ANALISIS METODE HEURISTIK PENGOLAHAN DATA *TRAVELLING SALESMAN PROBLEM* TERHADAP JUMLAH TITIK

Abyan Hafizh Raihan[1)], Dr. Drs. Iman Firmansyah M.Sc.[2)]
Program Studi Teknik Industri, Fakultas Teknik, Universitas Pasundan
Email[1)] : mezcloth95@yahoo.com

ABSTRAK

Penelitian ini dimaksudkan untuk menentukan metode heuristik pengolahan data Travelling Salesman Problem yang efektif dalam jarak tempuh yang minimum dan efisien dalam waktu proses pengolahan data. Persoalan rute ini termasuk dalam kelas NP-hard dimana terdapat banyak alternatif rute yang meningkat secara ekponensial seiring banyaknya jumlah kota yang akan dikunjungi. Nearest Neighbour, Saving Algorithm, Nearest Insertion, Cheapest Insertion, dan Farthest Insertion merupakan metode heuristik yang dapat digunakan dalam pengolahan data Travelling Salesman Problem. Metode heuristik tersebut diterjemahkan kedalam bahasa pemrograman untuk dirancang sebuah perangkat lunak yang dapat membantu dalam pengolahan data disertakan fungsi yang dapat menampilkan waktu proses pengolahan data. Dari hasil pengolahan data sekunder dapat disimpulkan bahwa metode yang efektif dalam pengolahan data matriks jarak adalah metode Nearest Neighbour dan metode yang efisien dalam waktu proses pengolahan data adalah Saving Algorithm.

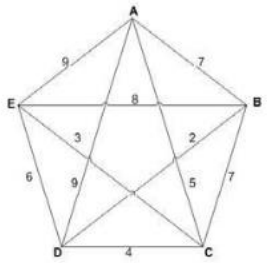
Kata kunci : Travelling Salesman Problem, Efektif, Efisien, Metode Heuristik

1. PENDAHULUAN

1.1 Latar Belakang

Travelling Salesman Problem merupakan masalah kombinasi optimasi dalam operasi penelitian dan ilmu komputer, dengan daftar kota-kota yang akan dikunjungi, cara ini sangat tepat untuk menemukan dengan waktu sesingkat mungkin setiap kota yang akan dikunjungi dengan total jarak terpendek. Untuk dapat menentukan urutan dari sejumlah kota yang harus dilalui, setiap kota harus dikunjungi satu kali dalam perjalannya, dan perjalanan tersebut harus berakhir pada kota keberangkatannya, dengan jarak antara kota satu dan lainnya sudah diketahui. Permasalahan penentuan rute ini termasuk ke dalam kelas *NP-hard* yang pada umumnya menggunakan pendekatan heuristik untuk mencari solusinya. Permasalahan utama adalah bagaimana seorang *salesman* dapat mengatur rute perjalananannya untuk mengunjungi sejumlah kota yang diketahui jarak satu kota dengan kota lainnya sehingga jarak yang ditempuh merupakan jarak minimum.

Untuk menyelesaikan persoalan tersebut, tidak mudah dilakukan karena terdapat ruang pencarian dari sekumpulan permutasi sejumlah kota. Maka *Travelling Salesman Problem* kemudian dikenal dengan persoalan *Non Polinomial*. Gambaran sederhana dari pengertian tersebut adalah sebagai berikut:



Sumber: Annies, 2002

Gambar 1 Posisi Kota-kota yang akan dilewati

Persoalan rute terpendek merupakan suatu jaringan pengarahan rute perjalanan di mana seseorang pengarah jalan ingin menentukan rute terpendek antara beberapa kota berdasarkan rute alternatif yang tersedia yang biasanya terdiri dari beberapa kota tujuan. Masalah ini umumnya menggunakan representasi *graph* untuk memodelkan persoalan yang diwakili sehingga lebih memudahkan penyelesaiannya. Masalahnya adalah bagaimana cara mengunjungi *vertek* pada *graph* dari *vertek* awal ke *vertek* akhir dengan bobot minimum, dalam hal ini bobot yang digunakan adalah jarak dan kota-kota yang dikunjungi diasumsikan sebagai *vertek* yang saling terhubung.

Masalah rute terpendek suatu masalah klasik yang sering dijumpai dalam kehidupan sehari-hari di berbagai sektor kehidupan, antara lain di bidang transportasi, komunikasi dan komputasi. Masalah ini menjadi masalah yang penting karena berkaitan

dengan masalah meminimumkan jarak dan efisiensi waktu yang dibutuhkan. Masalah rute terpendek untuk semua pasangan *vertek* adalah masalah menentukan lintasan terpendek untuk setiap *vertek* yang ada, dengan mengoptimalkan fungsi tujuan (objektif) tertentu.

Permasalah rute merupakan permasalahan yang cukup sederhana namun memiliki perhitungan yang kompleks berdasarkan jumlah tujuan yang akan dirancang sebuah rute, oleh karena itu untuk mempersingkat pengolahan data permasalahan rute tersebut diperlukan sebuah perangkat lunak dengan tujuan mempercepat waktu proses pengolahan data menimbang banyaknya alternatif yang dapat terjadi dalam sebuah penentuan rute.

1.2 Perumusan Masalah

Untuk mendapatkan hasil alternatif rute dengan jarak minimum pada *Travelling Salesman Problem*, pengolahan data menggunakan metode heuristik untuk penelitian ini akan diteliti beberapa metode heuristik untuk mengetahui metode mana yang mempunyai rute alternatif dengan jarak paling minimum.

Berdasarkan permasalahan diatas, penelitian ini dapat dirumuskan untuk membantu dalam pemecahan masalah adalah sebagai berikut:

> Diantara metode heuristik yang diteliti, metode mana yang efektif dan efisien dalam pengolahan data jarak dalam *Travelling Salesman Problem* dengan kriteria total jarak tempuh dan waktu proses pengolahan data yang minimum?

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah:

Mengolah data sekunder untuk menentukan sejumlah metode heuristik *Travelling Salesman Problem* yang efektif dalam hasil pengolahan data dan waktu proses pengolahan data, oleh karena itu dirancang perangkat lunak untuk dapat mengolah data tersebut.

1.4 Batasan Penelitian

Untuk mempermudah dan memfokuskan permasalahan dari penelitian ini, ruang lingkup pembahasan penelitian ini dibatasi oleh:

1. Penelitian ini dibatasi dengan perancangan perangkat lunak berbasis *web* namun dalam penelitian ini perangkat lunak dirancang secara lokal.
2. Data sekunder yang digunakan adalah data yang berasal dari *TSPLib* dengan maksimum jumlah titik adalah 100 titik.

2. LANDASAN TEORI

2.1 Riset Operasional

Riset operasi adalah suatu aplikasi dari berbagai metode ilmiah untuk tujuan penguraian terhadap masalah-masalah yang kompleks yang muncul dalam pengarahan dan pengelolaan dari suatu sistem besar (manusia, mesin-mesin, bahan-bahan, dan uang) dalam bidang perindustrian, bisnis, pemerintahan, dan pertahanan.

2.1.2 Tahapan Studi Riset Operasional

Kegiatan yang dilakukan pada tahap pertama terdiri dari penentuan tujuan optimasi, identifikasi alternatif keputusan dan sumber daya yang membatasi kegiatan atau aktifitas untuk mencapai tujuan. Merumuskan atau mendefinisikan persoalan yang akan dipecahkan sesuai dengan tujuan yang akan dicapai berdasarkan keadaan objektif. Biasanya harus memperhatikan tiga hal yaitu :

1. Uraian yang tepat mengenai tujuan yang akan dicapai.
2. Identifikasi dari pada adanya alternatif dalam keputusan yang menyangkut suatu sistem.
3. Mengenali adanya pembatasan-pembatasan (*limitation*, *restriction* dan juga persyaratan-persyaratan yang diperlukan sistem yang bersangkutan dengan pemecahan persoalan).

2.1.3 Optimasi

Brogran (1991, 501) menyatakan bahwa optimasi adalah proses untuk mencapai hasil yang ideal atau optimal (nilai efektif yang dicapai). Optimasi secara intuisi berarti melakukan pekerjaan dengan cara terbaik. Sedangkan menurut Licker (2003, 170), optimasi berasal dari kata bahasa Inggris *optimization* memiliki arti memaksimumkan atau meminimumkan sebuah fungsi yang diberikan untuk beberapa macam kendala.

Optimasi adalah proses pencarian solusi yang terbaik, tidak selalu keuntungan paling tinggi yang bisa dicapai jika tujuan pengoptimalan adalah memaksimumkan keuntungan, atau tidak selalu biaya paling kecil yang bisa ditekan jika tujuan pengoptimalan adalah meminimumkan biaya. Tiga elemen permasalahan optimasi yang harus diidentifikasi, yaitu tujuan, alternatif keputusan dan sumber daya yang membatasi. Tujuan bisa berbentuk maksimisasi atau minimisasi. Bentuk maksimisasi

digunakan jika tujuan pengoptimalan berhubungan dengan keuntungan, penerimaan dan sejenisnya. Sedangkan bentuk minimisasi akan dipilih jika tujuan pengoptimalan berhubungan dengan biaya, waktu, jarak dan sejenisnya.

2.2 Distribusi

Distribusi meliputi semua aspek dalam pengiriman barang kepada agen. Sebenarnya, distribusi merupakan bagian dari material handling, karena material *handling* merupakan perpindahan material pada setiap saat dan setiap titik. Ada beberapa permasalahan yang biasa dihadapi dalam distribusi berkaitan dengan optimasi jaringan distribusi adalah:

1. Titik Depot
   Titik depot sangat menentukan kelancaran pendistribusian barang, sehingga barang dapat sampai pada agen tepat pada waktunya.
2. Penentuan rute dan jadwal pengiriman

Secara umum permasalahan penjadwalan dan penentuan rute pengiriman memiliki beberapa tujuan yang ingin dicapai seperti tujuan untuk meminimumkan biaya pengiriman, meminimumkan waktu atau meminimumkan jarak tempuh. Salah satu dari tujuan tersebut bisa menjadi fungsi tujuan (*objective function*) dan yang lainnya menjadi kendala (*constraint*). Misalnya, fungsi tujuannya adalah meminimumkan biaya pendistribusian, namun ada kendala time window dan kendala maksimum jarak tempuh tiap kendaraan, disamping kendala lain seperti kapasitas atau kendala lainnya. Pada penulisan skripsi ini, manajemen distribusi merupakan pengelolaan terhadap kegiatan untuk memindahkan surat kabar dari suatu depot ke sejumlah agen dimana proses pemindahan tersebut akan membentuk atau menghasilkan rute distribusi yang dibatasi oleh kapasitas kendaraan.

2.3 Algoritma

Menurut Thomas H. Cormen (2009:5), Algoritma adalah prosedur komputasi yang mengambil beberapa nilai atau kumpulan nilai sebagai input kemudian di proses sebagai output sehingga algoritma merupakan urutan langkah komputasi yang mengubah input menjadi output. Algoritma merupakan cara yang dapat ditempuh oleh komputer dalam mencapai suatu tujuan, terdiri atas langkah-langkah yang terdefinisi dengan baik, menerima input, melakukan proses, dan menghasilkan output. Meskipun tidak selalu, biasanya sebuah algoritma memiliki sifat bisa dihitung (*computable*) atau bisa dihitung. Sebuah algoritma dikatakan benar (*correct*), jika algoritma tersebut berhasil mengeluarkan output yang benar untuk semua kemungkinan input.

