Penerapan Shortest Path First dalam Routing Protocol Menggunakan Pendekatan GBFS, UCS, dan A*

Febryan Arota Hia - 13521120
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521120@std.stei.itb.ac.id

Abstract—Algoritma Shortest Path First (SPF) dalam *routing protocol* merupakan salah satu contoh algoritma yang umum digunakan dalam penjadwalan/penentuan jalur jaringan. Algoritma SPF dapat menggunakan pendekatan GBFS, UCS, dan A* yang bertujuan untuk menentukan jalur terpendek antara dua node (*router)* dalam jaringan berdasarkan metrik-metrik yang telah ditentukan. Metrik tersebut dapat berupa biaya, *bandwith, hop,* jarak antar *router* atau *delay*. Ketiga pendekatan tersebut diharapkan dapat memberikan pilihan terbaik pada setiap langkah berdasarkan informasi yang tersedia untuk menghasilkan solusi optimal.

Keywords—Algoritma Shortest Path First, Algoritma A*, Greedy Best First Search

I. PENDAHULUAN

Dalam era teknologi yang semakin maju, konektivitas jaringan menjadi sangat penting dalam mendukung pertukaran informasi dan komunikasi antar perangkat komputer. Dalam sebuah jaringan komputer, *router* mempunyai peranan yang penting dalam proses pengiriman paket data. *Router* mempunya fungsi untuk mengirimkan paket data dari satu *network* ke *network* lain dengan menentukan jalur terbaik untuk mencapai *network* tujuannya. Dengan kata lain, *router* dapat diibaratkan sebagai "polisi lalu lintas" pada jaringan komputer.

Dalam menjalankan tugasnya yang kompleks, *router* menggunakan protokol *routing* yang memungkinkan *router* untuk memilih jalur terpendek atau *shortest path* untuk mencapai *network* tujuannya. Pemilihan jalur tersebut dapat ditentukan berdasarkan kriteria-kriteria tertentu seperti biaya, *bandwith*, atau jarak fisik antar *router*. Penentuan jalur sangat berdampak pada kecepatan dan efisiensi pemindahan data nantinya. Untuk itu, dibutuhkan algoritma yang dapat menentukan opsi jalur yang paling baik.

Pendekatan GBFS, UCS, dan A* dapat menjadi salah satu alternatif untuk pemilihan opsi terbaik. Pendekatan ini dapat menentukan pilihan pada setiap langkah berdasarkan informasi yang tersedia saat itu degan tujuan mencapai solusi optimal secara lokal.

II. TEORI DASAR

A. *Graf*

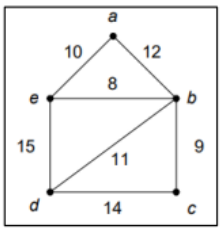
Graf adalah suatu sttruktur yang terbentuk dari simpul dan sisi. Graf didefinisikan sebagai $G = (V, E)$, dengan V adalah himpunan yang berisi simpul (vertices) dan E adalah himpunan yang berisi sisi (edges). Sisi merepresentasikan hubungan antara simpul-simpul pada graf

$$V = \{v1 , v2 , v3 , \cdots vn\}$$

$$E = \{e1 , e2 , e3 , \cdots en\}$$

Graf dapat dibagi menjadi dua berdasarkan ada atau tidaknya bobot pada sisinya, yaitu

1. Graf tak berbobot (*uneweighted graph*): graf yang tidak memiliki bobot pada sisinya
2. Graf berbobot (*weighted graph*): graf yang memiliki bobot pada sisinya



Gambar 1 Contoh Graf Berbobot
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

Terminologi Graf:

1. Ketetanggaan (*Adjacent*): Dua buah simpul dikatakan bertetangga bila keduanya terhubung langsung.
2. Bersisian (*Incidency*): pada Gambar 1 sisi(a,b) bersisian dengan simpul a dan b
3. Lintasan: berisan yang menghubungkan simpul-simpul yang berselang-seling dengan sisi-sisi membentuk suatu lintasan.

