa *exact stockout recourse* dengan demikian hanya peluang kejadian gamma yang dapat dihitung pada permutasi tersebut sebagai berikut:

$$\gamma_{permut1} = p_5(25) \times p_9(0) \times p_{13}(75) \times p_2(50)$$
$$= 0{,}339 \times 0{,}111 \times 0{,}73 \times 0{,}445 = 0{,}012;$$

Sedangkan peluang kejadian delta adalah nol ($\delta_{permut1} = 0$) karena Permut_1 tersebut mewakili kejadian *demand* dimana tidak terjadi *normal stockout recourse*.

Untuk kejadian *demand* seperti yang ditunjukkan oleh Permut_2 pada TABEL I, dapat dilihat bahwa yang terjadi pada kejadian *demand* ini adalah *normal stockout recourse* pada pengecer 2 sehingga dihitung peluang kejadian delta tersebut adalah sebesar $\delta_{permut2} = 0{,}191$, sedangkan peluang kejadian gamma adalah nol ($\gamma_{permut2} = 0$).

Permut_3 pada TABEL I mewakili kejadian *demand* dimana tidak terjadi *stockout recourse* sama sekali sehingga peluang kejadian gamma maupun deltanya adalah nol. Permutasi yang seperti ini mewakili kejadian dimana jumlah *demand* yang terjadi persis sama dengan kapasitas truk.

ALGORITMA. 1
PSEUDOCODE MENGHITUNG EKSPEKTASI GAMMA

```
1  Fungsi calculate_gamma_expectation():
2      Input: tourApriori; demand dan peluangnya untuk
       setiap pengecer; threshold;
3      totalGammaExpectation = 0
4      Bangkitkan permutasiBiner = permutasi sejumlah
       angka biner yang mewakili seluruh kombinasi kejadian
       demand yang mungkin
5      set permutasiBiner = permutasiBiner with
       kejadian total demand tourApriori > threshold
6      for setiap permutasi dalam permutasiBiner:
7          set kapasitasTruk = threshold
8          for setiap digit '1' dalam permutasi:
9              kurangi kapasitasTruk sesuai dengan besar demand
               dari pengecer terkait
10             if kapasitasTruk tersisa == 0:
11                 if ini adalah demand dari pengecer urutan terakhir pada
                   tourApriori:
12                     continue ke permutasi yang lain
13                 else:
14                     exact_stockout = True
15             if kapasitasTruk tersisa ≤ 0:
16                 if masih ada digit '1' selanjutnya dalam permutasi:
17                     Pulihkan kapasitasTruk ke nilai threshold
18             if exact_stockout == True:
19                 Hitung a = peluang kejadian demand total
                   permutasi ini
20                 Hitung b = waktu tempuh rute exact stockout recourse
21                 Hitung c = ekspektasi waktu bongkar dan pelayanan
                   seluruh pengecer hingga terjadi exact stockout
22                 Hitung ekspekGammaPermut_i = ( a * b) + c
23                 Simpan hasil jika ini adalah peristiwa gamma yang baru
                   (redundancy avoidance)
24                 Pulihkan kapasitasTruk ke nilai threshold
25         Tambahkan ekspekGammaPermut_i ke
           totalGammaExpectation
26     return totalGammaExpectation
```

ALGORITMA. 2
PSEUDOCODE MENGHITUNG EKSPEKTASI DELTA

```
1  Fungsi calculate_delta_expectation():
2      Input: tourApriori; demand dan peluangnya untuk
       setiap pengecer; threshold;
3      totalDeltaExpectation = 0
4      Bangkitkan permutasiBiner = permutasi sejumlah
       angka biner yang mewakili seluruh kombinasi kejadian
       demand yang mungkin
5      set permutasiBiner = permutasiBiner with
       kejadian total demand tourApriori > threshold
6      for setiap permutasi dalam permutasiBiner:
7          set kapasitasTruk = threshold
8          for setiap digit '1' dalam permutasi:
9              kurangi kapasitasTruk sesuai dengan besar
               demand dari pengecer terkait
10             if kapasitasTruk tersisa > 0 and
                   kapasitasTruk tersisa < demand dari
                   pengecer terkait:
11                 exact_stockout = True
12             elif kapasitasTruk tersisa == 0:
13                 if ini bukan demand pengecer urutan terakhir pada
                   tourApriori:
14                     Pulihkan kapasitasTruk ke nilai
                       threshold
15             else:
```

```
kurangi kapasitasTruk sesuai dengan besar demand dari pengecer terkait
if exact_stockout == True:
    Hitung a = peluang kejadian demand permutasi ini
    Hitung b = waktu tempuh rute normal stockout recourse
    Hitung c = ekspektasi waktu bongkar dan pelayanan seluruh pengecer hingga terjadi exact stockout
    Hitung ekspekDeltaPermut_i = ( a * b) + c
    Simpan hasil jika ini adalah peristiwa delta yang baru (redundancy avoidance)
    if ini bukan demand dari pengecer urutan terakhir pada tourApriori:
        Pulihkan kapasitasTruk = kapasitasTruk tersisa + threshold - demand dari pengecer terkait
    Tambahkan ekspekDeltaPermut_i ke totalDeltaExpectation
return totalDeltaExpectation
```

Perlu pula untuk diperhatikan bahwa dalam algoritma heuristik yang dikembangkan ini, dapat terjadi redundansi pada perhitungan peluang gamma ataupun delta. Contohnya pada perhitungan peluang delta di Permut_2 yang berulang pada perhitungan delta di Permut_4. Meskipun kedua permutasi tersebut adalah dua permutasi yang berbeda, namun keduanya mewakili satu kejadian yang sama yaitu kejadian *normal-stockout* pada pengecer 2. Dengan demikian dalam perhitungan ekspektasi gamma dan delta ini dilakukan penyaringan perhitungan yang berulang (*redundancy avoidance*) agar perhitungan ekspektasinya tidak berlebih.