2.4 *Travelling Salesman Problem*

Travelling Salesman Problem secara umum merupakan masalah optimasi yang kompleks dimana dengan bertambahnya variabel secara linear maka waktu yang dibutuhkan untuk memecahkan masalah itu bertambah secara eksponensial. Permasalahan *Travelling Salesman Problem* ada 2 macam, yaitu *Symmetric Travelling Salesman Problem* dan *Assymmetric Travelling Salesman Problem*. Jumlah kombinasi perjalanan yang dihasilkan untuk kasus symmetric *Travelling Salesman Problem* adalah (n-1)!/2 dan untuk *Assymetric Travelling Salesman Problem* jumlah kombinasi adalah (n-1)!, dan n merupakan banyaknya kota yang akan dibuat sebuah rute. Tabel dibawah ini menunjukan banyaknya jumlah kombinasi pada sejumlah kota. Dengan melihat data tersebut, maka permasalahan *Travelling Salesman Problem* unbtuk jumlah kota yang bersar adalah permasalahan yang kompleks dan sulit untuk dipecahkan. Jadi, untuk memecahkan *Travelling Salesman Problem* diperlukan algoritman yang pada intinya adalah mengurangi pengecekan terhadap semua kombinasi yang mungkin.

Tabel 1. Tabel Jumlah Kombinasi

Jumlah Kota (n)	**Jumlah Kombinasi *Symmetric* (n-1)!/2**	**Jumlah Kombinasi *Assymetric* (n-1)!**
5	12	24
6	60	120
7	360	720
8	2520	5040
9	20160	40320
10	181440	362880

2.4.1 *Nearest Neighbour Method*

Metode *Nearest Neighbour* merupakan metode paling sederhana untuk menyelesaikan masalah *Travelling Salesman Problem*. Pilihlah salah satu node yang mewakili suatu kota atau lokasi awal. Selanjutnya, pilih node tujuan atau kota yang akan dikunjungi berikutnya, dengan pertimbangan hanya memilih kota yang memiliki jarak terdekat dengan kota yang sebelumnya dikunjungi. Kemudian, setelah seluruh kota dikunjungi atau seluruh nodes telah terhubung, maka tutup rute perjalanan dengan kembali ke kota asal (node asal). Secara umum langkah-langkah dari metode ini adalah sebagai berikut :

1. Langkah 0 : Inisialisasi.
2. Langkah 1 : Pilih kota yang akan selanjutnya dikunjungi berdasarkan jarak terdekat dari kota asalnya.
3. Langkah 2: Tambahkan pada urutan rute berikutnya.
4. Langkah 3 : Jika semua kota telah terhubung, maka kota terakhir dalam rute akan di hubungkan ke kota awal atau depot.

2.4.2 *Method of Clarke and Wright: The Saving Algorithm*

Metode ini digunakan untuk meminimalkan jarak atau waktu atau biaya pengiriman dengan mempertimbangkan hambatan yang ada. Metode ini dipublikasikan sebagai suatu algoritma yang digunakan sebagai solusi untuk permasalahan rute kendaraan dimana sekumpulan rute pada setiap langkah ditukar untuk mendapatkan sekumpulan rute yang lebih baik, dan metode ini digunakan untuk mengatasi permasalahan yang cukup besar, dalam hal ini adalah jumlah rute yang banyak.

Berikut ini merupakan langkah-langkah dari metode *Clarke and Wright Savings Heuristric* Menurut Chopra (2010,380) adalah sebagai berikut:

1. Mengidentifikasi matriks jarak

Matriks jarak merupakan jarak lokasi satu dengan lokasi lainnya yang akan dikunjungi oleh kendaraan. Jarak yang diketahui akan merngambarkan biaya yang dikeluarkan dalam melakukan transportasi di antara dua lokasi yang berbeda. Cara untuk menghitung jarak dari setiap lokasi dapat dilakukan dengan beberapa cara. Contoh salah satu cara adalah dengan mengetahui waktu tempuh yang dibutuhkan oleh suatu kendaraan dari satu lokasi ke lokasi lainnya. Dengan mengasumsikan rata-rata kecepatan yang digunakan, maka jarak akan diketahui dengan rumus:

$$D = v\ x\ t$$

dimana

D = Jarak antara dua lokasi yang berbeda (km)

V = Kecepatan rata-rata kendaraan (km/jam)

t = Waktu tempuh kendaraan (jam)

2. Mengidetifikasi *saving* matriks

Saving matriks merupakan penghematan suatu kendaraan mengunjungi beberapa lokasi secara bersamaan dibandingkan dengan mengunjungi satu per satu lokasi. Berikut merupakan gambaran *Saving* matriks S(x,y) :

Tabel 2.Gambaran Proses Metode *Saving*

Pabrik → Konsumen x → Pabrik
Dan
Pabrik → Konsumen y → Pabrik
Menjadi
Pabrik → Konsumen x → Konsumen y → Pabrik

Pada gambaran di atas ini, dapat dilhiat rute yang baru akan menghemat waktu dan jarak tempuh dari kendaraan dalam mendistribusikan pesanan konsumen. Nilai dari *saving* matriks dirumuskan sebagai berikut :

$$S_{ij} = C_{ki} + C_{kj} - C_{ij}$$

Dimana :

S : Merupakan *Saving*

C : Merupakan jarak

k : Kota asal

i : Kota tujuan pertama

j : Kota tujuan kedua

2.4.3 *Method of Nearest Insertion*

Pada *Method of Nearest Insertion* yang dilakukan pertama kali adalah menentukan titik untuk disisipkan dengan mencari titik bebas yang paling dekat dengan suatu titik pada tur. Algoritma pada dasarnya melakukan sebuah operasi minimum pada jarak dari titik bebas untuk suatu titik pada tur. Selanjutnya dengan algoritma ini, ditentukan link terbaik untuk menyisipkan titik ini. Berikut merupakan langkah cara penyelesaian pada metode ini :

1. Pilih sebuah kota sebagai kota asal.
2. Pilih kota terdekat dari kota asal pertama.
3. Pilih kota terdekat dari kota asal kedua.
4. Ulangi hingga tersisa 1 kota yang tidak terhubung.
5. Masukan kota yang belum terpilih kedalam rute sementara dengan rumus:
   $F = C_{ik} + C_{kj} - C_{ij}$
   C: Jarak
   k: Kota yang belum terpilih
   i : Kota pertama dalam *insertion*
   j : Kota kedua dalam *insertion*
6. Hubungkan kota terakhir pada tur ke kota asal.

