Aplikasi Informed dan Uninformed Search Dalam Penentuan Rute Liburan Keluarga

Fadhil Muhammad Rafi' 13518079
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518079@std.stei.itb.ac.id@itb.ac.id

***Abstrak*—Keluarga merupakan suatu satuan masyarakat yang terdiri dari orang tua dan anak yang memiliki rasa empati dan sayang antaranggotanya. Rasa sayang dan empati tersebut harus dibina dan dibentuk dari waktu ke waktu melalui interaksi yang mereka bangun. Salah satu cara meningkatkan interaksi tersebut biasanya dilakukan dengan melakukan wisata di masa liburan. Untuk dapat melaksanakan liburan dengan dana yang mangkus, keluarga tersebut harus dapat menentukan rute liburan yang tepat sehingga tidak boros mengeluarkan biaya untuk pindah dari satu daerah ke daerah lainnya. Dalam dunia informatika, terdapat algoritma pencarian rute yang dapat membantu tiap keluarga dalam menentukan rute liburannya sehingga dana yang dikeluarkan adalah minimum. Algoritma pencarian rute tersebut dibagi menjadi dua, yaitu *Uninformed Search* yang tidak menggunakan data tambahan dan *Informed Search* yang memerlukan data heuristik yang diperlukan untuk pencarian rute.**

***Keywords*—Rute, Liburan, *Informed Search*, *Uninformed Search*, mangkus.**

I. PENDAHULUAN

Sebuah keluarga selalu berusaha untuk meningkatkan kekeluargaannya dengan memperbanyak interaksi serta meningkatkan kualitas interaksi mereka. Salah satu caranya adalah dengan melakukan liburan bersama keluarga. Liburan merupakan salah satu bentuk interaksi yang berkualitas bagi keluarga sehingga dapat meningkatkan rasa kekeluargaan di antara angora keluarga. Namun, biaya yang dikeluarkan untuk melaksanakan liburan tidaklah murah, banyak biaya yang harus dikeluarkan tidak terkecuali biaya operasional dalam perjalanan. Pemangkasan biaya operasional berdasarkan jarak tempuh dari tiap *checkpoint* dapat dilakukan untuk mengurangi dana yang dikeluarkan tanpa mengurangi kualitas liburan.

Sebuah keluarga harus dapat menentukan rute perjalanan liburannya dengan tepat sehingga dapat meminimalisasi dana yang harus dikeluarkan oleh keluarga tersebut. Dengan pandangan bahwa biaya yang dikeluarkan selama perjalanan berbanding lurus dengan jarak tempuh antar *checkpoint*, tanpa menghiraukan kondisi lintasan jalan dan faktor X lainnya, suatu keluarga dapat menentukan rute perjalanan terpendek yang dapat ditempuh dengan menggunakan algoritma pencarian rute. Dalam masalah ini, algoritma pencarian rute yang akan digunakan dapat dibagi menjadi dua jenis, yaitu *Informed Search* dan *Uninformed Search.*

Algoritma pencarian adalah mekanisme pemecahan masalah yang paling umum di dalam inteligensi buatan. Di dalam permasalahan-permasalahan kecerdasan buatan, urutan langkah-langkah yang dibutuhkan untuk memperoleh solusi merupakan suatu isu yang penting untuk diformulasikan. Hal ini harus dilakukan dengan mengidentifikasikan proses *try and error* secara sistematis pada eksplorasi setiap alternatif jalur yang ada. Algoritma pencarian secara umum dibagi menjadi dua, seperti yang sudah disebutkan sebelumnya, yaitu *Informed Search* dan *Uninformed Search.* Dalam masalah pencarian rute ini, *Uninformed Search* tidak membutuhkan data tambahan untuk melakukan penghitungan jarak antar-*checkpoint* sedangkan *Informed Search* membutuhkan data heuristik sebagai tambahan untuk melakukan penghitungan rute yang paling optimal.

II. TEORI DASAR

A. *Uninformed Search*

Uninformed Search biasa disebut dengan *Blind Search*. Hal ini menunjukkan bahwa algoritma pencarian ini adalah algoritma yang "buta" atau tidak memiliki informasi tambahan mengenai kondisi di luar status yang saat ini sedang dihadapinya. Algortima ini melakukan pencarian dengan basis graf yang berisikan simpul-simpul yang mewakilkan status-status yang dapat dicapai dalam pemecahan suatu masalah. Simpul tujuan harus sudah diketahui di awal sehingga ketika mencapai simpul tujuan, algoritma pencarian ini dapat berhenti tanpa melakukan pengecekan pada simpul-simpul yang lain. Simpul-simpul tersebut akan didaftarkan pada *queue* dan simpul yang terletak di pangkal *queue* akan dibangkitkan sehingga tiap simpul anak dari simpul tersebut akan dihidupkan dan didaftarkan juga pada *queue*. Algoritma *Uninformed Search* terdiri dari beberapa jenis, antara lain *Breadth-First Search* (BFS), *Depth-First Search* (DFS), *Uniform Cost Search* (UCS), *Depth-Limited Search* (DLS), dan *Iterative Deepening Depth-First Search* (IDS).

