BAB II

KAJIAN TEORI

Pada bab ini, diuraikan mengenai landasan teori yang akan digunakan dalam bab selanjutnya. Landasan teori yang dibahas pada bab ini yaitu mengenai teori graf, optimisasi, *Travelling Salesman Problem* (TSP), algoritma semut, logika *fuzzy*, *toolboox fuzzy* pada *Matlab*, dan jalan beserta beberapa karakteristiknya.

A. Teori Graf

Dalam kehidupan sehari-hari, banyak persoalan yang disimpulkan sebagai persoalan yang berhubungan dengan himpunan dan relasi binary, dengan logika dari persoalan tersebut sering kali dapat digambarkan dengan sebuah graf (Wibisono, 2008:125). Teori graf merupakan pokok bahasan yang sudah ada sejak lama tapi masih memiliki banyak terapan hingga saat ini. Menurut catatan sejarah, masalah jembatan Könisberg merupakan masalah yang pertama kali menggunakan graf yakni pada tahun 1736 oleh seorang matematikawan Swiss, L.Euler (Munir, 2010:354).

1. Definisi Graf

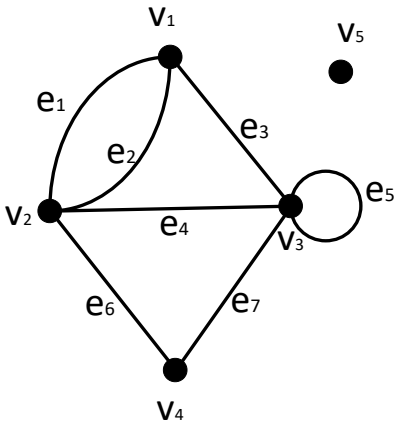
Menurut (Munir, 2010:356) secara matematis, graf didefiniskan sebagai berikut:

Suatu graf G terdiri atas pasangan himpunan (*V*,*E*), ditulis dengan notasi G=(*V*,*E*), dua himpunan yaitu himpunan tak kosong *V* yang unsur-unsurnya disebut simpul (*vertices* atau *node*) dan himpunan *E* yang unsur-unsurnya disebut rusuk (*edges* atau *arcs*) yang menghubungkan sepasang simpul.

Simpul pada graf dapat dilabeli dengan huruf, seperti a,b,c,..., dengan bilangan asli 1,2,3,... atau gabungan keduanya. Sedangkan rusuk yang menghubungkan simpul v_1 dan simpul v_2 dinyatakan dengan pasangan (v_1, v_2) atau biasa dinyatakan dengan lambang $e_1, e_2, \ldots, e_n$. Dengan kata lain, jika e_1 adalah rusuk yang menghubungkan simpul v_1 dan simpul v_2, maka e_1 dapat ditulis sebagai $e_1 = (v_1, v_2)$ atau $e_1 = (v_2, v_1)$.

2. Terminologi Graf

Berikut ini didefinisikan beberapa terminologi (istilah) yang sering digunakan dalam graf. Graf pada Gambar 2.1 akan digunakan untuk memperjelas terminologi yang didefinisikan.



Gambar 2. 1 Graf

a. Bertetangga *(adjacent)*

Menurut (Munir, 2010:365) dua buah simpul pada graf G dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah rusuk. Dengan kata lain, u bertetangga dengan v jika (u, v) adalah sebuah rusuk pada graf G. Pada Gambar 2.1 , simpul v_1 bertetangga dengan simpul v_2 dan v_3, tetapi simpul v_1 tidak bertetangga dengan simpul v_4 dan v_5.

b. Bersisian *(incident)*

Untuk sembarang rusuk $e = (u, v)$ rusuk e dikatakan bersisian dengan simpul u dan v. Pada Gambar 2.1, rusuk (v_1, v_2) bersisian dengan simpul v_1 dan simpul v_2, rusuk (v_2, v_3) bersisian dengan simpul v_2 dan simpul v_3, untuk contoh rusuk yang tidak bersisian pada rusuk (v_1, v_2) tidak bersisian dengan simpul v_3.

c. Simpul Terpencil (*Isolated Vertex*)

Simpul terpencil adalah simpul yang tidak mempunyai rusuk yang bersisian dengannya. Atau, dapat juga dinyatakan simpul terpencil adalah simpul yang tidak satupun bertetangga dengan simpul-simpul yang lain. Pada Gambar 2.1 , simpul terpencil adalah simpul v_5 (Munir, 2010:365).

d. Rusuk Gelang *(Loop)*

Sebuah rusuk yang menghubungkan sebuah simpul dengan dirinya sendiri disebut rusuk gelang. Pada Gambar 2.1 rusuk e_5 = (v_3, v_3) dinamakan rusuk gelang (*loop*).

e. Rusuk Ganda

Dua buah rusuk atau lebih yang menghubungkan simpul-simpul yang sama merupakan rusuk ganda. Pada Gambar 2.1 rusuk $e_1 = (v_1, v_2)$ dan $e_2 = (v_1, v_2)$ dinamakan rusuk ganda karena kedua rusuk tersebut menghubungkan dua simpul yang sama, yaitu v_1 dan v_2.

f. Derajat *(degree)*

Menurut Munir (2010: 365) derajat suatu simpul pada graf adalah banyaknya ujung rusuk yang hadir pada suatu simpul. Dinotasikan dengan $d(v)$. Pada Gambar 2.1 $d(v_1) = d(v_2) = 3$, $d(v_3)$ = 5, $d(v_4)$ = 2, $d(v_5)$ = 0.

g. Graf Berlabel/Berbobot (*Weighted Graph*)

Menurut (Munir, 2010:376) graf berbobot adalah graf yang setiap rusuk, simpul, atau rusuk dan simpul diberi sebuah harga (bobot).

h. Graf Kosong (*Null Graph* atau *Empty Graph*)

Graf yang himpunan rusuknya merupakan himpunan kosong atau tidak mempunyai rusuk hanya mempunyai simpul disebut sebagai graf kosong dan ditulis sebagai N_n, n dalam hal ini adalah jumlah simpul (Munir, 2010: 366).

i. Graf Bagian (*Subgraph*)

Sebuah graf H disebut graf bagian dari graf G, ditulis H ⊆ G, jika V(H) ⊆ V(G) dan E(H) ⊆ E(G). artinya semua simpul H termuat di G dan semua rusuk di H termuat di G. (Munir, 2010: 372).

3. Jenis Graf

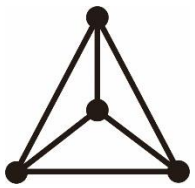
Graf dapat dikelompokan menjadi beberapa kategori bergantung pada sudut pandang pengelompokanya. Berdasarkan ada tidaknya rusuk ganda atau gelang pada suatu graf, secara umum graf dapat dikelompokan menjadi dua jenis (Munir, 2010:357-358):

a. Graf Sederhana *(*simple *graph)*

Graf yang tak mengandung rusuk ganda ataupun gelang (*loop*) dinamakan graf sederhana. Pada graf sederhana rusuk merupakan pasangan tak-terurut. Jadi, menuliskan rusuk (u, v) sama saja dengan (v, u). Ada beberapa graf sederhana khusus yang dijumpai pada banyak aplikasi, diantaranya:

1) Graf Teratur (*regular graph*)

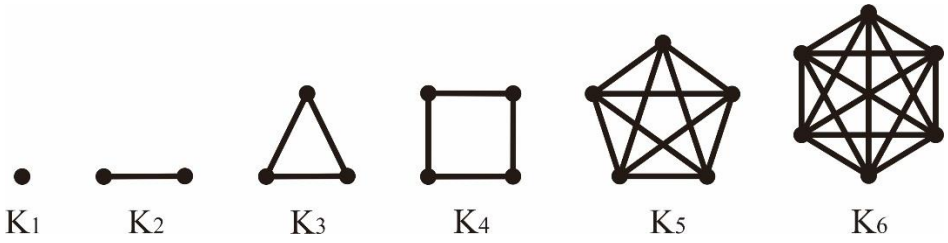
Graf yang setiap simpulnya mempunyai derajat yang sama disebut graf teratur (Munir, 2010:378). Jika G suatu graf teratur berderajat r maka banyaknya rusuk adalah $\frac{1}{2}nr$.



Gambar 2.2 Graf Teratur Berderajat 3

2) Graf Lengkap (*complete graph*)

Graf lengkap adalah graf sederhana yang setiap simpulnya mempunyai rusuk ke semua simpul lainya. Graf lengkap dengan *n* buah simpul dilambangkan dengan K_n. setiap simpul pada K_n berderajat *n*-1. Banyak rusuk pada graf lengkap yang terdiri dari *n* buah simpul adalah $\frac{n(n-1)}{2}$ (Munir, 2010:377):

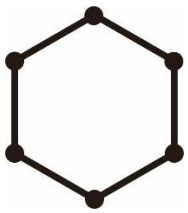


Gambar 2.3 Graf Lengkap Kn, 1 ≤ n ≤ 6

3) Graf Lingkaran *(Cycle Graph)*

Graf lingkaran adalah graf sedehana yang setiap simpulnya berderajat dua. Graf lingkaran dengan n simpul dilambangkan dengan C_n. Jika simpul-simpul pada C_n adalah $V_1, V_2, V_3, \ldots V_n$, maka rusuk-rusuknya adalah (V_1, V_2), (V_2, V_3), …,

(V_{n-1}, V_n) dan (V_n, V_1). Dengan kata lain, ada rusuk dari simpul terakhir V_n ke simpul pertama V_1. (Munir, 2010:377)



Gambar 2.4 Graf Lingkaran

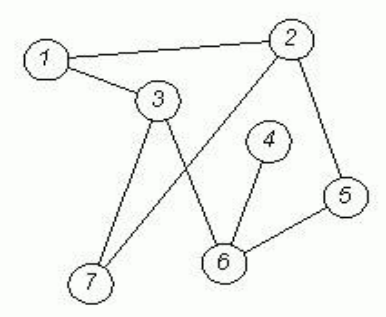
b. Graf Tak Sederhana

Graf yang mengandung rusuk ganda ataupun gelang dinamakan graf tak-sederhana. Terdapat dua macam graf tak-sederhana, yakni graf semu (*pseudograph*) dan graf ganda (*multigraph*). Graf semu adalah graf yang mengandung gelang (*loop*) sedangkan graf ganda adalah graf yang didalamnya terdapat rusuk ganda. Rusuk ganda yang menghubungkan sepasang simpul bisa lebih dari dua buah.

Berdasarkan orientasi arah pada rusuk, maka secara umum graf dibedakan menjadi dua jenis (Munir, 2010:358):

c. Graf Tak-Berarah (*undirected graph* atau *undigraph*)

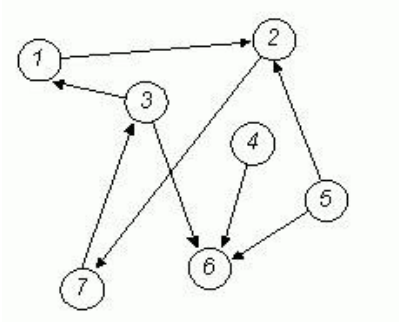
Graf yang rusuknya tidak mempunyai orientasi arah disebut graf tak-berarah. Pada graf tak berarah, urutan pasangan simpul yang dihubungkan oleh rusuk tidak diperhatikan. Jadi $(u, v) = (v, u)$ adalah rusuk yang sama.



