**Optimasi *Waiting Time* pada Simulasi *Intelligent Traffic Light* *Control* menggunakan *Markov Decision Process***

***Waiting Time Optimization* *in Intelligent Traffic Light* *Control Simulation using* *Markov Decision Process***

**Tugas Akhir**

**Diajukan untuk memenuhi sebagian dari syarat**

**untuk memperoleh gelar Sarjana**

**dari Program Studi Sarjana Teknik Informatika**

**Fakultas Informatika**

**Universitas Telkom**

**Beryl Ramadhian Aribowo**

**1103120118**



**Program Studi Sarjana Teknik Informatika**

**Fakultas Informatika**

**Universitas Telkom**

**Bandung**

**2016**

**Lembar Pernyataan**

Dengan ini saya menyatakan bahwa Tugas Akhir dengan judul “Optimasi *Waiting Time* pada Simulasi *Intelligent Traffic Light* *Control* menggunakan *Markov Decision Process*” beserta seluruh isinya adalah benar-benar karya saya sendiri dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan. Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila kemudian ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya ini, atau ada klaim dari pihak lain terhadap keaslian karya saya ini.

Bandung, 2 Mei 2016

Yang membuat pernyataan,

(Beryl Ramadhian Aribowo)

**Lembar Pengesahan**

**Optimasi *Waiting Time* pada Simulasi *Intelligent Traffic Light* *Control* menggunakan *Markov Decision Process***

***Waiting Time Optimization* *in Intelligent Traffic Light* *Control Simulation using* *Markov Decision Process***

**Beryl Ramadhian Aribowo**

**1103120118**

Tugas Akhir ini telah diterima dan disahkan untuk memenuhi sebagian dari syarat

untuk memperoleh gelar pada Program Studi Sarjana Teknik Informatika

Fakultas Informatika

Universitas Telkom

Bandung, 2 Mei 2016

Menyetujui

|  |  |
| --- | --- |
| Pembimbing I  Tjokorda Agung Budi Wirayuda S.T., M.T.  NIP: 06830333-1 | Pembimbing II  Said Al Faraby S.T., M.Sc.  NIP. 11890804-3 |
| Ketua Program Studi  Sarjana Teknik Informatika  M. Arif Bijaksana, Ir., M.Tech., Ph.D.  NIP. 03650312-4 | |

**Abstrak**

*Traffic Light Control* adalah sebuah topik yang mendapatkan banyak perhatian karena sifatnya yang sangat penting disebabkan berkaitan dengan optimasi lalu lintas, selain itu disebabkan juga masalah kepadatan lalu lintas adalah masalah umum yang sering terjadi. Beberapa algoritma baik yang sederhana maupun *advanced* sudah banyak diterapkan oleh peneliti lainnya sebelumnya. Penelitian ini menghasilkan sebuah skema untuk *Traffic Light Control* dengan menggunakan metode *model based Reinforcement Learning* yaitu *Markov Decision Process* (MDP). Model MDP dibentuk melalui observasi *environment* atau dengan kata lain infrastruktur lalu lintas yang dibentuk melalui sebuah *Traffic Light Control Simulator*, yaitu *Green Light District*. Pada penelitian ini dihasilkan sebuah model MDP yang dapat mengoptimasi *waiting time* pada infrastruktur lalu lintas, berdasarkan hasil pengujian dengan beberapa skenario dan parameter tertentu, secara keseluruhan model MDP yang diimplementasi pada penelitian ini dapat mengungguli algoritma dasar *Traffic Light Control* lainnya.

**Kata kunci**: *Traffic Light Control*, *Reinforcement Learning*, *Markov Decision Process*, *Waiting Time*, *Green Light District Simulator*, Optimasi.