B. *Breadth-First Search* (BFS)

Pencarian dengan *Breadth-First Search* menggunakan teknik dimana langkah pertamanya adalah mengekspansi simpul akar dari pohon yang kemudian akan menghasilkan simpul hidup dari suksesor-suksesor simpul akar tersebut, setelah itu dilanjutkan

semua suksesor dari simpul akar juga ekspansi dan menghasilkan simpul hidup yang lainnya lagi. Hal ini terus dilakukan berulang-ulang hingga pencarian mencapai ke daun pohon atau hingga tingkat yang paling bawah dimana simpul tidak memiliki suksesor lagi. Namun, perlu diingat algoritma pencarian ini akan berhenti ketika mencapai simpul tujuannya.

C. *Depth-First Search* (DFS)

Teknik pencarian dengan *Depth-First Search* adalah dengan terus melakukan ekspansi menuju simpul yang paling dalam pada tree atau biasa disebut daun. Simpul terdalam atau daun dicirikan dengan tidak adanya suksesor dari simpul itu. Setelah simpul itu selesai diekspansi, maka simpul tersebut akan ditinggalkan, dan dilakukan pencarian kembali ke simpul yang belum diekspansi dan pencarian juga akan terus dilakukan hingga ke simpul daun. Metode pencarian ini akan terus diulang hingga seluruh simpul sudah dibangkitkan atau menemukan simpul solusi.

D. *Uniform Cost Search* (UCS)

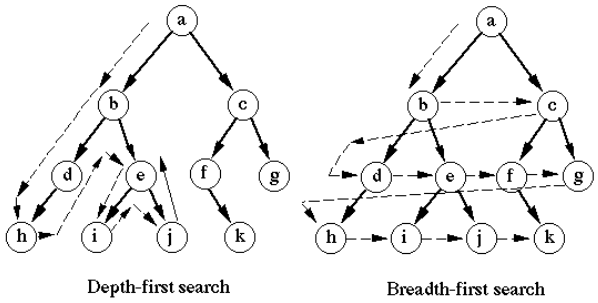
Algoritma pencarian dengan menggunakan *Breadth-First Search* akan menjadi optimal ketika menemukan nilai yang minimum pada suatu jalur. Dengan sedikit perluasan, dapat ditemukan sebuah algoritma yang dapat menentukan nilai optimal dengan melihat nilai pada tiap simpul di antara simpul-simpul yang ada. Selain menjalankan fungsi algoritma BFS, *Uniform Cost Search* melakukan ekspansi simpul dengan nilai simpul yang paling kecil. Hal ini dilakukan untuk memangkas waktu yang dibutuhkan untuk menemukan suatu simpul solusi. Pada algoritma BFS, kita perlu melakukan ekspansi pada tiap simpul, namun pada algoritma ini kita hanya perlu melakukan ekspansi pada simpul yang memiliki nilai minimum sehingga dapat memangkas banyak waktu dalam menemukan solusi. Hal ini bisa dilakukan dengan membuat antrian pada suksesor yang ada berdasarkan nilai simpul (simpul disimpan dalam bentuk *priority queue*).

E. *Depth-Limited Search* (DLS)

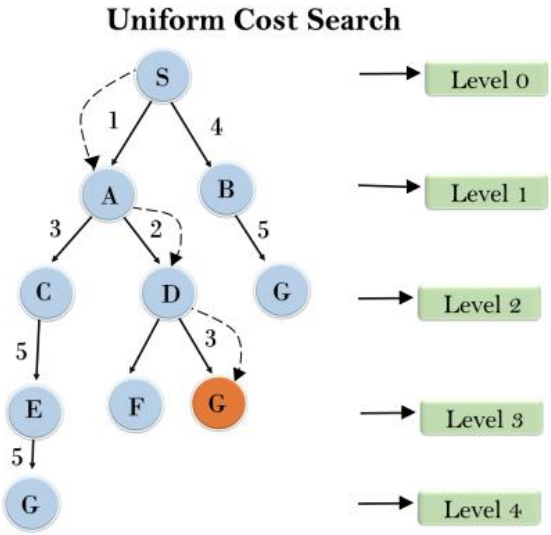
Algoritma pencarian dengan menggunakan DFS akan berlanjut terus sampai menuju kedalaman paling akhir dari suatu pohon. Permasalahan yang muncul pada DFS adalah ketika proses pencarian tersebut menemui *infinite-state space* sehingga memiliki kemungkinan besar mencapai tujuan dengan jalur yang tidak mangkus. Hal ini disebabkan oleh mekanisme algoritma ini yang terus mencari hingga ke simpul paling akhir terlebih dahulu pada tiap jalurnya. Hal ini bisa diatasi dengan menginisiasikan batas kedalaman (*depth*) pada level tertentu semenjak awal pencarian sehingga node pada tingkat kedalaman tersebut akan diperlakukan seolah-olah mereka tidak memiliki simpul suksesor

