Analisis Algoritma Pencarian Rute Terpendek Dengan Algoritma Dijkstra dan Bellman - Ford

Gilang Adi Nugroho[1], Agam Syauqi Lamaida[2], Yahdi Ahsana[3]
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13012@students.if.itb.ac.id[1],
if13056@students.if.itb.ac.id[2], if13123@students.if.itb.ac.id[3]

Abstrak

Permasalan pencarian rute terpendek merupakan suatu masalah yang sangat terkenal di dunia Informatika. Dari dahulu hingga sekarang telah dikembangkan berbagai algoritma untuk memecahkan permasalahan ini. Persoalan yang terkenal dalam pencarian rute terpendek adalah persoalan pedagang keliling (traveling salesperson problem - TSP). Seperti yang telah ditulis di atas, hingga saat ini telah banyak yang menemukan solusi untuk pencarian rute terpendek ini. Salah satunya yang terkenal adalah algoritma dijkstra. Selain itu, sebagai pembanding keefektivan algoritma ini kami membandingkannya dengan algoritma Bellman – Ford yang juga merupakan algoritma yang cukup banyak dipakai dalam peermasalahan ini.

Kata kunci: *Dijkstra,Bellman – ford, Shortest path, Travellong salesperson.*

1. Pendahuluan

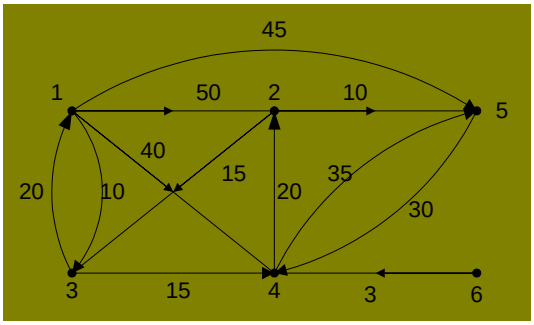
Permasalahan utama pencarian rute terpendek tentu saja mencari rute atau jalur terpendek yang memungkinkan. Namun untuk implementasinya, persoalan ini dapat dikembangkan lebih luas lagi diantaranya untuk mencari biaya minimum, dll. Intinya adalah mencari solusi yang palin efektiv yang dapat diterpakan dalam persoalan yang dihadapi.

2. Analisis Algoritma Dijkstra dan Bellman-Ford

2.1 Algoritma Dijkstra

Algoritma Dijkstra, dinamakan sesuai dengan nama penemunya, seorang ilmuwan komputer berkebangsaan Belanda yang bernama Edsger Dijkstra, adalah algoritma yang digunakan untuk mencari lintasan terpendek pada sebuah graf berarah.

Cara kerja algoritma Dijkstra memakai stategi *greedy*, dimana pada setiap langkah dipilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih.



Gambar 1. Graf Berarah

Iterasi	Simpul yang dipilih	Lintasan	y						D					
			1	2	3	4	5	6	1	2	3	4	5	6
Inisial	-	-	0	0	0	0	0	0	0	50 (1,2)	10 (1,3)	40 (1,4)	45 (1,5)	∞ (1,6)
1	1	1	1	0	0	0	0	0	∞	50 (1,2)	10 (1,3)	40 (1,4)	45 (1,5)	∞ (1,6)
2	3	1,3	1	0	1	0	0	0	∞	50 (1,2)	10 (1,3)	25 (1,3,4)	45 (1,5)	∞ (1,6)
3	4	1,3,4	1	0	1	1	0	0	∞	45 (1,3,4,2)	10 (1,3)	25 (1,3,4)	45 (1,5)	∞ (1,6)
4	2	1,3,4,2	1	1	1	1	0	0	∞	45 (1,3,4,2)	10 (1,3)	25 (1,3,4)	45 (1,5)	∞ (1,6)
5	5	1,5	1	1	1	1	1	0	∞	45 (1,3,4,2)	10 (1,3)	25 (1,3,4)	45 (1,5)	∞

Tabel 1. Penyelesaian Graf pada Gambar 1

2.2 Algoritma Bellman-Ford

Algoritma Bellman-Ford ,seperti halnya algoritma dijkstra,digunakan untuk mencari lintasan terpendek pada sebuah graf berarah. Yang membedakan keduanya adalah pada algoritma Bellman-ford bisa digunakan untuk graf yang memiliki sisi dengan bobot negatif, walaupun menggunakan waktu yang lebih lama.

Kompleksitas algoritma ini sebesar O(nm) dimana n adalah jumlah simpul dan m adalah jumlah sisi.

Berikut adalah salah satu contoh implementasi dari algoritma Bellman-Ford :

```
/* pendefinisian tipe data untuk sebuah
graf */
record vertex {
  list edges
  real distance
  vertex predecessor
}
record edge {
  node source
  node destination
  real weight
}

function BellmanFord(list vertices, list
edges, vertex source)
/* pengimplementasian terhadap graf yang
direpresentasikan oleh list of simpul dan
sisi, dan mengubah simpul sehingga jarak
dan predesesornya menyimpan jarak
terpendek */

// inisialisasi graf
for each vertex v in vertices:
 if v is source then
    v.distance = 0
 else
    v.distance := infinity
    v.predecessor := null

// periksa setiap simpul
for i from 1 to size(vertices):
  for each edge uv in edges:
    u := uv.source
    v := uv.destination
// uv is the edge from u to v
    if v.distance>u.distance+uv.weight
      v.distance:=u.distance+uv.weight
      v.predecessor := u

// mencari loop yang berbobot negatif
for each edge uv in edges:
  u := uv.source
  v := uv.destination
  if v.distance > u.distance + uv.weight
    error "Graf mengandung loop negatif"
```

3.Kesimpulan

Baik algoritma dijkstra maupun bellman-ford sama2 digunakan untuk mencari lintasan terpendek. Namun, tidak seperti algoritma dijkstra, algoritma bellman-ford dapat digunakan pada graf yang mengandung simpul negatif, selama graf tersebut tidak mengandung kalang negatif yang dapat dicapai dari titik awal. Algoritma dijkstra lebih menguntungkan dari sisi *running time* , namun untuk permasalahan khusus yang mengandung simpul negatif, algoritma bellman-Ford lah yang lebih menguntungkan.

4. Daftar Pustaka

[1] Munir Rinaldi, Diktat Kuliah Strategi Algoritmik, 193, 2005.
[2] http://en.wikipedia.org/wiki/Bellman-Ford_algorithm
[3]http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm