Aplikasi Optimisasi Penentuan Lokasi Warehouse dengan Runner-Root Algorithm pada PT X

Sanitya Purnomo [1], Yulia [2], Henry Palit [3]
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-mail: sanityapurnomo97@gmail.com [1], yulia@petra.ac.id [2], hnpalit@petra.ac.id [3]

ABSTRAK

Aplikasi berbasis *website* yang akan dibuat adalah aplikasi optimisasi penentuan lokasi sewa *warehouse* yang dapat membantu perusahaan untuk mengetahui dimana lokasi *warehouse* yang paling optimal dengan berbagai parameter yang telah ditentukan. Proses perhitungan untuk menghasilkan rekomendasi menggunakan metode *runner-root algorithm*. Metode *runner-root algorithm* adalah algoritma optimisasi numerik yang terinspirasi oleh tanaman *strawberry* untuk memecahkan masalah *multi-variable*. Tahapan analisa dan desain terhadap data dan kebutuhan perusahaan diperlukan untuk menghasilkan informasi yang baik. Informasi yang dihasilkan oleh aplikasi diharapkan dapat mendukung proses pengambilan keputusan oleh perusahaan.

Hasil akhir dari pengembangan aplikasi ini adalah sebuah aplikasi optimisasi untuk melakukan penentuan lokasi *warehouse*. Aplikasi yang dikembangkan mempunyai kesimpulan bahwa aplikasi lebih dibutuhkan pada kota-kota yang memiliki jumlah *customer* dan jumlah pilihan *warehouse* yang banyak. Jumlah dari iterasi juga menentukan keoptimalan hasil *fitness value* dari perhitungan.

Kata Kunci: *Runner-root*, *Google Maps*, *Decision Support System*, Penentuan Lokasi

ABSTRACT

Website-based application that will be created is an optimization application to determine the warehouse leasing location that can help company to find out where the warehouse location is the most optimal with a variety of predetermined parameters. Calculation process to generate the recommendations is using runner-root algorithm. Runner-root algorithm is a numerical optimization algorithm inspired by the strawberry plant for solving continuous multi-variable problems. Stages of analysis and design of record data and company needs are needed to produce good information. Information generated by the application is expected to support the decision-making process by the company.

The final result of the development is an optimization application for determining the warehouse leasing location. The developed application has conclusion that applications are more needed in cities that have a large number of customers and alternatives of warehouses. The number of iterations also determine the optimization of the fitness value results.

Keywords: *Runner-root*, *Google Maps*, *Decision Support System*, *Location Determination*

1. PENDAHULUAN

Setiap perusahaan produksi selalu berusaha untuk memberikan nilai tambah baru, baik itu bagi kemajuan perusahaan sendiri ataupun bagi *customer* perusahaan. Hal ini diperuntukkan agar perusahaan tetap dapat mempertahankan dan mengembangkan bisnis. Persaingan antara perusahaan yang semakin ketat membuat perusahaan mengupayakan berbagai cara untuk meningkatkan efisiensi dan meningkatkan hubungan dengan *customer* (*engage*). Oleh karena itu, salah satu perusahaan produksi di Jember ingin meningkatkan efisiensi perusahaan dengan tetap memperhatikan jalinan hubungan dengan *customer* yaitu dengan mencari letak lokasi *warehouse* yang terbaik.

PT. X adalah sebuah perusahaan yang bergerak di bidang agrobisnis. Pusat utama dari perusahaan PT. X terletak di Jember, Jawa Timur, Indonesia. Untuk dapat menjalankan bisnisnya, PT. X memiliki beberapa *warehouse* dimana *warehouse* digunakan sebagai tempat penyimpanan barang jadi yang kemudian akan dikirimkan ke *customer* sesuai dengan permintaan. *Warehouse* yang sebagian besar dimiliki oleh PT. X adalah *warehouse* sewa dari pihak ketiga (*vendor*). Faktor utama yang menjadi pertimbangan perusahaan adalah jarak yang dekat dengan *customer* agar *travel time* yang diperlukan untuk mengirimkan barang tidak membutuhkan waktu yang terlalu lama dan harga biaya sewa *warehouse*.

Namun, permasalahan yang terjadi adalah selama ini penentuan lokasi *warehouse* hanya menggunakan perkiraan dari *team* logistik. Ketika kontrak sewa *warehouse* akan habis dalam beberapa bulan, *team* logistik baru akan mengadakan rapat untuk membahas lokasi penentuan sewa *warehouse* selanjutnya. Hal ini menyebabkan waktu penentuan lokasi *warehouse* yang baru harus dilakukan dengan sangat cepat sehingga hasil keputusan penentuan lokasi terkadang kurang memiliki pertimbangan yang baik dan kurang dapat mempertimbangkan berbagai parameter.

Berdasarkan penjelasan di atas, maka skripsi ini dibuat untuk memberikan sebuah aplikasi yang dapat membantu perusahaan dalam menentukan lokasi *warehouse* yang sesuai dengan faktor pertimbangan yang dimiliki oleh perusahaan. Penentuan lokasi ini juga dibantu dengan dukungan visualisasi dari *Google Maps*. Data yang akan digunakan adalah data dari perusahaan mengenai data *customer*, data histori order *customer*, dan data lokasi *warehouse* yang mungkin disewa oleh perusahaan.