F. *Iterative Deepening Depth-First Search* (IDS)

Iterative Deepening Depth-First Search merupakan sebuah strategi umum yang biasanya dikombinasikan dengan *Depth-First Search*, yang akan menemukan berapa batas kedalaman (*depth limit*) terbaik untuk digunakan. Hal ini dilakukan secara iterative dengan menambah limit secara bertahap, mulai dari 0, 1, 2, dan seterusnya sampai tujuan sudah ditemukan. Jika dilihat, metode pencarian pada algoritma ini dilakukan pada simpul berdasarkan tingkat kedalamannya sehingga serupa dengan *Breadth-First Search.* Bedanya pada algoritma ini kita selalu mengulang dari simpul akar hingga simpul pada kedalaman tertentu dalam pembangkitan simpulnya sehingga dasar metodenya tetap menggunakan *Depth-First Search*, namun memiliki tingkat kesesuaian yang lebih baik dibanding DFS dalam menemukan solusi.



Gambar 2.1 Perbandingan urutan pembangkitan BFS dan DFS. Sumber: https://vivadifferences.com/difference-between-dfs-and-bfs-in-artificial-intelligence/



Gambar 2.2 Algoritma Pencarian Uniform Cost Search
Sumber: https://www.javatpoint.com/ai-uninformed-search-algorithms

G. *Informed Search*

Informed Search sering disebut juga dengan *Heuristic Search.* Hal ini karena pencarian dengan algoritma ini menggunakan *knowledge* yang spesifik kepada permasalahan yang dihadapi di samping dari definisi masalahnya itu sendiri. *Knowledge* tersebut merupakan data tambahan yang menjadi acuan pembangkitan simpul pada algoritma ini. Metode ini mampu menemukan solusi lebih mangkus dengan waktu yang lebih singkat daripada metode *Uninformed Search*. Pada pencarian dengan menggunakan metode *Uniform Cost Search* (salah satu bagian dari algoritma *Uninformed Search*), kita membandingkan nilai pada simpul yang telah dihidupkan dan kemudian akan melakukan ekspansi pada simpul dengan nilai yang terkecil. Nilai jalur ini biasanya dilambangkan dengan

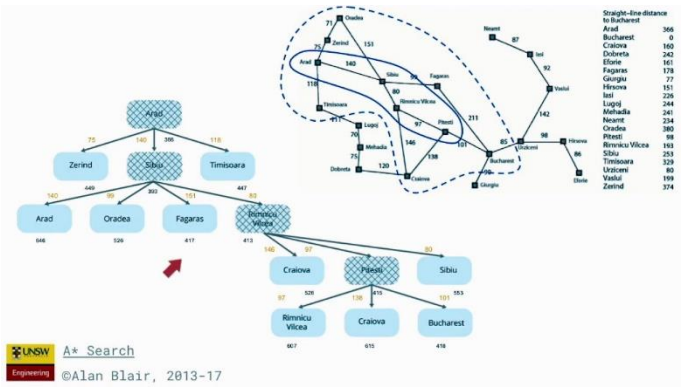
g(n). Lebih lanjut lagi dari metode pencarian tersebut, pada algoritma *Informed Search*, kita akan mengenal nilai estimasi (prediksi) dari tiap simpul (misal n) ke simpul tujuan yang berupa data heuristik. Nilai estimasi ini biasanya dilambangkan dengan *h(n)*. Jika n adalah simpul tujuan, maka nilai *h(n)* adalah nol. Algoritma *Informed Search* yang terkenal ada dua, yaitu *Greedy Best First Search* dan *A* Search*.