B. *Jaringan Komputer*

Jaringan merupakan kumpulan perangkat-perangkat yang saling terhubung dan memungkinkan adanya komunikasi antara pengguna dan perangkat dengan mentransfer paket data. Pada penerapannya, jaringan merupakan salah satu aplikasi dari graf berarah. Dalam jaringan, *node* dapat dijadikan sebagai representatif dari perangkat keras seperti komputer, laptop, *smarth phone, router*, dan lainnya). Lalu, busur dapat dinyatakan sebagi penghubung atau konektivitas antar perangkat tesebut.

Menurut Iwan sofana (2013) "Jaringan komputer disebut juga himpunan 'interkoneksi' antara dua komputer anoumous atau lebih yang terhubung dengan media transmisi kabel atau tanpa kabel dan sebuah komputer dapat membuat komputer lainnya restart, shutdown, atau melakukan kontrol lainya, maka komputer-komputer tersebut bukan anoumous (tidak melakukan kontrol terhadap komputer lain dengan akses penuh)".

Secara umum jaringan komputer terbagi menjadi tiga jenis, yaitu:

a. Local Area Network (LAN)

   LAN adalah jaringan lokal yang mencakup area yang relatif kecil. Jaringain ini digunakan untuk menghubungkan perangkat-perangkat di area terbatas seperti perangkat-perangkat di dalam sebuah gedung atau kantor.

b. Metropolitan Area Network (MAN)

   MAN adalah jaringan yang mencakup area yang lebih besar dari LAN, seperti area sebuah kota atau wilayah. Jaringan ini menghubungkan beberapa LAN yang berdekatan dengan menggunakan kabel. MAN dapat digunakan oleh organisasi besar atau pemerintahan

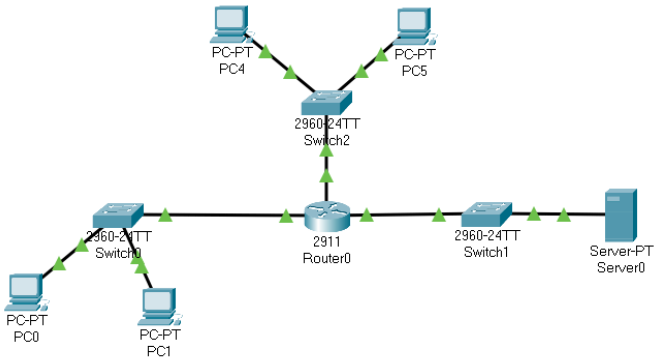
c. Wide Area Network (WAN)

   WAN adalah jaringan luas yang mencakup area geografis yang luas sepeti wilayah negara, benua, atau seluruh dunia. Jaringan ini menghubungkan berbagai lokasi yang terpisah secara geografis seperti jalur telepon, satelit, atau koneksi *wireless.*

C. *Routing Protocol*

Routing protocol adalah aturan atau cara pencarian jalur terbaik yang digunakan untuk mengirimkan paket data dari *node* pengirim (*sender*) ke *node* penerima (*receiver*). Paket tersebut akan melewati beberapa *node* yang terhubung melalui mekanisme pembentukan tabel *routing.* Setiap *routing protocol* memiliki cara dan metode yang berbeda dalam melakasanakan tugasnya.

Setiap *routing protocol* memiliki tujuan utama yaitu mencari jalur terbaik atau optimal untuk mengirimkan paket data antar *node* pengirim dan *nodeI* penerima. Dalam jaringan komputer, *node* dapat berupa *router, switch,* atau perangkat lain. *Routing protocol* menggunakan mekanisme pembentukan tabel untuk menyimpan informasi tentang jariangan untuk membuat keputusan jalur mana yang harus diambil atau dilewati oleh paket data.



