Perbandingan Metode Tabu Search dengan Metode Hungarian Algorithm untuk Penentuan Driver Assignment pada Simulasi Taksi Online

Eryn, Andreas Handjojo, Tanti Octavia
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: erynchandra@gmail.com, Handojo@petra.ac.id, tanti@petra.ac.id

ABSTRAK

Taksi merupakan salah satu transportasi umum yang banyak digunakan oleh masyarakat di seluruh belahan dunia. Seiring berjalannya waktu, taksi terus berkembang hingga sekarang ini muncul taksi *online*. Taksi juga berkembang dalam fungsi dispatch dalam pemasangan *driver* dengan *request* dari pelanggan. Dengan berkembangnya fungsi pemasangan *driver* ke *request* ini, maka parameter-parameter yang digunakan untuk mendapatkan hasil *matching* juga semakin bertambah, dan kompleksitasnya semakin meningkat. Namun, dengan kondisi-kondisi yang semakin berkembang, proses *matching* antar *driver* dan pelanggan tetap dituntut untuk menggunakan waktu dan sumber daya seminimal mungkin dan tetap menghasilkan hasil yang optimal. Maka dari itu, pada proses *matching* ini dilakukan banyak perkembangan agar mendapat hasil pemasangan *driver* ke pelanggan yang dapat meminimalkan waktu dan jarak.

Aplikasi dibuat dengan bahasa pemrograman python, bootstrap 3 untuk *front end menu*, dan django sebagai *framework*. Google Maps API digunakan untuk mendapatan nilai waktu dan jarak penjemputan dan sebagai *front end* pada bagian peta *online*. MySQL digunakan sebagai *database* untuk menyimpan data-data simulasi.

Hasil akhir dari program ini yaitu suatu aplikasi simulasi *assignment* antara *driver* dan *passenger* dengan menggunakan dua algoritma dan 3 faktor, yaitu waktu, jarak, dan *last trip* oleh *driver*. Simulasi dapat menampilkan perbandingan dari hasil *assignment* untuk memudahkan proses perbandingan algoritma.

Kata Kunci: *hungarian algorithm, tabu search, google maps api,* taksi *online,* simulasi

ABSTRACT

Taxi in one of the public transportation that is widely used by people in all parts of the world. Over time, taxis have continued to grow until now online taxis have emerged. Taxis also develop in the dispatch function in installing drivers with requests from customers. As the function of installing drivers to this request develops, the parameters used to get matching results also increase and the complexity increases. However, with the conditions that are increasingly developing, the matching process between drivers and customers is still required to use the least amount of time and resources and still produce optimal results. So from this, in this matching process a lot of progress was made in order to get the results of installing drivers to customers that can minimize time and distance.

The application is made in Python programming language, Bootstrap 3 for the front end menu, and django as a framework. The Google Maps API is used to get time and distances and as a front end in the online map section. MySQL is used as a database to store simulation data.

The final result of this program is an assignment simulation application between driver and passenger using two algorithms and 3 factors, that is time, distance and last trip by the driver. Simulation can display a comparison of the results of the assignment to facilitate the process of comparison of algorithms.

Keywords: *hungarian algorithm, tabu search, google maps api, taxi online, simulation.*

1. PENDAHULUAN

Taksi merupakan salah satu transportasi umum yang banyak digunakan oleh masyarakat di seluruh belahan dunia. Seiring berjalannya waktu, taksi terus berkembang hingga sekarang ini muncul taksi *online*. Taksi juga berkembang dalam fungsi *dispatch* dalam pemasangan *driver* dengan *request* dari pelanggan. Dengan berkembangnya fungsi pemasangan *driver* ke *request* ini, maka parameter-parameter yang digunakan untuk mendapatkan hasil *matching* juga semakin bertambah, dan kompleksitasnya semakin meningkat. Namun, dengan kondisi-kondisi yang semakin berkembang, proses *matching* antar *driver* dan pelanggan tetap dituntut untuk menggunakan waktu dan sumber daya seminimal mungkin dan tetap menghasilkan hasil yang optimal. Maka dari itu, pada proses *matching* ini dilakukan banyak perkembangan agar mendapat hasil pemasangan *driver* ke pelanggan yang dapat meminimalkan waktu dan jarak.