**Abstract**

*Traffic Light Control is a topic which recieved many attention from the rersearchers due to its important nature, because it is the most determining factor to traffic optimization, another reason is because traffic over-capacity or traffic jam is common problem and one of the most infamous problem in everyday routine. Some basic and advanced algorithms have been devised by researchers to tackle this problem since years ago. Specifically, this research would also do the same, by producing a Traffic Light Control scheme by using a model based Reinforcement Learning which is Markov Decision Process (MDP). This particular MDP model is obtained through observation to the environment which is an infrastructure that comes from a traffic light simulator, Green Light District. This research will produce a MDP model that would be able to optimize the waiting time of the tested infrastructure, based on the test results, this particular MDP model outperformed the other basic Traffic Light Control algorithms.*

**Keywords**: *Traffic Light Control*, *Reinforcement Learning*, *Markov Decision Process*, *Waiting Time*, *Green Light District Simulator*, *Optimization*.

**Lembar Persembahan**

*Alhamdulillah*, puji dan syukur dipanjatkan ke hadirat Allah SWT atas segala rahmat, karunia, dan hidayah-Nya sehingga Tugas akhir ini dapat diselesaikan, tidak lupa juga ucapan terima kasih untuk segenap keluarga dan rekan-rekan serta semua pihak yang telah secara langsung maupun tidak langsung memberikan *support* dalam pembuatan tugas akhir ini. Oleh karena itu, rasa terima kasih sebesar-besarnya disampaikan kepada :

1. Bapak Bambang Aribowo dan Ibu Neneng Hafidah selaku kedua orang tua penulis yang selalu mendukung, memberi doa, serta memberi arahan dari sejak lahir tanpa ada hentinya.
2. Bapak Tjokorda Agung Budi Wirayuda selaku pembimbing I dan Bapak Said Al Faraby selaku pembimbing II yang telah memberikan arahan, ilmu baik konsep maupun praktik, pencerahan, dan membimbing penulis dalam pembuatan tugas akhir ini.
3. Bapak Fazmah Arif Yulianto dan Bapak Mahmud Dwi Sulistyo selaku dosen wali IF-36-06 yang telah membimbing serta mendidik penulis dari semester awal perkuliahan.
4. Semua Bapak dan Ibu Dosen di Universitas Telkom yang telah memberikan ilmu nya serta mendidik penulis, ilmu nya akan bermanfaat terus hingga kelak.
5. Kawan-kawan Informasix (IF-36-06) yang bersama-sama berjuang dari semester awal, yang telah bersama-sama melalui berbagai macam suka dan duka perkuliahan.
6. Pihak-pihak lainnya yang telah membantu penulis dalam menyelesaikan tugas akhir ini yang tidak dapat disebutkan satu persatu.

**Kata Pengantar**

Puji dan syukur dipanjatkan ke hadirat Allah SWT, atas izin-Nya Tugas Akhir dengan judul ” *Optimasi Waiting Time* pada Simulasi *Intelligent Traffic Light Control* menggunakan *Markov Decision Process*” dapat diselesaikan dalam rangka memenuhi syarat kelulusan studi pada program sarjana di Fakultas Informatika Universitas Telkom.

Tugas akhir ini masih memiliki kekurangan, tidaklah sempurna, maka dari itu kritik serta saran yang membangun akan sangat diterima. Sangat diharapkan penelitian dalam tugas akhir ini dapat dikembangkan lagi kedepannya sehingga dapat memberikan hasil yang lebih baik. Semoga tugas akhir ini mampu memberi manfaat dan hikmah yang besar bagi pembacanya.

Bandung, 2 Mei 2016

Beryl Ramadhian Aribowo

Daftar Isi

[Abstrak i](#_Toc449975689)

[Abstract ii](#_Toc449975690)

[Lembar Persembahan iii](#_Toc449975691)

[Kata Pengantar iv](#_Toc449975692)

[Daftar Isi v](#_Toc449975693)

[Daftar Gambar vii](#_Toc449975694)

[Daftar Tabel viii](#_Toc449975695)

[Daftar Istilah ix](#_Toc449975696)

[1. Pendahuluan 1](#_Toc449975697)

[**1.1** **Latar belakang** 1](#_Toc449975698)

[**1.2** **Perumusan masalah** 2](#_Toc449975699)