2. TINJAUAN PUSTAKA

2.1 *Decision Support Systems*

Decision Support System (DSS) atau disebut juga dengan Sistem Penunjang Keputusan (SPK) merupakan sebuah sistem komputer yang berfungsi untuk membantu pengguna dalam pengambilan

suatu keputusan dari masalah-masalah yang ada [6]. Dalam sebuah DSS, data menjadi komponen utama yang sangat dibutuhkan. Data tersebut digunakan untuk mencari pola dan penyelesaian dari masalah yang ada.

Decision Support System memiliki beberapa karakteristik, antara lain [6]:

1. Membantu para pengambil keputusan dalam proses pengambilan keputusan.
2. Membantu pemegang keputusan, dengan menyatukan cara pikir manusia dan informasi yang terkomputerisasi.
3. Mendukung pengambilan keputusan di segala jenjang, namun sangat efektif dalam jenjang taktis dan strategis.
4. Membantu dalam setiap fase proses pengambilan keputusan: *intelligence*, *design*, *choice*, dan *implementation*.
5. Mendukung berbagai jenis proses dan gaya pengambilan keputusan.
6. Dapat beradaptasi dan fleksibel terhadap perubahan kondisi.
7. Sebuah sistem yang interaktif dan *friendly* dengan pengguna sehingga dapat digunakan oleh manajemen dengan sedikit atau tanpa bantuan dari profesional komputer.
8. Peningkatan yang efektif dalam pengambilan keputusan.
9. Pemegang keputusan tetap memegang kendali secara penuh terhadap setiap langkah pengambilan keputusan dalam menyelesaikan permasalahan. DSS hanya digunakan untuk membantu, bukan menggantikan pemegang keputusan.
10. Mengarah pada pembelajaran, yaitu mengarah pada kebutuhan baru dan penyempurnaan sistem.
11. Dapat dikembangkan dan dimodifikasi.
12. Dapat digunakan sebagai *standalone tool* atau diintegrasikan dengan aplikasi lain.

2.2 Penentuan Lokasi Sewa *Warehouse* Perusahaan

Perusahaan yang dituju adalah perusahaan agrobisnis yang memiliki salah satu kegiatan utama perusahaan yaitu mengirimkan produk akhir/barang jadi perusahaan ke *customer*. Hasil barang jadi hasil olahan dari pabrik yang akan dikirimkan kepada *customer* disimpan dalam sebuah *warehouse* terlebih dahulu. Dalam usaha perusahaan untuk meningkatkan layanan kepada pelanggan, perusahaan berusaha mencari lokasi *warehouse* yang dekat dengan lokasi *customer* sehingga kebutuhan *customer* dapat dengan cepat terpenuhi. Tetapi dalam pelaksanaannya, masih banyak kekurangan yang terjadi. Tidak lain hal ini disebabkan karena tidak adanya informasi pendukung yang mampu memudahkan manajemen dalam pengambilan keputusan lokasi sewa *warehouse*.

Menurut Chitkara, penentuan lokasi *warehouse* yang strategis adalah salah satu keputusan penting bagi perusahaan untuk meningkatkan kegiatan operasional sehari-hari. Lokasi *warehouse* yang strategis adalah wilayah penempatan *warehouse* sebuah perusahaan yang dapat memberikan keuntungan maksimal terhadap perusahaan dengan tetap mempertimbangkan aspek *customer*. Secara garis besar, tujuan utama dari pemilihan letak lokasi *warehouse* untuk meningkatkan performa dari perusahaan dalam melakukan barang ke *customer* sehingga perusahaan dapat memperoleh keuntungan – keuntungan seperti mengurangi waktu tunggu (*delay*) ke *customer*, meningkatkan kepuasan *customer*, dan mempermudah aktivitas perusahaan [3].

2.3 *Haversine Formula*

Rumus *haversine* adalah rumus akurat yang digunakan untuk menentukan jarak antara 2 titik pada sebuah bidang bola dengan mempertimbangkan garis bujur dan garis lintangnya. Formula dari trigonometri bola ini akan menghubungkan sisi dan sudut segitiga pada bola. Formula ini tetap dapat berjalan dengan sangat baik untuk perhitungan numerik bahkan pada jarak yang kecil.

Rumus *haversine* pertama kali dijelaskan oleh dijelaskan oleh Roger Sinnott di majalah *Sky & Telescope* pada tahun 1984. Sinnott menjelaskan bahwa pemisahan sudut antara 2 *latitude* dan *longitude* yang ada di bumi dapat dihitung secara akurat dengan menggunakan rumus *haversine*. Sejak saat itu, rumus ini banyak digunakan untuk kepentingan navigasi. Perhitungan formula yang digunakan dapat dilihat pada Gambar 1.

