BAB II

LANDASAN TEORI

2.1 Penelitian yang Relevan

Sebagai referensi penelitian tentang analisis perbandingan perhitungan jalan terdekat pada algoritma *Dijkstra* dan algoritma *A** untuk memetakan wilayah tower BTS Telkomsel di Kota Semarang Jawa Tengah. Pada penelitian yang akan dibuat, penulis mengambil beberapa penelitian jurnal yang sudah ada sebelumnya, antara lain:

1. Penelitian yang dilakukan oleh Hu (2018), dengan judul "*Model and Algorithm for the Large Material Distribution Problem in Maritime Transportation*". Penelitian tersebut menyajikan model dan algoritma transportasi laut yang inovatif untuk mengatasi masalah distribusi material yang besar dalam situasi darurat pada perutean kapal. Teknik *fuzzy clustering* diterapkan untuk menentukan nilai parameter penurunan pada fungsi penurunan berdasarkan *geo-information*. Sebuah algoritma genetika hibrida dirancang untuk memecahkan model, di mana algoritma *Dijkstra* diintegrasikan ke dalam proses konstruksi solusi. Hasil dari penelitian ini menunjukkan penerapan model distribusi *relief-demand* dan efektivitas algoritma yang diusulkan.
2. Penelitian yang dilakukan oleh Chen (2020), dengan judul "*Application of Improved Dijkstra Algorithm in Coastal Tourism Route Planning*". Artikel tersebut menerapan algoritma *Dijkstra* yang disempurnakan dalam perencanaan rute wisata pantai guna mengirim barang ke tempat tujuan dengan cara paling efisien. Penelitian menggunakan GIS (*Geographic Information System*) dalam implementasinya. Penelitian ini menampilkan hasil rute yang paling optimal untuk digunakan dalam pengiriman barang ke tempat tujuan.

3. Penelitian yang dilakukan oleh (Bagheri, dkk, 2021), dengan judul "*An A-Star Algorithm for Semi-Optimization of Crane Location and Configuration in Modular Construction*". Penelitian tersebut menyajikan kerangka kerja baru berdasarkan algoritme penelusuran jalur terpendek terinformasi disebut dengan *A**, menyediakan rencana pengangkatan semi-optimal berdasarkan urutan pengangkatan yang telah ditentukan. Hasil dari penelitian ini menunjukkan pengurangan nilai jarak total yang signifikan dibandingkan dengan algoritma perencanaan pengangkatan yang digunakan sebelumnya.
4. Penelitian yang dilakukan oleh Mirahadi dan McCabe (2021), yang berjudul "*EvacuSafe : A Real-Time Model for Building Evacuation Based on Dijkstra's Algorithm*". Penelitian tersebut membahas mengenai model manajemen evakuasi *real time* yang mengidentifikasi bahaya kebakaran yang dapat mengurangi resiko kegagalan evakuasi dengan jalur resiko yang dihiitung menggunakan algoritma *Dijkstra*. Hasil dari penelitian ini berupa model yang diusulkan untuk manajemen evakuasi secara *real time* dapat secara signifikan menaikkan tingkat keselamatan evakuasi dibandingkan dengan rencana evakuasi konvensional.
5. Penelitian yang dilakukan oleh Sunita dan Garg (2021), yang berjudul "*Dynamizing Dijkstra: A solution to dynamic shortest path problem through retroactive priority queue*". Penelitian tersebut dilakukan dengan mendinamisasi algoritma *Dijkstra* yang membantu menyelesaikan masalah jalur terpendek *single source* secara efisien. Penelitian dilakukan analisis dengan membandingkan algoritma yang diusulkan dengan algoritma lain. Hasil dari penelitian ini menunjukan bahwa algoritma yang diusulkan memiliki kinerja yang lebih optimal dalam hal penggunaan waktu dan memori.

