PENERAPAN ALGORITMA *RECURSIVE BEST FIRST SEARCH* DALAM PENYELESAIAN *TRAVELING SALESMAN PROBLEM* DI PT. BINTANG SERVICE MANAGEMENT

Skripsi
disajikan sebagai salah satu syarat
untuk memperoleh gelar Sarjana Sains
Program Studi Matematika

oleh
FAOZI
4111411020

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SEMARANG
2016

PERNYATAAN

Saya menyatakan bahwa skripsi ini bebas plagiat, dan apabila di kemudian hari terbukti terdapat plagiat dalam skripsi ini, maka saya bersedia menerima sanksi sesuai ketentuan peraturan perundang-undangan.

Saya menyatakan bahwa dalam isi skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dirujuk dalam skripsi ini dan disebutkan dalam daftar pustaka.

Semarang, November 2016

Faozi
NIM 4111411020

PENGESAHAN

Skripsi yang berjudul

Penerapan Algoritma *Recursive Best First Search* (RBFS) dalam Penyelesaian *Traveling Salesman Problem* (TSP) di PT. Bintang Service Management

disusun oleh :

Faozi

4111411020

telah dipertahankan di hadapan sidang Panitia Ujian Skripsi Fakultas MIPA, Universitas Negeri Semarang pada :

Hari : Rabu

Tanggal : 9 November 2016

Panitia

Ketua

Prof. Dr. Zaenuri, S.E., M.Si. Akt.
NIP. 196412231988031001

Sekretaris

Drs. Arief Agoestanto, M.Si.
NIP. 196807221993031005

Ketua Penguji

Dr. Mulyono, M.Si.
NIP. 197009021997021001

Anggota Penguji
Pembimbing Utama

Drs. Amin Suyitno, M.Pd.
NIP. 195206041976121001

Anggota Penguji
Pembimbing Pendamping

Riza Arifudin, S.Pd., M.Cs.
NIP. 198005252005011001

MOTTO DAN PERSEMBAHAN

MOTTO

قُلْ إِنَّ صَلاتِي وَنُسُكِي وَمَحْيَايَ وَمَمَاتِي لِلهِ رَبِّ الْعَالَمِينَ

Katakanlah! Sesungguhnya shalatku, ibadahku, hidupku dan matiku hanya untuk Allah tuhan semesta alam (QS. Al Ana'm:162)

وَمَا خَلَقْتُ الْجِنَّ وَالْإِنسَ إِلَّا لِيَعْبُدُونِ

Dan tidak Ku ciptakan Jin dan Manusia melainkan untuk menyembah (beribadah) kepada Ku (QS. Al Dhariyat:65).

Skripsi ini aku persembahkan untuk :

1. *Orangtuaku tercinta*
2. *Kakakku dan keluarga besarku*
3. *Teman-teman Matematika 2011 UNNES*
4. *Semua sahabatku*
5. *Semua pihak yang telah menginspirasi, memotivasi dan membantuku dalam karya ini*
6. *Almamaterku.*

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penulisan skripsi yang berjudul "Penerapan Algoritma *Recursive Best First Search* (RBFS) dalam Penyelesaian *Traveling Salesman Problem* (TSP) di PT. Bintang Service Management".

Penulisan skripsi ini dapat terselesaikan karena adanya bimbingan, bantuan, dan dukungan dari berbagai pihak baik secara langsung maupun tidak langsung. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Prof. Dr. Fathur Rokhman, M.Hum., Rektor Universitas Negeri Semarang.
2. Prof. Dr. Zaenuri, S.E., M.Si., Akt., Dekan FMIPA Universitas Negeri Semarang.
3. Drs. Arief Agoestanto, M.Si., Ketua Jurusan Matematika FMIPA Universitas Negeri Semarang.
4. Drs. Mashuri, M.Si., Koordinator Program Studi Matematika FMIPA Universitas Negeri Semarang.
5. Drs. Amin Suyitno, M.Pd., Pembimbing pertama yang telah memberikan bimbingan, motivasi, dan pengarahan sehingga skripsi ini dapat terselesaikan.
6. Riza Arifudin, S.Pd., M.Cs., Pembimbing kedua yang telah memberikan bimbingan, motivasi, dan pengarahan hingga selesainya skripsi ini.
7. Dr. Mulyono, M.Si., Dosen penguji yang telah memberikan inspirasi, kritik, saran, dan motivasi kepada penulis, sehingga penulis dapat menyelesaikan skripsi.

8. Muhammad Kharis S.Si., M.Sc., Dosen Wali sejak Semester 1 hingga sekarang yang telah memberikan banyak motivasi, bimbingan dan arahan.
9. Staf Dosen Matematika Universitas Negeri Semarang yang telah membekali penulis dengan berbagai ilmu selama mengikut perkuliahan sampai akhir penulisan skripsi ini.
10. Staf Tata Usaha Universitas Negeri Semarang yang telah membantu penulis selama mengikuti perkuliahan dan penulisan skripsi ini.
11. Orangtua dan keluarga tercinta yang senantiasa mendoakan serta memberikan dukungan baik secara moral maupun spiritual.
12. Sahabat-sahabat penulis yang telah memberikan banyak motivasi, kritik, usulan yang menjadikan terselesaikannya penulisan skripsi ini.
13. Mahasiswa Matematika angkatan 2011 yang telah memberikan dorongan dan motivasi.
14. Semua pihak yang telah membantu terselesaikannya penulisan skripsi ini.

Penulis menyadari, bahwa masih banyak keterbatasan pengetahuan dan kemampuan yang penulis miliki. Penulis mengharapkan kritik dan saran untuk kebaikan pada penulisan-penulisan ilmiah yang lain. Semoga skripsi ini dapat berguna dan bermanfaat bagi pembaca.

Semarang, November 2016

Penulis

ABSTRAK

Faozi. 2016. *Penerapan Algoritma Recursive Best First Search (RBFS) Dalam Penyelesaian Traveling Salesman Problem (TSP) di PT. Bintang Service Management*. Skripsi, Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang. Pembimbing Utama Drs. Amin Suyitno, M.Pd. dan Pembimbing Pendamping Riza Arifudin, S.Pd., M.Cs.

Kata Kunci: Graf Hamilton, *Traveling Salesman Problem* (TSP), *Recursive Best First Search* (RBFS), *Hipertext Preprocessor* (PHP), LeafletJS.

Banyaknya perusahaan dalam industri, serta kondisi perekonomian saat ini telah menciptakan suatu persaingan yang ketat antar perusahaan. Persaingan tersebut mengakibatkan mengalihdayakan proses-proses yang bukan merupakan kompetensi utama (*core competence*) perusahaan tersebut ke pihak lain (*outsourcing*) agar perusahaan dapat memfokuskan pada kompetensi utama. Masalah yang terjadi pada pihak *outsourching* (PT. Bintang Service Management) dengan semakin banyaknya klien adalah semakin banyak pilihan rute perjalanan yang harus dilalui pihak *outsourcing* untuk melakukan pengadaan barang dan pengecekan barang di mana perusahaan berangkat dari kantor dan harus mengunjungi setiap perusahaan klien tepat satu kali, kemudian kembalilagi ke kantor. Masalah ini disebut *Traveling Salesman Problem* (TSP). *Traveling Salesman Problem* dapat divisualisasikan dalam bentuk graf Hamilton untuk diselesaikan dengan algoritma *Recursive Best First Search* (RBFS), sementara salah satu cara untuk mempermudah proses perhitungan dapat dibuat program menggunakan bahasa *Hipertext Preprocessor* (PHP).

Penelitian dilakukan dengan mengambil data klien dari perusahaan *otsourching* di wilayah Semarang, selanjutnya data dimodelkan dalam bentuk peta graf Hamilton menggunakan *library* LeafletJS dan data diproses menggunakan algoritma *Recursive Best First Search* sehingga diperoleh rute terpendek yang divisualisasikan dalam bentuk peta.hbjh Tujuan penelitian yaitu untuk mengetahui: (1) Penerapan algoritma *Recursive Best First Search* untuk mengatasi *Traveling Salesman Problem* di PT. Bintang Service Management; (2) Pembuatan program algoritma *Recursive Best First Search* dalam penyelesaian *Traveling Salesman Problem* di PT. Bintang Service Management menggunakan bahasa pemrograman *Hipertext Preprocessor*.

Hasil dari penelitian ini yaitu sebuah sikel Hamilton dengan bobot minimum yaitu Bintang Service Management – Semesta Bilingual School – My Kopi O – Hotel Grand Edge – City One Hotel – RS Panti Wilasa Citarum – Leko Gajah Mada – Dafam Hotel – 3 Durian – Kantor Imigrasi – Rumdenim – Goori Swalayan – Payon Amartha – Bintang Service Management. Rute tersebut dapat menjadi acuan pihak *outsourching* dalam penentuan rute perjalanan untuk melakukan pengadaan barang dan pengecekan barang ke pihak klien perusahaan secara lebih efektif dan efisien.

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR LAMPIRAN

BAB 1

PENDAHULUAN

1.2 Latar Belakang

Banyaknya perusahaan dalam industri, serta kondisi perekonomian saat ini telah menciptakan suatu persaingan yang ketat antar perusahaan. Persaingan dalam industri membuat setiap perusahaan semakin meningkatkan kinerja agar tujuan perusahaan dapat tetap tercapai. Menurut Hermuningsih (2013:128), tujuan utama perusahaan yang telah *go public* adalah meningkatkan kemakmuran pemilik atau para pemegang saham melalui peningkatan nilai perusahaan. Peningkatan nilai perusahaan dapat dilakukan dengan meningkatkan kinerja perusahaan, yaitu dengan mengalihdayakan atau menyerahkan proses-proses yang bukan merupakan kompetensi utama (*core competence*) perusahaan tersebut ke pihak lain agar perusahaan dapat memfokuskan pada kompetensi utama, aktivitas ini dikenal dengan istilah *outsourcing* (Iqbal & Dad, 2013:92).

