Pemanfaatan Algoritma Shortest-Path untuk Pencarian Jalur Efektif dalam Aplikasi-Aplikasi Peta

Nathan James Runtuwene
Teknik Informatika, Sekolah Teknik Elektro dan Informatik
Institut Teknologi Bandung
Bandung, Indonesia
13514083@std.stei.itb.ac.id

***Abstract*—Makalah ini berisi mengenai algoritma-algoritma yang digunakan oleh aplikasi-aplikasi pencari jalur untuk menentukan jalur paling efektif yang dapat dilalui. Algoritma yang digunakan merupakan kombinasi antara beberapa algoritma shortest-path dan juga metode-metode lain seperti machine learning dan big data mining untuk mempercepat proses penghitungan jalur efektif.**

Keywords— dijkstra; map; shortest-path;

I. PENDAHULUAN

Hampir semua aplikasi-aplikasi peta sekarang sudah memiliki fitur pencari jalan. Pencari jalan ini dapat menemukan jalan tercepat antara 2 titik yang ditentukan pengguna. Pengguna juga dapat menentukan dengan kendaraan apa yang pengguna ingin gunakan untuk mencapai tujuannya, contohnya apakah menggunakan mobil, bus, kereta, ataupun dengan berjalan kaki. Selain itu, pencari jalan ini juga dapat mempertimbangkan kemacetan dalam perjalanan, cuaca, dan juga jalan yang tidak rata. Penelitian dalam bidang ini memiliki kemajuan yang cukup signifikan dalam beberapa tahun terakhir akibat banyaknya pengguna yang menggunakannya dan juga keuntungan yang dapat diperoleh dari fitur ini.

Ada banyak sekali aplikasi yang menawarkan fitur ini, tetapi tidak ada yang benar-benar membuka dengan algoritma apa mereka mengimplementasikan fitur ini. Dalam makalah ini kita akan melihat beberapa algoritma pencarian jalan dasar yang mungkin merupakan salah satu fondasi dari algoritma yang digunakan oleh aplikasi-aplikasi peta ini dan juga bagaimana mereka merepresentasikan sebuah peta agar dapat digunakan bersama dengan algoritma-algoritma dasar tersebut.

II. DASAR TEORI

A. Teori Graf

Sebuah graf adalah kumpulan simpul yang dihubungkan oleh sisi. Ada banyak tipe dari graf, ada yang berarah, ada yang memiliki cost berbeda untuk sisi-sisinya, dan lain lain. Graf sangat berguna dalam bidang informatika karena dengan menggunakan graf kita dapat merepresentasikan banyak hal yang mungkin bentuknya abstrak atau tidak nyata seperti hubungan relasi antar orang menjadi sesuatu yang lebih nyata dan lebih dapat dimengerti oleh komputer. Selain hal-hal yang abstrak, sebuah graf juga dapat mempermudah merepresentasikan hal yang kompleks, seperti jalanan yang dapat dirubah menjadi bentuk graf dengan salah satunya adalah mengubah persimpangan menjadi simpul dan jalan antar persimpangan sebagai sisi.

B. Algoritma shortest-path

1) Dijkstra

Algoritma Dijkstra ditemukan oleh Edsger W. Dijkstra pada tahun 1956. Algoritma ini akan mencari shortest-path dalam sebuah graf dengan menggunakan konsep greedy.

Untuk suatu simpul tertentu dalam graf, algoritma ini akan mencari shortest-path dari simpul ini ke semua simpul lainnya. Alur kerja algoritma Dijkstra adalah sebagai berikut:

1. Simpul awal diinisialisasikan dengan nilai distance 0, dan simpul lainnya dengan nilai tak hingga.

2. Proses simpul dengan nilai distance terkecil sebagai berikut: Untuk semua simpul yang terhubung oleh tepat satu buah sisi dengan simpul itu, ubah nilai distance simpul tersebut menjadi nilai minimum antara nilai distance simpul sekarang dengan nilai distance simpul sebelumnya ditambah dengan cost sisi yang menghubungkan mereka.

