IMPLEMENTASI DAN ANALISIS PENGGUNAAN ALGORITMA A-STAR DENGAN PRIORITAS PADA PEMILIHAN RUTE LINTAS KENDARAAN RODA DUA

IMPLEMENTATION AND ANALYSIS THE USE A-STAR ALGORITHM WITH PRIORITY ON TWO-WHEELS VEHICLE ROUTING

Muhammad Hisyam Fadhlurrahman[1] **, Burhanuddin Dirgantara, Ir., MT.**[2] **, Asep Mulyana, ST., MT.**[3]

[1,2,3]Prodi S1 Sistem Komputer, Fakultas Teknik, Universitas Telkom

[1]fadhlurrahman.hsm@gmail.com, [2]burhanuddin@telkomuniversity.ac.id, [3]asepm267@gmail.com

Abstrak

Algoritma A* (*A-Star*) merupakan salah satu algoritma pencarian rute yang komplit dan optimal[4]. Algoritma *A-Star* menggabungkan nilai g(n) dan nilai h(n) untuk mencari bobot terendah dalam menentukan solusi. Di Indonesia, terutama di kota kota besar, terdapat beberapa jenis jalan yang dapat dilalui kendaraan. Selain itu, dalam kondisi macet pengendara sepeda motor cenderung untuk mencari jalan alternatif agar terhindar macet namun dengan kondisi yang optimal. Pada penelitian ini, dibuat sistem yang dapat menentukan rute alternatif sepeda motor. Sistem mendefinisikan prioritas jalan yang ada. Penelitian ini menggunakan Algoritma *A-Star* sebagai metode dalam mencari rute optimal. Prioritas diaplikasikan kedalam penghitungan nilai g(x). Dari hasil penelitian sistem dapat menghindari kemacetan dengan rute yang efektif.

Kata Kunci: algoritma pencarian rute, nilai heuristik, algoritma a-star

Abstract

A* (A-Star) Algorithm is one of complete and optimum routing algorithm. A-Star Algorithm combines g(n) and h(n) to calculate the cheapest solution. In Indonesia, especially in big city, there are several type of ways that can be passed through vehicle. Otherwise, in the traffic jam, bikers tend to find the other way to avoid the traffic jam that is still optimum. This research makes a system that can route alternative way for motorcycles. This system define the priority of each type of ways. This research uses A-Star Algorithm to find the optimum route. The priority is applied in calculating g(x). the testing results show that this system can avoid traffic jam with effective route.

Key Word: *routing algorithm, heuristic cost, a-star algorithm*

1. Pendahuluan

Pencarian rute erat kaitannya dengan dengan kehidupan sehari-hari. Ketika seseorang ingin berkendara, orang tersebut akan mencari rute terbaik untuk sampai ketujuan. Rute terbaik dapat berupa waktu tercepat, kondisi jalan yang baik, maupun kondisi jalan yang bebas dari kemacetan.

Algoritma A-Star merupakan salah satu algoritma pencarian rute yang optimal dan komplit. Optimal berarti rute yang dihasilkan adalah rute yang paling baik dan komplit berarti algoritma tersebut dapat mencapai tujuan yang harapkan. Dalam penerapannya, Algoritma A-Star menggunakan jarak sebagai proses kalkulasi nilai terbaik. Untuk pencarian rute yang dapat menghindari kemacetan, perlu ada informasi lain sehingga sistem dapat menemukan rute terbaik dengan mempertimbangkan kondisi kemacetan di jalan.

Dari permasalahan diatas, pada penelitian kali ini, dirancang sebuah sistem pencarian rute kendaraan untuk sepeda motor dengan menggunakan Algoritma A-Star. Implementasi menggunakan kendaraan roda dua karena dalam pemilihan rute kendaraan roda dua memiliki banyak pilihan jalur yang dapat dilalui sehingga dapat diperoleh pengaruh penggunaan prioritas pada Algoritma A-Star.

