Algoritma *Goal Programming* untuk *Driver Assignment* pada Simulasi Taksi *Online*

Lienny Ferlinda
Program Studi Informatika
Fakultas Teknologi Industri
Universitas Kristen Petra
Jl. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
liennyferlinda@gmail.com

Andreas Handojo
Program Studi Informatika
Fakultas Teknologi Industri
Universitas Kristen Petra
Jl. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
handojo@petra.ac.id

Tanti Octavia
Program Studi Industri
Fakultas Teknologi Industri
Universitas Kristen Petra
Jl. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
tanti@petra.ac.id

ABSTRAK

Taksi adalah salah satu alat transportasi umum yang banyak digunakan oleh masyarakat. Pemasangan *driver* dan penumpang dalam taksi online dapat dilakukan dengan cara menawarkan *order* penumpang kepada seluruh *driver* terdekat dari lokasi penjemputan. Cara ini memiliki efisiensi waktu tinggi. Namun, hal ini menyebabkan naiknya tingkat pembatalan karena *driver* tidak memiliki cukup waktu untuk melihat detail pesanan, hanya asal menerima pesanan saja. Hal ini dapat berujung pada menurunnya tingkat kepuasan penumpang dan pendapatan taksi online. Oleh karena itu, faktor-faktor lain seperti *rating driver*, tingkat pembatalan pesanan *driver*, jumlah *order* yang sudah diselesaikan *driver* penting untuk dipertimbangkan dalam proses pemasangan untuk menghasilkan pemasangan yang efisien.

Proses pemasangan *driver* dan penumpang akan dilakukan dengan metode *Goal Programming* karena metode ini cocok digunakan untuk masalah dalam pengambilan keputusan yang melibatkan lebih dari satu tujuan (*multi-objectives*).

Hasil penelitian menunjukkan bahwa *Goal Programming* menghasilkan total waktu kalkulasi tertinggi. Selain itu, rata-rata waktu tunggu penumpang dan jarak tempuh penjemputannya terendah. Metode *Hungarian Algorithm* memiliki waktu kalkulasi yang lebih cepat dibandingkan dengan metode *Goal Programming*, Namun, jumlah pemasangannya lebih rendah. Selain itu, rata-rata waktu tunggu penumpang dan jarak tempuh penjemputannya lebih tinggi. Metode *Random Assignment* memiliki waktu kalkulasi tercepat dan tingkat keberhasilan pemasangan tertinggi. Namun, rata-rata waktu tunggu penumpang dan jarak tempuh penjemputannya jauh di atas kedua metode pembandingnya. Penambahan faktor di luar waktu dan jarak jemput dalam persentase yang rendah dapat mempengaruhi nilai rata-rata faktor tersebut menjadi lebih baik. Namun, rata-rata waktu dan jarak tempuh penjemputannya menjadi bertambah.

Kata Kunci: Goal Programming, Hungarian Algorithm, Taksi Online, Pemasangan

ABSTRACT

Taxi is one of the most common means of public transportation. Assignment of drivers to passengers in online taxi can be done by offering passenger orders to all drivers closest to the pick-up location. This method has high time efficiency. However, this causes an increase in the cancellation rate because drivers don't have enough time to view order details, just as long as they accept orders. This can lead to a decrease in the level of passenger satisfaction and online taxi revenue. Therefore, other factors such as driver rating, driver order cancellation rate, number of orders completed by drivers are important to consider in the assignment process to produce an efficient assignment.

The process of assigning drivers and passengers will be carried out using the Goal Programming method because this method is suitable for problems in decision making that involve more than one goal (multi-objectives).

The results showed that Goal Programming resulted in the highest total calculation time. In addition, the average waiting time for passengers and pick-up distance are the lowest. The Hungarian Algorithm method has a faster calculation time compared to the Goal Programming method, however, the number of assignment is lower. In addition, the average waiting time for passengers and their pick-up distances is higher. The Random Assignment method has the fastest calculation time and the highest assignment success rate. However, the average waiting time for passengers and the distance to pick them up is far above the two comparison methods. The addition of factors other than time and pick-up distance in a low percentage can affect the average value of these factors for the better. But, the average pick-up time and distance is increasing.

Keywords: *Goal Programming, Hungarian Algorithm, Online Taxi, Assignment*

1. LATAR BELAKANG

Taksi adalah salah satu alat transportasi umum yang banyak digunakan oleh masyarakat. Pada awalnya, taksi hanya bisa dipesan melalui telepon dan pemberhentian di jalan. Namun saat ini, taksi dapat dipesan secara online melalui aplikasi dalam *smartphone*. Taksi *online* memiliki berbagai kelebihan dibanding pendahulunya taksi konvensional, diantaranya adalah calon penumpang tidak perlu menghabiskan pulsa untuk telepon, harga yang sudah ditentukan sebelum proses pemesanan, tersedianya beragam opsi pembayaran dan calon penumpang dapat melihat posisi *driver* dalam peta.