H. *Greedy Best First Search*

Algoritma *Greedy Best First Search* melakukan ekspansi kepada simpul yang memiliki jarak terdekat dengan simpul tujuan. Namun, ekspansi yang dilakukan pada metode ini berdasarkan data heuristik berupa taksiran biaya yang dibutuhkan dari calon simpul yang akan dibangkitkan untuk menuju simpul tujuan. Nilai heuristik ini biasanya menggunakan simbol *h(n)* dengan nilai aslinya disimbolkan dengan *h*(n)*. Penggunaan *h(n)* ketimbang *h*(n)* bertujuan untuk memangkas waktu yang dibutuhkan untuk menemukan simpul solusi. Dengan kata lain, yang dibandingkan untuk penentuan ekspansi simpul adalah nilai estimasinya saja. Pada algoritma pencarian ini nilai *h(n)* bisa merupakan nilai yang *non-admissible* jika nilainya lebih besar dari nilai yang sebenarnya *(h(n) > h*(n))* sehingga algoritma ini bisa disebut *non-complete*. Jika nilai tersebut digunakan dalam pencarian, maka dapat dipastikan jalur yang digunakan dalam *Greedy Best First Search* bukan jalur yang paling mangkus, atau bahkan jalur yang salah.

I. *A* Search*

Salah satu bentuk dari *Best First Search* yang paling dikenal adalah algoritma *A* Search* (dibaca dengan "*A-star*"). Sedikit berbeda dengan *Greedy Best First Search* yang hanya melihat kepada nilai taksiran dari suatu simpul menuju simpul tujuannya (*h(n)*), pencarian rute dengan menggunakan *A* Search* melihat kepada jumlah nilai dari jalur simpul awal menuju suatu simpul (misal simpul n), yaitu *g(n)* dengan nilai estimasi simpul tersebut menuju simpul tujuan, yaitu *h(n)*. Algoritma ini memiliki tingkat kemangkusan yang cukup serupa dengan *Greedy Best First Search*, namun memiliki ketepatan yang cukup tinggi karena mempertimbangkan biaya dari simpul awal menuju suatu simpul (misal simpul n) dalam melakukan ekspansi dari tiap simpul yang ada.



Gambar 2.3 Algoritma Pencarian A Search dengan data heuristik*
Sumber: A. Blair, D. Collien, D. Ripley & S. Griffith, 2017. Constructivist Simulations for Path Search Algorithms

III. PEMBAHASAN

A. Pemilihan Algoritma *Uninformed Search*

Algoritma *Uninformed Search* yang akan digunakan pada bab ini dideduksi dari penjelasan mengenai *Uninformed Search* pada bab II. Pada bab II, dapat kita simpulkan bahwa algoritma yang paling mangkus adalah *Depth-First Search* namun memiliki kelemahan yaitu apabila sudah menemukan tujuan pada suatu simpul maka pencarian akan berhenti sehingga apabila rute pertama untuk mencapai tujuan dengan biaya yang tidak optimal, algoritma ini tidak dapat mencapai solusi yang tepat. Sedangkan untuk algoritma *Breadth-First Search* memiliki tingkat keakuratan yang lebih tinggi namun dengan kompleksitas waktu yang lebih tinggi. Depth-First Search *(DFS)* dan Iterative Deepening Depth-First Search *(IDS)* membangkitkan simpul-simpul layaknya *Breadth-First Search* namun dengan metode *Depth-First Search* dimana simpul-simpul dibangkitkan dengan batasan kedalaman secara iteratif yang selalu dimulai dari simpul akar dalam pembangkitan tiap kedalamannya. Kemudian, terdapat *Uniform Cost Search* dengan metode pembangkitan serupa dengan *Breadth-First Search* namun dengan mempertimbangkan nilai yang dimiliki oleh tiap simpul sehingga memiliki kompleksitas waktu yang relative rendah dan tingkat keakuratan yang cukup tinggi. Oleh karena itu, algoritma *Uninformed Search* yang akan digunakan untuk analisis masalah ini adalah *Uniform Cost Search.*

B. Pemilihan Algoritma *Informed Search*

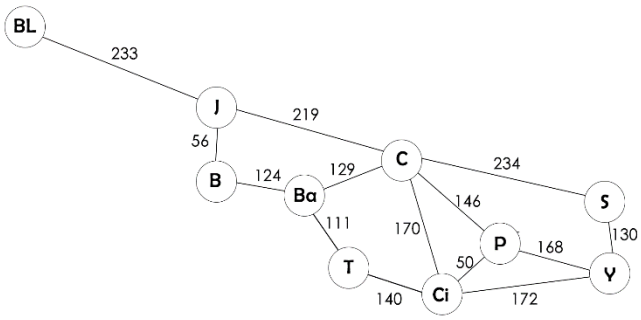
Algoritma *Greedy Best First Search* merupakan algoritma pencarian yang hanya melihat nilai taksiran dari simpul yang akan dibangkitkan ke simpul tujuan dan apabila sudah menemukan tujuan maka algoritma ini akan berhenti, kelemahan algoritma ini adalah bersifat *non-complete* jika terdapat nilai heuristic yang tidak admissible. Algoritma *A* Search* merupakan algoritma dengan tingkat kemangkusan yang tinggi serta bersifat *complete* dengan tingkat keakuratan yang tinggi. Hal ini karena *A* Search* juga mempertimbangkan biaya yang dibutuhkan dari simpul awal menuju suatu simpul yang akan dibangkitkan. Tidak hanya bergantung pada nilai heuristic yang dimiliki oleh tiap simpul yang akan dibangkitkan untuk mencapai simpul tujuan. Oleh karena itu, Algoritma *Greedy Best First Search* akan digunakan dengan pertimbangan kemangkusan waktu bersanding dengan *Uniform Cost Search* yang unggul dalam masalah keakuratan. Selain itu, akan digunakan pula algoritma *A* Search* yang unggul dalam kedua hal di atas.