Haversine formula:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

where φ *is latitude,* λ *is longitude,* R *is earth's radius (mean radius = 6,371km); note that angles need to be in radians to pass to trig functions!*

Gambar 1. *Haversine formula* [2]

2.4 *Runner-Root Algorithm*

Algoritma *metaheuristic* telah menjadi algoritma yang sering digunakan oleh peneliti, ilmuwan, dan akademisi yang disebabkan kemampuan algoritma ini untuk memecahkan masalah dan memberikan solusi yang mendekati optimal untuk berbagai masalah dengan rincian yang mendalam [5]. Salah satu contoh dari algoritma *metaheuristic* adalah algoritma *runner-root algorithm*.

Runner-root algorithm digunakan untuk mencari optimisasi dalam sebuah pemecahan masalah sehingga ditemukan solusi yang terbaik. Algoritma ini memodelkan pencarian dalam bentuk *runner* (pelari) dan *root* (akar). *Runner* akan membantu pencarian pada area-area yang jauh dari sumber yang berfungsi sebagai *global search*, sedangkan *root* akan membantu pencarian pada area-area yang dekat dengan sumber yang berfungsi sebagai *local search*.

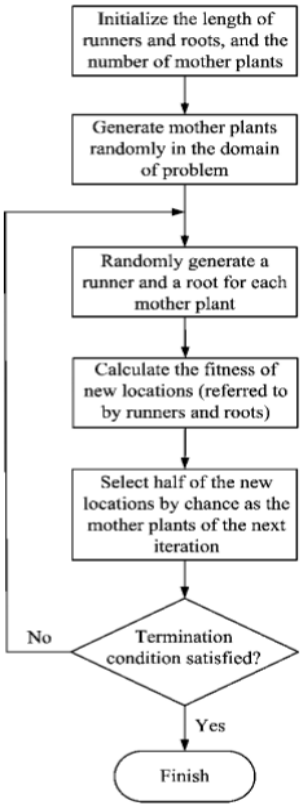
Sama seperti algoritma *metaheuristic* yang lain, *runner-root algorithm* juga memiliki *fitness function*. *Fitness function* adalah tipe fungsi objektif tertentu yang digunakan untuk meringkas seberapa dekat solusi yang telah diberikan untuk mencapai tujuan yang telah ditetapkan. *Fitness function* ini berguna untuk memandu simulasi menuju solusi desain yang optimal. *Fitness function* akan membuang solusi yang kurang optimal dan akan menjadi penentu kapan iterasi selesai dilakukan [4].

Langkah – langkah dalam menerapkan *runner-root algorithm* yang dapat dilihat pada Gambar 2 adalah [1]:

1. Inisialisasi panjang dari *runner* dan *root*, dan *mother plant* (tanaman induk).
2. Membuat (*generate*) *mother plant* secara acak/random dalam permasalahan yang terjadi.
3. Menghasilkan *runner* dan *root* untuk setiap *mother plant* secara acak. Mendefinisikan x_j (iterasi ke-i) dimana j menunjukkan iterasi *mother plant* tanaman induk untuk iterasi sebelumnya.
4. Mendefiniskan X_{prop} matriks dimana matriks X_{prop} adalah gabungan dari 2 matriks, yaitu X_{root} dan $X_{runner.}$ R1 dan R2 adalah matriks acak yang memiliki elemen dalam kisaran [-

0.5;0,5]. Dist_root dan dist_runner adalah angka skalar yang mewakili jarak *root* dan jarak *runner* dari *mother plant*.

5. Menghitung hasil *fitness function* dari lokasi baru yang ditemukan oleh *runner* dan *root*.
6. Memilih setengah dari lokasi baru (lokasi yang paling baik) dengan menggunakan *chance* sebagai *mother plant* untuk iterasi selanjutnya.
7. Apabila kondisi terpenuhi, selesai. Apabila kondisi belum terpenuhi, akan kembali ke tahapan yang ketiga hingga kondisi terpenuhi.



Gambar 2. Langkah-langkah *runner-root algorithm* [1]

2.5 *Google Maps API*

API (*application programming interface*) merupakan rangkaian instruksi atau program yang digunakan untuk memberikan akses atau layanan yang berbasis *website*. API digunakan sebagai perantara perangkat lunak yang memungkinkan dua aplikasi untuk berkomunikasi satu dengan yang lainnya. Dengan adanya API, maka *software developer* lain dapat memanfaatkan layanan yang tersedia untuk mengembangkan aplikasi yang sedang dibuat.

Google Maps memiliki layanan *service API* yang cukup banyak dan lengkap. Terdapat API khusus yang disediakan untuk membangun program dengan basis *Android*, *iOS*, *Web API* (*javascript*), dan *Web Service APIs*. Layanan yang diberikan pun cukup beragam, antara lain layanan untuk menggambarkan peta pada *website*, visualisasi *latitude* dan *longitude* pada *website*, memberikan informasi kepadatan lalu lintas pada peta tersebut, memberikan informasi jarak dan waktu tempuh antar kedua titik destinasi, dan sebagainya. *API* yang disediakan oleh *Google Maps* untuk ditampilkan pada halaman *website* ini dapat dimodifikasi oleh pengguna sesuai dengan kebutuhan yang diperlukan.

Untuk dapat menggunakan layanan yang diberikan, pengguna harus memiliki *API Key* terlebih dahulu. Kemudian mengaktifkan *service-service* yang diinginkan pada *google console*. Pemanggilan *Google Maps API* tidak berbayar atau gratis sampai dengan jumlah pemanggilan tertentu.