Gambar 2.5 Graf tak Berarah

d. Graf Berarah (*directed graph* atau *digraph)*

Graf yang setiap rusuknya diberikan orientasi arah disebut sebagai graf berarah. Rusuk pada graf berarah biasanya disebut busur *(arc).* Pada graf berarah, (u, v) dan (v, u) menyatakan dua buah busur yang berbeda, dengan kata lain $(u, v) \neq (v, u)$. Untuk busur (u, v) simpul u dinamakan simpul asal *(initial vertex)* dan simpul v dinamakan simpul terminal *(terminal vertex).*



Gambar 2.6 Graf Berarah

4. Lemma Jabat Tangan

Jumlah derajat semua simpul pada suatu graf adalah genap, yaitu dua kali jumlah rusuk pada graf tersebut. Misal G suatu graf dengan $|V(G)| = n$, $|E(G)| = m$ dan $d(v_i)$ adalah derajat simpul $v_i, i = 1,2,3, \dots, n$, maka $\sum_{v \in V} d(v) = 2m$

Akibat dari teorema jabat tangan diatas, dapat diturunkan teorema berikut:

Teorema. *Jika G suatu graf, maka banyaknya simpul yang berderajat ganjil adalah genap.*

Bukti:

Misalkan V_1 dan V_2 masing-masing adalah himpunan simpul yang berderajat genap dan berderajat ganjil pada graf $G = (V, E)$, maka

$$\sum_{v \in V_1} d(v) + \sum_{v \in V_2} d(v) = \sum_{v \in V} d(v)$$

Berdasarkan teorema jabat tangan, maka diperoleh $\sum_{v \in V} d(v)$ genap. Karena $\sum_{v \in V_1} d(v)$ genap maka $\sum_{v \in V_2} d(v)$ adalah genap. Jadi $|V_2|$ genap.

5. Keterhubungan

Sebuah graf G disebut terhubung jika untuk setiap dua simpul u dan v di G terdapat lintasan di G yang menghubungkan kedua simpul tersebut. Jika tidak, maka G disebut graf tak-terhubung. (Munir, 2010: 371) Keterhubungan dalam graf meliputi unsur-unsur keterhubungan, sikel, dan sirkuit, lintasan terpendek, dan pohon.

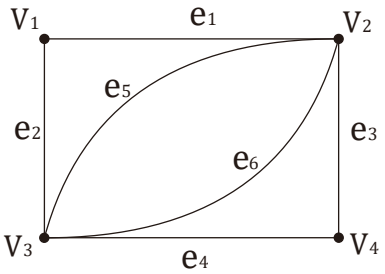
a. Unsur Keterhubungan

Unsur-unsur keterhubungan meliputi:

1) Perjalanan *(Walk)*

Perjalanan (*walk*) pada graf G dengan $V(G) = \{v_1, v_2, v_3, \ldots., v_n\}$ dan $E(G) = \{e_1, e_2, e_3, \ldots., e_n\}$ merupakan barisan berhingga $W = \{v_1, e_1, v_2, e_2, \ldots., v_n e_n\}$ yang suku-sukunya berupa simpul dan rusuk secara bergantian sedemikian sehingga $v_1$dan v_{i+1} adalah simpul ujung dari e_i,. Simpul v_1 dan v_n berturut-turut

disebut simpul awal dan simpul akhir dari W. Suatu perjalanan dikatakan perjalanan tertutup jika simpul awal dan simpul akhirnya sama. Sebagai contoh, (perhatikan Gambar 2.8) $W_1 = (v_1, e_1, v_2, e_6, v_3, e_5, v_2)$ merupakan sebuah perjalanan (*walk*) pada graf G. Jalan $W_2 = (v_1, e_1, v_2, e_6, v_3, e_5, v_2, e_1, v_1)$ merupakan sebuah perjalanan tertutup karena simpul awal dan simpul akhirnya sama.



Gambar 2.7 Graf G

2) Jalur *(Trail)*

Jalur (*trail*) adalah perjalanan tanpa rusuk berulang. Sebagai contoh perhatikan Gambar 2.7, $J = (v_1, e_1, v_3, e_6, v_2, e_5, v_3,)$ merupakan sebuah jalur *(trail)* pada graf G, karena tidak memuat rusuk berulang.

3) Lintasan *(Path)*

Lintasan (*path*) adalah perjalanan tanpa rusuk dan simpul berulang. Sebagai contoh perhatikan Gambar 2.7. $L = (v_1, e_1, v_2, e_6, v_2, e_4, v_4,)$ merupakan sebuah lintasan (*path*) pada graf G, karena tidak memuat simpul dan rusuk berulang.

b. Sikel (Cycle) dan Sirkuit (Circuit)

1) Sikel *(Cycle)*

Sikel (*cycle*) adalah jalur tertutup yang simpul awal dan semua simpul internalnya (simpul selain simpul awal dan simpul akhir) berbeda. Perhatikan Gambar 2.7. $S = (v_1, e_2, v_2, e_5, v_2, e_1, v_1,)$ merupakan sebuah sikel pada graf G karena simpul awal dan semua simpul internalnya berbeda. H disebut sikel Hamilton jika H adalah sikel yang memuat semua simpul dari G. $H = (v_1, e_1, v_2, e_3, v_4, e_4, v_3, e_2, v_1)$ merupakan sebuah sikel Hamilton karena sikel H memuat semua simpul pada graf G.

2) Sirkuit *(Circuit)*

Sirkuit (*circuit*) adalah jalur tertutup. Perhatikan Gambar 2.7. $C = (v_1, e_1, v_2, e_5, v_3, e_6, v_2, e_3, v_4, e_4, v_3, e_2, v_1)$ merupakan sebuah sirkuit pada graf G.

c. Lintasan Terpendek *(The Shortest Path)*

Lintasan terpendek merupakan lintasan minimum yang diperlukan untuk mencapai suatu tempat dari tempat tertentu. Pencarian lintasan terpendek ini sendiri diperlukan untuk mengurangi waktu dan biaya *(cost)* yang dikeluarkan untuk menempuh jarak menuju suatu tempat.

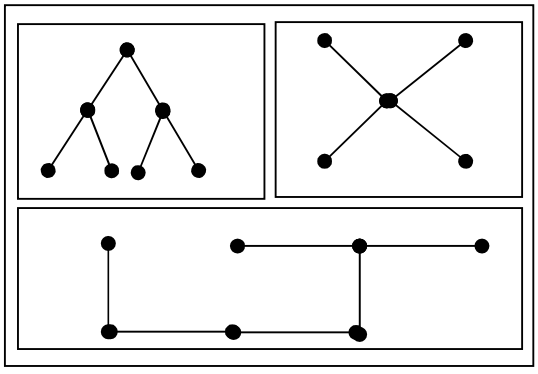
Masalah lintasan ini dapat diselesaikan dengan pendekatan graf. Graf yang digunakan adalah graf yang tidak berbobot, yaitu graf yang tidak memiliki suatu nilai atau bobot. Permasalahannya adalah bagaimana cara mengunjungi satu verteks pada graf dari verteks awal hingga verteks akhir dengan bobot minimum.

Ada beberapa macam persoalan lintasan terpendek, antara lain:

1. Lintasan terpendek antara dua buah simpul tertentu (*a pair shortets path*)
2. Lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*)
3. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*singlesource shortest path*)
4. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*)

d. Pohon *(Tree)*

Dalam graf $G = (V(G), E(G)$, sebuah lintasan (*path*) yang awal dan akhir adalah simpul (*vertex*) yang sama disebut sirkuit. Jika tidak memiliki sirkuit, maka graf tersebut dinamakan hutan *(forest)*. Dalam sebuah hutan, simpul *(vertex)* berderajat satu disebut titik akhir atau daun *(leaf)*. Hutan yang terhubung adalah pohon *(tree)*. (Joyner, Nguyen & Cohen, 2012:105)



Gambar 2.8 Contoh Tree

6. Representasi Graf Berarah pada Matriks

Misalkan G adalah graf berarah yang terdiri dari n titik tanpa garis parallel. Matriks hubung yang sesuai dengan graf G adalah matriks persegi $A_{nxn} = (a_{ij})$ (Siang, 2009:269) dengan

$$a_{ij} = \begin{cases} 1 & \text{jika ada sisi dari titik } v_i \text{ ke titik } v_j \\ 0 & \text{jika tidak ada sisi dari titik } v_i \text{ ke titik } v_j \end{cases} \quad (2.1)$$

Untuk graf berbobot, a_{ij} menyatakan bobot tiap rusuk yang menghubungkan simpul i dengan simpul j. Tanda "∞" digunakan untuk menyatakan bahwa tidak ada rusuk dari simpul i ke simpul j atau dari simpul i dengan simpul i itu sendiri, sehingga a_{ij} dapat diberi nilai tak berhingga.

B. Optimisasi *(Optimisation)*

1. Definisi Optimisasi

Optimisasi adalah suatu proses untuk mencapai hasil yang ideal atau optimal (nilai efektif yang dapat dicapai). Dalam disiplin matematika optimisasi merujuk pada studi permasalahan yang mencoba untuk mencari nilai minimal atau maksimal dari suatu fungsi nyata. Untuk dapat mencapai nilai *optimal* baik minimal atau maksimal tersebut, secara sistematis dilakukan pemilihan nilai variabel integer atau nyata yang akan memberikan solusi optimal. (Wardy, 2006:2)

2. Definisi Nilai Optimal

Nilai optimal adalah nilai yang didapat dengan melalui suatu proses dan dianggap menjadi suatu solusi jawaban yang paling baik dari semua solusi yang ada (Wardy, 2006:2). Nilai optimal dapat dicari dengan dua cara, yaitu:

1. Cara konvensional, yaitu mencoba semua kemungkinan yang ada dengan mencatat nilai yang didapat cara ini kurang efektif, karena optimasi akan berjalan secara lambat.

2. Cara kedua adalah dengan menggunakan suatu rumus sehingga nilai optimal dapat diperkirakan dengan cepat dan tepat.

3. Macam-macam Permasalahan Optimisasi

Persoalan yang berkaitan dengan optimisasi sangat kompleks dalam kehidupan sehari-hari. Nilai optimal yang didapat dalam optimisasi dapat berupa besaran panjang, waktu, jarak dan lain-lain. Berikut ini adalah beberapa persoalan yang memerlukan optimisasi:

1. Menentukan lintasan terpendek dari suatu tempat ke tempat yang lain.
2. Menentukan jumlah pekerja seminimal mungkin untuk melakukan suatu proses produksi agar pengeluaran biaya pekerja dapat diminimalkan dan hasil produksi tetap maksimal.
3. Mengatur rute kendaraan umum agar semua lokasi dapat dijangkau.
4. Mengatur *routing* jaringan kabel telepon agar biaya pemasangan kabel tidak terlalu besar.