Gambar 1 Ilustrasi *Routing*
Sumber: https://www.computernetworkingnotes.com/ccna-study-guide/types-of-routing-and-types-of-routes-explained.html

Routing protocol menggunakan metrik-metrik yang ditentukan untuk memilih jalur yang terbaik. Metrik adalah suatu nilai hasil dari perhitungan algoritma yang dipakai oleh *routing protocol*. Metrik tersebut dapat berupa jarak ke tujuan atau biaya ke tujuan. Jenis metrik yang digunakan bergantung pada jenis *routing protocol* yang digunakan.

Berikut beberapa metrik *routing* yang sering digunakan:

a. Hop Count: Metrik ini akan menghitung jumlah *router* atau *hop* yang harus dilewati oleh paket data untuk mencapai *node* tujuan. Setiap *hop* memiliki nilai tetap sehingga jalur dengan jumlah *hop* terkecil dijadikan sebagai jalur terbaik.

b. Bandwith: Metrik ini akan menentukan jalur terbaik dengan menggunakan kapasitas atau kecepatan *link* sebagai penentu. Jalur dengan *bandwith* yang lebih besar dianggap sebagai jalur yang terbaik.

c. Delay: Metrik ini mengacu pada waktu yang dibutuhkan oleh paket data untuk melewati satu *router* atau *link*. Jalur dengan *delay* terendah akan dianggap sebagai jalur yang terbaik.

d. Cost: Metrik ini mengacu pada biaya yang terkait dengan penggunaan jalur atau *link*. Cost dapat bergantung pada banyak faktor. Faktor-faktor tersebut ditentukan oleh administrator jaringan. Jalur dengan cost terendah dianggap sebagai jalur terbaik.

e. Load: Metrik ini mengacu pada *load* yang diukur dengan tingkat pembebanan dalam penggunaan suatu jalur. Metrik ini mencerminkan jumlah lalu lintas atau paket yang sedang ditangani pada jalur tersebut.

Setiap *routing protocol* memiliki cara dan pendekatan yang berbeda dalam penggunaan metrik. Pemilihan metrik yang digunakan bergantung pada kebutuhan dan karakterisitik jaringan. *Routing protocol* mungkin saja menerapkan lebih dari satu metrik untuk menentukan jalur terbaik.

D. Algoritma Greedy

Algoritma *greedy* adalah algoritma yag memecahkan masalah dengan tamak yakni mengambil solusi terbaik pada setiap iterasi dengan harapan bahwa solusi optimum lokal tersebut dapat menghasilkan solusi optimum global juga. Algoritma ini cukup populer dan mengandalkan pendekatan yang sederhana untuk persoalan optimalisasi, yakni masalah mengenai pencarian maksimasi dan minimasi. Akan tetapi algoritma ini mempunyai kelemahan yaitu hanya dapat mencari nilai optimul lokal, bukan optimum global yang sebenarnya diinginkan.

Secara umum, langkah-langkah algoritma *greedy* adalah sebagai berikut:

1. Mengambil pilihan terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi pada langkah ke depannya.
2. Berharap bahwa dengan mimilih optimal lokal pada setiap langkah akan berakhir dengan optimum global.

Berikut merupakan elemen-elemen algoritma *greedy*:

1. Himpunan kandidat, C: berisi kandidat yang akan dipilih pada setiap langkah
2. Himpunan solusi, S: berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dpilih sudah memberikan solusi
4. Funsgi seleksi (selection function): memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi *greedy* ini bersifat heuristik.
5. Fungsi kelayakan (*feasible*): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif: memaksimumkan atau meminimum-kan

E. Algortima Greedy Best First Seacrh

Greedy Best First Search atau yang sering disebut dengan Best First Search adalah algoritma yang digunakan untuk mencari nilai optimal. Algoritma ini berfokus pada pengenmbangan pohon pencarian dengan mempertimbangkan nilai heuristik pada setiap langkah pencarian.