3. Ulangi langkah 2 sampai setiap simpul sudah diproses, nilai distance di setiap simpul adalah nilai shortest-path antara simpul awal dengan simpul tersebut.

Algoritma Dijkstra adalah algoritma shortest path yang cukup umum digunakan karena algoritma ini dapat mencari hasil shortest-path dengan kompleksitas yang hampir linear, tidak seperti algoritma Bellman-Ford atau Floyd-Warshall. Namun algoritma Dijkstra memiliki kelemahan yaitu dia tidak bisa menghitung shortest path suatu graf bila pada graf tersebut memiliki sisi dengan harga negatif.

2) A Algorithm*

Algoritma A* dijalankan mirip dengan algoritma Dijkstra, perbedaannya adalah pada algoritma ini lebih ditujukan pada pencarian shortest-path antara 2 buah titik tertentu dan juga algoritma ini menggunakan heuristik sebagai pertimbangan simpul mana yang akan diproses terlebih dahulu. Algoritma ini pertama dipublikasikan oleh Peter Hart, Nils Nilsson dan Bertram Raphael dari Stanford Research Institute.

Keuntungan algoritma A* dibandingkan dengan algoritma Dijkstra adalah algoritma A* mungkin dapat menemukan hasil dengan lebih cepat karena dia dapat mengabaikan simpul-simpul yang dianggap terletak jauh dari simpul tujuan atau memeiliki nilai heuristik yang besar. Penentuan nilai heuristik ini memiliki banyak cara tetapi ini sangat penting dan akan sangat berpengaruh pada jalannya algoritma ini. Penentuan nilai heuristik yang baik dapat menemukan solusi yang benar dalam waktu yang sangat singkat dibandingkan algoritma Dijkstra, sedangkan penentuan heuristik yang buruk dapat berpengaruh pada performansi algoritma atau bisa juga menyebabkan hasil yang dihasilkan menjadi salah.

3) Jump-Point search

Algoritma ini adalah optimisasi dari algoritma A* yang hanya berlaku untuk graf berbentuk grid dengan cost yang sama rata. Algoritma ini akan mengabaikan simpul-simpul dengan didasari oleh asumsi yang dapat dibuat mengenai tetangga-tetangga simpul sekarang. Akibatnya algoritma ini dapat melakukan "loncatan" sepanjang garis lurus dari suatu simpul ke simpul lainnya dibandingkan harus memproses satu-satu simpul dari awal sampai simpul tersebut.

4) Prim

Algoritma Prim sebenarnya bukan algoritma shortest-path, melainkan algoritma untuk mengkalkulasikan Minimum Spanning Tree dari suatu graf, tetapi minimum spanning tree dari suatu graf ini dapat digunakan sebagai perbandingan antara beberapa rute yang mungkin diambil.

III. Implementasi Algoritma Shortest-Path dalam Peta

A. Representasi peta sebagai graf

Agar kita bisa menggunakan algoritma shortest-path dalam sebuah peta kita perlu untuk pertama merepresentasikan peta tersebut sebagai suatu graf. Langkah pertama yang cukup umum dilakukan adalah merepresentasikan setiap persimpangan sebagai simpul dan jalan antar persimpangan sebagai sisi. Bisa juga ditambahkan biaya suatu sisi sebagai lama waktu yang dibutuhkan untuk berjalan dari suatu persimpangan ke persimpangan lainnya yang dihubungkan oleh sisi tersebut. Penghitungan biaya ini dapat menggunakan beberapa faktor antara lain cuaca, waktu atau jam, kondisi jalan, dan lain-lain.

Hal lain yang dilakukan oleh beberapa aplikasi peta ini adalah memberikan lapisan-lapisan untuk tipe-tipe jalan (jalan tol, jalan utama, jalan kecil) agar algoritma nanti dapat melakukan pencarian dengan mengutamakan tipe-tipe jalan tertentu dan menghindari beberapa tipe jalan lainnya.

Jika seseorang pengguna ingin mengetahui jalan dari suatu tempat ke tempat lain yang jauh, beberapa aplikasi ini melakukan pembagian segmen atau daerah terhadap bagian-bagian tertentu agar proses pencarian tidak perlu memperhitungkan terlalu banyak simpul dalam sekali jalan.

