Representasi Graf Berarah dalam Mencari Solusi Jalur Optimum Menggunakan Algoritma A*

Denny Nugrahadi
Teknik informatika ITB, Bandung, email: d_nugrahadi@yahoo.com

Abstract – *Makalah ini membahas mengenai graf, algoritma A*, persoalan shortest path, bagaimana graf digunakan dalam merepresentasikan persoalan shortest path, bagaimana graf dapat menunjukkan tahap-tahap dalam proses pengerjaan algoritma A* untuk menampilkan cara kerja algoritma tersebut*

Kata Kunci*: graf, A*, simpul, sisi, cost, goal*

1. PENDAHULUAN

Di dunia ini terdapat permasalahan yang kompleks dan saling berkaitan satu sama lain. Untuk mempermudah analisis terhadap permasalahan tersebut serta mencari solusi penyelesaiannya, maka dikembangkan berbagai macam pemodelan untuk menyederhanakan masalah yang dihadapi, sehingga pemecahan terhadap persoalan itu dapat dilakukan dari dasar-dasarnya dulu, sebelum kemudian menambahkan faktor-faktor yang meruwetkan berdasarkan kenyataan di dunia nyata. Salah satunya adalah dengan menggunakan graf dalam memodelkan permasalahan yang bisa dijabarkan dengan sisi-sisi dan simpul-simpul, seperti persoalan mencari Jalur Terpendek, yang akan dijabarkan dalam makalah ini. Untuk algoritma pencariannya, yang akan digunakan adalah algoritma A* yang menggunakan fungsi heuristic dalam menentukan simpul yang dipilih. Ini untuk menunjukkan bagaimana representasi graf dengan atribut jarak pada tiap sisi-sisi yang ada.

2. ISI

2.1. Graf dan Persoalan Pencarian Jalan

Suatu graf dapat didefinisikan sebagai pasangan dari 2 himpunan yaitu himpunan tidak kosong dari simpul-simpul dan himpunan sisi-sisi yang menghubungkan simpul-simpul tersebut. Secara geometri, graf digambarkan sebagai sekumpulan simpul yang dihubungkan dengan sekumpulan garis-garis yang merepresentasikan sisi-sisinya.

Graf memiliki banyak fungsi, kebanyakan berupa representasi dari suatu skema persoalan yang kompleks menjadi lebih sederhana, seperti dalam merepresentasikan rangkaian listrik, isomer senyawa kimia karbon, dan berbagai hal lainnya. Dalam makalah ini, graf akan digunakan untuk merepresentasikan kota-kota dan jalur-jalurnya dalam persoalan Pencarian Jalur Terpendek (*shortest-path*).

2.2. Algoritma A*

Algoritma A* adalah suatu algoritma pencarian untuk *graf*. Ciri dari algoritma ini adalah bahwa ia menggunakan suatu fungsi heuristik(biasa disimbolkan dengan *f(x)*) yang mempertimbangkan jarak dan biaya untuk menentukan urutan di mana Pencari mengunjungi simpul di dalam *graf*. Fungsi heuristiknya merupakan penjumlahan dari dua fungsi yaitu fungsi penghitung biaya suatu jalur(biasa disimbolkan dengan *g(x)*) yang menghitung biaya dari simpul awal ke simpul sekarang serta fungsi perkiraan heuristik yang memperkirakan jarak dari simpul saat ini ke simpul tujuan akhir(fungsi ini umumnya dilambangkan dengan *h(x)*).

Pada A*, pencarian dilakukan secara inkremen pada semua rute yang mengarah ke simpul mulai sampai berhasil ditemukan jalur ke tujuan yang terdekat. Untuk itu, algoritma ini pertama memulainya dengan memeriksa rute yang “kelihatannya” paling mungkin mengarah ke tujuan. Algoritma ini memeriksa jarak yang sudah dilalui dari awal, dan bukan dari jarak dari simpul sebelum sekarang ke simpul yang sekarang.