C. Pemodelan Graf dari Rute Liburan

Untuk melakukan pemodelan, terlebih dahulu akan ditentukan kota asal dan kota tujuan sehingga dapat dicari opsi-opsi rute yang dapat dilalui dari kota asal hingga sampai ke kota tujuan. Setelah menentukan kota asal dan kota tujuan, akan ditentukan kota-kota dan rute-rute jalan yang dapat diambil oleh suatu keluarga dalam mencapai kota tujuan melalui kota asal. Penentuan dilakukan dengan menggunakan aplikasi *Google Maps.* Setelah itu graf akan digambar bersesuaian dengan kota-kota dan rute-rute yang telah diambil pada *Google Maps*. Tujuan penggambaran dengan graf adalah agar jalur dari tiap kota dapat tampak lebih jelas. Pada kasus ini, kota asal yang diambil adalah Kota Bandarlampung sedangkan kota tujuan yang diambil

adalah Kota Yogyakarta. Pada *Gambar 3.1*, titik-titik merah merepresentasikan kota-kota yang dapat dilalui. Pada *Gambar 3.2*, tiap simpul diberi inisial dari nama kota yang dapat dilalui, kemudian nilai yang terdapat pada sisi adalah *cost* jalur antarkota (dalam hal ini merupakan jarak antarkota)

Gambar 3.1. Peta kota-kota yang dapat dilalui dari Bandarlampung menuju Yogyakarta.
Sumber: https://www.google.com/maps/@-6.5966909,105.6907382,7z



Gambar 3.2. Graf rute perjalanan dari Bandarlampung ke Yogyakarta

D. Pengambilan Data Heuristik

Untuk keperluan pencarian rute menggunakan *Informed Search* dilakukan pengambilan data heuristik berupa jarak garis lurus dari tiap kota menuju kota Yogyakarta. Data yang diperoleh seluruhnya *admissible* sehingga dapat digunakan dalam *Greedy Best First Search* dan *A* Search*. Data yang diperoleh dapat dilihat pada *Tabel 3.1*.

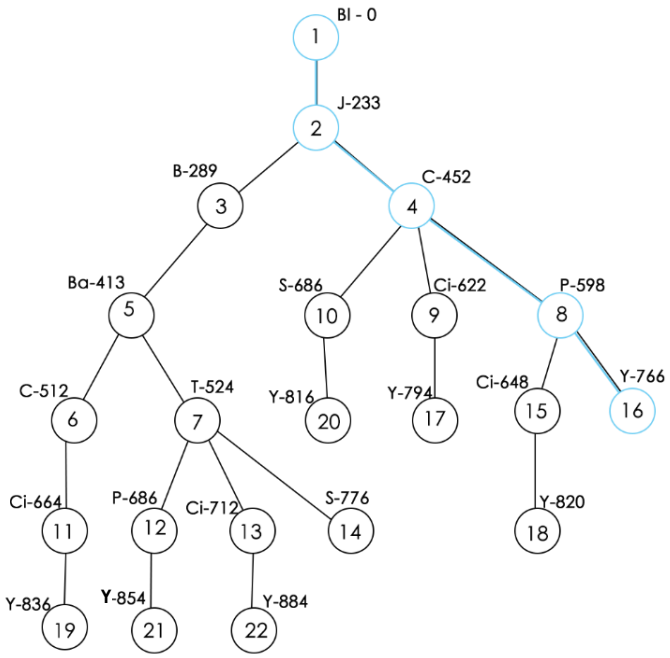
Kota	Cost	Kota	Cost
Bandarlampung	629	Tasikmalaya	248
Jakarta	446	Purwokerto	132
Bogor	421	Cilacap	156
Bandung	320	Semarang	108
Cirebon	246	Yogyakarta	0

Tabel 3.1 Data heuristik jarak dari tiap kota ke Yogyakarta.