Telah dilakukan beberapa penelitian dalam memecahkan masalah *driver assignment*, dengan metode seperti Hungarian Algorithm dan *Minimum Value of Selected Factor in Data Collection* (MVSF) yang disesuaikan dari kondisi jalanan yang berbeda [1], *Inverted Index Data Structure* untuk mencari rute terpendek dalam lingkungan yang dinamis [11], dan *Ellipsoid Spatiotemporal Accessibility Method* (ESTAM) dan *Dynamic Programming Algorithm* untuk proses transit dalam sistem *ridesharing* [7].

Hungarian Algorithm dan MVSF dapat memberikan hasil yang optimal tetapi MVSF menunjukkan hasil yang menurun dan tidak stabil dengan semakin bertambah banyaknya *driver* dan *request*. Metode *Inverted Index Data Structure* memberikan hasil dengan waktu yang optimal dan sesuai untuk *real-time* dengan waktu kalkulasi yang cepat, begitu juga dengan metode ESTAM.

Hungarian Algorithm dapat memberikan hasil *assignment* yang sangat baik berdasarkan *the Great Allocation Theorem* dan *the Konig Theorem* [2] [5]. Dalam *the Great Allocation Theorem* mengatakan bahwa jika bilangan real ditambahkan atau dikurangkan dari semua entri baris atau kolom dalam matriks biaya, maka *assignment* optimal untuk hasil matriks biaya juga merupakan *assignment* optimal untuk matriks biaya asli [6].Dengan menggunakan metode *Tabu Search*, semua kemungkinan solusi dapat dicapai dan menghasilkan solusi yang terbaik dari eksekusi yang telah dijalankan. Terdapat juga *Tabu List* yang dapat mencegah pengulangan solusi [12]. Selain itu, dengan memanfaatkan struktur memori pada metode *Tabu Search* yang berbeda dan mempunyai fungsi masing-masing, sistem dapat menjalankan assignment yang simpel maupun yang memerlukan komputasi yang kompleks sesuai parameternya, dengan memilih struktur memori yang cocok.

Pada penelitian ini, Tabu Search akan digunakan untuk *driver assignment* dengan faktor jarak penjemputan dan waktu penjemputan agar mendapatkan hasil *assignment* yang dapat meminimalkan waktu dan jarak penjemputan, dan adapun faktor jumlah *trip* yang sudah dilakukan oleh *driver* dalam seminggu yang akan menentukan *assignment*.

2. DASAR TEORI

2.1 *Assignment Problem*

Assignment problem digunakan untuk menetapkan sejumlah objek ke sejumlah objek lainnya dengan cara yang terbaik [10]. Aplikasi-aplikasi dari *assignment problem* seperti menetapkan guru ke kelas-kelas, kendaraan ke rute jalan, dan masih banyak lagi. Assignment yang optimal terjadi ketika rating dari objek pekerja pada objek pekerjaan menjadi maksimum atau tertinggi. Tetapi, masalah muncul ketika jumlah kemungkinan untuk *assignment* menjadi terlalu banyak untuk dicoba satu persatu. Dari ini, diperlukan algoritma yang efisien untuk mendapatkan *assignment* yang optimal [8]. *Driver Assignment* merupakan bagian dari *assignment problem*, dimana pada *driver assignment* dilakukan pencocokan satu *driver* ke satu *request* dari pelanggan (dari beberapa *driver* dan beberapa *request*). Dengan *assignment* ini, dapat ditetapkan sesuai dengan harga terendah dan waktu tunggu yang rendah, dari jarak tempuh dan waktu tempuh.