2.2 GIS (Geographic Information System)

Sistem Informasi Geografis (SIG) merupakan sebuah teknologi berbasis komputer yang digunakan untuk mengumpulkan, mengelola, menganalisis,

dan menampilkan data yang memiliki acuan spasial atau geografis. Teknologi ini berperan penting dalam membantu proses pengambilan keputusan yang berkaitan dengan lokasi atau wilayah tertentu Singh dan Katiyar (2021). Proses pemberian informasi dalam GIS dimulai dari menangkap, menyimpan, memanipulasi, menganalisis, dan mengelola informasi geografi berdasarkan lokasi (Zhang, dkk, 2022).

GIS terdiri dari sistem berupa peta digital yang mempresentasikan objek dalam database. Sehingga dapat melakukan analisis static berbasis informasi spasial dalam peta dan menjadikannya lebih mudah dimengerti daripada informasi berupa teks (Ma, 2020). Karena kemudahan yang dimiliki, GIS banyak dimanfaatkan oleh kalangan industri dan pemerintahan (Scarlett, dkk, 2019). Selain itu, GIS dapat diimplementasikan pada berbagai perangkat mobile dan dekstop dengan sistem operasi yang berbeda (Isihak, dkk, 2022).

GIS dapat digambarkan sebagai kumpulan basis data yang memungkinkan kumpulan data besar dari informasi spasial dan tabular untuk kemudian disimpan dan diolah menjadi sebuah informasi geografi yang ada. GIS menawarkan berbagai manfaat berupa fitur dan teknologi seperti *query*, tampilan *database*, menganalisis spasial melalui GIS yang dapat meningkatkan pendekatan konvensional untuk pemrosesan yang lebih efisien Singh dan Katiyar (2021). Penggunaan GIS saat ini merupakan sebuah inovasi yang sedang berkembang dalam bidang teknologi informasi komunikasi yang ada di Indonesia. GIS memiliki banyak kegunaan berdasarkan kepentingannya masing-masing seperti database spasial, mobile GIS, dan web GIS (Scarlett, dkk, 2019). GIS juga memiliki kemampuan berkomunikasi yang mudah dimengerti, *powerful* karena dapat memberikan semua informasi yang sesuai kondisi geografi, dan dapat mendistribusikan informasi sesuai lokasi (Zhang, dkk, 2022).

2.3 Algoritma *Dijkstra*

Algoritma *Dijkstra* merupakan salah satu metode yang digunakan untuk mencari lintasan terpendek pada sebuah *graph* berbobot, di mana seluruh bobot

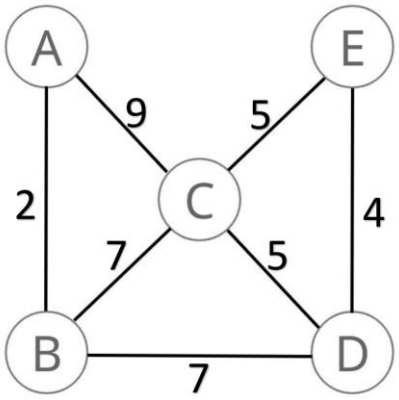
sisi bernilai positif. Algoritma ini bekerja dengan menghitung jarak minimum dari titik asal menuju *node-node* lainnya secara bertahap. Sehingga apabila *graph* tersebut memiliki bobot yang bernilai negatif maka lintasan tersebut tidak dapat dilintasi dan penyelesaiannya berupa *infiniti* (Chen, 2020). Algoritma ini memilih titik terdekat yaitu titik dengan bobot yang terkecil, bobot tersebut merupakan bobot yang dimiliki tiap edge (Ernest, dkk, 2022). Algoritma ini ditemukan oleh Edsger Wybe *Dijkstra* pada tahun 1959 dan merupakan salah satu dari banyaknya algoritma *shortest path* yang popular digunakan dalam dalam pengukuran atau pemecahan masalah dalam hal optimalisasi Mirahadi dan McCabe(2021). Dalam proses pencarian jalur terpendek, algoritma *Dijkstra* bekerja dengan mengevaluasi bobot terkecil dari *graph* berbobot yang dimulai dari *node* awal. Jalur terpendek ditentukan berdasarkan akumulasi nilai bobot terkecil yang menghubungkan dua atau lebih *node*, dan hasil akhir yang diperoleh adalah total jarak minimum dari seluruh jalur yang mungkin (Barzegar, dkk, 2019).

