Aplikasi Warehouse Inventory Picking Order pada PT. XYZ Menggunakan Symbiotic Organism Search Algorithm

Adelyn Thungriallu, Andreas Handojo, Tanti Octavia
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236, Indonesia
Telp. (031) - 2983455, Fax. (031) - 8417658
Email: adelynthung@gmail.com, handojo@petra.ac.id, tanti@petra.ac.id

ABSTRAK

Warehouse Inventory Picking Order merupakan proses mengambil atau meletakkan barang pada gudang sesuai daftar permintaan. Proses ini merupakan proses yang penting pada perusahaan seperti PT.XYZ, karena melibatkan biaya operasional dan dapat mempengaruhi tingkat kepuasan pelanggan. Dibutuhkan efisiensi yang menekankan kecepatan dan akurasi pengambilan atau peletakan barang di gudang. Semakin efisien proses *picking order*, maka akan semakin efisien biaya operasional dan waktu yang dibutuhkan hingga proses ini selesai.

Sayangnya, proses ini seringkali dilakukan secara manual dan menjadi tidak efisien karena lama dan berulang-ulang. Untuk menjawab permasalahan tersebut, dirancang aplikasi berbasis web yang dapat menentukan rute pengambilan dan peletakan barang untuk meminimalkan jarak tempuh dengan menggunakan metode *Symbiotic Organism Search Algorithm* (SOS *Algorithm*) untuk data diskrit. Hasil akhir yang didapatkan adalah masalah *routing* dapat diselesaikan menggunakan SOS *Algorithm* untuk data diskrit dengan tingkat penurunan jarak rata-rata sebesar 38.49% untuk 25 data.

Kata Kunci: Gudang, *Picking Order*, *Routing*, *Symbiotic Organism Search Algorithm*, Optimasi Jarak

ABSTRACT

Warehouse Inventory Picking Order is the process of taking or placing goods in the warehouse according to the demand list. This is an important process in a company, for example PT. XYZ, because it involves operational costs and can affect the customer satisfaction. Efficiency is needed, especially in the speed and accuracy of picking up or placing products in the warehouse. If the picking order process is efficient, the operational costs and time required will be more efficient too.

Unfortunately, this process is often done manually and becomes inefficient due to long and repetitive processes. To answer this problem, a web-based application was designed to determine the route for picking and placing products to minimize the travel distance by using the Discrete Symbiotic Organism Search Algorithm (SOS Algorithm) for discrete data. The final result is that the routing problem can be solved using SOS Algorithm for discrete data with an average distance reduction rate of 38.49% for 25 data.

Keywords: *Warehouse, Picking Order, Routing, Symbiotic Organism Search Algorithm, Distance Optimization.*

1. PENDAHULUAN

PT. XYZ merupakan salah satu perusahaan yang bergerak dalam pendistribusian barang di Manado, Sulawesi Utara. Terdapat dua macam cara peletakan barang di gudang pada PT. XYZ, yakni dalam bentuk *pallet* di lantai gudang untuk barang besar dan berat, sedangkan untuk barang yang relatif kecil dan ringan disimpan pada rak. Tidak ada sistem khusus dalam pengambilan atau peletakan barang, *staff* gudang hanya akan mengambil barang sesuai nota dari departemen administrasi. Selain itu, ada kasus tertentu dimana *staff* tidak mengetahui letak barang yang akan diambil pada gudang, biasanya terjadi pada *staff* baru. Hal ini membuat proses pengambilan dan peletakan barang atau yang disebut *warehouse inventory picking order* menjadi tidak efisien karena lama dan berulang-ulang.

Warehouse Inventory Picking Order merupakan proses yang penting pada perusahaan, karena melibatkan biaya operasional dan dapat mempengaruhi tingkat kepuasan pelanggan. Dibutuhkan efisiensi yang menekankan kecepatan dan akurasi pengambilan atau peletakan barang di gudang. Efisiensi diperlukan agar dapat memenuhi pengambilan barang dengan lebih cepat dan tepat tanpa harus merugikan *customer* karena keterlambatan pengiriman. Semakin efisien proses *picking order*, maka akan semakin efisien biaya operasional perusahaan dan waktu yang dibutuhkan akan semakin singkat.