Untuk mendapatkan semua kemungkinan kejadian *demand* secara lengkap, maka pada suatu *tour* apriori berukuran 5 pengecer harus dibangkitkan sebanyak $2^5 = 32$ buah permutasi biner untuk dianalisa dan dihitung total peluang kejadian gamma dan/atau deltanya.

Untuk menghitung gamma (γ_i) dan delta (δ_i) seperti ditunjukkan oleh persamaan (4) dan (5), dikembangkan *pseudo code* seperti yang dapat dilihat pada ALGORITMA. **1** dan

ALGORITMA. **2**.

D. Perancangan Algoritma Solusi

Secara garis besar algoritma solusi yang dikembangkan mengikuti prinsip dua tahap yaitu *tour construction phase* dan *tour improvement phase* yang sangat populer dan masih terus digunakan diberbagai riset VRP hingga kini [16]. Kedua tahapan tersebut dapat diuraikan sebagai berikut:

ALGORITMA. 3
PSEUDOCODE TOUR CONSTRUCTION

```
inisialisasi: set: tourOpti_τ = [ ]
while masih ada pengecer yang belum masuk ke tourOpti_τ do:
    Buat sebuah tourApriori_j berisi acak 4 s/d 6 pengecer
    Hitung E[L_τj]
    if E[L_τj] < waktuKerja yang tersedia:
        Tambahkan satu pengecer acak ke tourApriori_j
        Hitung E[L_τj]
    else:
        Kurangi satu pengecer acak dari tourApriori_j
        Hitung E[L_τj]
    Tambahkan tourApriori_j ke tourOpti_τ
Periksa status solusi
if status == 'feasibles solution':
    Hitung E[L_τ] = Σ_∀j E[L_τj]
else:
    Hitung E[L_τ]~ = Σ_∀j E[L_τj] + αn_ot + βl
```

1) Tour Construction Phase: Tahap ini bertujuan untuk mengkonstruksi suatu *tour* distribusi sebagai sebuah *initial solution* (bibit solusi awal) bagi permasalahan (lihat ALGORITMA. **3**). Proses konstruksi dilakukan dengan menggunakan metode yang sangat sederhana dan hemat sumber daya komputasi, yaitu metode *cluster-first-route-second* (CFRS) [17]. Dengan demikian, langkah pertama CFRS adalah mengelompokkan pengecer ke dalam beberapa klaster kecil yang terdiri dari 4, 5, atau 6 pengecer. Selanjutnya, untuk setiap klaster tersebut dibentuk sebuah *tour* distribusi dengan urutan kunjungan yang acak ke setiap pengecer yang menjadi anggota dari klaster tersebut. *Tour* berukuran kecil ini dipilih dengan pertimbangan ketersediaan waktu kerja per hari dan kapasitas kendaraan yang digunakan. Karena kendaraan yang digunakan untuk operasi distribusi hanya satu unit, maka jumlah *tour* distribusi yang dihasilkan pada tahap konstruksi ini mencerminkan jumlah hari distribusi selama satu periode T. Setiap *tour* kemudian dihitung ekspektasi waktu distribusinya sehingga diperoleh ekspektasi total waktu distribusi sepanjang periode T. Pada tahap awal, algoritma ini memperbolehkan solusi yang dihasilkannya untuk bersifat tidak layak (*infeasible*) sebagai bagian dari proses perturbasi. Artinya, solusi yang melanggar batas jam kerja pada satu atau lebih *tour* distribusi masih diperbolehkan, selama waktu total distribusi yang dihasilkan tetap menjadi yang paling pendek. Sejumlah *tour* yang tidak memenuhi kriteria ini (n_{ot}) kemudian diberikan penalti berupa tambahan waktu yang cukup signifikan sebesar α agar *tour* tak layak ini tereliminasi pada tahap perbaikan solusi pada proses selanjutnya. Jumlah hari atau *tour* distribusi yang terbentuk sepanjang periode T pada tahap konstruksi ini bersifat acak. Namun, tentu saja, solusi dengan jumlah hari atau *tour* yang lebih sedikit dalam suatu periode T lebih diinginkan untuk dipilih. Oleh karena itu setiap hari atau *tour* yang terbentuk yaitu l dalam suatu periode T akan dikenai konsekuensi berupa tambahan bobot waktu (*supplementary weighting*) dengan besaran yang cukup signifikan yaitu sebesar β, sehingga pada akhirnya solusi dengan jumlah hari distribusi terpendeklah yang akan terpilih. Karena jumlah hari atau *tour* distribusi yang terbentuk sepanjang periode T hanya dapat dibentuk pada fase konstruksi ini, maka penting untuk mengkonstruksi sejumlah bibit solusi awal (*seed solutions*) yang bervariasi selama fase konstruksi untuk kesuksesan dalam mencari solusi berkualitas selama fase perbaikan. Namun,

pembangkitan terlalu banyak bibit solusi dapat memperlambat pencarian solusi karena memperbesar ukuran tetangga solusi yang harus diperiksa pada fase perbaikan [18].