[**1.3** **Tujuan** 2](#_Toc449975700)

[**1.4** **Batasan Masalah** 2](#_Toc449975701)

[**1.5** **Metodologi penyelesaian masalah** 2](#_Toc449975702)

[**1.6** **Sistematika Penulisan** 3](#_Toc449975703)

[2. Kajian Pustaka 4](#_Toc449975704)

[**2.1** ***Traffic Light*** 4](#_Toc449975705)

[**2.2** ***Intelligent Traffic Light Control*** 5](#_Toc449975706)

[**2.3** **Pemodelan dan Simulasi** 5](#_Toc449975707)

[2.3.1 Pemodelan 5](#_Toc449975712)

[2.3.2 Simulasi 5](#_Toc449975713)

[2.3.3 Discrete Event Simulation 7](#_Toc449975714)

[**2.4** ***Green Light District Simulator*** 8](#_Toc449975715)

[**2.5** ***Reinforcement Learning*** 8](#_Toc449975716)

[2.5.1 Model Based Reinforcement Learning 9](#_Toc449975719)

[2.5.2 Markov Decision Process 9](#_Toc449975720)

[2.5.3 Dynamic Programming 10](#_Toc449975721)

[2.5.4 Value Iteration 11](#_Toc449975722)

[3. Metodologi dan Desain Sistem 12](#_Toc449975723)

[**3.1** **Gambaran Umum Sistem** 12](#_Toc449975724)

[**3.2** **Deskripsi Kebutuhan Sistem** 12](#_Toc449975725)

[**3.2.1** **Kebutuhan Fungsional** 12](#_Toc449975726)

[**3.2.2** **Spesifikasi Perangkat Keras** 13](#_Toc449975727)

[**3.2.3** **Spesifikasi Perangkat Lunak** 13](#_Toc449975728)

[**3.3** **Desain Sistem** 13](#_Toc449975729)

[3.3.1 Infrastruktur/environment 13](#_Toc449975734)

[3.3.2 Desain MDP 14](#_Toc449975735)

[3.3.2.1 *State* 14](#_Toc449975736)

[3.3.2.2 *Action* 15](#_Toc449975737)

[3.3.2.3 *Transition Matrix* 15](#_Toc449975738)

[3.3.2.4 *Immediate Reward* 16](#_Toc449975739)

[3.3.2.5 *Value Iteration* 16](#_Toc449975740)

[**3.4** **Skenario Pengujian** 17](#_Toc449975741)

[3.4.1 Klasifikasi Kepadatan Jalur 18](#_Toc449975743)

[3.4.2 Algoritma basic TLC 18](#_Toc449975744)

[3.4.3 Optimasi parameter MDP 18](#_Toc449975745)

[3.4.4 Perbandingan MDP dengan basic TLC 19](#_Toc449975746)

[4. Pengujian dan Analisis 20](#_Toc449975747)

[**4.1** **Pengujian Sistem** 20](#_Toc449975748)

[**4.2** **Tujuan Pengujian** 20](#_Toc449975749)

[**4.3** **Skenario Pengujian** 20](#_Toc449975750)

[**4.4** **Analisis Hasil Pengujian** 20](#_Toc449975751)

[4.4.1 Penentuan Kepadatan Lalu Lintas 20](#_Toc449975757)

[4.4.2 Pengujian Algoritma basic TLC 21](#_Toc449975758)

[4.4.3 Pencarian Parameter Optimal MDP 22](#_Toc449975759)

[4.4.4 Perbandingan best MDP dengan basic TLC 24](#_Toc449975760)

[5. Kesimpulan dan Saran 28](#_Toc449975761)

[**5.1** **Kesimpulan** 28](#_Toc449975762)

[**5.2** **Saran** 28](#_Toc449975763)

[Daftar Pustaka 29](#_Toc449975764)

[Lampiran 30](#_Toc449975765)