Untuk menjawab masalah tersebut, dirancang sebuah aplikasi berbasis web yang dapat menentukan rute pengambilan dan peletakan barang dengan meminimalkan jarak tempuh menggunakan metode *Symbiotic Organism Search Algorithm* (SOS *Algorithm*). SOS *Algorithm* merupakan algoritma optimasi metaheuristik baru yang kuat untuk optimasi numerik dan masalah desain teknik. SOS *Algorithm* juga dapat digunakan untuk masalah diskrit, seperti *Capacitated Vehicle Routing Problem* dan *Travelling Salesman Problem* [2]. Dikarenakan *routing* merupakan masalah diskrit, maka SOS *Algorithm* yang akan digunakan adalah SOS untuk data diskrit atau *Discrete SOS*. SOS *Algorithm* mensimulasikan strategi interaksi simbiosis yang diadopsi oleh organisme untuk bertahan hidup dan berkembang biak di ekosistem. SOS *Algorithm* secara iteratif menggunakan populasi kandidat solusi yang menjanjikan dalam proses mencari solusi global yang optimal [1]. Oleh karena itu metode ini cocok digunakan dalam pencarian rute *picking order* pada gudang.

2. DASAR TEORI

2.1 *Warehouse Inventory Picking Order*

Warehouse Inventory Picking Order merupakan proses pengambilan barang pada gudang untuk memenuhi permintaan pelanggan. *Warehouse Inventory Picking Order* adalah salah satu aspek penting dalam pergudangan yang menghabiskan sekitar 55% biaya operasi pusat distribusi [5]. Hal ini pula berdampak

pada kepuasan pelanggan, yakni pada aspek kecepatan dan keakuratan pengiriman barang. *Picking Order* merupakan salah satu masalah yang banyak diteliti untuk dipecahkan dengan mempertimbangkan jumlah waktu minimum pengambilan barang, keterbatasan sumber daya yang ada, juga biaya yang dibutuhkan.

2.2 *Symbiotic Organism Search Algorithm*

Symbiotic Organism Search (SOS) *Algorithm* merupakan algoritma optimasi metaheuristik baru yang kuat untuk optimasi numerik dan masalah desain teknik. Contoh metode optimasi metaheuristik lain seperti Tabu Search, Particle Swarm Optimization, dan Genetic Algorithm. SOS *Algorithm* mengadopsi interaksi simbiosis organisme untuk bertahan hidup dan berkembang biak di ekosistem, yakni Mutualisme, Komensalisme, dan Parasitisme. SOS *Algorithm* memiliki kinerja yang sangat baik dalam menyelesaikan berbagai masalah numerik yang kompleks. SOS *Algorithm* secara iteratif menggunakan populasi kandidat solusi yang menjanjikan dalam proses mencari solusi global yang optimal [1].

Algoritma dimulai dengan melakukan inisialisasi ekosistem yang terdiri atas organisme sebanyak eco_size. Selanjutnya dilakukan pemilihan terhadap Xbest dari semua organisme yang ada. Iterasi sebanyak i dilakukan dengan mengacu pada kriteria pengulangan hasil yang telah ditetapkan, misalnya jika Xbest tidak mengalami perubahan sebanyak n maka pengulangan selesai. Pengulangan tersebut disebut *outer loop*. Selanjutnya dilakukan pengulangan sebanyak eco_size sebagai *inner-loop* untuk masuk ke dalam tahapan Mutualisme, Komensalisme, dan Parasitisme. Hasil dari *inner-loop* adalah pembaruan Xbest dan hasil dari *outer-loop* adalah hasil akhir atau rute berdasarkan Xbest terakhir.

