BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian sebelumnya yang mendasari metode pada penelitian ini terdiri dari penelitian tentang implementasi *cloud computing* (Bittencourt dkk., 2018), pemanfaatan layanan *RESTful webservice* dan *Maps-API service* sebagai alat deteksi kondisi lalu-lintas jalan raya (Prehofer dan Gerostathopoulos, 2017; Mishra dkk., 2018). Pada penelitian *cloud computing, RESTful dan Maps-API service* telah digunakan untuk memprediksi kemacetan dari suatu jalan. Akan tetapi penelitian tersebut masih belum dapat memberikan hasil pembacaan secara kuantitatif dan belum memanfaatkan teknologi lain yang berpotensi untuk dapat diintegrasikan, maka pada penelitian ini mengintegrasikan dengan beberapa ide penelitian lain seperti penelitian tentang hubungan dan integrasi antara jarak, waktu dan kecepatan (Wilkening, 1981), implementasi dari analogi dan algoritma aliran air untuk mengatasi kemacetan lalu lintas jalan raya (Utama dkk., 2017), penelitian tentang persamaan *linear* antara variabel yang mempengaruhi dan dipengaruhi (Altman, 2015), *load-balancing* dan komputasi awan (*cloud computing*) (Fardbastani dan Sharifi, 2019), penelitian tentang penggunaan komputasi awan dan metode *rule-based* dalam memecahkan masalah yang berhubungan dengan *IoT* (Suryono dkk., 2019) dan ilmu kajian tentang kapasitas jalan raya (Fred dan Scott, 2012 ; Binamarga, 1997). Kombinasi ide-ide ini menghasilkan metode baru yang dapat digunakan untuk melakukan pengukuran kecepatan kendaraan dan evaluasi persimpangan secara *online, real-time,* mudah dan praktis. Hasil output dari pengukuran ini akan diproses dengan menggunakan sebuah sistem berbasis algoritma genetika untuk menghasilkan *intelligent traffic light*.

2.2 Dasar Teori

Dalam dunia komputasi evolusioner, dikenal adanya algoritma genetika, yaitu algoritma yang meniru prinsip dari evolusi genetika makhluk hidup, dimana selalu akan terjadi proses evolusi dan seleksi alam, setiap individu yang mempunyai sifat genetik (nilai *fitness*) terbaik akan bertahan hidup sementara yang lainnya akan gugur, individu-individu tersebut akan saling bersilang dan bermutasi untuk menghasilkan individu baru dengan sifat genetika baru dari kromosom hasil persilangan atau mutasi yang mempunyai nilai *fitness* tertentu (Gen dan Lin, 2007). Algoritma genetika banyak dipakai untuk mencari nilai optimal menurut kondisi tertentu. Algoritma ini sangat cocok untuk menyelesaikan masalah-masalah *non-linier* dengan tingkat kompleksitas tinggi (Mokshin dkk., 2019), Algoritma ini juga sudah berkembang dengan sangat pesat dan banyak diimplementasikan juga pada berbagai permasalahan machine learning (Petroski dkk., 2017), termasuk dalam bidang penjadwalan yang paling optimal berdasarkan rule dan penggolongan tertentu untuk diimplementasikan pada *traffic light* (Odeh dkk., 2015), akan tetapi pemanfaatan algoritma genetika pada penelitian-penelitian sebelumnya belum terintegrasi dengan metode lain yang juga berpotensi untuk dapat digunakan dalam menyelesaikan masalah penjadwalan *traffic light.*

TCP/IP adalah protokol yang digunakan untuk komunikasi jaringan internet. Protokol ini adalah protokol terbesar yang digunakan untuk komunikasi data di seluruh dunia. Arsitektur *TCP/IP* mempunyai perkembangan yang sangat pesat dan menjadi protokol utama yang digunakan dalam komunikasi *Internet of Things (IoT)* (Shang dan Droms, 2016). *Cloud, fog* dan *edge computing* merupakan teknologi yang saat ini banyak digunakan untuk implementasi I*nternet of Things (IoT)*, karena teknologi ini memilik banyak keunggulan yaitu sangat fleksibel, mudah dikembangkan, dan dapat terintegrasi dengan berbagai macam sistem lain (Bittencourt dkk., 2018; Tao dkk., 2019). *Cloud computing* adalah teknologi yang menjadikan internet sebagai pusat pengelolaan data dan aplikasi. *Cloud computing* dapat berupa server yang terhubung secara *online* (Ibrahim dan Hemayed, 2019), sedangkan *fog computing* merupakan infrastruktur komputasi yang terdistribusi pada sistem *IoT* untuk memperluas platform komputasi. *Fog computing* ini

terhubung langsung dengan *edge computing,* sedangkan *edge computing* adalah jaringan *micro* yang memproses atau menyimpan data secara lokal, dan terhubung langsung dengan instrumen. Hal tersebut dikendalikan melalui *microcontroller* (Bittencourt dkk., 2018; Alamgir Hossain dkk., 2018).

REST (Representational State Transfer) webservice merupakan protokol komunikasi data dan integrasi antar sistem yang berbeda-beda dengan menggunakan protokol *web.* Arsitektur berbasis *REST* mempunyai banyak keunggulan dan dapat digunakan secara luas*,* diantaranya adalah penerapan *resource oriented architectures*, seperti sistem *caching* dan prinsip *loose-coupling* antar bagian dalam sistem yang diintegrasikan. Prinsip *loose coupling integration* ini juga memberikan banyak kemudahan untuk melakukan *update* komponen, pengalihan maupun penerapan berbagai teknologi alternatif. *REST-ful webservice* juga sangat cocok untuk penerapan integrasi dan komunikasi data pada sistem berbasis *FPGA (Field-Programmable Gate Array)* maupun IoT yang heterogen dengan berbasis *cloud* (Ruiz, 2017).

MQTT (Message Queuing Telemetry Transport) protokol merupakan sebuah protokol yang berjalan diatas diatas *TCP/IP* dan digunakan untuk sharing data antar perangkat, *MQTT* pada dunia *IoT* saat ini adalah protokol yang sangat populer dan banyak digunakan untuk melaporkan kondisi pada sensor pada *edge computing* dan menerima perintah eksekusi dari *fog* maupun *cloud computing,* prinsip kerja dari *MQTT* adalah diatur melalui *MQTT Broker*. Perangkat lain akan berfungsi sebagai *subscriber*. Semua *subscriber* yang tergabung dalam topik dan subtopik yang sama akan saling menerima pesan yang dikirimkan (Kashyap dkk, 2018).

Perangkat dalam sistem ini saling terhubung melalui media *internet* dan berkomunikasi dengan menggunakan *REST-ful Webservice* dan *MQTT.* Penggunaan *REST-ful Webservice* dan *MQTT* mempunyai banyak keunggulan, dan merupakan protokol integrasi dan komunikasi *IoT* yang banyak digunakan saat ini. dan kepadatan arus lalu-lintas di jalan dapat dipantau dengan memanfaatkan jaringan *internet* yang mengkombinasikan antara teknologi *cloud, fog, edge computing,* dan *RESTful web service* (Mishra dkk, 2018*).* Penelitian ini merupakan kombinasi

dengan teknologi *cloud computing* berbasis algoritma genetika yang tersambung dengan sistem IoT melalui protokol *MQTT* dan *web service.*

Komputasi awan (cloud computing) banyak digunakan dalam sistem berbasis perangkat lunak sebagai layanan. Dalam banyak penelitian sebelumnya komputasi awan diterapkan dalam analisa *big data* dan implementasi layanan *RESTful webservice* diberbagai bidang termasuk *smartcity* dan transportasi (Bittencourt dkk., 2018). *RESTful webservice (Representational State Transfer)* adalah protokol komunikasi data dan integrasi antara berbagai sistem yang dapat digunakan secara luas dengan memanfaatkan protokol *HTTP/Web,* banyak keunggulan yang ditawarkan oleh arsitektur integrasi berbasis *RESTful,* diantaranya yaitu *caching system* dan prinsip *loose coupling* antara bagian-bagian dalam sistem terintegrasi sehingga dapat memberikan kemudahan dalam menciptakan arsitektur berbasis layanan, (Suryono dkk., 2019), *RESTful* sangat cocok untuk mengintegrasikan data pada sistem yang heterogen dengan berbasis *cloud* (Prehofer dan Gerostathopoulos, 2017). *Google distance matrix API-Service* adalah layanan *Map-API* yang disediakan oleh *Google*, layanan ini memberikan informasi jarak dari dua titik di jalan, durasi waktu yang diperlukan kendaraan untuk melintas dari titik awal ke titik tujuan secara *real-time* dan memperkirakan durasi rata-rata kendaraan yang melewati jalan berdasarkan data historis (Mishra dkk., 2018). Dalam penelitian sebelumnya tentang penerapan algoritma aliran air dalam optimasi rekayasa lalu lintas jalan menganalogikan bahwa lalu lintas kendaraan di jalan seperti air yang mengalir dalam pipa atau media saluran air, oleh karena itu pemodelan aliran air dapat diterapkan untuk memecahkan masalah yang terkait dengan lalu lintas jalan (Utama dkk., 2017).

Penelitian ini menggunakan teori-teori tentang algoritma genetika beserta implementasi dan pengembangannya, *cloud, fog* dan *edge computing, RESTful web service, IoT dan MQTT.* Karena penelitian ini bertujuan untuk memecahkan masalah dibidang transportasi, maka penelitian ini juga membahas sedikit tentang teori transportasi yang berkaitan dengan kapasitas, monitoring, evaluasi jalan raya dan teori tentang *traffic light.*

Dalam implementasi algoritma genetika, melibatkan teori tentang hubungan kecepatan, jarak dan waktu, hubungan antar variabel yang saling berpengaruh secara linear dan teori tentang penggunaan perhitungan rata-rata dalam *load balancing.* Implementasi dalam dunia transportasi, meliputi penggunaan teori tentang penghitungan kecepatan arus bebas, hubungan antara kecepatan kendaraan, *flow*/arus kendaraan yang melintas, kepadatan (*density*), dan kemacetan lalu-lintas, serta teori tentang *traffic light.*

2.2.1 Teori Kecepatan, *Linear Regression* dan *Load Balancing*

Penelitian yang meneliti hubungan antara kecepatan, jarak dan waktu memiliki konsep dasar dan menunjukkan hubungan seperti yang ditunjukkan pada rumus 2.1 (Wilkening, 1981).

$$v = \frac{d}{t} \tag{2.1}$$

dimana v adalah kecepatan kendaraan, d adalah jarak, t adalah durasi waktu tempuh.

Metode *simple linear regression* merupakan metode statistik yang digunakan untuk menentukan sejauh mana hubungan sebab akibat antara variabel penyebab terhadap variabel akibatnya secara linear. Secara formula metode ini dapat dirumuskan seperti pada persamaan 2.2.

$$y = a + bx + c \tag{2.2}$$

dimana y adalah variabel yang merupakan akibat (*dependent variable*), x adalah variabel faktor penyebab (*independent variable*), a adalah suatu konstanta, b adalah koefisien regresi (kemiringan) dan c adalah variabel acak lain yang mempengaruhi (Altman dan Krzywinski, 2015).

Dalam penelitian tentang *load-balancing* membahas bagaimana sumber daya dapat dibagi dengan seimbang untuk mendapatkan kinerja maksimum dari

suatu sistem (Fardbastani dan Sharifi, 2019). Prinsip pembagian beban adalah mengambil nilai rata-rata yang dapat dirumuskan dengan rumus 2.3.

$$\bar{a} = \frac{(\sum_{x=1}^{n} x)}{n} \tag{2.3}$$

dimana ā adalah nilai rata-rata x1..xn, n adalah jumlah entitas.

2.2.2 Kecepatan Kendaraan dan Prediksi Kemacetan

Pada penelitian tentang prediksi kemacetan dapat disimpulkan, kecepatan rata-rata kendaraan yang melintas di jalan memiliki hubungan dengan tingkat indikator kemacetan. Untuk mengukur kapasitas jalan dan tingkat kepadatan/kemacetan secara lebih akurat harus mengacu pada Manual Kapasitas Jalan Raya (*Highway Capacity Manual/HCM*) dari masing-masing negara (Hamad dan Kikuchi, 2002). Pada kasus prediksi kemacetan secara umum, dalam kondisi normal dan tidak terjadi hujan memiliki skala indikator kemacetan 0-100 dibandingkan dengan skala kecepatan 0-120 km / jam. Perbandingan antara tingkat indikator kemacetan dan kecepatan kendaraan diperlihatkan dalam rumus 2.4 (Hamad dan Kikuchi, 2002).

$$j = \frac{100}{120}.v \tag{2.4}$$

Dimana *j* adalah indikator kemacetan secara umum dari lalu lintas, *v* adalah kecepatan kendaraan sekarang, v adalah kecepatan rata-rata kendaraan pada ruas jalan.