3. ANALISIS DAN DESAIN

3.1 Analisis Permasalahan

Permasalahan yang dihadapi oleh PT. X antara lain:

1. PT. X mengalami kebimbangan untuk menentukan lokasi penyewaan *warehouse* selanjutnya karena perusahaan belum mempunyai informasi apakah lokasi *warehouse* saat ini sudah optimal atau belum. Selain itu, perusahaan belum memiliki informasi perbandingan keoptimalan antara berbagai lokasi *warehouse* yang mungkin disewa oleh PT. X berdasarkan kombinasi dari berbagai parameter yang ditentukan, seperti pembobotan *customer*, lama *travel time* menuju *customer*, biaya sewa *warehouse*, dan kapasitas *warehouse*.
2. Setelah dapat menentukan lokasi sewa *warehouse* selanjutnya, PT. X harus melakukan penentuan distribusi setiap *warehouse* melakukan *supply* ke customer mana saja yang berdasarkan jarak hanya berdasarkan intuisi bersama antara *team* logistik dan *team sales*. Hal ini dikarenakan perusahaan belum memiliki sebuah sistem yang dapat membantu *team* dalam menentukan distribusi *supply*.

3.2 Analisis Kebutuhan

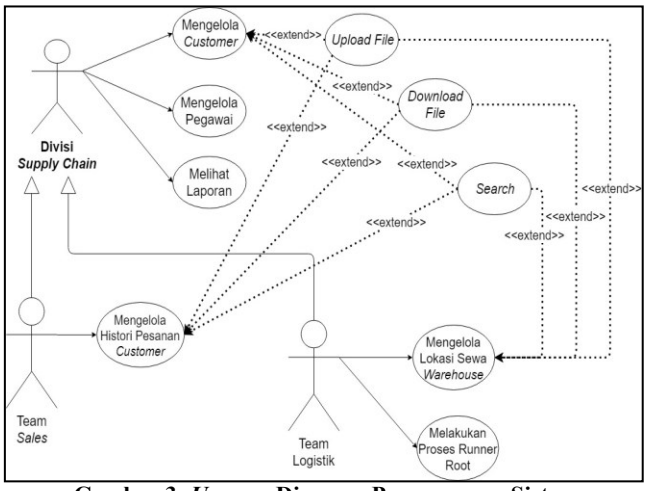
Dari permasalahan-permasalahan yang terjadi, maka dapat disimpulkan bahwa PT. X membutuhkan suatu sistem yang mampu memberikan informasi yang mendukung perusahaan dalam menentukan lokasi penyewaan *warehouse* selanjutnya. Sistem yang dibutuhkan mempunyai kriteria sebagai berikut:

1. Sistem dapat secara otomatisasi melakukan perhitungan dan memberikan rekomendasi mengenai lokasi sewa *warehouse* perusahaan selanjutnya yang paling optimal berdasarkan kombinasi dari berbagai parameter yang telah ditentukan. Setelah itu, sistem dapat menghasilkan *report* perhitungan dari hasil rekomendasi yang dianggap optimal. Sistem ini dapat dijadikan sebagai bahan pertimbangan oleh perusahaan dalam menentukan lokasi *warehouse* selanjutnya sehingga cara perusahaan menentukan lokasi sewa *warehouse* selanjutnya tidak sepenuhnya tanpa analisa yang pasti, namun berdasarkan perhitungan yang telah dilakukan.
2. Sistem dapat melakukan perhitungan dan memberikan informasi mengenai pembagian distribusi setiap *warehouse* melakukan *supply* ke *customer* mana saja. Informasi yang dihasilkan oleh sistem dapat membantu mempercepat dan mempermudah pekerjaan *team* logistik dan *team sales*.

3.3 Desain Sistem

Desain sistem digambarkan dalam bentuk diagram usecase pada Gambar 3. Pada diagram tersebut, terdapat 2 *actor* yang berperan dalam aplikasi yang dibuat, yakni *team* logistik dan *team sales* yang tergabung dalam divisi *supply chain*. Divisi *supply chain* ini mempunyai peranan untuk dapat melakukan pengelolaan pada data *customer*, pengelolaan pada data pegawai, dan melihat hasil rekomendasi dari proses *runner-root*. *Team sales* mempunyai peranan tambahan sebagai *team* yang dapat melakukan input data histori pesanan *customer*. *Team* logistik mempunyai peranan tambahan untuk dapat melakukan input lokasi *warehouse* yang mungkin disewa oleh perusahaan, mengelola truk yang dimiliki

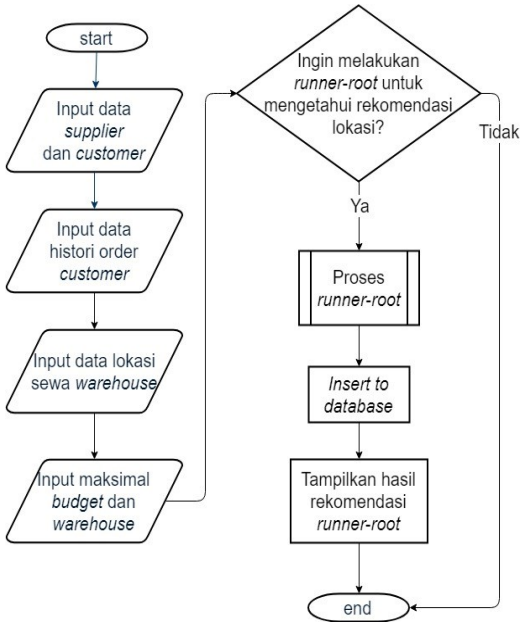
oleh perusahaan dan memproses semua data yang telah diinput untuk menghasilkan lokasi sewa selanjutnya sesuai dengan parameter yang ditentukan atau melakukan proses *runner-root*.