Outsourcing dilakukan untuk memberikan respon atas perkembangan ekonomi secara global dan perkembangan teknologi yang begitu cepat serta berkembangnya persaingan yang bersifat global dan berlangsung sangat ketat. Menurut Çiçek & Özer (2011:113), o*utsourcing* telah terbukti dapat meningkatkan daya saing usaha secara signifikan, karena dengan melakukan *outsourcing*, perusahaan dapat lebih fokus dalam menjalankan aktivitas kompetensi utamanya, sehingga dapat mendukung kecepatan perusahaan dalam merespon tuntutan pasar.

PT. Bintang Service Management sebagai salah satu perusahaan *outsourcing* telah menangani lebih dari tiga puluh perusahaan yang tersebar di wilayah Bandung, Indramayu, Cirebon dan sebagian besar area Jawa Tengah khusunya Semarang, Purwodadi, Pati, Pemalang, Slawi, Tegal, dan Brebes. Banyaknya perusahaan yang ditangani PT. Bintang Service Management mengakibatkan semakin banyak pilihan rute perjalanan yang harus dilalui pihak *outsourcing* untuk memenuhi beberapa kebutuhan perusahaan klien khususnya kegiatan pengadaan barang, dan pengecekan barang yang rutin dilakukan setiap akhir bulan, sehingga waktu yang ditempuh juga semakin lama dan cenderung tidak efektif.

Permasalahan digambarkan di mana pihak perusahaan berangkat dari kantor PT. Bintang Service Management yaitu Jl. Kalipepe Baru No.9 Pudakpayung Semarang, harus mengunjungi setiap perusahaan klien di wilayah Semarang tepat satu kali dan kembali lagi ke kantor PT. Bintang Service Management. Permasalahan yang dihadapi tersebut dalam matematika disebut *Traveling Salesman Problem* (TSP), *Traveling Salesman Problem* merupakan salah satu masalah optimalisasi yaitu suatu permasalahan untuk menemukan sikel Hamilton yang memiliki total bobot sisi minimum (Hlaing & Khine, 2011: 405).

Menurut Kusrini & Istiyanto (2007:20), ada beberapa metode yang bisa menyelesaikan TSP, antara lain: *Linear Programming* (LP), Algoritma Genetik, *Nearest Neighbourhood Heuristi*c (NNH) dan *Cheapest Insertion Heuristic* (CIH). Menurut Sutojo, *et al*. (2011:83), teknik pencarian heuristik merupakan suatu

strategi untuk melakukan proses pencarian secara selektif dan dapat memandu proses pencarian yang memiliki kemungkinan sukses paling besar.

Algoritma Heuristik merupakan salah satu algoritma alternatif yang dapat digunakan untuk masalah tersebut, sebab prosesnya cepat dan memberikan hasil yang diinginkan. Algoritma yang umum digunakan dalam pencarian heuristik adalah *Generate and Test*, *Hil Caimbing*, A*, *Best First Search*, dan *Recursive Best First Search*.

Menurut Korf (1993), *Rekursif Best First Search* (RBFS) adalah algoritma *linear space* yang memperluas titik pencarian dalam terbaik-pertama bahkan dengan fungsi biaya *nonmonotonic*, dan menghasilkan titik yang lebih sedikit dari *Best first Search* dengan fungsi biaya monoton, sehingga menghasilkan *routing* dari graf yang memiliki nilai optimal.

Berdasarkan uraian yang telah dipaparkan di atas, penulis tertarik untuk menerapkan algoritma *Recursive Best First Search* dalam penyelesaian *Traveling Salesman Problem*, sehingga skripsi ini diberi judul "Penerapan Algoritma *Recursive Best First Search* (RBFS) dalam Penyelesaian *Traveling Salesman Problem* (TSP) di PT. Bintang Service Management".

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam penelitian ini adalah sebagai berikut.

1. Bagaimana penerapan algoritma *Recursive Best First Search* dalam penyelesaian *Traveling Salesman Problem* di PT. Bintang Service Management?
2. Bagaimana pembuatan program algoritma *Recursive Best First Search* dalam penyelesaian *Traveling Salesman Problem* di PT. Bintang Service Management dengan menggunakan bahasa pemrograman PHP?

1.3 Batasan Masalah

Pada penelitian ini diperlukan batasan-batasan agar tujuan penelitian dapat tercapai. Adapun batasan masalah yang dibahas pada penelitian ini adalah sebagai berikut.

1. Algoritma yang digunakan adalah *Recursive Best First Search*.
2. Objek penelitian ini dititikberatkan hanya pada klien Bintang Service Management di Semarang tahun 2016.
3. Pencarian rute terpendek tidak memperhatikan kepadatan lalu lintas, lampu lalu lintas, portal jalan, pengalihan jalan dan halangan sejenisnya.
4. Sisi pada graf yang dimaksudkan dibuat ruas garis lurus untuk mempermudah pembuatan program.
5. Graf yang menjadi inputan merupakan graf Hamilton.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah sebagai berikut.

1. Mengetahui dan memahami penerapan algoritma *Recursive Best First Search* dalam penyelesaian *Traveling Salesman Problem* di PT. Bintang Service Management.
2. Mengetahui dan memahami cara pembuatan program algoritma *Recursive Best First Search* dalam penyelesaian *Traveling Salesman Problem* di PT. Bintang Service Management menggunakan bahasa pemrograman *Hipertext Preprocessor* (PHP).

1.5 Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut.

1. Memberikan wawasan tentang algoritma *Recursive Best First Search* dalam Penyelesaian *Traveling Salesman Problem* di PT. Bintang Service Management.
2. Memberikan alternatif rute terpendek bagi pengambil keputusan PT. Bintang Service Management untuk mendapatkan rute perjalanan yang lebih efektif dan efisien.

BAB 2

TINJAUAN PUSTAKA

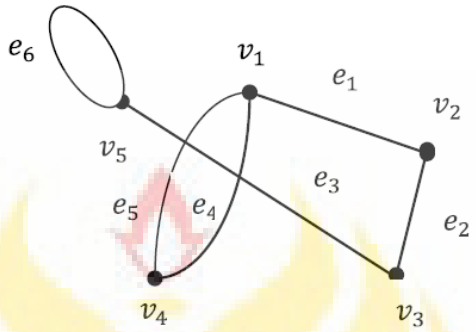
Beberapa peneliti telah melakukan penelitian tentang *Traveling Salesman Problem* (TSP) seperti penelitian TSP menggunakan algoritma semut yang dilakukan oleh Katiyar, *et al*. (2014). Penelitian TSP menggunakan algoritma genetika yang berjudul *Solving Travelling Salesman Problem Using Genetic Algorithm* oleh Gupta & Panwar (2013). Selain itu, penelitian serupa juga dillakukan di Indonesia seperti penelitina tentang pencarian rute terpendek pada PT. Jalur Nugraha Ekakurir (JNE) Semarang oleh Anitya, *et al*. (2013), penelitian algoritma fuzzy evolusi yang digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah optimasi dengan bantuan perangkat lunak Matlab oleh Anggit, *et al*. (2014).

2.1 Graf

2.1.1 Definisi Graf

Graf G adalah pasangan $(V(G), E(G))$, di mana $V(G)$ adalah himpunan berhingga titik-titik (*vertices*) yang tak kosong dan $E(G)$ adalah himpunan sisi (mungkin kosong), sedemikian sehingga setiap sisi (*edge*) e di $E(G)$ adalah pasangan tak berurutan dari titik-titik di $V(G)$. Himpunan titik dari G dinotasikan dengan $V(G)$, sedangkan himpunan sisi dinotasikan dengan $E(G)$ (Budayasa, 2007:1).

Jumlah titik dari graf G disebut order graf G yang dinotasikan dengan $|V(G)|$ sedangkan jumlah sisi dari graf disebut size graf G yang dinotasikan dengan $|E(G)|$. Gambar 2.1 merupakan contoh graf dengan 5 titik dan 6 sisi.



Gambar 2.1 Graf dengan 5 titik dan 6 sisi

Dalam sebuah graf, seperti terlihat pada Gambar 2.1, dimungkinkan adanya suatu sisi yang dikaitkan dengan pasangan (v_5, v_5). Sisi yang dua titik ujungnya sama disebut *loop* atau gelang. Pada Gambar 2.1, sisi e_6 merupakan sebuah *loop*. Dalam sebuah graf dimungkinkan adanya lebih dari satu sisi yang dikaitkan dengan sepasang titik. Pada Gambar 2.1, sisi e_4 dan e_5 sisi dikaitkan dengan pasangan titik (V_1, V_4). Menurut Sutarno, *et al*. (2003: 60), pasangan sisi semacam ini disebut sisi-sisi pararel atau sisi rangkap. Sebuah graf yang tidak memiliki *loop* dan tidak memiliki sisi rangkap disebut graf sederhana. Jika sebuah titik v_i merupakan titik ujung dari suatu sisi e_i, maka v_i dan disebut v_i saling berinsidensi e_i atau titik v_i terkait (*incident*) dengan sisi e_i (Sutarno, *et al*., 2003:60).