Ada banyak sekali aplikasi yang menawarkan fitur ini, tetapi tidak ada yang benar-benar mengekspos pada publik dengan algoritma apa mereka mengimplementasikan fitur ini. Dalam makalah ini kita akan melihat beberapa algoritma pencarian jalan dasar yang mungkin merupakan salah satu fondasi dari algoritma yang digunakan oleh aplikasi-aplikasi peta ini dan juga bagaimana mereka merepresentasikan sebuah peta agar dapat digunakan bersama dengan algoritma-algoritma dasar tersebut.

Beberapa aplikasi juga memiliki fitur untuk menentukan jalan yang harus diambil dengan menggunakan bus atau kereta. Aplikasi ini akan menyimpan semua rute-rute dan stasiun-stasiun atau halte-halte bus atau kereta yang ada dalam suatu kota tertentu, lalu mengkalkulasikan jarak terdekat dari titik tertentu ke suatu stasiun untuk menentukan jalan terdekat yang perlu ditempuh, karena rute-rute bus atau kereta cenderung tidak akan pernah berubah, pre-processing bisa sangat membantu dalam hal ini.

Penghitungan harga sisi suatu jalan juga memiliki beberapa cara. Salah satu aplikasi yang cukup terkenal dan sering digunakan yaitu Google Maps menggunakan crowd-sourcing untuk menentukan kecepatan rata-rata kendaraan di suatu ruas jalan tertentu. Hal ini kemudian digunakan untuk menentukan waktu yang diperlukan atau harga dari suatu ruas jalan tertentu.

B. Algoritma pencarian jalan

Algoritma yang diimplementasikan sebagai algoritma pencari jalan tidak benar-benar menggunakan salah satu algoritma tersebut melainkan gabungan dari algoritma-algoritma yang sudah disebutkan diatas dengan beberapa algoritma lain termasuk machine learning yang dapat membantu menentukan kecenderungan jalur yang diambil oleh orang banyak. Contoh dari cara-cara yang digunakan bisa dengan antara lain mencari jalur dengan melakukan dijkstra atau path finding dari kedua ujung (awal dan tujuan) lalu bertemu di tengah, atau dengan membagi-bagi daerah agar tidak banyak persimpangan yang harus diperhitungkan.

Selain dari yang sudah disebutkan diatas, salah satu cara yang sering sekali digunakan untuk mempercepat waktu proses pencarian jalan adalah dengan melakukan pre-processing, yaitu dengan melakukan proses terlebih dahulu terhadap jalan-jalan yang sering dilewati. Hal ini dapat mempersingkat perhitungan dengan drastis, tetapi juga membutuhkan memori yang sangat banyak.

Satu lagi cara yang cukup umum digunakan oleh aplikasi-aplikasi pencari jalan adalah dengan memberikan dahulu solusi atau jalan sementara yang belum tentu optimal, tetapi sudah cukup untuk memperkirakan waktu yang diperlukan untuk mencapai suatu tujuan tertentu.

IV. KESIMPULAN DAN SARAN

Dari yang sudah disebutkan diatas, algoritma-algoritma shortest path memiliki manfaat yang sangat berguna, untuk itu sangat penting untuk dilakukan lebih banyak penelitian dan percobaan dalam bidang ini untuk mencari algoritma yang seefisien mungkin dapat menemukan jalan yang efektif.

UCAPAN TERIMA KASIH

Makalah ini dibuat di Institut Teknologi Bandung. Terima kasih kepada bapak Rinaldi Munir yang telah memberikan materi untuk membuat makalah ini dan juga kesempatan untuk membuat makalah ini.

REFERENCES

[1] Delling, D.; *Sanders, P.*; Schultes, D.; *Wagner, D.* (2009). "Engineering route planning algorithms". Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation. Springer. pp. 117–139

[2] Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs" (PDF). *Numerische Mathematik*

[3] http://research.microsoft.com/pubs/207102/MSR-TR-2014-4.pdf

[4] http://www.cc.gatech.edu/~thad/6601-gradAI-fall2012/02-search-Gutman04siam.pdf