2.2.3 Kecepatan Arus Bebas Kendaraan

Kecepatan arus bebas kendaraan adalah kecepatan rata-rata yang akan dipilih pengemudi apabila jalan dalam keadaan kosong, atau kecepatan kendaraan yang tidak dipengaruhi oleh kendaraan lain yang melintas di jalan. Penghitungan

kecepatan arus bebas pada dasarnya ditentukan oleh kecepatan arus dasar kendaraan ringan (*LV*) dan dipengaruhi oleh faktor-faktor lain yang dapat berbeda-beda disetiap negara, (Fred dan Scott, 2013). Pada penelitian ini menggunakan formula yang ditetapkan oleh Manual Kapasitas Jalan Raya (Highway Capacity Manual) yang berlaku di Indonesia. Penghitungan kecepatan arus bebas dalam penelitian ini ditunjukkan pada formula 2.5 (Binamarga, 1997).

$$F_V = (FV_O + FV_W) \times FFV_{SF} \times FFV_{CS} \quad (2.5)$$

F_V adalah kecepatan arus bebas kendaraan ringan *(LV)* dalam km/jam, FV_O adalah kecepatan arus dasar kendaraan ringan *(LV)* dalam km/jam, FV_W adalah penyesuaian lebar jalur lalu lintas efektif (km/jam), FFV_{SF} adalah faktor penyesuaian hambatan samping dan lebar bahu atau jarak penghalang, FFV_{CS} adalah faktor penyesuaian untuk ukuran kota. Pada jalan di perkotaan kecepatan arus bebas dasar (FV_O) dapat dilihat pada tabel 2.1. Penghitungan nilai *Fv* ini sangat diperlukan untuk melakukan prediksi penghitungan *flow* kendaraan yang melewati suatu ruas jalan.

Tabel 2.1 Tabel kecepatan arus bebas dasar (FV_O)

Tipe Jalan	**Kecepatan rata-rata kendaraan (km/jam)**
Enam lajur terbagi (6/2 D) atau tiga lajur satu arah	57
Empat lajur terbagi (4/2 D) atau 2 lajur 1 arah	55
4 lajur tidak terbagi	51
4 lajur terbagi atau 2 lajur 1 arah	42

Nilai penyesuaian lebar jalur efektif FV_W dapat dilihat pada tabel 2.2.

Tabel 2.2 Penyesuai lebar jalur efektif

Tipe jalan	Lebar jalur lalu-lintas efektif (Wc) dalam meter	FVw
Empat lajur terbagi atau jalan satu arah	Per lajur : 3,00 3,25 3,50 3,75 4,00	 -4 -2 0 2 4
Empat lajur tidak terbagi	Per lajur : 3,00 3,25 3,50 3,75 4,00	 -4 -2 0 2 4
Dua lajur tidak terbagi	Total : 5 6 7 8 9 10 11	 -9,5 -3 0 3 4 6 7

Faktor penyesuai kecepatan arus bebas juga dipengaruhi oleh hambatan samping (FFV_{SF}), nilai dari faktor penyesuai ini dibagi dalam 2 jenis jalan, yaitu jalan dengan bahu seperti pada tabel 2.3 dan jalan dengan kereb penghalang seperti pada tabel 2.4.

Tabel 2.3 Nilai FFV_{SF} untuk jalan dengan bahu

Tipe jalan	Kelas Hambatan Samping (SFC)	Faktor penyesuai untuk hambatan samping dan lebar bahu jalan			
		Lebar bahu rata-rata efektif (Ws) dalam meter			
		≤ 0,5m	1,0m	1,5m	≥2m
Empat lajur terbagi (4/2D)	Sangat rendah	1,02	1,03	1,03	1,04
	Rendah	0,98	1,00	1,02	1,03
	Sedang	0,94	0,97	1,00	1,02

	Tinggi	0,89	0,93	0,96	0,99
	Sangat tinggi	0,84	0,88	0,92	0,96
Empat lajur tak terbagi (4/2 UD)	Sangat rendah	1,02	1,03	1,03	1,04
	Rendah	0,98	1,00	1,02	1,03
	Sedang	0,93	0,97	0,99	1,02
	Tinggi	0,87	0,93	0,94	0,98
	Sangat tinggi	0,80	0,86	0,90	0,95
Dua lajur tidak terbagi (2/2 UD) atau jalur satu arah	Sangat rendah	1,00	1,01	1,01	1,01
	Rendah	0,96	0,98	0,99	1,00
	Sedang	0,90	0,93	0,96	0,99
	Tinggi	0,82	0,86	0,90	0,95
	Sangat tinggi	0,73	0,79	0,85	0,91

Tabel 2.4 Nilai FFV_{SF} untuk jalan dengan kereb penghalang

Tipe jalan	Kelas Hambatan Samping (SFC)	Faktor penyesuai untuk hambatan samping dan lebar bahu jalan			
		Lebar bahu rata-rata efektif (Ws) dalam meter			
		≤ 0,5m	1,0m	1,5m	≥2m
Empat lajur terbagi (4/2D)	Sangat rendah	1,00	1,01	1,01	1,02
	Rendah	0,97	0,98	0,99	1,00
	Sedang	0,93	0,95	0,97	0,99
	Tinggi	0,97	0,90	0,93	0,96
	Sangat tinggi	0,81	0,85	0,88	0,92
Empat lajur tak terbagi (4/2 UD)	Sangat rendah	1,00	1,01	1,01	1,02
	Rendah	0,96	0,98	0,99	1,00
	Sedang	0,91	0,93	0,96	0,98
	Tinggi	0,84	0,87	0,90	0,94
	Sangat tinggi	0,77	0,81	0,85	0,90
Dua lajur tidak terbagi (2/2 UD)	Sangat rendah	0,98	0,99	0,99	1,00
	Rendah	0,93	0,95	0,96	0,98
	Sedang	0,87	0,89	0,92	0,95

atau jalur satu arah	Tinggi	0,78	0,81	0,84	0,88
	Sangat tinggi	0,68	0,72	0,77	0,82

Sedangkan nilai dari FFV_{CS} atau faktor penyesuaian untuk ukuran kota, dapat dilihat pada tabel 2.5.

Tabel 2.5 Faktor penyesuai ukuran kota *(FFV_{CS})*

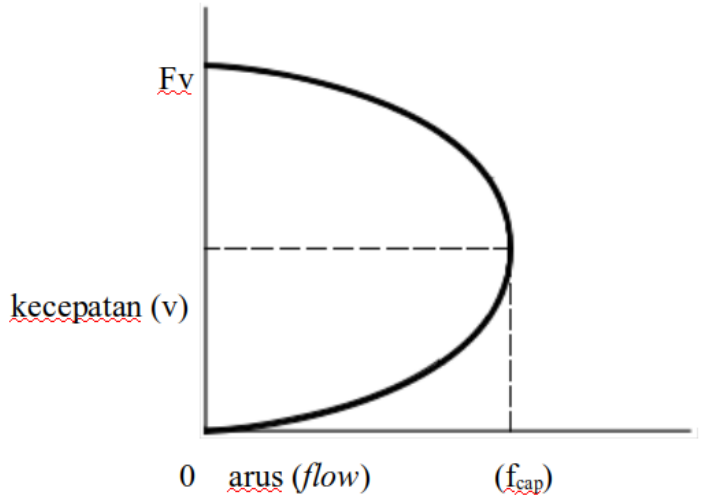
Jumlah penduduk (dalam juta)	Faktor penyesuaian untuk ukuran kota
< 0,1	0,9
0,1 - 0,5	0,93
0,5 – 1,0	0,95
1,0 – 3,0	1,00
> 3,0	1,03

2.2.4 Kecepatan dan Arus Kendaraan (*Flow*)

Arus kendaraan menunjukkan jumlah kendaraan yang lewat pada suatu jalan per satuan waktu. Kendaraan terbagi atas kendaraan ringan (*LV*), kendaraan berat (*HV*), dan sepeda motor (*MC*). Arus kendaraan (flow) mengacu standart pada mobil penumpang atau kendaraan ringan (*LV*). Hubungan antara arus kendaraan dan kecepatan kendaraan pada suatu jalan raya ditunjukkan oleh rumus 2.6 (Fosu dkk, 2020).

$$f = j_{max}\left(v - \frac{v^2}{F_v}\right) \tag{2.6}$$

Dimana f adalah arus kendaraan (*flow*) yang bernilai antara 0 sampai dengan kapasitas flow dari jalan (f_{cap}), j_{max} adalah kapasitas kepadatan jalan, v adalah kecepatan rata-rata kendaraan yang melintas, dan F_v adalah kecepatan bebas pada ruas jalan. Sehingga membentuk suatu grafik seperti ditunjukkan pada gambar 2.1 (Fosu dkk, 2020).



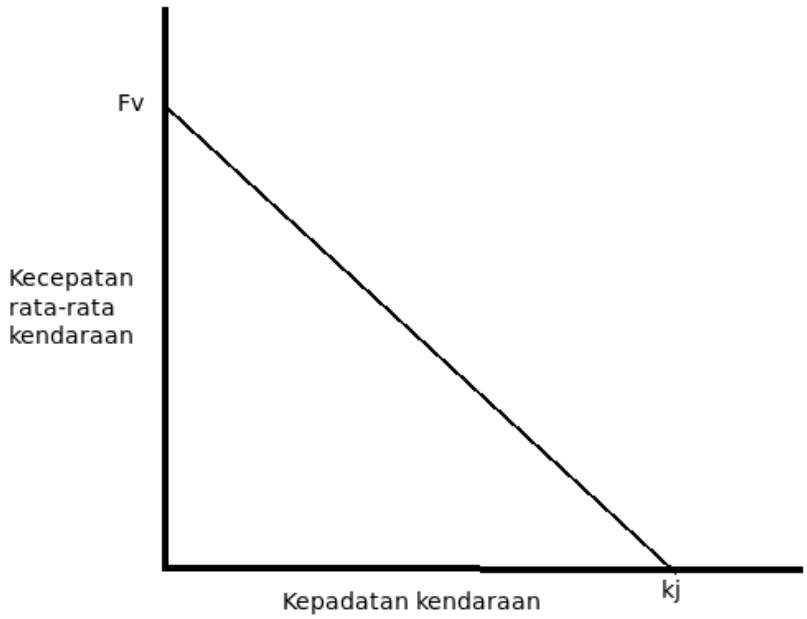
Gambar 2. 1 Grafik hubungan antara kecepatan dan arus

2.2.5 Kecepatan dan Kepadatan Jalan (*Density*)

Antara kepadatan kendaraan dan kecepatan rata-rata kendaraan yang melintas di jalan raya mempunyai hubungan yang linear (Fred dan Scott, 2013), artinya semakin melambat kecepatan rata-rata kendaraan yang melintas, maka hal itu menunjukkan bahwa tingkat kepadatan kendaraan pada jalan tersebut semakin tinggi. Persamaan 2.7 menunjukkan hubungan antara kecepatan dan kepadatan kendaraan.

$$v_x = F_v\left(1 - \frac{k}{k_j}\right) \tag{2.7}$$

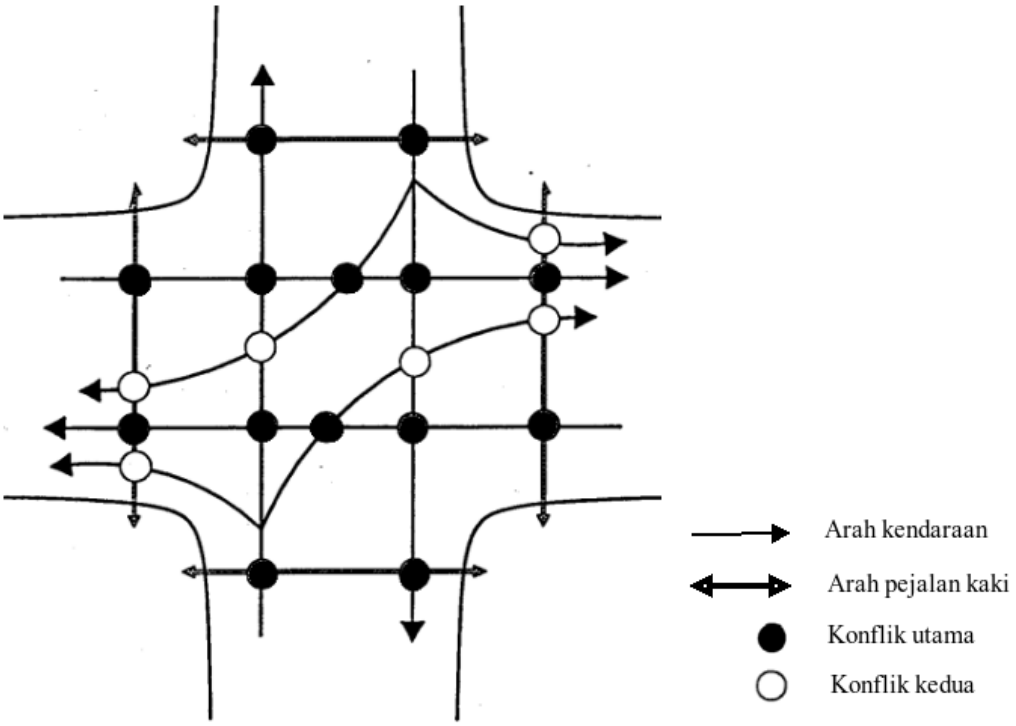
Dimana v_x adalah kecepatan rata-rata kendaraan yang melintas, Fv adalah kecepatan arus bebas kendaraan ringan (*LV*), k adalah kepadatan kendaraan di jalan (jumlah kendaraan per satuan luas), k_j adalah jumlah kendaraan jenuh (jumlah kendaraan per satuan luas). Grafik relasi antara kecepatan dan kepadatan jalan dapat dilihat pada gambar 2.2.



