BAB II

LANDASAN TEORI

2.1 *Shortest Path Problem*

Dalam teori graf, s*hortest path problem* merupakan permasalahan untuk menemukan jalan antara dua simpul dalam graf sehingga jumlah dari bobot kanal penyusunnya dapat bernilai seminimal mungkin. Permasalahan *shortest path* dapat diimplementasikan untuk graf berarah, tanpa arah, atau gabungan keduanya. Penelitian ini terfokus pada *shortest path problem* dengan pendekatan menggunakan topologi jaringan *mesh*, sehingga graf yang digunakan merupakan graf berarah. Graf berarah mengharuskan simpul untuk berturut-turut dihubungkan melalui kanal ke tujuan yang tepat.

2.1.1 Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh ilmuwan komputer Belanda Edsger W. Dijkstra pada tahun 1956 dan diterbitkan pada tahun 1959 (Misa, 2010). Algoritma ini memecahkan masalah jalan jalan terpendek satu sumber untuk graf dengan bobot kanal non-negatif. Algoritma ini sering digunakan dalam *routing* dan sebagai *sub-routine* dalam algoritma graf lainnya (Misa, 2010).

Metode pelabelan Dijkstra merupakan prosedur utama dalam algoritma tersebut (Faramroze Engineer, 2005). Keluaran dari metode pelabelan berupa rangkaian alur dari simpul sumber ke suatu set simpul lain. Alur keluaran merupakan jarak terpendek dari simpul sumber yang dapat ditemukan oleh

algoritma. Tiga informasi penting yang diperlukan untuk setiap simpul dalam metode pelabelan diantaranya label jarak, label simpul sebelumnya, dan set simpul yang berlabel permanen.

Label jarak menyimpan batas jarak terpendek dari sumber ke tujuan, sedangkan label simpul sebelumnya mencatat indeks simpul yang dilalui. Jika simpul belum ditambahkan ke dalam set simpul, maka dianggap belum terjangkau. Biasanya label jarak simpul yang belum terjangkau diatur bernilai tak terbatas. Ketika diketahui bahwa jalan terpendek dari suatu simpul ke simpul lain merupakan benar-benar jalan terpendek, maka simpul tersebut disebut label permanen. Namun, jika masih terdapat kemungkinan ditemukannya jalan terpendek yang berasal dari simpul lain, maka simpul tersebut dianggap hanya berlabel sementara.

Nilai dari label jarak merupakan batas atas jarak jalur terpendek ke suatu simpul. Jika simpul untuk sementara diberi label dan nilai label jarak menjadi jarak jalan akhir yang terpendek maka simpul tersebut akan dimasukkan dalam set simpul berlabel permanen. Secara iteratif menambahkan simpul berlabel sementara dengan label jarak terkecil ke set simpul berlabel permanen, algoritma Dijkstra dapat menjamin optimalitas dari pencarian. Keuntungan dengan pelabelan pada algoritma Dijkstra adalah pencarian tersebut dapat dihentikan ketika semua simpul tujuan secara permanen telah terlabel.

Berikut adalah *pseudocode* dari algoritma Dijkstra (Anderson, 2010).

```
initialize the cost of each node to ∞
initialize the cost of the source to 0
while there are unknown nodes left in the graph
    select an unknown node b with the lowest cost
```

```
mark b as known
for each node a adjacent to b
    if b's cost + cost of (b, a) < a's cost
        a's cost = b's cost + cost of (b, a)
        a's prev path node = b
```

2.1.2 Algoritma A*

Algoritma A* diperkenalkan oleh Peter Hart, Nils Nilsson dan Bertram Raphael dari Stanford Research Institute dengan mengintegrasikan konsep heuristik ke dalam prosedur pencarian (Hart, 1968). Dalam pencarian, simpul berikutnya diberikan label permanen jika memiliki nilai jarak yang minimal (yang diukur dari simpul awal). Pemilihan simpul didasarkan pada jarak dari simpul awal ditambah perkiraan jarak ke tujuan (perkiraan heuristik). Untuk membangun jalur terpendek dari simpul asal ke tujuan, digunakan jarak asli dari akumulasi sepanjang kanal (seperti dalam algoritma Dijkstra) ditambah perkiraan jarak ke tujuan. Sehingga diperlukan suatu informasi global tentang jaringan untuk memandu dalam pencarian jalur terpendek tersebut. Pada dasarnya algoritma ini menggunakan gabungan dua informasi, diantaranya informasi batas atas dan perkiraan jarak ke simpul tujuan. Algoritma A* lebih mementingkan jalan yang mengarah ke simpul tujuan dibandingkan yang bergerak menjauh dari tujuan.

Berikut adalah *pseudocode* dari algoritma A* (Nilsson, 1998).

```
create the open list of nodes, initially containing only our starting node
    create the closed list of nodes, initially empty
        while we have not reached goal
            consider the best node in the open list
            if this node is the goal then we're done
            else
                move the current node to the closed list
```

```
for each neighbor
    if this neighbor is in the closed list and current g value is lower
        update the neighbor with the new, lower, g value
        change the neighbor's parent to our current node
    else if this neighbor is in the open list and our current g value is lower
        update the neighbor with the new, lower, g value
        change the neighbor's parent to our current node
    else this neighbor is not in either the open or closed list
        add the neighbor to the open list and set its g value
```

2.1.3 Algoritma Floyd-Warshall

Algoritma Floyd-Warshall juga dikenal sebagai algoritma Floyd, algoritma Roy-Warshall, algoritma Roy-Floyd, atau algoritma WFI. Algoritma ini dipublikasian oleh Robert Floyd pada tahun 1962 (Weisstein, 2009). Namun, pada dasarnya sama dengan algoritma sebelumnya diterbitkan oleh Bernard Roy pada tahun 1959 dan juga oleh Stephen Warshall pada 1962 untuk menemukan penutupan transitif dari grafik (Weisstein, 2009). Perumusan modern algoritma Warshall sebagai tiga bersarang untuk-loop pertama kali dijelaskan oleh Peter Ingerman, juga pada tahun 1962 (Weisstein, 2009).