Algoritma ini menggunakan fungsi evaluasi *f(n)* untuk setiap node. Pada setiap langkah, *greedy best first search* memilih simpul yang memiliki nilai heuristik terendah.

f(n) = h(n), dimana *h(n)* adalah fungsi estimasi atau heuristik dari *n* ke tujuan. Funsgi heuristik dapat dibuat berbagai macam. Misalnya manhattan distance, yaitu mendapatkan hasil jarak hanya dengan menjumlahkan perbedaan absis dan ordinat antara titik awal dan tujuan.

Greedy best first search mengabaikan nilai cost/biaya yang sebenarnya untuk mencapai simpul tertentu. Algoritma ini sesuai namanya, cenderung serakah karena hanya mempertimbangkan simpul dengan nilai heuristik terendah pada setiap langkah. Oleh karena itu, algoritma ini tidak menjamin untuk selalu memberikan solusi optimal. Namun, algoritma ini mempunyai kelebihan dalam memberikan solusi dengan cepat dan relatif mudah untuk diimplementasikan.

F. Algortima UCS

Algoritma UCS merupakan algoritma pencarian yang menggunakan biaya kumulatif terendah untuk menemukan jalur dari *node* asal ke *node* tujuan. Algoritma ini dikategorikan sebagai algoritma pencarian tak berinformasi atau *blind search* karena tidak menggunakan informasi heuristik tentang lokasi atau jarak ke *node* tujuan.

Implementasi untuk algoritma ini menggunakan *f(n) = g(n)* dimana *g(n)* adalah cost untuk setiap simpulnya.

Algoritma UCS memberikan solusi optimal dalam hal biaya karena mencari jalur dengan biaya terendah pada setiap langkah. Namun, algoritma ini mungkin tidak efisien dalam beberapa kasus.

*G. Algortima A**

Algoritma A star (A*) adalah algoritma pencarian jalur terpendek pada graf berbobot yang digunakan untuk menemukan jalur terpendek antara dua titik dalam graf. Algoritma ini merupakan perluasan dari algoritma Dijkstra dengan memasukkan informasi heuristik untuk mengoptimalkan pencarian.

Algoritma A star menggunakan dua jenis informasi untuk mengevaluasi setiap simpul atau *node* pada graf:

a. Biaya sejauh ini g(n): yaitu biaya terkecil yang telah ditemukan dari titik awal hingga simpul saat ini
b. Perkiraan biaya h(n): yaitu perkiraan biaya yang diperlukan untuk mencapai simpul tujuan dari simpul saat ini

Perkiraan biaya ini diperoleh dengan menggunakan fungsi heuristik yang memperkirakan jarak atau waktu yang diperlukan untuk mencapai simpul tujuan. Salah satu contoh fungsi heuristik yang umum digunakan adalah jarak euclidean atau jarak Manhattan. Setiap simpul atau *node* pada graf dinilai berdasarkan kombinasi biaya sejauh ini dan perkiraan biaya untuk mencapai simpul tujuan

F(n) = g(n) + h(n), Algoritma A star memilih simpul dengan nilai *f(n)* untuk diproses selanjutnya. Proses ini berlanjut hingga simpul tujuan ditemukan atau tidak ada simpul lagi yang dapat diekspan.

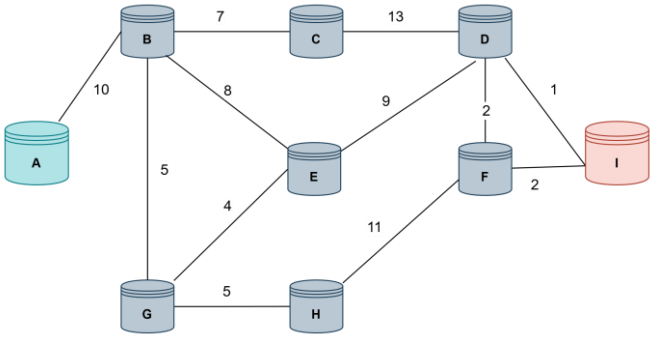
Dibandingkan dengan algoritma *greedy best first search*, algoritma A* selalu memberikan hasil optimum apabila fungsi heuristik yang digunakan memenuhi sifat *admissible*. Ini berarti algoritma A* akan selalu menemukan solusi dengan biaya minimal jika fungsi heuristik yang digunakan memenuhi syarat.