**Daftar Gambar**

[**Gambar 2‑1. Kemungkinan state pada lampu lalu lintas setiap satu time step (http://www.driving-lessons southampton.co.uk/mediac/400\_0/media/Traffic\_lights\_4\_states.jpg).** 4](#_Toc449975796)

[**Gambar 2‑2 : Contoh state dan kondisi pengubah state pada satu lampu lalu lintas (http://www2.hawaii.edu/~esb/2003fall.ics331/project5states.png).** 5](#_Toc449975797)

[**Gambar 2‑3. Skema studi simulasi [6].** 6](#_Toc449975798)

[**Gambar 2‑4. Contoh sekuens pembangkitan random variable (seed) dengan LCG(a=5, c=1, m=16, dan X0=1).** 7](#_Toc449975799)

[**Gambar 2‑5. Contoh representasi MDP dengan state Si, aksi Ai, dan nilai reward Ri[http://bluehawk.monmouth.edu/rclayton/web-pages/f11-520/mdpf01.png].** 10](#_Toc449975800)

[**Gambar 3‑1. Blok Diagram garis besar sistem.** 12](#_Toc449975801)

[**Gambar 3‑2. Infrastruktur yang digunakan untuk penelitian.** 14](#_Toc449975802)

[**Gambar 3‑3. ilustrasi pembagian segment pada satu jalur.** 15](#_Toc449975803)

[**Gambar 4‑1. AJWT basic TLC pada spawn rate (0,135).** 21](#_Toc449975804)

[**Gambar 4‑2. AJWT basic TLC dengan spawn rate (0,25).** 22](#_Toc449975805)

[**Gambar 4‑3. AJWT basic TLC dengan spawn rate (0,4).** 22](#_Toc449975806)

[**Gambar 4‑4. AJWT MDP pada spawn rate (0,135).** 23](#_Toc449975807)

[**Gambar 4‑5. AJWT MDP pada spawn rate (0,25).** 24](#_Toc449975808)

[**Gambar 4‑6. AJWT MDP pada spawn rate (0,4).** 24](#_Toc449975809)

[**Gambar 4‑7. AJWT MDP versus basic TLC dengan spawn rate (0,135).** 25](#_Toc449975810)

[**Gambar 4‑8. AJWT MDP vs basic TLC dengan spawn rate (0,25).** 26](#_Toc449975811)

[**Gambar 4‑9. AJWT MDP versus basic TLC dengan spawn rate (0,4).** 26](#_Toc449975812)

[**Gambar 4‑10. AJWT MDP vs basic TLC - all scenario average.** 27](#_Toc449975813)

**Daftar Tabel**

[**Tabel 3‑1. Potongan Tabel Transisi dari initial state LLLL pada model 3 segment** 15](#_Toc449975814)

[**Tabel 3‑2. Potongan Final Policy hasil Value Iteration pada salah satu junction dengan model 3 segment** 17](#_Toc449975815)

[**Tabel 3‑3. Parameter Spawn Rate dan Seed untuk semua skenario pengujian** 18](#_Toc449975816)

[**Tabel 3‑4. MDP dengan masing-masing parameter pembeda yaitu Segment dan Gamma** 19](#_Toc449975817)

[**Tabel 4‑1. Klasifikasi Kepadatan Lalu Lintas** 21](#_Toc449975818)

**Daftar Istilah**

|  |  |
| --- | --- |
| *Action* | Perintah yang dapat diterima *agent* atau pekerjaan yang dapat dilakukan *agent* yang menyebabkan memungkinkannya terjadinya perpindahan *state*. |
| *Delta* | Parameter *threshold* yang digunakan untuk menghentikan *value iteration*. |
| *Final Policy* | Kumpulan *state* dan best *action* yang ditentukan dari *value iteration*. |
| *Gamma* | Sebuah parameter yang memotong nilai *reward* untuk *state* di masa depan. |
| *Reward* | Nilai yang didapatkan *agent* ketika memasuki suatu *state s*. |
| *State* | Keadaan dalam sistem MDP/posisi *agent* yang merupakan inti MDP yang ditentukan oleh parameter tertentu dari *environment* nya, dalam konteks TLC *state* disini adalah keadaan kepadatan *junction*. |
| *State Utility* | Nilai *reward* suatu *state* yang sudah digabungkan dengan *reward* *state* tetangganya yang dipotong dengan *gamma*. |
| *Spawn Rate* | Probabilitas akan munculnya sebuah kendaraan per satu *time step* pada sebuah *edge node*. |
| *Time Step* | Ukuran waktu pada GLD *simulator*. |
|  |  |
|  |  |
|  |  |