Contoh :

Pada Gambar 2.1 di atas, sisi e_1, e_4 dan e_5 adalah sisi-sisi yang terkait dengan titik v_1. Dua buah sisi yang tidak pararel disebut bertetangga (*adjacent*), bila kedua sisi tersebut terkait dengan titik yang sama. Selain itu, dua buah titik disebut bertetangga jika kedua titik tersebut merupakan titik-titik ujung dari sisi yang sama (Sutarno, *et al*., 2003: 60).

Contoh :

Dalam Gambar 2.1, e_1 dan e_2 merupakan dua sisi yang bertetangga. Titik v_3 dan v_5 adalah dua titik yang saling bertetangga, sedangkan titik v_2 dan v_4 merupakan dua titik yang tidak saling bertetangga.

2.1.2 Komponen-Komponen Graf

Ada beberapa terminologi dari teori graf yang digunakan untuk menjelaskan apa yang dilihat ketika melihat suatu graf. Graf dapat dilihat dari komponen-komponen penyusunnya.

2.1.2.1 Titik (Verteks)

Titik pada graf G dapat dilabeli dengan huruf, misalkan $v, w, p, \ldots$ atau dengan menggunakan bilangan asli 1,2,3, ... atau gabungan keduanya. Jumlah titik pada graf dapat dinyatakan dengan n = |v|.

2.1.2.2 Sisi (Edge)

Sisi yang menghubungkan titik v_i dengan titik v_j dinyatakan dengan pasangan(v_i, v_j), atau dengan lambang $e_1, e_2, e_3, \ldots$. Dengan kata lain, jika e adalah sisi yang menghubungkan titik v_i dengan titik v_j, maka e dapat dituliskan sebagai $e = (v_i, v_j)$, dimana i, j adalah indeks angka bilangan asli 1,2,3, ...

2.1.2.3 *Derajat* (*degree*)

Jumlah atau banyaknya sisi yang terkait dengan suatu titik (*loop* dihitung dua kali), disebut derajat (*degree*) dari titik tersebut dinotasikan $d(v_i)$. Derajat suatu titik sering juga disebut valensi dari titik tersebut. Derajat minimum dari graf G dinotasikan dengan $\delta(G)$ dan derajat maksimumnya dinotasikan dengan $\Delta(G)$ (Siang, 2002: 205).

Contoh :

Pada Gambar 2.1, terlihat bahwa untuk setiap titik v di G derajat titiknya adalah $d(v_1) = 3,\ d(v_2) = 2,\ d(v_3) = 2,\ d(v_4) = 2,\ d(v_5) = 3$. Sehingga $\delta(G) = 2$ dan $\Delta(G) = 3$.

2.1.2.4 *Ukuran*

Ukuran (*Size*) dari suatu graf G adalah banyaknya titik yang dimiliki oleh graf G.

2.1.3 Keterhubungan

2.1.3.1 Jalan (Walk)

Misalkan G adalah sebuah graf. Sebuah jalan (*walk*) di adalah barisan berhingga (tak kosong) $W = v_0e_1v_1e_2v_2 \ldots e_kv_k$ yang suku-sukunya bergantian titik dan sisi, sedemikian sehingga v_{i-1} dan v_i adalah titik-titik akhir sisi e_i, untuk $1 \leq i \leq k$. Titik v_0 dan v_k berturut-turut disebut titik awal dan titik akhir W. Sedangkan titik-titik $v_1, v_2, \ldots, v_k$ disebut titik-titik internal dari W. Panjang dari jalan W adalah banyaknya sisi dalam W (Budayasa, 2007:6).

2.1.3.2 Jejak (*trail*)

Misalkan G adalah sebuah graf dan W Jalan di graf G. Jika semua sisi $e_1, e_2, \dots, e_k$ dalam jalan W berbeda, maka W disebut jejak (*trail*) (Budayasa, 2007:6).

2.1.3.3 Lintasan (*path*)

Misalkan G adalah sebuah graf dan W Jalan di graf G. Jika semua titik $v_0, v_1, v_2, \dots, v_k$ dalam jalan W juga berbeda, maka W disebut sebuah lintasan (*path*).

2.1.3.4 Jejak Tertutup

Misalkan G adalah sebuah graf dan W Jejak di graf G. Jejak disebut tertutup jika titik awal dan titik akhir dari W identik (sama). Jejak tertutup disebut juga sirkuit.

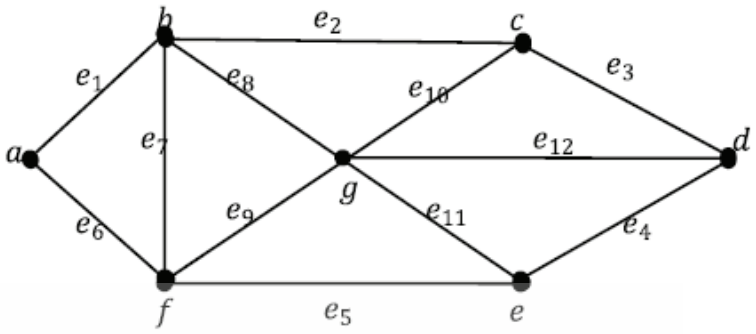
2.1.3.5 Sikel

Misalkan G adalah sebuah graf dan W sirkuit di graf G. Sirkuit yang titik awal dan titik internalnya berlainan disebut sikel (*cycle*). Banyaknya sisi dalam suatu sikel disebut panjang dari sikel tersebut. Sikel dengan n titik dinotasikan dengan C_n.

2.1.3.6 Sikel Hamilton

Misalkan G sebuah graf, sebuah sikel di G yang memuat semua titik di G disebut sikel Hamilton (Budayasa, 2007:129).

Contoh dari jalan, jalan tertutup, jejak, jejak tertutup, lintasan dan sikel dapat dilihat pada Gambar 2.2.



Gambar 2.2 Jalan, Jejak, Lintasan, dan Sikel dalam suatu Graf

Jalan : $ae_6fe_7be_2ce_3de_3c$.

Jalan tertutup : $ae_1be_2ce_{10}ge_{12}ge_9fe_6a$.

Jejak : $ae_1be_2ce_{10}ge_{11}ee_4de_3$.

Jejak tertutup : $ae_6fe_5ee_4de_{12}ge_8be_1a$.

Lintasan : $ae_1be_2ce_{10}ge_9fe_5ee_4a$.

Sikel : $ae_6fe_5ee_4de_{12}ge_8be_1a$.

2.1.4 Beberapa Jenis Graf

2.1.4.1 Graf Berbobot

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga. Bobot pada tiap sisi dapat berbeda-beda bergantung pada masalah yang dimodelkan dengan graf. Bobot dapat menyatakan jarak antara dua buah tiang listrik, kapasitas, biaya perjalanan antara dua buah kota, waktu tempuh pesan (*message*) dari sebuah simpul komunikasi ke simpul komunikasi lain, ongkos produksi, dan sebagainya. Graf yang setiap sisinya diberikan suatu bobot dinamakan dengan graf berbobot.

2.1.4.2 Graf Sederhana

Graf yang tidak mengandung *loop* maupun sisi ganda dinamakan graf sederhana.

2.1.4.3 Graf Tak Sederhan

Graf yang mengandung sisi ganda atau *loop* dinamakan graf tak-sederhana.

2.1.4.4 Graf Tak Berarah

Graf tak berarah adalah graf yang semua garisnya tidak memiliki arah. Misalkan graf G terdiri dari suatu himpunan V dari titik-titik dan suatu himpunan E dari garis-garis sedemikian rupa sehingga setiap garis e ϵ E dikaitkan dengan pasangan titik tak terurut. Jika terdapat sebuah garis e yang menghubungkan titik v dan w, maka dapat dituliskan dengan $e = (v, w)$ atau $e = (w, v)$ yang menyatakan sebuah garis antara v dan w.

2.1.4.5 Graf Berarah

Menurut Budayasa (2007:214), sebuah graf berarah D adalah suatu pasangan berurutan dari dua himpunan V(D) yaitu himpunan berhingga tak kosong yang anggota-anggotanya yang disebut titik $\Gamma(D)$ yaitu himpunan berhingga (boleh kosong) yang anggota-anggotanya disebut busur sedemikian hingga setiap busur merupakan pasangan berurutan dari dua titik di $V(D)$. Jika v_1 dan v_2 adalah dua titik pada graf berarah D dan $e = (v1, v2)$ sebuah busur D, maka e disebut busur keluar dari titik v_1 dan e disebut busur menuju titik v_2. Untuk efisiensi, busur $e = (v1, v2)$ sering ditulis (i, j).

2.1.4.6 Graf Euler

Jejak Euler ialah jejak yang melalui tiap sisi di dalam graf. Bila jejak tersebut kembali ke titik asal, membentuk jejak tertutup (sirkuit). Jejak tertutup itu dinamakan sirkuit Euler. Jadi, sirkuit Euler ialah sirkuit yang melewati tiap sisi di dalam graf. Graf yang mempunyai sirkuit Euler disebut graf Euler (*Eulerian Graph*) (Munir, 2009: 404).

2.1.4.7 Graf Hamilton

Lintasan Hamilton ialah lintasan yang melalui tiap titik di dalam graf. Sikel Hamilton adalah sikel yang melalui tiap titik di dalam graf. Graf yang memiliki sikel Hamilton dinamakan graf Hamilton (*Hamiltonian Graph*) (Munir, 2009:408).