IV. ANALISIS

A. Pencarian Rute dengan *Uniform Cost Search*

Pada *Gambar 4.1*, pencarian dimulai dengan membangkitkan simpul satu yang merupakan kota Bandarlampung. Angka pada simpul menunjukkan urutan simpul dihidupkan. Selanjutnya akan dihidupkan simpul dua yang merupakan Kota Jakarta, kemudian simpul dua akan langsung dibangkitkan tanpa melakukan perbandingan nilai dengan yang lain karena simpul hidup yang ada hanyalah simpul dua dan mengakibatkan hidupnya simpul tiga (Kota Bogor) dan simpul empat (Kota Cirebon). Selanjutnya, simpul lima akan diekspansi dan menghidupkan simpul enam (Tasikmalaya) dan simpul tujuh (Cirebon). Selanjutnya, simpul yang diekspansi adalah simpul empat karena memiliki nilai yang paling rendah di antara simpul-simpul hidup lainnya. Ekspansi simpul empat menghidupkan simpul delapan (Purwokerto), simpul sembilan (Cilacap), dan simpul sepuluh (Semarang), Selanjutnya simpul yang akan diekspansi secara berturut-turut adalah simpul enam dan simpul tujuh yang menghasilkan simpul sebelas (Cilacap) serta simpul 12 (Purwokerto), simpul 13 (Cilacap), dan simpul 14 (Semarang). Selanjutnya, simpul yang akan diekspansi adalah simpul delapan yang menghidupkan simpul 14 (Cilacap) dan simpul 15 (Yogyakarta). Di sini, simpul yang mengarahkan ke kota tujuan sudah berhasil dihidupkan dan simpul inilah yang berisikan jalur dengan biaya minimal yang dapat ditempuh oleh sebuah keluarga. Namun, pencarian tidak boleh diberhentikan sebelum simpul 15 dibangkitkan. Oleh karena itu, simpul-simpul harus tetap dibangkitkan dengan cara yang sama, yaitu dengan melihat nilai terkecil dari daftar simpul yang hidup. Pada *Gambar 4.1*, terdapat beberapa rute yang berhasil ditemukan untuk mencapai Kota Yogyakarta dari Kota Bandarlampung, namun dengan biaya yang pastinya lebih besar dibandingkan simpul 15 sebelumnya.



Gambar 4.1 Pohon Uniform Cost Search untuk mencari rute dari Kota Bandarlampung menuju Kota Yogyakarta.

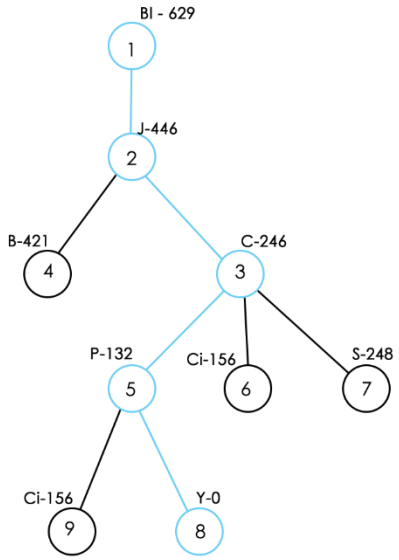
Pada *Gambar 4.1*, dapat dilihat bahwa terdapat banyak rute lain yang dapat jadi pertimbangan oleh seorang keluarga dalam perjalanan liburannya, yaitu *Bl – J – C – Ci – Y* dengan biaya 794, *Bl – J – C – S – Y* dengan biaya 816, *Bl – J – B- Ba - C – Ci – Y* dengan biaya 836, *Bl - J – B – Ba – T – P – Y* dengan biaya 854, dan *Bl – J – B – Ba – T – Ci – Y* dengan biaya 884. Untuk solusi optimalnya dapat dilihat dari sisi simpul dan *border* simpul yang bewarna biru, yaitu *Bl – J – C – P – Y* dengan biaya 766.

Waktu yang dibutuhkan oleh *Uniform Cost Search* dalam menemukan solusi untuk masalah ini adalah sangat lama. Penghitungan kompleksitas waktu dilakukan secara manual dengan menghitung jumlah dari jumlah simpul yang hidup pada tiap pembangkitan simpul, kecuali pada pembangkitan simpul optimal. Jumlah simpul yang dibangkitkan hingga mendapatkan solusi optimal adalah 15 simpul sehingga dari *queue* yang berisikan simpul-simpul hidup yang telah didaftarkan terdapat 1 + 2 + 2 + 3 + 5 + 5 + 7 + 7 + 7 + 7 + 7 + 7 + 7 + 7 = 74. Oleh karena itu, waktu yang dibutuhkan untuk menemukan solusi adalah 74 satuan waktu.