Fase Mutualisme akan mengambil Xi dan Xj untuk dibandingkan dengan hasil Xi dan Xj baru. Xi merupakan iterasi organisme pada ekosistem, sedangkan Xj merupakan hasil random dari organisme pada ekosistem. Selanjutnya Xi dan Xj akan di *update* jika hasil Xi dan Xj yang baru lebih baik jika dibandingkan dengan yang lama. Berikut adalah persamaan dalam fase Mutualisme:

$$X_{inew} = X_i + rand(0,1) * (X_{best} - Mutual\_Vector * \mathrm{BF}_1) \quad (1)$$

$$X_{jnew} = X_j + rand(0,1) * (X_{best} - Mutual\_Vector * \mathrm{BF}_2) \quad (2)$$

$$Mutual\_Vector = \frac{X_i + X_j}{2} \quad (3)$$

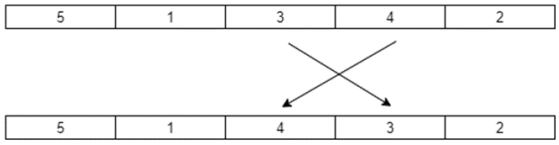
Fase Komensalisme merupakan kelanjutan dari fase Mutualisme. Xi yang menjadi output fase Mutualisme akan menjadi input untuk fase ini. Diambil pula Xj sebagai hasil random dari organisme pada ekosistem. Xj akan digunakan menjadi faktor yang mempengaruhi keberlangsungan hidup organisme Xi. Xi akan di *update* jika hasil Xi new lebih baik daripada Xi sebelumnya. Berikut adalah persamaan fase Komensalisme:

$$X_{inew} = X_i + rand(-1,1)^*(X_{best} - X_j) \quad (4)$$

Fase Parasitisme merupakan fase terakhir, dimana Xi akan diberikan Parasite Vector yang akan mencoba untuk menggantikan Xj pada ekosistem. Jika *Parasite Vector* memiliki hasil yang lebih baik, maka *Parasite Vector* akan mengambil posisi Xj pada ekosistem, sedangkan jika lebih buruk, maka Parasite Vector tidak akan ditambahkan dalam ekosistem.

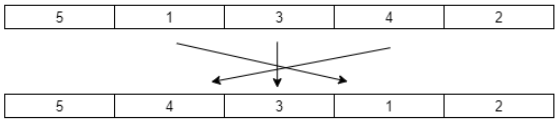
Terdapat tiga tahap yang diperlukan untuk melakukan optimasi pada data diskrit, sebelum masuk ke fase Mutualisme, Komensalisme dan Parasitisme. Ketiga tahap tersebut adalah *Swap Mutation Operator*, *Inversion Mutation Operator*, dan *Insertion Operator* [3]. Ketiga tahap tersebut adalah sebagai berikut:

1. *Swap Mutation Operator* : Pada tahap ini, dilakukan pertukaran posisi dari 2 data random dalam sebuah vektor solusi. Dipilih i dan j secara acak, lalu nilai dari posisi i dan j ditukar untuk membentuk sebuah vektor solusi baru yang kemudian dibandingkan dengan vektor solusi sebelumnya. Jika ternyata vektor solusi baru lebih baik dari yang lama, maka vektor solusi lama digantikan dengan vektor solusi baru.



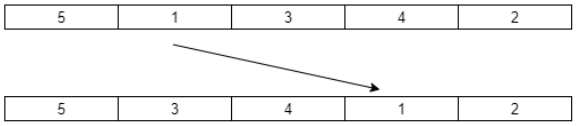
Gambar 1. *Swap Mutation Operator*

2. *Inversion Mutation Operator* : Pada tahap ini, dilakukan pembalikan terhadap data vektor solusi. Dipilih i dan j secara acak, lalu dibuat sebuah vektor solusi baru dengan melakukan pembalikan dari index ke i hingga ke j. Setelah itu vektor solusi baru dibandingkan dengan yang lama. Jika vektor solusi baru lebih baik, maka vektor solusi lama digantikan dengan yang baru.



Gambar 2. *Inversion Mutation Operator*

3. *Insertion Operator* : Pada tahap ini, dilakukan penyisipan data vektor solusi ke index yang baru. Pertama-tama dilakukan pemilihan i dan j secara acak. Setelah itu, dibuat sebuah vektor solusi baru dengan mengambil data dari index i dan disisipkan ke index j.