2.4.4 *Method of Cheapest Insertion*

Pada *Method of Cheapest Insertion* yang dilakukan pertama kali adalah menentukan setiap

titik yang masih tersisa dan bebas atau titik yang belum dikunjungi yang menghasilkan link optimal untuk menyisipkan titik ini. Penalti penyisipan adalah jumlah jarak ke titik bebas dikurangi dari link yang akan dihapus. Pada *Cheapest Insertion* kemudian dilakukan pemilihan titik untuk disisipkan sebagai titik penyisipan dengan penalti minimum. Berikut merupakan langkah cara penyelesaian pada metode ini:

1. Pilih sebuah kota sebagai kota asal.
2. Pilih kota terdekat dari kota asal.
3. Masukan kota yang belum terpilih kedalam rute sementara dengan rumus:
   $F = C_{ik} + C_{kj} – C_{ij}$
   C : jarak
   k : kota yang belum terpilih
   i : Kota pertama dalam insertion
   j : kota kedua dalam insertion
4. Ulangi tahap ketiga hingga seluruh kota terhubung.
5. Kota terakhir dalam rute dihubungkan ke kota asal pertama.

2.4.5 *Method of Farthest Insertion*

Pada *Method of Farthest Insertion* yang dilakukan pertama kali adalah menentukan setiap titik bebas yang memiliki jarak ke manapun pada tur terkecil. Kemudian masukkan titik bebas yang memiliki maksimum jarak terkecil ke titik pada tur. Berikut merupakan langkah cara penyelesaian pada metode ini:

1. Pilih sebuah kota sebagai kota asal
2. Pilih kota kedua dengan jarak terjauh.
3. Pilih kota kedua dengan jarak terjauh dari kota yang terpilih sebelumnya, sisipkan kota tersebut.
4. Ulangi tahap 3 hingga tersisa 2 kota.
5. Pilih kota terdekat dalam tur sementara, sisipkan kota tersebut.
6. Masukan kota yang belum terpilih kedalam rute sementara dengan rumus:
   $F = C_{ik} + C_{kj} – C_{ij}$
   C : Jarak
   k : Kota yang belum terpilih
   i : Kota pertama dalam *insertion*
   j : Kota kedua dalam *insertion*

2.5 Bahasa Pemrograman *Ruby on Rails*

2.5.1 Sejarah *Ruby*

Ruby on *Rails* atau sering disingkat *Rails* merupakan sebuah *framework* aplikasi *web full-stack* yang ditulis menggunakan bahasa pemrograman *Ruby*. Aplikasi *web* adalah perangkat lunak aplikasi yang diakses menggunakan *web browser* melalui sebuah jaringan (internet). Sedangkan *framework* dapat dipandang sebagai fondasi dari aplikasi *web* yang menangani sebagian besar detail level-bawah yang bersifat repetitif dan membosankan dalam menuliskan kodenya, sehingga pengembang dapat fokus pada pembangunan fungsionalitas aplikasi. *Full-stack* menunjuk kepada luasnya fungsionalitas yang dapat diberikan oleh *framework Rails* mulai dari penanganan antarmuka pada klien sampai urusan *database* di *server*. Sedangkan *Ruby* adalah sebuah bahasa *scripting* berorientasi objek yang diciptakan pertama kali oleh Yukihiro Matsumoto pada awal 1990-an (Lenz, 2008:2).

2.5.2 Karakteristik *Rails*

Rails mempunyai karakteristik yang berbeda dibandingkan *web framework* lainnya, diantaranya:

1. Berbasis arsitektur *MVC (Model-View-Controller)* Pemisahan data, presentasi dan logika kontrol dalam 3 tempat yang terpisah yang lebih memudahkan dalam memelihara kode yang sedang dibuat dan mengontrol perpindahan data.
   a. Model
      Arsitektur model diatur oleh *ActiveRecord* yang merupakan salah satu lapisan dalam kode *Rails* yang menyediakan *ORM (Object-Relational Mapping)* antara *Rails* dan *database*.
   b. *View*
      Bagian *view* dari arsitektur menangani presentasi data, misalnya pada sebuah *web browser*.
   c. *Controller*
      Controller bersama-sama dengan *view* tergabung dalam *ActionPack*, bertugas mengambil masukkan pengguna dan merespon dengan meminta suatu operasi pada model dan mempersiapkan hasilnya untuk ditampilkan oleh bagian *view*.
2. *Rails* merupakan sebuah *framework* yang *full-stack*
3. *Script*
   Banyak *script Rails* tersedia yang memudahkan otomatisasi tugas-tugas pada siklus pengembangan yang dapat mengurangi beban kerja.
4. Validasi
   Rails memiliki metoda untuk memvalidasi apapun mulai dari pembuatan, penampilan, ukuran atau panjang sesuatu dan lain-lain.

