BAB II
KAJIAN TEORI

2.1 Kajian Penelitian Sebelumnya

Traveling salesman problem (TSP) merupakan salah satu permasalahan yang telah sering diangkat dalam berbagai studi kasus dengan penerapan berbagai metode penyelesaian. Berikut adalah pemaparan beberapa studi kasus mengenai *traveling salesman problem*.

Penelitian yang pertama merupakan analisis penyelesaian TSP dengan metode *brute-p*Andrew Wilson dan kawan-kawan ini membahas tentang komputasi paralel yang memiliki kompleksitas tinggi sehingga dibutuhkan waktu pengerjaan yang cepat. Salah satu pendekatan komputasi paralel adalah *Graphic Processing Unit* (GPU) *Computing*, dimana dalam sebuah GPU terdapat banyak *thread* yang mampu ditugaskan secara paralel. Metode yang digunakan adalah dengan metode *brute-force*, dimana metode *brute-force* bekerja dengan mengenumerasi semua kandidat kemungkinan yang ada, sehingga menghasilkan solusi yang terbaik. Pada penelitian ini akan diimplementasikan *brute-force* dengan teknik *Exhaustive Search* pada GPU, dan menganalisis jumlah *thread process* pada setiap *thread* dan pengaruh *block* dan *thread* pada GPU. Setelah dilakukan penelitian dan uji statistik, didapatkan bahwa perubahan *thread process* yang semakin besar akan mengakibatkan persentase penurunan waktu semakin besar dan juga semakin banyak *thread process* maka kecepatan komputasi GPU akan menuju satu titik, karena kecepatan komputasi pada *device* GPU telah mencapai titik maksimal.

Penelitian selanjutnya adalah permasalahan TSP pada pencarian rute terpendek di PT. Jalur Nugraha Ekakurir (JNE) Semarang (Wicaksana, 2013). Penyelesaian untuk permasalahan TSP pada penelitian ini memanfaatkan algoritma fuzzy evolusi, dimana algoritma tersebut digunakan untuk suatu pencarian nilai dalam sebuah masalah optimasi dengan bantuan perangkat lunak Matlab. Dengan menggunakan data pengiriman barang oleh kurir dari PT. Jalur Nugraha Ekakurir (JNE) Semarang beserta alamatnya diperoleh data jarak antar lokasi dimana proses pencarian jarak diambil dari Google Maps melalui situs *http://getlatlon.yohman*. Metode ini dilakukan karena dengan cara ini akan didapatkan titik koordinat yaitu garis lintang (latitude) dan garis bujur (longitude) antar lokasi secara lebih akurat tanpa harus mengeluarkan banyak waktu dan biaya dalam pencariannya. Data tersebut kemudian diproses menggunakan bantuan *software* Matlab kemudian didapatlah hasil optimal permasalahan jaringan TSP dalam pengiriman barang oleh PT. Jalur Nugraha Ekakurir ke rumah penerima barang di wilayah Kota Semarang dengan menggunakan variasi populasi dan generasi pada algoritma Fuzzy Evolusi.

Penentuan rute belanja dengan penyelesaian algoritma greedy (Megariza, 2011) juga merupakan kajian penelitian tentang permasalahan *traveling salesman problem*. Efisiensi waktu dalam berbelanja sangat dipengaruhi oleh urutan pembelian barang. Urutan atau rute dalam pembelian barang dapat dimodelkan dengan TSP, dengan *cost* berupa waktu dan *constraint* yang mempengaruhinya adalah kepadatan pengunjung dan jarak. Pada algoritma Greedy Heuristic pemilihan lintasan akan dimulai pada lintasan yang memiliki nilai paling minimum, setiap mencapai suatu kota, algoritma ini akan memilih kota selanjutnya yang belum dikunjungi dan memiliki jarak yang paling minimum. Algoritma ini disebut juga Nearest Neighbour. Kompleksitas algoritma ini sangat mengagumkan, tetapi hasil yang kita dapat bisa sangat jauh dari hasil yang optimal, semakin banyak kota semakin besar pula perbedaan hasil yang dicapai. Kesimpulan dari penelitian ini adalah bahwa penerapan pemodelan dengan TSP dalam studi kasus berbelanja dapat diterapkan. Selain itu, algoritma greedy juga dapat diterapkan dalam studi kasus ini, walaupun belum tentu menghasilkan hasil yang optimal.