Gambar 3. *Usecase* Diagram Perancangan Sistem

3.4 Desain Flowchart

Secara umum, alur sistem bekerja dapat dibagi menjadi 3 bagian utama, yaitu input data yang diperlukan untuk melakukan proses *runner-root*, proses *runner-root*, dan menampilkan hasil rekomendasi *runner-root* kepada pengguna. *Flowchart* cara kerja sistem dapat dilihat pada Gambar 4. Pada proses *runner-root* sendiri, ada 5 bagian utama untuk melakukan perhitungan, yaitu *calculate* biaya, *calculate* kapasitas, perhitungan pembobotan *customer*, perhitungan pembobotan *supplier*, dan perhitungan *travel time*.



Gambar 4. *Flowchart* Cara Kerja Sistem

3.4.1 Calculate Biaya

Proses perhitungan biaya digunakan untuk mencari total biaya yang diperlukan untuk menyewa seluruh gudang berdasarkan gudang pilihan hasil kombinasi yang diterima sebagai parameter. Sistem akan mencari biaya sewa dari setiap gudang pada *database* berdasarkan data awal yang telah diinput oleh pengguna. Total biaya adalah penjumlahan total dari semua biaya sewa *warehouse*. Jika perhitungan biaya telah mendapatkan hasil, maka hasil akan dikembalikan kembali pada proses *runner-root*.

3.4.2 Calculate Kapasitas

Proses perhitungan kapasitas juga tidak berbeda jauh dengan proses perhitungan biaya. Proses perhitungan kapasitas digunakan untuk mencari total kapasitas yang diperlukan untuk menyewa seluruh gudang berdasarkan gudang pilihan hasil kombinasi yang diterima sebagai parameter. Sistem akan mencari biaya sewa dari setiap gudang pada *database* berdasarkan data awal yang telah diinput oleh pengguna. Total kapasitas adalah penjumlahan total dari semua kapasitas sewa *warehouse*. Jika perhitungan kapasitas telah mendapatkan hasil, maka hasil akan dikembalikan kembali pada proses *runner-root*.

3.4.3 Perhitungan Pembobotan Customer

Proses perhitungan antara rata-rata jumlah pembelian *customer*, frekuensi pembelian *customer*, dan jarak antara *customer* dan *warehouse* yang terdekat digunakan agar setiap *customer* dapat memiliki bobot yang berbeda. *Customer* yang lebih sering melakukan pembelian dan dalam sekali pembelian terdapat banyak barang akan lebih diutamakan daripada *customer* yang lainnya. Untuk setiap iterasi *customer* yang dijalankan, sistem akan mencari rata-rata pembelian *customer* dan jumlah frekuensi pembelian *customer* pada *database*. Sistem juga akan melakukan perhitungan jarak antara lokasi *customer* dan *warehouse*. Jika perhitungan telah selesai, maka hasil akan dikembalikan pada proses *runner-root*.

3.4.4 Perhitungan Pembobotan Supplier

Proses perhitungan antara frekuensi pembelian *supplier* dan jarak antara *supplier* dan *warehouse* yang terdekat digunakan agar setiap *supplier* dapat memiliki bobot yang berbeda. *Supplier* yang lebih sering dibeli akan lebih diutamakan daripada *supplier* yang lainnya. Untuk setiap iterasi *supplier* yang dijalankan, sistem akan mencari jumlah frekuensi pembelian *supplier* pada *database*. Sistem juga akan melakukan perhitungan jarak antara lokasi *supplier* dan *warehouse*. Jika perhitungan antara frekuensi pembelian dan jarak *supplier* telah selesai, maka hasil akan dikembalikan pada proses *runner-root*.

3.4.5 Perhitungan Travel Time

Proses perhitungan *travel time* digunakan untuk mencari total *travel time* yang diperlukan untuk memenuhi kebutuhan seluruh *supplier* dan *customer* perusahaan berdasarkan gudang pilihan hasil kombinasi. Setiap *supplier* dan *customer* akan dicari jarak minimum ke gudang yang mana dari daftar lokasi sewa *warehouse*. Untuk perhitungan *customer*, sistem juga akan memperhatikan total *travel time* berdasarkan hari/jadwal pembelian. Hasil perhitungan ini akan ditambahkan pada pada *fitness value*. Apabila gudang terlalu jauh, nilai *fitness value* akan sangat besar, sehingga kemungkinan tereliminasi pada iterasi selanjutnya sangat besar.