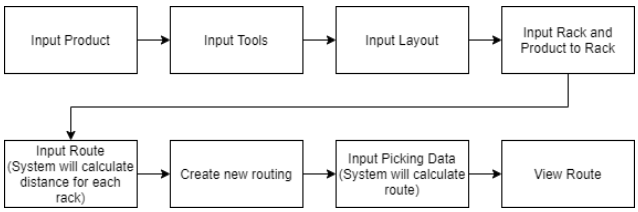


Gambar 3. *Insertion Operator*

3. DESAIN SISTEM

3.1 Desain Aplikasi

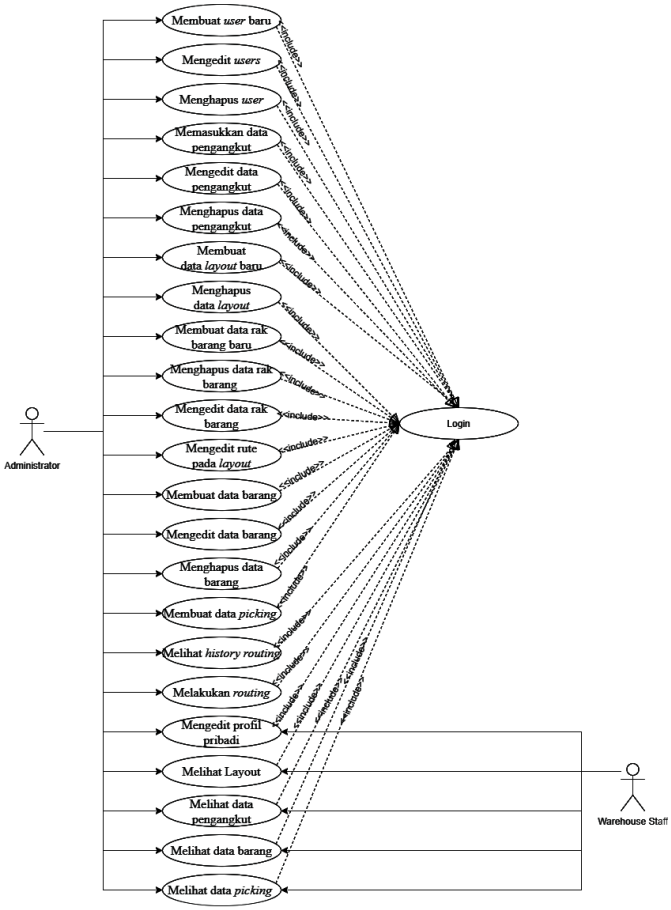
Untuk menggambarkan cara kerja aplikasi secara keseluruhan, maka dibuat *flowchart* yang dapat dilihat pada Gambar 4. Pertama-tama pengguna (user) diminta untuk mengisi data terlebih dahulu. Data yang dibutuhkan seperti data produk, pengangkut yang digunakan, layout yang akan dipakai, rak dan barang pada rak, kemudian rute layout. Setelah itu, user dapat melakukan *routing* dengan membuat data *routing* baru dan mengisi daftar produk yang akan diambil/diletakkan. Kemudian sistem akan melakukan *routing* dengan mencari jarak tempuh terdekat. User dapat melihat hasil *routing* pada menu View Route.



Gambar 4. *Flowchart* Aplikasi

3.2 Desain Hak Akses Pengguna

Pada sistem yang dirancang terdapat 2 tipe pengguna, yaitu *Administrator* dan *Warehouse Staff*. Masing-masing pengguna memiliki hak akses yang berbeda. Desain hak akses digambarkan melalui usecase diagram yang dapat dilihat pada Gambar 5.