Gambar 2.2 Grafik hubungan antara kecepatan dan kepadatan kendaraan

2.2.6 Sinyal Lalu-lintas (*Traffic Light*)

Penerapan sinyal *traffic light* bertujuan untuk memisahkan lintasan gerakan lalu-lintas yang saling bertentangan dalam dimensi waktu, sehingga tidak terjadi konflik dan dapat mengurai kepadatan kendaraan sehingga suatu ruas jalan dapat dimanfaatkan secara optimal. Hal ini tidak lepas dari sistem penjadwalan dan pengaturan arus lalu-lintas disuatu ruas jalan. Penjadwalan sinyal *traffic light* membentuk suatu siklus yang terdiri atas beberapa fase. Penghitungan sinyal harus disesuaikan dengan kondisi lalu-lintas, arah dan kapasitas kepadatan jalan. Secara umum sinyal pengaturan lalu-lintas menggunakan lampu tiga warna (merah, kuning dan hijau). Arah mobilisasi lalu-lintas yang datang dari jalan-jalan yang saling berpotongan (konflik utama) dan gerakan membelok dari arah lurus yang berlawanan atau arah dari pejalan kaki (konflik kedua) merupakan faktor penting yang harus diakomodir dalam menentukan penjadwalan dalam setiap fase. Pemetaan arah dan konflik (konflik utama dan konflik kedua) pada sebuah persimpangan dapat dilihat pada gambar 2.3 (Binamarga, 1997).



Gambar 2.3 Pemetaan arah dan konflik persimpangan

Parameter-parameter yang menjadi penentu dalam pengaturan sinyal *traffic light* adalah fase, waktu siklus, waktu hijau, waktu hijau maksimum, waktu hijau minimum, rasio hijau, waktu merah semua, waktu kuning, periode antar hijau dan waktu hilang, seperti yang dijelaskan pada tabel 2.6.

Tabel 2.6 Parameter dalam *traffic light*

Variabel	***Parameter***	**Keterangan**
i	Fase	Fase merupakan bagian dari siklus dengan lampu hijau yang disediakan bagi kombinasi tertentu arah lintasan lalu-lintas. *i* menunjukkan indeks dari nomor fase (fase 1, fase 2 fase 3 dan seterusnya.)
c	Waktu siklus	Waktu total dari urutan lengkap indikasi sinyal disemua fase (*c* dalam detik)

g	Waktu hijau	Waktu nyala lampu hijau dalam fase (g dalam detik)
g_{max}	Waktu hijau maksimum	Waktu hijau maksimum yang diijinkan dalam satu fase (g_{max} dalam detik)
g_{min}	Waktu hijau minimum	Waktu hijau minimum yang diperlukan dalam satu fase (g_{min} dalam detik)
GR	*Green Ratio*/Rasio Hijau	Rasio/perbandingan antara durasi waktu hijau terhadap durasi waktu siklus ($GR = g/c$)
ALL-RED	Waktu merah semua	Waktu dimana sinyal merah menyala semua secara bersamaan diantara peralihan dua fase yang berurutan. (*ALL-RED* dalam detik)
AMBER	Waktu kuning	Waktu dimana sinyal kuning dinyalakan saat peralihan dari sinyal hijau ke merah. (*AMBER* dalam detik)
IG	Antar hijau	Periode kuning+merah semua antar dua fase yang berurutan (*IG* dalam detik)
LTI	Waktu hilang	Jumlah durasi total semua periode antar hijau dalam satu siklus lengkap. Waktu hilang dapat juga diperoleh dari beda antara waktu siklus lengkap dan jumlah total waktu hijau dalam satu siklus. (*LTI* dalam detik)

2.2.7 Komputasi Awan (*Cloud Computing*)

Komputasi awan bekerja dengan cara menggabungkan antara pemanfaatan teknologi komputer (*computing*) dan pengembangan teknologi berbasis *internet* yang merupakan metafora dari awan. Dalam balik *Cloud Computing* terdapat abstraksi dari infrastruktur kompleks yang tersembunyi (Botta, 2016). *Cloud computing* tersaji dalam bentuk layanan (*as a service*) yang dapat diakses lewat *internet (*dianalogikan sebagai awan/*cloud*) tanpa harus mengetahui secara pasti apa yang ada didalamnya, ahli dengannya, atau memiliki kendali terhadap infrastruktur teknologi yang membantunya (Diaby dan Rad, 2017).

Komputasi awan merupakan suatu konsep yang meliputi Perangkat Lunak Sebagai Layanan (*Software As A Service*), *Web 2.0* dan teknologi terbaru lainnya secara luas yang berbasis internet dengan tujuan untuk memenuhi kebutuhan

komputasi pengguna. *Cloud computing* memiliki banyak keunggulan, diantaranya adalah skalabilitas, kemudahan akses, kreasi yang lebih luas, memberikan kemudahan dalam ketersediaan data, dan meningkatkan tingkat keamanan apabila diimplementasikan secara benar.

Pemanfaatan *cloud computing* saat ini sudah berevolusi sedemikian jauh dan menjadi sangat luas. Teknologi ini mengubah banyak hal diberbagai bidang, namun secara esensial *cloud computing* mempunyai karakteristik mempunyai jaringan yang luas, layanannya dapat termonitor atau terkendali dengan baik, layanannya bersifat secara otomatis secara dinamis sesuai dengan kebutuhan, bersifat fleksibel dan dinamis, dapat digunakan secara tersebar diberbagai lokasi secara mandiri (Diaby dan Rad, 2017).

Karakteristik utama dari *cloud computing* adalah penggunaan atau distribusi *resource,* menghubungkan dan mengintegrasikan antar infrastruktur, antar pengguna secara *online* dan *real-time* tanpa terkendala jarak dan waktu, menggunakan jaringan internet untuk menjalankan teknologi virtualisasi, sangat fleksibel dalam skalabilitas sehingga sangat mudah untuk menyesuaikan dengan perkembangan yang terjadi, Memberikan banyak efisiensi pada manajemen dan interaksi layanan (Escamilla dkk, 2018).

2.2.8 Model dan Cara Kerja Komputasi Awan (*Cloud Computing*)

Komputasi awan (*cloud computing*) bekerja dengan cara menyimpan, mengolah dan mereplikasi data dengan menggunakan sistem yang terabstraksi melalui teknologi internet (*cloud*). Segala macam bentuk proses komputasi dilakukan pada sistem *cloud,* sehingga pada setiap komputer klien yang terhubung ke *cloud system* tidak perlu memasang suatu aplikasi tertentu yang melakukan tugas komputasi. Tugas komputasi pada sistem *cloud* bisa dari komputasi yang bersifat umum dan ringan hingga tugas komputasi berat seperti analisis data yang kompleks atau menjalankan proses algoritma genetika. *Cloud system* saat ini didominasi oleh pengolahan data dan penyajian informasi dengan *interface* berbasis *web. Interface* ini menjadi penghubung antara klien dari *cloud system* dengan *engine* yang

melakukan proses komputasi hingga menghasilkan sebuah *output* yang hendak dicapai. Model layanan dari *cloud computing* adalah dalam bentuk :

1. *Infrastructure as a Service (IaaS)*

   Model *IaaS* menyediakan layaan dalam bentuk sumber daya komputasi (*computer resouces*) diantaranya adalah server, jaringan, media penyimpanan (*storages*), *data center space* dan infrastruktur lainnya yang berbasis internet.

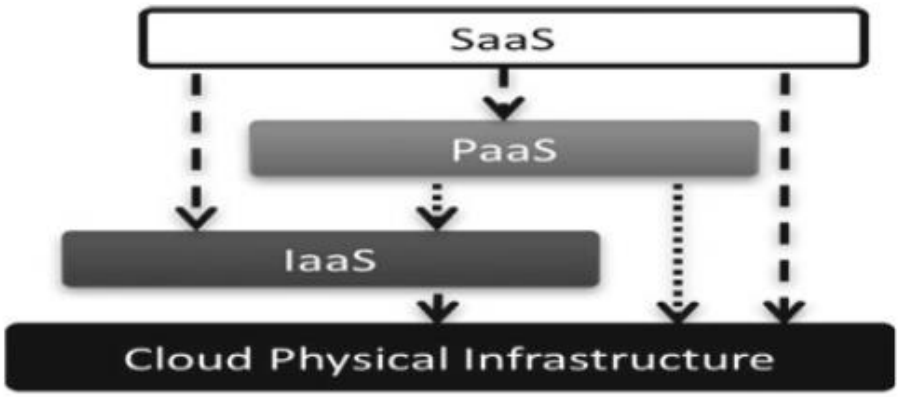
2. *Platform as a Service (PaaS)*

   Model ini menyediakan segala macam layanan yang dibutuhkan oleh pengguna dalam lingkungan sistem *cloud* untuk menjalankan aplikasi (biasanya berbasis web) tanpa harus memikirkan perangkat keras, perangkat lunak dan infrastruktur lain yang dibutuhkan.

3. *Software as a Service (SaaS)*

   Model ini menjalankan suatu aplikasi di komputer *cloud* yang sudah siap untuk digunakan oleh pengguna secara langsung melalui jaringan internet. *SaaS* biasanya berbentuk aplikasi atau layanan jadi berbasis web.

Secara hirarki *IaaS* berjalan secara langsung diatas infrastruktur fisik, *PaaS* berjalan diatas *IaaS* dan infrastruktur fisik, sedangkan *SaaS* berjalan diatas *Paas, IaaS* dan infrastruktur fisik dari *cloud computing* (Escamilla dkk., 2018). Gambar 2.4 menunjukkan model layanan dalam *cloud computing.*



Gambar 2.4 Model layanan *cloud computing*

2.2.9 Komputasi Kabut (*Fog Computing*)

Dalam implementasi *IoT* penggunaan *cloud computing* secara langsung mempunyai banyak keterbatasan, maka *fog computing* ini didesain untuk menjadi jembatan yang menghubungkan antara layanan *cloud* dengan perangkat-perangkat yang terhubung. Dalam penelitian ini, secara spesifik *fog computing* difungsikan sebagai penghubung antara *cloud* dan *edge computing.* Komputasi kabut (*Fog computing*) merupakan model komputasi dengan cara desentralisasi infrastruktur. Pada *fog computing* sumber daya yang digunakan (*resource*) dialokasikan diantara sumber data dan *cloud system* atau *data center* lainnya. Secara implementasi *fog computing* bekerja dengan cara memperluas komputasi dan layanan *cloud* ke tepi jaringan. Tujuan dari *cloud computing* adalah untuk menjalankan komputasi dan pengambilan data secara otomatis dan *real-time* tanpa terkendala oleh keterbatasan lebar pita kecepatan akses (*bandwidth*) maupun kualitas *latency* dari jaringan *internet. Fog computing* banyak digunakan pada sistem *Internet of Things (IoT),* jaringan sensor, *real-time data analytics* dan integrasi dengan berbagai teknologi (Suryono dkk., 2019; Tao dkk., 2019).

2.2.10 *Edge Computing*

Edge computing merupakan proses komputasi yang bekerja dengan cara memproses lalu-lintas *Internet of Things (IoT)* dan penyimpanan data sedekat mungkin dari sumber data ke pusat data sehingga dapat mengurangi latensi dan penggunaan *bandwidth internet*. *Edge computing* memindahkan proses dari *cloud* ke tepi (*edge)* mendekati instrumen *IoT* yang bertujuan untuk mengurangi proses komunikasi antara *client* dan *server cloud* secara terus menerus melalui jaringan *internet.* Tujuan utama dari *edge computing* ini adalah untuk meningkatkan skalabilitas dari sebuah sistem *IoT* (Hossain dkk, 2018).

Edge Computing seringkali juga merupakan perluasan dari arsitektur *fog computing* yang menggunakan pusat data lokal untuk memperluas jangkauan jaringan *cloud*. Ketika proses pengumpulan data, *edge computing* akan menentukan apakah harus dikirim kembali ke *server* pusat untuk diproses atau diproses pada

node edge computing secara lokal. Proses ini bertujuan untuk merampingkan kinerja jaringan dan memastikan bahwa perangkat lokal dapat menganalisis dan menerapkan data lebih cepat dan mengatur jumlah lalu lintas *bandwidth* yang mengalir ke dan dari inti jaringan. Manfaat utama *Edge Computing* yang paling besar adalah memungkinkan *cloud computing* untuk membangun jaringan *internet of things* (*IoT*). *Edge Computing* memungkinkan data yang dihasilkan oleh perangkat *IoT* diproses lebih dekat dari tempatnya sehingga tidak harus melewati rute panjang menuju *cloud computing*. Proses komputasi yang dilakukan ditepi jaringan memungkinkan terjadinya proses analisa data secara real time. Sama dengan *fog computing, edge computing* sangat ideal dalam berbagai keadaan. Salah satunya adalah ketika sebuah sistem *IoT* memilik koneksi jaringan yang buruk. Akan tidak efisien jika perangkat IoT harus selalu terhubung dengan *cloud*. Penggunaan *edge computing* lainnya adalah berkaitan dengan sistem yang sensitif terhadap kualitas *latency* jaringan. *Edge computing* dapat mengurangi *latency*, karena data tidak harus melintasi jaringan ke *data center* atau *cloud* secara terus menerus untuk dapat diproses (Hossain dkk, 2018).