Kajian selanjutnya adalah pada penerapan algoritma genetika TSP *with time window* untuk studi kasus antar jemput *laundry* (Suprayogi, 2014). Antar jemput *laundry* dengan pelanggan yang memiliki waktu khusus untuk menerima barang adalah salah satu contoh kasus pemilihan rute. Kasus ini juga harus dipertimbangkan waktu ketersediaan setiap pelanggan. Pencarian solusi untuk permasalahannya adalah dengan mengkombinasikan solusi-solusi (kromosom) untuk menghasilkan solusi baru dengan menggunakan operator genetika (seleksi, crossover dan mutasi). Untuk mencari solusi terbaik digunakan beberapa kombinasi probabilitas *crossover* dan mutasi serta ukuran populasi dan ukuran generasi. Dari nilai-nilai parameter yang digunakan, didapatkan solusi yang memungkinkan untuk melayani semua pelanggan dengan *time window* masing – masing.

Ringkasan dari kajian yang dilakukan terhadap penelitian sebelumnya dan perbandingan dengan penelitian tugas akhir ini dapat dilihat dalam Tabel 2.1:

Tabel 2.1 Kajian penelitian sebelumnya dan penelitian ini

Peneliti	Judul	Metode yang Digunakan	Studi Kasus
Wilson, Andrew dkk, 2013	Analisis Penyelesaian *Traveling salesman problem* Dengan Metode *Brute-force* Menggunakan *Graphic Processing Unit*	Metode *Brute-force*	Komputasi paralel
Wicaksana, Dinar A dkk, 2014	Solusi *Traveling* Salesman Problem Menggunakan Algoritma Fuzzy Evolusi	Algoritma Fuzzy Evolusi	Pencarian rute terpendek pada PT. Jalur Nugraha Ekakurir (JNE) Semarang
Megariza, 2011	Penentuan Rute Belanja dengan TSP dan Algoritma Greedy	Algoritma Greedy	Berbelanja
Suprayogi, D.A. dan Mahmudy,W.F., 2014	Penerapan Algoritma Genetika *Traveling salesman problem* with Time Window: Studi Kasus Rute Antar Jemput Laundry	Algoritma Genetika	Rute Antar Jemput Laundry
Kholidah, Kartika Nur, 2017	Aplikasi Pembuat *Itinerary* Wisata di Provinsi Daerah Istimewa Yogyakarta dengan Pendekatan Solusi *Traveling salesman problem* dan *K-means clustering*	Metode *brute-force* dan *k-means clustering*	Pembuatan *Itinerary* Wisata di Provinsi Daerah Istimewa Yogyakarta

2.2 *Traveling salesman problem*

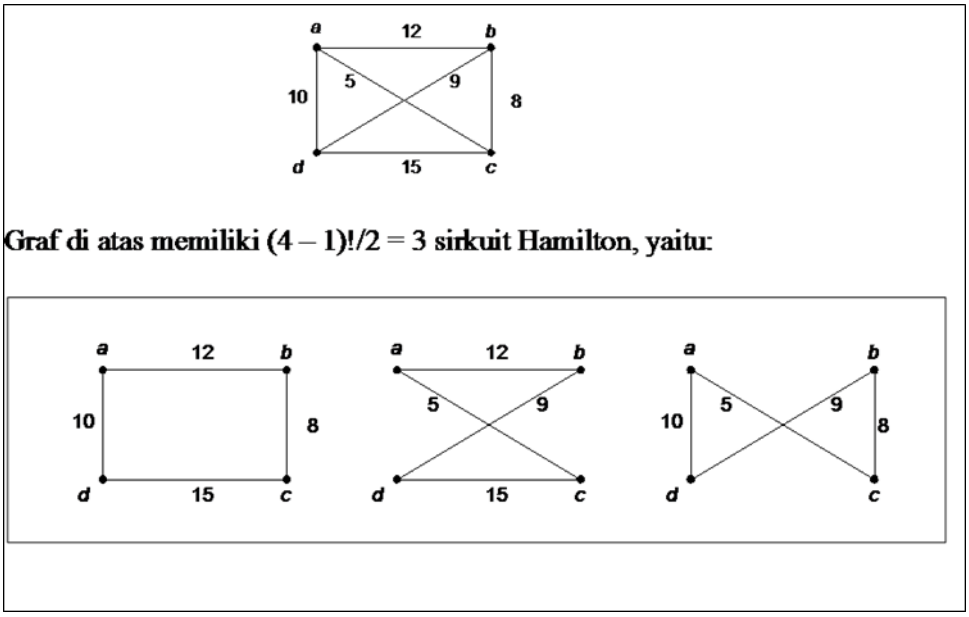
Traveling salesman problem (TSP) merupakan salah satu permasalahan yang sangat terkenal di dalam teori graf. TSP pertama kali dikemukakan oleh matematikawan Irlandia Sir William Rowan Hamilton dan matematikawan Inggris Thomas Kirkman pada tahun 1800-an (Megariza, 2011). Permasalahan ini merupakan jenis permasalahan optimasi dengan deskripsi permasalahannya sebagai berikut: "Diberikan sejumlah kota dan jarak antar kota. Tentukan sirkuit terpendek yang harus dilalui oleh seorang salesman apabila salesman itu berangkat dari sebuah kota asal dan menyinggahi setiap kota tepat satu kali dan kembali lagi ke kota asal keberangkatan" (Munir, 2012).