Gambar 5. *Use Case Diagram*

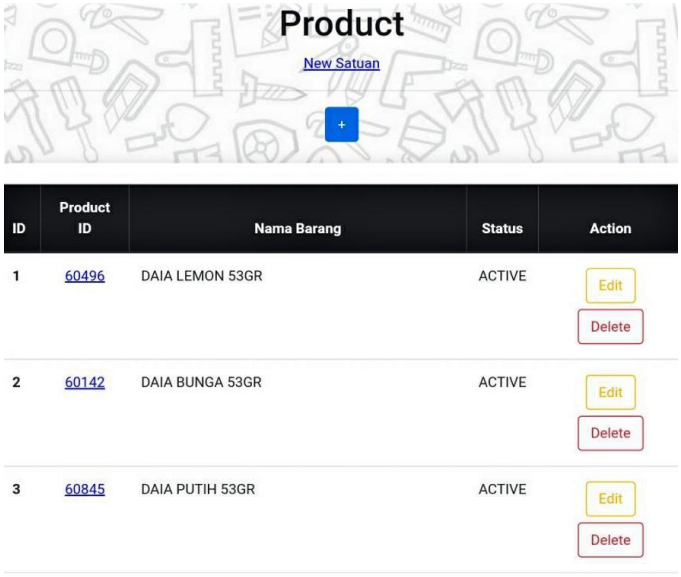
Administrator dapat membuat *user* baru, mengedit *users*, menghapus *user*, melihat dan mengedit profil pribadi. *Warehouse Staff* hanya dapat melihat dan mengedit profil pribadi saja. *Administrator* dapat memasukkan data pengangkut, mengedit data pengangkut, menghapus data pengangkut. *Warehouse Staff* hanya dapat melihat data Pengangkut. Selain itu, *Administrator* dapat membuat data layout baru, menghapus data layout, membuat data rak barang baru pada layout, menghapus data rak barang pada layout, mengedit data rak barang pada layout, mengedit rute pada layout. Sedangkan *warehouse staff* hanya dapat melihat data layout beserta raknya. *Administrator* juga dapat membuat data barang, mengedit data barang, menghapus data barang. *Warehouse Staff* hanya dapat melihat data barang. *Administrator* dapat memasukkan data picking, melakukan *routing* terhadap data picking, dan melihat *history* routing, sedangkan *Warehouse Staff* hanya dapat melihat data picking yang sesuai dengan id dari user tersebut.

4. PENGUJIAN SISTEM

4.1 Pengujian Aplikasi

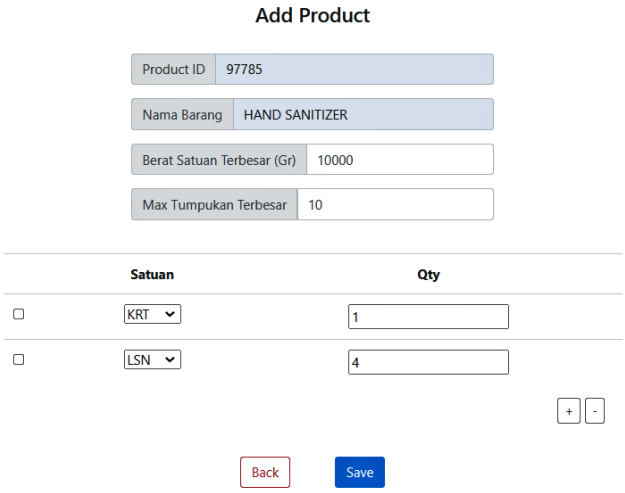
4.1.1 Fitur manajemen data

Fitur manajemen data digunakan untuk memasukkan, mengubah, atau menghapus data pengguna, produk, dan pengangkut. Untuk memanajemen data-data tersebut, user perlu memilih menu data yang diinginkan. Sebagai contoh, user ingin memanajemen data produk, maka user perlu mengklik menu Product dan akan muncul tampilan seperti pada Gambar 6.



Gambar 6. Halaman Produk

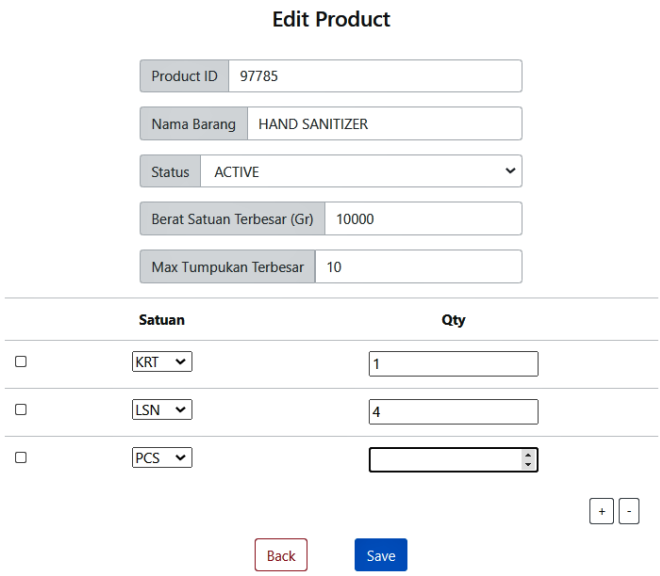
Jika ingin menambahkan data produk, maka user perlu mengklik tombol "+" pada bagian atas halaman produk. Lalu user diminta mengisi data yang sesuai seperti tampak pada Gambar 7.