4. HASIL

Pengujian pada bab ini dilakukan untuk memastikan bahwa setiap proses yang dilakukan berjalan dengan benar. Selain itu, pengujian bertujuan untuk menunjukkan validitas sistem dalam melakukan setiap perhitungan yang ada. Pada bab ini, akan ditunjukkan juga hasil akhir dari proses perhitungan dengan menggunakan metode *runner-root*.

4.1 Pembuatan *Mother Plant*

Pada saat pembuatan *mother plant*, sistem akan melakukan 2 kali pengecekan apakah terdapat kombinasi yang kembar atau tidak. Pengecekan pertama adalah pengecekan apakah kombinasi gudang yang akan dipilih menjadi *mother plant* ini sudah pernah menjadi kombinasi yang dipilih sebelumnya. Apabila sudah pernah dipilih sebelumnya, maka sistem akan melakukan *random* ulang. Pengecekan kedua adalah pengecekan tiap individu di dalam tiap kombinasi. Sistem akan melakukan pengecekan apakah gudang ke-n yang dipilih dalam satu kombinasi sudah pernah dipilih sebelumnya. Apabila sudah pernah dipilih sebelumnya, maka sistem akan melakukan *random* ulang. Pengecekan kembar ini diperlukan agar hasil rekomendasi yang dikeluarkan oleh sistem dapat berbeda satu dengan yang lainnya.

Sistem juga memperhatikan batasan-batasan yang telah diinputkan oleh pengguna. Batasan-batasan ini seperti maksimal *budget* dan maksimal jumlah *warehouse*. Apabila total biaya yang terbentuk pada *mother plant* sudah melebihi dari *budget* yang ditentukan, maka sistem tidak akan memasukkan lokasi *warehouse* ke dalam *mother plant*. Jumlah dari pembentukan satu *mother plant* juga bervariasi dari satu hingga jumlah batas maksimal *warehouse* yang telah ditentukan

Sebagai contoh, akan dilakukan simulasi untuk melihat jumlah *mother plant* yang diinisialisasi, pengecekan kembar, dan pengecekan batasan. Apabila total *mother plant* yang diinisialisasi sebanyak 3 dapat dilihat pada Gambar 5.

Mother-Plant ke: 1
Warehouse Code: 59
Warehouse Name: Prajurit
Warehouse City: Probolinggo

Warehouse Code: 63
Warehouse Name: Mastrip
Warehouse City: Probolinggo

Warehouse Code: 64
Warehouse Name: Dringu
Warehouse City: Probolinggo
Total Biaya Warehouse: 1,210,000,000

Mother-Plant ke: 2
Warehouse Code: 57
Warehouse Name: Mawar
Warehouse City: Probolinggo
Total Biaya Warehouse: 850,000,000

Mother-Plant ke: 3
Warehouse Code: 56
Warehouse Name: Ketapang
Warehouse City: Probolinggo

Warehouse Code: 57
Warehouse Name: Mawar
Warehouse City: Probolinggo
Total Biaya Warehouse: 1,550,000,000

Gambar 5. Contoh *Output* Inisialisasi *Mother Plant*

4.2 Pembuatan *Runner* dan *Root*

Setiap *mother plant* yang telah terbentuk pada awal akan menghasilkan *runner* dan *root*. Yang dimaksud dengan *runner* adalah lokasi *warehouse* yang memiliki jarak yang jauh dengan lokasi *mother plant*, sedangkan *root* adalah lokasi *warehouse* yang memiliki jarak yang dekat dengan lokasi *mother plant*. Jadi, jarak antara lokasi *mother plant* dengan *root* akan lebih dekat daripada jarak antara lokasi *mother plant* dengan *runner*.

Sistem melakukan pemilihan *runner* dan *root* menggunakan nilai median. Pemilihan ini dilakukan agar tidak terjadi ketimpangan yang akan menyebabkan sistem susah menemukan *runner* atau menemukan *root* dimana kondisi ini dapat mengakibatkan *looping* susah berhenti. Sebagai contoh, akan dilakukan simulasi pembuatan *runner* dan *root* dari *mother plant* yang diinisialisasi. *Mother plant* yang ter-*generate* adalah *warehouse* dengan kode 57. Pilihan *root* yang dipilih dari *warehouse* tersebut adalah *warehouse* dengan kode nomor 67 yang dapat dilihat pada Gambar 6.

Mother-Plant ke:9
Warehouse Code: 57
Informasi Jarak
Array ([disCode] => 553 [whCode1] => 57 [whCode2] => 63 [distance] => 0.79668326631515)
Array ([disCode] => 551 [whCode1] => 57 [whCode2] => 61 [distance] => 3.1192191730021)
Array ([disCode] => 556 [whCode1] => 57 [whCode2] => 66 [distance] => 3.3657380556275)
Array ([disCode] => 549 [whCode1] => 57 [whCode2] => 59 [distance] => 4.6837983850654)
Array ([disCode] => 557 [whCode1] => 57 [whCode2] => 67 [distance] => 4.7670233739207)
Array ([disCode] => 548 [whCode1] => 57 [whCode2] => 58 [distance] => 4.7805860853781)
Array ([disCode] => 554 [whCode1] => 57 [whCode2] => 64 [distance] => 5.3223251936144)
Array ([disCode] => 536 [whCode1] => 56 [whCode2] => 57 [distance] => 5.407483164816)
Array ([disCode] => 558 [whCode1] => 57 [whCode2] => 68 [distance] => 5.8443564000593)
Array ([disCode] => 555 [whCode1] => 57 [whCode2] => 65 [distance] => 6.5614723171033)
Array ([disCode] => 550 [whCode1] => 57 [whCode2] => 60 [distance] => 7.2713829583525)
Array ([disCode] => 552 [whCode1] => 57 [whCode2] => 62 [distance] => 8.8440520761848)
Pilihan warehouse root : 67