ALGORITMA. 4
PSEUDOCODE TOUR IMPROVEMENT

```
1  input: tourOpti; status
2  while iterasi < iter_max do:
3      Bangkitkan peluang Pr{e_a}
4      if Pr{e_a} < P_1: #exploration strategy
5          Bangkitkan Pr{e_b}
6          if Pr{e_b} < P_2: #inter-tour circular exchange
7              Periksa status solusi;
               if status == 'feasibles solution': Hitung E[L_τ];
               else: Hitung E[L_τ]~
8              tourOpti_τ* ← Lakukan
               swap_inter_tour(tourOpti_τ)
9              E[L_τ]*← Hitung total ekspektasi waktu
               tourOpti_τ*
10             if E[L_τ]*< E[L_τ] or E[L_τ]*<
               E[L_τ]~ :tourOpti_τ ← tourOpti_τ*
11             Periksa status solusi;
               if status == 'feasibles solution': Hitung E[L_τ];
               else: Hitung E[L_τ]~
12         else: #overtime cut-off
13             Periksa status solusi;
               if status == 'feasibles solution': Hitung E[L_τ];
               else: Hitung E[L_τ]~
14             tourApriOver ← find semua tourApriori_j
               pada tourOpti_τ
               with E[L_τj] < waktuKerja
15             tourApriMin ← find n tourApriori_j pada
               tourOpti_τ with minimum E[L_τj]
16             tourOpti_τ* ← Ambil satu pengecer acak dari
               setiap tourApriOver sisipkan acak ke
               tourApriMin
17             E[L_τ]*← Hitung total ekspektasi waktu
               tourOpti_τ*
18             if E[L_τ]*< E[L_τ] or E[L_τ]*<
               E[L_τ]~ :tourOpti_τ ← tourOpti_τ*
19             Periksa status solusi;
               if status == 'feasibles solution': Hitung E[L_τ];
               else: Hitung E[L_τ]~
20     else: #exploitation strategy: intra-tour swap
21         Periksa status solusi;
           if status == 'feasibles solution': Hitung E[L_τ]; else:
           Hitung E[L_τ]~
22         tourOpti_τ* ← Lakukan
           swap_intra_tour(tourOpti_τ)
23         E[L_τ]*← Hitung total ekspektasi waktu
           tourOpti_τ*
24         if E[L_τ]*< E[L_τ] or E[L_τ]*<
           E[L_τ]~ :tourOpti_τ ← tourOpti_τ*
25         Periksa status solusi;
           if status == 'feasibles solution': Hitung E[L_τ]; else:
           Hitung E[L_τ]~
```

2) Tour Improvement Phase: Pada tahapan ini akan dicari solusi-solusi yang merupakan tetangga dari solusi awal yang mungkin memiliki kualitas kinerja yang lebih baik dari solusi sementara saat ini yaitu memiliki ekspektasi total waktu distribusi yang lebih rendah [19]. Proses perbaikan kualitas solusi dengan melakukan pencarian solusi-solusi tetangga (*move*) ini dilakukan dengan menggunakan strategi eksploitatif dan strategi eksploratif yang dapat dilihat pada ALGORITMA. **4**.

Terdapat dua metode dalam strategi eksploratif yang menentukan struktur dari tetangga solusi pada tahap perbaikan solusi yaitu: *inter-tour circular exchange* (ITCE) dan *overtime cut-off* (OTCO). Metode ITCE membangun tetangga solusi dengan cara memodifikasi solusi saat ini yaitu dengan memindahkan satu pengecer acak dari *tour* pertama ke *tour* kedua, begitu juga *tour* kedua memindahkan satu pengecer acak ke *tour* ketiga, dan seterusnya secara hingga *tour* terakhir memindahkan satu pengecer acak ke *tour* pertama (estafet melingkar). Sedangkan metode OTCO bekerja membangun tetangga dari solusi saat ini dengan cara memindahkan satu atau lebih pengecer dari setiap *tour* yang memiliki nilai ekspektasi waktu distribusinya melebihi batas waktu kerja yang tersedia ke sejumlah *tour* lain yang memiliki ekspektasi waktu distribusi terendah. Adapun strategi eksploitatif yang digunakan disini adalah *intra-tour swap* yang bekerja membangun tetangga solusi dengan cara mempertukarkan urutan pengecer dalam satu *tour*.

Setiap tetangga solusi yang ditemukan kemudian dievaluasi kinerjanya dengan menghitung ekspektasi total waktu distribusinya. Dalam setiap iterasi, tetangga solusi yang memiliki kinerja lebih baik dari solusi sebelumnya akan disimpan sebagai solusi terbaik untuk sementara hingga tercapai batas iterasi maksimum dari proses pencarian. Jika pada algoritma *tour construction phase* diperbolehkan untuk menerima solusi tidak layak untuk sementara waktu, maka pada algoritma *tour improvement phase* ini, solusi tidak layak hanya boleh diterima jika belum ditemukan solusi layak dan jika nilai ekspektasi total waktu distribusinya lebih rendah dari solusi tidak layak sebelumnya. Begitu ditemukan suatu solusi layak, maka algoritma *tour improvement* ini tidak akan mengizinkan lagi untuk menerima solusi tidak layak dan hanya mengizinkan untuk menerima solusi layak berikutnya yang memiliki kinerja lebih baik.

