PENERAPAN *ALGORITMA BIDIRECTIONAL* A* PADA *MOBILE NAVIGATION SYSTEM*

Indra Siregar - 13508605

Program Studi Teknik Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10, Bandung 40132
e-mail: if18605@students.if.itb.ac.id

ABSTRAK

***Mobile Navigation System* (MNS) adalah suatu sistem navigasi dengan menggunakan perangkat bergerak. MNS banyak digunakan dalam pencarian rute terpendek ke suatu lokasi atau jalan tertentu. MNS sangat penting pada daerah yang belum sepenuhnya diketahui lokasi jalan dan tempat-tempat penting di daerah tersebut, atau pada daerah perkotaan dengan arus lalu lintas yang padat. Untuk mendapatkan rute terbaik berdasarkan biaya tertentu, misalnya jarak terdekat, bahan bakar paling sedikit, arus lalu lintas paling sedikit, pada MNS digunakan algoritma *Bidirectional* A*. Algoritma ini adalah modifikasi dari A*, dengan pencarian dilakukan pada dua arah, yaitu simpul asal dan simpul tujuan. Algoritma *Bidirectional* A* memiliki kelebihan dalam hal memori dan waktu pencarian dibandingkan dengan A*. Makalah ini membahas penerapa algoritma *Bidirectional* A* pada MNS.**

Kata kunci: *Mobile Navigation System*, A*, *Bidirectional* A*, *Global Positioning System*, pencarian rute terpendek.

1. PENDAHULUAN

Mobile Navigation System (MNS) atau sistem navigasi berbasis perangkat bergerak (*smartphone* maupun *Personal Digital Assistant*) dan *Global Positioning System* (GPS) sudah lama digunakan di negara-negara maju di Eropa dan Amerika. MNS banyak digunakan oleh orang-orang yang belum mengenal seluk beluk suatu kota sehingga belum tahu nama-nama jalan di kota tersebut. Selain itu MNS juga banyak digunakan di kota-kota besar dengan tingkat kemacetan tinggi. MNS digunakan untuk mencari rute lalu lintas terbaik yaitu yang memiliki tingkat kemacetan relatif rendah dibanding dengan rute yang lainnya.

Dari penjelasan ringkas di atas mengenai MNS, dapat dilihat bahwa MNS bisa memberikan rute perjalanan dari suatu posisi (misalnya A) ke suatu posisi yang lainnya (misalnya B) dengan biaya tempuh paling murah. Biaya tempuh yang dimaksud dapat berupa jarak tempuh, waktu tempuh, atau penggunaan bahan bakar. Saat ini sudah banyak beredar produk yang menawarkan sistem navigasi yang portable, dapat dibawa dengan tangan atau dipasang pada mobil atau motor.

Disamping metode untuk pencarian rute, beberapa faktor lain yang penting dalam pembangunan MNS. Kombinasi yang tepat dari faktor-faktor tersebut akan menghasilkan MNS yang prosesnya sangat cepat dan hasilnya akurat. Faktor-faktor tersebut adalah sebagai berikut:

1. Kesalahan GPS dalam menentukan posisi objek. Kebanyakan produsen MNS mengklaim bahwa tingkat akurasi GPS produknya berada dalam hitungan sentimeter (artinya kesalahan GPS tidak lebih dari 1 meter).
2. Strategi penyimpanan peta pada basisdata.
3. Sistem operasi yang digunakan.
4. Semua perangkat lunak yang digunakan untuk membangun MNS.

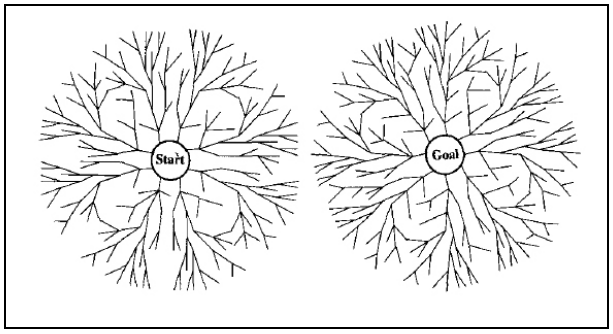
Pada makalah ini hanya akan dibahas metode pencarian rute yang digunakan pada MNS dengan memanfaatkan algoritma *Bidirectional* A*. Metode ini dipilih karena *Bidirectional* A* menggunakan memori yang lebih sedikit (seiring dengan jumlah simpul yang semakin banyak) dibandingkan dengan A* dengan hasil peformansi yang lebih baik.