2. Dasar Teori

2.1 Permasalahan Lintasan Terpendek [4]

Permasalahan lintasan tependek dapat digambarkan sebagai upaya pencarian lintasan yang memiliki biaya *minimum cost*. Biaya lintasan adalah jumlah biaya semua *arc* yang membentuk lintasan tersebut. Ada beberapa asumsi yang digunakan dalam perhitungan lintasan terpendek, yaitu:

1. Jaringan berarah (*directed network*).
2. Ada lintasan berarah dari satu *node* sumber ke semua *node* lain.
3. Tidak ada siklus negatif, yaitu siklus dengan total biaya negatif.
4. Biaya tiap *arc* merupakan bilangan bulat.

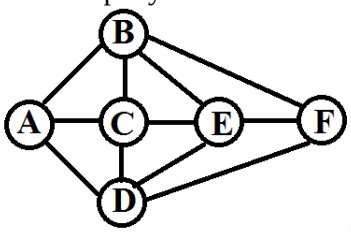
Formulasi masalah dalam permasalahan lintasan terpendek dapat dinyatakan dalam bentuk aliran minimum dimana tiap *node* yang dituju dianggap memiliki permintaan sebesar satu unit dan *node* sumber memiliki *supply* sebanyak yang diminta. Lintasan terpendek dari *nose* S (*node* sumber) ke *node* T (*node* tujuan) diformulasikan sebagai berikut:

2.2 *Graf* [5]

Graf adalah kumpulan simpul (*nodes*) yang terhubung satu sama lain melalui sisi (*edges*). Suatu graf G terdiri dari dua himpunan V dan E. *Verteks* (simpul): V = himpunan simpul yang terbatas dan tidak kosong. *Edge* (sisi): E = himpunan sisi yang menghubungkan sepasang simpul. Simpul – simpul pada graf dapat merupakan obyek sembarang seperti: kota, atom – atom suatu zat, jenis buah, dan sebagainya. Sisi dapat menunjukkan relasi sembarang seperti rute penerbangan, jalan raya, dan lain- lain. Notasi graf G (V, E) menyatakan bahwa graf G memiliki V simpul dan E sisi.

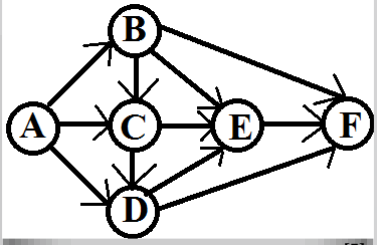
Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis:

1. Graf tak-berarah (*undirected graph)*
   Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah.



Gambar 2.1 Graf Tak Berarah [5]

2. Graf berarah (*directed graph* atau *digraph*)
   Graf yang setiap sisinya diberikan orientasi arah disebut graf berarah.



Gambar 2.2 Graf Berarah [5]

2.3 Algoritma A* (A Star) [4]

Algoritma A* merupakan algoritma *pathfinding* pengembangan dari Algoritma BFS (*Best First Search*). Seperti halnya pada BFS, untuk menemukan solusi, A* juga 'dituntun' oleh fungsi heuristik. Perbedaan cara kerja A* dengan BFS adalah selain memperhitungkan *cost* dari *currentstate* ke tujuan dengan fungsi *heuristic* (seperti BFS), algoritma ini juga mempertimbangkan *cost* yang telah ditempuh selama ini dari *initial state* menuju ke *current state*. Jadi, apabila jalan yang ditempuh sudah terlalu panjang dan ada jalan lain yang lebih kecil *cost*-nya namun memberikan solusi yang sama apabila dilihat dari *goal*, maka jalan baru yang lebih pendek itulah yang akan dipilih. Notasi yang dipakai oleh Algoritma A* adalah sebagai berikut:

$$f(n) = g(n) + h(n) \qquad (1)$$

f(n) = biaya estimasi terendah
g(n) = biaya dari node awal ke node n
h(n) = perkiraan biaya dari node n ke node akhir

Dalam penerapannya, Algoritma A* memiliki beberapa terminologi dasar diantaranya starting point, simpul *(nodes*), A, *open list*, *closed list*, harga (*cost*), halangan (*unwalkable)*. [2]

1. *Starting point* adalah sebuah terminologi untuk posisi awal sebuah benda.
2. A adalah simpul yang sedang dijalankan dalam algortima pencarian jalan terpendek.
3. Simpul adalah petak-petak kecil sebagai representasi dari area *pathfinding*. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga.
4. *Open list* adalah tempat menyimpan data simpul yang mungkin diakses dari starting point maupun simpul yang sedang dijalankan.
5. *Closed list* adalah tempat menyimpan data simpul sebelum A yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan.
6. Harga adalah nilai yang diperoleh dari penjumlahan, jumlah nilai tiap simpul dalam jalur terpendek dari *starting point* ke A, dan jumlah nilai perkiraan dari sebuah simpul ke simpul tujuan.
7. Simpul tujuan yaitu simpul yang dituju.
8. Halangan adalah sebuah atribut yang menyatakan bahwa sebuah simpul tidak dapat dilalui oleh A.

Berikut ini adalah *pseudocode* Algoritma A*. [5]

```
function A*(start,goal)
closedset := the empty set    // The set of nodes already evaluated.
openset := {start}    // The set of tentative nodes to be evaluated, initially
containing the start node
came_from := the empty map    // The map of navigated nodes.

g_score[start] := 0    // Cost from start along best known path.
// Estimated total cost from start to goal through y.
f_score[start] := g_score[start] + heuristic_cost_estimate(start, goal)

while openset is not empty
        current := the node in openset having the lowest f_score[] value
        if current = goal
        return reconstruct_path(came_from, goal)

                remove current from openset
                add current to closedset
        for each neighbor in neighbor_nodes(current)
                if neighbor in closedset
                        continue
                tentative_g_score := g_score[current] +
                dist_between(current,neighbor)

                if neighbor not in openset or tentative_g_score <
                g_score[neighbor]
                came_from[neighbor] := current
                g_score[neighbor] := tentative_g_score
                f_score[neighbor] := g_score[neighbor] +
                heuristic_cost_estimate(neighbor, goal)
                if neighbor not in openset
                add neighbor to openset
                        return failure
function reconstruct_path(came_from,current)
total_path := [current]
while current in came_from:
```

current := came_from[current]
total_path.append(current)
return total_path

2.4 Fungsi Heuristik [6]

Algoritma A* sebagai algoritma pencarian yang menggunakan fungsi heuristik untuk menuntun' pencarian rute, khususnya dalam hal pengembangan dan pemeriksaan *node-node* pada peta. Terdapat beberapa fungsi heuristic umum yang bisa dipakai untuk algoritma A* ini. Salah satunya adalah yang dikenal dengan istilah '*Manhattan Distance*'. Fungsi heuristic ini digunakan untuk kasus dimana pergerakan pada peta hanya lurus (horizontal atau vertikal), tidak diperbolehkan pergerakan diagonal. Perhitungan nilai heuristikuntuk *node* ke-n menggunakan *Manhattan Distance* adalah sebagai berikut:

$$h(n) = (abs(n(x) - goal(x)) + abs(n(y) - goal(y))) \quad (2)$$

Untuk mendekati kenyataan, *cost* untuk perpindahan *node* secara diagonal dan orthogonal dibedakan. *Cost* diagonal adalah 1.4 kali *cost* perpindahan secara orthogonal, maka fungsi heuristic yang digunakan adalah sebagai berikut:

$$h\_diagonal(n) = min(abs(n(x) - goal(x)) + abs(n(y) - goal(y))) \quad (3)$$

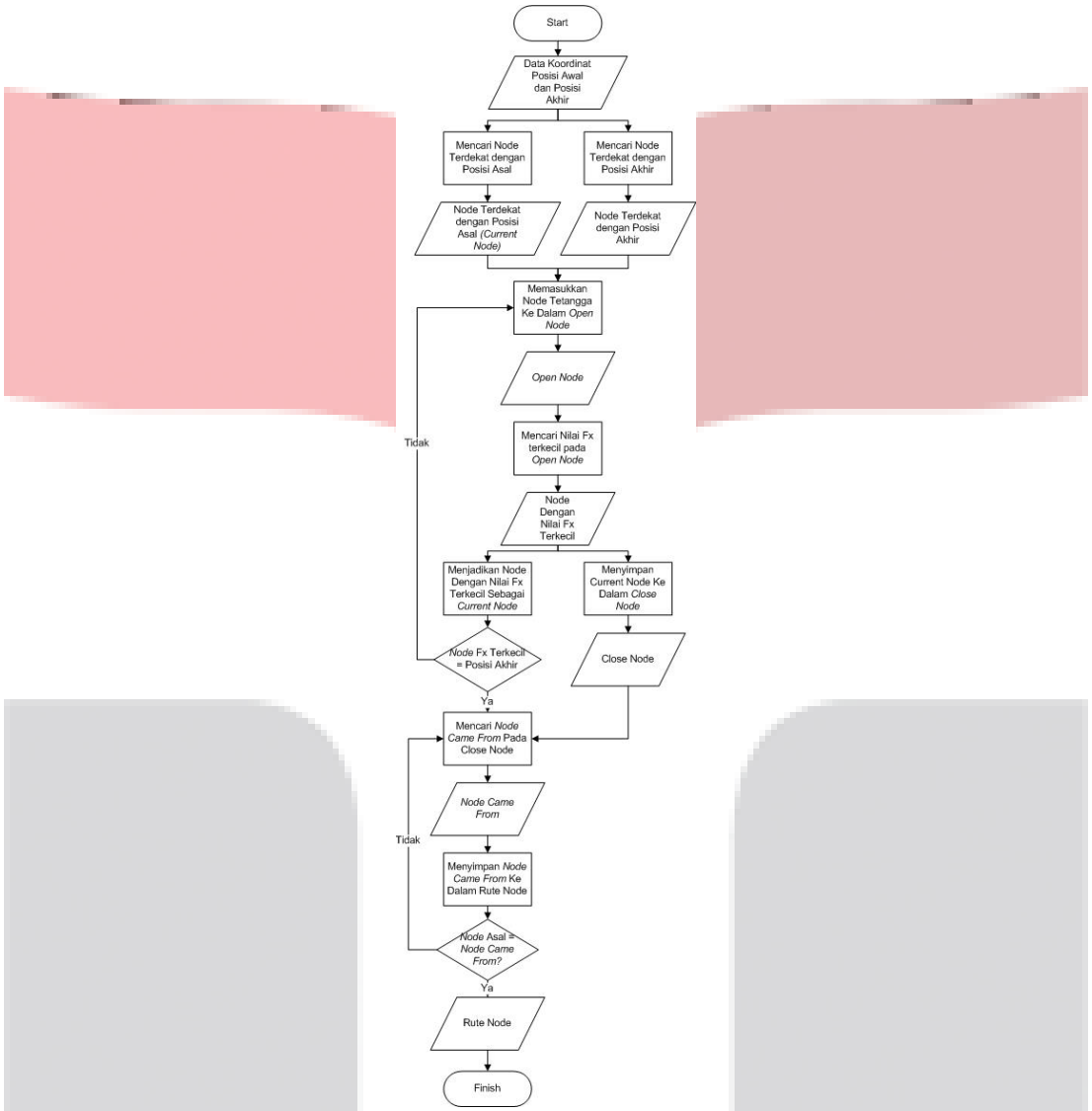
$$h\_orthogonal(n) = (abs(n(x) - goal(x)) + abs(n(y) - goal(y))) \quad (4)$$

$$h(n) = h\_diagonal(n) + (h\_orthogonal\ (n) - (2 * h\_diagonal(n))) \quad (5)$$

dimana h_diagonal(n) adalah banyaknya langkah diagonal yang bisa diambil untuk mencapai *goal* dari *node* n. h_orthogonal adalah banyaknya langkah lurus yang bisa diambil untuk mencapai *goal* dari *node* n. Nilai *heuristic* kemudian diperoleh dari h_diagonal(n) ditambah dengan selisih h_orthogonal(n) dengan dua kali h_diagonal(n). Dengan kata lain, jumlah langkah diagonal kali *cost* diagonal ditambah jumlah langkah lurus yang masih bisa diambil dikali *cost* pergerakan lurus.