5. *AJAX*
   Rails mendukung *AJAX* dimana sebuah *browser* dapat meng-*update* atau mengubah sebagian dari halaman *web* yang merupakan objek *XMLHTTPRequest* sehingga memungkinkan *update* dilakukan tanpa sepengetahuan dan persetujuan dari pengguna *browser (backgrounding).*
6. *Migration*
   Rails memiliki kemampuan untuk melakukan migrasi dari satu skema *database* ke yang lainnya dengan mudah hanya dengan melalui kode *Ruby* saja.
7. *Console*
   Rails mengijinkan pengembang untuk menguji aplikasi *Rails* secara *black-box* menggunakan perangkat *irb* dalam sebuah konsol.
8. *Environment* dan Pengujian
   Secara default, *Rails* menyediakan tiga *environment* paralel: *development*, *test* dan *production*. Demikian juga *Rails* menyediakan dukungan yang lengkap terhadap aktivitas pengujian.
9. *Rake*
   Perkakas pengembangan mirip make yang ditulis dalam bahasa *Ruby* yang berguna dalam menyelesaikan tugas-tugas yang berkaitan dengan proyek seperti pembuatan/penghapusan *database* maupun pengubahan skema *database* atau pemrosesan *file-file* dalam jumlah banyak.

2.6 Basis Data *MySQL*

2.6.1 Definisi Basis Data *MySQL*

Menurut Yenie Kustiyahningsih (2010, 145) Basis data adalah sekumpulan informasi yang diatur agar mudah dicari. Dalam arti umum basis data adalah sekumpulan data yang diproses degan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan tepat, yang dapat digambarkan sebagai aktivitas dari satu atau lebih organisasi yang berelasi.

Menurut Arbie (2004, 5), *MySQL* adalah sebuah sistem manajemen *database* relasi (*relational database management system*) yang bersifat *open source*. *MySQL* merupakan suatu *database*. *MySQL* dapat juga dikatakan sebagai *database* yang sangat cocok bila dipadukan dengan *PHP*. Secara umum, *database* berfungsi sebagai tempat atau wadah untuk menyimpan, mengklasifikasikan data secara prefosional. *MySQL* bekerja menggunakan *SQL* Language (*Structure Query Language*). Itu dapat diartikan bahwa *MySQL* merupakan standar penggunaan *database* di dunia untuk pengolahan data.

Pengertian *MySQL* menurut *MySQL* manual adalah sebuah *open source software database SQL* (Search Query Language) yang menangani *system* manajemen *database* dan sistem manajemen *database* relational. *MySQL* adalah *open source software* yang dibuat oleh sebuah perusahaan Swedia yaitu *MySQLAB*. *MySQL* mempunyai fitur-fitur yang sangat mudah dipelajari bagi para penggunanya dan dikembangkan untuk menangani *database* yang besar dengan waktu yang lebih singkat. Kecepatan, konektivitas dan keamanannya yang lebih baik membuat *MySQL* sangat dibutuhkan untuk mengakses *database* di internet.

MySQL termasuk jenis *RDBMS* (*Relational Database Management System*). Sedangkan *RDBMS* sendiri akam lebih banyak mengenal istilah seperti tabel, baris, dan kolom digunakan dalam perintah-perintah di *MySQL*. *MySQL* merupakan sebuah basis data yang mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom. Di dalam *PHP* telah menyediakan fungsi untuk koneksi ke basis data dengan sejumlah fungsi untuk pengaturan baik menghubungkan maupun memutuskan koneksi dengan *server database MySQL* sebagai sarana untuk mengumpulkan informasi.

Pada umumnya, perintah yang paling sering digunakan dalam *MySQL* adalah *select* (mengambil), *insert* (menambah), *update* (mengubah), dan *delete* (menghapus). Selain itu, *SQL* juga menyediakan perintah untuk membuat *database*, *field*, ataupun *index* guna menambah atau menghapus data.

2.6.2 Keunggulan *MySQL*

Alasan yang mengacu menggunakan *MySQL* adalah *MySQL* merupakan *database* yang mampu berjalan di semua sistem operasi. Selain itu, sangat mudah sekali untuk dipelajari dan sepertinya hosting *server* juga banyak sekali mengadopsi *MySQL* sebagai standar *database*. Dan tentunya juga bersifat gratis atau *free*.

Saat ini *MySQL* juga tidak hanya gratis, semenjak *MySQL* dibeli oleh SUN, *MySQL* tidak lagi menikmati fitur-fitur barunya, karena telah dibatasi penggunanya. Fitu-fitur tersebut hanya bisa didapat jika membeli lisensinya. Berikut beberapa kelebihan yang dimiliki oleh *MySQL*:

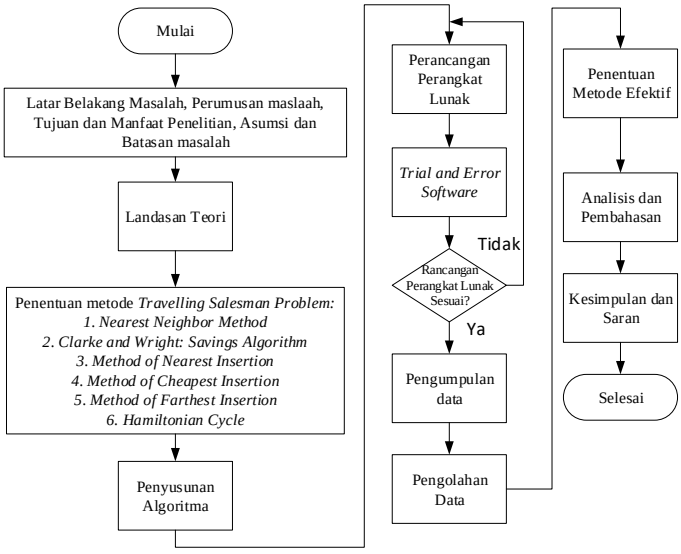
1. Bersifat *open source*, yang memiliki kemampuan untuk dapat dikembangkan lagi.

2. Menggunakan bahasa *SQL*(*Structure Query Language*), yang merupakan standar bahasa dunia dalam pengolahan data.
3. *Super perfomance* dan *reliable*, tidak bisa diragukan, proses *database*-nya sangat cepat dan stabil.
4. Sangat mudah dipelajari.
5. Memiliki dukungan *support (group)* pengguna *MySQL*.
6. Mampu lintas *platform*, dapat berjalan di berbagai sistem operasi.
7. *Multiuser*, dimana *MySQL* dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami konflik.