III. PEMBAHASAN

A. *Gambaran Kasus*

Dari teori dasar yang telah dijelaskan sebelumnya, kita bisa mengetahui bahwa *routing protocol* menggunakan konsep *shorthest first path* untuk menentukan jalur terbaik dari pengirim ke penerima. Jalur terpendek tersebut dapat ditentukan dengan menggunakan algoritma *greedy best first search* dan algoritma A*.

Sebagai contoh kasus, misalkan kita mempunyai sebuah struktur jaringan sebagai berikut.



Gambar 2 Struktur Jaringan pada Contoh Kasus
Sumber: Dokumen penulis

Pada gambar 2, simpul A berperan sebagai pengirim dan simpul I sebagai penerima. Busur pada gambar merepresentasikan waktu *delay* yang dialami ketika proses pengiriman data dari satu *router* ke *router* sedang berlangsung. Pada kasus ini digunakan jumlah hop (*router* yang perlu dilewati) menuju simpul penerima sebagai penentu fungsi heuristik *f(n)* dalam pemilihan jalur serta total waktu *delay* dari root sebagai *g(n)*.

B. *Routing dengan Greedy Best First Search*

Dalam menentukan jalur menggunakan *greedy best first search* digunakan fungsi heuristik sebagai penentu. Seperti yang sudah disebutkan sebelumnya, pada kasus ini digunakan jumlah *hop* dari suatu *router* hingga sampai ke penerima sebagai *f(n) = h(n)*.

Untuk menentukan jumlah *hop* pada setiap simpul kita hanya perlu menjumlahkan total minimal simpul yang dilewati untuk mencapai simpul akhir (dalam kasus ini simpul I).

Misal, untuk menentukan *hop* simpul A:

1. A – B – C – D – I = 4 *hop*
2. A – B – G – H – F – I = 5 *hop*
3. A – B – G – E – D – F – I = 6 *hop*
4. A – B – E – D – F – I = 5 *hop*
5. A – B – G – E – D – I = 5 *hop*

   Dst.

Dari seluruh kemungkinan jalur A ke I, ditemukan jumlah *hop* minimal yaitu 4 *hop*. Dengan cara yang sama, tentukan masing-masing jumlah *hop* menuju penerima dari tiap *router*.

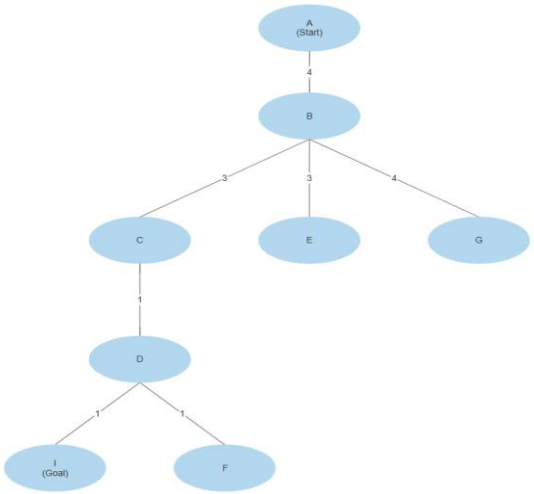
Berikut adalah jumlah *hop* masing-masing *router*.