2.2.11 *Extreme Programming*

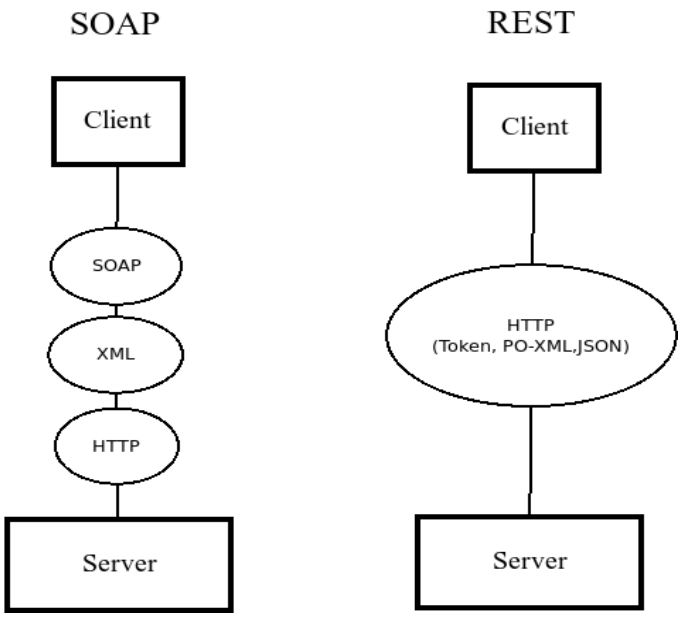
Extreme Programming (XP) adalah sebuah metode atau model pengembangan sistem/perangkat lunak dengan cara menyederhanakan berbagai tahapan dalam proses pengembangan tersebut sehingga menjadi lebih *adaptif* dan fleksibel. *XP* bukan hanya berfokus pada penulisan kode program (*coding*) tetapi meliputi seluruh area pengembangan perangkat lunak. *XP* mengambil pendekatan 'ekstrim' dalam bentuk pengembangan secara *iterative* dan melibatkan berbagai pihak *stakeholder* dalam pengembangannya.

2.2.12 Web services, Integrasi dan *MQTT*

Web services merupakan mekanisme komunikasi *client-server* dengan menggunakan protokol HTTP/HTTPS (*Hypertext Transfer Protocol/Secure*) melalui jaringan berbasis *TCP/IP.* Komunikasi ini dilakukan antar host mesin yang telah disepakati sebelumnya dan bersifat lintas *platform,* sehingga berbagai jenis

perangkat dapat terintegrasi dan saling berkomunikasi dengan menggunakan *web services.*

Secara umum terdapat 2 jenis *web services,* yaitu *SOAP (Simple Object Access Protocol),* dan REST (*Representational State Transfer*). *SOAP* adalah sebuah spesifikasi protokol untuk pertukaran pesan/informasi terstruktur dalam implementasi layanan web di jaringan komputer berbasis *TCP/IP*. Format pesan SOAP menggunakan XML (*Extensible Markup Language*) dan berjalan bergantung pada protokol layer aplikasi lainnya, seperti *HyperText Transfer Protocol/Secure (HTTP/S), Simple Mail Transfer Protocol (SMTP)*, *File Transfer Protocol (FTP)* dan lain sebagainya untuk transmisi dan negosiasi. Pesan yang dikirim berupa *XML* dengan spesifikasi yang sudah terstandarisasi dengan format tertentu. Pada *RESTful web service* juga menggunakan layanan *web* (*HTTP/HTTPS*) dalam melakukan pertukaran informasi, akan tetapi *RESTful* lebih bersifat fleksibel apabila dibandingkan dengan *SOAP.* Pada *RESTful web service, server* menyediakan sumberdaya (*resources*) yang dapat diakses oleh *client* dengan cara mengakses sebuah *URI (Uniform Resource Identifier)* atau *ID* tertentu, sedangkan pertukaran informasi dilakukan antara *client* dan *server* dalam format *JSON/XML. RESTful* mempunyai keunggulan apabila dibandingkan dengan *SOAP* karena bentuknya yang lebih sederhana, ringkas, mudah dipelajari dan tidak membutuhkan layer pertukaran pesan tambahan, sehingga prosesnya akan menjadi lebih cepat dan mudah untuk dikembangkan atau diintegrasikan dengan sistem lain (mempunyai skalabilitas yang baik). Mekanisme keamanan dalam *RESTful* biasanya dalam bentuk token, setelah proses authentikasi komunikasi langsung terbentuk dan terjadi pertukaran data antara *client* dan *server.* Perbandingan antara SOAP dan REST dapat dilihat pada gambar 2.5 (Malik, 2017).



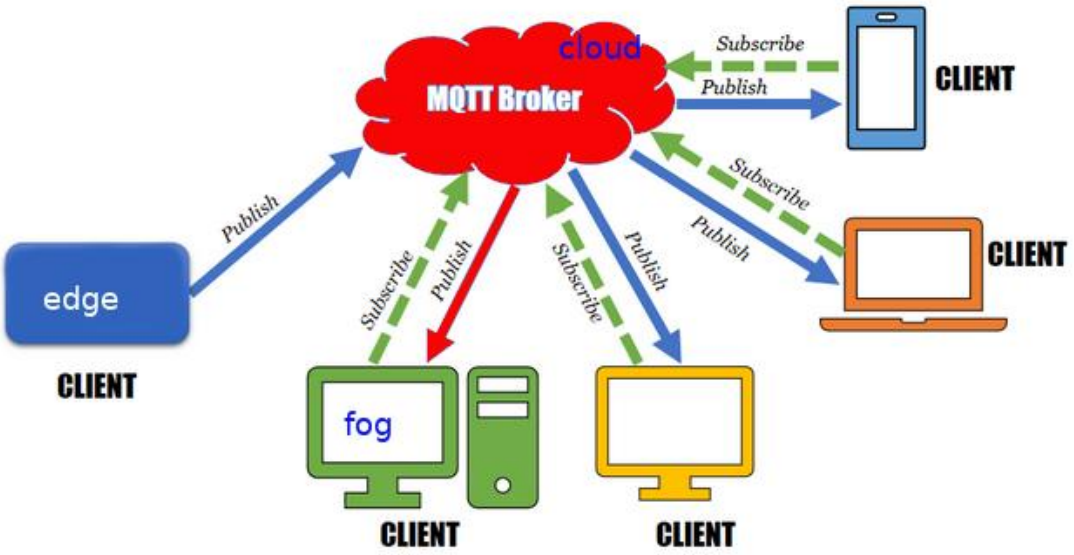
Gambar 2.5 Perbandingan *REST* dan *SOAP*

MQTT adalah protokol pengiriman pesan (*messaging)* antar perangkat yang sangat ringan dan cepat. Protokol ini banyak digunakan pada *IoT,* aplikasi web, aplikasi *mobile,* dan komunikasi lintas peralatan. Protokol MQTT dapat dijalankan pada berbagai macam *platform* sistem operasi dan aplikasi yang dibangun dari berbagai macam bahasa pemrograman. Hampir semua bahasa pemrograman *IoT* mendukung protokol ini.

MQTT bekerja dengan cara menggunakan pola *publish-subscribe.* Setiap *node* dalam jaringan MQTT harus bergabung (*subscribe*) ke server MQTT dengan topik dan subtopik tertentu. *Node* yang tergabung dalam topik dan subtopik yang sama akan dapat saling mempublikasikan dan menerima pesan. Pada jaringan *IoT, MQTT* digunakan sebagai sarana komunikasi antar *node (messaging)* yang tergabung dalam suatu jaringan. Sistem *messaging* ini berguna untuk mengirimkan perintah ke perangkat tertentu, maupun mengambil status pada perangkat tertentu (*M2M/Machine-to-Machine Communication*). Pada sistem *IoT* yang terintegrasi, protokol *MQTT* dipasang pada *cloud computing*, *fog computing* maupun *edge computing.* Pada gambar 2.6 memberikan ilustrasi sebuah jaringan *MQTT* yang

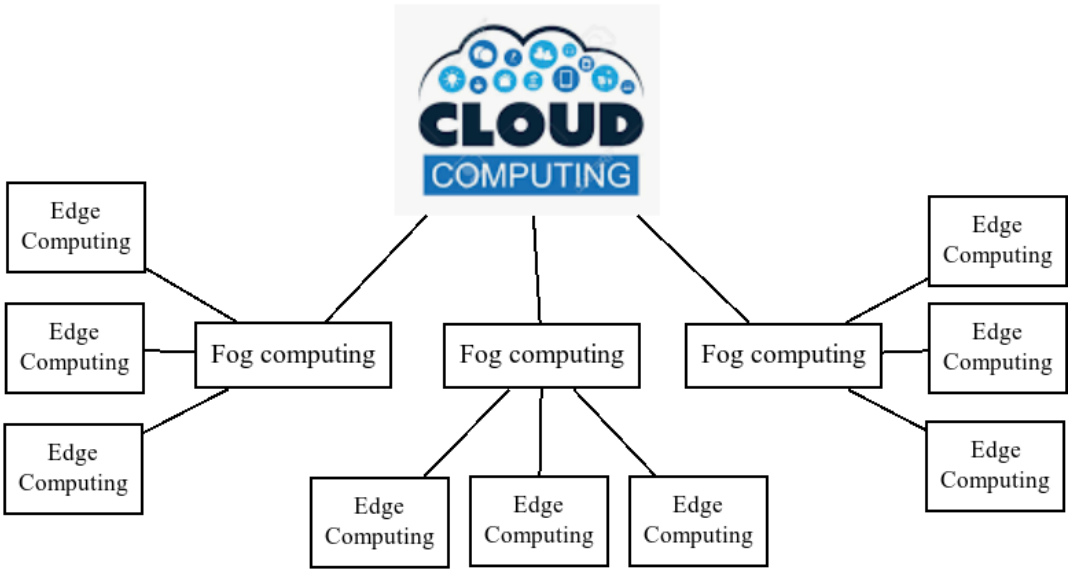
tergabung dalam suatu topik dan subtopik yang sama (*subscribe*), sedang menerima pesan yang dikirim (*publish*) oleh *edge computing.*

Protokol dan spesifikasi standart dari *MQTT* ditetapkan dalam konsorsium OASIS (*Organization for the Advancement of Structured Information Standards*), dan sejak *MQTT* versi 3.1.1 telah ditetapkan dalam *International Organization for Standardization/International Electrotechnical Commisiion 20922* (ISO/IEC 20922).



Gambar 2.6 Komunikasi *MQTT*

Sistem *Cloud, fog dan edge computing* yang menerapkan protokol *MQTT* pada dasarnya merupakan suatu teknologi terintegrasi yang sudah diterapkan dibanyak model sistem *Internet of Things (IoT)* (Kashyap dkk, 2018). Integrasi antara *cloud, fog* dan *edge computing* dapat dilihat pada gambar 2.7.



Gambar 2.7 Integrasi *Cloud fog* dan *edge computing*

2.2.13 Sumber Data

Pada sistem *intelligent traffic light***,** ada banyak kemungkinan sumber yang bisa dijadikan sebagai data input bagi sistem *intelligent traffic light* (Liu dkk., 2018; Mishra dkk., 2018) data input ini akan menjadi masukan bagi sistem *cloud* yang kemudian akan diolah menjadi banyak hal yang berguna untuk menciptakan sistem lalu-lintas cerdas. Akan tetapi pada penelitian ini hanya akan fokus pada penggunaan *map and travelling API* berbasis *Google Distance Matrix API-Service (Mishra dkk., 2018)*.

Google Maps-API merupakan API (*Application Programming Interface*) yang disediakan oleh *Google* yang bertujuan untuk memberikan layanan peta. *Google Maps-API* menggunakan protokol standart *webservice,* sehingga dapat diakses secara langsung dalam halaman website dengan menggunakan *javascript container* dan *HTML* dengan tujuan untuk menyajikan data spasial secara langsung pada suatu halaman situs. Akan tetapi dalam penelitian ini *Google Maps-API* diakses dengan menggunakan modul *googlemaps* dari bahasa pemrograman *python,* hal ini bertujuan agar data yang diperoleh dari *Google Maps-API* masih berupa data mentah yang dapat diolah lebih lanjut kedalam *cloud system engine.*

Penggunaan *Google Maps-API* membutuhkan *Web API-Key* yang kita dapat kita peroleh melalui akun *google* yang kita miliki. Setiap *Web API-Key* yang kita miliki diperuntukan spesifik hanya untuk satu aplikasi, sehingga kita membutuhkan beberapa *key* untuk beberapa aplikasi yang kita kembangkan.