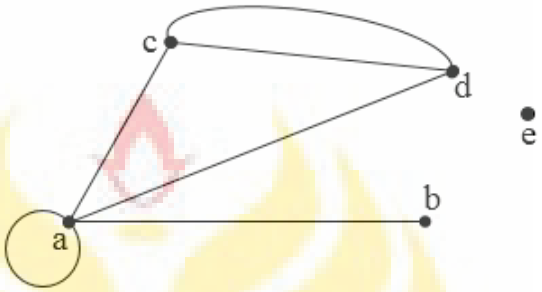
2.1.5 Representasi Graf dalam Matriks

Matriks dapat digunakan untuk menyatakan suatu graf. Hal ini sangat membantu untuk membuat program komputer yang berhubungan dengan graf. Dapat menyatakan graf sebagai suatu matriks, maka perhitungan-perhitungan yang diperlukan dapat dilakukan dengan mudah.

Matriks ketetanggaan atau matriks berhubungan langsung digunakan untuk menyatakan graf dengan cara menyatakannya dalam jumlah garis yang menghubungkan titik-titiknya. Jumlah baris dan kolom matriks ketetanggaan sama dengan jumlah titik dalam graf.

Misalkan G adalah sebuah graf dengan *n* titik. Matriks ketetanggaan dari graf G adalah matriks bujur sangkar (persegi) berordo *n*, $X(G) = x(ij)$, dengan

elemen $x(ij)$ menyatakan banyaknya sisi yang menghubungkan titik ke-i ke titik ke-j. Dengan definisi ini memungkinkan untuk menyatakan sebuah graf yang memiliki sisi paralel atau *loop* dengan matriks ketetanggaan (Sutarno, *et al*., 2003: 79). Contoh graf yang memiliki sisi paralel dan *loop* apat dilihat pada Gambar 2.3.



Gambar 2.3 Graf H yang memiliki sisi paralel dan loop

Matriks ketetanggannya:

$$X(H) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Matriks ketetanggaan juga digunakan untuk menyatakan graf berbobot, yaitu elemen-elemenya menyatakan bobot garis.

2.1.6 Matriks Ketetanggaan untuk Graf Berbobot

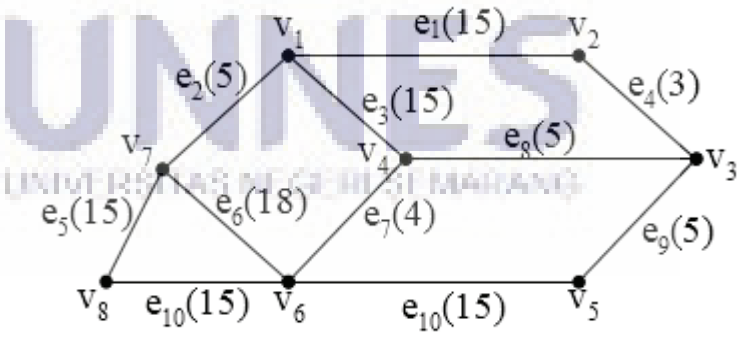
Diketahui G graf berbobot dengan setiap sisi dengan suatu bilangan riil tak negatif. Matriks yang bersesuaian dengan graf berbobot G adalah matriks ketetanggan atau matriks keterhubungan $X(G) = x(ij)$ dengan x_{ij} = bobot garis yang menghubungkan titik v_i dengan titik v_j. Jika titik v_i tidak berhubungan langsung dengan titik v_j maka $x_{ij} = \infty$, dan $x_{ij} = 0$, jika $i = j$ (Siang, 2002, p. 262).

Sebagai contoh dalam suatu provinsi, terdapat 8 kota $(v_1, v_2, v_3, \ldots v_8,)$ yang akan dihubungkan dengan jaringan-jaringan listrik. Biaya pemasangan jaringan listrik yang akan dibuat antar dua kota apat dilihat pada Tabel 2.1.

Tabel 2.1 Biaya Pemasangan Jaringan Listrik

Sisi	Kota yang dihubungkan	Biaya per satuan
e_4	$v_2 - v_3$	3
e_7	$v_4 - v_6$	4
e_2	$v_1 - v_7$	5
e_8	$v_3 - v_4$	5
e_9	$v_3 - v_5$	5
e_1	$v_1 - v_2$	15
e_3	$v_1 - v_4$	15
e_{10}	$v_6 - v_8$	15
e_5	$v_7 - v_8$	15
e_{11}	$v_5 - v_6$	15
e_6	$v_6 - v_7$	18

Graf berbobot untuk menyatakan jaringan listrik di 8 kota digambarkan pada Gambar 2.4. Angka dalam kurung menyatakan bobot garis yang bersangkutan. Bobot tersebut menyatakan biaya pemasangan jaringan listrik.



Gambar 2.4 Graf berbobot

Matriks keterhubungan untuk menyatakan graf berbobot pada gambar tersebut adalah matriks $X(\mathrm{G}) = x(\mathrm{ij})$ dengan,

x_{ij}= bobot garis yang menghubungkan titik v_i dengan titik v_j,

x_{ij} = ∞, jika titik v_i tidak berhubungan langsung dengan titikv_j,

x_{ij} = 0, jika i = j.

$$X(G) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{array} \\ \begin{bmatrix} 0 & 15 & \infty & 15 & \infty & \infty & 5 & \infty \\ 15 & 0 & 3 & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & 0 & 5 & 5 & \infty & \infty & \infty \\ 15 & \infty & 5 & 0 & \infty & 4 & \infty & \infty \\ \infty & \infty & 5 & \infty & 0 & 15 & \infty & \infty \\ \infty & \infty & \infty & 4 & 15 & 0 & 18 & 15 \\ 5 & \infty & \infty & \infty & \infty & 18 & 0 & 15 \\ \infty & \infty & \infty & \infty & \infty & 15 & 15 & 0 \end{bmatrix} \end{array}$$

Dalam program komputer, sel dengan harga ∞ diisi dengan suatu bilangan yang harganya jauh lebih besar dibandingkan dengan harga elemen-elemen yang bukan ∞.

2.2 Bintang Service Management (BSM)

PT Bintang Service Management (BSM) adalah perusahaan spesialis jasa *outsourcing* (*manpower supply*) yang berbasis di Semarang. PT Bintang Service Management berkontribusi membantu masyarakat dan pemerintah secara luas dalam mengurangi pengangguran, meningkatkan kualitas sumber daya manusia melalui pendidikan dan pelatihan yang berjenjang, serta menyalurkan dengan sistem alih daya atau *outsourcing* (Yoanita, 2016).

Management Profesional PT. Bintang Service Management menawarkan pemenuhan kebutuhan pemakai jasa agar memperoleh efisiensi biaya, meningkatkan produktifitas dan mengoptimalkan profit pengguna jasa. Ruang lingkup pelayanan PT. Bintang Service Management antara lain: jasa kebersihan

(*Cleaning Service*), *Security*, *Driver*, dan berbagai jenis *Human Resource Provider*, adapun klien-klien PT. Bintang Service Management antara lain: kantor, rumah sakit, hotel, perbankan, sekolah, kampus, mall, *supermarket*, *factory*, *restaurant*, *food court*, dan sebagainya.

Sistem *outsourcing* membantu perusahaan untuk menghindari eskalasi biaya dalam bentuk kenaikan gaji, pensiun, pesangon tenaga kerja, serta terbantu dalam pengelolaan tenaga kerja. PT. Bintang Service Management memberikan solusi dalam hal kebutuhan tenaga kerja yang dibutuhkan oleh Perusahaan atau kantor berbagai bidang. Selain di Semarang, area kerja proyek PT. BSM mencakup wilayah Bandung, Indramayu, dan sebagian besar area Jawa Tengah seperti Semarang Purwodadi, Pati, Pemalang, Slawi, Tegal, dan Brebes.

2.3 Teknik-Teknik Optimasi

Optimasi merupakan salah satu disiplin ilmu dalam matematika yang fokus untuk mendapatkan nilai minimum atau maksimum secara sistematis dari suatu fungsi, peluang, maupun pencarian nilai lainya dalam berbagai kasus. Optimasi sangat berguna hampir di semua bidang terutama bidang industri dalam rangka melakukan usaha secara efektif dan efisien untuk mencapai target hasil yang ingin dicapai. Tentunya hal ini akan sangat sesuai dengan prinsip ekonomi yang berorientasikan untuk senantiasa menekan pengeluaran untuk menghasilkan keluaran yang maksimal. Optimasi ini juga penting karena persaingan saat ini sudah benar benar sangat ketat (Pradhana, 2006).

Seperti yang dikatakan di awal, bahwasanya optimasi sangat berguna bagi hampir seluruh bidang yang ada, maka berikut ini adalah contoh bidang yang sangat terbantu dengan adanya teknik optimasi tersebut. Bidang tersebut, antara lain: arsitektur, jaringan komputer, sinyal dan pemprosesan gambar (*signal and immage processing*), telekomunikasi, ekonomi perindustrian, transportasi, perdagangan, pertanian, perikanan, perkebunan, perhutanan, dan sebagainya.

Teknik optimasi secara umum dapat dibagi menjadi dua bagian, yang pertama adalah Pemrograman Matematika (*Mathematical Programming*), dan yang kedua adalah Kombinasi Optimasi (*Combinatorial Optimatimization*). Dalam bidang *mathematical programming* dapat dibagi menjadi dua kembali, yaitu mendukung mesin vektor (*support vector machines*) dan gradient descent. Dan pada bidang *Combinatorial Optimization* kembali difokuskan lagi ke dalam dua bidang, yaitu Teori Graph (*Graph Theory*) dan Algoritma Genetik (*Genetic Algorithm*). Pemfokusan bidang tersebut dikarenakan beberapa parameter, diantaranya, Restorasi (*Restoration*), Pemilihan fitur (*Feature selection*), Klasifikasi (*Classification*), *Clustering*, *RF assignment*, *Compression,* dan sebagainya.