1. **Pendahuluan**
   1. **Latar belakang**

Kepadatan lalu lintas merupakan masalah yang memiliki pengaruh cukup besar terhadap perputaran perekonomian terutama pada kota-kota besar dimana padatnya lalu lintas sudah menjadi masalah sehari-hari, yang disebabkan faktor-faktor penyebab terjadinya peningkatan kepadatan jalan, diantaranya jumlah penduduk setempat yang terus bertambah, sehingga secara langsung bertambah volume kendaraan pada satu area, adanya penyempitan ruas jalan serta faktor infrastruktur lainnya, atau tidak baiknya konfigurasi lampu lalu lintas pada suatu persimpangan, sehingga menyebabkan bertambahnya antrian kendaraan pada persimpangan tersebut.

Faktor yang diteliti pada tugas akhir ini adalah faktor konfigurasi lampu lintas pada persimpangan, dengan mencari konfigurasi lampu lalu lintas yang cocok dapat mengurangi kepadatan lalu lintas pada persimpangan.

Beberapa jenis solusi sebelumnya sudah ditawarkan untuk meminimalkan kepadatan lalu lintas melalui optimasi konfigurasi lampu lalu lintas, diantaranya dengan menggunakan algoritma pengaturan lalu lintas yang sederhana, misalnya, algoritma pengaturan lalu lintas *Longest Que* yang memiliki sifat selalu memberikan prioritas untuk jalur dengan kepadatan lalu lintas paling tinggi atau bahkan pengaturan lalu lintas *Random* yang memberikan lampu hijau dengan urutan yang acak sepanjang waktu.

Para pakar juga sudah melakukan penelitian dengan algoritma yang lebih *advanced* dalam mengatur lampu lalu lintas, yaitu dengan memanfaatkan sistem yang menggunakan *Artificial Intelligence* (AI), diantaranya dengan menggunakan metode *Fuzzy System* untuk TLC [1], *Genetic Algorithm* [2], *Multi Agent System* [3], dan beberapa macam *Reinforcement Learning* [4, 5].

Penelitian kali ini akan mengoptimalkan konfigurasi lampu lalu lintas dengan menggunakan metode *model based Reinforcement Learning* yaitu *Markov Decision Process* (MDP).

* 1. **Perumusan masalah**

Rumusan masalah yang dibentuk dalam penelitian ini adalah sebagai berikut :

1. Bagaimana memodelkan keadaan lalu lintas menjadi bentuk matematis pada *Markov Decision Process* (MDP)?
2. Bagaimana mengaplikasikan MDP pada *Intelligent Traffic Light Control Simulator*?
3. Bagaimana mencari parameter yang dapat mengoptimalkan performansi MDP dengan ukuran *waiting time* pada *Simulator*?
   1. **Tujuan**

Berdasarkan perumusan masalah, tujuan dari penelitian ini adalah sebagai berikut :

1. Memodelkan keadaan lalu lintas dengan MDP.
2. Mengaplikasikan MDP pada *Intelligent Traffic Light Control Simulator.*
3. Mencari parameter untuk MDP sehingga mendapatkan *waiting time* optimal.
   1. **Batasan Masalah**

Tugas akhir ini memiliki ruang lingkup dalam penelitiannya sehingga tidak terlalu luas, berikut adalah batasan-batasan dalam tugas akhir ini :