Gambar 7. Halaman untuk menambah data produk

Jika ingin mengubah data produk yang sudah pernah dimasukkan sebelumnya, maka user perlu mengklik tombol Edit pada produk yang diinginkan. Lalu user dapat mengubah data pada halaman Edit Product seperti tampak pada Gambar 8.

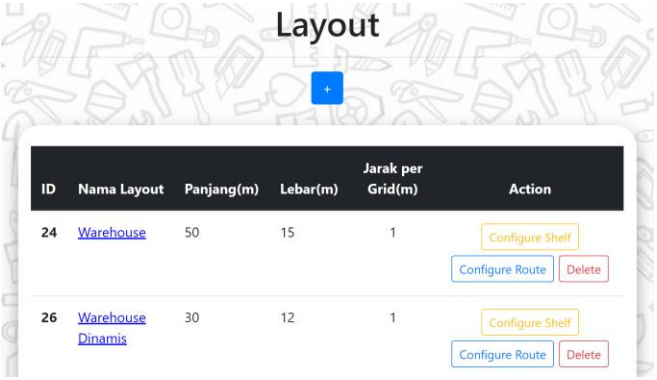
Jika ingin menghapus data produk, maka user perlu mengklik tombol Delete pada produk yang diinginkan. Kemudian akan muncul kotak konfirmasi sebelum menghapus data. Data akan dihapus setelah user menyetujui penghapusan data pada kotak konfirmasi. Untuk melakukan manajemen data selain data produk, maka user tinggal mengklik menu yang sesuai. Menu yang digunakan yaitu Product untuk data produk, menu Users untuk data user, dan menu Tools untuk data pengangkut.



Gambar 8. Halaman untuk Mengubah Data Pengangkut

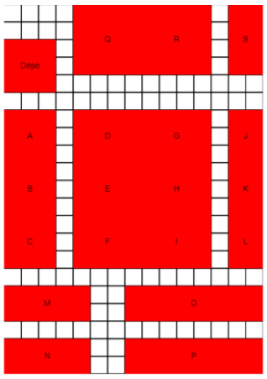
4.1.2 *Fitur manajemen layout*

Fitur manajemen layout digunakan untuk mengubah atau menghapus data pada layout, yaitu rak, barang pada rak, dan rute layout yang dipilih. Sebelum mengubah data pada layout, terlebih dahulu user mengklik menu layout dan halaman akan ditampilkan seperti pada Gambar 9.



Gambar 9. Halaman Layout

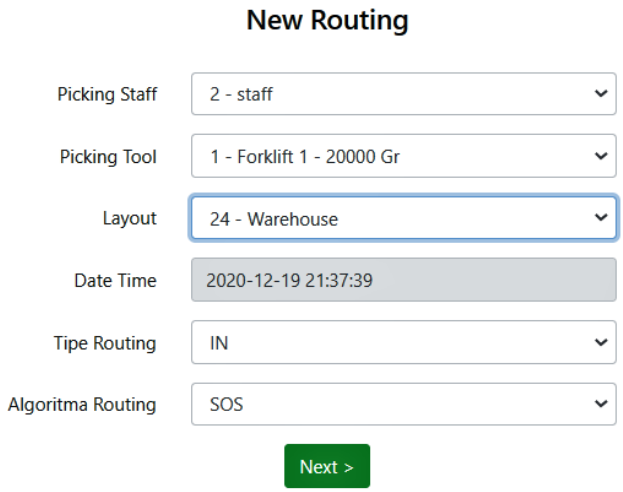
Jika ingin menambahkan data layout baru, maka user perlu mengklik tombol "+" pada bagian atas halaman layout kemudian mengisi data yang diperlukan. Selanjutnya, jika user ingin melakukan manajemen data rak pada layout yang diinginkan, maka user perlu mengklik tombol Configure Shelf. Untuk menambahkan data rak baru, maka user perlu mengklik tombol "Add" dan diminta untuk mengisi grid yang diinginkan untuk menjadi rak beserta titik pengambilan rak tersebut. Ketika user ingin menambahkan data barang pada sebuah rak, maka user perlu mengklik Edit pada rak yang diinginkan kemudian mengisi data barang. Jika user ingin mengatur rute pada layout, maka user perlu mengklik Configure Route pada layout yang diinginkan kemudian mengklik grid untuk dijadikan rute.