Router	A	B	C	D	E	F	G	H	I
Hop	4	3	2	1	2	1	3	2	0

Tabel 1 Data *Hop Router*

Dengan menggunakan algoritma GBFS pemilihan simpul akan ditentukan berdasarkan *h(n)*. Pada kasus ini, pencarian jalur akan dimulai pada simpul start yaitu simpul, lalu diperiksa seluruh simpul tetangganya. Simpul tetangga yang memiliki nilai *hop* paling kecil akan dipilih. Simpul yang dipilih tersebut diperiksa kembali seluruh simpul tetangganya, lalu dipilih simpul dengan nilai *hop terkecil*. Langkah tersebut dilakukan seterusnya hingga sampai ke simpul tujuan. Apabila dalam proses pencarian terjadi pencarian buntu, maka pencarian akan dilakukan pada *node* tetangga sebelumnya yang terkecil kedua.

Pada kasus ini jalur yang dihasilkan pencarian adalah A – B – C – D – I dengan total waktu *delay* 10 + 7 + 13 + 4 = 34 satuan. Berikut adalah ilustrasi pohon pencariannya.

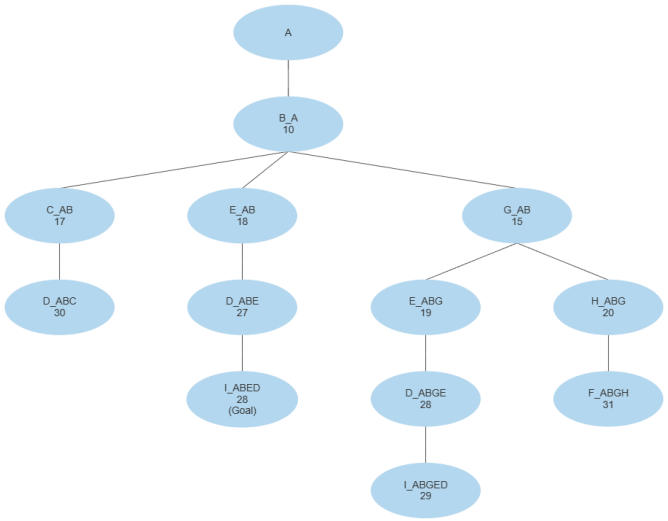


Gambar 3 Pohon Pencarian Menggunakan GBFS
Sumber: Dokumen penulis

C. *Routing dengan UCS*

Dalam menentukan jalur menggunakan *UCS* digunakan total waktu *delay* dari root atau simpul start sebagai penentu. Seperti yang sudah disebutkan sebelumnya, pada kasus ini digunakan total waktu delay sebagai *f(n) = g(n)*. Untuk menentukan jalur pada UCS, digunakan konsep *priority queue* untuk menentukan simpul ekspansi dan simpul hidup dimana semakin kecil jumlah waktu *delay* akan diletakkan semakin depan pada *queue* simpul hidup karena *queue* dibuat terurut membesar berdasarkan *f(n)*. Untuk setiap simpul yang telah diekspansi, tidak perlu lagi dimasukkan ke dalam simpul hidup setelahnya.

Berikut adalah simpul ekspansi dan simpul hidup pada pencarian dengan menggunakan algoritma UCS.



Gambar 4 Pohon Pencarian Menggunakan UCS
Sumber: Dokumen penulis

Iterasi	Simpul-E	Simpul Hidup
1	A	B_{A-10}
2	B_{A-10}	G_{AB-15} - C_{AB-17} – E_{AB-18}
3	G_{AB-15}	C_{AB-17} – E_{AB-18} – E_{ABG-19} – H_{ABG-20}
4	C_{AB-17}	E_{AB-18} – E_{ABG-19} – H_{ABG-20} – D_{ABC-30}
5	E_{AB-18}	E_{ABG-19} – H_{ABG-20} – D_{ABE-27} – D_{ABC-30}
6	E_{ABG-19}	H_{ABG-20} – D_{ABE-27} – $D_{ABGE-28}$ – D_{ABC-30}
7	H_{ABG-20}	D_{ABE-27} – $D_{ABGE-28}$ – D_{ABC-30} – $F_{ABGH-31}$
8	D_{ABE-27}	$D_{ABGE-28}$ – $I_{ABED-28}$ – D_{ABC-30} – $F_{ABGH-31}$
9	$D_{ABGE-28}$	$I_{ABED-28}$ – $I_{ABGED-29}$ – D_{ABC-30} – $F_{ABGH-31}$
10	$I_{ABED-28}$	Goal