B. Pencarian Rute dengan *Greedy Best First Search*

Pencarian dengan menggunakan *Greedy Best First Search* dimulai dari kota yang sama seperti sebelumnya, yaitu Kota Bandarlampung yang direpresentasikan dengan simpul satu. Berdasarkan *Gambar 4.2*, pencarian diawali dengan membangkitkan simpul satu yang mengakitbatkan hidupnys simpul dua (Jakarta). Kemudian, simpul dua yang merupakan simpul hidup tunggal akan langsung dibangkitkan dan menghidupkan simpul tiga (Bogor) dan simpul empat (Cirebon). Berdasarkan nilai heuristik yang telah diperoleh, Cirebon memiliki jarak garis lurus yang lebih kecil dibandingkan Bogor sehingga simpul empatlah yang akan dibangkitkan selanjutnya. Pembangkitan simpul empat menghidupkan simpul lima (Purwokerto), simpul enam (Cilacap), dan simpul tujuh (Semarang). Dari semua simpul yang hidup, simpul lima memiliki nilai heuristik yang lebih kecil dibandingkan simpul empat, enam, dan tujuh sehingga simpul lima akan dibangkitkan dan menghidupkan simpul delapan (Yogyakarya) dan simpul sembilan (Cilacap). Simpul delapan memiliki nilai heuristik yang paling rendah di antara simpul-simpul hidup lainnya sehingga simpul delapan akan dibangkitkan. Pembangkitan simpul delapan berhasil menemukan solusi hingga mencapai kota tujuan, yaitu Kota Yogyakarta.

Pada pencarian dengan algoritma *Greedy Best First Search*, hanya satu solusi yang ditemukan, dan solusi tersebut merupakan solusi yang paling optimal, yaitu *Bl – J – C – P – Y* dengan biaya 766. Waktu yang dibutuhkan oleh algoritma ini relatif lebih singkat karena hanya membangkitkan lima simpul. Hal ini karena nilai heuristik cenderung menurun setiap bertambahnya langkah. Jumlah simpul hidup yang dibandingkan pada tiap langkahnya juga jauh lebih sedikit dibandingkan pada *Uniform Cost Search*. Kompleksitas waktu dari algoritma ini dihitung secara manual dengan menghitung tiap simpul hidup yang ada pada tiap langkahnya. Jumlah simpul hidup dapat dihitung dari *queue*, dalam hal ini terdapat 1 + 2 + 4 + 5 = 12 simpul hidup. Oleh karena itu pencarian dengan menggunakan *Greedy Best First Search* membutuhkan waktu sebesar 10 satuan waktu.

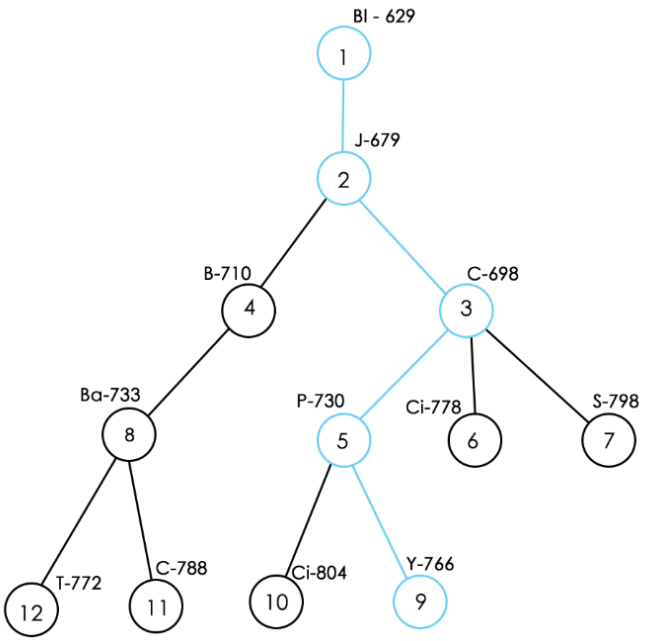


Gambar 4.2 Pohon Greedy Best First Search untuk mencari rute dari Kota Bandarlampung menuju Kota Yogyakarta.