Karena kelayakan solusi ditentukan oleh ada atau tidaknya batas waktu kerja yang dilanggar pada setiap *tour* distribusi selama periode T, maka strategi OTCO pada algoritma *tour improvement* perlu untuk selalu dijalankan guna memperoleh tetangga-tetangga solusi yang layak. Dalam tahap perbaikan kualitas solusi ini, pilihan untuk menjalankan strategi eksploratif atau eksploitatif dalam suatu iterasi ditentukan oleh suatu peluang yang dibangkitkan secara acak yaitu $\Pr\{e_a\}$. Jika $\Pr\{e_a\} < P_1$ tertentu, maka pilihan strategi yang harus dijalankan adalah strategi eksploratif. Kemudian pilihan jenis strategi eksploratif apa yang akan dijalankan dalam suatu iterasi ditentukan oleh peluang acak $\Pr\{e_b\}$. Jika $\Pr\{e_b\} < P_2$ tertentu, maka strategi ITCE yang harus dieksekusi dan jika $\Pr\{e_b\} > P_2$, maka strategi OTCO yang akan terpilih untuk dijalankan.

III. Pengujian Kinerja Algoritma Pencarian Solusi PVRPSD

Untuk menguji kinerja algoritma solusi yang telah dikembangkan maka dilakukan eksperimen pengujian yang akan diuraikan pada sub-sub bab berikut.

TABEL II
PENGECER KE-i, DEMAND k DAN PELUANG DEMAND UNTUK PENGECER TERSEBUT $\Pr\{D_i = k\}$

i	k	$\Pr\{D_i = k\}$	i	k	$\Pr\{D_i = k\}$
1	50	0,65	26	100	0,659
2	50	0,445	27	75	0,471
3	50	0,789	28	50	0,806
4	75	0,597	29	25	0,549
5	25	0,339	30	75	0,671
6	25	0,627	31	75	0,344
7	25	0,494	32	50	0,808
8	50	0,602	33	100	0,647
9	50	0,111	34	100	0,616
10	100	0,771	35	50	0,445
11	25	0,623	36	25	0,641
12	50	0,676	37	25	0,536
13	75	0,73	38	50	0,553
14	75	0,629	39	100	0,4
15	100	0,656	40	50	0,422
16	50	0,753	41	25	0,676
17	75	0,591	42	25	0,569
18	75	0,695	43	75	0,698
19	100	0,549	44	25	0,833
20	75	0,471	45	50	0,44
21	25	0,583	46	75	0,583
22	25	0,687	47	50	0,677
23	100	0,491	48	75	0,443
24	100	0,984	49	50	0,774
25	50	0,522	50	100	0,548

A. *Instance Dasar*

Instance dasar pada eksperimen ini terdiri atas depot tunggal dan sejumlah pelanggan berupa toko-toko pengecer, *demand*, dan peluang *demand* yang dapat dilihat pada Tabel 2. Lokasi toko-toko pengecer ini secara geografis saling berdekatan sedangkan terdapat jarak yang signifikan jauhnya antara lokasi depot dengan lokasi klaster toko-toko pengecer ini seperti yang dapat dilihat pada Gambar 1.

Sebuah truk dialokasikan untuk operasi distribusi ini dengan Q = 150 unit barang. Waktu kerja yang tersedia adalah 480 menit/hari, waktu memuat produk di depot, membongkar produk dan pelayanan di pengecer bersifat acak dan berdistribusi normal dengan $(\mu = 15; \sigma = 5)$ menit. Nilai penalti yang digunakan adalah sebesar α =300 menit dan nilai *supplementary weighting* $\beta = 500$ menit.

B. *Rancangan Eksperimen*

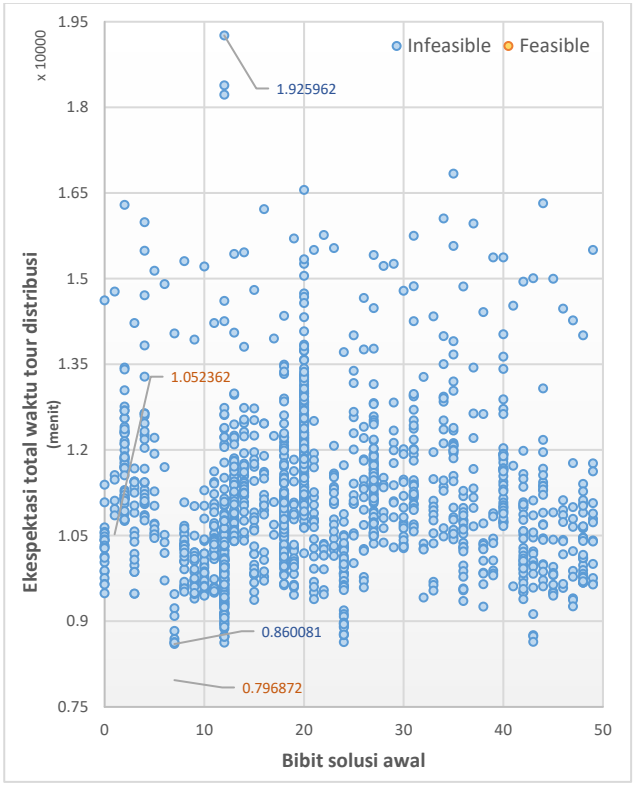
Eksperimen dilakukan dengan menggunakan tiga faktor utama yang diduga berpengaruh pada kinerja algoritma solusi. Pertama, kombinasi antara jumlah iterasi pencarian dan jumlah bibit solusi awal yang dibangkitkan, disebut faktor A; kedua, rasio peluang untuk melakukan pencarian menggunakan strategi eksploratif atau eksploitatif sebagai faktor B; dan ketiga, rasio peluang untuk menggunakan metode pencarian ITCE atau OTCO yang disebut faktor C.