2.2 *Tabu Search*

Tabu Search merupakan algoritma *meta-heuristic* yang cocok untuk memecahkan masalah *combinatorial optimization*. Menurut [9], Tabu Search dapat digambarkan sebagai teknik berulang yang bergerak selangkah demi selangkah dari solusi awal ke solusi yang mendekati *global optimum*. Pada *Tabu Search* semua kemungkin akan dicoba dan ketika menemukan hasil yang lebih baik dari sebelumnya, maka akan dimasukkan ke dalam *Tabu List*, dimana dalam *Tabu List* tersebut hasil di dalamnya tidak akan diubah-ubah lagi.

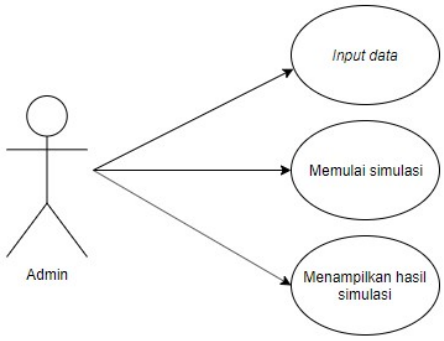
Ada tiga jenis struktur memori pada *Tabu Search*: [4]

- *Short-term* / jangka pendek: Daftar solusi yang baru saja dipertimbangkan. Jika ada solusi potensial muncul dalam *tabu list*, tidak dapat dikunjungi lagi sampai mencapai pada titik *expiration*/kedaluwarsa.
- *Intermediate-term* / jangka menengah: Aturan intensifikasi yang dimaksudkan untuk bias pencarian menuju area yang menjanjikan dari ruang pencarian..
- *Long-term* / jangka panjang: Aturan diversifikasi yang mendorong pencarian ke wilayah baru (contohnya: pengaturan ulang ketika pencarian tersangkut di plateau atau jalan buntu yang tidak optimal).

Pada *short-term memory* sudah dapat menghasilkan solusi yang lebih baik daripada metode pencarian lokal yang konvensional, tetapi dengan *intermediate* dan *long-term* lebih baik lagi digunakan untuk menyelesaikan permasalahan yang lebih sulit [10]. Contoh penggunaan algoritma *Tabu Search* untuk menyelesaikan *Traveling Salesman Problem*: [3]

3. DESAIN SISTEM

3.1 *Use Case Diagram*



Gambar 1. Use Case

Use case, pada Gambar 1, merupakan gambaran dari interaksi yang dapat dilakukan oleh aktor pada sebuah sistem. Simulasi taksi *online* ini, aktor yang berperan adalah *admin*. *Admin* dapat melakukan beberapa aktivitas seperti *input data*, memulai simulasi, dan menampilkan hasil simulasi

3.2 *Activity Diagram*

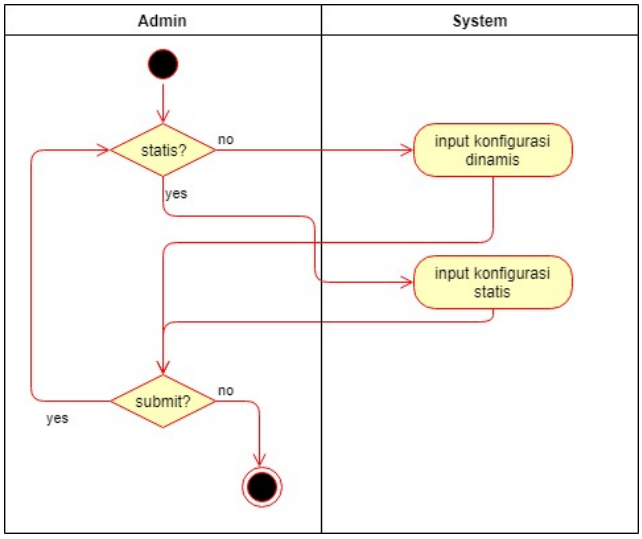
Activity diagram digunakan untuk menjelaskan alur kerja dari komponen-komponen yang ada pada sistem *assignment driver* dan *passenger* ini. *Activity diagram* yang akan dijabarkan pada sub bab ini terdiri atas tiga *activity*, yaitu *input data*, memulai simulasi, dan menampilkan hasil simulasi.