Simulasi taksi online merupakan sebuah sistem yang dibuat berdasarkan ketentuan-ketentuan tertentu sehingga dapat melakukan *generate driver* dan penumpang, melakukan pemasangan, *auto save data*, hingga visualisasi kemunculan *driver* dan penumpang dalam peta. Simulasi taksi *online* yang dibuat oleh Eryn dan Liadi dalam penelitian sebelumnya dapat melakukan *generate driver* dan penumpang berdasarkan *random* lokasi dan waktu tertentu tanpa memperhatikan adanya kecenderungan suatu area lokasi ramai pada jam tertentu saja misalnya pada jam berangkat kantor akan banyak *order* menuju

pusat kota dan perkantoran, sedangkan sebaliknya pada jam pulang kantor akan banyak *order* dari pusat kota dan perkantoran. Selain itu, faktor-faktor yang diperhitungkan terbatas pada waktu penjemputan, jarak penjemputan dan jumlah *trip* yang dilakukan *driver*, sehingga *admin* tidak bisa melakukan percobaan dan analisis terhadap pengaruh faktor lain.

Berbagai metode pemasangan seperti *Hungarian Algorithm, Tabu Search*, dan *Branch and Bound* telah diusulkan dalam penelitian sebelumnya. *Hungarian Algorithm* dapat memberikan hasil *assignment* optimal untuk matriks biaya. *Tabu Search* dapat menghasilkan solusi terbaik dari semua kemungkinan solusi yang dieksekusi. *Tabu Search* telah terbukti efektif di berbagai masalah optimasi seperti pewarnaan grafik dan masalah *salesman*, dan juga telah diterapkan dalam masalah praktis seperti penjadwalan dan desain sirkuit elektronik [2]. Namun, *Tabu Search* memiliki kelemahan yaitu jumlah iterasinya bisa sangat tinggi [8]. Hasil perhitungan algoritma *Branch and Bound* lebih baik jika dibandingkan dengan *Tabu Search* dan *Hungarian Algorithm*. Namun, algoritma *Branch and Bound* memiliki waktu komputasi yang lama karena jumlah *nodes* dalam percabangan *tree* yang terlalu banyak.

Pemasangan *driver* dan penumpang dalam taksi *online* dapat dilakukan dengan cara menawarkan *order* penumpang kepada seluruh *driver* terdekat dari lokasi penjemputan. Cara ini cukup mudah dan memiliki efisiensi waktu yang tinggi. Namun, hal ini menyebabkan naiknya tingkat pembatalan pesanan karena *driver* tidak memiliki cukup banyak waktu untuk melihat detail pesanan, hanya asal menerima pesanan saja. Hal ini dapat berujung pada menurunnya tingkat kepuasan penumpang dan pendapatan taksi *online* [5]. Oleh karena itu, faktor-faktor lain seperti *rating driver*, tingkat pembatalan pesanan *driver*, jumlah order yang sudah diselesaikan *driver* di hari yang sama, dan lain sebagainya penting untuk dipertimbangkan dalam proses pemasangan untuk menghasilkan pemasangan yang efisien.

Simulasi taksi *online* yang dapat menerima *input* berbagai faktor yang dapat diatur oleh *admin* dapat memudahkan *admin* untuk melakukan percobaan terhadap berbagai faktor-faktor lain. Dalam menyelesaikan *assignment* problem dengan *multi-objectives goal*, maka dalam skripsi ini diusulkan metode *Goal Programming*. Dalam algoritma yang telah digunakan sebelumnya, *Hungarian Algorithm* memiliki keterbatasan yaitu batasan yang tidak relatif. *Goal Programming* cocok digunakan untuk masalah dalam pengambilan keputusan yang melibatkan lebih dari satu tujuan (*multi-objectives*) karena melalui variabel deviasinya, *Goal Programming* secara otomatis menangkap informasi tentang pencapaian relatif dari tujuan-tujuan yang ada dengan meminimalkan deviasi agar tidak terlalu jauh dari ketentuan/target [1]. Dalam metode ini, terlebih dahulu dapat ditentukan prioritas atau bobot dari masing-masing tujuan yang ingin dicapai. Melalui metode ini, faktor-faktor lain yang dapat mempengaruhi proses pemasangan *driver* dan penumpang dapat digunakan sebagai tujuan/*goal* dalam metode *Goal Programming*.

Pada skripsi ini, akan berfokus pada pembuatan simulasi taksi online berbasis *website* yang dapat melakukan *generate driver d*an penumpang sesuai ketentuan *random* lokasi dan probabilitas kemunculan yang telah ditetapkan serta menyimpan data simulasi secara otomatis. *Website* ini juga memiliki berbagai fitur untuk mendukung proses pemasangan *driver* dan penumpang dengan memperhatikan berbagai tujuan dan prioritasnya yang sebelumnya telah ditentukan oleh *admin*. Dalam simulasi ini, tingkat pembatalan pesanan dan *rating driver* juga akan diperhitungkan sehingga simulasi menjadi lebih realistis. Selain itu, sistem dapat menerima *input data driver* dan penumpang dalam bentuk excel sehingga memudahkan *admin* untuk mengedit dan menambahkan data di luar program. Proses pemasangan *driver* dan penumpang dilakukan dengan metode *Goal Programming* yang akan dibandingkan rata-rata waktu kalkulasinya dengan metode *Hungarian Algorithm* dan *Random Assignment.*

2. LANDASAN TEORI

2.1 *Assignment Problem*

Assignment Problem berkaitan dengan bagaimana menetapkan n objek ke m objek lain dengan cara terbaik [10]. Hal ini diperlukan untuk melakukan tugas sebanyak mungkin dengan menugaskan paling banyak satu agen untuk setiap tugas dan paling banyak satu tugas untuk setiap agen, sedemikian rupa sehingga total biaya penugasan diminimalkan. *Driver Assignment Problem* juga merupakan salah satu masalah dalam transportasi yaitu proses memasangkan *driver* dan penumpang sehingga biaya yang yang dikeluarkan seminimal mungkin [11].

2.2 *Goal Programming*

Goal Programming merupakan sebuah metode yang pertama kali dikembangkan oleh Charnes dan Cooper pada awal tahun 1960 dan dikembangkan lagi oleh Ijiri, Lee, dan Ignizio. *Goal Programming* adalah metode dalam pengambilan keputusan dengan berbagai kriteria [12]. *Goal Programming* bertujuan untuk meminimalkan penyimpangan-penyimpangan dari tujuan-tujuan tertentu.Model *Goal Programming* paling sedikit terdiri dari tiga komponen yaitu: fungsi tujuan, kendala tujuan dan kendala non-negatif. Berikut ini adalah langkah-langkah perumusan *Goal Programming:*

1) Menentukan variabel keputusan
2) Menyatakan sistem kendala
3) Menentukan nilai-nilai sisi kanan dan menentukan koefisien teknologi yang cocok dan variabel keputusan yang diikutsertakan dalam kendala.
4) Menentukan prioritas
5) Menentukan urutan prioritas dari tujuan-tujuan yang ada.
6) Menentukan bobot
7) Membuat penilaian terhadap deviasi dari tujuan-tujuan yang ada. Jika persoalan tidak memiliki urutan tujuan, langkah ini dapat dilewati.
8) Menyatakan fungsi tujuan
9) Memilih variabel simpangan/deviasi yang akan dimasukkan ke dalam fungsi tujuan.
10) Menyatakan keperluan non-negatif. Langkah ini merupakan bagian resmi dari perumusan masalah *Goal Programming*.

2.3 *Hungarian Algorithm*

Hungarian Algorithm adalah algoritma yang dapat digunakan untuk memecahkan masalah *assignment* yang pertama kali ditemukan dan dipublikasikan oleh Harold Kuhn pada tahun 1995 [6]. Untuk dapat menerapkan Metode Hungarian, jumlah objek yang akan harus ditugaskan harus sama dengan jumlah objek lainnya sehingga dapat membentuk n x n matriks berbobot. Berikut ini adalah langkah-langkah dari *Hungarian Algorithm* untuk masalah minimasi:

1) Kurangi semua nilai dari setiap kolom dalam matriks n x n dengan angka terkecilnya.
2) Kurangi semua nilai dari setiap kolom dalam matriks n x n dengan nilai terkecilnya jika kolom tersebut belum mempunyai nilai 0.

3) Periksa apakah nilai 0 telah diperoleh di dalam setiap baris dan kolom setelah penugasan, algoritma bisa diberhentikan karena penugasan optimal telah diperoleh.
4) Dilakukan penutupan semua nilai 0 dengan menggunakan garis vertikal dan horizontal seminimal mungkin.
5) Ditentukan nilai terkecil dari nilai-nilai yang tidak tertutup garis. Lalu semua nilai yang tidak tertutup garis dikurangkan dengan nilai terkecil tersebut.
6) Kembali ke langkah 3.

2.4 *Random Assignment*

Random Assignment merupakan modul Python yang mengimplementasikan *pseudo-random number generators* untuk berbagai distribusi. Python menggunakan Mersenne Twister sebagai *core generator* [9]. Mersenne Twister adalah salah satu generator nomor acak yang paling banyak digunakan. Namun, penggunaanya tidak cocok untuk semua tujuan terutama tujuan kriptografi.

Fungsi yang akan digunakan untuk proses *Random Assignment* adalah random.randint(a,b) yang akan menghasilkan angka acak N dengan tipe data integer yang berada pada rentang angka a <= N <= b. Setiap *order* penumpang akan dipasangkan dengan *driver* dari *list driver* dengan index ke-i dimana i adalah hasil dari fungsi random.randint(0, jumlah driver-1).