Adapun cara untuk membuat optimasi yang baik, adalah dengan memperhatikan hal-hal berikut (Pradhana, 2006).

1) Model titik awal (Model dan starting Point).
2) Menuju minimum/maksimum.
3) Mengelompokan masalah optimasi yang baik.
4) Menentukan permulaan.

5) Kendala pembatas memberikan sebuah pilihan.

Adapun hal lain secara global yang penting untuk diperhatikan adalah fokus terhadap model dan masalah serta cara berpikir yang analitis. Kita harus fokus terhadap model dan masalah agar tujuan utama dari kasus tersebut tercapai, jangan sampai terlalu konsen pada optimasi tetapi tujuannya sendiri malah tidak tercapai. Sedangkan berpikir analitis dimaksudkan agar kita peka terhadap keadaan dan mampu berpikir secara bebas untuk menemukan solusi solusi yang diperlukan. Sebagai contoh sederhana implementasi teknik optimasi ini, yaitu untuk mengoptimalkan performa komputer pada saat memakai suatu program agar berjalan lebih lancar. Caranya adalah dengan mematikan program-program yang running namun sebenarnya tidak diperlukan. Jika komputer kita tidak sedang membutuhkan koneksi dengan jaringan, sebaiknya semua *service* yang mendukung ataupun berhubungan dengan jaringan, ada baiknya dimatikan.

2.4 *Traveling Salesman Problem* (TSP)

Travelling Salesman Problem (TSP) termasuk ke dalam persoalan yang sangat terkenal dalam teori graf. Nama persoalan ini diilhami oleh masalah seorang pedagang yang berkeliling mengunjungi sejumlah kota. Deskripsi persoalannya adalah sebagai berikut: diberikan sejumlah kota dan jarak antar kota. Tentukan sikel terpendek yang harus dilalui oleh seorang pedagang bila pedagang itu berangkat dari sebuah kota asal dan menyinggahi setiap kota tepat satu kali dan kembali lagi ke kota asal keberangkatan (Sarma, *et al*., 2011:2540).

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 perjalanan pedagang tidak lain adalah menentukan sikel Hamilton yang memiliki bobot minimum pada sebuah graf terhubung.

Pada persoalan *Traveling Salesman Problem* ini, jika setiap simpul mempunyai sisi ke simpul yang lain, maka graf yang merepresentasikannya adalah graf lengkap berbobot. Pada sembarang graf lengkap dengan n buah simpul($n > 2$), jumlah sikel Hamilton yang berbeda adalah $(\frac{(n-1)!}{2})$. Rumus ini dihasilkan dari kenyataan bahwa dimulai dari sembarang simpul kita mempunyai $n-1$ buah sisi untuk dipilih dari simpul pertama, $n-2$ sisi dari simpul kedua, $n-3$ dari simpul ketiga, dan seterusnya. Ini adalah pilihan yang *independen*, sehingga kita memperoleh $(n-1)!$ pilihan. Jumlah itu harus dibagi dengan 2, karena tiap sikel Hamilton terhitung dua kali, sehingga semuanya ada $(\frac{(n-1)!}{2})$ buah sikel Hamilton.

2.5 Pencarian Heuristik

Dalam kecerdasan buatan (*artificial intelligence*) khususnya materi pencarian, ada dua metode yang umum digunakan yaitu metode pencarian buta (*Bind Search*) atau *Un-Informed Search* dan metode pencarian terbimbing (*Heuristic Search*) atau *Informed Search*. Dalam metode pencarian buta terapat dua algoritma yang umum digunakan, yaitu pencarian melebar pertama (*Breadth First Search*) dan pencarian mendalam pertama (*Dept First search*). Menurut Sutojo, *et al*. (2011:83), pencarian buta pada umumnya tidak efisien karena waktu akses dan memori yang dibutuhkan cukup besar. Untuk mengatasi masalah tersebut maka

ditambahkan suatu informasi pada domain yang bersangkutan sehingga proses pencarian yang baru akan dihasilkan. Pencarian seperti ini disebut sebagai *Informed Search* atau pencarian heuristik yaitu pencarian berdasarkan panduan.

Teknik pencarian heuristik merupakan suatu strategi untuk melakukan proses pencarian secara selektif dan dapat memandu proses pencarian yang memiliki kemungkinan sukses paling besar, namun dengan kemungkinan mengorbankan kelengkapan. Agoritma yang umum digunakan dalam pencarian heuristik adalah sebagai berikut:

1) *Generate and Test*,
2) *Hil Caimbing*,
3) *Best First Search*,
4) A*, dan
5) *Simulated Annelaing*.

2.6 Algoritma *Recursive Best First Search* (RBFS)

Rekursif Best First Search (RBFS) adalah algoritma *linear space* yang memperluas titik (*node*) pencarian dalam terbaik-pertama bahkan dengan fungsi biaya *nonmonotonic*, dan menghasilkan lebih sedikit *node* dari *Best first Search* dengan fungsi biaya monoton. Algoritma RBFS memperluas beberapa *node* lebih dari satu kali, sehingga lebih efisien per generasi simpul dari standar pencarian terbaik pertama *Best First Search* (BFS), namun, mungkin berjalan lebih cepat secara keseluruhan pada masalah di mana simpul generasi dan evaluasi sangat efisien (Korf, 1993).

Menurut Korf (1996), untuk kasus khusus di mana biaya sama dengan kedalaman, RBFS adalah asimtotik optimal dalam ruang dan waktu, menghasilkan $O(bd)$ *node*. Secara umum, dengan fungsi biaya monoton, RBFS menemukan solusi optimal, sementara memperluas sedikit *node* secara berulang untuk memperdalam pencarian, meskipun perbedaannya tidak signifikan dalam semua kasus. Secara umum *preudo code* dari algoritma *Rekursif Best- First Search* dapat dilihat pada Gambar 2.5.

```
RBFS( N, F(N), Bound)
   if F(N) > Bound then return F(N);
   if goal(N) then exit search;
   if N has no children then return ∞;
   for each child Ni of N do
      if f(N) < F(N) then F(Ni) := max(F(N),f(Ni)
                     else F(Ni) := f(Ni);
   Sort Ni in increasing order of F(Ni);
   if only one child then F(N2) := ∞;
   while F(N1) ≤ Bound and F(N1) < ∞ do
         F(N1) := RBFS(N1,F(N1),min(Bound,F(N2)))
         insert N1 in sorted order
   return F(N1).
```

Gambar 2.5 *Pseudo Code Recursive Best First Search*

Misalkan seorang manager perusahaan Jaya Baru yang berpusat di Jakarta ingin melakukan pengecekan layanan ke semua anak cabang perusahaannya di pulau Jawa, adapun anak perusahaan tersebut tersebar di Bandung, Cirebon, Semarang, Yogyakarta, dan Surakarta. Bagaimana cara manager perusahaan tersebut mengunjungi semua anak perusahaannya tepat satu kali dan kembali lagi

ke kantor pusat di Jakarta? Masalah tersebut dapat dikelompokan dalam masalah *Traveling Salesman Problem*, di mana seorang manager dari perusahaan pusat di Jakarta (titik a) harus mengunjungi semua anak cabang perusahaan di Bandung (titik b), Cirebon (titik c), Yogyakarta (titik d), Semarang (titik e), dan Surakarta (titik f) tepat satu kali dan kembali lagi ke Jakarta (titik a). Jarak antar perusahaan dapat dilihat pada Table 2.2.

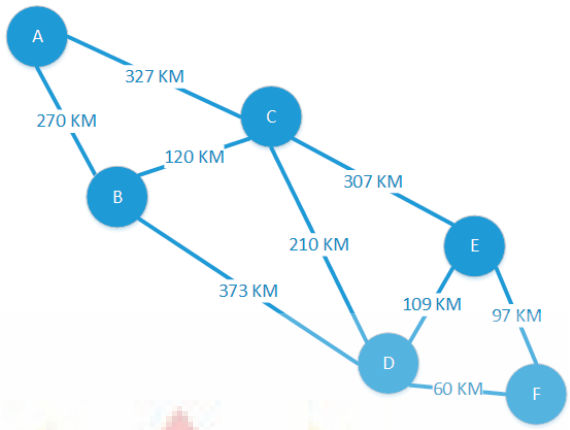
Tabel 2.2 Jarak antar perusahaan Jaya Baru dan antar anak perusahaan Jaya Baru

Sisi	Perusahaanyang dihubungkan	Jarak (KM)
e_1	$a-b$	270
e_2	$a-c$	327
e_3	$b-c$	120
e_4	$b-d$	373
e_5	$c-d$	210
e_6	$c-e$	307
e_7	$d-e$	109
e_8	$d-f$	60
e_9	$e-f$	97

Dalam bentuk matrik keadaan tersebut dapat di jelaskan sebagai berikut.

$$X(H) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 0 & 270 & 327 & \infty & \infty & \infty \\ 270 & 0 & 120 & 373 & \infty & \infty \\ 327 & \infty & 0 & 210 & \infty & \infty \\ \infty & \infty & 210 & 0 & 109 & 60 \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & 0 & 60 & 97 & 0 \end{bmatrix} \end{array}$$

Kondisi perusahaan, anak perusahaan yang ingin dikunjungi dan jarak antar perusahaannya dapat digambarkan dalam sebuah graf seperti Gambar 2.6.