3.2.1 Activity Diagram Input Data

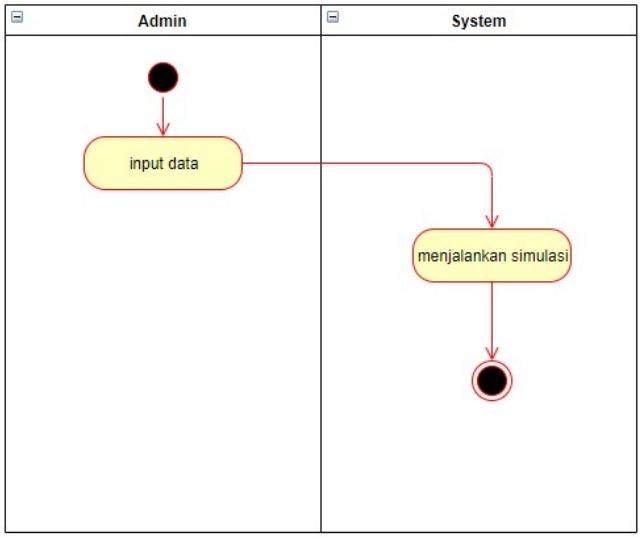
Proses input data ini dilakukan oleh admin sebelum simulasi dimulai. *Admin* memiliki dua pilihan konfigurasi yaitu statis dan dinamis. Konfigurasi statis mengharuskan *admin* untuk memasukan data konfigurasi. Konfigurasi dinamis mengharuskan admin memasukkan data seperti nama simulasi, *latitude*, dan *longitude*, jumlah maksimal kemunculan *driver* dan *passenger*. Setelah memasukkan data konfigurasi baik statis maupun dinamis, admin harus melakukan konfirmasi atas data yang dimasukan dengan menekan submit. Proses ini ditampilkan pada Gambar 2.

3.2.2 Activity Diagram Memulai Simulasi

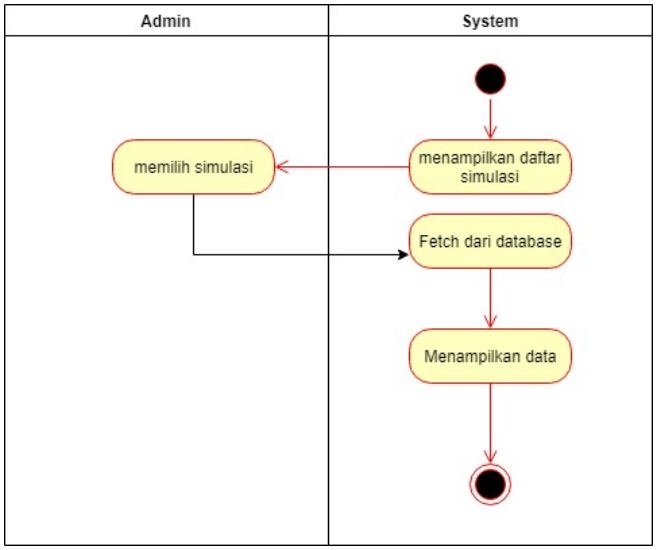
Proses memulai simulasi diawali dengan *admin* yang memasukkan data konfigurasi seperti ditampilkan pada Gambar 3. Simulasi akan dijalankan sesuai dengan konfigurasi yang telah dimasukkan oleh *admin* dengan membangkitkan data *driver* dan *passenger* pada *google maps*. Lalu, dilanjutkan dengan proses *assignment* menggunakan algoritma *tabu search*.