1.1 *Bidirectional* A*

Searching adalah teknik yang paling mudah digunakan untuk membangun MNS. Terdapat banyak metode dalam teknik searching yang telah dipublikasikan oleh para peneliti. Setiap metode memiliki kelebihan dan kekurangannya masing-masing.

Metode searching yang paling populer untuk pencarian jalur terpendek adalah algoritma **Dijkstra**. Algoritma ini mudak diimplementasikan, tapi kinerjanya kurang bagus untuk graf yang besar dengan puluhan ribu simpul. Algoritma lain yang juga sangat populer adalah algoritma A* (yang dibahas pada makalah ini).

Algoritma A* adalah suatu algoritma *Best First Search* yang menggabungkan *Uniform Cost Search* dan *Greedy Best-Fit Search*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dengan perhitungan biaya seperti ini, algoritma A* adalah *complete dan optimal*. Modifikasi dari algoritma A* adalah *Bidirectional* A*. Berbeda dengan A* dimana pencarian dilakukan pada satu arah, pada *Bidirectional* A* pencarian dilakukan pada dua arah, yaitu dari simpul asal dan simpul tujuan.



Gambar 1 *Bidirectional Search*

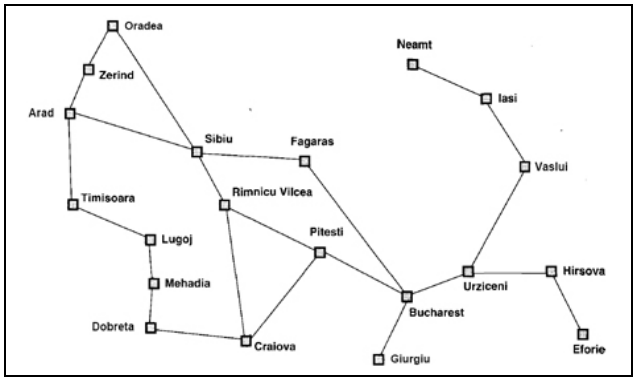
2. METODE

Untuk membangun MNS yang handal diperlukan banyak pertimbangan yang matang. Pertimbangan pertama adalah representasi peta jalan dan tempat-tempat penting dalam format yang mudah diakses dan ter-update. Pertimbangan kedua adalah menentukan teknik dan metode yang paling sesuai untuk menghasilkan rute yang optimal dalam waktu yang sangat singkat, berdasarkan representasi peta yang sudah dibangun.

2.1 Representasi peta

Peta dapat direpresentasikan (dalam hal ini adaah peta jalan dan tempat-tempat penting) ke dalam sebuah basisdata. Misalnya setiap persimpangan jalan dan tempat-tempat penting adalah simpul. Masing-masing simpul terhubung dengan simpul-simpul lainnya melalui sebuah busur yang memiliki biaya tempuh (jarak, waktu, atau bahan bakar). Pada saat pencarian rute, dapat dibayangkan setiap simpul memiliki simpul sebelumnya (*parent*) dan simpul sesudahnya (*successor*).

Gambar di bawah ini adalah sebuah peta sederhana. Terdapat banyak persimpangan jalan yang berupa pertigaan, perempatan, dan persimpangan lainnya. Terdapat pula lampu lalu lintas (*traffic light*) pada beberapa persimpangan jalan tersebut dan terdapat tanda arah, yang menyatakan bahwa jalan tersebut satu arah. Pada peta tersebut terlihat banyak tempat-tempat penting seperti hotel, sekolah, rumah sakit dan bandara.



Gambar 2 Peta Sederhana

Langkah selanjutnya dalam pembangunan MNS adalah merepresentasikan peta menjadi suatu basisdata. Setiap tempat-tempat penting dan jalan pada gambar peta tersebut merupakan record penyusun sebuah tabel. Representasi basisdata untuk peta tersebut dapat dilihat pada tabel di bawah ini.