4. Pencarian Rute Terpendek

Pencarian rute terpendek dapat dilakukan menggunakan dengan dua metode, yaitu metode konvensional dan metode heuristik. Metode konvensional diterapkan dengan cara perhitungan matematis, sedangkan metode heuristik diterapkan dengan perhitungan kecerdasan buatan dengan menentukan basis pengetahuan dan perhitungannya.

a. Metode heuristik

Heuristik adalah seni dan ilmu pengetahuan yang berhubungan dengan suatu penemuan. Kata ini berasal dari akar yang sama dalam bahasa Yunani dengan kata

"eureka", berarti 'untuk menemukan'. Ada beberapa algoritma pada metode heuristik yang biasa digunakan dalam pencarian rute terpendek. Salah satunya adalah Algoritma Semut. Metode yang tepat dalam permasalahan penelitian ini adalah metode heuristik.

b. Metode konvensional

Metode konvensional berupa algoritma yang menggunakan perhitungan matematis biasa. Ada beberapa metode konvensional yang biasa digunakan untuk melakukan pencarian rute terpendek, diantaranya Djikstraa, Floyd - Warshall, dan algoritma Bellman-Ford.

C. Travelling Salesman Problem (TSP)

Travelling Salesman Problem (TSP) adalah bagaimana menentukan rute optimal perjalanan salesman yang harus melalui semua kota tujuan tepat satu kali dan harus kembali ke kota awal (Goodaire & Parmenter, 1997:377). Masalah tersebut dapat dimodelkan ke dalam graf berbobot dengan setiap kota tujuan digambarkan sebagai titik dan ruas jalan (rusuk) sebagai rusuk berbobot mewakili panjang ruas jalan antara dua kota. Masalah perjalanan salesman tersebut dalam graf adalah mencari lintasan tertutup minimum yang memuat semua titik dalam suatu graf berbobot tersebut.

Kota dapat dinyatakan sebagai simpul graf, sedangkan rusuk menyatakan jalan yang menghubungkan antar dua buah kota. Bobot pada rusuk menyatakan jarak antara dua buah kota. Persoalan perjalanan pedagang tidak lain adalah menentukan sirkuit Hamilton yang memiliki bobot minimum pada sebuah graf terhubung.

Asumsi dasar dari model TSP adalah setiap titik hanya dilalui sebanyak satu kali, seorang harus kembali ke titik awal dia berangkat, dan jarak antar dua titik adalah jarak yang terdekat (Davendra, 2010: 7). Pada persoalan TSP ini, jika setiap simpul mempunyai rusuk ke simpul yang lain, maka graf yang merepresentasikannya adalah graf lengkap berbobot. Pada sembarang graf lengkap dengan n buah simpul $(n > 2)$, banyaknya sirkuit Hamilton yang berbeda adalah $\frac{(n-1)!}{2}$. Rumus ini dihasilkan dari kenyataan bahwa dimulai dari sebarang simpul kita mempunyai $n-1$ buah rusuk untuk dipilih dari simpul pertama, $n-2$ rusuk dari simpul kedua, $n-3$ dari simpul ketiga, dan seterusnya. Ini adalah pilihan yang independen, sehingga kita memperoleh $(n-2)!$ pilihan. Jumlah itu harus dibagi dengan 2, karena tiap sirkuit Hamilton terhitung dua kali, sehingga semuanya ada $\frac{(n-1)!}{2}$ buah sirkuit Hamilton.

D. Algoritma Semut

Penerapan awal Algoritma Semut adalah pada permasalahan TSP. TSP dipilih menjadi kasus rute terpendek pertama yang diterapkan karena TSP merupakan suatu modifikasi perilaku koloni semut buatan yang dengan mudah diadaptasi ke dalam Algoritma Semut. Selain itu, TSP juga mudah dipahami dan penguraian langkah-langkah algoritma tidak dikaburkan dengan banyak istilah teknis (Dorigo & Caro, 1999:146).

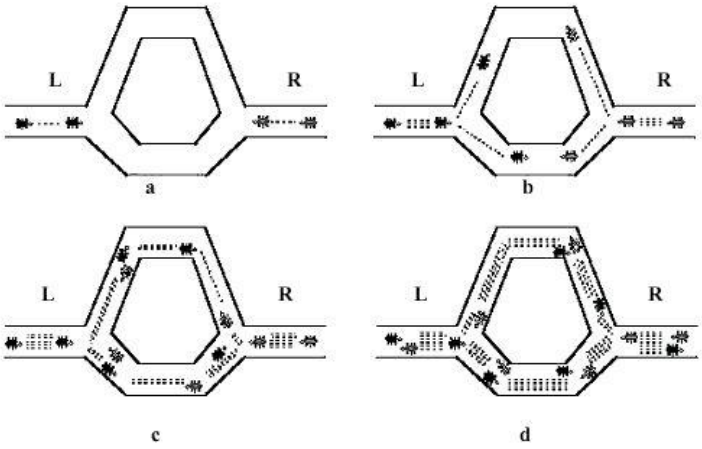
1. Sejarah Algoritma Semut

Pada tahun 1996, dunia ilmu pengetahuan pun ikut belajar dari semut dengan diperkenalkannya algoritma semut, atau *Ant Colony Optimization*, sebagai sebuah simulasi multi agen yang menggunakan metafora alami semut untuk

menyelesaikan *problem* ruang fisik. Algoritma semut diperkenalkan oleh Moyson dan Manderick dan secara meluas dikembangkan oleh Marco Dorigo. Algoritma ini menggunakan teknik probabilistik untuk menyelesaikan masalah komputasi dengan menemukan rute terbaik. Algoritma ini diambil dengan analogi oleh perilaku semut dalam menemukan rute dari koloninya menuju makanan.

2. Definisi Algoritma Semut

Algoritma semut diadopsi dari perilaku koloni semut yang dikenal sebagai sistem semut. Koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilewati. Semakin banyak semut yang melewati suatu lintasan maka semakin jelas bekas jejak kakinya. Hal ini menyebabkan lintasan yang dilalui semut dalam jumlah sedikit semakin lama semakin berkurang kepadatan semut yang melewatinya, atau bahkan akan tidak dilewati sama sekali. Sebaliknya lintasan yang dilalui semut dalam jumlah banyak semakin lama akan semakin bertambah kepadatan semut yang melewatinya atau bahkan semua semut melewati lintasan tersebut. (Dorigo, Gambardella, 1996:2)



Gambar 2.9 Perjalanan Semut Menemukan Sumber Makanan

Gambar 2.9 a menunjukkan perjalanan semut dalam menemukan rute terpendek dari sarang ke sumber makanan. Terdapat dua kelompok semut yang melakukan perjalanan. Kelompok semut L berangkat dari arah kiri ke kanan dan kelompok semut R berangkat dari kanan ke kiri. Kedua kelompok berangkat dari titik yang sama dan dalam posisi pengambilan keputusan jalan sebelah mana yang akan diambil. Kelompok L membagi dua kelompok lagi. Sebagian melewati jalan atas dan sebagian melewati jalan bawah. Hal ini juga berlaku pada kelompok R.

Gambar 2.9 b dan c menunjukkan bahwa kelompok semut berjalan pada kecepatan yang sama dengan meninggalkan *feromon* atau jejak kaki di jalan yang telah dilalui. *Feromon* yang ditinggalkan oleh kumpulan semut yang melewati jalan atas telah mengalami banyak penguapan karena semut yang melewati jalan atas berjumlah lebih sedikit dibandingkan jalan yang di bawah. Hal ini disebabkan jarak yang ditempuh lebih panjang dibandingkan jalan bawah. Sedangkan *feromon* yang berada pada bagian bawah penguapannya cenderung lebih lama. Karena semut yang melewati jalan bawah lebih banyak daripada semut yang melewati jalan atas.

Rute yang banyak dilewati oleh semut, maka akan semakin banyak pula jejak *feromon* yang ditinggalkan, sebaliknya rute yang paling sedikit dilewati oleh semut, maka akan sedikit pula *feromon* yang ditinggalkan bahkan akan menguap dan menghilang. Semut-semut akan memilih melewati rute yang banyak jejak *feromon*. Gambar 2.9 d menunjukkan bahwa semut-semut yang lain pada akhirnya memutuskan untuk melewati jalan bawah karena *feromon* yang ditinggalkan masih banyak, sedangkan *feromon* pada jalan atas sudah banyak menguap

sehingga semut-semut yang melewati rute atas semakin berkurang jumlahnya. Dari sinilah kemudian terpilihlah rute terpendek antara sarang dan sumber makanan.

3. Langkah-Langkah dalam Algoritma Semut

Beberapa langkah dalam menjalankan Algoritma Semut dijelaskan sebagai berikut:

Langkah 1:

a. Inisialisasi harga parameter-parameter algoritma semut.

Dalam algoritma semut, terdapat beberapa parameter yang perlu diinisialisasikan, yaitu:

1) Intensitas jejak feromon antar simpul (τ_{ij}) dan perubahannya ($\Delta\tau_{ij}$)

Penetapan nilai feromon awal dimaksudkan agar tiap-tiap ruas memiliki nilai ketertarikan untuk dikunjungi oleh tiap-tiap semut. τ_{ij} harus diinisialisasikan sebelum memulai siklus dan digunakan dalam persamaan probabilitas tempat yang akan dikunjungi. Nilai dari semua feromon pada awal perhitungan ditetapkan dengan angka kecil, yaitu $0 \leq \tau_{ij} \leq 1$. $\Delta\tau_{ij}$ adalah perubahan harga intensitas jejak semut. $\Delta\tau_{ij}$ diinisialisasikan setelah selesai satu siklus dan digunakan untuk memperbaharui intensitas jejak semut serta digunakan untuk menentukan τ_{ij} untuk siklus selanjutnya.

2) Banyak simpul (n) termasuk d_{ij} (jarak antar simpul)

Banyak simpul (n) merupakan jumlah simpul yang akan dikunjungi semut pada tiap siklus. Nilai n ditentukan oleh pengguna.

3) Penentuan simpul awal dan simpul tujuan

Semut yang akan melakukan tur harus memulai perjalanan dari simpul awal ke simpul tujuan. Dalam kasus TSP, simpul tujuan merupakan simpul awal.

4) Tetapan siklus-semut (Q)

Q merupakan konstanta yang digunakan dalam persamaan untuk menentukan $\Delta\tau_{ij}$ dengan nilai $Q \geq 0$.

5) Tetapan pengendali intensitas jejak semut (α)

α digunakan pada persamaan probabilitas simpul yang akan dikunjungi dan berfungsi sebagai pengendali intensitas jejak semut. Hal tersebut dimaksudkan untuk menghindari akumulasi feromon yang tidak terbatas pada rusuk tersebut. Akumulasi feromon yang tidak terbatas tidak sesuai logika karena tingkat feromon yang ditinggalkan tidak mungkin bertambah kuat tetapi akan terus berkurang. Nilai parameter α diberi nilai $0 \leq \alpha \leq 1$.

6) Tetapan pengendali visibilitas *(β)*

β digunakan dalam persamaan probabilitas simpul yang akan dikunjungi dan berfungsi sebagai pengendali visibilitas dengan nilai $\beta \geq 0$. Hal tersebut dimaksudkan untuk menghindari akumulasi yang tidak terbatas pada perhitungan visibilitas.