Gambar 10. Grid untuk Layout

4.1.3 *Fitur routing*

Untuk melakukan *routing* user perlu mengisi data *routing* beserta barang terlebih dahulu. Tampilan dapat dilihat pada Gambar 11.



Gambar 11. Halaman untuk membuat *routing* baru

Jika user ingin melihat hasil *routing* yang pernah dilakukan maka user mengklik menu History dan akan muncul tabel hasil *routing*. Jika ingin melihat rute hasil *routing*, maka user mengklik tombol View dan akan muncul tampilan untuk melihat rute pengambilan/peletakan barang.

4.2 Verifikasi Model

Untuk mengetahui apakah algoritma dan model yang dibuat sudah benar dan sesuai, maka dilakukan verifikasi model. Verifikasi model dilakukan dengan perhitungan manual juga perhitungan sistem dan keduanya menggunakan data permintaan.

Model permintaan yang digunakan memiliki jumlah data pengambilan sebanyak 4 data namun memiliki berat total lebih dari kapasitas pengangkut. Diketahui kapasitas pengangkut adalah 10.000 Gram atau 10 Kg. Masing-masing data memiliki rak ambil dan jaraknya dapat dilihat pada Tabel 2. Data permintaan tampak seperti Tabel 1.

Tabel 1. Data permintaan Verifikasi Model

Item	Id Barang	Qty	Satuan	Berat Total (Gr)	Rak	Berat Total (Gr)
1	1	3	KRT	11448	D	11448
2	3	2	KRT	7632	D	7632
3	16	2	KRT	12672	F	12672
4	21	2	KRT	21600	I	21600

Tabel 2. Data jarak antar rak

Rak	Depo	D	F	I
Depo	0	4	10	19
D	4	0	6	15
F	10	6	0	9
I	19	15	9	0

Pertama-tama, sistem akan membagi data sesuai berat dan kapasitas alat angkut. Setelah itu, sistem akan melakukan *routing* sesuai permintaan dan kembali ke Depo jika kapasitas alat angkut sudah penuh. Hal tersebut akan dilakukan terus menerus hingga permintaan habis. Parameter yang digunakan pada verifikasi model adalah *ecosystem size* sebesar 20, iterasi sebesar 100, dan *non-improvement* sebesar 20. Hasil *routing* dari aplikasi dapat dilihat pada Gambar 12 dan 13. Sedangkan hasil perhitungan manual dapat dilihat pada Tabel 3 dan 4.

Tabel 3. Hasil perhitungan manual verifikasi model

Id Barang	0	16	0	1	0	21	0	1
Rak	Depo	F	Depo	D	Depo	I	Depo	D
Berat	0	9988	0	9964	0	9900	0	1484
Jarak	0	10	10	4	4	19	19	4
Akumulasi Jarak	0	10	20	24	28	47	66	70

Tabel 4. Hasil perhitungan manual verifikasi model (lanjutan)

Id Barang	16	21	0	21	0	3	0
Rak	F	I	Depo	I	Depo	D	Depo
Berat	2684	1800	0	9900	0	7632	0
Jarak	6	9	19	19	19	4	4
Akumulasi Jarak	76	85	104	123	142	146	**150**

Layout	24	Warehouse
Tipe	OUT	
Jarak Total (M)	150	
Comp. Time	0.284762	

Gambar 12. Hasil *routing* sistem (1)