Algoritma Floyd-Warshall melakukan pencarian jalur terpendek dengan cara mengkomputasi semua pasangan simpul kedalam sebuah graf berbobot (Ariffin, 2011). Algoritma ini diawali dengan mencari semua jarak minimal antara pasangan simpul tanpa melewati simpul perantara. Nilai-nilai ini dicatat dalam sebuah tabel jarak minimal. Jarak minimal antara pasangan simpul ditentukan dengan melakukan perbandingan dengan nilai sebelumnya. Setiap perubahan akan dimasukkan ke dalam tabel jarak minimal. Hal ini dilakukan oleh simpul pertama

hingga simpul terakhir, proses akan berulang dengan menggunakan hasil sebelumnya sebagai pembanding.

Berikut adalah *pseudocode* algoritma Floyd-Warshall (Kenwood, 2008).

```
initialise dists to the adjacency matrix of the graph
for each k from 0 to number of node
    for each i from 0 to number of node
        for each j from 0 to number of node
            if dists[i][j] > dists[i][k] + dists[k][j]
                dists[i][j] = dists[i][k] + dists[k][j]
```

2.2 Beban Komputasi

Beban komputasi berkaitan dengan sumber daya yang diperlukan untuk menghitung dan memproses solusi sebagai fungsi dari ukuran masalah (Ruohonen, 2008). Ukuran masalah diukur dari banyaknya jumlah masukan dan sumber daya yang dibutuhkan. Untuk membuat perbandingan beban komputasi, diperlukan perhitungan banyaknya operasi yang dilakukan dalam memproses solusi tersebut.

2.3 Simulator

Simulator adalah model dari suatu sistem yang digunakan untuk menguji dan mempelajari sistem sebelum diimplementasikan pada dunia nyata (Banks, 2001). Simulator saat ini kebanyakan berupa perangkat lunak sehingga meminimalisasi biaya pemodelan. Perangkat lunak simulasi dapat juga digunakan sebagai permainan seperti simulasi penerbangan, simulasi berkendara, dan sebagainya. Simulator diharapkan dapat meminimalisasi resiko kegagalan dalam penerapannya di dunia nyata.

2.3.1 Simulator Jaringan

Kompleksitas dari sistem komunikasi modern merupakan salah satu pendorong penggunaan simulasi (Banks, 2001). Sistem komunikasi modern dibutuhkan untuk beroperasi pada kecepatan data tinggi dengan daya dan bandwidth yang terbatas. Kompleksitas ini hasil dari arsitektur sistem komunikasi modern dan dari lingkungan di mana sistem ini digunakan. Pengembangan paket perangkat lunak yang kuat yang ditargetkan untuk sistem komunikasi telah mempercepat perkembangan simulasi komunikasi. Dengan demikian, peningkatan kompleksitas sistem telah disertai dengan peningkatan daya komputasi.

Pertumbuhan teknologi komputer diiringi dengan pertumbuhan yang cepat dalam teori simulasi. Akibatnya, alat dan metodologi makin mendukung desain dan analisis sehingga lebih mudah dipahami daripada yang terjadi beberapa dekade yang lalu. Motivasi penting untuk penggunaan simulasi adalah bahwa simulasi adalah alat yang berharga untuk memperoleh wawasan tentang perilaku sistem. Sebuah simulasi yang dikembangkan dengan benar, dapat menjadi sebuah laboratorium bagi sistem tersebut. Pengukuran dengan mudah dapat dibuat pada berbagai titik dalam sistem yang diteliti.

2.3.2 Simulator yang Digunakan

Pada penelitian ini penulis mengunakan perangkat lunak simulasi OMNeT++. OMNeT++ adalah sebuah simulator kejadian diskret yang moduler, mudah dikembangkan dengan memanfaatkan komponen berbasis C++ (Varga, 2003). OMNeT++ digunakan karena menyediakan antar muka yang kaya dan bahasa pemrograman yang abstrak (Weingartner, 2009). Kerangka kerja yang

terintegrasi digunakan untuk membangun simulator jaringan. Jaringan dimaksudkan dalam arti luas yang mencakup jaringan komunikasi kabel dan nirkabel, *on-chip*, jaringan antrian, dan sebagainya. Pada OMNeT++ juga disediakan fungsi spesifik domain seperti misalnya dukungan untuk jaringan sensor nirkabel, jaringan *ad-hoc*, internet protokol, jaringan fotonik, dan sebagainya. Model kerangka kerja ini dikembangkan sebagai proyek independen yang *open source*. OMNeT++ menggunakan basis Eclipse IDE dan lingkungan *runtime* grafis. Terdapat komponen untuk simulasi *real-time*, emulasi jaringan, bahasa pemrograman alternatif (Java, C#), integrasi database, dan beberapa fungsi lainnya (Varga, 2003).