Pada permasalahan ini, kota dapat dinyatakan sebagai simpul graf, sedangkan sisi menyatakan jalan yang menghubungkan antar dua buah kota. Bobot pada sisi menyatakan jarak antara dua buah kota. Persoalan TSP tidak lain adalah menentukan sirkuit Hamilton yang

memiliki bobot minimum pada sebuah graf terhubung. Sirkuit Hamilton sendiri merupakan lintasan tertutup (sirkuit) yang melalui setiap simpul di dalam graf tepat satu kali, kecuali simpul asal (sekaligus simpul akhir) yang dilalui dua kali.

Persoalan TSP ini memiliki kompleksitas waktu yang tinggi. Apabila setiap simpul memiliki sisi ke simpul yang lain, maka graf yang merepresentasikannya adalah graf lengkap berbobot. Pada sembarang graf lengkap dengan n buah simpul (n>2), jumlah sirkuit Hamilton yang berbeda adalah (n-1)!/2. Dapat dibayangkan jika simpul yang akan dikunjungi sangat banyak, maka waktu yang diperlukan untuk mengevaluasi solusi yang mungkin (untuk mendapatkan solusi terbaik) akan sangat lama. Maka dari itu, selain menggunakan metode deterministik, banyak penelitian yang mengembangkan metode untuk menyelesaikan TSP menggunakan metode heuristik seperti algoritma genetika ataupun *swarm intelligence*.

Untuk kasus TSP dengan jumlah simpul yang tidak terlalu banyak, dapat digunakan pendekatan metode *brute-force* (*exhaustive enumeration*) atau menggunakan algoritma Held-Karp (*dynamic programming*) untuk mencari urutan kunjungan dengan total jarak minimum. Pada Gambar 2.1 ditunjukkan contoh kasus TSP dengan 4 simpul dan juga solusinya menggunakan metode *brute-force*, yaitu dengan mengevaluasi semua kemungkinan solusi.



Gambar 2.1 Contoh kasus TSP dan penyelesaiannya (Munir, 2012)

Berdasarkan contoh pada Gambar 2.1, sirkuit Hamilton terpendek adalah yang ditunjukkan oleh graf paling kanan yaitu dengan urutan kunjungan: (a, c, b, d, a) atau (a, d, b, c, a) yang memiliki $panjang\ sirkuit = 5 + 8 + 9 + 10 = 32$. Solusi ini relatif jauh lebih kecil apabila dibandingkan dengan dua buah graf yang sebelumnya yang memiliki

panjang sirkuit 45 dan 41. Sehingga apabila seorang salesman melakukan kunjungan ke setiap kota sesuai dengan sirkuit pada graf yang paling kanan, maka dia dapat menghemat biaya tempuh yang harus dikeluarkan.

2.3 *K-means Clustering*

K-means merupakan salah satu algoritma yang digunakan untuk *clustering* yang akan membagi data menjadi beberapa kelompok. Algoritma *k-means* merupakan salah satu metode *non-hierarki clustering* yang dapat mengelompokkan data ke dalam beberapa kelompok berdasarkan kesamaan data. Mekanisme ini memungkinkan data yang memiliki karakteristik yang sama dikelompokkan menjadi satu *cluster* dan data yang memiliki karakteristik yang berbeda dikelompokkan dalam *cluster* lainnya.