2.2.14 Penggunaan *Map* and *Travelling API* pada *Intelligent Traffic Light*

Integrasi dengan layanan *map-API* yang terhubung dengan aplikasi mobile berbasis *GPS* memungkinkan sistem ini untuk dapat mengambil keputusan lebih cerdas. Saat ini terdapat banyak layanan *API* yang sudah tersedia di internet yang berkaitan dengan rute, waktu kedatangan dan semua yang berkaitan dengan perjalanan. Dengan memasukkan koordinat *origin* dan *destination* yang dijadikan sebagai input *API-Service*, maka jarak, estimasi waktu kedatangan yang didapatkan dari *API Service* tersebut dapat dijadikan sebagai input untuk diproses menjadi kecepatan (Király dan Abonyi, 2015). Data kecepatan tersebut kemudian di olah menjadi prediksi tingkat kemacetan, kualitas pengaturan lalu-lintas dalam persimpangan dan menentukan durasi yang tepat bagi *traffic light*.

Pada penelitian sebelumnya yang serupa tentang *intelligent traffic light. Google distance matrix MAPS-API* digunakan sebagai alat untuk mendeteksi tingkat kemacetan dengan cara menghitung tingkat kewajaran durasi perjalanan dari suatu titik ke titik lainnya. Hasil dari durasi perjalanan diklasifikasikan menjadi 3, yaitu tidak ada kemacetan, tingkat kemacetan sedang, dan tingkat kemacetan tinggi. Data dan perbandingan tingkat kemacetan tersebut kemudian digunakan untuk membentuk perbandingan durasi fase-fase dalam satu siklus *intelligent traffic light* (Mishra dkk., 2018), akan tetapi sayangnya penelitian tersebut belum menerapkan metode *machine learning* dan algoritma yang menentukan penjadwalan, maupun evaluasi terhadap suatu persimpangan jalan. Penjadwalan fase dan siklus *traffic light* dalam penelitian tersebut juga belum tentu mencapai titik terbaik dan tidak terukur secara pasti, sehingga masih perlu untuk dikembangkan.

2.3 Penyelesaian Terbaik dan Optimasi

Sebuah permasalahan terkadang mempunyai penyelesaian yang lebih dari satu solusi. Solusi terbaik merupakan target yang dicari dalam penelitian ini, sehingga diperlukan suatu algoritma pencarian nilai yang paling optimum dalam suatu kumpulan kemungkinan solusi bagi permasalahan. Penentuan nilai optimum sangat bergantung kepada perumusan masalah, metode penyelesaian, penentuan nilai minimum atau maksimum dan toleransi yang diizinkan. Otimasi merupakan suatu proses yang selalu berhubungan dengan *input*, karakteristik metode yang digunakan, proses matematis dan pengujian terhadap *output*, sehingga suatu keluaran optimum yang mewakili sebuah solusi didapatkan guna menyelesaikan sebuah permasalahan (Haupt, 2004).

Optimasi dapat dipandang sebagai proses pencarian nilai maksimal maupun minimal berdasarkan kriteria tertentu dalam sebuah ruang lingkup dengan persyaratan tertentu. Penentuan nilai minimal ataupun maksimal sebagai nilai optimum ditentukan oleh formula dan penilaian terhadap variabel-variabel yang mempengaruhi permasalahan dan variabel-variabel yang merepresentasikan solusinya.

2.3.1 Metode *Heuristik* dan Algoritma Genetika

Metode *heuristik* merupakan suatu cara untuk menemukan sebuah solusi optimasi yang merujuk pada pemikiran tertentu yang dianggap masuk akal untuk diterima. Penyelesaian yang ditemukan dengan metode *heuristik* terkadang bukanlah solusi terbaik, tetapi dapat diterima karena sudah mencapai persentasi kadar kriteria tertentu dari solusi optimum (Turban dan Aronson, 1998).

Algoritma genetika merupakan suatu metode heuristik yang dikembangkan berdasarkan teori evolusi makhluk hidup (Teori Evolusi Darwin), dimana suatu individu terbentuk dan berkembang biak melalu proses reproduksi, dan menghasilkan individu-individu baru dengan karakteristik yang di pengaruhi oleh kromosom dan gen pembentuk individu tersebut. Setiap individu pada makhluk hidup mempunyai tingkat kekuatan/kebugaran (*fitness*) dalam berjuang menghadapi berbagai keadaan seleksi alam. Sebagian individu yang mempunyai

tingkat kekuatan/kebugaran yang memadai akan dapat bertahan hidup, sedangkan individu yang lainnya akan gugur. Prinsip-prinsip genetika dan seleksi alam ini kemudian digunakan untuk membentuk sebuah metode optimasi heuristik yang dikenal dengan algoritma genetika yang pertama kali perkenalkan oleh John Holland dan dikembangkan oleh muridnya David Goldberg. Algoritma genetika memanfaatkan prinsip proses seleksi yang dikenal dengan prinsip evolusi, dalam evolusi individu secara terus menerus mengalami perubahan, dan hanya individu yang kuat yang mampu untuk bertahan hidup (Goldberg dan Holland, 1988; Gen dan Lin, 2007).

Algoritma genetika memiliki beberapa kelebihan (Kramer, 2017), yaitu :

1. Dalam pencarian solusi dari sebuah permasalahan, algoritma ini hanya melakukan sedikit perhitungan matematis. Pencarian dalam algoritma ini tidak terlalu memperhatikan formula relasi antara proses, masalah dan variabel-variabelnya secara langsung. Algoritma ini yang mengendalikan fungsi obyektif dan kendala yang didefinisikan dalam suatu ruang pencarian.
2. Algoritma ini sangat efektif pada ruang pencarian global, karena terdapat operator-operator evolusi dalam penentuan nilai variabel yang mewakili sebuah solusi.
3. Algoritma ini sangat fleksibel dan sangat mudah untuk dikombinasikan (*hybrid*) dengan metode lain untuk meningkatkan efektifitasnya.

Algoritma genetika mempunyai ciri khas yang membedakan dengan algoritma pencarian yang lain (Goldberg, 1989), yaitu :

1. Kawasan representasi solusi dari masalah yang di kodekan dalam bentuk kromosom berasal dari rangkaian nilai yang bukan berasal dari kawasan masalah itu sendiri.
2. Algoritma ini mencari solusi bukan dari sebuah titik awal tertentu, tapi dari individu-individu dalam sebuah populasi yang merepresentasikan solusi. Dengan demikian kemungkinan untuk terjebak dalam nilai optimum lokal akan sangat kecil.

3. Algoritma ini tidak didasarkan pada pengetahuan bantuan (*auxiliary knowledge*) tapi di dasarkan pada nilai fungsi obyektif saja.
4. Algoritma ini bekerja dengan menggunakan aturan probabilistik (*probabilistic rule*), dan tidak menggunakan aturan deterministik (*deterministic rule*).

2.3.2 Pemetaan Dalam Algoritma Genetika

Algoritma genetika diadopsi dari proses evolusi alam dan sifat-sifat genetika. Proses komputasi dalam algoritma ini dapat dianalogikan dengan proses seleksi makhluk hidup dalam sebuah populasi. Dalam algoritma genetika dikenal istilah-istilah yang juga terdapat pada sistem genetika dan evolusi makhluk hidup seperti berikut (Marsland, 2015) :

a. Populasi

   Populasi adalah kumpulan dari individu-individu yang membentuk suatu kelompok besar. Populasi ini adalah merepresentasikan kumpulan kemungkinan solusi yang akan dianalisa.

b. Individu

   Individu dalam algoritma genetika akan merepresentasikan suatu nilai atau keadaan yang menyatakan salah satu solusi dari permasalahan yang akan dicari solusinya, Individu memiliki kromosom, *gen*, dan nilai *fitness*.

c. *Gen*

   Gen adalah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam kesatuan gen yang dinamakan kromosom.

d. Kromosom

   Kromosom adalah gabungan dari gen-gen yang membentuk nilai tertentu

e. Generasi

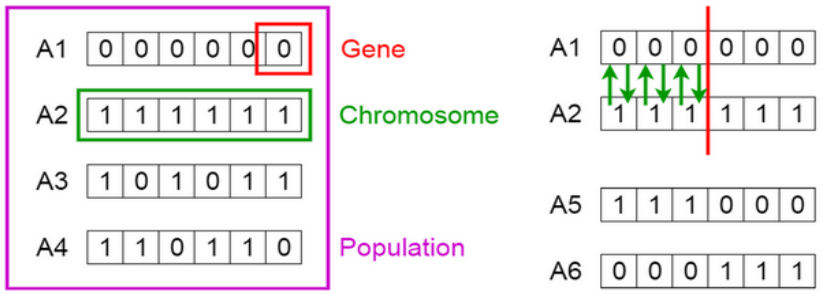
   Generasi menyatakan satuan-satuan siklus proses evolusi

f. *Fitness value*

   Fitness value adalah suatu nilai yang dihitung dari nilai-nilai *gen* dan kromosomnya yang menyatakan kualitas dari suatu individu, nilai fitness

ini merepresentasikan kualitas dari solusi yang ditawarkan dari suatu individu.

Illustrasi gen, kromosom dan populasi dapat dilihat pada gambar 2.8.



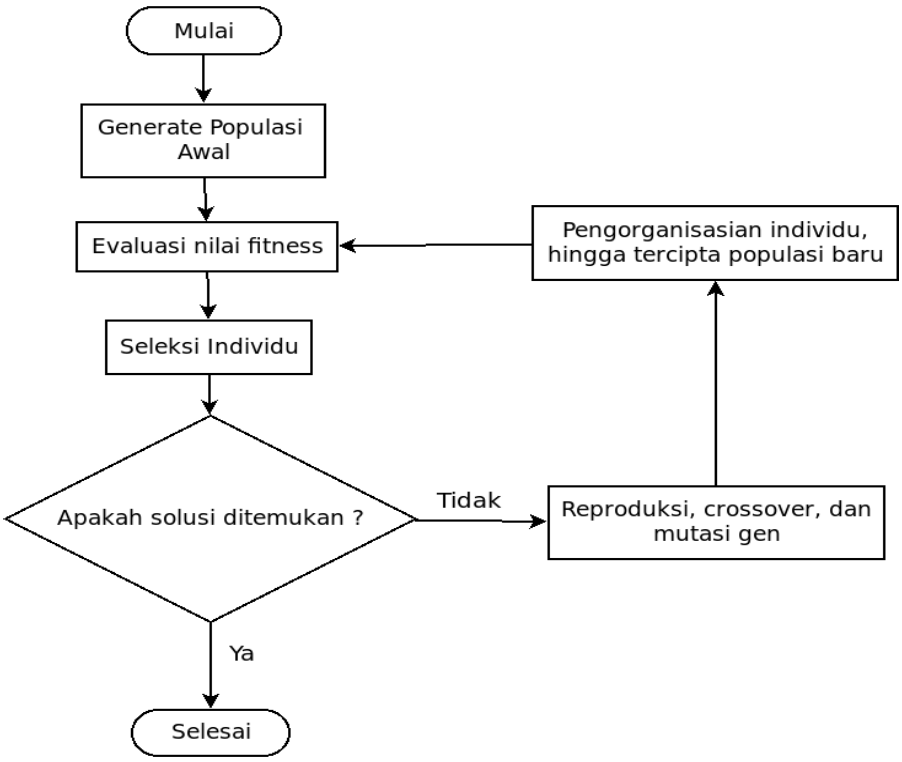
Gambar 2.8 Illustrasi Gen, Kromosom dan Populasi

2.3.3 Siklus Algoritma Genetika

Seperti halnya pada proses alamiah, proses komputasi dimulai dengan membangkitkan individu-individu yang menunjukkan sejumlah kemungkinan penyelesaian. Individu-individu ini membentuk kumpulan individu yang disebut populasi. Penentuan populasi awal dalam algoritma genetika dilakukan secara acak, dari populasi awal ini kemudian akan membentuk populasi-populasi berikutnya yang di bentuk dengan menggunakan operator-operator algoritma genetika. Generasi-generasi berikutnya dibentuk melalui proses yang diadopsi pada proses alamiah. Generasi berikutnya terbentuk dari kromosom-kromosom hasil seleksi yang bertahan dari generasi sebelumnya, sebagian lagi dibentuk dari kromosom-kromosom hasil persilangan (*crossover*), mutasi, dan pertumbuhan yang dihasilkan dari operator tertentu.