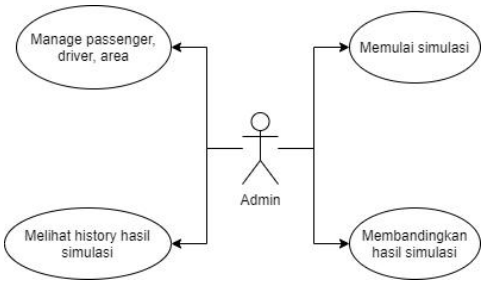
2.5 Penelitian Sebelumnya

Penelitian mengenai pemasangan sudah pernah dibuat sebelumnya. Penelitian oleh Hassan mengangkat masalah penentuan penerimaan siswa untuk mengoptimalkan alokasi siswa ke berbagai fakultas. Metode yang diusulkan dalam penelitian ini adalah *Preemptive Goal Programming*. Hasil dari penggunaan metode G*oal Programming* terbukti dapat menyesuaikan kebutuhan universitas untuk memenuhi persyaratan dengan prioritas tertinggi sesuai dengan bobotnya [4]. Selain itu, terdapat penelitian oleh Zhang mengenai pemasangan *driver* dan *order* pada taksi. Metode yang diusulkan dalam penelitian ini adalah *novel combinatorial optimization*. Hasil dari penelitian ini adalah sistem *dispatch* berdasarkan *combinatorial optimization*. Dari model ini bisa didapatkan estimasi probabilitas dari *driver* menerima sebuah *order*. Tujuan dari penelitian ini adalah memaksimalkan *global success rate* dan memberikan *user experience* yang terbaik. Selain itu, program juga dapat memprediksi daftar tujuan *user* saat *user* membuka aplikasi [13]. Dalam penelitian lain yang dilakukan Liadi yang menggunakan metode *branch and bound* untuk mencapai *optimal cost* dalam pemasangan *driver* dan *passenger*. Hasil penelitian menunjukkan bahwa Algoritma *Branch and Bound* memiliki waktu kalkulasi yang terbesar dibandingkan dengan algoritma lain seperti tabu search dan *Hungarian Algorithm.* Namun, hasil perhitungannya masih lebih baik dibandingkan dengan algoritma *Tabu Search* [7]. Sementara itu, pada penelitian lain oleh Eryn, dilakukan perbandingan metode *Hungarian Algorith*m dan *Tabu Search* untuk melakukan proses matching yang optimal dengan menggunakan waktu dan sumber daya seminimal mungkin. Hasil penelitian ini menunjukkan bahwa *Hungarian Algorithm* memiliki waktu kalkulasi paling kecil. Seiring bertambahnya data, kedua algoritma ini memiliki kenaikan waktu kalkulasi yang kecil [3]. Namun, dalam penelitian Eryn dan Liadi, faktor yang digunakan hanya waktu, jarak dan jumlah *trip* selain itu, masih belum bisa ditentukan prioritas pembobotannya.

3. DESAIN SISTEM

3.1 *Use Case Diagram*

Use case diagram adalah gambaran dari interaksi yang dapat dilakukan oleh aktor pada sebuah sistem. Seperti dalam Gambar 1, dalam simulasi *driver assignment* ini hanya terdapat satu aktor yang berperan yaitu *admin*. *Admin* dapat melakukan beberapa aktivitas seperti memulai simulasi, membandingkan hasil simulasi, melihat *history* hasil simulasi, dan *manage passenger, driver* dan *area.*



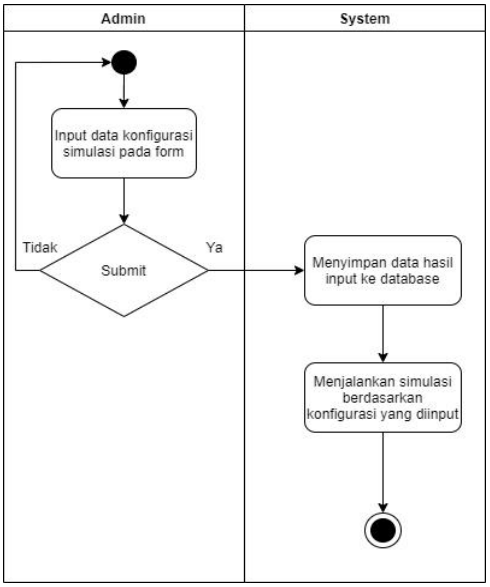
Gambar 1. *Use Case Diagram*

3.2 *Activity Diagram*

Activity diagram merupakan gambaran alur dari sebuah sistem. *Activity diagram* di bawah ini bertujuan untuk menjelaskan aktivitas yang tertera pada Gambar 1. Dalam sub bab ini akan dijelaskan empat aktivitas yaitu memulai simulasi, melihat *history* hasil simulasi, membandingkan hasil simulasi, dan *manage passenger, driver* dan *area.*

3.2.1 Activity Diagram Memulai Simulasi

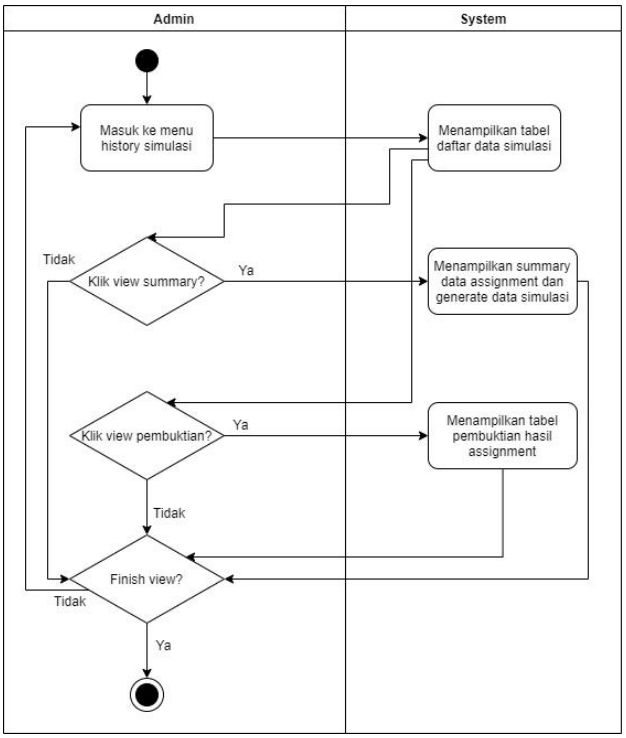
Sesuai dengan Gambar 2, untuk menjalankan simulasi, pertama-tama admin perlu *input* data konfigurasi yang dibutuhkan seperti nama simulasi, metode, *start date*, *end date*, *start time*, *end time*, faktor-faktor apa saja yang ingin diperhitungkan beserta persentase prioritasnya serta memilih area apa saja yang akan digunakan. Selanjutnya, admin menekan tombol *submit* dan *system* akan menyimpan data konfigurasi ke *database* dan menyambungkan ke halaman simulasi dimana simulasi akan dijalankan berdasarkan data konfigurasi yang telah disimpan.