Untuk faktor A, diujikan tiga level kombinasi jumlah [solusi awal:iterasi] yang digunakan dalam pencarian solusi yaitu: [100:100], [50:500], dan [10:1000]. Level-level pada faktor A ini disusun sedemikian dengan tujuan untuk mengungkapkan manakah yang akan lebih berpotensi dalam memberikan solusi terbaik yaitu: jumlah initial solution yang lebih banyak, ataukah jumlah iterasi pencarian. Faktor B disusun menggunakan tiga level rasio peluang untuk memilih antara dua strategi pencarian [eksploratif:eksploitatif] yaitu: [30%:70%], [50%:50%] dan [70%:30%]. Sedangkan pada faktor C digunakan pula tiga level rasio peluang untuk memilih antara dua metode pencarian [ITCE:OTCO] yaitu: [30%:70%], [50%:50%] dan [70%:30%]. Total blok yang digunakan dalam eksperimen pengujian kinerja algoritma solusi ini adalah 27 blok eksperimen.

Pada eksperimen ini digunakan dua ukuran kinerja untuk algoritma pencarian solusi PVRPSD yang telah dikembangkan yaitu ukuran kinerja utama dan pendukungnya. Ukuran kinerja utama adalah nilai ekspektasi total waktu tour distribusi terendah yang berhasil diperoleh pada setiap blok eksperimen. Sedangkan ukuran kinerja pendukung adalah durasi waktu komputasi yang diperlukan untuk memperoleh solusi tersebut.

TABEL III
HASIL PENGUJIAN DENGAN MENGGUNAKAN 27 BLOK EKSPERIMEN

Blok	Faktor A [InitSol: Iteration]	Faktor B [explor: exploit]	Faktor C [ITCE: OTCO]	Ekspektasi total waktu tour distribusi (menit)	Durasi waktu komputasi (detik)	Total kinerja
1			[30%:70%]	8032,89	3094,57	
2		[30%:70%]	[50%:50%]	8387,67	3012,01	
3			[70%:30%]	8363,69	2932,35	
4			[30%:70%]	8258,53	2278,15	
5	[100:100]	[50%:50%]	[50%:50%]	8748,42	2166,51	
6			[70%:30%]	8418,29	2185,01	
7			[30%:70%]	8066,15	1305,96	
8		[70%:30%]	[50%:50%]	8269,89	1327,38	
9			[70%:30%]	8538,32	1324,12	
10			[30%:70%]	8378,65	7844,72	
11		[30%:70%]	[50%:50%]	7968,72	7730,82	
12			[70%:30%]	8205,89	8171,17	
13			[30%:70%]	8285,46	5186,99	
14	[50:500]	[50%:50%]	[50%:50%]	8119,01	5727,51	
15			[70%:30%]	8053,42	5239,22	
16			[30%:70%]	8500,16	3539,62	
17		[70%:30%]	[50%:50%]	7991,84	3253,77	
18			[70%:30%]	8216,95	3387,09	
19			[30%:70%]	8718,53	2614,67	
20		[30%:70%]	[50%:50%]	8646,42	3031,89	
21			[70%:30%]	8783,94	3253,71	
22			[30%:70%]	8459,44	2430,19	
23	[10:1000]	[50%:50%]	[50%:50%]	8885,27	2375,24	
24			[70%:30%]	8697,28	2027,68	
25			[30%:70%]	8825,14	1257,94	
26		[70%:30%]	[50%:50%]	8692,39	1164,22	
27			[70%:30%]	8201,07	1208,91	



Gambar. 3 Solusi hasil pencarian dengan konfigurasi level faktor A [50:500], level faktor B [30%-70%], dan level faktor C [50%:50%].

IV. Hasil dan Analisis

A. *Hasil Kinerja Algoritma Secara Umum*

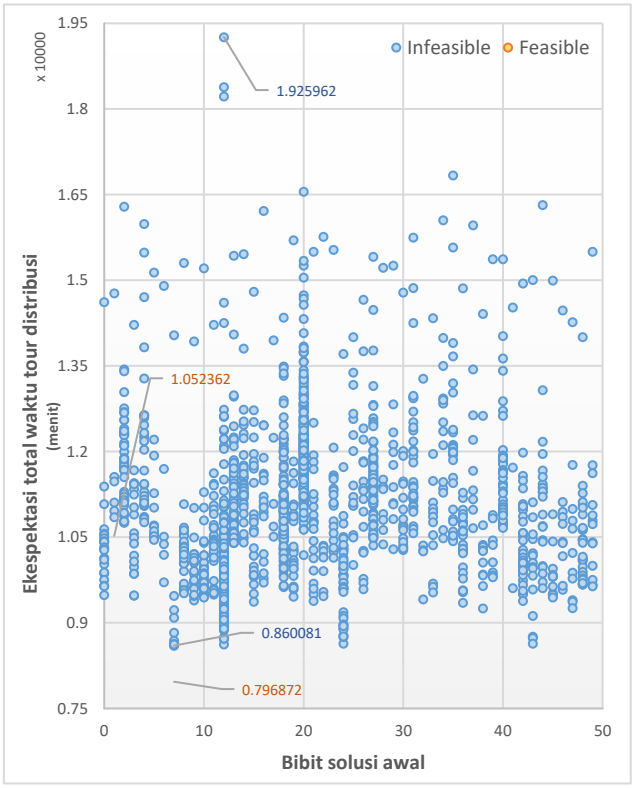
Berdasarkan ukuran kinerja utama dari algoritma pencarian solusi yang telah dirancang ini, maka solusi terbaik berupa nilai ekspektasi total waktu *tour* distribusi terendah sebesar 7968,72 menit berhasil dicapai oleh blok eksperimen nomor 11 dengan level faktor A [50:500], level faktor B [70%-30%], dan level faktor C [50%:50%] (lihat TABEL III). Hasil pencarian solusi pada blok eksperimen nomor 11 ini secara lengkap dapat dilihat pada Gambar. 3.