Barang	Rak	Qty	Tipe Satuan	Urutan Ambil
-	122 - Depo	-	-	1
16 - RINSO + MOLTO 44GR	128 - F	227	2 - PCS	2
-	122 - Depo	-	-	3
1 - DAIA LEMON 53GR	126 - D	188	2 - PCS	4
-	122 - Depo	-	-	5
21 - FLOORCLNR SOKLIN BIRU BOTOL 900ML	131 - I	11	2 - PCS	6
-	122 - Depo	-	-	7
1 - DAIA LEMON 53GR	126 - D	28	2 - PCS	8
16 - RINSO + MOLTO 44GR	128 - F	61	2 - PCS	9
21 - FLOORCLNR SOKLIN BIRU BOTOL 900ML	131 - I	2	2 - PCS	10
-	122 - Depo	-	-	11
21 - FLOORCLNR SOKLIN BIRU BOTOL 900ML	131 - I	11	2 - PCS	12
-	122 - Depo	-	-	13
3 - DAIA PUTIH 53GR	126 - D	2	1 - KRT	14
-	122 - Depo	-	-	15

Gambar 13. Hasil *routing* sistem (2)

Dari perhitungan manual di atas, dapat dilihat bahwa jarak perhitungan manual dan jarak perhitungan aplikasi sama besar, yaitu 150 meter.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil perancangan sistem dan uji coba yang dilakukan, kesimpulan yang didapatkan adalah sebagai berikut:

- Algoritma SOS dapat digunakan pada kasus diskrit seperti masalah *routing* pada proses *picking order*.
- *Routing* pada gudang dilakukan dengan mempertimbangkan berat barang, jarak dan maksimal tumpukan, dan digunakan parameter *eco_size* sebesar 10, iterasi 100, dan *non-improvement* 20 memiliki rata-rata waktu komputasi sebesar 0.23 detik.
- Semakin tinggi jumlah data, *ecosystem size* dan iterasi, akan semakin tinggi pula waktu komputasi yang dibutuhkan.

5.2 Saran

Dari hasil perancangan sistem dan uji coba yang dilakukan, saran yang dapat dipertimbangkan adalah sebagai berikut:

- Dalam melakukan *routing*, selain mempertimbangkan berat, jarak dan maksimal tumpukan, dapat juga ditambahkan volume atau dimensi barang sesuai dengan rak dan alat pengangkut yang digunakan.
- Menggunakan parameter level rak pada saat melakukan *routing* sebagai kasus yang dapat dipecahkan pada perkembangan selanjutnya.
- Sebaiknya dilakukan sistem penjadwalan *routing* sebagai pengganti sistem manual peletakan/pengambilan barang gudang pada perusahaan.

6. DAFTAR PUSTAKA

[1] Cheng, M. Y, Prayogo, D. 2014. Symbiotic organism search: A new metaheuristic optimization. Computers and Structures.

[2] Eki R., Yu V. F., Budi S., Redi A. A. N. P. 2015. Symbiotic Organism Search (SOS) for Solving the Capacitated Vehicle

Routing Problem. World Academy of Science, Engineering and Technology International Journal of Industrial and Manufacturing Engineering Vol:9, No:5

[3] Ezugwu A. E. S, Adewumi A. O. 2017. Discrete Symbiotic Organisms Search Algorithm for Travelling Salesman Problem

[4] Ezugwu A. E. S, Adewumi A. O, Frîncu M. E. 2017. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem

[5] Glynn F. 2020, February 4. The ultimate guide to warehouse order picking. Retrieved from https://6river.com/warehouse-order-picking-guide/

[6] Hayati E. N., Adhi A., Liana L., Evitriani, Sari R. M. 2019. Metaheuristics untuk Menyelesaikan Permasalahan Vehicle Routing Problem : Partial Comparison Optimization. Jurnal DINAMIKA TEKNIK, Vol .XII, No. 1

[7] Murray M. 2019, November 30. Order Picking in The Warehouse. Retrieved from https://www.thebalancesmb.com/order-picking-in-the-warehouse-2221190

[8] Toofani A. 2012. Solving Routing Problem using Particle Swarm Optimization. International Journal of Computer Applications (0975 – 8887) Volume 52– No.18

[9] What Are the Different Types of Order Picking Methods in the Warehouse. 2018, April 5. Retrieved from https://www.apsfulfillment.com/warehousing-solutions/what-are-different-types-order-picking-methods-warehouse/