Gambar 2. *Activity Diagram* Memulai Simulasi

3.2.2 Activity Diagram Melihat History Hasil Simulasi

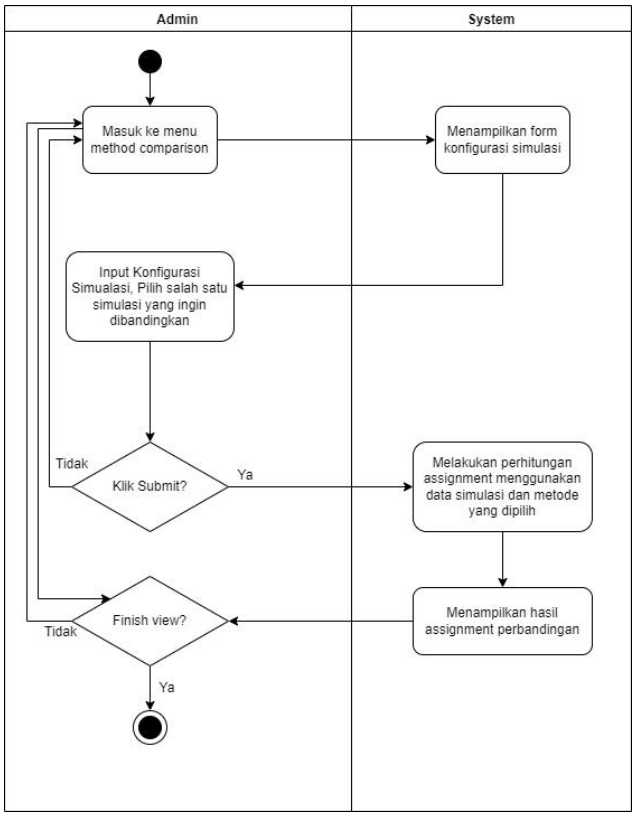
History hasil simulasi dapat dilihat oleh *admin* dengan masuk ke menu *history* simulasi lalu, *system* akan *fetch* daftar riwayat simulasi dari *database* dan menampilkannya dalam bentuk tabel. Lebih lanjut alur dapat dilihat pada Gambar 3.



Gambar 3. *Activity Diagram* Melihat *History* Hasil Simulasi

3.2.3 Activity Diagram Membandingkan Hasil Simulasi

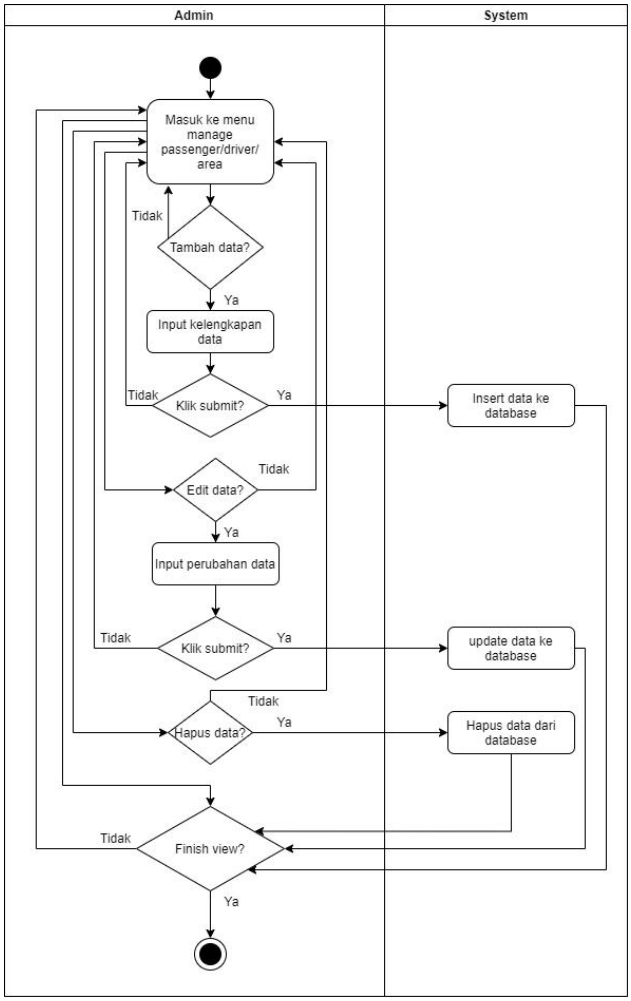
Seperti pada Gambar 4, langkah ini dimulai ketika *admin* masuk ke dalam menu *method comparison*, *admin* memilih salah satu satu simulasi dan metode pembanding yang diinginkan. Selanjutnya, tekan tombol *submit* dan sistem akan melakukan proses *assignment* berdasarkan data *generate* pada simulasi yang dipilih sebelumnya.



Gambar 4. *Activity Diagram* Membandingkan Hasil Simulasi

3.2.4 Activity Diagram Manage Passenger, Driver, Area

Manage data passenger/driver/area dapat dilakukan oleh admin dengan masuk ke halaman *manage passenger/driver/area*, lalu *system* akan *fetch* data yang diminta dari *database* dan menampilkannya dalam bentuk tabel berisi data terkini dari menu yang dipilih. Dalam halaman ini, admin dapat melakukan *insert, update, delete* data. Selain itu, admin juga bisa melakukan *import* dan *export* data dalam bentuk *file Excel. Activity diagram* dapat dilihat pada Gambar 5.



Gambar 5 ***Activity Diagram Manage Passenger, Driver, Area***

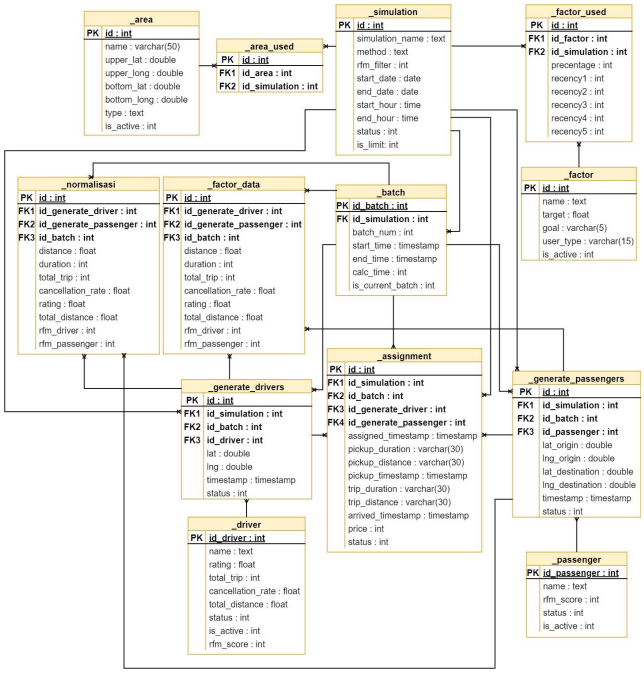
3.3 *Flowchart*

Flowchart pada Gambar 6 di bawah ini akan menggambarkan proses yang harus dilakukan dalam pemasangan *driver* dan *passenger*. Proses yang dapat dilakukan antara lain: *generate driver* dan *passenger*, menghitung seluruh faktor yang digunakan menjadi matriks biaya, melakukan *assignment* dengan algoritma, dan menyimpan hasil pemasangan ke *database*.

Gambar 6. ***Flowchart Assignment Driver*** **dengan** ***Passenger***

3.4 Desan *Database*

Entity relationship diagram adalah sebuah model untuk menggambarkan hubungan antar entitas dalam sebuah sistem. Model ERD yang digambarkan pada Gambar 7 menunjukkan bahwa terdapat 12 entitas yaitu _area, _area_used, _assignment, _batch, _driver, _factor, _factor_data, _factor_used, _generate_drivers, _generate_passengers, _passenger, dan _simulation.



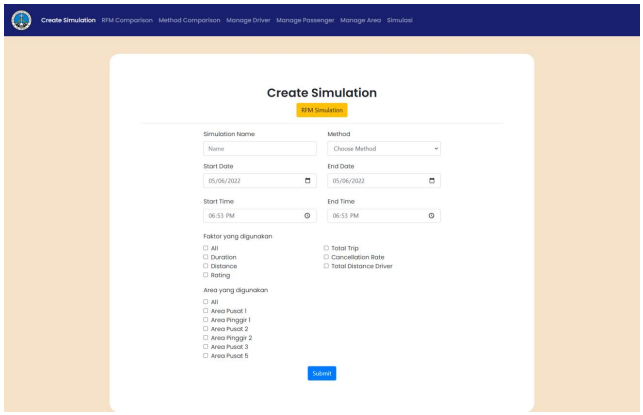
Gambar 7. *Entity Relationship Diagram*

3.5 Desain *Interface*

Desain *interface* menunjukkan desain visual bagaimana simulasi berjalan. *Platform* yang digunakan berbasis *website*.