No	Simpul	X	Y	S1	S2	S3	…	…
1	A	3	15	B	C			
2	B	53	15	A	D	E		
3	C	3	65	A	F			
4	D	53	55	B	C	G		
…								
15	Hotel A	8	15	A	B			
…								

Gambar 3 Representasi Basisdata

2.2 Pencarian Rute dengan *Bidirectional* A*

Secara matematis, perhitungan biaya yang digunakan pada algoritma A* dapat dituliskan sebagai berikut:

$$f(n) = g(n) + h(n)$$

Dimana $f(n)$ merupakan total biaya, $g(n)$ adalah biaya sebenarnya dan $h(n)$ adalah biaya perkiraaan. Untuk masalah pencarian yang tidak begitu kompleks, A* sudah menghasilkan pencarian optimal. Namun untuk masalah

yang lebih kompleks (graf dengan 100 juta simpul), A* akan menghadapi masalah waktu dan memori yang dibutuhkan. Oleh karena itu telah dibuat beberapa modifikasi algoritma A*, salah satunya adalah algoritma *Bidirectional* A*.

Algoritma Bidirectional A* atau A* dari dua arah adalah modifikasi dari algoritma A* dengan arah pencarian dari dua arah, yaitu simpul asal dan simpul tujuan.

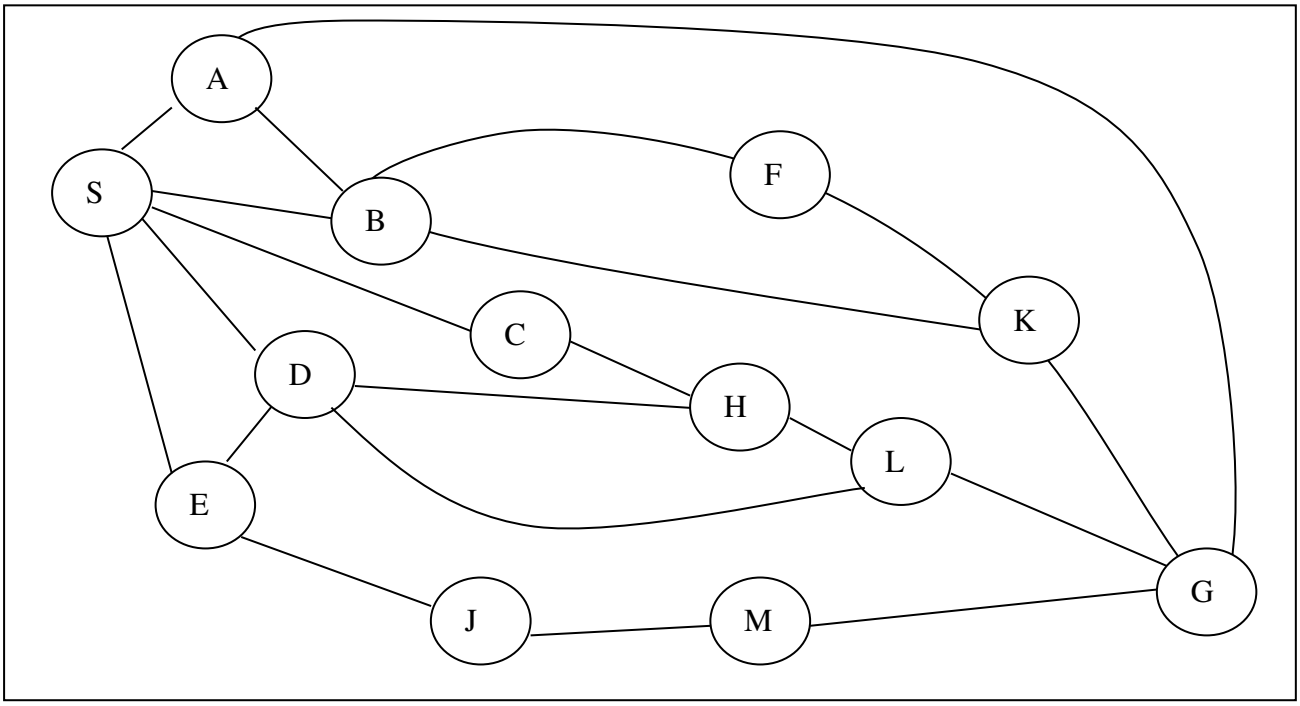
Pada algoritma *Bidirectional* A*, pencarian dihentikan jika *BestNode* dari arah simpul asal telah berada dalam senarai *CLOSED* dari arah simpul tujuan. Kemudian dilakukan pengecekan apakah harus mengganti *parent* dari *BestNode* tersebut dari arah simpul tujuan. Atau sebaliknya pencarian dihentikan jika *BestNode* dari arah simpul tujuan telah berada dalam senarai *CLOSED* dari arah simpul asal. Juga akan dilakukan pemeriksaan apakah akan mengganti *parent* dari *BestNode* tersebut dari arah simpul asal.