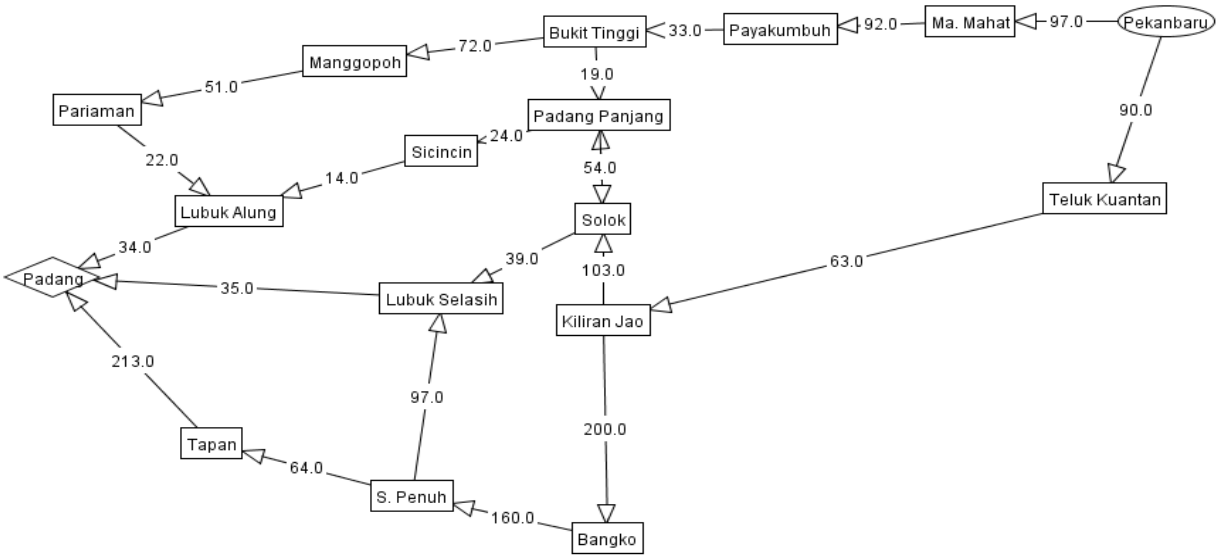
Dari simpul awal, algoritma ini akan meng-*expand* ke simpul dengan nilai *f(x)* terkecil, yaitu simpul yang memiliki nilai biaya per keuntungan terbesar. A*

menyimpan sekumpulan solusi parsial, berupa simpul-simpul yang belum di-*expand* yang masing-masing adalah simpul dari hasil ekspansi yang sudah dilakukan. Simpul-simpul ini disimpan di dalam antrian prioritas(*priority queue*). Prioritas yang diberikan ke jalur x ditentukan dari fungsi *f(x) = g(x) + h(x)*. Fungsi ini akan diteruskan sampai ada tujuan yang memiliki nilai f(x) yang lebih rendah dari simpul manapun dalam antrian, atau bila semua simpul sudah dilewati. Semakin rendah nilai *f(x)* maka semakin tinggilah prioritasnya.

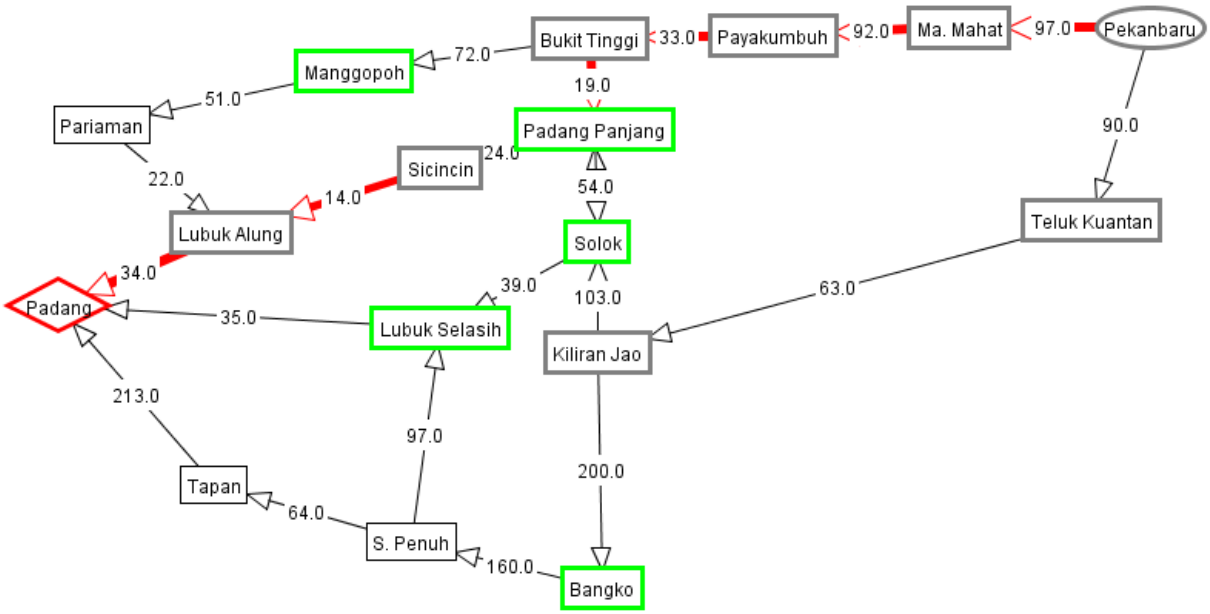
Algoritma ini bisa dijabarkan dalam *pseudo-code* seperti di bawah ini:

```
function A*(start,goal)
     var closed := ''the empty
set''
     var q :=
make_queue(path(start))
     while q ''is not empty''
          var p := remove_first(q)
          var x := ''the last Simpul
of p''
          if x in closed
               continue
          if x = goal
               return p
          ''add x to closed''
          For (each y in
successors(x))
               enqueue(q, p, y)
     return failure
```