Untuk menentukan label *cluster* dari data, maka diperlukan perhitungan jarak antara data dengan masing-masing pusat *cluster*. Terdapat beberapa cara yang dapat digunakan untuk melakukan perhitungan jarak, seperti *euclidean distance*, *manhattan distance*, dan *chebichey distance*. Metode *k-means* bertujuan untuk meminimalkan jumlah dari kuadrat jarak antara semua titik data dengan pusat *cluster* masing-masing. Pada Gambar 2.2 ditunjukkan deskripsi dari algoritma *k-means* (Murty dan Devi, 2011).

Langkah 1: Pilih sebanyak k data dari total n data yang tersedia untuk dijadikan sebagai titik-titik pusat *cluster* awal.

Langkah 2: Untuk sebanyak $n - k$ data yang tersisa, tentukan label *cluster*-nya ke salah satu dari k *cluster*. Label *cluster* dari setiap data ditentukan dengan mencari pusat *cluster* dengan jarak yang paling dekat.

Langkah 3: Hitung pusat *cluster* yang baru berdasarkan kumpulan data saat ini, yaitu dengan menghitung rata-rata dari semua data di *cluster* yang sama.

Langkah 4: Tentukan label *cluster* dari sebanyak n data ke pusat *cluster* yang baru dengan jarak yang paling dekat.

Langkah 5: Jika tidak ada perubahan label *cluster* untuk semua data pada dua iterasi yang berurutan maka proses *clustering* selesai; jika masih ada perubahan maka ulangi langkah 3.

Gambar 2.2 Algoritma *k-means*

Algoritma *k-means* memiliki beberapa kelebihan diantaranya yaitu sederhana, mudah diimplementasikan, dan prosesnya cepat. Meskipun demikian, keluaran dari *k-means* sangat bergantung pada titik pusat (*centroid*) awal yang ditentukan secara acak. Oleh karena itu, dalam

aplikasi praktis *k-means* harus dijalankan beberapa kali dengan *centroid* awal yang berbeda-beda untuk menghasilkan *centroid* akhir yang dianggap paling bagus.

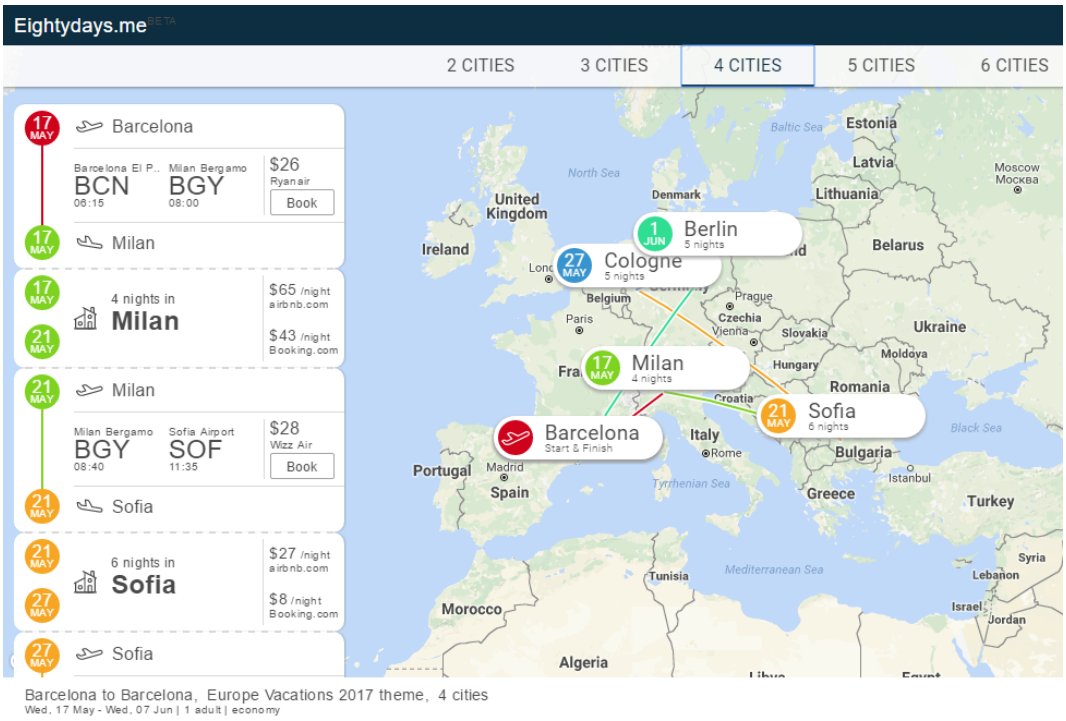
Studi kasus untuk algoritma *k-means* ini adalah pada *clustering* kualitas beras berdasarkan ciri fisik (Agustina, 2014). Perhitungan pada program klasterisasi kualitas beras dengan *k-means* diawali dengan menentukan jumlah *cluster*. Penelitian ini menggunakan 3 *cluster* untuk menentukan kualitas beras, dengan keterangan bahwa *cluster* 1 adalah beras kualitas buruk, *cluster* 2 adalah beras kualitas sedang, dan *cluster* 3 adalah beras kualitas baik. Setelah menentukan jumlah *cluster*, kemudian menentukan 3 pusat *cluster* awal, langkah selanjutnya adalah mengalokasikan data ke dalam *cluster*, kemudian menghitung jarak setiap data terhadap setiap pusat *cluster*. Suatu data akan menjadi anggota dari suatu *cluster* yang memiliki jarak terkecil dari pusat *cluster*-nya. Setelah itu menghitung pusat *cluster* baru sebagai acuan untuk iterasi berikutnya. Iterasi berhenti apabila posisi data tidak berubah, maka diperolehlah hasil akhirnya.