Menurut Mirahadi dan McCabe(2021) tahapan dalam penerapan algoritma Dijkstra dapat dijelaskan sebagai berikut:

1. Menetapkan nilai bobot untuk setiap simpul terhadap simpul lainnya. *Node* awal diberikan nilai nol, sedangkan node lainnya diinisialisasi dengan nilai tak hingga (∞) sebagai tanda bahwa belum ada jalur yang ditemukan.
2. Seluruh *node* diberi label "belum dikunjungi" (*unvisited*), kemudian *node* awal ditetapkan sebagai titik awal (*start node*).
3. Hitung jarak dari *node* awal menuju seluruh *node* yang terhubung. Apabila ditemukan jarak baru yang lebih kecil dibandingkan jarak sebelumnya, maka jarak sebelumnya akan digantikan dengan nilai yang lebih kecil tersebut.
4. Setelah seluruh *node* tetangga dihitung jaraknya, *node* yang telah dianalisis ditandai sebagai "sudah dikunjungi" (*visited*). Node yang telah dikunjungi tidak akan dianalisis kembali karena nilai jaraknya telah dianggap sebagai jarak minimum yang pasti..

5. Pilih simpul "belum dikunjungi" dengan nilai jarak terkecil sebagai simpul awal yang baru, kemudian ulangi proses dari langkah ketiga.

Untuk menentukan jalur terpendek dari *graph* yang ditunjukkan pada Gambar 2.1 dengan titik asal = A, dan titik tujuan = E adalah sebagai berikut:



Gambar 2. 1 Soal *Graph* dengan Algoritma *Djikstra*

Langkah 1:

- Buatlah tabel untuk hasil lintasan dengan keterangan i (*iteration*), u(*unvisited*), c (*current node*), dan titik A, B, C, D, E seperti pada Tabel 2.1.
- i = 0 adalah keadaan awal.
- Pada i = 1 dapat dilihat pada Tabel 2.1, AB = 2, AC = 9. Lintasan AB adalah lintasan dengan bobot terendah dengan bobot = 2.
- Maka, tentukan B sebagai node "start" untuk i = 2.

Tabel 2.1 Iterasi 0 dan 1

iteration	*unvisited*	*current node*	A	B	C	D	E
0	**ABCDE**	-	0,-	∞,-	∞,-	∞,-	∞,-
1	**BCDE**	**A**	0,-	2,A	9,A	∞,-	∞,-

Langkah 2:

- Pada i = 2 dapat dilihat pada Tabel 2.2, BC = 2+7 = 9, BD = 2+7 = 9.
- Pada *node* C sekarang [9,B] sama dengan *node* C sebelumnya [9,A] maka nilai *node* C bisa menjadi [9,B] atau [9,A].

- Tentukan C sebagai *node* "*start*" sesuai alfabet order untuk i = 3.

Tabel 2.2 Iterasi 2

iteration	*unvisited*	*current node*	A	B	C	D	E
0	ABCDE	-	0,-	∞,-	∞,-	∞,-	∞,-
1	BCDE	A	0,-	2,A	9,A	∞,-	∞,-
2	CDE	B	0,-	2,A	9,B	9,B	∞,-

Langkah 3:

- Pada i = 3 dapat dilihat pada Tabel 2.3, CD = 9+5 = 14, CE = 9+5 = 14.
- Pada *node* D sekarang [14,C] lebih besar dari *node* sebelumnya [9,B] maka nilainya diganti menjadi [9,B] dari [14,C].
- Tentukan D sebagai *node* "start" untuk i = 4.