Proses pencarian pada blok eksperimen nomor 11 telah berhasil menemukan total sebanyak 1878 buah solusi dengan 379 solusi diantaranya adalah solusi *feasible* (titik-titik yang berwarna kuning). Dapat diamati pula pada gambar tersebut bahwa setiap bibit solusi awal yang digunakan menghasilkan jumlah solusi yang tidak sama dan tidak selalu menghasilkan solusi yang *feasible*. Misalnya pada bibit solusi awal ke-12 dan ke-20, tidak satupun solusi *feasible* yang berhasil ditemukan (semua titik-titik solusi berwarna biru). Selain itu satu solusi *feasible* (jika berhasil ditemukan ada) hanya akan selalu diikuti oleh solusi *feasible* berikutnya dengan nilai ekspektasi yang lebih rendah. Hal ini sesuai aturan algoritma perbaikan solusi yang telah dirancang.

TABEL IV
JADWAL LENGKAP TOUR DISTRIBUSI SOLUSI TERBAIK

T	*Tour*	Ekspektasi waktu tour distribusi	*Supple-mentary weighting*	Total
1	[50, 33, 44, 7]	313,94	500	813,94
2	[30, 13, 4, 37, 3]	395,18	500	895,18
3	[16, 20, 9, 29]	241,33	500	741,33
4	[31, 17, 39, 18, 32]	476,27	500	976,27
5	[5, 26, 27, 11, 21]	367,65	500	867,65
6	[48, 42, 19, 43]	475,92	500	975,92
7	[38, 10, 8, 35, 36]	396,1	500	896,1
8	[1, 24, 25, 46]	462,65	500	962,65
9	[15, 40, 47, 22, 41]	339,68	500	839,68
	Total	**3468,72**	**4500**	**7968,72**

Pada



Gambar. 3 Solusi hasil pencarian dengan konfigurasi level faktor A [50:500], level faktor B [30%-70%], dan level faktor C [50%:50%].

dapat dilihat jadwal lengkap *tour* distribusi dari solusi terbaik ini. Periode distribusi terpendek yang dapat dibuat adalah sepanjang $T = 9$ hari kerja dengan jadwal *tour* distribusi spesifik untuk setiap harinya. Meskipun nilai solusi terbaik yang diperoleh adalah sebesar 7968,72 menit, namun nilai ekspektasi total waktu *tour* distribusi yang sesungguhnya hanyalah sebesar 3468,72 menit. Nilai sesungguhnya ini diperoleh setelah nilai solusi terbaik yang ditemukan dikurangi dengan total *supplementary weighting*. Nilai *supplementary weighting* adalah bobot waktu yang ditambahkan untuk setiap hari yang ada dalam periode *T* pada fase konstruksi solusi awal, agar proses pencarian solusi yang dilakukan oleh algoritma dapat bekerja memilih solusi dengan *T* terpendek.

Jika diamati pada TABEL III, ukuran hasil kinerja utama blok eksperimen nomor 11 yang memberikan nilai ekspektasi terendah ini tidak diikuti ukuran kinerja pendukung yang memuaskan karena hasil yang diperoleh tersebut memerlukan durasi waktu komputasi nomor dua terpanjang dari seluruh blok eksperimen yaitu sebesar 7730,82 detik.

Jika ukuran kinerja pendukung dianggap cukup penting, karena waktu komputasi yang terlalu panjang mungkin tidak sesuai dengan tuntutan pembuatan keputusan operasional yang harus serba cepat, maka ukuran kinerja utama dan ukuran kinerja pendukung dapat diagregasikan ke dalam suatu skor generik total kinerja seperti yang ditunjukkan oleh kolom total kinerja pada Tabel 3. Dengan demikian kriteria kinerja alternatif yang dapat digunakan dalam mengevaluasi kinerja algoritma pencarian solusi PVRPSD ini adalah total kinerja terendah yang dapat dihasilkan oleh suatu blok eksperimen.

Dengan menggunakan acuan total kinerja ini, maka total kinerja terendah dihasilkan oleh blok eksperimen nomor 7 dengan ekspektasi total waktu tour distribusi sebesar 8066,15 menit dan waktu komputasi sebesar 1305,96 detik. Nilai ekspektasi total waktu tour distribusi yang dihasilkan blok nomor 7 ini hanya 1,22% lebih buruk dari perolehan blok nomor 11, namun memerlukan waktu komputasi yang lebih cepat secara signifikan, yaitu sekitar 5,92 kali lebih cepat.