Gambar 2.6 Graf perusahaan Jaya Baru dan anak perusahaannya

Pada langkah pertama dari perusahaan pusat a yang rute yang terbuka ada dua yaitu a, b dengan jarak 270 km dan a, c dengan jarak 327 km, dengan demikian diperoleh rute minimum a, b dengan jarak 270 km dan nilai minimum ke dua (*threshold*) yaitu rute a, c dengan jarak 327 km. Langkah kedua buka rute yang merupakan rute minimum pada langkah satu yaitu rute a, b dengan jarak 270 km, rute yang terbuka dari rute a, b yaitu rute a, b, c dengan jarak 390 km dan rute a, b, d dengan jarak 634 km. Bandingkan jarak pada rute yang terbuka pada langkah dua dengan *threshold* pada langkah satu, jarak yang paling minimum akan menjadi rute yang dibuka pada langkah ke tiga. *Threshold* pada langkah dua ditentukan dengan mencari nilai minimum ke dua dari *threshold* sebelumnya dibandingkan dengan semua rute yang mungkin dilewati pada langkah dua dan langkah sebelumnya yang belum menjadi *threshold* atau nilai minimum, artinya kita mencari minimum dari ke dua dari rute a, c dengan jarak 270 km, rute a, b, c dengan jarak 390 km dan rute a, b, d dengan jarak 634 km, sehingga di peroleh *threshold* rute a, b, c dengan jarak 390 km.

Langkah ketiga dan seterusnya dapat dilakukan sama seperti langkah ke dua, sehingga diperoleh tabel perhitungan seperti Tabel 2.3.

Tabel 2.3 contoh perhitungan algoritama dengan RBFS

No	Open	Rute	Minimum	Threshold
1.	a, a 0	a,b 270 a,c 327	a,b 270	a,c 327
2.	a, b 270	a,b,c 390 a,b,d 643	a,c 327	a,b,c 390
3.	a, c 327	a,c,d 537 a,c,e 634 a,c,b447	a,b,c 390	a,c,b 447
3.	a, b, c 390	a,b,c,d 600 a,b,c,e 697	a,c,b 447	a,c,d 537
4.	a, c, b 447	a,c,b,d 820	a,c,d 537	a,b,c,d 600
5.	a, c, d 537	a,c,d,e 646 a,c,d,f 597	a,c,d,f 597	a,b,c,d 600
6.	a, c, d, f 597	a,c,d,f,e 694	a,b,c,d 600	a,c,e 634
7.	a, b, c, d 600	a,b,c,d,e 709 a,b,c,d,f 660	a,c,e 634	a,b,d 643
8.	a, c, e 634	a,c,e,f 731 a,c,e,d 743	a,b,d 643	a,c,d,e 646
9.	a, b, d 643	a,b,d,e 752 a,b,d,c 853 a,b,d,f 703	a,c,d,e 646	a,b,c,d,f 660
10.	a, c, d, e 646	a,c,d,e,f 743	a,b,c,d,f 660	a,c,d,f,e 694
11.	a, b, c, d, f 660	a,b,c,d,f,e 757	a,c,d,f,e 694	a,b,c,e 697
12.	a, c, d, f, e 694	∞	a,b,c,e 697	a,b,d,f 703
13.	a, b, c, e 697	a,b,c,e,f 794 a,b,c,e,d 806	a,b,d,f 703	a,b,c,d,e 709
14.	a, b, d, f 703	a,b,d,f,e 800	a,b,c,d,e 709	a,c,e,f 731
15.	a, b, c, d, e 709	a,b,c,d,e,f 806	a,c,e,f 731	a,c,e,d 743
16.	$a, c,$ e, f 731	a,c,e,f,d 791	a,c,e,d 743	a,b,d,e 752
17.	a,c,e,d 743	a,c,e,d,f 803	a,b,d,e 752	a,b,c,d,f,e 757
18.	a,b,d,e 752	a,b,d,e,c 1059 a,b,d,e,f 849	a,b,c,d,f,e 757	a,c,e,f,d 791
19.	a,b,c,d,f,e 757	∞	a,c,e,f,d 791	a,b,c,e,f 794
20.	a,c,e,f,d 791	a,c,e,f,d,b1164	a,b,c,e,f 794	a,b,d,f,e 800

21.	a,b,c,e,f 794	a,b,c,e,f,d 854	a,b,d,f,e 800	a,c,e,d,f 803
22.	a,b,d,f,e 800	a,b,d,f,e,c 1107	a,c,e,d,f 803	a,b,c,e,d 806
23.	a,c,e,d,f 803	∞	a,b,c,e,d 806	a,c,b,d 820
24.	a,b,c,e,d 806	a,b,c,e,d,f 866	a,c,b,d 820	a,b,d,e,f 849
25.	a,c,b,d 820	a,c,b,d,f 880 a,c,b,d,e 929	a,b,d,e,f 849	a,b,d,c 853
26.	a,b,d,e,f 849	∞	a,b,d,c 853	a,b,c,e,f,d 854
27.	a,b,d,c 853	a,b,d,c,e 1160	a,b,c,e,f,d 854	a,b,c,e,d,f 866
28.	a,b,c,e,f,d 854	∞	a,b,c,e,d,f 866	a,c,b,d,f 880
29.	a,b,c,e,d,f 866	∞	a,c,b,d,f 880	a,c,b,d,e 929
30.	a,c,b,d,f 880	a,c,b,d,f,e 977	a,c,b,d,e 929	a,b,d,e,c 1059
31.	a,c,b,d,e 929	a,c,b,d,e,f 1026	a,c,b,d,e,f 1026	a,b,d,e,c 1059
32.	a,c,b,d,e,f 1026	∞	a,b,d,e,c 1059	a,b,d,f,e,c 1107
33.	a,b,d,e,c 1059	∞	a,b,d,f,e,c 1107	a,b,d,c,e 1160
34.	a,b,d,f,e,c 1107	$a,b,d,f,e,c,-a,$ 1434	a,b,d,c,e 1160	a,c,e,f,d,b1164
35.	a,b,d,c,e 1160	a,b,d,c,e,f 1257	a,c,e,f,d,b1164	a,b,d,c,e,f 1257
36.	a,c,e,f,d,b1164	$a,c,e,f,d,b,-a,$ 1434	a,b,d,c,e,f 1257	$a,b,d,f,e,c,-a,$ 1434
37.	a,b,d,c,e,f 1257	∞	$a,b,d,f,e,c,-a,$ 1434	∞
38.	$a,b,d,f,e,c,-a,$ 1434	∞	∞	∞

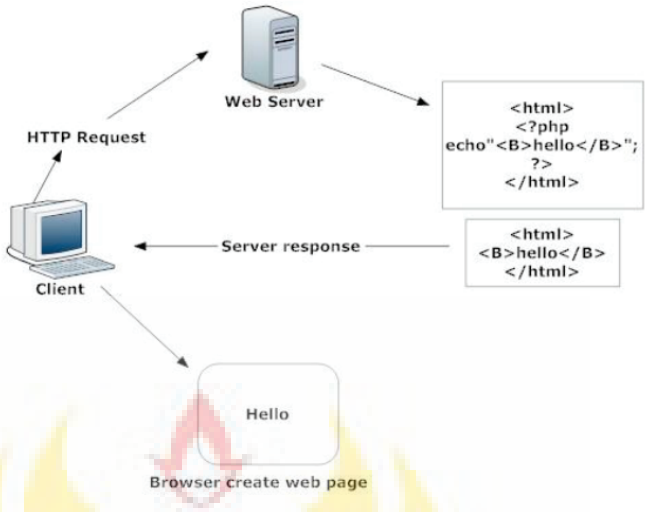
Pada langkah ke tiga puluh delapan kita dapat melihat bahwa rute yang terbuka yaitu rute $C = \{a, b, d, f, c, a\}$ dengan jarak 1434 km dan tidak ada rute yang mungkin dilewati lagi, begitu juga dengan rute dengan jarak minimum dan *threshold*, keduanya bernilai ∞, sehingga diperoleh rute terpendek yang dapat dilalui yaitu dari Jakarta – Bandung – Yogyakarta – Surakarta – Semarang – Cirebon – Jakarta dengan bobot 1434 km.

2.7 *Hipertext Preprocessor* (PHP)

Hypertext Preprocessor (PHP) merupakan bahasa pemrograman yang difungsikan untuk membangun suatu *website* dinamis. Menurut (Anuja, *et al*., (2014:1040), pada awalnya PHP ditulis menggunakan bahasa C untuk alasan kinerja dalam bentuk form web dan komunikasi dengan database oleh Rasmus Lerdorf, implementasi ini disebut sebagai PHP / FI (*personal home* page/*form interpreter*). Namun, sekarang PHP disebut *Hypertext Preprocessor* karena setiap kali ada permintaan untuk setiap halaman PHP di *address bar* browser yang *request* pertama dikirim ke server setelah di interprestasikan dalam file PHP kemudian dikembalikan respon dalam format HTML.

PHP menyatu dengan kode HTML, HTML digunakan sebagai pembangun atau pondasi dari kerangka layout *website* sedangkan PHP difungsikan sebagai pemroses data, sehingga dengan adanya PHP sebuah *website* akan mudah untuk di-*maintenance*.