7) Visibilitas antar simpul (η_{ij})

Visibilitas antar simpul (η_{ij}) digunakan dalam persamaan probabilitas simpul yang akan dikunjungi. Sebelum memasuki perhitungan probabilitas dalam perhitungan algoritma semut maka terlebih dahulu dilakukan perhitungan awal

untuk menghitung visibilitas antar simpul. Visibilitas antar simpul ini bergantung pada jarak antar titik. Nilai η_{ij} diperoleh dari persamaan :

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (2.2)$$

dengan :

dij : jarak dari simpul awal ke simpul tujuan

8) Jumlah semut (m)

Jumlah semut (m) merupakan banyak semut yang akan melakukan siklus dalam Algoritma Semut. Nilai m ditentukan oleh pengguna.

9) Tetapan penguapan jejak semut (ρ)

Tetapan penguapan jejak semut (ρ) digunakan untuk memperbaharui intensitas jejak feromon semut (τ_{ij}) untuk siklus selanjutnya dan ditetapkan suatu parameter (ρ) dengan nilai $0 \leq \rho \leq 1$.

10) Jumlah siklus maksimum ($NCmax$)

Jumlah siklus maksimum ($NCmax$) bersifat tetap selama algoritma dijalankan, sedangkan τ_{ij} akan selalu diperbaharui. τ_{ij} diperbaharui pada setiap siklus algoritma mulai dari siklus pertama *(NC – 1)* sampai tercapai jumlah siklus maksimum ($NC = NCmax$) atau sampai terjadi konvergensi.

b. Pengisian kota pertama ke dalam *tabu list*

Setelah inisialisai τ_{ij} dilakukan, lalu semut pertama ditempatkan pada simpul pertama dan mulai inisialisasi untuk simpul kedua. Hasil inisialisasi simpul pertama harus diisikan sebagai elemen pertama dalam *tabu list*. *Tabu list* digunakan untuk menyimpan simpul-simpul yang telah dilewati. Hasil dari

langkah tersebut adalah terisinya elemen pertama setiap semut dengan indeks simpul pertama, yang berarti bahwa setiap $tabu_k(i)$ dapat berisi indeks semua simpul.

c. Penyusunan rute kunjungan

Penyusunan rute kunjungan dilakukan oleh koloni semut dari simpul pertama ke simpul kedua. Kemudian masing-masing semut memilih salah satu simpul dari simpul-simpul yang tidak terdapat pada tabuk sebagai simpul tujuan. Perjalanan koloni semut dilanjutkan terus menerus hingga mencapai simpul tujuan. Jika *s* menyatakan indeks urutan kunjungan, titik asal dinyatakan sebagai $tabu_k(s)$, dan titik-titik lainnya dinyatakan sebagai $\{N - tabu_k\}$, maka untuk menentukan kota tujuan digunakan persamaan probabilitas kota untuk dikunjungi sebagai berikut (Kusumadewi & Purnomo, 2005:367):

$$p_{ij}^{k} = \begin{cases} \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k\prime\in[N-Tabu_k]}(\tau_{ik\prime})^{\alpha}(\eta_{ik\prime})^{\beta}} & , \quad j \in [N - Tabu_k] \\ 0 & , \quad yang\ lain \end{cases} \tag{2.3}$$

dengan i sebagai simpul awal dan j sebagai simpul tujuan. Parameter α dan β digunakan untuk mengendalikan tingkat kepentingan relative dari jejak dan visibilitas.

d. Perhitungan panjang jalur setiap semut, pencarian rute terpendek, dan perhitungan perubahan harga intensitas feromon

1) Perhitungan panjang jalur setiap semut (L_k)

Perhitungan panjang tur setiap semut atau L_k dilakukan setelah semua semut menyelesaikan satu siklus. Perhitungan dilakukan dengan persamaan berikut:

$$L_k = d_{tabu(n),tabu(1)} + \sum_{m=1}^{n-1} d_{tabu(s),tabu(s+1)} \quad (2.4)$$

dengan $\sum_{m=1}^{n-1} d_{tabu(s),tabu(s+1)}$ adalah jarak rusuk dari titik s sampai titik $s+1$ pada tabu list yang ditempati oleh semut k, dan $d_{tabu(n),tabu(1)}$ merupakan jarak antara simpul n (akhir) dan simpul pertama (awal) pada *tabu list* yang ditempati oleh semut k.

2) Pencarian rute terpendek

Setelah perhitungan L_k tiap semut selesai, akan diperoleh rute terpendek pada setiap iterasi (L_{minNC}). Panjang rute terpendek secara keseluruhan disimbolkan dengan L_{min}.

3) Perhitungan perubahan harga intensitas feromon antar simpul ($\Delta\tau_{ij}{}^k$)

Koloni semut akan meninggalkan jejak feromon pada lintasan antar simpul yang dilaluinya. Adanya penguapan dan perbedaan jumlah semut yang lewat menyebabkan kemungkinan terjadinya perubahan harga intensitas feromon antarsimpul. $\Delta\tau_{ij}{}^k$ adalah perubahan intensitas jejak feromon antar simpul dan Q adalah tetapan siklus semut. Persamaan $\Delta\tau_{ij}{}^k$ disajikan sebagai berikut (Kusumadewi & Purnomo, 2005:366):

$$\Delta\tau_{ij}{}^k \begin{cases} \frac{Q}{L_k}, & \text{jika semut ke-k melewati pasangan (i, j)} \\ 0, & \text{jika yang lain} \end{cases} \quad (2.5)$$

e. Perhitungan harga intesitas jejak feromon untuk siklus selanjutnya dan atur ulang harga perubahan jejak feromon antar simpul.

1) Perhitungan harga intesitas jejak feromon untuk siklus selanjutnya

Intensitas jejak feromon semut antar simpul pada semua lintasan antar simpul ada kemungkinan berubah karena adanya penguapan dan perbedaan jumlah semut yang melewati. Untuk siklus selanjutnya, semut yang akan melewati lintasan tersebut harga intensitasnya telah berubah. Harga intensitas jejak kaki semut untuk siklus selanjutnya dihitung dengan persamaan:

$$\tau'_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{n} \Delta\tau_{ij}{}^{k} \qquad (2.6)$$

dengan ρ adalah tetapan penguapan jejak semut, τ'_{ij} adalah intensitas jejak feromon antarsimpul, dan $\Delta\tau_{ij}{}^{k}$ adalah perubahan intensitas jejak feromon.

2) Atur ulang harga perubahan intensitas jejak feromon antar simpul

Perubahan harga intensitas feromon antar simpul untuk siklus selanjutnya perlu diatur kembali agar memiliki nilai sama dengan nol.

f. Pengosongan tabu list

Langkah berikutnya adalah pengosongan tabu list. Tabu list dikosongkan dan langkah **b** diulangi jika NCmax belum tercapai atau belum terjadi konvergensi (semua semut hanya menemukan satu tur yang sama dengan jarak yang sama pula). *Tabu list* dikosongkan agar dapat diisi lagi dengan urutan simpul yang baru pada siklus selanjutnya. Langkah algoritma diulang lagi dari pengisian kota pertama ke dalam tabu list dengan parameter intensitas jejak feromon yang telah diperbaharui.

Perhitungan akan dilanjutkan hingga semut telah menyelesaikan perjalanannya mengunjungi tiap-tiap simpul. Hal tersebut akan diulangi hingga NCmax tercapai atau telah mencapai konvergensi.

E. Logika *Fuzzy*

Teori himpunan *fuzzy* merupakan pengembangan dari himpunan tegas. Teori ini pertama kali dikenalkan oleh Lotfi Asker Zadeh pada tahun 1965, seorang ilmuwan Amerika Serikat berkebangsaan Iran dari Universitas California di Barkeley (Klir, 1997:6).

1. Definisi Logika *Fuzzy*

Logika *fuzzy* merupakan suatu metode pengambilan keputusan berbasis aturan yang digunakan untuk memecahkan keabu-abuan masalah pada sistem yang sulit dimodelkan atau memiliki ambiguitas. Dasar logika *fuzzy* adalah teori himpunan *fuzzy*.

Menurut (Kusumadewi, 2003:3) pada himpunan tegas *(crisps)*, nilai keanggotaan suatu item *x* dalam suatu himpunan A yang dituliskan dengan μ_A(x), dimana memiliki dua buah kemungkinan nilai yaitu:

1. Satu (1), yang memiliki arti bahwa suatu item menjadi anggota dalam suatu himpunan tertentu.
2. Nol (0), yang memiliki arti bahwa suatu item tidak menjadi anggota dalam suatu himpunan tertentu.

Menurut (Kusumadewi, 2003:158) himpunan *fuzzy* memiliki dua atribut yaitu:

1. Lingustik, adalah penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami.

   Contohnya : PENDEK, SEDANG, TINGGI
2. Numeris, yakni suatu nilai (angka) yang menunjukan ukuran dari suatu variabel

   Contohnya : 140, 160, 180

Adapun beberapa alasan mengapa digunakannya logika *fuzzy* adalah (Kusumadewi, 2003:154):

1. Konsep logika *fuzzy* mudah dimengerti.
2. Penggunaan logika *fuzzy* yang fleksibel.
3. Logika *fuzzy* mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks.
4. Tidak perlu adanya proses pelatihan untuk memodelkan pengtahuan yang dimiliki oleh pakar.
5. Logika *fuzzy* didasari pada bahasa sehari-hari sehingga mudah dimengerti.

2. Fungsi Keanggotaan *Fuzzy*

Fungsi keanggotaan adalah suatu kurva yang menunjukan pemetaan titik-titik *input* data (sumbu x) kepada nilai keanggotaanya sering juga disebut derajat keanggotaan yang mempunyai interval dari 0 sampai 1. Terdapat beberapa jenis fungsi yang bisa digunakan untuk mendapatkan nilai keanggotaan dalam fungsi

keanggotaan. Menurut (Kusumadewi, 2003:160) beberapa jenis fungsi tersebut diantaranya :

a. Representasi Linear

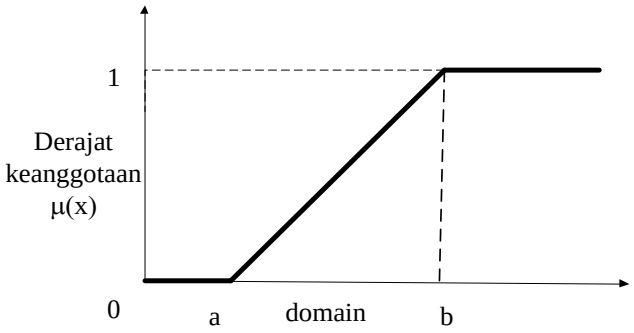
Representasi Linear adalah pemetaan input ke derajat keanggotannya digambarkan sebagai suatu garis lurus. Pada representasi linear terdapat dua jenis himpunan *fuzzy* yang linear yakni representasi linear naik dan representasi linear turun.