2.4 Aplikasi Pembuat *Itinerary* Wisata

Saat ini, telah banyak aplikasi yang dikembangkan untuk memudahkan para wisatawan. Kebanyakan diantaranya merupakan aplikasi yang memberikan rekomendasi dan *review* mengenai destinasi-destinasi yang menarik untuk dikunjungi, seperti aplikasi tripadvisor. Masih jarang ditemui aplikasi yang dapat digunakan dimana kasusnya pengguna telah menentukan destinasi-destinasi yang ingin dikunjungi dan mulai menyusun jadwal atau urutan kunjungan secara otomatis. Biasanya calon wisatawan harus menyusun sendiri jadwalnya dan baru kemudian mencatatnya di aplikasi-aplikasi yang dapat membantu menyimpan catatan *itinerary*.

Berdasarkan referensi yang ditemukan oleh penulis, terdapat sebuah aplikasi bernama Eightydays yang merupakan sebuah aplikasi yang dapat membantu wisatawan untuk merancang rencana perjalanan untuk mengunjungi beberapa kota di benua Eropa (Rizzo, 2017). Aplikasi ini dibuat untuk memberikan wisatawan beberapa pilihan yang tersedia ketika akan berwisata ke Eropa. Pengembang aplikasi menyadari bahwa untuk membuat rencana perjalanan seringkali menghabiskan waktu yang sangat lama jika dilakukan secara manual. Dengan menggunakan Eightydays, pengguna hanya perlu memasukkan kota dimana perjalanan akan diawali dan diakhiri. Kemudian pengguna juga harus memasukkan tanggal keberangkatan, durasi perjalanan dan banyaknya kota yang ingin dikunjungi. Aplikasi akan secara otomatis membuatkan *itinerary* beserta informasi penerbangan dan informasi akomodasi di setiap kota.

Contoh tampilan aplikasi Eightydays dapat dilihat pada Gambar 2.3. Pada contoh tersebut, pengguna hanya perlu melilih kota "Barcelona" sebagai tempat perjalanan akan diawali dan diakhiri, kemudian memasukkan tanggal keberangkatan dan kepulangan. Dan pada saat pengguna memilih untuk mengunjungi 4 kota dalam durasi waktu tersebut, maka aplikasi akan memberikan rekomendasi *itinerary* bagi pengguna yaitu Barcelona-Milan-Sofia-Cologne-Berlin-Barcelona lengkap dengan tanggal-tanggalnya dan perkiraan biayanya.



Gambar 2.3 Contoh tampilan aplikasi eightydays

Sistem yang dimiliki oleh Eightydays masih belum lengkap. Sistem ini cocok jika diterapkan untuk membuat *itinerary* antar kota yang jaraknya cukup jauh (dijangkau menggunakan pesawat). Padahal di satu kota sendiri wisatawan masih perlu untuk membuat *itinerary* kunjungan. Maka dari itu, perlu dikembangkan aplikasi tambahan yang dapat menyelesaikan persoalan ini, sehingga calon wisatawan tidak lagi menghabiskan banyak waktu untuk merancang *itinerary* yang efisien.