Gambar 2. ***Activity Diagram Input Data***



Gambar 3. ***Activity Diagram*** **Memulai Simulasi**



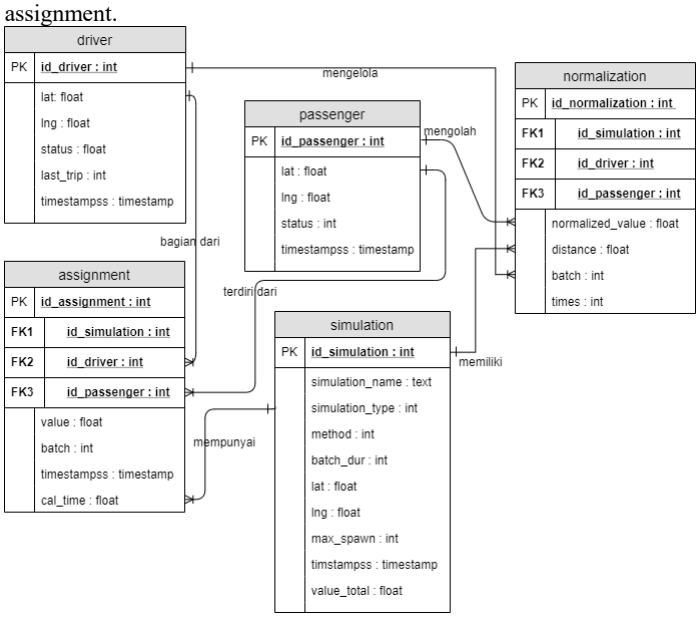
Gambar 4. ***Activity Diagram*** **Menampilkan Hasil Simulasi**

3.2.3 Activity Diagram Menampilkan Hasil Simulasi

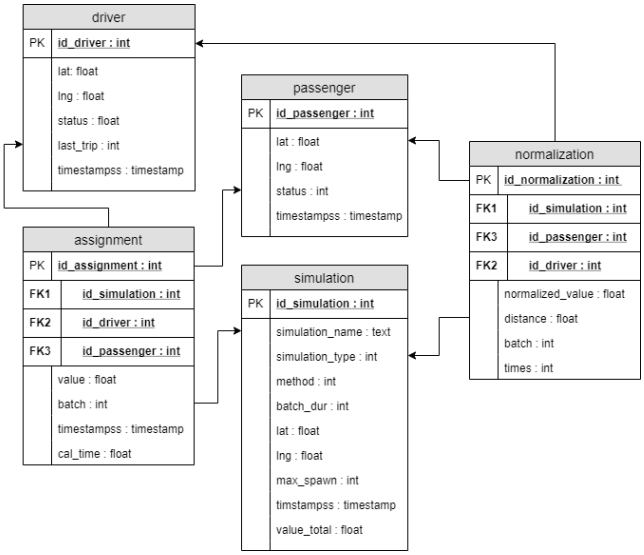
Proses menampilkan hasil simulasi dimulai dengan *admin* yang akan memilih nama simulasi, seperti pada Gambar 4. Nama simulasi yang akan dipilih *admin* diambil dari *database* simulasi yang pernah dilakukan. Lalu, *system* akan mengambil data simulasi yang dipilih dari *database* dan menampilkan datanya pada layar.

3.3 Entity Relationship Diagram (ERD)

Entity relationship diagram adalah sebuah model untuk menggambarkan hubungan antara entity dalam sebuah sistem. Model erd yang digambarkan pada Gambar 5dan Gambar 6, menunjukkan bahwa terdapat empat entity yaitu driver, passenger, simulation dan assignment.



Gambar 5. ERD ***Logical***



Gambar 6. ERD ***Physical***

4. IMPLEMENTASI SISTEM

Program ini merupakan hasil implementasi dari Bagian 3 sebelumnya. Program dibuat menggunakan bahasa pemrograman Javascript, html, dan python. Program ini menggunakan bantuan *framework* Django.