Tabel 2.3 Iterasi 3

iteration	*unvisited*	*current node*	A	B	C	D	E
0	ABCDE	-	0,-	∞,-	∞,-	∞,-	∞,-
1	BCDE	A	0,-	2,A	9,A	∞,-	∞,-
2	CDE	B	0,-	2,A	9,B	9,B	∞,-
3	DE	C	0,-	2,A	9,B	9,B	14,C

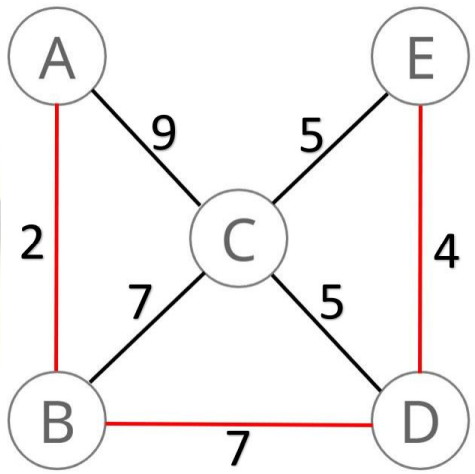
Langkah 4:

- Pada i = 4 dapat dilihat pada Tabel 2.4, DE = 9+4 = 13.
- Pada *node* E sekarang [14,C] lebih besar dari *node* sebelumnya [13,D] maka nilainya tidak berubah atau tetap [13,D].
- Tentukan E tidak bisa sebagai *node* "start" karena *node* E sebagai *node* tujuan.
- Selesai.

Tabel 2.4 Iterasi 4

iteration	*unvisited*	*current node*	A	B	C	D	E
0	ABCDE	-	0,-	∞,-	∞,-	∞,-	∞,-
1	BCDE	A	0,-	2,A	9,A	∞,-	∞,-
2	CDE	B	0,-	2,A	9,B	9,B	∞,-
3	DE	C	0,-	2,A	9,B	9,B	14,C
4	E	D	0,-	2,A	9,B	9,B	13,D

Jadi, hasil rute dalam algoritma *Dijkstra* adalah A-B-D-E ditunjukkan pada Gambar 2.2.



Gambar 2.2 Hasil *Graph* dengan Algoritma *Dijkstra*

2.4 Algoritma *A**(A-Star)

Dalam ilmu komputer, menemukan jalur terpendek dengan mengevaluasi kemungkinan jalur yang dilewati dan menghasilkan solusi yang optimal dalam grafik selalu menjadi topik yang menantang. Algoritma *A** diperkenalkan oleh Stanford Research Institute pada tahun 1968, merupakan salah satu algoritma *pathfinding* paling menjanjikan karena kinerja yang tinggi, keunggulannya dapat dilihat dari konsumsi memori dan mencari jarak terdekat untuk menghasilkan solusi yang optimal (Bagheri, dkk, 2021). Algoritma ini telah memainkan peran penting dalam berbagai disiplin ilmu, diantaranya adalah

sistem navigasi satelit untuk kendaraan jalan raya, perencanaan jalur robot, kendaraan, dan lainnya (He, dkk, 2022). Algoritma *A** memperkenalkan fungsi heuristis berdasarkan algoritma *Dijkstra* dan memilih jalur terbaik dengan menghitung nilai jarak untuk mencapai target, dengan perhitungan nilai jarak dimulai dari awal sehingga diperoleh solusi perencanaan jalur terbaik dengan menghitung nilai fungsi heuristis pada setiap *node* (Martins, dkk, 2022).

Algoritma A* merupakan salah satu varian dari metode *pencarian Best First Search* (BFS), yang bekerja dengan mengevaluasi setiap *node* berdasarkan kombinasi dua fungsi nilai jarak, yaitu *g(n)* dan *h(n)* (He, dkk, 2022). Nilai *g(n)* merepresentasikan nilai jarak yang telah dikeluarkan dari titik awal menuju simpul *n,* sedangkan *h(n)* merupakan estimasi nilai jarak dari simpul *n* menuju simpul tujuan. Kombinasi kedua nilai tersebut menghasilkan fungsi evaluasi f(n), yang dituliskan dalam bentuk persamaan 2.1 sebagai berikut:

$$f(n) = g(n) + h(n) \quad (2.1)$$

Dengan:

f(n) = Total nilai jarak evaluasi dari *node n*

g(n) = Nilai jarak aktual dari simpul awal menuju *node n*

h(n) = Estimasi nilai jarak dari *node n* menuju simpul tujuan

Sebuah fungsi heuristik dianggap efektif apabila mampu menghasilkan nilai estimasi yang mendekati nilai aktual. Salah satu metode heuristik yang umum digunakan dalam pencarian jalur terpendek pada struktur graph adalah *Euclidean Distance*. Fungsi ini didasarkan pada jarak lurus tanpa hambatan antara dua titik, dan sering dimanfaatkan untuk mengukur tingkat kemiripan antar vektor (Nakagawa, dkk, 2021). Perhitungan Euclidean Distance dapat dilakukan menggunakan rumus 2.2 dan 2.3 berikut :

$$h(AB) = \sqrt{\sum_{k=1}^{n} (x_{Ak} - x_{Bk})^2} \quad (2.2)$$

$$h(AB) = \sqrt{\left(x_{A1} - x_{B1}\right)^2 + \left(x_{A2} - x_{B2}\right)^2} \quad 2.3$$

Dengan:

h(AB) = Estimasi dari jarak A ke B

x_{A1} = Koordinasi X pada titik A

x_{A2} = Koordinasi Y pada titik A

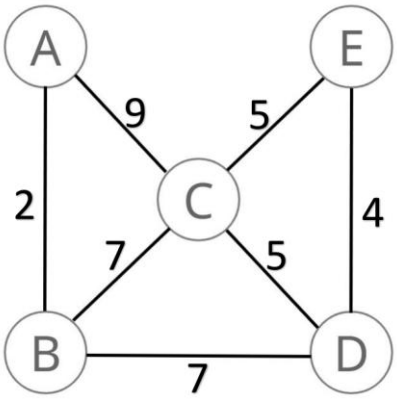
x_{B1} = Koordinasi X pada titik B

x_{B2} = Koordinasi Y pada titik B

Langkah-langkah algoritma *A** adalah sebagai berikut (Nakagawa, dkk, 2021):

1. *Node* "*start*" sebagai *successor*.
2. Menghitung semua nilai jarak evaluasi yang terhubung dari *successor*.
3. Dilakukan proses evaluasi terhadap simpul penerus (*successor*), apabila simpul tersebut merupakan simpul tujuan, maka pencarian dihentikan. Namun, jika bukan, maka proses dilanjutkan ke tahapan selanjutnya.
4. Tentukan *successor* baru dari nilai jarak evaluasi yang paling baik (nilai jarak paling minimum) dan alfabet order (apabila ada kedua nilai jarak evaluasi memiliki nilai yang sama) dari *successor-successor* sebelumnya.
5. Kemudian kembali ke langkah 4.

Sebagai ilustrasi akan ditentukan jalur terpendek dari *graph* yang ditunjukkan pada Gambar 2.3 dengan titik asal = A, dan titik tujuan = E. Setiap titik memiliki koordinat, yaitu: A(0,0), B(2,3), C(0,5), D(4,4), E(4,8).



Gambar 2.3 Soal *Graph* dengan Algoritma *A**

Langkah 1:

$h(AE) = \sqrt{(X_E - X_A)^2 + (Y_E - Y_A)^2}$

$h(AE) = \sqrt{(6-0)^2 + (8-0)^2}$

$h(AE) = \sqrt{(6)^2 + (8)^2}$

$h(AE) = \sqrt{36 + 64}$

$h(AE) = \sqrt{100}$

$h(AE) = 10$

$h(BE) = \sqrt{(X_E - X_B)^2 + (Y_E - Y_B)^2}$

$h(BE) = \sqrt{(6-2)^2 + (8-3)^2}$

$h(BE) = \sqrt{(4)^2 + (5)^2}$

$h(BE) = \sqrt{16 + 25}$

$h(BE) = \sqrt{41}$

$h(BE) = 6{,}4$

$h(CE) = \sqrt{(X_E - X_C)^2 + (Y_E - Y_C)^2}$

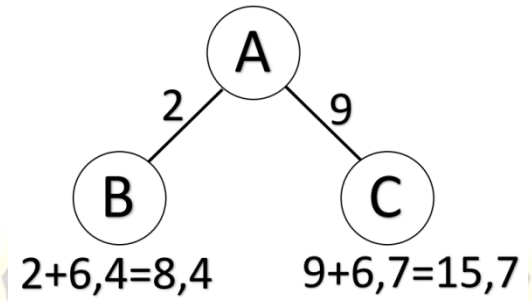
$h(CE) = \sqrt{(6-0)^2 + (8-5)^2}$

$h(CE) = \sqrt{(6)^2 + (3)^2}$

$h(CE) = \sqrt{36 + 9}$

$h(CE) = \sqrt{45}$

$h(CE) = 6{,}7$



Langkah 2:

Nila *path h(BE)* memiliki nilai 8,4 adalah *path* paling kecil maka *path* berikutnya dilanjutkan dari *path h(BE)*.

$h(DE) = \sqrt{(X_E - X_D)^2 + (Y_E - Y_D)^2}$

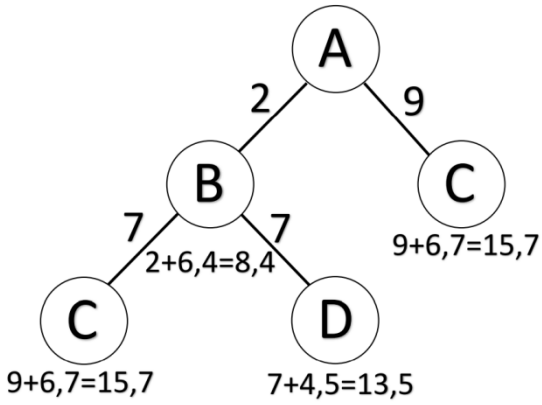
$h(DE) = \sqrt{(6-4)^2 + (8-4)^2}$

$h(DE) = \sqrt{(2)^2 + (4)^2}$

$h(DE) = \sqrt{4 + 16}$

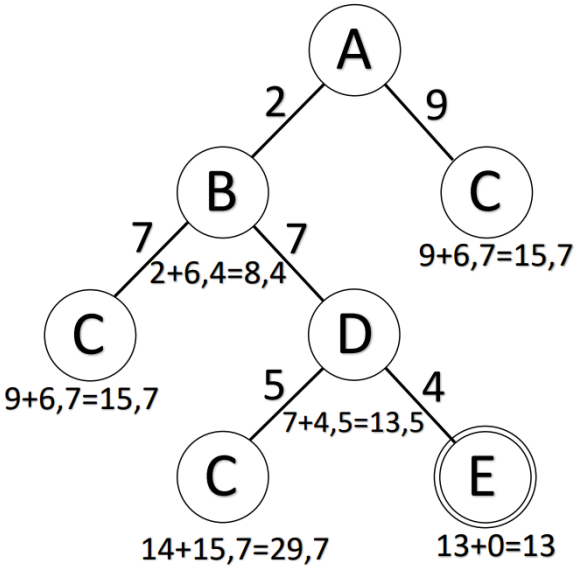
$h(DE) = \sqrt{20}$

$h(DE) = 4{,}5$

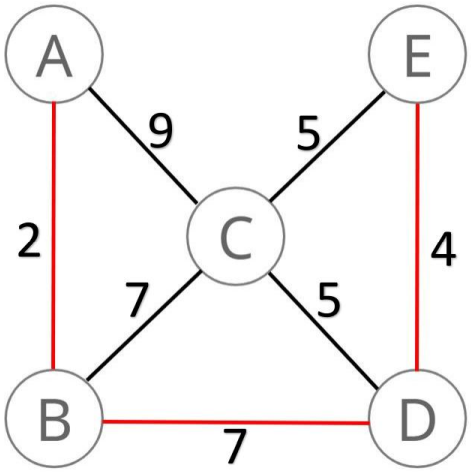


Langkah 3:

Nila *path h(DE)* memiliki nilai 13,5 adalah *path* yang paling kecil maka *path* berikutnya dilanjutkan dari *path h(DE)*.



Dengan demikian, jalur yang diperoleh melalui penerapan Algoritma *A* (A-Star)* adalah A–B–D–E, sebagaimana ditampilkan pada Gambar 2.4.



Gambar 2.4 Hasil *Graph* dengan Algoritma *A**