3. KERANGKA PEMECAHAN MASALAH

Diagram alir dari pemecahan masalah ditunjukkan pada Gambar 2.

Pada tahapan penelitian yang dilakukan terlihat bahwa proses pengumpulan data sekunder berasal dari data sekunder *TSPLib* yang berupa data titik koordinat kemudian data tersebuh diolah menjadi sebuah matriks data lalu diolah lebih lanjut dengan metode heuristik *Travelling Salesman Problem* untuk mengetahui ruta alternatif pada setiap metode.



Gambar 2 Kerangka Pemecahan Masalah

4. PENGUMPULAN DAN PENGOLAHAN DATA

4.1 Pengumpulan Data

Pengumpulan data pada penelitian ini menggunakan data sekunder berasal dari kumpulan tes data *TSPLib*, untuk penelitian ini menggunakan 8 data sekunder dengan jumlah kota atau titik yang berbeda disetiap data sekunder yang akan diolah. Data-data sekunder tersebut adalah sebagai berikut:

Tabel 3 Data sekunder yang digunakan

Nama Data	Jumlah Titik/Kota
Burma14	14
GR17	17
Ulysess22	22
Bay29	29
Swiss42	42
Berlin52	52
St70	70
Rat99	99

4.2 Pengolahan Data

4.2.1 Konversi Data Koordinat Menjadi Data Matriks Jarak

4.2.1.1 Konversi Data Sekunder Burma14

Data koordinat yang diperoleh untuk data sekunder Burma14 adalah sebagai berikut:

Tabel 4 Tabel Koordinat Data Sekunder Burma14

Titik Ke -	Koordinat X	Koordinat Y
1	16,47	96,1
2	16,47	94,44
3	20,09	92,54
4	22,39	93,37
5	25,23	97,24
6	22	96,05
7	20,47	97,02
8	17,2	96,29
9	16,3	97,38
10	14,05	98,12
11	16,53	97,38
12	21,52	95,59
13	19,41	97,13
14	20,09	94,55

Data-data koordinat tersebut akan diolah menjadi matriks jarak secara *Euclidean* dengan rumus menghitung jarak antar 2 titik koordinat adalah sebagai berikut:

$$Jarak\ Titik\ 1\ Ke\ Titik\ 2 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Jarak antara titik awal dan titik selanjutnya dapat menyesuaikan berdasarkan rumus perhitungan tersebut. Berikut merupakan contoh perhitungan jarak antara 2 titik pada data sekunder Burma14:

Jarak Titik 1 ke Titik 2 = $\sqrt{(16{,}47-16{,}47)^2+(94{,}44-96{,}1)^2}$

$= \sqrt{0^2+(-1{,}66)^2}$

$= \sqrt{2{,}7556}$

$= 1{,}66$

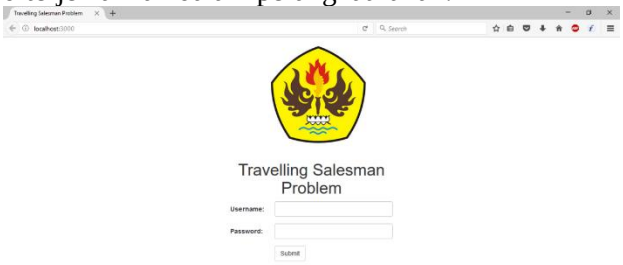
Tabel 5 Tabel Matriks Jarak Burma14

Titik Dari	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-	1,66	5,08	6,52	8,83	5,53	4,10	0,75	1,29	3,15	1,28	5,08	3,12	3,94
2	1,66	-	4,09	6,02	9,20	5,76	4,76	1,99	2,94	4,40	2,94	5,18	3,98	3,62
3	5,08	4,09	-	2,45	6,96	4,00	4,50	4,73	6,15	8,22	6,01	3,37	4,64	2,01
4	6,52	6,02	2,45	-	4,80	2,71	4,12	5,96	7,29	9,60	7,10	2,38	4,80	2,59
5	8,83	9,20	6,96	4,80	-	3,44	4,77	8,09	8,93	11,21	8,70	4,06	5,82	5,80
6	5,53	5,76	4,00	2,71	3,44	-	1,81	4,81	5,85	8,22	5,63	0,66	2,81	2,43
7	4,10	4,76	4,50	4,12	4,77	1,81	-	3,35	4,19	6,51	3,96	1,77	1,07	2,50
8	0,75	1,99	4,73	5,96	8,09	4,81	3,35	-	1,41	3,64	1,28	4,38	2,36	3,37
9	1,29	2,94	6,15	7,29	8,93	5,85	4,19	1,41	-	2,37	0,23	5,52	3,12	4,73
10	3,15	4,40	8,22	9,60	11,21	8,22	6,51	3,64	2,37	-	2,59	7,89	5,45	7,02
11	1,28	2,94	6,01	7,10	8,70	5,63	3,96	1,28	0,23	2,59	-	5,30	2,89	4,55
12	5,08	5,18	3,37	2,38	4,06	0,66	1,77	4,38	5,52	7,89	5,30	-	2,61	1,77
13	3,12	3,98	4,64	4,80	5,82	2,81	1,07	2,36	3,12	5,45	2,89	2,61	-	2,67
14	3,94	3,62	2,01	2,59	5,80	2,43	2,50	3,37	4,73	7,02	4,55	1,77	2,67	-

Dengan perlakuan yang sama untuk konversi titik koordinat menjadi data matriks jarak untuk data sekunder *TSPLib* Ulysess22, Berlin52, St70 dan Rat99

4.2.2 Implementasi Pengolahan Data Sekunder Pada Perangkat Lunak

Setelah data-data sekunder tersebut diolah lebih lanjut sehingga terkonversi menjadi data matriks jarak, data sekunder tersebut dapat diolah lebih lanjut dengan perangkat lunak untuk diproses menjadi sebuah rute. Data matriks jarak tersebut disimpan kedalam *Softfile Microsoft Excel* agar dapat diterjemahkan dalam perangkat lunak.