3.5.1 Halaman New Simulation

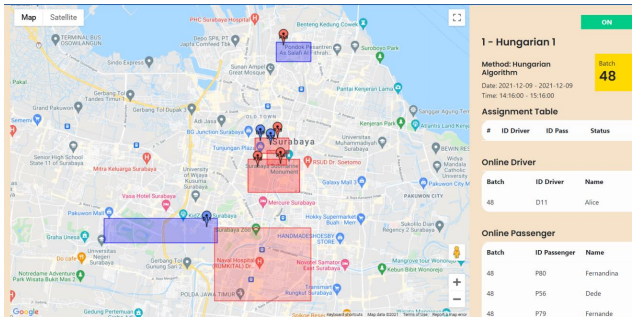
Halaman *New Simulation* berisi *form* yang berisi pilihan konfigurasi untuk memulai simulasi baru. *Input* yang harus diisi diantaranya yaitu: nama simulasi, metode simulasi, tanggal mulai dan selesai, waktu mulai dan selesai, faktor-faktor yang akan digunakan dalam perhitungan *assignment*, serta area yang digunakan. Desain halaman ini dapat dilihat pada Gambar 8.



Gambar 8. Halaman *New Simulation*

3.5.2 Halaman Simulasi

Halaman simulasi menampilkan jalannya simulasi secara *realtime* berdasarkan konfigurasi yang sudah dimasukkan sebelumnya. Di halaman ini, *admin* dapat melihat informasi simulasi yang sedang berjalan, peta dengan area yang diwakili bentuk kotak berwarna biru untuk daerah pinggir dan kotak berwarna merah untuk daerah pusat. Selain itu, *admin* juga bisa melihat persebaran *driver* dan *passenger* yang sedang *online*, rute penjemputan, rute perjalanan, tabel *assignment*, tabel *online driver* dan *passenger*. Halaman visualisasi dapat dilihat pada Gambar 9.



Gambar 9. Halaman Simulasi

4. PENGUJIAN SISTEM

4.1 Uji Coba 1

Pengujian ini dilakukan pada tanggal 16 Mei 2022 pukul 22:17 WIB. Pengujian ini digunakan untuk mengetahui perbandingan hasil pemasangan yang didapatkan dari 3 algoritma yang berbeda pada jam normal. Simulasi dilakukan dalam 60 *batch* dengan durasi waktu 1 menit per *batch*. Faktor yang digunakan adalah *pick-up duration* (batas 10 menit) dan *pick-up distance* (batas 3.5 km) dengan bobot masing-masing 50%. Jumlah *driver* dan *passenger* dibangkitkan secara acak dengan range 1-5 orang setiap *batch*-nya.

Hasil Uji Coba 1 menunjukkan bahwa metode *Goal Programming* menghasilkan total dan rata-rata waktu kalkulasi tertinggi. Namun, menghasilkan rata-rata waktu tunggu terendah dan rata-rata jarak tempuh penjemputan terendah. Metode *Hungarian Algorithm* memiliki rata-rata waktu kalkulasi lebih baik dari *Goal Programming*. Namun, menghasilkan rata-rata waktu tunggu dan rata-rata jarak tempuh penjemputan yang lebih besar dari *Goal Programming*. Sedangkan, metode *Random Assignment* menghasilkan jumlah pemasangan terbanyak dengan rata-rata waktu kalkulasi tercepat. Namun, menghasilkan rata-rata waktu tunggu terlama dan rata-rata jarak tempuh penjemputan terbesar.

Dari uji coba ini dapat disimpulkan bahwa metode *Goal Programming* memiliki waktu kalkulasi yang paling lama, namun dapat menghasilkan hasil pemasangan yang lebih baik. Hal tersebut dapat dilihat dari rata-rata waktu tunggu penumpang dan rata-rata jarak tempuh penjemputan yang dihasilkan lebih rendah dibandingkan dua metode pembandingnya. Selain itu, diketahui bahwa metode *Random Assignment* memiliki waktu kalkulasi yang sangat cepat dan menghasilkan jumlah pemasangan terbesar namun, waktu dan jarak tempuh dalam penjemputan jauh lebih besar dibandingkan dua metode lainnya.

4.2 Uji Coba 2

Pengujian ini dilakukan pada tanggal 23 Mei 2022 pukul 17:00 WIB. Pengujian ini digunakan untuk menganalisa hasil pemasangan berkaitan dengan jumlah *driver* dan *passenger* pada jam sibuk seperti jam berangkat dan pulang kerja. Simulasi ini terdiri atas 30 *batch* dengan durasi 1 menit setiap *batch*. Jumlah *driver* dan *passenger* diacak dengan *range* 6-10 per *batch*-nya. Sementara untuk pengacakan lokasi kemunculan *driver* dan *passenger* yaitu sebanyak 70% kemungkinan kemunculan *driver* dan *passenger* dari pusat kota, sementara sisanya 30% dari pinggir kota. Faktor yang digunakan adalah durasi (batas 10 menit) dan jarak tempuh penjemputan (batas 3.5 km) dengan bobot 50%.

Hasil Uji Coba 2 menunjukkan bahwa metode *Goal Programming* memiliki rata-rata waktu kalkulasi terlama, namun menghasilkan rata-rata waktu tunggu penumpang dan rata-rata jarak penjemputan terendah. Metode *Hungarian Algorithm* memiliki rata-rata waktu kalkulasi lebih cepat daripada *Goal Programming*. Namun, jumlah pemasangan yang dihasilkan paling sedikit. Sedangkan, metode *Random Assignment* memiliki rata-rata waktu kalkulasi tercepat dan jumlah pemasangan paling banyak.