3. Perancagan Sistem dan Implementasi

Berikut diagram alir Algoritma A-Star



Gambar 3.1 Algoritma A*

Setelah memperoleh data *latitude* dan *longitude* asal dan tujuan dari *smartphone*, sistem akan mencari lokasi *node* terdekat dengan titik asal (*current node)* dan *node* terdekat dengan titik tujuan *(dest node)*. Titik asal dan tujuan disimpan untuk diolah kembali kemudian.

Sistem akan mencari *node* yang terhubung dengan *current node* dari *database*. Sistem akan memilih satu node dengan nilai f(x) yang paling kecil. Node tersebut kemudian akan menjadi current node yang baru sementara current node sebelumnya akan disimpan. Proses ini akan dilakukan sampai current node sama dengan dest node. Node-node yang telah disimpan kemudian dikirim kembali ke smartphone.

Sistem ini menggunakan prioritas pada *vertex* graf dengan kondisi sebagai berikut.

Tabel 3.1 Tabel Prioritas

No.	Prioritas	Keterangan
1	5	Jalur yang hanya dapat dilalui pejalan kaki.
2	4	Ruas jalan utama yang mengalami kemacetan.
3	3	Jalan alternatif yang hanya dapat dilalui sepeda

		motor, sepeda, dan pejalan kaki.
4	2	Jalan alternatif yang dapat dilalui mobil dan sepeda motor.
5	1	Ruas jalan utama

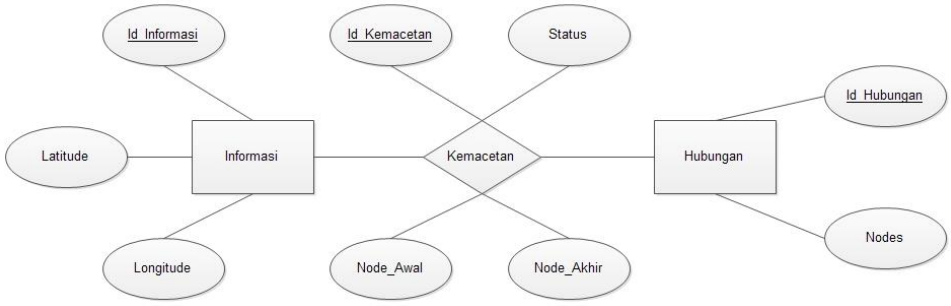
Penghitungan bobot dilakukan pada saat penghitungan nilai g(x) sehingga rumusnya adalah sebagai berikut.

$$g_{total}(x) = g_{antara\_dua\_node}(x) * prioritas \quad (6)$$

$$f(x) = h(x) + g_{terlewati}(x) \quad (7)$$

3.1 Perancangan *Database*

Dalam implementasinya, graf direpresentasikan ke dalam matriks. Matriks tersebut disimpan kedalam basis data. Berikut ini adalah gambar ERD dari database tersebut.