1) Representasi Linear Naik

Jenis yang pertama yaitu representasi linear naik. Kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol bergerak menuju nilai domain yang memiliki derajat keanggotaan yang lebih tinggi. Fungsi keanggotaan untuk kurva representasi linear naik adalah sebagai berikut (Kusumadewi, 2003:160):

$$\mu(x) = \begin{cases} 0 & ; \quad x \leq a \\ \frac{x-a}{b-a} & ; \quad a \leq x \leq b \\ 1 & ; \quad b \leq x \end{cases} \qquad (2.7)$$

Representasi grafik untuk fungsi keanggotaan linear naik ditunjukan pada Gambar 2.7 berikut:



Gambar 2.10 Representasi Linear Naik

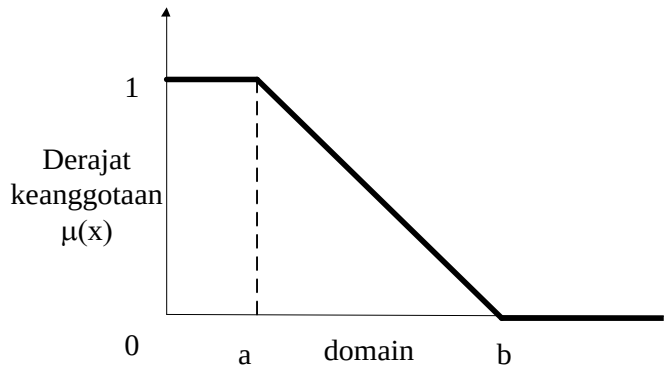
2) Representasi Linear Turun

Jenis yang kedua adalah representasi linear turun. Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi yakni satu pada sisi kiri, kemudian bergerak menuju nilai domain yang memiliki derajat keanggotaan lebih rendah (Kusumadewi, 2003:161).

Fungsi keanggotaan untuk kurva representasi linear turun adalah sebagai berikut:

$$\mu(x) = \begin{cases} 1 & ; \quad x \leq a \\ \frac{b-x}{b-a} & ; \quad a \leq x \leq b \\ 0 & ; \quad b \leq x \end{cases} \qquad (2.8)$$

Representasi grafik untuk fungsi keanggotaan linear turun ditunjukan pada Gambar 2.8 Sebagai berikut:



Gambar 2.11 Representasi Linear Turun

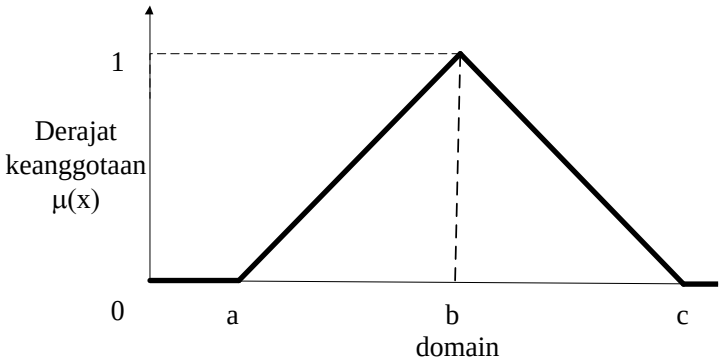
b. Representasi Kurva Segitiga (*triangular*)

Representasi kurva segitiga adalah pemetaan *input* data ke derajat keanggotaan yang digambarkan sebagai suatu kurva berbentuk segitiga. Pada dasarnya kurva segitiga merupakan gabungan antara 2 garis linear (Kusumadewi, 2003:162).

Fungsi keanggotaan untuk representasi kurva segitiga adalah sebagai berikut:

$$\mu(x) = \begin{cases} 0 & ; \quad x \leq a \\ \frac{x-a}{b-a} & ; \quad a \leq x \leq b \\ \frac{c-x}{c-b} & ; \quad b \leq x \leq c \\ 0 & ; \quad c \leq x \end{cases} \tag{2.9}$$

Representasi grafik fungsi keanggotaan segitiga di atas ditunjukan pada Gambar 2.9 Berikut:



Gambar 2.12 Representasi Kurva Segitiga

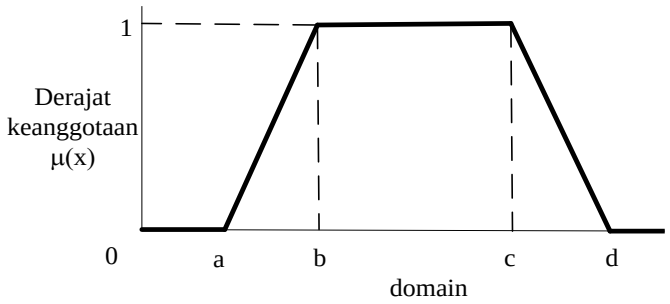
c. Representasi Kurva Trapesium

Representasi kurva trapesium merupakan pemetaan *input* data ke derajat keanggotaan yang digambarkan sebagai suatu kurva dengan bentuk trapesium. Pada dasarnya kurva trapesium ini memiliki bentuk segitiga, tetapi terdapat perbedaan pada beberapa titik yang memiliki keanggotaan 1 (Kusumadewi, 2003:163).

Fungsi keanggotaan untuk representasi kurva trapesium adalah sebagai berikut:

$$\mu(x) = \begin{cases} 0 & ; \quad x \leq a \\ \frac{x-a}{b-a} & ; \quad a \leq x \leq b \\ 1 & ; \quad b \leq x \leq c \\ \frac{d-x}{d-c} & ; \quad c \leq x \leq d \\ 0 & ; \quad d \leq x \end{cases} \tag{2.10}$$

Representasi grafik untuk fungsi keanggotaan trapesium di atas ditunjukan pada Gambar 2.10 sebagai berikut:



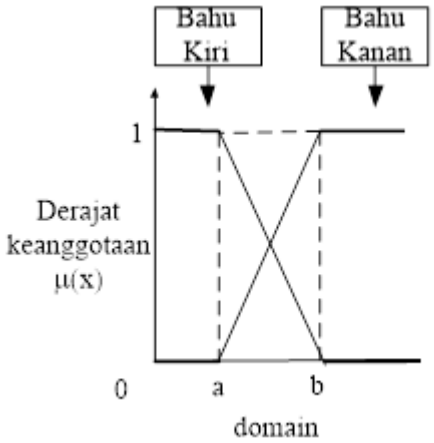
Gambar 2.13 Representasi Kurva Trapesium

d. Representasi Kurva Bentuk Bahu

Menurut Kusumadewi (2003:165) Representasi kurva bentuk bahu adalah pemetaan *input* ke derajat keanggotaan yang digambarkan sebagai suatu garis lurus yang konstan tanpa kenaikan maupun penurunan derajat keanggotaan.

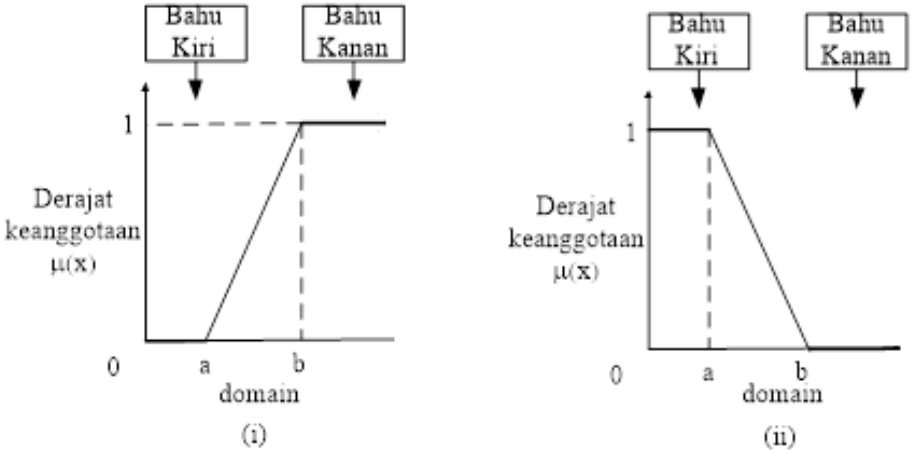
Kurva bahu tidak hanya merepresentasikan satu buah himpunan, melainkan terdiri dari beberapa himpunan. Berbeda dengan kurva linear, kurva segitiga, dan kurva trapesium yang merepresentasikan menjadi salah satu himpunan saja.

Representasi grafik untuk kurva bentuk bahu ditunjukan pada Gambar 2.14 Sebagai berikut:



Gambar 2.14 Representasi Kurva Bentuk Bahu Satu

Representasi kurva bentuk bahu pada Gambar 2.15 terdiri dari dua himpunan fungsi keanggotaan sebagai berikut:



Gambar 2.15 Representasi Kurva Bentuk Bahu Dua

Fungsi keanggotaan untuk representasi kurva bahu adalah sebagai berikut:

$$\mu(x) = \begin{cases} 0 & ; \quad x \leq a \\ 1 & ; \quad b \leq x \end{cases} \quad \text{(i)}$$

$$\mu(x) = \begin{cases} 1 & ; \quad x \leq a \\ 0 & ; \quad b \leq x \end{cases} \quad \text{(ii)} \qquad (2.11)$$

Dalam penelitian ini digunakan representasi kurva naik, turun, segitiga, dan trapezium.

3. Sistem *Fuzzy*

Sistem *fuzzy* merupakan sistem berdasarkan aturan maupun pengetahuan himpunan *fuzzy*. Sistem *fuzzy* memiliki beberapa keistimewaan (Wang, 1997:6), yaitu:

a. Sistem *fuzzy* cocok digunakan pada sistem pemodelan karena variabelnya bernilai real.

b. Sistem *fuzzy* menyediakan kerangka yang digunakan untuk menggabungkan aturan-aturan *fuzzy* Jika-Maka yang bersumber dari pengalaman manusia.

c. Terdapat berbagai pilihan dalam menentukan fuzzifier dan deffuzifier sehingga dapat diperoleh sistem *fuzzy* yang paling sesuai dengan model.

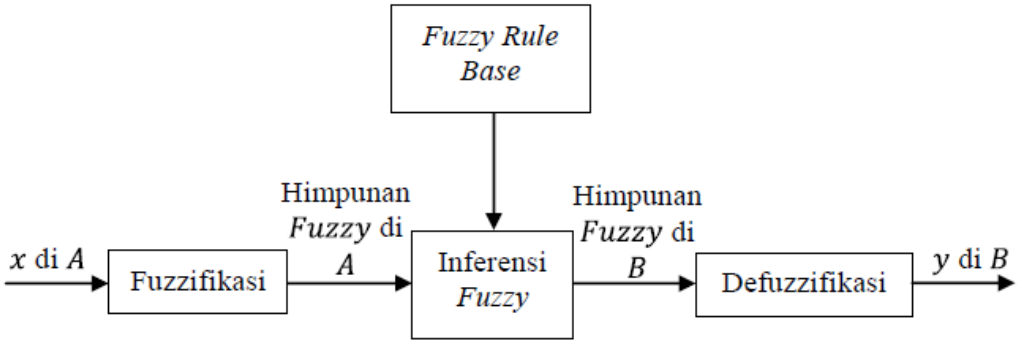
Secara umum, dalam sistem *fuzzy* terdapat empat elemen dasar (Wang, 1997:89), yaitu:

a. Basis kaidah (*rule base*), berisi aturan-aturan secara linguistik yang bersumber dari para pakar.

b. Mekanisme pengambil keputusan (*inference engine*), merupakan bagaimana para pakar mengambil suatu keputusan dengan menerapkan pengetahuan (*knowledge*).

c. Proses fuzzifikasi (*fuzzification*), yaitu mengubah nilai dari himpunan tegas ke nilai *fuzzy*.

d. Proses defuzzifikasi (*defuzzification*), yaitu mengubah nilai *fuzzy* hasil inferensi menjadi nilai tegas.