Tabel 2 Pencarian Jalur Terpendek UCS

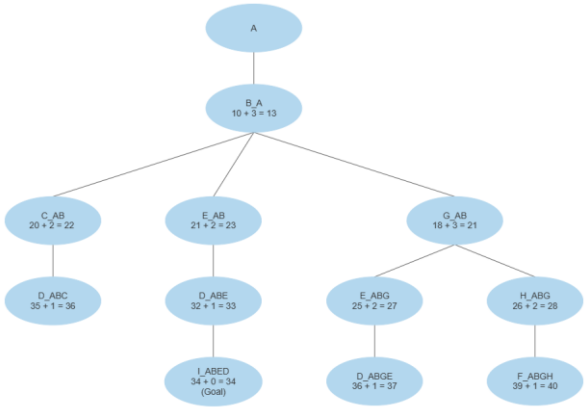
Jalur yang dihasilkan oleh algoritma UCS adalah A – B – E - D – I dengan total waktu *delay* 28 satuan.

*D. Routing dengan Algoritma A**

Dalam menentukan jalur menggunakan A* digunakan total waktu *delay* dari root atau simpul start sebagai *g(n)* dan jumlah *hop* sebagai fungsi heuristik *h(n)*. Seperti yang sudah disebutkan sebelumnya, pada kasus ini digunakan perhitungan untuk setiap *node*, *f(n) = g(n) + h(n).*

Langkah pertama yang dilakukan adalah mengekspan simpul awal yaitu simpul A. Dari simpul A kita dapat memilih simpul B dengan cost 10 (waktu *delay*) + 3 (jumlah *hop* simpul B) = 13. Karena tidak ada lagi simpul yang bertetangga pada simupul A, maka selanjutnya kita ekspan simpul B. Pada simupl B terdapat beberapa simpul tetangga yaitu A, C, E dan G. Untuk menghitung masing-masing cost, gunakan cara yang sama seperti menghitung cost simpul B dari A. Untuk simpul A, kita tidak perlu lagi memasukkannya pada simpul hidup karena pada tahap sebelumnya kita telah mengekspan simpul tersebut.

Misal, untuk menghitung cost simpul C, *g(n)* = 23 + 7 = 20, dengan *h(n)* = 2. Makan *f(n)* = 20 + 2 = 22. Gunakan cara yang sama untuk setiap simpul pada simpul hidup yang terurut membesar. Lalu pilih simpul dengan cost terkecil untuk diekspan kembali hingga akhirnya ditemukan simpul akhir dengan cost terkecil. Berikut adalah pohon hasil pencarian jalur menggunakan algoritma A*.



Gambar 5 Pohon Pencarian Menggunakan A*
Sumber: Dokumen penulis

Iterasi	Simpul-E	Simpul Hidup
1	A	B_{A-13}
2	B_{A-13}	G_{AB-21} - C_{AB-22} – E_{AB-23}
3	G_{AB-21}	C_{AB-22} – E_{AB-23} – E_{ABG-27} – H_{ABG-28}
4	C_{AB-22}	E_{AB-23} – E_{ABG-27} – H_{ABG-28} – D_{ABC-36}
5	E_{AB-23}	E_{ABG-27} – H_{ABG-28} – D_{ABE-33} – D_{ABC-36}
6	E_{ABG-27}	H_{ABG-28} – D_{ABE-33} – D_{ABC-36} – $D_{ABGE-37}$
7	H_{ABG-28}	D_{ABE-33} – D_{ABC-36} – $D_{ABGE-37}$ – $F_{ABGH-40}$
8	D_{ABE-33}	$I_{ABED-34}$ – D_{ABC-36} – $D_{ABGE-37}$ – $F_{ABGH-41}$
9	$I_{ABED-34}$	Goal

Tabel 3 Pencarian Jalur Terpendek A*

Berdasarkan hasil pencarian menggunakan algoritma A*, didapatkan jalur akhir yaitu A – B – E – D – I dengan total waktu *delay* sebanyak 28 satuan.