PHP merupakan bahasa pemrograman yang berjalan pada sisi server sehingga PHP disebut juga sebagai bahasa *Server Side Scripting* artinya bahwa dalam setiap menjalankan PHP membutuhkan *web server* untuk menjalankanya. Adapun proses eksekusi kode PHP didalam sisi *server* ditunjukan oleh Gambar 2.7.

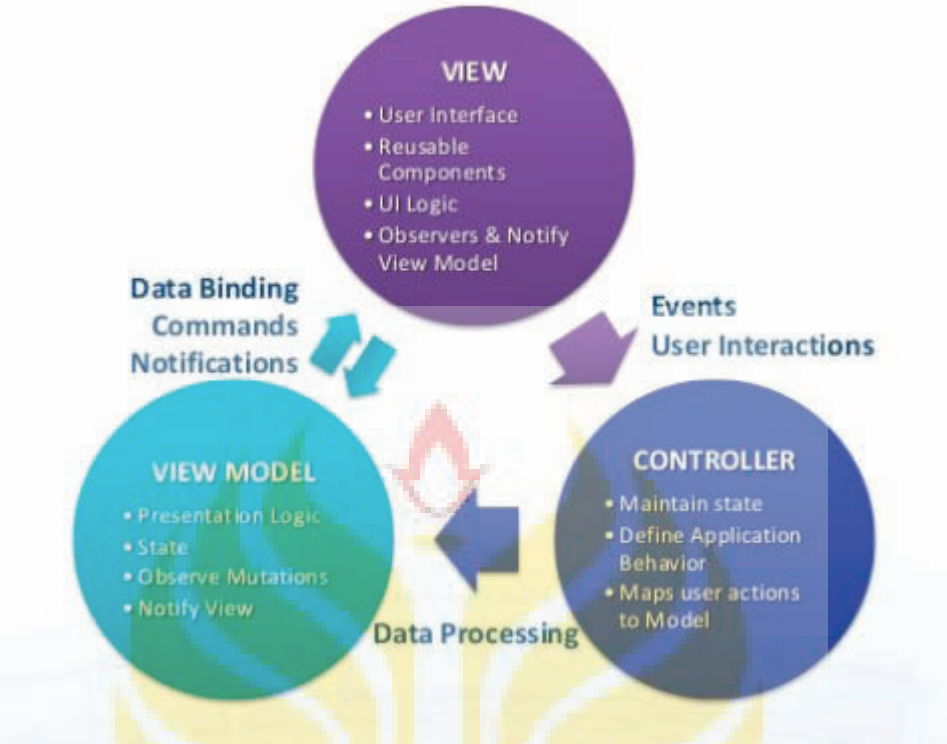


Gambar 2.7 Skema kerja *Hopertext Prepocessor* (PHP)

2.8 AngularJS

AngularJS adalah kerangka struktural (*structural framework*) untuk aplikasi web dinamis. AngularJS memungkinkan untuk menggunakan HTML sebagai bahasa template dan memungkinkan untuk memperluas sintaks HTML mengungkapkan komponen aplikasi dengan jelas dan ringkas (Google, 2016). Menurut (Ambulkar, 2016), AngularJS merupakan JavaScript *framework* yang dikembangkan oleh Google untuk memudahkan proses pembuatan website yang *responsive*, dan memiliki performa yang halus (*smoot performance*).

Secara umum angular menggunakan *design pattern* MVVM (Model View ViewModel) design pattern ini dapat dilihat seperti Gambar 2.8.



Gambar 2.8 Konsep MVVM AngularJS

Pada dasarnya aliran data berasal dari *Controller* kemudian dikirim ke *ViewModel* sebagai *presentation logic* selanjutnya data tersebut dilakukan binding ke *View* sehingga *user* bisa melihat persentasi data. Dalam *user interface* yang tersedia *user* bias melakukan interaksi mengirim perintah atau data ke *Controller* tanpa melakukan *load* aplikasi secara penuh, inilah yang disebut *Two-Way Data Binding*.

2.9 LeafletJS

LeafletJS merupakan *open-source* javascript *library* yang yang digunakan untuk pembuatan peta interaktif pada *canvas* HTML5. (Dimitrijević, *et al*., 2014). LeafletJS memiliki desain yang sederhana, memiliki kinerja dan kegunaan yang mudah. Leaflet bekerja secara efisien di semua platform

desktop dan *mobile*, dapat dikolaborasikan dengan banyak *library*, memiliki tampilan yang menarik, mudah digunakan dan terdokumentasi API dengan baik dan sederhana, memiliki *source code* yang mudah dibaca dan mudah untuk berkontribusi (LeafletJS, 2016).

Salah satu produk dari LeafletJS yaitu angular-leaflet-directive, direktif ini memungkinkan untuk menanamkan peta pada aplikasi AngularJS dan berinteraksi dua arah dengan melalui lingkup AngularJS dan peta leaflet API *library*. Angular-leaflet-directive dapat memudahkan kita dalam membuat peta secara sederhana. Misalakan untuk membuat *marker* pada peta seperti yang ditunjukan pada Gambar 2.11, sintak yang di perlukan dalam HTML yang diperluka ditunjukan pada Gambar 2.9.

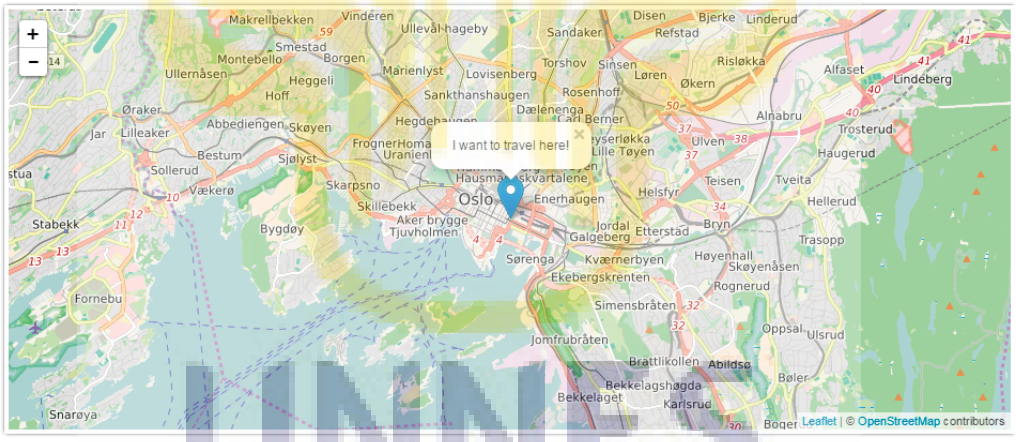
```
<div ng-controller="MarkerController">
    <leaflet markers="markers" center="osloCenter"></leaflet>
</div>
```

Gambar 2.9 *Source-code* HTML untuk menampilkan marker LeafletJS

Sementara *Controller* yang dibutuhkan seperti untuk menampilkan marker tersebut dapat dilihat pada Gambar 2.10.

```
app.controller("MarkerController", [ '$scope', function($scope) {
    angular.extend($scope, {
        osloCenter: {
            lat: 59.91,
            lng: 10.75,
            zoom: 12
        },
        markers: {
            osloMarker: {
                lat: 59.91,
                lng: 10.75,
                message: "I want to travel here!",
                focus: true,
                draggable: false
            }
        },
        defaults: {
            scrollWheelZoom: false
        }
    });
}]);
```