Gambar 2.16 Susunan Sistem Fuzzy (Wang, 1997)

Sistem *fuzzy* terdiri dari fuzzifikasi, pembentukan aturan (*Fuzzy rule base*), inferensi *fuzzy*, dan defuzzifikasi. Empat tahapan sistem *fuzzy* dijelaskan sebagai berikut:

a. Fuzzifikasi

Fuzzifikasi adalah pemetaan dari himpunan tegas ke himpunan *fuzzy* dengan suatu fungsi keanggotaan (Wang, 1997: 105). Melalui fungsi keanggotaan yang telah disusun maka nilai-nilai *input* tersebut menjadi informasi *fuzzy* yang selanjutnya akan digunakan untuk proses pengolahan secara *fuzzy*. Aturan *Fuzzy*

Aturan *fuzzy* merupakan inti dari suatu sistem *fuzzy*. Aturan yang yang digunakan pada himpunan *fuzzy* adalah aturan *if-then* atau Jika- Maka. Aturan *fuzzy IF-THEN* merupakan pernyataan yang direpresentasikan dengan

$$IF < proposisi\ fuzzy > THEN < proposisi\ fuzzy >$$

Proposisi *fuzzy* dibedakan menjadi dua, proposisi *fuzzy atomic* dan proposisi *fuzzy compound*. Proposisi *fuzzy atomic* adalah pernyataan single dimana x sebagai variabel linguistik dan A adalah himpunan *fuzzy* dari x. Proposisi *fuzzy*

compound adalah gabungan dari proposisi *fuzzy atomic* yang dihubungkan dengan operator "or" "and", dan "not". (Wang, 1997:62-63)

Dengan aturan *fuzzy*, pengetahuan dan pengalaman manusia dapat direpresentasikan menggunakan bahasa alami yang dikenal dengan aturan Jika-Maka (Wang, 1997:91). Aturan Jika-Maka dapat ditulis sebagai berikut:

$$Ru^i: Jika\ x_1\ adalah\ A_1^i \circ \dots \circ\ x_n\ adalah\ A_n^i\ maka\ y\ adalah\ B^i$$

dengan A_n^i dan B^i adalah himpunan $U_i \subset R$ dan $V \subset R$ sedangkan $(x_1, x_2, \dots, x_n)^T \in U$ dan $y \in V$

Dengan,

Ru^i menyatakan aturan ke-i

x_n adalah input ke-n pada himpunan U

A_n^i adalah himpunan *fuzzy* untuk input ke-n di U_i

y adalah *output* pada himpunan V

B^i adalah himpunan *fuzzy* untuk *output* di V

$\circ$ menyatakan operasi komposisi *fuzzy*, missal AND atau OR

Pernyataan yang mengikuti Jika disebut anteseden, sedangkan pernyataan yang mengikuti Maka disebut konsekuen. Untuk mendapatkan aturan Jika-Maka dapat dilakukan melalui beberapa cara, yaitu:

1) Menanyakan kepada operator manusia (ahlinya) yang mengetahui hubungan keterkaitan dari variabel-variabel yang akan dihubungkan.
2) Menggunakan algoritma pelatihan berdasarkan data-data masukan dan keluaran.

Aturan *fuzzy* terdiri dari himpunan aturan-aturan dan hubungan antar aturan dalam himpunan.

b. Inferensi *Fuzzy*

Inferensi *fuzzy* merupakan tahap evaluasi pada aturan *fuzzy*. Inferensi *fuzzy* merupakan penalaran menggunakan *input fuzzy* dan aturan *fuzzy* untuk memperoleh *output fuzzy*. Sistem inferensi *fuzzy* memiliki beberapa metode, namun yang sering digunakan dalam berbagai penelitian adalah (Kusumadewi & Purnomo, 2013:31-75):

1) Metode Mamdani

Metode Mamdani pertama kali diperkenalkan oleh Ibrahim Mamdani pada tahun 1975. Metode ini merupakan metode paling sederhana dan paling sering digunakan pada penelitian dibandingan penelitian lainnya. Inferensi metode mamdani menggunakan fungsi implikasi min, sedangkan komposisi aturannya mengunakan max. Metode mamdani sering disebut dengan metode MIN-MAX. Keluaran untuk n aturan metode mamdani didefinisikan sebagai

$$\mu_{B^k}(y) = \max_k \left[min \left[\mu_{A_1^k}(x_i), \mu_{A_2^k}(x_j) \right] \right] \qquad (2.12)$$

dengan

$k = 1,2, \dots, n,$

$\mu_{A_1^k}, \mu_{A_2^k}$ menyatakan himpunan *fuzzy* pasangan antesenden ke-k,

B^k merupakan himpunan *fuzzy* konsekuen ke-k.

2) Metode Tsukamoto

Pada metode Tsukamoto, implikasi setiap aturan berbentuk implikasi

"Sebab-Akibat" / implikasi "*Input-Output*" dimana antara anteseden dan konsekuen harus ada hubungannya. Setiap aturan direpresentasikan menggunakan himpunan-himpunan *fuzzy*, dengan fungsi keanggotaan yang monoton. Kemudian untuk menentukan hasil tegas digunakan rumus penegasan (defuzifikasi) yang disebut "Metode rata-rata terpusat" atau "Metode defuzifikasi rata-rata terpusat (*Centet Average Deffuzzyfier*)" (Abdurrahman, 2011:18).

3) Metode Sugeno

Metode Sugeno mirip dengan metode mamdani. Perbedaan kedua metode itu terletak pada fungsi keanggotaan *output*. Jika *output* dari metode mamdani masih berupa himpunan *fuzzy,* maka *output* dari metode Sugeno berupa konstanta atau persamaan linier. Metode ini pertama kali dikenalkan oleh Takagi-Sugeno Kang pada tahun 1985 (Kusumadewi, 2002: 98). Metode sugeno terbagi menjadi dua sistem yaitu orde-nol yang memiliki *output* berupa konstanta dan orde-satu yang memiliki *output* berupa persamaan linier. Defuzzifikasi metode sugeno adalah dengan cara mencari nilai rata-ratanya. Jika pada metode mamdani proses *defuzzifikasi* menggunakan agregasi daerah di bawah kuva, maka pada metode Sugeno agregasi berupa *singeleton-singeleton.*

Output dari sistem inferensi masih berupa himpunan *fuzzy*, oleh karena itu harus diubah ke himpunan tegas dengan proses defuzzifikasi.

4. Defuzzifikasi

Defuzzifikasi atau penegasan adalah fungsi yang mengubah himpunan *fuzzy* hasil dari proses inferensi *fuzzy* menjadi himpunan tegas. Nilai dari hasil

defuzzifikasi adalah *output* dari model *fuzzy*. Terdapat tiga jenis defuzzifikasi (Wang, 1997:109-112), yaitu:

a. *Center of Gravity* (COG) / Centroid

Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil titik pusat (z*) daerah *fuzzy*. Secara umum dirumuskan:

$$z^* = \frac{\int_z z\mu_z(z)dz}{\int_z \mu_z(z)dz} \tag{2.13}$$

dengan,

$\int_z$ merupakan integral biasa,

z merupakan himpunan tegas,

$\mu_z(z)$ merupakan derajat keanggotaan dari nilai tegas z.

Untuk domain diskrit dimana $\mu(z)$ didefinisikan dalam himpunan *universal* $\{z_1, z_2, z_3, \dots, z_n\}$, rumus defuzzifikasi *centroid* yang digunakan

$$z^* = \frac{\sum_{j=1}^{n} z_j \mu_z(z_j)}{\sum_{j=1}^{n} \mu_z(z_j)} \tag{2.14}$$

dengan

z_j merupakan nilai tegas ke-j

$\mu_z(z_j)$ merupakan derajat keanggotaan dari nilai tegas ke-j.

b. *Center Average Fefuzzifier* (CAD)

Defuzzifikasi ini dapat digunakan jika fungsi keanggotaan *output* dari beberapa proses *fuzzy* memiliki bentuk yang sama. Metode ini mengambil nilai rata-rata menggunakan pembobotan berupa derajat keanggotaan. Rumus yang digunakan pada defuzzifikasi ini adalah:

$$z^* = \sum \frac{z\mu_z(z)}{\mu_z(z)} \quad (2.15)$$

dengan,

z merupakan nilai tegas,

$\mu_z(z)$ merupakan derajat keanggotaan dari nilai tegas z.

c. *Maximum Defuzzier*

Secara konsep, defuzzifikasi maksimum memilih z^* sebagai titik V sehingga $\mu_z(z)$ bernilai maksimum. Didefinisikan sebagai himpunan

$$hgt(z) = \left\{ z \in V | \mu_z(z) = \sup_{z \in V} \mu_z(z) \right\} \quad (2.16)$$

dengan $hgt(z)$ merupakan himpunan semua titik di V sehingga $\mu_z(z)$ mencapai nilai maksimum. Defuzzifikasi maksimum mendefinisikan z^* sebagai titik-titik pada $hgt(z)$.

Bila $hgt(z)$ hanya memuat satu titik, maka z^* dapat langsung ditentukan. Namun bila $hgt(z)$ memuat lebih dari satu titik, maka dapat memilih salah satu dari tiga jenis defuzzifikasi maksimum, yaitu (Wang, 1997:112):

1) *Smallest of Maxima*

Solusi tegas diperoleh dengan cara mengambil nilai terkecil dari domain yang memiliki derajat keanggotaan maksimal. Dapat ditulis sebagai berikut:

$$z^* = inf\{z \in hgt(z)\}. \quad (2.17)$$

2) *Largest of Maxima*

Solisi tegas diperoleh dengan cara mengambil nilai terbesar dari domain yang memiliki derajat keanggotaan maksimal. Dapat ditulis sebagai berikut

$$z^* = sup\{z \in hgt(z)\}. \quad (2.18)$$

3) *Mean of Maxima*

Solusi tegas diperoleh dengan cara mengambil nilai rata-rata domain yang memiliki derajat keanggotaan maksimal. Dapat ditulis sebagai berikut

$$z^* = \frac{\int_{hgt(z)} z dz}{\int_{hgt(z)} dz} \quad (2.19)$$

dengan $\int_{hgt(z)}$ merupakan integral biasa untuk bagian kontinu dari $hgt(z)$ dan penyajian terakhir untuk bagian diskrit dari $hgt(z)$.