Gambar 3 Tampilan Awal Perangkat Lunak

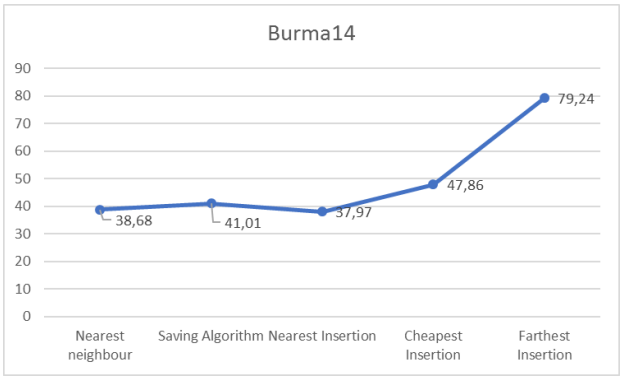
4.2.3 Rekapitulasi Hasil Pengolahan Data

Tabel 6 Rekapitulasi Hasil Pengolahan Data

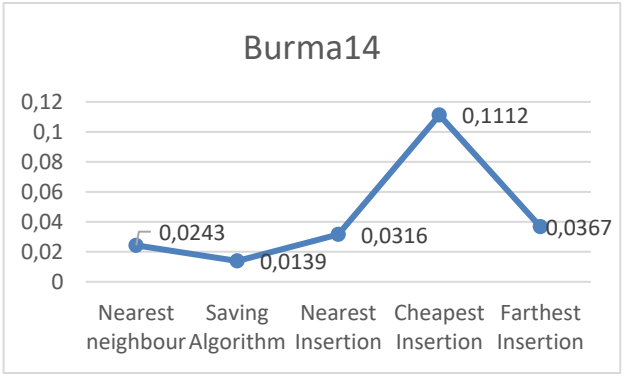
	Burma14		Gr17		Ulysess22		Bay29	
Metode	Jarak Tempuh	Waktu Proses	Jarak Tempuh	Waktu Proses	Jarak Tempuh	Waktu Proses	Jarak Tempuh	Waktu Proses
Nearest neighbour	38,68	0,0243	2187	0,0261	89,63	0,0355	2258	0,0417
Saving Algorithm	41,01	0,0139	3056	0,0135	90,9	0,0163	2881	0,022
Nearest Insertion	37,97	0,0316	2372	0,0377	82,21	0,0462	2315	0,0562
Cheapest Insertion	47,86	0,1112	2854	0,1623	90,96	0,275	2754	0,4576
Farthest Insertion	79,24	0,0367	5774	0,0405	236,37	0,0538	7941	0,0687

Tabel 7 Rekapitulasi Hasil Pengolahan Data

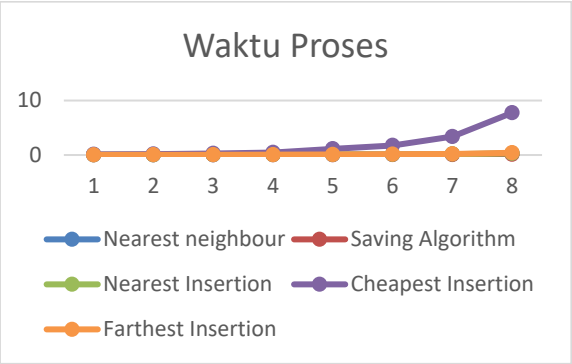
	Swiss48		Berlin52		St70		Rat99	
Metode	Jarak Tempuh	Waktu Proses	Jarak Tempuh	Waktu Proses	Jarak Tempuh	Waktu Proses	Jarak Tempuh	Waktu Proses
Nearest neighbour	1630	0,0631	8980,9	0,0891	805,51	0,1307	1564,72	0,258
Saving Algorithm	1586	0,0304	10166,59	0,0505	924,86	0,0581	1618,67	0,1174
Nearest Insertion	1640	0,0863	8982,02	0,113	815,55	0,1619	1512,84	0,2993
Cheapest Insertion	1740	1,1283	11920,53	1,7513	1072,32	3,3781	1656,98	7,7475
Farthest Insertion	6434	0,0941	37742,6	0,1464	5242,17	0,1925	12139,15	0,4115



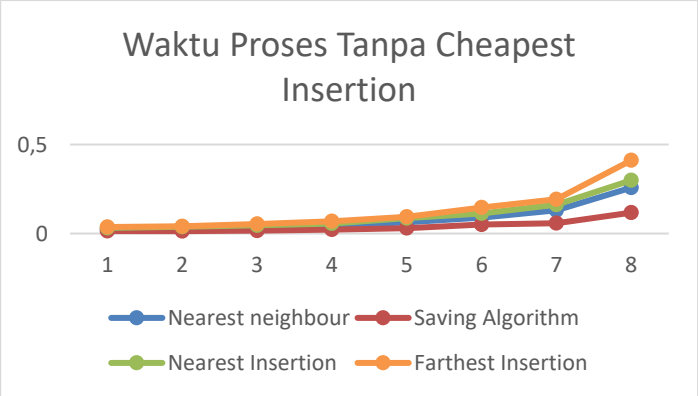
Gambar 4 Grafik Jarak Tempuh Burma14



Gambar 5 Grafik Waktu Proses Burma14



Gambar 6 Grafik Waktu Proses Keseluruhan



Gambar 7 Grafik Waktu Proses Keseluruhan Tanpa *Cheapest Insertion*

5. ANALISIS DAN PEMBAHASAN MASALAH

5.1 Analisis Metode Nearest Neighbour

Peningkatan waktu proses tersebut dipengaruhi oleh iterasi yang dilakukan oleh perangkat lunak dalam pengolahan data dengan menggunakan metode *Nearest Neighbour*, iterasi tersebut adalah pemilihan kota tujuan selanjutnya dengan membandingkan jarak antara masing-masing kota tujuan selanjutnya, iterasi tersebut diulang terus menerus hingga seluruh kota tujuan terhubung kemudian dilakukan perhitungan penjumlahan jarak dari rute alternatif yang telah terbentuk.