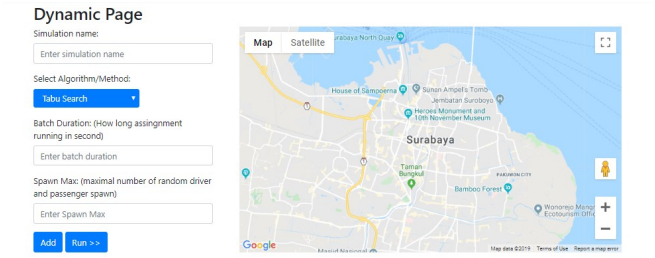
4.1 Pembuatan Fitur

Program ini terdiri dari fitur-fitur yang merupakan fungsionalitas untuk memperudah kerja admin. Fitur-fitur yang ada yaitu, input data simulasi, memulasi simulasi, dan menampilkan hasil simulasi. Fitur-fitur tersebut akan dijelaskan lebih lanjut pada subbab.

Program ini terdiri dari fitur-fitur yang merupakan fungsionalitas untuk mempermudah kerja *admin.* Fitur-fitur yang ada yaitu, *input data* simulasi, memulasi simulasi, dan menampilkan hasil simulasi.

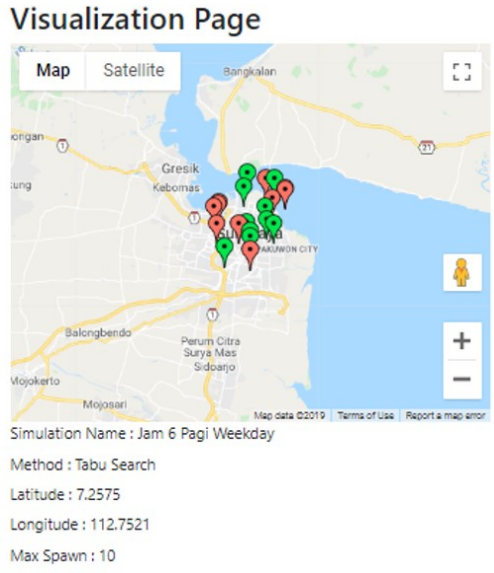
5. ANALISA DAN PENGUJIAN

Pada Gambar 7 terdapat *form* untuk memasukkan konfigurasi siumlasi yang akan dijalankan.



Gambar 7. Halaman Konfigurasi Simulasi

Hasil *assignment* ditampilkan pada halaman visualisasi, beserta dengan data *driver, passenger,* dan *normalization* yang ditunjukkan pada Gambar 8, Gambar 9, Gambar 10, dan Gambar 11.



Gambar 8. Halaman Visualisasi Simulasi (1)

Data Passenger

ID Passenger	Latitude	Longitude
P977	-7.202006832733928	112.74885519949119
P978	-7.221024206471508	112.74273573403653
P979	-7.285501006053786	112.75216633765763
P980	-7.249720373571972	112.77487896751313
P981	-7.273773896955889	112.78732680996683
P982	-7.273174148193986	112.74723525000624
P983	-7.268291756445292	112.7775220830711
P984	-7.209861828579678	112.78796309892103
P985	-7.308080810534148	112.71553630772887
P986	-7.2927568166726715	112.7527316531383

Gambar 9. Halaman Visualisasi Simulasi (2)

Data Normalization

ID Driver	ID Passenger	Distance	Time	Normalization
D11	P977	9154	1693	20.885715424471904
D11	P978	6604	1355	18.303130937579937
D11	P979	11669	1802	22.62805113531585
D11	P980	5970	903	16.34111112829634
D11	P981	10672	1554	21.202752193595593
D11	P982	9733	1471	20.400486701605566
D11	P983	9275	1307	19.566114965222965