Informasi Jarak
Array ([disCode] => 553 [whCode1] => 57 [whCode2] => 63 [distance] => 0.79668326631515)
Array ([disCode] => 551 [whCode1] => 57 [whCode2] => 61 [distance] => 3.1192191730021)
Array ([disCode] => 556 [whCode1] => 57 [whCode2] => 66 [distance] => 3.3657380556275)
Array ([disCode] => 549 [whCode1] => 57 [whCode2] => 59 [distance] => 4.6837983850654)
Array ([disCode] => 557 [whCode1] => 57 [whCode2] => 67 [distance] => 4.7670233739207)
Array ([disCode] => 548 [whCode1] => 57 [whCode2] => 58 [distance] => 4.7805860853781)
Array ([disCode] => 554 [whCode1] => 57 [whCode2] => 64 [distance] => 5.3223251936144)
Array ([disCode] => 536 [whCode1] => 56 [whCode2] => 57 [distance] => 5.407483164816)
Array ([disCode] => 558 [whCode1] => 57 [whCode2] => 68 [distance] => 5.8443564000593)
Array ([disCode] => 555 [whCode1] => 57 [whCode2] => 65 [distance] => 6.5614723171033)
Array ([disCode] => 550 [whCode1] => 57 [whCode2] => 60 [distance] => 7.2713829583525)
Array ([disCode] => 552 [whCode1] => 57 [whCode2] => 62 [distance] => 8.8440520761848)
Pil warehouse runner : 65

Gambar 6. Pilihan *runner* dan *root* dari *Mother Plant*

4.3 *Fitness Function*

Fitness function adalah total hasil penjumlahan dari semua perhitungan yang dilakukan oleh sistem pada setiap kategori. Hasil dari *fitness function* juga mempertimbangkan pembobotan yang telah disesuaikan dengan kebutuhan dari perusahaan. Kategori yang digunakan dalam *fitness function* adalah perhitungan pembobotan *customer*, perhitungan pembobotan *supplier*, perhitungan *travel time*, perhitungan biaya, dan perhitungan kapasitas.

Pembobotan dari *fitness function* menentukan prioritas kategori dalam perhitungan. Kategori perhitungan pembobotan *customer* dan kategori perhitungan *travel time* mendapatkan bobot yang lebih besar daripada kategori yang lain. Hal ini dikarenakan perusahaan lebih ingin mengutamakan pertimbangan pada *customer* daripada kategori yang lain dalam penentuan lokasi *warehouse*. Sebagai contoh, dilakukan perhitungan *fitness function* seperti pada Gambar 7.

Warehouse Code	60
Warehouse Name	Brantas
Warehouse Name	Probolinggo
Biaya	0.237
Kapasitas	0.993
Pembobotan Customer	0.002
Pembobotan Supplier	0.00006
Travel Time	1.207
Fitness Value	**3.828**

Warehouse Code	56
Warehouse Name	Ketapang
Warehouse Name	Probolinggo
Warehouse Code	58
Warehouse Name	Bengawan
Warehouse Name	Probolinggo
Warehouse Code	60
Warehouse Name	Brantas
Warehouse Name	Probolinggo
Biaya	0.782
Kapasitas	0.977
Pembobotan Customer	0.001
Pembobotan Supplier	0.00004
Travel Time	0.392
Fitness Value	**2.644**

Warehouse Code	58
Warehouse Name	Bengawan
Warehouse Name	Probolinggo
Warehouse Code	60
Warehouse Name	Brantas
Warehouse Name	Probolinggo
Warehouse Code	64
Warehouse Name	Dringu
Warehouse Name	Probolinggo
Warehouse Code	65
Warehouse Name	Mayangan
Warehouse Name	Probolinggo
Warehouse Code	66
Warehouse Name	Hayam Wuruk
Warehouse Name	Probolinggo
Biaya	1.518
Kapasitas	0.956
Pembobotan Customer	0.001
Pembobotan Supplier	0.00004
Travel Time	0.329
Fitness Value	**3.240**

Gambar 7. Hasil Perhitungan *Fitness Function*

Dari hasil perhitungan, dapat dilihat bahwa semakin banyak *warehouse* tidak menunjukkan bahwa *fitness value* akan semakin baik (semakin kecil), namun apabila *warehouse* terlalu sedikit, *fitness value* juga tidak baik. Hal ini telah sesuai dengan ketentuan perusahaan agar sistem dapat menghasilkan rekomendasi yang mempertimbangkan pembobotan dan pengaruh semua parameter yang diberikan. Hasil dari *fitness function* ini juga akan menjadi penentu suatu kandidat akan tereliminasi atau akan berlanjut pada iterasi yang berikutnya.