B. Pengaruh Faktor Eksperimen Secara Individual

Hasil eksperimen pada TABEL III menunjukkan bahwa level faktor A [50:500] yaitu kombinasi parameter algoritma pencarian solusi berupa penggunaan sejumlah 50 bibit solusi awal dan 500 iterasi telah berhasil memberikan hasil solusi terbaik. Hasil ini dapat dilihat pada nilai-nilai ekspektasi total waktu tour distribusi yang berhasil diperoleh pada blok eksperimen nomor 10 sampai dengan nomor 18 yang secara rata-rata lebih rendah dari blok yang lain. Namun semua blok dengan level faktor A ini, memerlukan waktu komputasi yang rata-rata dua kali lebih panjang dibandingkan dengan level faktor A yang lain. Adapun level faktor A yang terburuk adalah kombinasi parameter [10:1000] pada blok eksperimen nomor 19 hingga nomor 27. Pada kombinasi parameter [10:1000] ini jumlah bibit solusi awal yang digunakan terlalu sedikit. Sehingga, meskipun iterasi yang dijalankan sangat besar, namun hal tersebut tetap tidak mampu menghasilkan nilai-nilai solusi yang kualitasnya lebih baik dari blok eksperimen yang lain, bahkan sebagian besarnya menjadi yang terburuk. Level faktor A [100:100] adalah yang paling ideal karena blok-blok eksperimen dengan level faktor ini menghasilkan rata-rata nilai ekspektasi yang paling dekat dengan yang dihasilkan level faktor A [50:500] namun dengan waktu komputasi yang jauh lebih singkat. Fakta ini menunjukkan pentingnya menjaga kesimbangan rasio antara jumlah bibit solusi awal yang digunakan dengan jumlah iterasi yang dipilih bagi unjuk kerja utama dan unjuk kerja pendukung dari algoritma solusi yang diuji.

Level faktor B [70%:30%] yaitu parameter yang mengatur algoritma agar 70% waktu komputasi digunakan untuk strategi eksplorasi terlihat menghasilkan durasi waktu komputasi yang secara rata-rata lebih rendah dari blok eksperimen lain. Hasil ini dapat dilihat pada sejumlah blok dengan level faktor B [70%:30%] yaitu blok nomor 7 hingga nomor 9; dan blok nomor 25 sampai dengan nomor 27. Data ini menunjukkan bahwa strategi eksplorasi pada algoritma yang telah dirancang memiliki kompleksitas komputasi yang lebih rendah dari pada strategi eksploitasi sehingga strategi ini cenderung menghabiskan sumber daya komputasi yang lebih sedikit. Hal ini dikonfirmasi oleh durasi komputasi yang rata-rata lebih panjang pada blok eksperimen yang menggunakan level faktor B [30%:70%] yaitu: blok nomor 10 hingga nomor 12, yang menggunakan 70% waktu untuk menjalankan strategi eksploitasi.

Dalam eksperimen ini, Level faktor C tidak menunjukkan pola pengaruh yang konsisten terhadap kinerja algoritma pencarian solusi. Hal ini dapat dimengerti karena metode ITCE dan OTCO, yang digunakan dalam eksperimen, keduanya merupakan bagian dari strategi eksplorasi. Ini berarti pengaruh positif terhadap kinerja algoritma berupa waktu komputasi yang lebih rendah disebabkan oleh strategi eksplorasi secara keseluruhan. Namun, tidak dapat dipastikan metode eksplorasi mana yang lebih unggul di antara ITCE dan OTCO.

A. *Pengaruh Interaksi Antar Faktor Eksperimen*

Faktor A pada seluruh level akan selalu memberikan hasil lebih baik ketika berinteraksi dengan level faktor B [70%:30%] dalam hal kecepatan komputasi. Artinya, alokasi strategi eksplorasi yang lebih besar akan selalu mengurangi beban komputasi yang harus dilakukan.

Meskipun solusi terbaik berupa nilai ekspektasi terendah diperoleh pada blok nomor 11 yang merupakan hasil interaksi level faktor A [50:500] dengan level faktor B [30%:70%], namun interaksi kedua level faktor ini tidak memberikan hasil kinerja yang konsisten baik pada blok-blok eksperimen yang lain.

V. KESIMPULAN

Pada penelitian ini dirancang suatu metode solusi heuristik untuk menyelesaikan permasalahan PVRPSD pada kasus operasi distribusi produk ke daerah pelosok. Pada kasus ini tidak dimungkinkan untuk melakukan pengisian ulang kendaraan lebih dari dua kali dalam satu hari karena lokasi depot yang terpisah jauh dari klaster pengecer. Ketersediaan kendaraan tunggal menyebabkan harus dilakukan perencanaan multi periode untuk suatu horizon perencanaan.

Hasil pengujian terhadap kinerja algoritma pencarian solusi yang dikembangkan dilakukan menggunakan 27 blok eksperimen yang mengkombinasikan tiga faktor yang merupakan parameter-parameter yang diduga berpengaruh terhadap kinerja pencarian solusi.

Hasil eksperimen menunjukkan bukti bahwa kombinasi antara jumlah bibit solusi awal yang digunakan dan jumlah iterasi pada proses pencarian solusi berpengaruh pada

kinerja algoritma pencarian solusi. Penggunaan jumlah iterasi dan solusi awal yang seimbang dan jumlahnya tidak terlalu berlebihan akan menghasilkan solusi berkualitas baik tanpa harus menghabiskan waktu komputasi secara tidak efisien.

Penggunaan yang lebih intensif strategi eksploratif baik metode ITCE maupun metode OTCO pada proses pencarian solusi terbukti memberikan hasil-hasil solusi dengan kualitas yang lebih baik dibandingkan dengan penggunaan strategi eksploitatif.

UCAPAN TERIMA KASIH / ACKNOWLEDGMENT

Penulis berterima kasih atas masukan dan saran dari para *reviewers* serta kepada Fakultas Teknik Universitas Tanjungpura atas pendanaan penelitian ini melalui skema dana riset DIPA FT-UNTAN Tahun 2023.