C. Pencarian Rute dengan *A* Search*

Pencarian dengan menggunakan algoritma *A* Search* mempertimbangkan nilai sebenarnya dari dari kota awal menuju kota n dan nilai taksiran dari kota n menuju kota tujuan. Oleh karena itu, akan digunakan nilai dari graf dan nilai data heuristik. Pencarian dimulai dengan membangkitkan simpul satu (Bandarlampung) yang mengakibatkan hidupnya simpul dua (Jakarta). Selanjutnya simpul dua dibangkitkan, menghidupkan simpul tiga (Cirebon) dan empat (Bogor), akibat simpul tiga punya biaya yang lebih rendah, maka simpul tiga dibangkitkan, menghidupkan simpul lima (Purwokerto), simpul enam (Cilacap), dan simpul tujuh (Semarang). Melihat simpul empat memiliki biaya yang paling rendah sehingga simpul empat dibangkitkan dan menghidupkan simpul delapan (Bandung). Selanjutnya, biaya terendah dimiliki oleh simpul lima sehingga simpul lima dibangkitkan dan menghidupkan simpul sembilan (Yogyakarta) dan simpul sepuluh (Cilacap). Pada kondisi ini, simpul solusi sudah ditemukan, yaitu simpul sembilan, namun simpul tersebut belum dibangkitkan sehingga pencarian belum bisa diberhentikan. Pencarian dilanjutkan dengan membangkitkan simpul delapan yang menghidupkan simpul sebelas (Cirebon) dan simpul dua belas (Tasimalaya). Dari seluruh simpul yang hidup, simpul sembilan memiliki biaya terendah sehingga simpul sembilan dibangkitkan dan pencarian dihentikan.

Pada *Gambar 4.2*, pencarian rute dengan algoritma ini membangkitkan tujuh simpul hingga dapat menemukan solusinya. Pencarian berakhir dengan satu solusi optimal yang ditemukan, yaitu *Bl – J – C – P – Y* dengan biaya 766. Kompleksitas waktu dari algoritma ini dihitung secara manual dengan menghitung tiap simpul hidup yang ada pada tiap langkahnya. Jumlah simpul hidup dapat dihitung dari *queue*, dalam hal ini terdapat 1 + 2 + 4 + 4 + 5 + 6 = 22 simpul hidup. Oleh karena itu pencarian dengan menggunakan *A* Search* membutuhkan waktu sebesar 22 satuan waktu.



Gambar 4.2 Pohon Greedy Best First Search untuk mencari rute dari Kota Bandarlampung menuju Kota Yogyakarta.

V. KESIMPULAN

Pada masalah ini, algoritma *Greedy Best First Search* memberikan solusi yang optimal dengan waktu yang paling singkat dibandingkan *Uniform Cost Search* dan *A* Search.* Algoritma *Greedy Best First Search* hanya memerlukan waktu sebesar $^1/_6$ dari *Uniform Cost Search* dan $^1/_2$ dari *A* Search.* Namun pencarian menggunakan *Uniform Cost Search* memberikan beberapa opsi rute yang dapat dilalui dalam perjalanan liburan. Hal ini menguntungkan apabila terdapat faktor X lain yang mempengaruhi perjalanan sehingga sebuah keluarga dapat membuat perencanaan lebih dari satu. Namun, jika faktor X diabaikan, *Greedy Best First Search* sudah cukup baik untuk memberikan solusi optimal dengan waktu yang sangat singkat.

VI. UCAPAN TERIMA KASIH

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena atas rahmat-Nya penulis dapat menyelesaikan makalah ini dengan baik. Penulis juga mengucapkan terima kasih kepada Ibu Masayu Leylia Khodra dan Bapak Rinaldi Munir, selaku pengajar mata kuliah Strategi Algoritma yang telah memberi bimbingan dalam perkuliahan sehingga saya dapat menyelesaikan makalah ini. Penulis juga mengucapkan terimakasih kepada keluarga dan teman-teman tercinta yang tidak henti-hentinya memberikan dukungan moral serta bantuan dalam proses perkuliahan Strategi Algoritma.

VI. REFERENCES

[1] https://vivadifferences.com/difference-between-dfs-and-bfs-in-artificial-intelligence/ Diakses pada 2 Mei 2020, pukul 02.32
[2] https://www.geeksforgeeks.org/search-algorithms-in-ai/Diakses pada 2 Mei 2020, pukul 02.46
[3] https://www.javatpoint.com/ai-uninformed-search-algorithms Diakses pada 2 Mei 2019, pukul 03.21
[4] A. Blair, D. Collien, D. Ripley & S. Griffith, 2017. Constructivist Simulations for Path Search Algorithms. Dibaca pada 2 Mei 2020, pukul 14.07
[5] http://www.cs.cmu.edu/afs/cs/academic/class/15780-s13/www/lec/07-Informed-search.pdf Diakses pada 2 Mei 2020, pukul 17.21
[6] https://www.maps.google.com Diakses pada 2 Mei 2020, pukul 21.03

VII. TAUTAN VIDIO

Untuk lebih memahami makalah ini, penulis memberikan penjelasan melalui video yang dapat diakses melalui tautan berikut: https://youtu.be/p0gs6C_x_7g

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.

Bandarlampung, 3 Mei 2020

Fadhil Muhammad Rafi' 13518079