Seperti Algoritma Pencarian Melebar(*Breadth-first Search*), algoritma A* akan menghasilkan solusi selama solusi itu memang benar-benar ada.



Gambar 1 Representasi Status Awal



Gambar 2 Representasi Status Akhir

Bila fungsi heuristic h bersifat *admissible*, yang berarti bahwa fungsi itu tidak pernah melakukan overestimasi terhadap nilai minimal dalam mencapai tujuan, maka A* pun bersifat *admissible*, atau bisa dianggap optimal.

Kompleksitas waktu dari A* tergantung dari fungsi heuristiknya. Pada kasus terburuk, jumlah simpul yang diekspansi adalah eksponensial dari panjang solusi(dalam kasus ini jalur terpendeknya), tapi umumnya akan bernilai polinomial selama fungsi heuristic h memenuhi kondisi sebagai berikut:

$$|h(x) - h^*(x)| \leq O(\log h^*(x))$$

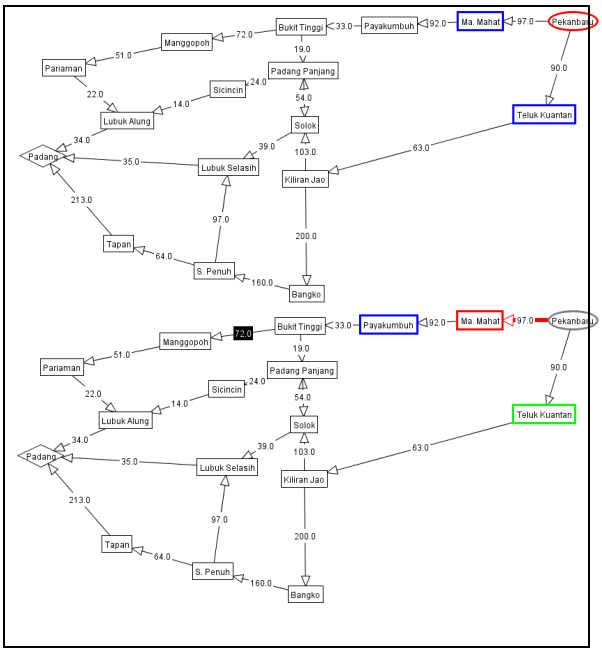
di mana *h** adalah heuristic optimal, yaitu jarak tepat untuk mencapai x dari tujuan. Dengan kata lain, kesalahan pada h tidak boleh lebih besar dari logaritma h* yang mengembalikan nilai jarak sebenarnya dari x ke simpul tujuan.