E. Hasil

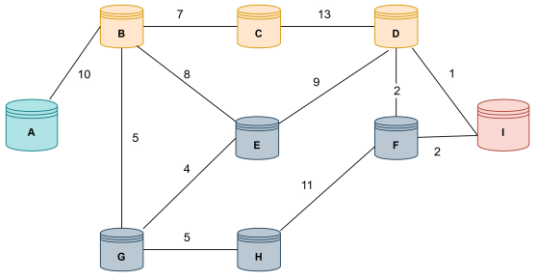
Berdasarkan ketiga algoritma pencarian, didapatkan hasil jalur akhir sebagai berikut.

	GBFS	UCS	A*
Hasil	A-B-C-D-I	A-B-E-D-I	A-B-E-D-I
Total *delay*	34	28	28
Iterasi	5	10	9

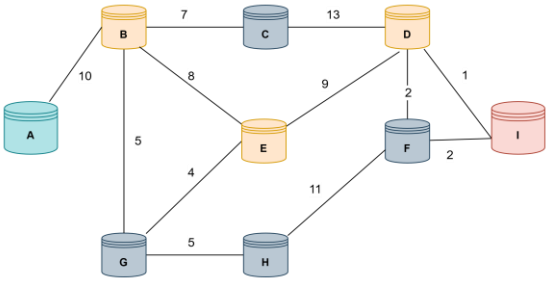
Tabel 4 Hasil Pencarian dari Ketiga Algortima

Berdasarkan hasil yang dapat dilihat pada Tabel 4, Algoritma UCS dan A* menghasilkan jalur yang sama yaitu A-B-E-D-I, sedangkan GBFS menghasilkan jalur A-B-C-D-I. Dari jalur tersebut, algoritma UCS dan A* menghasilkan waktu *delay* yang optimal dengan nilai sebesar 28 satuan. Sedangkan GBFS menghasilkan waktu *delay* yang tidak optimal yaitu 34 satuan.

Dalam aspek iterasi, GBFS hanya perlu melakukan 5 kali pemeriksaan, sedangkat UCS dan A* masing-masing memerlukan 10 dan 9 kali iterasi pemeriksaan.



Gambar 6 Hasil Akhir Algoritma GBFS
Sumber: Dokumen penulis



Gambar 7 Hasil Akhir Algoritma A* dan UCS
Sumber: Dokumen penulis

IV. KESIMPULAN

Dengan demikian, berdasarkan hasil pencarian yang telah dibahas untuk masing-masing algoritma, dapat disimpulkan bahwa algoritma A* dan UCS memberikan hasil yang lebih baik dibandingkan dengan GBFS dalam aspek total *delay*. Namun, dalam hal jumlah iterasi, GBFS memiliki performa yang lebih baik dibandingkan A* dan UCS.

REFERENCES

[1] Oris Krianto "ANILISIS JARINGAN DENGAN ROUTING PROTOKOL BERBASIS SPF (SHORTEST PATH FIRST) DJIKSTRA ALGORITHM"

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf diakses pada Mei 2023

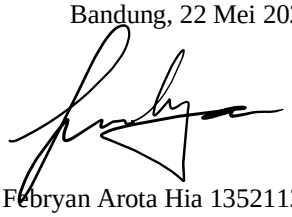
[3] Hari Antoni Musril, "PENERAPAN OPEN SHORTEST PATH FIRST (OSPF) UNTUK MENENTUKAN JALUR TERBAIK DALAM JARINGAN" 2017

[4] Oris Krianto, Mohamad Ihwani "Analisis Perbandingan Penggunaan Metric Cost dan Bandwidth Pada Routing Protocol OSPF" April 2017

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023



Febryan Arota Hia 13521120