4.4 Penilaian Aplikasi

Untuk mengetahui penilaian pengguna tentang aplikasi ini, dilakukan penelitian terhadap penggunaan aplikasi ini. Sampel dari penilaian ini adalah lima orang karyawan perusahaan, yang terdiri dari dua orang tim *sales* dan tiga orang tim logistik. Untuk mengumpulkan data, disebarkan kuesioner yang berisi indikator-indikator penilaian terhadap penggunaan aplikasi. Perihal penilaian dibagi menjadi tiga, yaitu *interface*, fungsional, dan kesimpulan, dimana setiap poin pada perihal akan dibuat detail penilaian indikator. Detail penilaian terhadap penggunaan aplikasi dapat dilihat pada Tabel 1.

Tabel 1. Tabel Penilaian terhadap Penggunaan Aplikasi

Perihal	Indikator	1	2	3	4	5
Interface	Tampilan aplikasi untuk melakukan *input* data			1	1	3
	Tampilan aplikasi untuk melihat data dalam bentuk tabel dan *map*				1	4
	Kemudahan penggunaan aplikasi				2	3
Fungsional	Kesesuaian dengan kebutuhan				1	4
	Hasil rekomendasi mampu menjawab kebutuhan				1	4
	Aplikasi berjalan sesuai harapan				2	3
Kesimpulan	Penilaian program secara keseluruhan				1	4

Keterangan skala penilaian:

- Nilai 1: Sangat buruk
- Nilai 2: Buruk
- Nilai 3: Cukup
- Nilai 4: Baik
- Nilai 5: Sangat baik

Penilaian secara keseluruhan terhadap kelayakan program:

1. Secara *interface* program 93.3% baik dan 6.7% cukup
2. Secara fungsional program 100% baik
3. Penilaian kesimpulan program untuk keseluruhan aplikasi 100% baik

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil perancangan dan pembuatan aplikasi, dapat diambil kesimpulan antara lain:

1. Semakin banyak jumlah iterasi yang dilakukan dalam melakukan pembuatan *runner* dan *root*, maka nilai *fitness value* semakin optimal sampai pada suatu kondisi ketika nilai *fitness value* telah mendekati nilai optimal.
2. Semakin banyak jumlah *customer* pada kota yang dilakukan perhitungan, maka hasil perhitungan akan semakin lama.
3. Hasil rekomendasi lebih cocok dan lebih dibutuhkan pada kota yang memiliki jumlah *customer* dan jumlah alternatif *warehouse* yang banyak karena perhitungan manual untuk memperhatikan berbagai parameter sulit dilakukan.
4. Hasil rekomendasi tidak memiliki perbedaan yang jauh dengan realita perusahaan saat ini pada kota yang memiliki jumlah *customer* yang tidak banyak.
5. Pembobotan nilai pada kategori *fitness value* menjadi salah satu faktor yang paling menentukan apakah *mother plant* akan berlanjut pada iterasi berikutnya atau tidak.
6. Pengguna aplikasi menilai keseluruhan tampilan aplikasi 93.3% baik, secara keseluruhan fungsional program 100% baik, dan keseluruhan aplikasi 100% baik. Hal ini menunjukkan bahwa sistem mampu menjawab kebutuhan perusahaan dimana aplikasi yang dikembangkan memiliki manfaat bagi perusahaan.

5.2 Saran

Saran yang dapat diberikan untuk menyempurnakan dan mengembangkan aplikasi ini lebih lanjut antara lain:

1. Pengambilan data yang langsung terkoneksi dengan *database real* sehingga data dalam sistem dapat langsung diperbaharui tanpa pengguna perlu melakukan unggah data kembali.
2. Penambahan ruang lingkup data untuk *customer* perusahaan pada kota-kota yang lain.
3. Studi lebih lanjut dan mendalam mengenai penentuan parameter dan pembobotan parameter pada awal pembuatan aplikasi agar konstanta pada pembobotan parameter sesuai.

6. DAFTAR PUSTAKA

[1] Bayat, F. M. 2015. *The Runner-Root Algorithm: A Metaheuristic for Solving Unimodal and Multimodal Optimization Problems Inspired by Runners and Root of Plants in Nature*. *Journal of Science Direct*, 33: 292-303.

[2] Brummelen, G. V. 2014. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry.* New Jersey: Princeton University Press.

[3] Chitkara, K. K. 2014. *Construction Project Management.* New Delhi: McGraw Hill Education.

[4] Haddad, O. B., Solgi, M., & Loaiciga, H. A. 2017. *Meta-heuristic and Evolutionary Algorithms for Engineering Optimization.* New York: John Wiley&Sons Inc

[5] Rajpurohit, J., Sharma, T. K., Abraham, A., & Vaishali. 2017. *Glossary of Metaheuristic Algorithms*. *International Journal of Computer Information Systems and Industrial Management Applications*. 9:181-205.

[6] Shabot, M., & Gardner, R. 2012. *Decision Support System in Critical Care.* Los Angeles: Springer-Verlag.