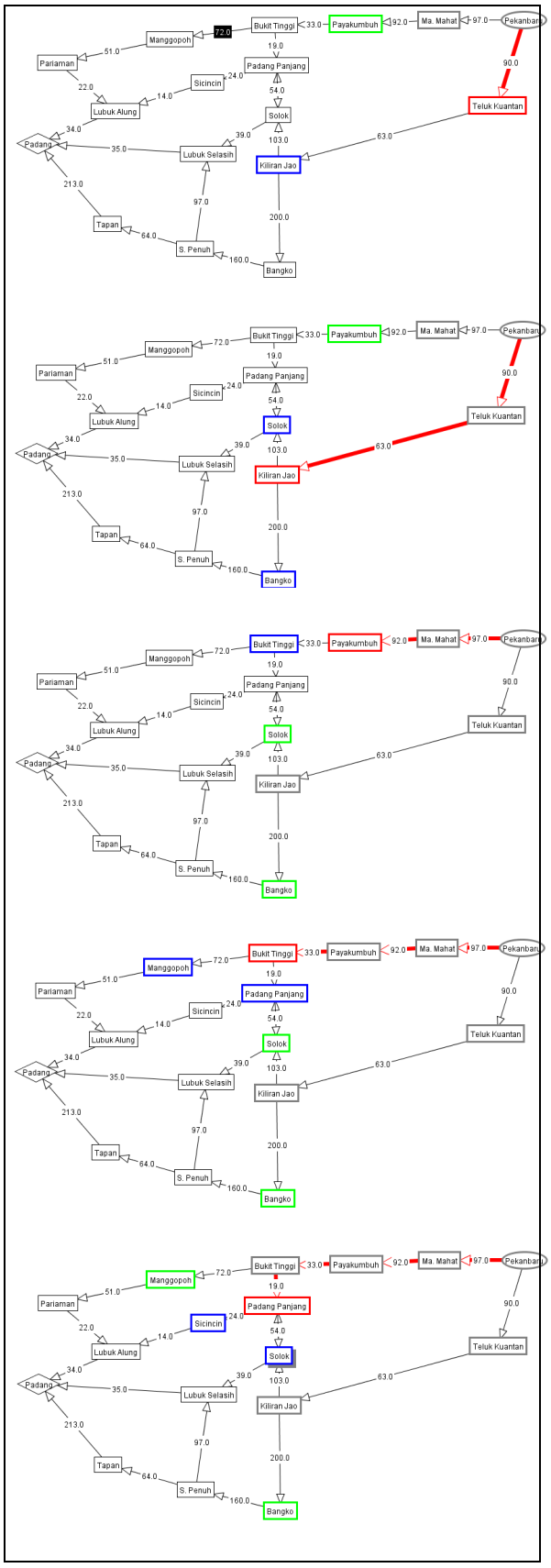
2.3. Proses Pencarian

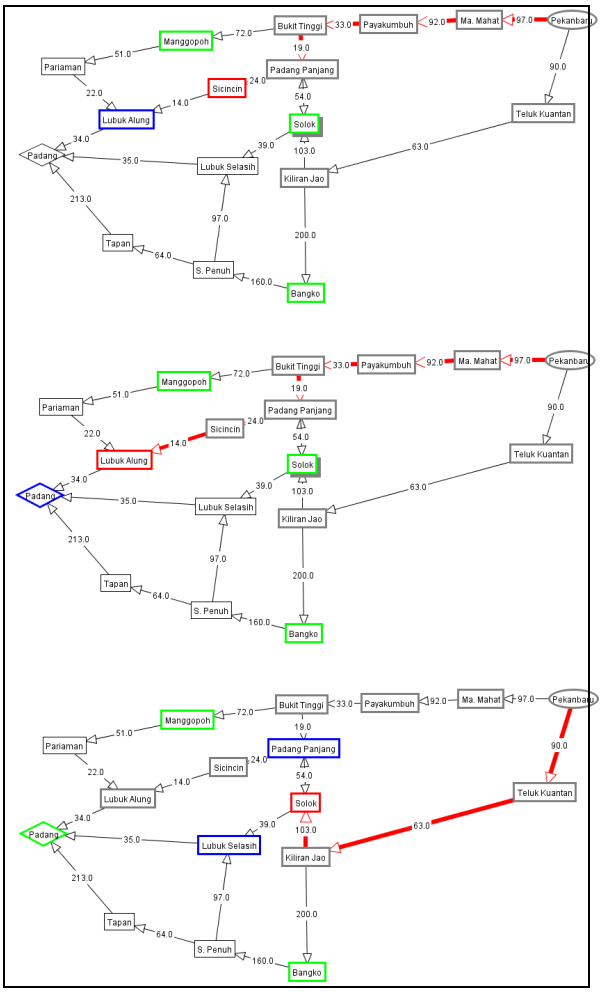
Untuk contoh ini, graf yang akan digunakan adalah graf berarah yang disertai dengan nilai-nilai pada sisi-sisinya, yang mana nilai-nilai tersebut mewakili *cost* atau biaya dari tiap *path*.