Gambar 2.10 *Controller* dalam pembuatan marker LeafletJS



Gambar 2.11 Contoh penggunaan marker pada peta dengan LeafletJS

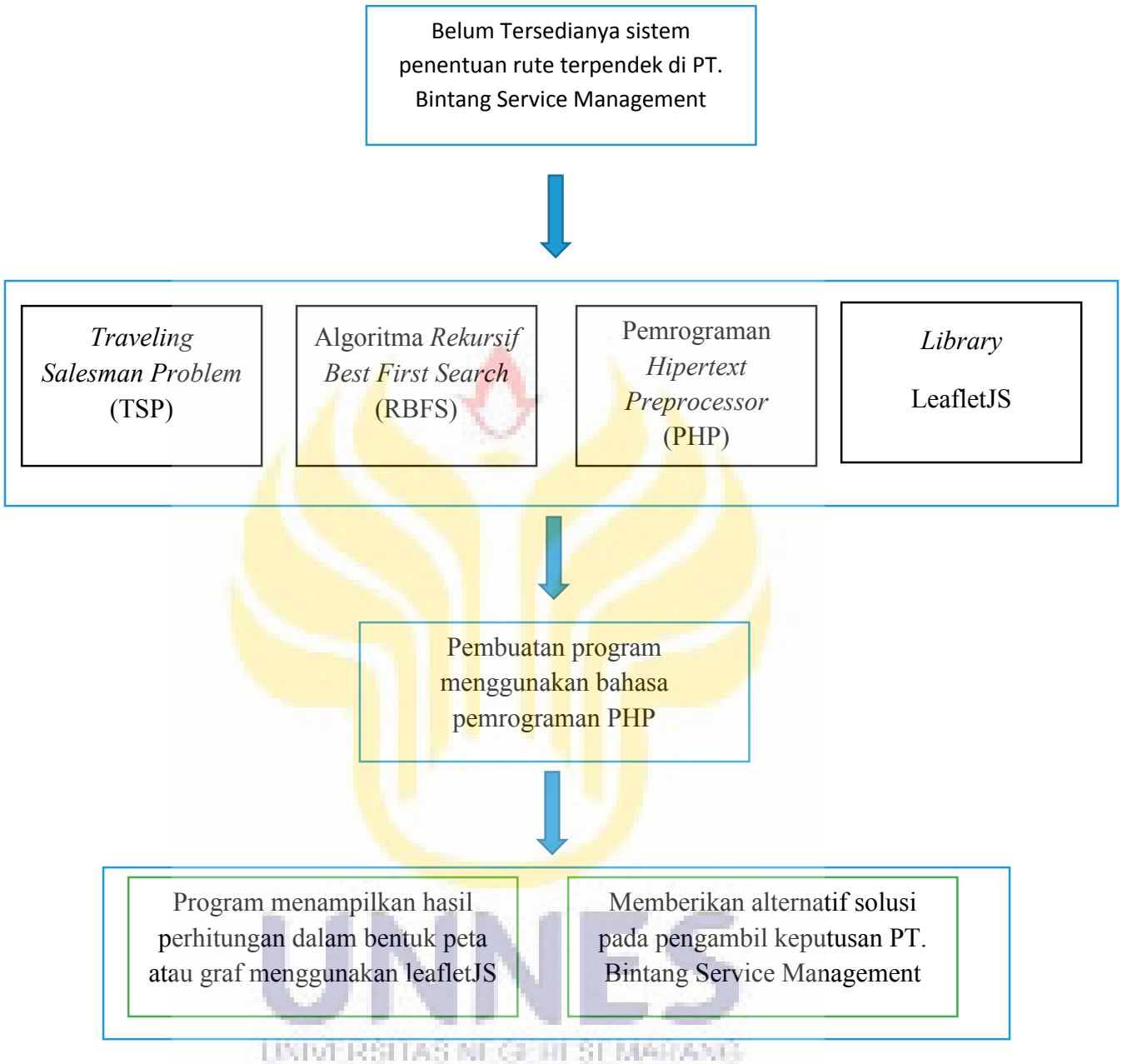
2.10 Kerangka Berpikir

Belum tersedianya sistem perhitungan untuk menentukan rute terpendek yang harus dilalui dalam pengadaan barang dan pengecekan barang kepada perusahaan klien PT. Bintang Service Management menyebabkan kurang efektif dan efisien dalam melayani permintaan klien, untuk mengatasi masalah

tersebut perlu diadakan sebuah sistem untuk menentukan rute terpendek agar perusahaan menjadi lebih efektif dan efisien dalam melayani kebutuhan perusahaan kliennya. Masalah rute terpendek yang dimaksud adalah masalah *Traveling Salesman Problem*.

Masalah *Traveling Salesman Problem* dapat diselesaikan dengan metode pencarian heuristik. Teknik pencarian heuristik merupakan suatu strategi untuk melakukan proses pencarian secara selektif dan dapat memandu proses pencarian yang memiliki kemungkinan sukses paling besar. Salah satu algoritma yang dapat digunakan adalah *Rekursif Best First Search* (RBFS). RBFS adalah algoritma *linear space* yang memperluas *node* pencarian dalam *Best first Search* dengan fungsi biaya nonmonotonic, dan menghasilkan lebih sedikit *node* dari *Best first Search* dengan fungsi biaya monoton.

Penyelesaian *Traveling Salesman Problem* dengan algoritma *Rekursif Best First Search* dapat diimplementasikan dengan bahasa pemrograman PHP. Pembuatan program tersebut diharapkan dapat membatu pengambil keputusan PT. Bintang Service Mangement dalam menyelesaikan *Traveling Salesman Problem* sehingga dalam menentukan rute perjalanan untuk melakukan pengadaan barang dan pengecekan barang ke pihak klien perusahaan secara lebih efektif dan efisien.



Gambar 2.12 Kerangka Berpikir

BAB 5

PENUTUP

5.1 Simpulan

Berdasarkan hasil analisis dan pembahasan pada bab 4, maka dapat disimpulkan sebagai berikut.

1. Algoritma *Recursive Best First Search* dapat diterapkan untuk menyelesaikan *Traveling Salesman Problem* di PT. Bintang Service Management. Hasil perhitungan algoritma berupa sikel Hamilton dengan bobot terkecil yang merupakan solusi dari *Traveling Salesman Problem*.
2. Pembuatan program algoritma *Recursive Best First Search* untuk penyelesaian *Traveling Salesman Problem* di PT. Bintang Service Management dengan bahasa PHP dengan memvisualisasikan keadaan data di lapangan (lokasi perusahaan) dalam bentuk peta (Graf), kemudian data diolah dengan algoritma RBFS dalam fungsi TSP, sehingga menghasilkan data dalam bentuk JSON, data JSON selanjutnya diparsing ke AngularJS menggunakan *library* LeafletJs sehingga keluaran program berupa sikel Hamilton yang divisualisasikan dalam bentuk peta di lapangan dan keterangan tentang sikel tersebut.

5.2 Saran

Berdasarkan simpulan hasil penelitian, saran yang perlu disampaikan sebagai berikut.

1. Bobot yang digunakan dalam penelitian ini berupa jarak (dalam satuan km) dan mengabaikan halangan seperti kepadatan lalu lintas, lampu lalu lintas, portal jalan, pengalihan jalan dan halangan sejenisnya. Diharapkan penelitian selanjutnya dapat menggunakan kepadatan lalu lintas, lampu lalu lintas, portal jalan, pengalihan jalan dan halangan sejenisnya.
2. Dalam penelitian ini penulis menggunakan bahasa pemrograman *Hypertext Preprocessor* (PHP) untuk menyelesaikan *Traveling Salesman Problem*, selain itu program tersebut dapat memvisualisasikan dalam bentuk peta. Peta yang di sajikan dihubungkan dalam bentuk ruas garis lurus, sehingga dalam penelitian ini saran yang dapat disampaikan adalah untuk penelitian-penelitian selanjutnya yang mengkaji masalah *Traveling Salesman Problem* dapat mencoba untuk menampilkan rute sesuai dengan keadaan yang sebenarnya.

DAFTAR PUSTAKA

Anggit, W. D., Alamsyah & Abidin, Z. 2014. Solusi Travelling Salesman Problem Menggunakan Algoritma Fuzzy Evolusi. *Unnes Journal of Mathematics,* 3(1):39-43.

Anitya, S. F., Sugiharti, E. & Dwijanto. 2013. Implementasi Algoritma Genetika untuk Menyelesaikan Travelling Salesman Problem. *UNNES Journal of Mathematics,* 2(2):118-120.

Anuja, D. A., Deshmukh, H. R., Khan, Z. I. & Popli, S. H. 2014. The Overview Of PHP. *International Journal of Pure and Applied Research in Engineering and Technology,* 2(9):1039-1043.

Budayasa, K. I. 2007. *Teori Graph an Aplikasinya.* Surabaya: Unesa University Press.

Çiçek, I. & Özer, B. 2011. The Effect of Outsourcing Human Resource on Organizational Performance: The Role of Organizational Culture. *International Journal of Business and Management Studies,* 3(2):131-144.

Gupta, S. & Panwar, P. 2013. Solving Travelling Salesman Problem using Genetic Algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering,* 3(6):376-380.

Hermuningsih, S. 2013. Buletin Ekonomi Moneter dan Perbankan. *Pengaruh Profitabilitas, Growth Opportunity, Sruktur Modal terhadap Nilai Perusahaan pada Perusahaan Publik di Indonesia*, Oktober, 127-144.

Hlaing, Z. C. S. S. & Khine, M. A. 2011. Solving Traveling Salesman Problem by using Improved Ant Colony Optimization Algorithm. *International Journal of Information and Education Technology,* 1(5):404-409.

Iqbal, Z. & Dad, A. M. 2013. Outsourcing: A Review of Trends, Winners & Losers and Future Directions. *International Journal of Business and Social Science,* 4(8):91-107.

Katiyar, S., Sonigoswami, Mehta, R. & Gaurav Singh. 2014. Implementation of Travelling Salesman Problem using Ant Colony Optimization. *Journal of Engineering Research and Applications,* 4(6):63-67.

Korf, R. E. 1993. Linear-space best-first search. *Artificial Intelligence,* 1(62):41-78.

Korf, R. E. 1996. *Artificial Intelligence Search Algoritm.* Los Angeles: Computer Science Universiti of California.

Kusrini & Istiyanto, J. E. 2007. Penyelesaian Travelling Salesman Problem dengan Algoritma Cheapest Insertion Heuristics dan Basis Data. *Jurnal Informatika,* 8(2):109-114.

Munir, R. 2009. *Matematika Diskrit.* 3th ed. Bandung: Informatika.

Pradhana, B. A. 2006. *Studi dan Implementasi Persoalan Lintasan Terpendek Suatu Graf dengan Algoritma Dijkstra dan Algoritma Bellman-Ford.* [Online] Tersedia di http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2006-2007/Makalah/Makalah0607-26.pdf [diakses 08-08-2006].

Sarma, S. V. M. Nagendram, N. V. & kumar, T. V. P. 2011. A note on Relations Between Barnette and sparse Graphs. *Intertional Journal of Mathematic Archive,* 2(12):2538 - 2542.

Siang, J. J. 2002. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer.* Yogyakarta: ANDI.

Sutarno, H., Nanang, P. & Nurjanah. 2003. *Matematika Diskrit.* Malang: UM PRESS.

Sutojo, T., Mulyanto, E. & Suhartono, V. 2011. *Kecerdasan Buatan.* I ed. Yogyakarta: ANDI.

Yoanita, P. 2016. *Bintang Service Management.* [Online] Tersedia di http://bsmos.co.id/ [diakses 31-07-2016].