Pada setiap generasi, kromosom-kromosom dari setiap individu akan dievaluasi dengan menggunakan fungsi *fitness* untuk mendapatkan *fitness value*-nya. Fungsi yang digunakan untuk menghitung nilai fitness ini ditentukan berdasarkan tingkat optimasi kualitas penyelesaian (solusi) dari permasalahan. Berdasarkan nilai fitness ini diambil beberapa kromosom yang terbaik untuk

kemudian diproses pada generasi berikutnya. Setiap generasi selalu akan dicari individu terbaik dan dievaluasi apakah individu tersebut mempunyai susunan kromosom yang dianggap sebagai solusi dari permasalahan. Apabila solusi yang dicari masih belum ditemukan, maka akan dilakukan proses reproduksi individu baru dengan cara *crossover* dan mutasi genetika. (Du dkk,2016) Kromosom-kromosom yang dibentuk dari generasi sebelumnya ini disebut sebagai kromosom anak(*offspring*), sehingga dibentuk populasi berikutnya, proses ini akan diulang secara terus-menerus hingga suatu solusi ditemukan, untuk jelasnya bisa dilihat pada *flow chart* pada gambar 2.9.



Gambar 2.9 *Flowchart* algoritma genetika

2.3.4 Pembentukan Populasi Awal

Pembentukan populasi awal adalah proses pembentukan individu-individu dalam suatu generasi. Pada tahap ini akan terjadi proses *encoding* yang merepresentasikan gen dari setiap kromosom yang dimiliki oleh individu. Pada populasi awal penggunaan fungsi inisialisasi yang tepat akan meningkatkan

performa secara signifikan (Hassanat dkk, 2018). Hasil *encode* adalah berupa angka atau objek yang merepresentasikan variabel yang menjadi bagian dari solusi dalam bentuk bilangan biner (*bit*), angka desimal, *trees, array, list,* atau berbagai bentuk *object.* Ada beberapa metode *encoding* dalam algoritma genetika, yaitu (Sivanandam dan Deepa, 2008):

1. ***Binary Encoding***

   Kromosom dibentuk dari kumpulan bit (*binary digit*) yang ditulis dengan angka 0 atau 1, masing-masing *bit* merepresentasikan karakteristik dari solusi. Antara masalah yang satu dan yang lain mempunya cara *encode* yang berbeda-beda sesuai dengan karakteristik yang direpresentasikan. Tabel 2.7 adalah contoh dari *binary encoding.*

   Tabel 2.7 *Binary encoding*

Kromosom A	1010101110010011
Kromosom B	1000101101001010

2. ***Octal Encoding***

   Octal encoding adalah metode dengan menggunakan bilangan *octal* (angka 0 sampai 7) dalam melakukan *encode* gen yang masing-masing merepresentasikan karakteristik dari solusi. Tabel 2.8 adalah contoh dari *octal encoding.*

   Tabel 2.8 *Octal encoding*

Kromosom A	123423467
Kromosom B	531354354

3. ***Hexadecimal Encoding***

   Hexadecimal encoding adalah metode dengan menggunakan bilangan heksadesimal (angka 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) dalam melakukan *encode* gen yang masing-masing merepresentasikan karakteristik dari solusi. Tabel 2.9 adalah contoh dari *hexadecimal encoding.*

Tabel 2.9 *Hexadecimal encoding*

Kromosom A	5A6432FBA357
Kromosom B	FFEA35321B4C

4. ***Permutation Encoding (Real Number Coding)***

   Setiap kromosom terdiri dari kumpulan angka yang mempunyai urutan tertentu. Pada metode ini terkadang dilakukan koreksi bisa dilakukan setelah satu rangkaian operasi dalam algoritma genetika selesai. Dalam *permutation encoding* setiap gen dalam kromosom bisa berupa bilangan *integer* atau *real* yang di representasikan secara berurutan. Tabel 2.10 adalah contoh dari *permutation encoding*.

Tabel 2.10 *Hexadecimal encoding*

Kromosom A	13567878
Kromosom B	12356789

5. ***Value Encoding***

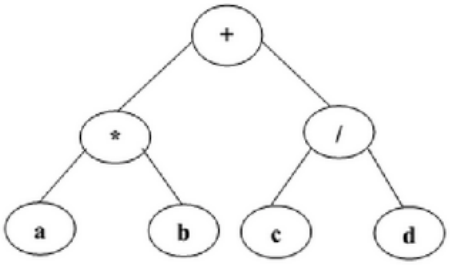
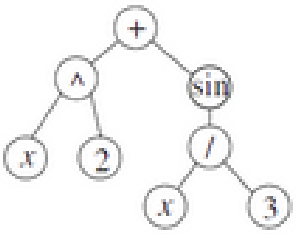
   Setiap gen dalam kromosom berupa nilai-nilai yang berhubungan langsung dengan masalah. *Value encoding* dapat menghasilkan hasil terbaik untuk masalah-masalah tertentu. Bilangan *real,* huruf alfabet, maupun objek-objek yang merepresentasikan keadaan real dapat digunakan dalam metode ini. Contoh dari *value encoding* dapat dilihat pada tabel 2.11.

Tabel 2.11 *Value encoding*

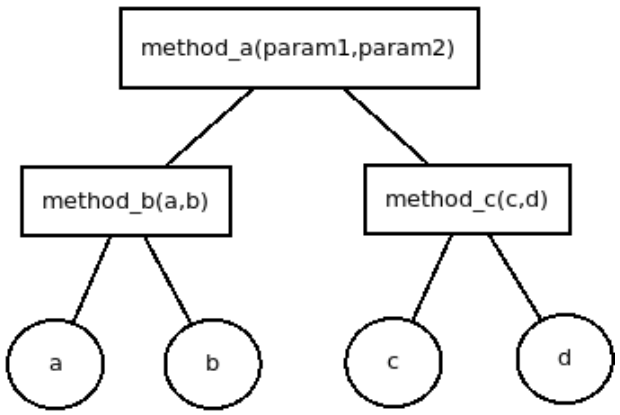
Kromosom A	1.02 0.32 2.11 0.76 4.22
Kromosom B	ABDEJDCDADDG
Kromosom C	(back),(right),(left),(left),(down),(up)

6. ***Tree Encoding***

Pengkodean ini digunakan untuk mengembangkan *expression* tertentu dalam pemrograman genetika. Setiap kromosom berupa pohon dari beberapa objek/*class* seperti fungsi/*method* dan perintah dari bahasa pemrograman. Contoh dari *tree encoding* dapat dilihat pada gambar 2.10 dan 2.11.

Kromosom A	Kromosom B
 ((a*b)+(c/d))	 $x^2+\sin(x/3)$

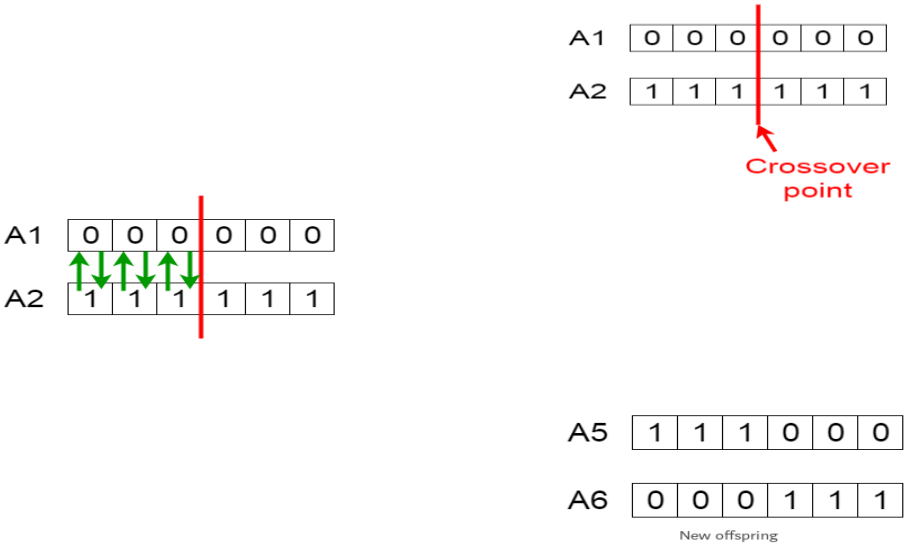
Gambar 2.10 *Tree encoding* sederhana

Kromosom C
 method_a(method_b(a,b),method_c(c,d))

Gambar 2.11 *Tree encoding* yang merepresentasikan *method*

2.3.5 Crossover

Pada proses *crossover* antar individu akan terjadi persilangan genetik antar kromosom yang membentuk individu baru, dengan susunan kromosom baru (Marsland, 2015). Illustrasi *crossover* secara umum dapat dilihat pada gambar 2.12.



Gambar 2.12 Illustrasi *crossover*

Secara umum *crossover* mempunyai beberapa jenis, yaitu *multi-point crossover* (*N-Point crossover*), penyilangan seragam (*uniform crossover*) dan penyilangan optimasi kombinatorial,

2.3.5.1 Multi-point Crossover *(N-Point* Crossover*)*

Metode ini memotong kromosom induk menjadi N+1 bagian, kromosom anak pertama dihasilkan dari warisan potongan urutan ganjil dari induk yang pertama, dan mewariskan potongan dalam urutan genap dari induk kedua. Sedangkan kromosom anak kedua didapat dari kromosom ganjil dari induk kedua tapi pada potongan bagian genapnya mewarisi kromosom induk pertama, untuk lebih jelasnya bisa dilihat pada gambar 2.13.

No	Induk	Anak
1	1 1 0 1 0 0 1 0	1 1 1 0 1 0 1 0
2	1 0 1 0 1 0 1 1	1 0 0 1 0 0 1 1

Gambar 2.13 *N-Point Crossover*

2.3.5.2 Penyilangan Seragam *(Uniform Crossover)*

Pada metode penyilangan seragam (*uniform crossover*), pertama dibangkitkan sebuah kromosom biner secara acak sejumlah gen yang ada pada kromosom yang akan disilangkan. Kromosom biner ini disebut *crossover mask.* Fungsi dari *crossover mask* adalah sebagai pembangkit aturan penurunan gen pada kromosom anak. Apabila posisi gen pada *crossover mask* bernilai 1 maka gen diwariskan dari induk pertama ke anak pertama, sedangkan apabila posisi gen pada *crossover mask* bernilai 0 maka gen akan diwariskan dari induk kedua ke anak pertama. Sedangkan pada anak kedua gen akan diturunkan dari induk kedua apabila pada *crossover mask* bernilai 1, sedangkan apabila *crossover mask* bernilai 0, maka gen akan diturunkan dari induk pertama ke anak kedua. Untuk lebih jelasnya bisa dilihat pada gambar 2.14.

No	*Parent*	Anak
1	1 1 0 1 0 1 1 1	1 0 1 1 1 1 1 1
Mask	1 0 0 1 0 1 1 0	1 0 0 1 0 1 1 0
2	1 0 1 0 1 1 0 1	1 1 0 0 0 1 0 1

Gambar 2.14 Penyilangan Seragam *(Uniform Crossover)*

2.3.5.3 Penyilangan Optimasi Kombinatorial

Penyilangan berbasis kombinatorial terdiri dari penyilangan berbasis posisi (*position based crossover*) dan berbasis urutan (*ordered based crossover*) (Kramer, 2017). Metode penyilangan berbasis posisi (*position based crossover*) bekerja dengan cara memilih sejumlah posisi gen secara acak, Pada posisi gen terpilih pada induk yang pertama diwariskan pada kromosom anak yang kedua, sedangkan gen-gen lainnya dari kromosom anak yang kedua diambil dari gen-gen induk (*parent*) kedua dengan urutan yang sama. Sedangkan untuk kromosom pada anak pertama diambil dari posisi terpilih dari gen pada *parent* kedua, sedangkan gen-gen yang lain diambil dari *parent* pertama pada urutan yang sama. Untuk lebih jelasnya bisa dilihat pada gambar 2.15.

No	*Parent*/induk	Anak
1	1 2 3 4 5 6 7 8	1 8 7 4 5 4 7 2
2	9 8 7 6 5 4 3 2	9 2 3 6 5 6 3 8

Gambar 2.15 *Position Based Crossover*

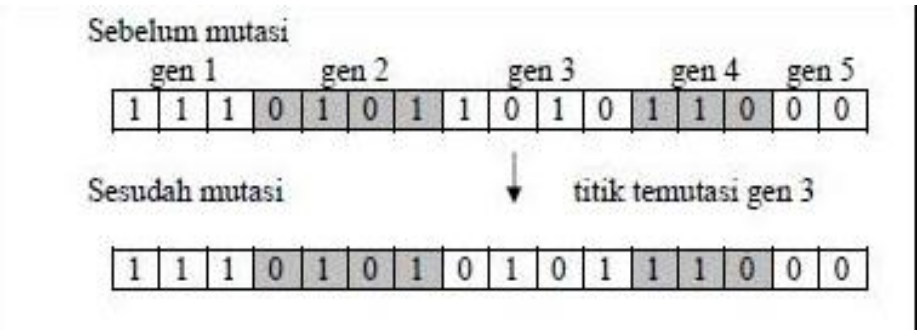
Pada penyilangan berbasis urutan (*Ordered Based Crossover*), kromosom anak pertama dibentuk dari parent pertama, akan tetapi pada gen-gen terpilih dilakukan perubahan dengan cara mengganti urutannya sesuai dengan urutan pada gen-gen yang nilainya sama dari kromosom parent yang kedua (seperti pada gambar 2.16).