F. Toolbox *Fuzzy* pada Matrix Laboratory (Matlab)

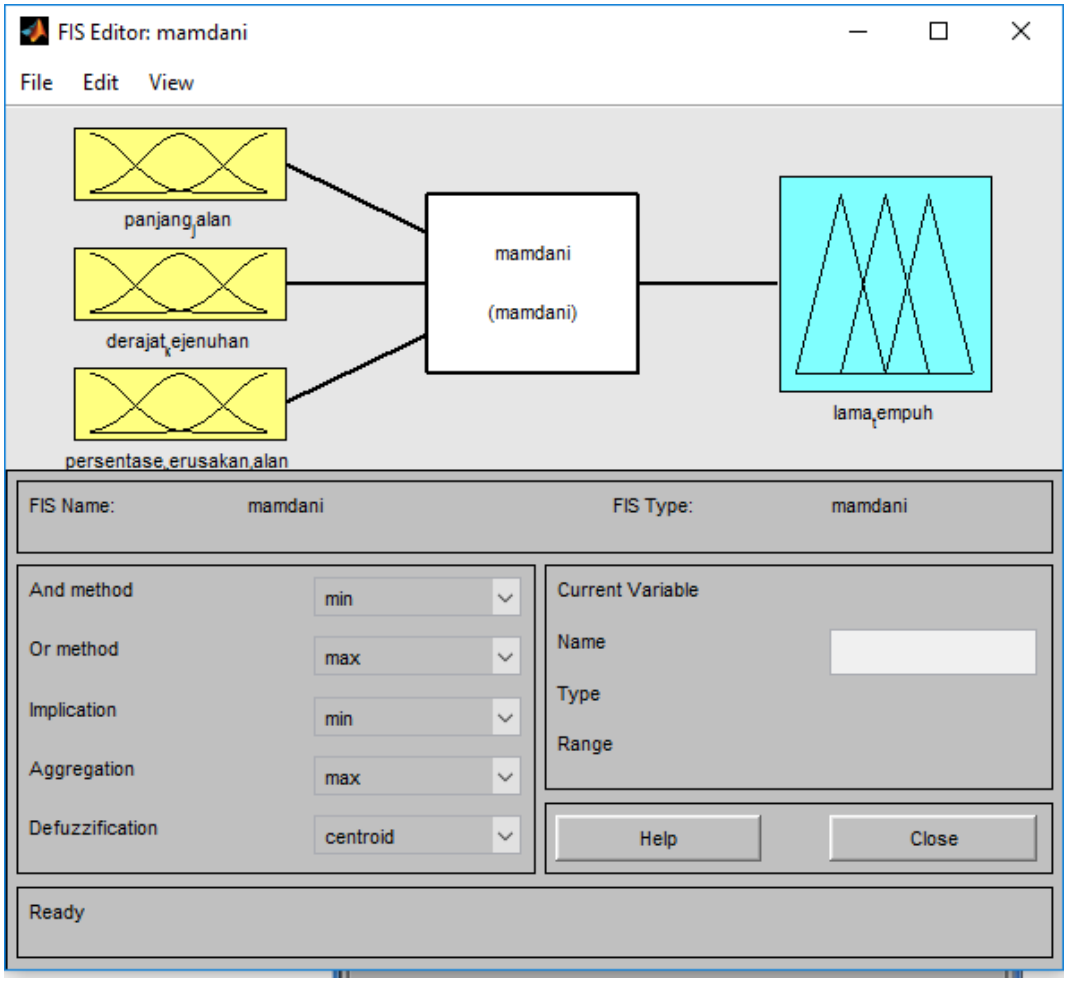
Matrix Laboratory (Matlab) merupakan perangkat lunak yang digunakan sebagai bahasa pemrograman tingkat tinggi. Matlab digunakan untuk komputasi, visualisasi dan pemrograman. Matlab banyak digunakan untuk perhitungan numerik keteknikan, komputasi simbolik, visualisasi grafis, analisis data matematis, statistika, simulasi pemodelan, dan desain GUI (Hartanto & Prasetyo, 2005:1). Matlab juga dilengkapi dengan berbagai *toolbox*. Beberapa bidang yang sudah tersedia *toolbox*nya dalam Matlab, meliputi, *fuzzy logic*, *neural network* (jaringan syaraf tiruan), *control system* (sistem kontrol, *signal processing* (pengolahan sinyal), dan *wavelet* (Naba, 2009:39).

Fuzzy logic toolbox adalah sekumpulan tool yang membantu dalam perancangan sistem *fuzzy* untuk diaplikasikan dalam berabagai bidang, seperti *automatic control*, *signal processing*, *identification system*, *pattern recognition*, *time series prediction*, *data* min*ing*, bahkan *financial applications* (Naba,

2009:79). Ada lima *tool* yang bisa digunakan pada *toolbox fuzzy* untuk membangun sistem *fuzzy*, yaitu: *Fuzzy Inference System* (*FIS*) *editor*, *membership function editor, rule editor, rule viewer,* dan *surface viewer* (Kusumadewi, 2002: 7-15), namun dalam skripsi ini hanya akan digunakan empat *tool* karena pengolahan data untuk pencarian rute terpendek menggunakan *output* yang dapat dilihat dari *rule viewer* saja. Berikut empat *toolbox fuzzy* tersebut.

1. *Fuzzy Inference System (FIS) Editor*

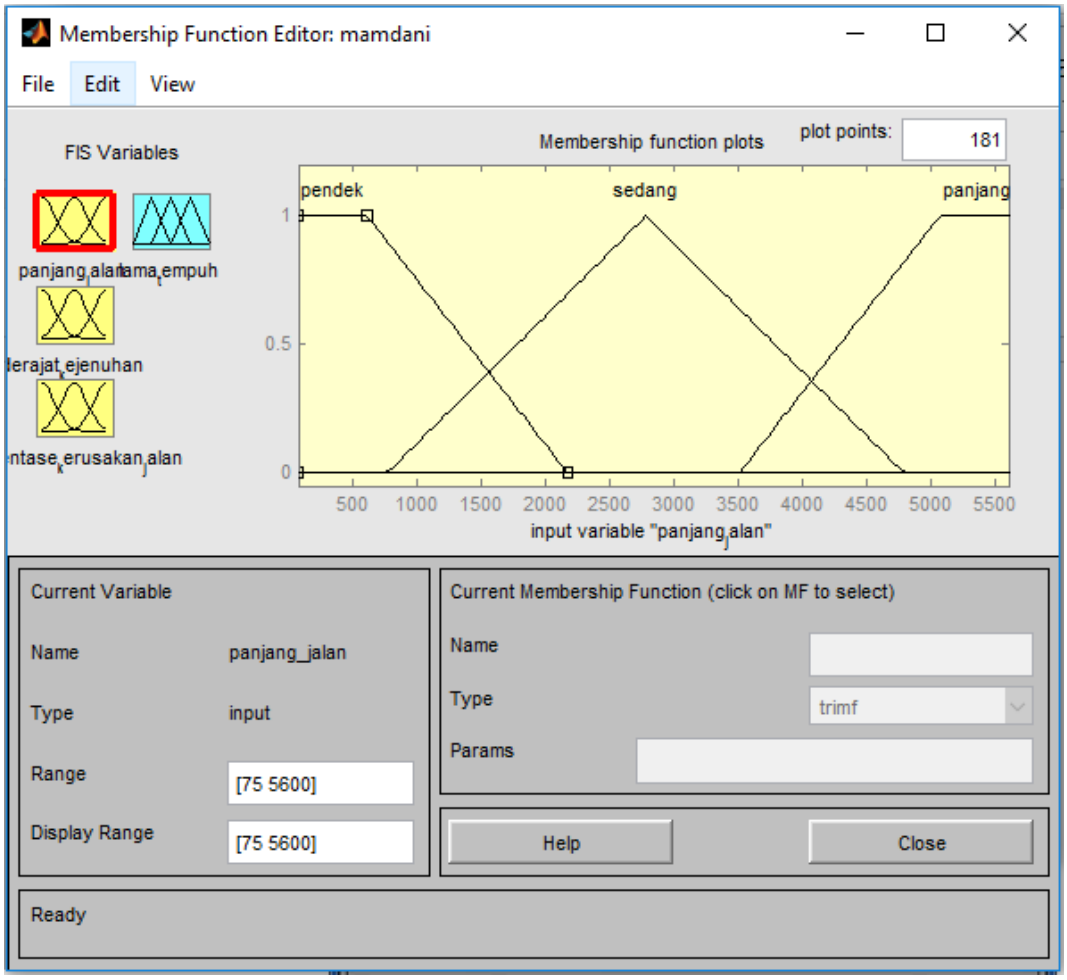
*FIS Editor m*erupakan tampilan awal *toolbox fuzzy*. *FIS Editor* dapat dipanggil dengan mengetikkan tulisan "*fuzzy*" pada *Command window*. Pada *FIS editor* hal yang harus diperhatikan adalah memilih inferensi *fuzzy* yang diinginkan. Berikut adalah tampilan *FIS editor*.



Gambar 2.17 *FIS Editor*

2. *Membership Function Editor*

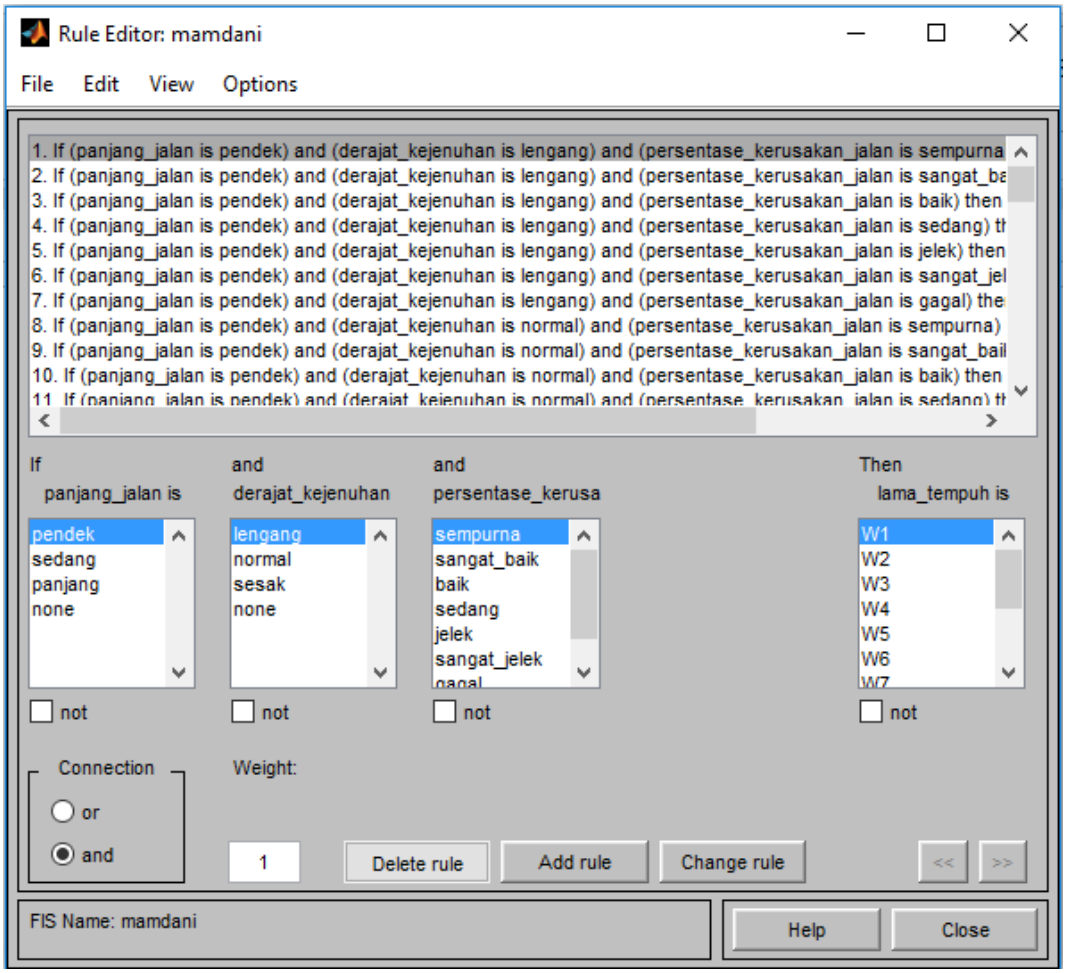
Membership function editor berfungsi untuk mengedit tiap fungsi keanggotaan pada *input* dan *output*. *Editor* ini dapat dipanggil dari *FIS Editor* dengan cara pilih *edit* → *membership function editor* atau *double click* ikon variabel *input/output*. Berikut adalah tampilan dari *membership function editor*.