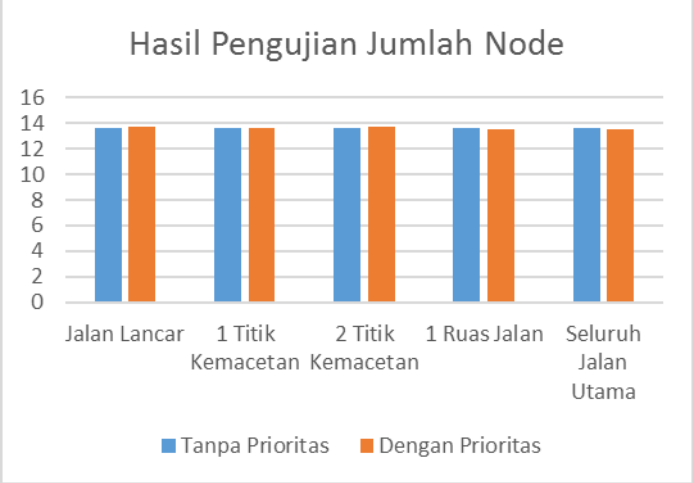
Gambar 3.2 Entity Relationship Diagram

4. Pengujian Sistem dan Analisis

Proses pengujian dilakukan dengan menggunakan Algoritma A-Star. Proses ini memperoleh titik acuan dari posisi *user* dan posisi akhir dari lokasi yang ditunjuk oleh *user*. Pengujian ini dilakukan menggunakan 5 lokasi sebagai titik acuan (posisi *user*). Dari masing-masing titik, penguji menentukan 10 lokasi tujuan berbeda yang kemudian rute kendaraan dipetakan pada aplikasi.

Pengujian dilakukan dengan menghitung total *node* yang ditempuh dari rute yang dihasilkan, waktu komputasi, serta memori yang digunakan berdasarkan jumlah *open node* yang disimpan. Pengujian ini membandingkan penggunaan Algoritma A-Star dengan Algoritma A-Star yang telah ditambahkan prioritas pada *vertex* graf. Tujuan pengujian ini adalah untuk menentukan apakah penerapan prioritas *vertex* pada graf yang terdapat pada sistem efektif.

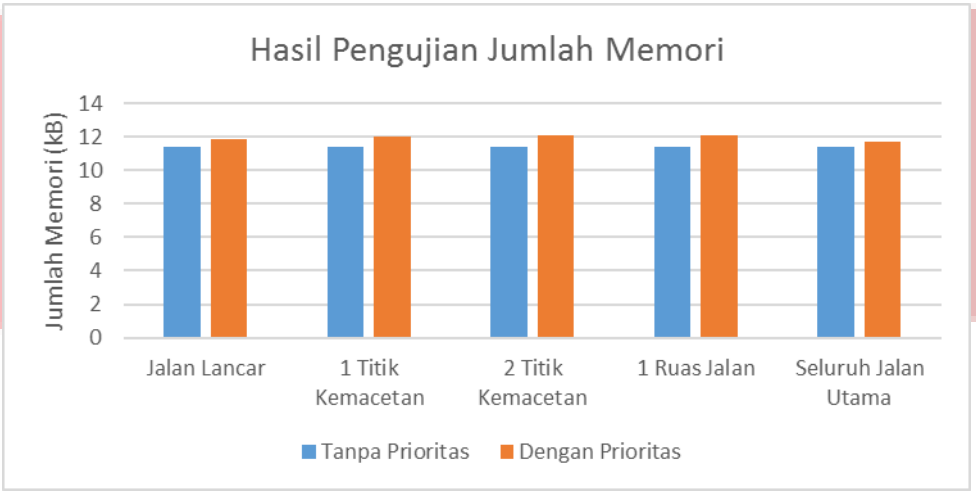
4.1 Hasil Pengujian Jumlah Node



Gambar 4.2 Grafik Hasil Pengujian Jumlah Node

Dari hasil pengujian diperoleh penggunaan jumlah node pada Algoritma A-Star menggunakan prioritas lebih sedikit dibandingkan tanpa prioritas. Hal ini menunjukkan bahwa penggunaan prioritas lebih efektif. Terlhat bahwa penggunaan prioritas dapat digunakan untuk proses pencarian rute kendaraan berdasarkan data kondisi jalan.