No	*Parent*/induk	Anak
1	1 3 2 6 5 4 7 8	1 8 4 6 5 3 7 2
2	9 8 7 6 5 4 3 2	9 2 4 6 5 7 3 8

Gambar 2.16 *Ordered Based Crossover*

2.3.6 Mutasi

Mutasi pada algoritma genetika merupakan proses mengubah nilai gen tertentu pada kromosom secara acak, atau menurut aturan mutasi, misalnya dengan mengganti nilai gen dengan inversi atau nilai lain, atau menggeser susunan rantai gen pada kromosom. Mutasi gen ini dimodelkan sebagaimana yang terjadi pada genetik makluk hidup di alam. Probabilitas mutasi gen biasanya dipilih sangat kecil seperti pada kehidupan alamiah yang memungkinkan terjadinya mutasi genetis tetapi mempunyai persentase yang sangat kecil. Illustrasi mutasi gen secara umum dapat dilihat pada gambar 2.17 (Marsland, 2015).



Gambar 2.17 Mutasi Gen Secara Umum

2.3.7 Jenis-Jenis Mutasi dalam Algoritma Genetika

Mutasi dalam algoritma genetika memiliki beberapa jenis yang populer, diantaranya yaitu :

1. Mutasi yang melibatkan kode biner, yaitu dengan cara mengubah posisi gen yang terkena mutasi menjadi nilai sebaliknya, sebagai contoh bisa dilihat pada gambar 2.18.

Sebelum mutasi	1	1	0	1	0	0	1	1
Setelah mutasi	1	1	0	1	1	0	1	1

Gambar 2.18 Mutasi Gen yang Melibatkan Kode Biner

2. Mutasi Optimasi Kombinatorial, merupakan mutasi optimasi dengan cara kombinasi, metode dalam mutasi optimasi kombinatorial diantaranya adalah mutasi berbasis posisi (*position based mutation*), mutasi berbasis

urutan (*order based mutation*) dan mutasi secara acak (*scramble mutation*). Metode mutasi berbasis posisi bekerja dengan cara memilih posisi gen secara acak kemudian dipindahkan ke posisi acak lainnya (seperti pada gambar 2.19).

Sebelum mutasi	1	2	3	4	5	6	7	8
Setelah mutasi	1	2	3	4	6	7	5	8

Gambar 2.19 Mutasi Gen Berbasis Posisi

Metode mutasi gen berbasis urutan bekerja dengan cara menukar 2 posisi gen yang terpilih secara acak (seperti pada gambar 2.20)

Sebelum mutasi	1	2	3	4	5	6	7	8
Setelah mutasi	1	2	3	4	7	6	5	8

Gambar 2.20 Mutasi Gen Berbasis Urutan

Metode mutasi berbasis acak (*scramble mutation*) dilakukan dengan memilih posisi beberapa gen secara acak, kemudian dilakukan penukaran urutan gen-gen tersebut secara acak juga (seperti pada gambar 2.21).

Sebelum mutasi	1	2	3	4	5	6	7	8
Setelah mutasi	1	2	3	4	6	7	5	8

Gambar 2.21 Mutasi Gen Secara Acak *(Scramble Mutation)*

2.3.8 *Fitness value*

Fitness value adalah suatu nilai pada individu yang digunakan sebagai evaluasi yang merepresentasikan kualitas dari suatu solusi. (Mokshin dan Sharnin, 2019). *Fitness value* dihasilkan melalui sebuah fungsi objektif yang menghasilkan

suatu nilai tertentu yang dapat dijadikan sebagai acuan untuk menentukan kualitas solusi dan seberapa dekat solusi tersebut terhadap titik yang paling optimal.

Fitness value dihitung dengan cara melakukan *decode* terhadap kromosom yang akan dievaluasi. Nilai hasil *decode* ini kemudian diolah oleh fungsi objektif hingga menghasilkan suatu nilai kuantitatif. Semakin kompleks dan semakin banyak kriteria optimasi dari suatu permasalahan, maka fungsi objektif *fitness value* akan menjadi semakin kompleks. Pada permasalahan dengan multi kriteria dari optimasi, diperlukan kombinasi dari beberapa fungsi objektif yang saling terintegrasi dan konsisten untuk menentukan *fitness value*. Perhitungan *fitness value* pada penelitian ini mengacu pada metode evaluasi kepadatan, kapasitas jalan, kecepatan rata-rata dan aturan sistem penjadwalan *traffic light*.

2.4 Seleksi dalam Algoritma Genetika

Dalam algoritma genetika selalu ada proses seleksi disetiap generasi untuk menentukan individu-individu yang akan bertahan pada populasi generasi berikutnya. Kromosom yang terpilih akan dipasangkan dengan kromosom lain melalui proses *crossover* untuk menghasilkan kromosom anak. Proses seleksi ini menggunakan beberapa model yaitu seleksi sebanding dengan *fitness value* (*Fitness Proportionate Selection*), Seleksi Turnamen, Seleksi Peringkat (*Rank Selection*) dan Seleksi Elitis.

2.4.1 Seleksi sebanding dengan *Fitness Value*

Pada seleksi yang sebanding dengan *fitness value* kemungkinan suatu kromosom akan bertahan hidup adalah sebanding dengan nilai yang dihasilkan oleh fungsi obyektif yang digunakan untuk menghitung *fitness value.* Sebagai contoh, jika ada kromosom sejumlah N, maka kemungkinan dari sebuah kromosom i(p_i) untuk bertahan hidup adalah *fitness value* dari kromosom i(f_i) dibagi dengan rata-rata dari *fitness value* seluruh anggota populasi seperti pada persamaan 2.8.

$$p_i = \frac{f_i}{\sum_{x=1}^{N} f_x} \quad (2.8)$$

Dimana p_i adalah kemungkinan/probabilitas dari kromosom ke i untuk bertahan di siklus generasi berikutnya, f_i adalah *fitness value* dari kromosom i, N adalah jumlah kromosom dalam suatu populasi. Implementasi seleksi sebanding dengan *fitness value* biasanya dimodelkan dengan sistem roda rolet (Du dkk, 2016). Roda rolet dibentuk oleh busur-busur sejumlah N (jumlah kromosom dalam populasi). Perbandingan besar sudut untuk masing-masing busur N adalah sebanding dengan perbandingan *fitness value* dari masing-masing kromosom. Seleksi sebanding dengan *fitness value* ini hanya dapat diterapkan pada pencarian nilai maksimal tanpa nilai negatif yang dihasilkan dari fungsi obyektif,

Proses seleksi dengan model roda rolet ini dilakukan dengan dua pendekatan berdasarkan jenis bilangan acak yang digunakan sebagai pemilih, yaitu dengan menggunakan bilangan bulat dan bilangan riil **(**Du dkk, 2016**).**

2.4.2 Pemilihan Kromosom Secara Acak Menggunakan Bilangan Bulat

Pada setiap kromosom dihitung nilai vektornya (*v*) dengan cara menghitung perbandingan antara *fitness value* dari kromosom tersebut terhadap total jumlah *fitness value.* Vektor *v* beranggotakan *M* elemen yaitu $v_1, v_2, v_3 \ldots, v_M$. Setiap v_j adalah bilangan bulat bernilai antara 1 sampai *N* (jumlah kromosom). Kromosom yang terpilih (v_r) diambil berdasarkan bilangan bulat *r* acak antara 1 sampai *M*. Sebagai contoh terdapat 8 kromosom seperti tabel 2.12.

Tabel 2.12 Kromosom dan fitness value dengan metode bilangan bulat

v_i	*Fitness value (f)*	Probabilitas	Perbandingan Fitness value	Banyak v_i
1	5	0.05	1/20	1
2	20	0.2	4/20	4
3	10	0.1	2/20	2
4	25	0.25	5/20	5
5	10	0.1	2/20	2
6	5	0.05	1/20	1
7	5	0.05	1/20	1
8	20	0.2	4/20	4

v_i	1	2	2	2	2	3	3	4	4	4	4	4	5	5	6	7	8	8	8	8
r	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Contoh pada tabel 2.12, bilangan *r* dibangkitkan secara acak antara 1 sampai 20 (*M*), angka yang muncul pada r menunjukkan kromosom terpilih (v_i).

2.4.3 Pemilihan Kromosom Secara Acak Menggunakan Bilangan Riil

Pada pemilihan secara acak menggunakan bilangan riil, mula-mula dibangkitkan bilangan *r* (bertipe bilangan riil) antara 0 sampai dengan jumlah fitness value dari semua kromosom pada suatu generasi. Kromosom terpilih adalah kromosom yang memenuhi kriteria seperti pada tabel 2.13.

Tabel 2.13 Ketentuan Kromosom terpilih dengan Metode Bilangan riil

Nomor Kromosom	Ketentuan
Kromosom ke 1 (v_1)	$r < f_1$
Kromosom ke *i* (v_i)	$\sum_{j=1}^{i-1} f_j \leq r < \sum_{j=1}^{i} f_j$ dan $i>1$

Sebagai contoh terdapat 8 kromosom seperti ditunjukkan pada tabel 2.14.

Tabel 2.14 Kromosom dan *Fitness Value* dengan Metode Bilangan Riil

v_i	f_i	$\sum_{j=1}^{i-1} f_j$	$\sum_{j=1}^{i} f_j$	$\frac{\sum_{j=1}^{i-1} f_j}{\sum_{j=1}^{N} f_j}$	$\frac{\sum_{j=1}^{i} f_j}{\sum_{j=1}^{N} f_j}$
1	5	0	5	0,00	0,05
2	20	5	25	0,05	0,25
3	10	25	35	0,25	0,35
4	25	35	60	0,35	0,60
5	10	60	70	0,60	0,70
6	5	70	75	0,70	0,75
7	5	75	80	0,75	0,80
8	20	80	100	0,80	1,00

Metode pemilihan kromosomnya adalah dengan membangkitkan bilangan acak *r* sejumlah N dengan ketentuan $0 \leq r < 1$.

2.4.4 Seleksi Peringkat

Metode seleksi peringkat bekerja dengan cara mengurutkan *fitness value* dari kromosom-kromosom dalam suatu populasi. *Fitness value* dengan nilai terbaik akan menempati urutan pertama, diikuti peringkat berikutnya secara berurutan. Probabilitas terpilihnya kromosom (p_i) ditentukan dengan fungsi distribusi, sedemikian rupa sehingga jumlah probabilitas semua kromosom sama dengan satu, secara formula dapat dituliskan seperti pada persamaan 2.9.

$$\sum_{i=1}^{N} p_i = 1 \tag{2.9}$$

Dimana p_i adalah probabilitas dari kromosom ke *i.* Fungsi distribusi yang digunakan dapat secara *linear* atau secara eksponensial. Persamaan 2.10 menunjukkan bentuk umum dari fungsi *linear* dan fungsi 2.11 menunjukkan bentuk umum probabilitas secara eksponensial.

$$p_i = a.i + b \tag{2.10}$$

$$p_i = a.e^{(b.i+c)} \tag{2.11}$$

dimana p_i adalah nilai probabilitas dari kromosom yang ke i, sedangkan a,b,c dan e adalah sebuah bilangan pembentuk fungsi (Du dkk, 2016).

2.4.5 Seleksi Turnamen

Seleksi turnamen dalam algoritma genetika menggunakan analogi sebuah turnamen, dimana akan dicari juara dalam sebuah kompetisi yang diikuti oleh peserta turnamen. Kromosom-kromosom dibagi menjadi beberapa grup secara acak. Setiap grup mempunyai anggota lebih dari satu kromosom. Pada setiap grup akan dicari kromosom dengan nilai fitness terbaik untuk dipertahankan pada populasi berikutnya.

2.5 Penentuan *Parameter* dalam Algoritma Genetika

Dalam algoritma genetika tidak ada aturan yang pasti dalam menentukan parameter, baik dalam hal ukuran populasi, probabilitas penyilangan, probabilitas mutasi maupun parameter lainnya. Dalam setiap langkahnya algoritma genetika tidak terlepas dari penggunaan bilangan acak, mulai dari pembentukan populasi awal, penyilangan (*crossover*) atau proses mutasi. Akan tetapi parameter algoritma pada umumnya menggunakan komposisi probabilitas penyilangan antara 60%-70%, probabilitas mutasi yang cukup kecil dan ukuran populasi berkisar antara 50-500 kromosom (Jong dan Schonfeld, 2001).

2.6 Optimasi Dengan Kendala pada Algoritma Genetika

Dalam menyelesaikan suatu permasalahan optimasi seringkali terdapat sebuah kendala yang harus ditangani. Tidak semua kromosom yang terbentuk selama proses algoritma genetika berada dalam kawasan penyelesaian permasalahan yang memiliki persyaratan tertentu (mempunyai kendala). Secara matematis kendala dirumuskan dalam sebuah persamaan atau pertidaksamaan tertentu. Kromosom-kromosom yang terbentuk harus dimanipulasi agar selalu berada dalam kawasan penyelesaian masalah yang memenuhi persamaan atau

pertidaksamaan (formula yang representasikan kendala). Kromosom yang tidak memenuhi kendala disebut kromosom tidak layak (*infeasible*).