REFERENSI

[1] G. D. Konstantakopoulos, S. P. Gayialis, and E. P. Kechagias, "Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification," *Oper. Res.*, vol. 22, no. 3, pp. 2033–2062, Jul. 2022, doi: 10.1007/s12351-020-00600-7.

[2] H. Zhang, H. Ge, J. Yang, and Y. Tong, "Review of Vehicle Routing Problems: Models, Classification and Solving Algorithms," *Arch. Comput. Methods Eng.*, vol. 29, no. 1, pp. 195–221, Jan. 2022, doi: 10.1007/s11831-021-09574-x.

[3] A. Mor and M. G. Speranza, "Vehicle routing problems over time : a survey," *4OR*, vol. 18, no. 2, pp. 129–149, 2020, doi: 10.1007/s10288-020-00433-2.

[4] R. Elshaer and H. Awad, "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants," *Comput. Ind. Eng.*, vol. 140, p. 106242, Feb. 2020, doi: 10.1016/j.cie.2019.106242.

[5] J. Oyola, H. Arntzen, and D. L. Woodruff, "The stochastic vehicle routing problem, a literature review, part I: models," *EURO J. Transp. Logist.*, vol. 7, no. 3, pp. 193–221, 2018, doi: 10.1007/s13676-016-0100-5.

[6] M. Gendreau, O. Jabali, and W. Rei, "Future research directions in stochastic vehicle routing," *Transp. Sci.*, vol. 50, no. 4, pp. 1163–1173, 2016, doi: 10.1287/trsc.2016.0709.

[7] F. Torres, M. Gendreau, and W. Rei, "Crowdshipping: An open VRP variant with stochastic destinations," *Transp. Res. Part C Emerg. Technol.*, vol. 140, p. 103677, Jul. 2022, doi: 10.1016/j.trc.2022.103677.

[8] A. A. Juan *et al.*, "A review of the role of heuristics in stochastic optimisation: from metaheuristics to learnheuristics," *Ann. Oper. Res.*, vol. 320, no. 2, pp. 831–861, Jan. 2023, doi: 10.1007/s10479-021-04142-9.

[9] J. Oyola, H. Arntzen, and D. L. Woodruff, "The stochastic vehicle routing problem, a literature review, Part II: solution methods," *EURO J. Transp. Logist.*, vol. 6, no. 4, pp. 349–388, Dec. 2017, doi: 10.1007/s13676-016-0099-7.

[10] Kellen D. Endler, Cassius T. Scarpin, Maria T. A. Steiner, and Alexandre C. Choueiri, "Systematic Review of the Latest Scientific Publications on the Vehicle Routing Problem," *Asia-Pac. J. Oper. Res.*, vol. 40, no. 6, p. 2250046.

[11] M. A. Ebrahimi, H. Dehghan Dehnavi, M. Mirabi, M. T. Honari, and A. Sadeghian, "Solving NP hard problems using a new genetic algorithm," *Int. J. Nonlinear Anal. Appl.*, no. Online First, Apr. 2022, doi: 10.22075/ijnaa.2022.26610.3364.

[12] D. J. Bertsimas, "A Vehicle Routing Problem with Stochastic Demand," *Oper. Res.*, vol. 40, no. 3, pp. 574–585, Jun. 1992, doi: 10.1287/opre.40.3.574.

[13] L. Calvet, D. Wang, A. Juan, and L. Bové, "Solving the multidepot vehicle routing problem with limited depot capacity and stochastic demands," *Int. Trans. Oper. Res.*, vol. 26, no. 2, pp. 458–484, Mar. 2019, doi: 10.1111/itor.12560.

[14] C. L. Quintero-Araujo, D. Guimarans, and A. A. Juan, "A simheuristic algorithm for the capacitated location routing problem with stochastic demands," *J. Simul.*, vol. 15, no. 3, pp. 217–234, Jul. 2021, doi: 10.1080/17477778.2019.1680262.

[15] D. R. Gaur, A. Mudgal, and R. R. Singh, "Improved approximation algorithms for cumulative VRP with stochastic demands," *Discrete Appl. Math.*, vol. 280, pp. 133–143, Jun. 2020, doi: 10.1016/j.dam.2018.01.012.

[16] F. Liu, C. Lu, L. Gui, Q. Zhang, X. Tong, and M. Yuan, "Heuristics for Vehicle Routing Problem: A Survey and Recent Advances." arXiv, Mar. 01, 2023. Accessed: Nov. 01, 2023. [Online]. Available: http://arxiv.org/abs/2303.04147

[17] B. Singh, L. Oberfichtner, and S. Ivliev, "Heuristics for a cash-collection routing problem with a cluster-first route-second approach," *Ann. Oper. Res.*, vol. 322, no. 1, pp. 413–440, Mar. 2023, doi: 10.1007/s10479-022-04883-1.

[18] F. Kocatürk, G. Y. Tütüncü, and S. Salhi, "The multi-depot heterogeneous VRP with backhauls: formulation and a hybrid VNS with GRAMPS meta-heuristic approach," *Ann. Oper. Res.*, vol. 307, no. 1–2, pp. 277–302, Dec. 2021, doi: 10.1007/s10479-021-04137-6.

[19] Pisinger , David and Ropke, Stefan, "Large neighborhood search," in *Handbook of Metaheuristics*, 3th ed., vol. 146, M. Gendreau and J.-Y. Potvin, Eds., in International Series in Operations Research & Management Science, vol. 146. , Boston, MA: Springer US, 2019.