5. KESIMPULAN

Simulasi dapat dijalankan sesuai dengan konfigurasi yang diinputkan *admin* di *form* pembuatan simulasi. Simulasi yang sedang berlangsung dapat divisualisasikan dalam *maps*, dengan adanya *marker driver* dan *passenger* serta *route pick-up* maupun saat sedang berjalan menuju ke lokasi tujuan. Selain itu, *website* dapat menampilkan riwayat simulasi dalam bentuk tabel maupun grafik diantaranya berisi jumlah pemasangan, waktu kalkulasi, rata-rata dan total waktu jemput serta rata-rata dan total jarak tempuh penjemputan. Metode *Goal Programming* menghasilkan total waktu kalkulasi tertinggi namun, tingkat keberhasilan pemasangannya tertinggi. Selain itu, rata-rata waktu tunggu penumpang serta rata-rata jarak tempuh penjemputannya terendah. Metode ini cocok digunakan untuk pemasangan yang membutuhkan beberapa faktor yang memiliki batasan tertentu. Metode *Hungarian Algorithm* memiliki waktu kalkulasi yang lebih cepat dibandingkan dengan metode *Goal Programming*, namun jumlah pemasangannya lebih rendah. Selain itu, rata-rata waktu tunggu penumpang serta rata-rata jarak tempuh penjemputannya lebih tinggi dibandingkan dengan metode *Goal Programming.* Metode ini lebih cocok digunakan untuk proses pemasangan yang tidak menggunakan batasan tertentu untuk faktor-faktornya sehingga, h*ungarian programming* dapat menghasilkan rata-rata waktu dan jarak yang lebih baik karena waktu kalkulasinya yang relatif lebih cepat. Sedangkan, metode *Random Assignment* memiliki waktu kalkulasi tercepat dan tingkat keberhasilan pemasangan tertinggi. Namun, rata-rata waktu tunggu penumpang dan jarak tempuh penjemputannya jauh di atas kedua metode pembandingnya. Sehingga, tidak cocok digunakan dalam proses pemasangan ini.

6. SARAN

Mencari alternatif lain untuk menggantikan Google Maps API, karena tingginya biaya yang akan dihabiskan untuk *request* perhitungan *distance* dan *duration*. Untuk simulasi dengan faktor lain di luar waktu dan jarak tempuh dilakukan dalam jumlah *batch* yang lebih banyak sehingga akan mendapatkan analisis yang lebih baik lagi. Perlunya penambahan fitur halaman statistik yang membuat *user* bebas bisa memilih sendiri simulasi yang ingin dilihat perbandingannya melalui tabel atau pun grafik.

7. REFERENSI

[1] Chowdary, B. V., & Jannes, S. 2002. Production planning under dynamic product environment: a multi-objective goal programming approach. The Annual Research Report.

[2] Connor, A.M. & Shea, K. 2000. A comparison of semi-deterministic and stochastic search techniques. I.C. Parmee [Ed.] Evolutionary Design & Manufacture (Selected Papers from ACDM'00), 287-298. Springer-Verlag: London.

[3] Eryn. 2020. Perbandingan metode Tabu Search dan metode Hungarian Algorithm untuk penentuan Driver Assignment pada simulasi taksi online. URI=https://dewey.petra.ac.id/catalog/digital/detail?id=46347

[4] Hassan, N. 2016. A Preemptive Goal Programming for Allocating Students into Academic Departments of a Faculty. International Journal of Mathematical Models and Method Applied Sciences, 10, 166-170.

[5] Jiang, et al. 2020. A Mutual Selection Mechanism of Ride-Hailing Based on Hidden Points, Wireless Communications and Mobile Computing, vol. 2020. DOI=https://doi.org/10.1155/2020/9520384.

[6] Kuhn, H. W. 1955. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1-2), 83–97. DOI:10.1002/nav.3800020109.

[7] Liadi. 2020. Algoritma branch and bound untuk driver assignment pada aplikasi simulasi taksi online. URI=https://dewey.petra.ac.id/catalog/digital/detail?id=46372.

[8] Liang, F. 2020. Optimization Techniques — Tabu Search. Towardsdatascience. URI=https://towardsdatascience.com/optimization-techniques-tabu-search-36f197ef8e25

[9] Python. (n.d.). random — Generate pseudo-random numbers. URI=https://docs.python.org/3/library/random.html.

[10] Singh, S., Dubey, G. C., & Shrivastava, R. 2012. A various method to solve the optimality for the transportation problem. International Journal of Mathematical Engineering and Science, 1(4), 21-28.

[11] Spliet, R., & Dekker, R. 2016. The driver assignment vehicle routing problem. Networks, 68(3), 212–223. DOI:10.1002/net.21694.

[12] Spronk J. 1981. International Series in Management Science/Operations Research book series. Springer, Dordrecht. DOI=https://doi.org/10.1007/978-94-009-8165-2_4

[13] Zhang, et al. 2017. A Taxi Order Dispatch Model based On Combinatorial Optimization. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 2151-2159). KDD 17. DOI:10.1145/3097983.3098138.