Tujuan dari contoh ini adalah mencari jalur terpendek dari simpul awal, yaitu Pekanbaru, mencapai simpul tujuan, yaitu Padang.

Berikut adalah gambar dari tahap-tahap pencarian:







Gambar 3 Representasi Proses Pencarian

Pada Gambar 2, jalur yang optimal ditunjukkan dengan sisi-sisi yang berwarna merah. Adapun antrian prioritas akhir yang didapat adalah sebagai berikut:

- Simpul: Manggopoh
  Path Cost: 294.
  Path: Pekanbaru --> Ma. Mahat --> Payakumbuh --> Bukit Tinggi --> Manggopoh
- Simpul: Lubuk Selasih
  Path Cost: 295.0
  Path: Pekanbaru --> Teluk Kuantan --> Kiliran Jao --> Solok --> Lubuk Selasih
- Simpul: Solok
  Path Cost: 295.0
  Path: Pekanbaru --> Ma. Mahat --> Payakumbuh --> Bukit Tinggi --> Padang Panjang --> Solok

- Simpul: Bangko
  Path Cost: 353.0
  Path: Pekanbaru --> Teluk Kuantan --> Kiliran Jao --> Bangko
- Simpul: Padang Panjang
  Path Cost: 310.0
  Path: Pekanbaru --> Teluk Kuantan --> Kiliran Jao --> Solok --> Padang Panjang

3. HASIL DAN PEMBAHASAN

Dari sederetan graf-graf pada gambar 2 di atas maka dapat di analisis bagaimana algoritma A* melakukan pencarian.

Dalam hal ini, terlihat bahwa yang pertama di ekspansi adalah Pekanbaru sebagai simpul awal. Setelah itu, maka pencarian dilakukan pada simpul-simpul yang di dapat dari ekspansi simpul Pekanbaru, yaitu Ma. Mahat dan Teluk Kuantan. Setelah memasukkan fungsi *f(x)* ke dalam ke dua simpul, maka masing-masing simpul di ekspansi berdasarkan urutan yang telah ditentukan dari nilai *f(x)* masing-masing simpul.

Sebagaimana terlihat dari graf-graf tersebut, hal ini berlangsung terus sampai didapati bahwa dari semua simpul-simpul yang bisa di ekspansi, maka jalur tercepat untuk mencapai Pekanbaru dari Padang adalah Pekanbaru → Ma. Mahat → Payakumbuh → Bukit Tinggi → Padang Panjang → Sicincin → Lubuk Alung → Padang, seperti yang bisa dilihat dari gambar 3.

4. KESIMPULAN

Dari hasil dan pembahasan di atas, maka terlihat bagaimana graf dapat digunakan untuk menunjukkan proses-proses pencarian dalam Algoritma A* dalam wujud yang lebih mudah ditelaah dan dipahami, terutama dibandingkan apabila kita hanya melihat antrian prioritas atau *priority queue* yang sulit untuk dipahami dengan hanya sepintas lihat. Dengan bantuan graf-graf tersebut maka pemahaman akan proses suatu algoritma, bukan hanya A*, serta dapat menunjukkan apakah suatu algoritma bekerjas sebagaimana dimaksudkan.

Dari hasil pengujian di atas, maka terlihat bahwa dalam kasus ini, Algoritma A* berhasil mendapatkan jalur yang optimal dalam menyelesaikan permasalahan. Perlu diperhatikan bahwa pencarian dilakukan dalam graf berarah, artinya ekspansi yang bisa dilakukan lebih sedikit dibandingkan bila pencarian dilakukan terhadap graf tidak berarah yang memungkinkan ekspansi yang lebih banyak untuk tiap simpul. Dalam pencarian pada graf yang tidak berarah, maka kemungkinan besar waktu yang dibutuhkan akan lebih lama dan tidak bisa selesai hanya dalam 10 langkah seperti yang ditampilkan pada gambar 2.

DAFTAR REFERENSI

1[1] Russel, Stuart & Peter Norvig. 2003. “Artificial Intelligence, a Modern Approach”.

2[2] Rinaldi Munir, MT. 2004. “Diktat Kuliah IF2151 Matematika Diskrit Edisi Keempat”.