Secara matematis, optimasi dengan kendala dapat dituliskan dengan rumus optimasi seperti pada persamaan 2.12, dengan kendala dituliskan seperti pada pertidaksamaan 2.13 dan persamaan 2.14 (Yen, 2006):

$$\text{optimasi } f(x) \tag{2.12}$$

$$g_i(x) \leq 0 \text{ , untuk } 1 \leq i \leq q \tag{2.13}$$

$$h_j(x) = 0 \text{ , untuk } q+1 \leq j \leq m \tag{2.14}$$

Penyelesaian bagi fungsi objektif $f(x)$ harus memenuhi kendala, yaitu kendala pertidaksamaan $g_i(x) \leq 0$ dan kendala persamaan $h_j(x) = 0$. Kendala pertidaksamaan dapat juga ditulis dalam bentuk fungsi lain. Kromosom-kromosom yang kemungkinan terbentuk pada proses pembentukan populasi, hasil *crossover,* maupun hasil mutasi yang tidak memenuhi kriteria kendala harus dilakukan penanganan khusus.

Secara garis besar ada tiga kemungkinan hasil pendekodean kromosom, yaitu berada di luar kawasan penyelesaian masalah (sama sekali tidak merepresentasikan penyelesaian masalah), kromosom merupakan representasi penyelesaian masalah hanya saja tidak memenuhi kriteria kendala dalam optimasi yang diselesaikan dan kromosom berada dalam kawasan penyelesaian masalah dan memenuhi syarat kendala dalam masalah yang dioptimasi. Strategi dalam menangani kendala dapat dikelompokkan menjadi 4 (Kramer, 2017) yaitu :

1. Strategi penolakan (*rejecting strategy*)

   Dalam strategi penolakan, seluruh kromosom hasil proses evolusi yang tidak layak akan dibuang. Strategi ini cukup populer, akan tetapi mempunyai keterbatasan, apabila semua kromosom pada generasi pertama atau hasil inisialisasi merupakan kromosom yang tidak layak, maka artinya semua kromosom akan tertolak.
2. Strategi perbaikan (*repairing strategy*)

Penggunaan strategi perbaikan bekerja dengan cara kromosom yang tidak layak akan diperbaiki kualitasnya hingga menjadi layak. Dengan perbaikan ini, kromosom yang tidak layak tersebut masih memungkinkan untuk tetap melanjutkan kelangsungan hidupnya.

3. Strategi penyesuaian operator genetika

   Pada penggunaan strategi ini, operator-operator dalam proses yang diterapkan harus mampu menghasilkan kromosom yang memenuhi kendala, sehingga kromosom-kromosom yang dihasilkan selalu memenuhi persamaan dan/atau pertidaksamaan kendala dalam penyelesaian masalah.

4. Strategi pemberian pinalti (*penalizing strategy*)

   Pada strategi ini masih memungkinkan pencarian berada pada kawasan yang tidak layak. Pada permasalahan yang dengan kendala yang kompleks, kawasan tidak layak merupakan bagian terbesar dari suatu populasi. Apabila proses pencarian hanya dilakukan pada kawasan yang layak saja maka akan menjadi sangat terbatas. Kromosom yang tidak layak ini tetap diterima dan dijadikan pijakan pada iterasi berikutnya.

2.7 Fungsi Evaluasi untuk Penanganan Kendala dengan Strategi Pinalti

Strategi pinalti adalah adopsi dari strategi konvensional. Langkah utama adalah menentukan nilai pinalti yang tepat agar proses pencarian dapat menuju ke titik konvergensi tanpa menyebabkan konvergensi dini. Fungsi pinalti harus ditambahkan kedalam fungsi objektif. Misalnya untuk masalah optimasi nilai maksimum, fungsi pinalti akan mengurangi besarnya nilai fungsi objektif. Makin besar tingkat penyimpangan terhadap kendala, maka pengurangan terhadap nilai fungsi objektifnya harus semakin besar. Dalam algoritma genetika, penambahan fungsi pinalti berada pada tahap evaluasi terhadap populasi. Adanya penambahan fungsi pinalti ini membuat keberadaan sejumlah kromosom yang tidak layak pada setiap generasi dapat dipertahankan untuk membantu proses pencarian menuju ke titik optimum.

Untuk menambahkan fungsi pinalti pada pendefinisian fungsi evaluasi, dilakukan dengan cara membuat fungsi pinalti sebagai suku penambah dari fungsi evaluasi, atau fungsi pinalti sebagai faktor pengali dari fungsi evaluasi. Jika suatu kromosom *v* mempunyai fungsi *f(v)* yang menyatakan evaluasi dari kromosom *v* dan terdapat fungsi *p(v)* sebagai pinalti dari kromosom *v,* maka implementasi fungsi pinalti dapat dilakukan dengan cara :

1. **Fungsi evaluasi dengan fungsi pinalti sebagai suku penambah**
   Pada metode ini fungsi evaluasi dibentuk dengan menambahkan hasil dari fungsi objektif *f(v)* dengan fungsi pinalti *p(v)* seperti pada persamaan 2.15.

   $$eval(v) = f(v) + p(v) \tag{2.15}$$

   Dimana *v* adalah kromosom, *f(v)* adalah fungsi objektif dan *p(v)* adalah fungsi pinalti. Pada pencarian optimasi dengan nilai maksimum, fungsi *p(v)* berfungsi sebagai pengurang dari fungsi objektif, dan nilai mutlak maksimal dari fungsi *p(v)* harus lebih kecil dari nilai mutlak minimal dari fungsi *f(v),* secara matematis dapat dituliskan seperti pada persamaan 2.16.

   $$(| p(v) | max) \leq (| f(v) | min) \tag{2.16}$$

   Fungsi $p(v) = 0$ menandakan kromosom layak, dan $p(v) < 0$ menandakan kromosom tidak layak. Sedangkan untuk masalah optimasi dengan pencarian nilai minimum, maka akan berlaku sebaliknya, fungsi pinalti berfungsi sebagai penambah dari fungsi objektif, dengan $p(v) = 0$ untuk kromosom layak, dan $p(v) > 0$ untuk kromosom tidak layak.

2. **Fungsi evaluasi dengan fungsi pinalti sebagai faktor pengali**
   Fungsi evaluasi terhadap kromosom *v* merupakan hasil perkalian dari fungsi objektif *f(v)* dan fungsi pinalti *p(v)* seperti ditunjukkan pada formula 2.17.

   $$eval(v) = f(v) \cdot p(v) \tag{2.17}$$

Optimasi dengan pencarian nilai maksimum, fungsi pinalti *p(v)* harus mengurangi nilai dari fungsi objektif *f(x),* sehingga *p(v) = 1* menandakan kromosom tersebut layak dan tidak terkena pinalti, sedangkan nilai fungsi pinalti harus memenuhi kriteria $0 \leq p(v) \leq 1$.

Sedangkan pada optimasi dengan pencarian nilai minimum, berlaku sebaliknya. Fungsi pinalti harus memperbesar nilai fungsi objektif, sehingga *p(v) = 1* menandakan kromosom tersebut layak dan tidak terkena pinalti, sedangkan nilai pinalti harus memenuhi kriteria $p(v) > 1$.

2.8 Syarat Berhenti Dalam Algoritma Genetika

Proses pencarian solusi yang paling optimal pada algoritma genetika akan berhenti apabila suatu syarat tertentu telah terpenuhi (Ding dan Gasvoda, 2004). Pada umumnya syarat berhenti didasarkan pada batas nilai *fitness* (*fitnes value*), batas nilai fungsi objektif, batas waktu komputasi, banyak generasi dan terjadinya kovergensi. Jenis permasalahan, tingkat kerumitan, perangkat keras dan tingkat urgensi dari sebuah keputusan merupakan faktor-faktor yang menjadi pertimbangan untuk memilih syarat berhenti dari sebuah proses pencarian. Pemilihan syarat berhenti haruslah tepat dan dapat bersinergi dengan sistem lain yang terintegrasi dengan algoritma genetika, sehingga tidak terjadi keterlambatan dan dapat memberikan *output* yang memenuhi kriteria standar yang diharapkan.

2.9 Kepadatan, Kapasitas Jalan dan Kecepatan Rata-Rata Kendaraan

Kepadatan lalu-lintas mempunyai keterkaitan dengan kecepatan rata-rata kendaraan yang melintas pada jalan tersebut (Quek dan Chew, 2014) dan faktor kapasitas jalan (Samra, 2018). Penghitungan kecepatan kendaraan yang melintas dapat dihitung dengan rumus 2.18.

$$v = \frac{d}{t} \tag{2.18}$$

v adalah kecepatan kendaraan, d adalah jarak, t adalah waktu tempuh. Semakin lambat rata-rata kecepatan kendaraan yang melintas pada suatu ruas jalan, maka akan mengindikasikan bahwa ruas tersebut mempunyai tingkat kemacetan yang semakin tinggi (Gaddam dan Rao, 2018)

Banyak faktor yang mempengaruhi kapasitas jalan, dan diperlukan analisa yang mendalam untuk memutuskan kapasitas dan kelayakan suatu jalan (Samra, 2018), Besarnya kapasitas jalan ini berpengaruh terhadap tingkat kecepatan normal suatu kendaraan dapat melintas pada jalan tersebut, nilai kelayakan kecepatan kendaraan pada suatu ruas jalan biasanya ditentukan oleh lembaga berwenang (Mishra dkk., 2018).

2.10 Rasio Kecepatan Kendaraan dan Efektifitas *Traffic Light (Traffic Light fitness value)*

Sistem Penjadwalan yang bertujuan untuk melakukan pembagian resouce yang baik, akan menjadwalkan sedemikian rupa sehingga resource akan terbagi secara seimbang dan merata. Performa dari sebuah sistem *load-balancing* dapat dinilai dari bagaimana sebuah beban dapat terbagi secara tepat dan merata (Xu dan Tian, 2017). Demikian pula *traffic light* adalah bertujuan untuk mengatur lalu-lintas pada masing-masing ruas jalan, sehingga ruas jalan dapat dimanfaatkan secara optimal. Perbedaan rata-rata kecepatan kendaraan dari masing-masing ruas jalan yang diatur dalam satu siklus penjadwalan *traffic light* mengindikasikan performa dari sistem tersebut.

Seperti sistem load-balancing pada umumnya, tingkat kesuksesan dari sebuah sistem *load-balancing* dapat dilihat dari seberapa jauh beban dapat terbagi secara merata (Xu dan Tian, 2017). Sistem *load-balancing* ini juga banyak diterapkan diberbagai bidang dan sudah dipergunakan untuk menyelesaikan permasalahan-permasalahan yang kompleks dan dinamis (Fardbastani and Sharifi, 2019). Demikian juga dalam penelitian ini mempunyai analogi yang serupa, sebuah *traffic light* yang baik adalah *traffic light* yang mampu mengatur dan membagi

jadwal penggunaan jalannya secara merata dan proporsional untuk masing-masing ruas jalan yang diaturnya, sehingga kecepatan ideal dari masing-masing ruas jalan adalah sama dengan kecepatan rata-rata dari kendaraan yang melintas disemua ruas jalan, sehingga dapat diformulakan dengan rumus 2.19.

$$v_t = \overline{v} = \frac{\sum_{x=1}^{n} v_x}{n} \tag{2.19}$$

Dimana v_t adalah target ideal kecepatan kendaraan, v adalah rata-rata kecepatan, v_x is kecepatan dari jalan x, n adalah jumlah kendaraan.

Tingkat keberhasilan suatu *traffic light* dalam mengatur lalu-lintas dapat dilihat dari nilai deviasi selisih dari kecepatan sebuah ruas jalan dengan kecepatan ideal (kecepatan rata-rata), sehingga hal ini menunjukkan tingkat efektifitas pengaturan lalu-lintas terhadap sejumlah (n) ruas jalan. Secara rumus dapat diformulakan dengan rumus 2.20.

$$e_x = |v_x - t_x| = \sqrt{(v_x - v_{tx})^2} \tag{2.20}$$

Dimana e_x adalah *traffic light effectiveness*, v_{tx} adalah target ideal kecepatan kendaraan dari jalan x, v_x adalah kecepatan dari jalan x.
Kemudian, untuk melihat efektivitas pengaturan lalu-lintas dari satu siklus sejumlah (n) ruas jalan dapat direpresentasikan dengan jarak *eucledian* (*eucledian distance*) antara koordinat yang merepresentasikan kecepatan masing-masing ruas jalan dengan kecepatan ideal dari masing-masing ruas jalan, secara formula dapat dituliskan seperti pada persamaan 2.21.

$$E = \sqrt{\sum_{x=1}^{n} (t_x - v_{tx})^2} \tag{2.21}$$

Dimana E adalah efektivitas keseluruhan dari n ruas jalan, tx adalah target kecepatan ideal dari ruas jalan x dan v_{tx} adalah kecepatan kendaraan pada ruas jalan x.