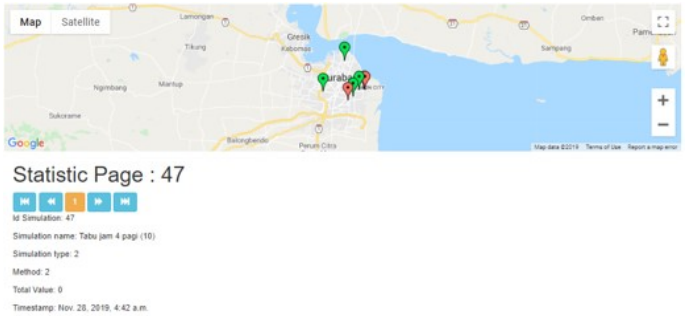
Gambar 10. Halaman Visualisasi Simulasi (3)

Assignment Result Data

ID Driver	ID Passenger	Value	Batch
D11	P984	12.0	0
D22	P980	6.5035986032373	0
D21	P979	16.332817543410886	0
D26	P978	20.669021135851235	0
D24	P986	12.783499863179122	0
D15	P983	8.464299221310744	0
D8	P982	5.026635068975569	0
D10	P977	16.130294414733825	0
D7	P981	14.81990950191252	0
D18	P985	13.868991130437886	0

Gambar 11. Halaman Visualisasi Simulasi (3)

Halaman *Statistic,* pada Gambar 12, Gambar 13, dan Gambar 14, berfungsi untuk menampilkan kembali hasil simulasi yang sudah dijalankan sebelumnya per *batch* disertai dengan tampilan grafik.

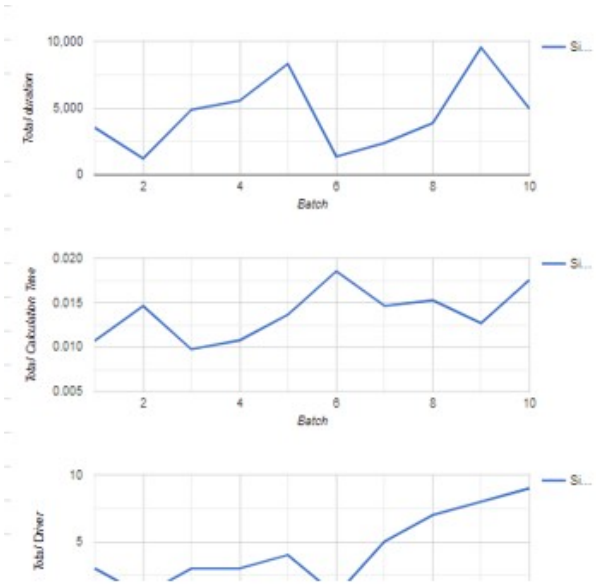


Gambar 12. Halaman *Statistic* (1)

ID Passenger	Latitude	Longitude
887	-7.300887830083193	112.77840523147195
888	-7.2865438407107085	112.79226886459965
889	-7.219090097608944	112.7561619723622
890	-7.290656470013111	112.70888762792602

ID Driver	ID Passenger	Normalized Value	Distance	Time
6	887	3.0	4564.0	845
6	888	10.499162796378613	4323.0	703
6	889	20.70233981636884	11955.0	1981
6	890	13.33465428244742	13345.0	2167
4	887	7.299882935446312	4564.0	845
4	888	6.357125014338973	4323.0	703
4	889	20.106387746305273	11955.0	1981
4	890	21.885486235543247	13345.0	2167
14	887	14.13233636150669	4564.0	845
14	888	13.18957844039935	4323.0	703
14	889	26.52848206820724	11955.0	1981
14	890	28.717939661603626	13345.0	2167

Gambar 13. Halaman *Statistic* (2)



Gambar 14. Halaman *Statistic* (3)

6. KESIMPULAN

Setelah dilakukan pengujian pada simulasi yang telah dibuat, dapat ditarik kesimpulan sebagai berikut:

- Simulasi statis dan dinamis dapat dijalankan dengan input konfigurasi simulasi dari admin.
- Aplikasi dapat menampilkan hasil penugasan dalam bentuk tabel dan grafik. Informasi grafik yang ditampilkan, yaitu rata-rata jarak, rata-rata waktu dan waktu kalkulasi.
- Dengan data request maksimal 10, algoritma *Branch and Bound* memiliki waktu kalkulasi yang paling besar dan algoritma *Hungarian* memiliki waktu simulasi yang terkecil. Waktu kalkulasi *Tabu Search* dan *Branch and Bound* tidak terlalu jauh berbeda.
- Dengan data request maksimal 10, algoritma *Branch and Bound* memiliki waktu kalkulasi yang paling besar dan algoritma *Hungarian* memiliki waktu simulasi yang terkecil. *Branch and Bound* menghasilkan *value* total yang terkecil, sedangkan algoritma *Tabu Search* menghasilkan *value* total yang terbesar
- Seiiring bertambahnya data, waktu kalkulasi algoritma *Hungarian* tetap stabil dengan kenaikan yang kecil, waktu kalkulasi *Tabu Search* mengalami kenaikan yang tidak banyak juga, dan waktu kalkulasi algoritma *Branch and Bound* mengalami kenaikan yang cukup besar.

7. DAFTAR PUSTAKA

[1] Andoko, V. P. 2019. Simulasi *Online Taxi's Dispatch System* Dengan Metode *Combinatorial Optimization*.

[2] De Oliveira, A. A. M., Souza, M. P., Pereira, M. de A., Reis, F. A. L., Almeida, P. E. M., Silva, E. J., & Crepalde, D. S. 2015. *Optimization of taxi cabs assignment in geographical location-based systems*. *Proceedings of the Brazilian Symposium on GeoInformatics*, 2015-*November*, 92–104.

[3] Fatmawati, Prihandono, B., & Noviani, E. 2015. Penyelesaian *Traveling Salesman Problem* dengan Metode *Tabu Search* - PDF. URI = https://docplayer.info/48172249-Penyelesaian-travelling-salesman-problem-dengan-metode-tabu-search.html

[4] Glover, F. 1990. *Tabu Search—Part* II. ORSA *Journal on Computing*, 2(1), 4–32. DOI = https://doi.org/10.1287/ijoc.2.1.4

[5] Konig, D. 1931. Gr´afok ´es m´atrixok. matematikai ´es fizikai lapok, 38.

[6] KUHN, H. W. 1955. *The hungarian method for the assignment problem*. *Naval Research Logistics Quarterly* 2, 83–97

[7] Masoud, N., & Jayakrishnan, R. 2017. *A real-time algorithm to solve the peer-to-peer ride-matching problem in a flexible ridesharing system. Transportation Research Part* B: *Methodological*, 106(October), 218–236. DOI = https://doi.org/10.1016/j.trb.2017.10.006

[8] Munkres, J. 1957. *Algorithms for the Assignment and Transportation Problems. Journal of the Society for Industrial and Applied Mathematics*, 5(1), 32–38. DOI = https://doi.org/10.1137/0105003

[9] Schreieck, M., Safetli, H., Siddiqui, S. A., Pflügler, C., Wiesche, M., & Krcmar, H. 2016. *A Matching Algorithm for Dynamic Ridesharing. Transportation Research Procedia*, 19(*June*),272–285. DOI= https://doi.org/10.1016/j.trpro.2016.12.087

[10] Shen, Y., & Kwan, R. S. K. 2001. *Computer-Aided Scheduling of Public Transport*, 505(*June*), 9–11. DOI = https://doi.org/10.1007/978-3-642-56423-9

[11] Shen, Y., & Kwan, R. S. K. 2011. *Tabu Search for Driver Scheduling*, (1995), 121–135. DOI = https://doi.org/10.1007/978-3-642-56423-9_7

[12] SINGH, S. 2013. *A Comparative Analysis of Assignment Problem. IOSR Journal of Engineering*, 02(08), 01–15. DOI =https://doi.org/10.9790/3021-0281011