Bidirectional A* menggunakan fungsi heuristik yang sama dengan A*. Secara lengkap algoritma *Bidirectional* A* diilustrasikan dalam algoritma berikut ini:

```
function BDA*(masalah) return solusi
    inputs: masalah: berisi ruang masalah, jenis sousi, S dan G
    local variables: BNs dan BNg: BestNode dari arah start dan goal
      OPENs dan OPENg: OPEN dari arah start dan goal
      CLOSEDs dan CLOSEDg: CLOSED dari arah start dan goal
    loop sampai goal ditemukan
      BNs, OPENs, CLOSEDs <- A*(OPENs, CLOSEDs, S, G)
      if BNs in CLOSEDg then {goal sudah ditemukan}
            V <- simpul di CLOSEDg yang sama dengan BNs
            g-terbaik <- g(G, v) {biaya sebenarnya dari G ke v melalui parent lama}
            loop untuk semua suk = simpul di CLOSEDg yang merupakan suksesor v
                  if g(G, v) melalui suk < g-terbaik then
                        g-terbaik <- g(G, v) melalui suk
                        parent-baru <- suk
                  end
            end
            g(v) <- g-terbaik
            parent dari v <- parent-baru
            return solusi
      end
      BNg, OPENg, CLOSEDg <- A*(OPENg, CLOSEDg, G, S)
      if BNg in CLOSEDs then {solusi sudah ditemukan}
            U <- simpul di CLOSEDs yang sama dengan BNg
            g-terbaik <- g(S, u) {biaya sebenarnya dari S ke u melalui parent lama}
            loop untuk semua suk = simpul di CLOSEDg yang merupakan suksesor u
                  if g(S, u) melalui suk < g-terbaik then
                        g-terbaik <- g(S, u) melalui suk
                        parent-baru <- suk
                  end
            end
            g(S, u) <- g-terbaik
            parent dari u <- parent-baru
            return solusi
    end
end
```

Sama dengan algoritma A*, algoritma *Bidirectional* A* adalah *complete* dan optimal. Dibandingkan dengan A* untuk ruang masalah yang besar, *Bidirectional* A* bisa lebih cepat dalam waktu proses dan hemat dalam pemakaian memori karena jumlah simpul yang dibandingkan lebih sedikit. Jumlah simpul yang dibandingkan oleh *Bidirectional* A* adalah setengah dari jumlah simpul yang dibandingkan oleh A*.

Pada gambar di bawah ini adalah representasi suatu permasalah pencarian rute dalam bentuk graf dan tabular. Simpul adalah himpunan dari *n*, $h_s(n)$ menyatakan biaya perkiraan (jarak garis lurus) dari suatu simpul di n menuju simpul tujuan (dalam hal ini adalah simpul *G*). Sedangkan $h_g(n)$ menyatakan biaya perkiraan dari simpul di *n* menuju simpul *S*.



Gambar 4 Graf Pencarian

n	*S*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*J*	*K*	*L*	*M*
$h_s(n)$	80	80	60	70	85	74	70	0	40	100	30	20	70

n	*S*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*J*	*K*	*L*	*M*
$h_g(n)$	0	10	15	25	30	5	20	90	45	25	50	70	60

Gambar 5 Representasi dalam Bentuk Tabular

Langkah-langkah pencarian solusi adalah sebagai berikut:

1. Pencarian maju menghasilkan BNs = *S*, OPENs = [*A*, *B*, *C*, *D, E*], dan *CLOSEDs* = [*S*]. Karena *BNs* tidak berada pada *CLOSEDg*, maka dilanjutkan pada pencarian mundur yang menghasilkan *BNg* = *G*, *OPENg* = [*A*, *K*, *L*, *M*], dan *CLOSEDg* = [*G*]. Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk iterasi berikutnya.
2. Pencarian maju menghasilkan *BNs* = *E*, *OPENs* = [*A*, *B*, *C*, *D*, *J*], dan *CLOSEDs* = [S, *E*]. Karena *BNs* tidak berada pada *CLOSEDg*, maka dilanjutkan pada pencarian mundur yang menghasikan *BNg* = *K*, *OPENg* = [*A*, *L*, *M*, *F*, *B*], dan *CLOSEDg* = [*G*, *K*]. Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk iterasi berikutnya.
3. Pencarian maju menghasilkan *BNs* = *B*, *OPENs* = [*A*, *C*, *D*, *F*, *J*, *K*] dan *CLOSEDs* = [*S*, *E*, *B*]. Karena *BNs* tidak berada pada *CLOSEDg*, maka dilanjutkan pada pencarian mundur yang menghasilkan *BNg* = *F*, *OPENg* = [*A*, *L*, *M*, *B*], dan *CLOSEDg* = [*G*, *K*, *F*]. Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk iterasi berikutnya.
4. Pencarian maju menghasilkan *BNs* = *A*, *OPENs* = [*C*, *D*, *F*, *J, K*, *G*] dan *CLOSEDs* = [*S*, *E, B*, *A*]. Karena *BNs* tidak berada pada posisi *CLOSEDg*, maka dilanjutkan pada pencarian mundur yang menghasilkan *BNg* = *B*, *OPENg* = [*A*, *L*, *M*], dan *CLOSEDg* = [*G*, *K*, *F*, *B*]. Karena *BNg* = *B* berada pada *CLOSEDs*, berarti solusi telah ditemukan. Kemudian dicari suksesor dari *B* yang sudah ada di dalam *CLOSEDs*. Simpul *A* adalah suksesor *B*, dan berada dalam *CLOSEDs*. Ternyata *g(S, B)* melalui *A* lebih kecil dari *g(S, B)*, maka parent dari *B* diubah, yang sebelumnya *G* menjadi *A*. Nilai *g* dan *f* pada *B* juga diubah. Nilai *g* yang sebelumnya adalah 25 menjadi 20 dan nilai *f* yang sebelumnya adalah 85 menjadi 80. Selanjutnya rute dan biaya total dapat ditelusuri balik dari *G* menuju *S* dan dari *S* menuju *G*. Hasil penelusuran balik dari kedua arah tersebut digabungkan sehingga menjadi solusi yang utuh. Hasil penelusuran balik menghasilkan rute *S-A-B-F-*

K-G dengan total jarak adalah 95 satuan. Rute ini merupakan rute terpendek pada graf.

Algoritma Bidirectional A* menghasilkan hasil yang optimal. Tanpa ada batasan waktu dan memori, *Bidirectional* A* adalah *complete* (selalu menemukan solusi, jika solusinya ada). Pada kasus di atas, *Bidirectional* A* menyimpan dan membangkitkan 15 simpul (dari 13 simpul yang ada pada graf). Hal ini disebabkan oleh adanya simpul yang disimpan pada dua arah pencarian. Berbeda dengan A* yang hanya membangkitkan dan menyimpan 10 simpul. Tetapi untuk masalah yang lebih kompleks, misalnya dengan jumlah simpul ribuan, performansi *Bidirectional* A* akan lebih baik dibandingkan dengan A*.

3. KESIMPULAN

Kesimpulan yang diperoleh, terkait dengan penerapan algoritma Bidirectional A* pada MNS, adalah sebagai berikut:

1. Pengembangan MNS harus mempertimbangkan kapasitas memori dan kecepatan pencarian rute. Oleh karena itu dibutuhkan algoritma yang memberikan hasil optimal, namun menggunakan memori seminimal mungkin.
2. *Bidirectional* A* menghasilkan hasil yang optimal dan *complete*. Untuk jumlah simpul yang semakin banyak, Bidirectional A* memiliki performansi yang lebih baik dari A*.

REFERENSI

[1] J. Russel, Stuart and Norvig, Peter, *Artificial Intelligence, A Modern Approach*, Prentice-Hall, 1995.

[2] Suyanto, *Artificial Intelligence, Searching, Reasoning, Planning and Learning*, Informatika, 2007.