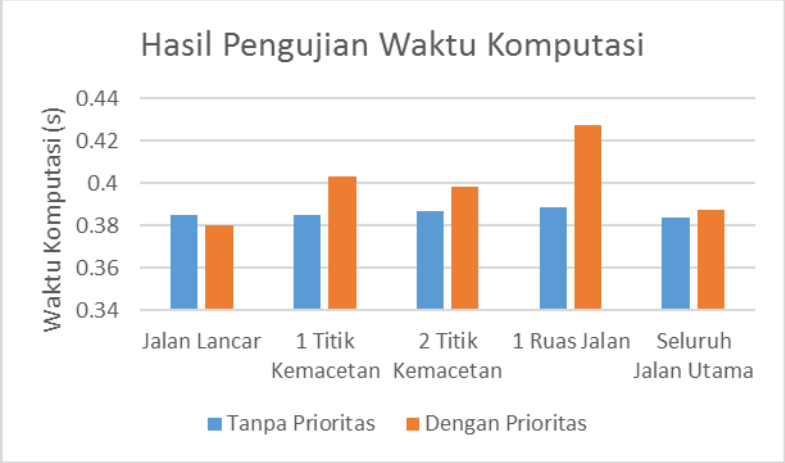
4.2 Hasil Pengujian Jumlah Memori



Gambar 4.3 Grafik Hasil Pengujian Jumlah Memori

Dari hasil pengujian diperoleh penggunaan memori pada Algoritma A-Star menggunakan prioritas lebih besar. Hal ini disebabkan proses kalkulasi nilai terbaik membutuhkan lebih banyak data. Dari hasil pengujian diperoleh penggunaan memori menggunakan prioritas lebih besar 4.27%.

4.3 Hasil Pengujian Waktu Komputasi



Gambar 4.4 Grafik Hasil Pengujian Waktu Komputasi

Dari hasil pengujian waktu komputasi, semakin banyak jumlah kemacetan yang terjadi, waktu komputasi yang dilakukan semakin lama. Sedangkan pada perbandingan tanpa dan dengan prioritas, penggunaan prioritas melakukan komputasi rata-rata 0.03515 detik lebih lama. Kedua hal ini disebabkan karena terdapat penambahan data pada saat proses komputasi.

5. Kesimpulan

Berdasarkan hasil pengujian pencarian rute dapat disimpulkan bahwa penggunaan prioritas pada Algoritma A-Star dapat digunakan untuk menentukan rute berdasarkan data kemacetan. Sistem menunjukkan rute alternatif untuk sepeda motor dengan tingkat akurasi sebesar 78 %. Hal ini disebabkan karena adanya prioritas pada setiap ruas jalan.

Dari pengujian jumlah memori yang digunakan, penggunaan prioritas menggunakan rata-rata 4.27 % lebih banyak dibandingkan dengan tanpa prioritas. Sedangkan dalam waktu komputasi, penggunaan prioritas melakukan komputasi rata-rata 0.03515 detik lebih lama. Dari dua kondisi tersebut, penggunaan prioritas pada Algoritma A-Star tergolong efektif.

Daftar Pustaka:

[1] Hardjasutanto, Fabrian Oktavino, 2010. **PENERAPAN ALGORITMA SEMUT UNTUK PENCARIAN JALUR TERPENDEK**, Bandung: Institut Teknologi Bandung.

[2] Melhorn, Kurt. Peter Sanders., 2007. **ALGORITHM AND DATA STRUCTURES THE BASIC TOOLBOX,** Karlsruhe: Springer.

[3] Putra, Rangga Dionata. Muhammad Aswin., Waru Djuriatno., 2012. **PENCARIAN RUTE TERDEKAT PADA LABIRIN MENGGUNAKAN METODE A*,** Jurnal, EECCIS.

[4] Russel, Stuart. Peter Norwig., 2003. **ARTIFICIAL INTELLIGENCE A MODERN APPROACH THIRD EDITION,** New Jersey: Pearson Education, Inc.

[5] Setyawan, Marhaendro Bayu. Nurlita Gamayanti. Abdullah Alkaff., **OPTIMASI RUTE PERJALANAN AMBULANCE MENGGUNAKAN ALGORITMA A-STAR,** Surabaya: Jurnal. ITS.

[6] Tanimoto, Steven L, 1987. **THE ELEMENTS OF ARTIFICIAL INTELLIGENCE AN INTRODUCTION USING LISP,** Washington: Computer Science Press, Inc.

[7] Witanti, Wina. Dewi Nurul Rahayu, 2013. **ANALISIS PENGARUH PENGGUNAAN NILAI HEURISTIK TERHADAP PERFORMANSI ALGORITMA A* PADA GAME PATHFINDING,** Cimahi: Jurnal, Seminar Nasional Informasi Indonesia.