Gambar 2.18 *Membership Function Editor*

3. *Rule Editor*

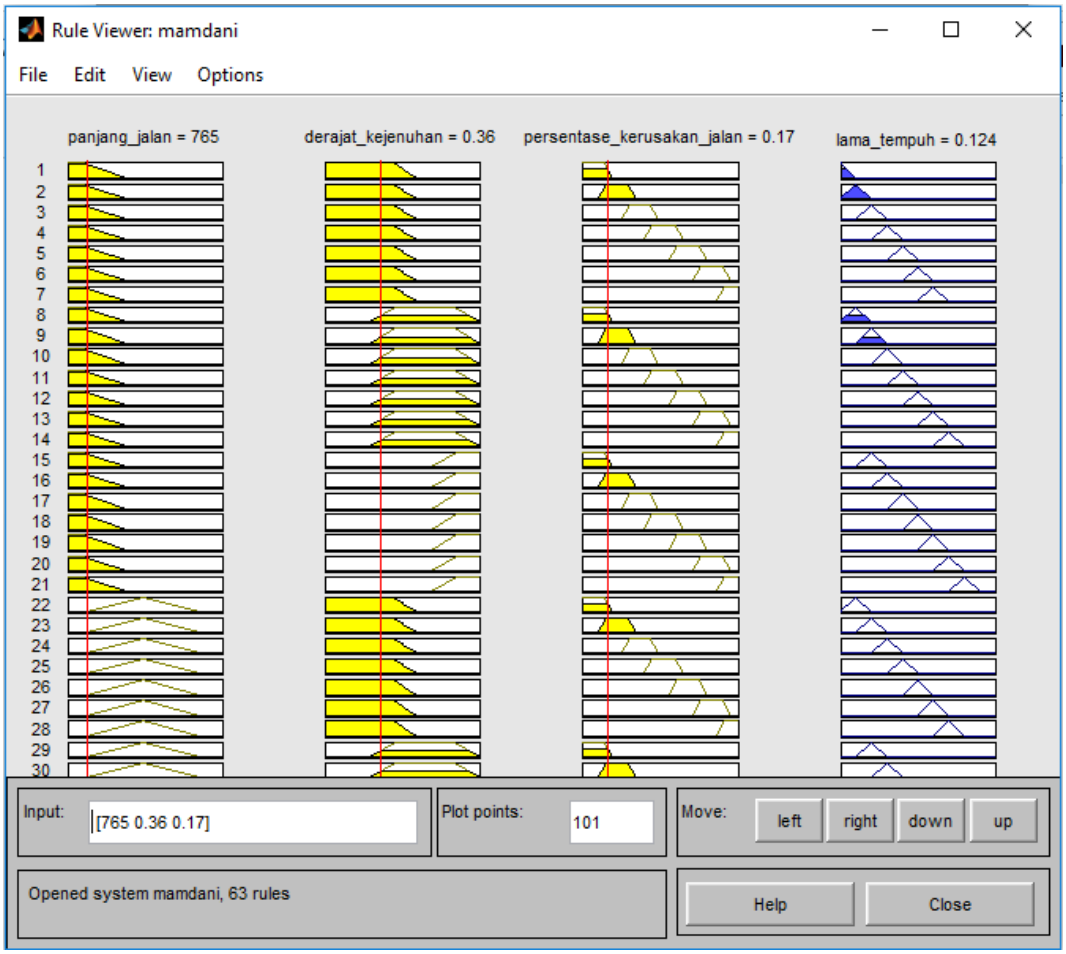
Rule editor berfungsi untuk mengedit aturan yang akan atau telah disusun. *Rule editor* dapat dipanggil dengan cara pilih *edit - rules*. *Rule* dapat mendefinisikan aturan JIKA-MAKA dengan mudah yaitu dengan mengklik sebuah item opsi nilai linguistik untuk tiap variabel *FIS*. Tampilan *rule editor* ditunjukkan pada gambar berikut:



Gambar 2.19 *Rule Editor*

4. *Rule Viewer*

Rule viewer dapat dipanggil dengan memilih menu *view → view rule. Rule Viewer* menampilkan proses keseluruhan dalam FIS. Berikut tampilan *rule viewer*.



Gambar 2.20 *Rule Viewer*

G. Jalan dan Beberapa Karakteristiknya

Jalan perkotaan merupakan segmen jalan yang mempunyai perkembangan secara permanen dan menerus, minimum pada satu sisi jalan, apakah berupa perkembangan lahan atau bukan. Termasuk jalan perkotaan yaitu jalan di atau dekat pusat perkotaan dengan penduduk lebih dari 100.000, maupun kurang dari 100.000 dengan perkembangan samping jalan yang permanen dan menerus (MKJI, 1997:5.3).

1. Kapasitas Jalan

Berdasarkan Manual Kapasitas Jalan Indonesia (1997), kapasitas adalah arus maksimum yang melewati suatu titik pada jalan bebas hambatan yang dapat dipertahankan per satuan jam dalam kondisi yang berlaku. Kapasitas suatu jalan dapat berdefinisi jumlah kendaraaan maksimum yang dapat bergerak dalam periode waktu tertentu. Kapasitas ruas jalan biasanya dinyatakan dengan kendaraan atau dalam satuan mobil penumpang (smp) per jam. Hubungan antara arus dengan waktu tempuh atau kecepatan tidaklah linear. Penambahan kendaraan tertentu pada saat arus rendah akan menyebabkan penambahan waktu tempuh yang kecil jika dibandingkan dengan penambahan kendaraan pada saat arus tinggi. Jika arus lalu lintas mendekati kapasitas, kemacetan mulai terjadi.

2. Nilai Arus Lalu-lintas

Arus lalu-lintas didalam Manual Kapasitas Jalan Indonesia didefinisikan sebagai jumlah kendaraan bermotor yang melewati suatu titik pada jalan per satuan waktu (MKJI, 1997;1.7). Nilai arus lalu lintas (Q) mencerminkan komposisi lalu-lintas, dengan menyatakan arus dalam satuan mobil penumpang (smp). Semua nilai arus lalu-lintas (per arah dan total) diubah menjadi satuan mobil penumpang (smp) dengan menggunakan ekivalensi mobil penumpang (smp) yang diturunkan secara empiris untuk tipe kendaraan berikut:

- Kendaraan ringan (LV) (termasuk mobil penumpang, minibus, pick-up) truk kecil dan jeep)
- Kendaraan berat (HV) (termasuk truk dan bus)
- Sepeda motor (MC)

Berikut tabel ekivalensi mobil penumpang yang bersumber dari Manual Kapasitas Jalan Indonesia:

Tipe jalan: Jalan tak terbagi	Arus lalu-lintas total dua arah (kend/jam)	Emp		
		HV	MC	
			Lebar jalur lalu-lintas Wc (m)	
			≤ 6	> 6
Dua-lajur tak-terbagi (2/2 UD)	0 ≥ 1800	1,3 1,2	0,5 0,35	0,40 0,25
Empat-lajur tak-terbagi (4/2 UD)	0 ≥ 3700	1,3 1,2	0,40 0,25	

Tabel 2.1. Emp untuk jalan perkotaan tak terbagi

Pada jalan perkotaan dua jalur tak terbagi dengan arus lalu-lintas total dua arahnya kurang dari 1800 kend/jam, nilai emp untuk kendaraan berat adalah 1,3, emp untuk sepeda motor yang lebar jalur lalu-lintasnya kurang dari sama dengan 6 m adalah 0,5, sedangkan yang lebar lebar jalur lalu-lintasnya lebih dari 6 m adalah 0,4. Pada jalan perkotaan dua jalur tak terbagi dengan arus lalu-lintas total dua arahnya lebih dari sama dengan 1800 kend/jam, nilai emp untuk kendaraan berat adalah 1,2, emp untuk sepeda motor yang lebar jalur lalu-lintasnya kurang dari sama dengan 6 m adalah 0,35, sedangkan yang lebar jalur lalu-lintasnya lebih dari 6 m adalah 0,25.

Pada jalan perkotaan empat jalur tak terbagi dengan arus lalu-lintas total dua arahnya kurang dari 3700 kend/jam, nilai emp untuk kendaraan berat adalah 1,3, emp untuk sepeda motor adalah 0,4. Pada jalan perkotaan empat jalur tak terbagi

dengan arus lalu-lintas total dua arahnya lebih dari sama dengan 3700 kend/jam, nilai emp untuk kendaraan berat adalah 1,2 emp untuk sepeda motor adalah 0,25.

Tipe jalan : Jalan satu arah dan jalan terbagi	Arus lalu-lintas per lajur (kend/jam)	Emp	
		HV	MC
Dua-lajur satu-arah (2/1) dan	0	1,3	0,40
Empat-lajur terbagi (4/2D)	≥ 1050	1,2	0,25
Tiga-lajur satu-arah (3/1) dan	0	1,3	0,40
Enam-lajur terbagi (6/2D)	≥ 1100	1,2	0,25

Tabel 2.2. Emp untuk jalan perkotaan terbagi dan satu arah

Pada jalan perkotaan dua lajur satu arah dengan arus lalu-lintas per lajur kurang dari 1050 kend/jam dan tiga jalur satu arah dengan arus lalu-lintas per lajur kurang dari 1100 kend/jam, nilai emp untuk kendaraan berat adalah 1,3, emp untuk sepeda motor adalah 0,4. Pada jalan perkotaan empat lajur tak terbagi dengan arus lalu-lintas total dua arahnya lebih dari sama dengan 3700 kend/jam tiga jalur satu arah dengan arus lalu-lintas per lajur lebih dari sama dengan 1100 kend/jam,, nilai emp untuk kendaraan berat adalah 1,2 emp untuk sepeda motor adalah 0,25.

3. Derajat Kejenuhan

Derajat kejenuhan (DS) didefinisikan sebagai arus lalu-lintas terhadap kapasitas, digunakan sebagai faktor utama dalam penentuan tingkat kinerja simpang dan segmen jalan (MKJI, 1997;5.19). Derajat kejenuhan dihitung dengan persamaan berikut:

$$DS = \frac{Q}{C} \quad (2.23)$$

dengan:

DS : Derajat kejenuhan

Q : Nilai arus lalu-lintas (smp/jam)

C : Kapasitas jalan (smp/jam).