5.2 Analisis Metode Saving Algorithm

Peningkatan waktu proses tersebut disebabkan oleh perulangan perhitungan hasil penghematan jarak dari dua kota tujuan selanjutnya, perulangan tersebut dilakukan setiap perpindahan kota asal dalam rute sementara, perhitungan penghematan tersebut diulang berkali-kali sehingga muncul jumlah hasil *saving* lalu hasil tersebut diurutkan dari yang paling besar ke yang paling kecil kemudian dipilih dua kota tujuan dengan hasil *saving* terbesar, proses tersebut diulang hingga seluruh kota. Jika terdapat 1 kota tujuan akhir, maka tidak perlu dilakukan proses perhitungan *saving*, kota tersebut dimasukkan dalam akhir rute sementara dan akan dihubungkan dengan kota awal mula berasal. Setelah rute akhir terbentuk maka akan dilakukan proses perhitungan penjumlahan total jarak tempuh.

5.3 Analisis Metode Nearest Insertion

Peningkatan waktu proses tersebut disebabkan oleh perulangan yang dilakukan pada penentuan kota selanjutnya dengan menentukan jarak paling kecil dan waktu proses juga dipengaruhi oleh perhitungan dari nilai *Insertion*, perhitungan nilai tersebut mempengaruhi dikarenakan rute sementara yang terbentuk cukup banyak sehingga diperlukan perhitungan nilai *Insertion* yang berulang-ulang.

5.4 Analisis Metode Cheapest Insertion

Peningkatan waktu proses tersebut disebabkan oleh perulangan perhitungan nilai *Insertion*, perulangan tersebut juga berdampak pada penyusunan rute sementara dikarenakan kota tujuan akan masuk diantara 2 kota sementara, semakin banyak kota yang akan diproses akan meningkatkan waktu proses pada perhitungan metode *Cheapest Insertion* ini.

5.5 Analisis Metode Farthest Insertion

Peningkatan waktu proses tersebut disebabkan oleh perulangan yang dilakukan pada penentuan kota selanjutnya dengan menentukan jarak paling besar dan waktu proses juga dipengaruhi oleh perhitungan dari nilai *Insertion*, perhitungan nilai tersebut mempengaruhi dikarenakan rute sementara yang terbentuk cukup banyak sehingga diperlukan perhitungan nilai *Insertion* yang berulang-ulang.

6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil dari penelitian pengolahan data *Travelling Salesman Problem* dengan beberapa metode heuristik dapat disimpulkan bahwa metode heuristik dengan kriteria hasil pengolahan data dan waktu proses yang minimum adalah

a. Metode heuristik dengan kriteria pengolahan data yang paling minimum dalam *Travelling Salesman Problem* adalah metode *Nearest Neighbour* karena pada delapan kali percobaan pengolahan data, metode tersebut dapat menyelesaikan lima data sekunder dengan hasil pengolahan data yang minimum diantara metode lainnya.

b. Metode heuristik dengan kriteria waktu proses yang minimum adalah metode *Saving Algirthm* karena pada penelitian ini metode tersebut memiliki waktu proses yang minimum dalam pengolahan data sekunder.

6.2 Saran

Setelah disimpulkan maka peneliti dapat memberikan saran yang mungkin dapat menjadi masukan bagi penelitian selanjutnya diantara sebagai berikut:

1. Perangkat lunak yang dirancang memiliki waktu konversi data yang meningkat seiring dengan kenaikan jumlah titik dan table matriks yang telah dibuat sehingga diharapkan perancangan perangkat lunak dapat meminimalisir waktu pengunggahan data.
2. Jika perangkat lunak mengolah data dengan jumlah titik yang banyak, tampilan alternatif rute akan sangat panjang, disarankan untuk memotong alternatif rute tersebut menjadi beberapa baris sehingga dapat dilihat secara keseluruhan dengan tidak menggeser halaman tampilan perangkat lunak.
3. Perngakat lunak dirancang secara *web* namun dapat dikembangkan dengan memberikan nama *domain* pada perangkat lunak tersebut sehingga pengguna dapat mengakses dimana saja.

DAFTAR PUSTAKA

1. Arbie, 2004, *Manajemen Database dengan MySQL,* Andi Publisher, Yogyakarta
2. Brogran, W. L., 1991, *Modern Control Theory,* Prentice Hall Inc., New Jersey
3. Chopra, S., Meindl, P., 2010, *Supply Chain Management: Strategy, Plainning and Operation: Fourth Edition,* Pearson Education Inc., New Jersey
4. Cormen, Thomas H., 2009, *Introduction to Algorithms,* The MIT Press, Cambridge
5. Kustiyahningsih, Yenie, 2010, *Pemrograman Basis Data Berbasis Web Menggunakan PHP dan MySQL,* Graha Ilmu, Yogyakarta
6. Lenz, Patrick, 2008, *Simply Rails 2,* Sitepoint, Melbourne
7. Licker, M. D., 2003, *Dictionary of Mathematics: Second Edition,* McGraw-Hill, New York
8. Susanta, B., 1994, *Program Linear*, B. Publisher, Yogyakarta
9. Taha, Hamdy A., 1971 , *Operations Research An Introduction: Third Edition,* MacMillan Publishing Co. Inc., New York
10. Wijaya, Andi, 2013, *Pengantar Riset Operasi,* Mitra Wacana Media, Bekasi