1. Data kendaraan yang melewati persimpangan diambil dari satu infrastruktur persimpangan yang tetap dan hanya satu jenis, yaitu dua buah persimpangan dengan masing-masing persimpangan memiliki empat jalur masuk dan jalur keluar.
2. Kendaraan yang diamati bertipe mobil.
3. Parameter MDP yang dicari untuk optimasi *waiting time* adalah jumlah *state* dan *discount factor*/*gamma*.
   1. **Metodologi penyelesaian masalah**

Tahap-tahap penelitian pada Tugas Akhir ini adalah sebagai berikut :

1. Studi Literatur.

Mencari sumber literatur untuk menjadi acuan dan pembanding, yang bertema *Intelligent Traffic Light Control* serta teori-teori dasar mengenai *Reinforcement Learning* dan MDP dalam bentuk *paper*, jurnal, atau buku.

1. Perancangan Infrastruktur lalu lintas dan Model MDP.

Merancang infrastruktur persimpangan yang akan digunakan untuk penelitian dan membentuk model MDP dari keadaan lalu lintas pada simulator, merancang bagaimana ruang lingkup *state*, *action*, serta *reward* pada MDP, menentukan parameter-parameter yang akan digunakan untuk implementasi MDP.

1. Implementasi MDP pada *Intelligent Traffic Light Control Simulator.*

Menerapkan model MDP yang sudah dirancang sebelumnya menjadi bentuk *code* program pada *Intelligent Traffic Light Control Simulator*, dari tahap pengambilan data, *training*/*learning*, hingga testing dan analisis statistik.

1. Pengumpulan Data dan *training* model MDP.

Mengumpulkan data kepadatan tiap jalur dari persimpangan jalan kemudian dilakukan *training* pada model MDP yanag telah diimplementasi menggunakan data yang telah diambil.

1. Pengujian dan Analisis

Melakukan *testing* pada model MDP berdasarkan hasil *training* sebelumnya, kemudian melakukan analisis terhadap performansinya berdasarkan parameter *waiting time* pada simulator dengan membandingkan *waiting time* dari model MDP terhadap metode-metode lainnya pada *simulator*.

1. Penyusunan Laporan Tugas Akhir

Menyusun laporan tugas akhir yang berisi dokumentasi serta hasil analisis penelitian secara detail.

* 1. **Sistematika Penulisan**

Penulisan Tugas Akhir ini dibagi menjadi beberapa bab berdasarkan poin-poin utama, yaitu :

1. Bab 1 : Pendahuluan

Bab Pendahuluan berisi landasan-landasan penelitian yang dilakukan, yaitu latar belakang, rumusan masalah, tujuan dan batasan masalah, serta metodologi penyelesaian masalah.

1. Bab 2 : Kajian Pustaka

Bab Kajian Pustaka berisi kumpulan teori yang digunakan dalam penelitian ini, yaitu teori mengenai *Intelligent Traffic Light Control Simulator, Reinforcement Learning, Markov Decision Process* (MDP).

1. Bab 3 : Perancangan Sistem

Bab Perancangan Sistem berisi langkah-langkah perancangan model, pengambilan data, dan implementasi MDP pada *Intelligent Traffic Light Control*.

1. Bab 4 : Pengujian dan Analisis

Bab Pengujian dan Analisis berisi hasil-hasil pengujian performansi dari MDP pada *Intelligent Traffic Light Control* serta beberapa metode lainnya sebagai pembanding, dan analisis kedudukan performansi MDP terhadap metode lainnya.

1. Bab 5 : Kesimpulan dan Saran.

Bab Kesimpulan dan Saran berisi kesimpulan hasil pengujian dan analisis, serta saran untuk pengembangan penelitian selanjutnya.

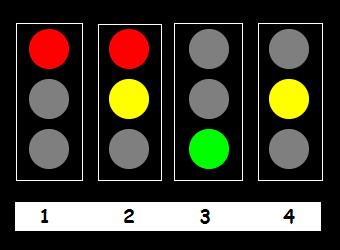
# Kajian Pustaka

* 1. ***Traffic Light***

*Traffic Light* atau lampu lalu lintas adalah alat untuk mengatur pergerakkan baik kendaraan maupun manusia (yang menyebrang di sebuah jalan) di persimpangan jalan, lampu lalu lintas terdiri dari tiga buah lampu yang berbeda warna yang memberi tanda pengemudi kendaraan tindakan apa yang harus diambil ketika lampu dengan warna tertentu menyala, tiga warna lampu tersebut sesuai dengan standar internasional adalah merah, kuning, dan hijau. Tindakan/aksi yang perlu diambil oleh pengemudi kendaraan ketika salah satu warna lampu aktif pada umumnya adalah sebagai berikut :

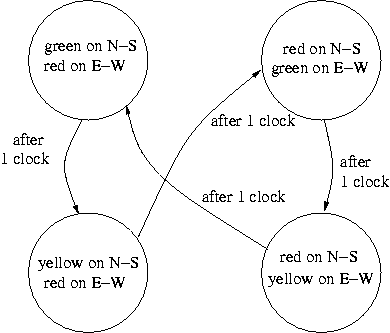
* Warna lampu hijau menyala artinya kendaraan yang bersesuaian dengan lampu lalu lintas tersebut diperbolehkan untuk bergerak melalui persimpangan jalan.
* Warna lampu merah menyala artinya kendaraan harus berhenti.
* Warna lampu kuning menyala pada umumnya memiliki arti bahwa kendaraan boleh melintas, tetapi diharapkan untuk *aware* khususnya di Indonesia artinya boleh melintas tetapi harus bersiap untuk berhenti karena sekuens setelah lampu warna kuning adalah lampu warna merah.

Dalam menentukan urutan warna lampu yang menyala dan warna apa saja yang menyala dalam satu waktu sebuah lampu lalu lintas memiliki *state* warna lampu, dimana pada tiap negara mengikuti standar *state* yang berbeda.



**Gambar 2‑1. Kemungkinan state pada lampu lalu lintas setiap satu time step (http://www.driving-lessons southampton.co.uk/mediac/400\_0/media/Traffic\_lights\_4\_states.jpg).**

Setiap lampu lalu lintas memiliki pengaturan pergantian *state* yang berbeda berdasarkan lama waktu sebuah *state*, pada umumnya lampu lalu lintas menggunakan pengaturan *fixed* *time*, artinya pada sebuah jalan satu *state* di satu lampu lalu lintas memiliki lama waktu yang statis. Beberapa negara mulai mengembangkan *intelligent traffic light control* dimana pengaturan lama waktu sebuah *state* tidak statis, tapi berdasarkan parameter input tertentu sehingga dapat menghasilkan lalu lintas yang lebih efisien dari segi waktu.



**Gambar 2‑2 : Contoh state dan kondisi pengubah state pada satu lampu lalu lintas (http://www2.hawaii.edu/~esb/2003fall.ics331/project5states.png).**

* 1. ***Intelligent Traffic Light Control***

*Intelligent Traffic Light Control* adalah *Traffic Light Control* (TLC) yang berbasiskan *Artificial Intelligence* (AI), dimana konfigurasi lampu pada setiap persimpangan jalan sudah dapat berpindah *state* secara automatis berbasiskan persamaan yang lebih kompleks ketimbang *traffic light control* yang sederhana seperti konfigurasi statis.

Diantaranya beberapa *Intelligent Traffic Light Control* yang sudah dikembangkan dengan metode-metode AI yang berbeda, diantaranya, TLC dengan menggunakan *Fuzzy System* dengan dua variabel *input* yaitu *arrival* dan *queue,* dan variabel output *extenstion time* (berapa lama waktu lampu hijau diperpanjang) [1]; *Genetic Algorithm* dengan *performance index* yang berdasarkan *weight* dari tiap jalur pada persimpangan untuk fungsi *fitness* [2]; *Multi Agent System* dengan kumpulan *knowledge rule* dan melakukan observasi terhadap *traffic pattern* untuk menentukan konfigurasi lampu setiap jalur pada persimpangan jalan [3]; dan *reinforcement learning* yang bersifat *model free* dengan menggunakan *Q-learning* untuk mendapatkan *optimal policy* [5].

* 1. **Pemodelan dan Simulasi**

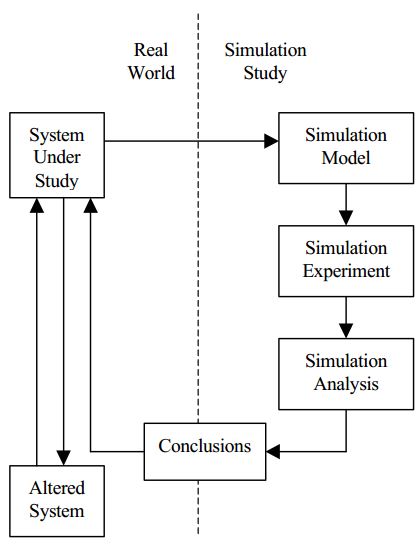


### Pemodelan

Pemodelan adalah proses membangun sebuah model, model adalah sebuah representasi dari sistem, model dapat menunjukan tujuan utama sistem dan bagaimana sistem bekerja. Salah satu tujuan utama penggunaan model adalah untuk melakukan analisa dan dapat memprediksi efek perubahan terhadap sistem [6]. Sebuah model harus dapat merepresentasikan sistem namun tidak boleh juga terlalu kompleks, maka sebuah model harus sederhana namun komunikatif dalam artian dapat menyampaikan maknanya.

### Simulasi

Simulasi adalah sebuah teknik untuk merepresentasikan dan melakukan eksperimen dari sebuah pemodelan sistem, simulasi juga dapat dikatakan sebagai teknik pemecahan masalah dengan cara observasi terhadap performansi sebuah sistem [7]. Simulasi dilakukan karena apabila melakukan eksperimen terhadap sistem dengan cara mengimplementasikan model sistem secara langsung memakan terlalu banyak *resource* atau tidak *cost effective* [6]. Tujuan utama dari dilakukannya simulasi terhadap sebuah sistem adalah untuk mengetahui performansi sistem, sejauh mana sistem telah memenuhi harapan/tujuan, dan juga untuk mengurangi kemungkinan terjadinya kesalahan ketika mengimplementasikan sistem yang sesungguhnya dengan cara memperbaiki kesalahan menggunakan acuan performansi sistem yang diuji melalui simulasi.



**Gambar 2‑3. Skema studi simulasi [6].**

Pada kasus tertentu, simulasi harus dilakukan sebelum tahap implementasi sistem, karena faktor-faktor berikut :

* Terlalu memakan banyak *resource* atau tidak mungkin untuk implementasi secara langsung.
* Sistem dapat dimodelkan namun memiliki skala yang terlalu besar apabila langsung diaplikasikan.
* Sulit memperoleh data untuk aplikasi nyata.

Simulasi terhadap pemodelan apabila dilakukan dapat memberikan keuntungan sebagain berikut [6] :

* Mendapatkan pemahaman yang lebih baik mengenai sistem yang akan dibangun melalui observasi terhadap simulasi.
* Dapat menguji hipotesa sistem.
* Menghemat waktu untuk mengobservasi sebuah proses yang kompleks.
* Mempelajari pengaruh faktor-faktor eksternal maupun internal terhadap sistem.
* Melakukan percobaan dengan situasi yang tidak diketahui oleh sistem
* Mengetahui variabel yang paling berpengaruh terhadap sistem.
* Mengetahui batasan/*limit* kemampuan sebuah sistem.
* Dapat mengembangkan sistem yang baik dengan waktu yang lebih singkat.

### *Discrete Event Simulation*

*Discrete Event Simulation* (DES) adalah simulasi yang mengalami perubahan atau pergantian keadaan/*state* setiap ada *event* yang terjadi pada sistem, dimana waktu terjadinya setiap event adalah diskret (terdapat titik pemisah yang jelas antar waktu).

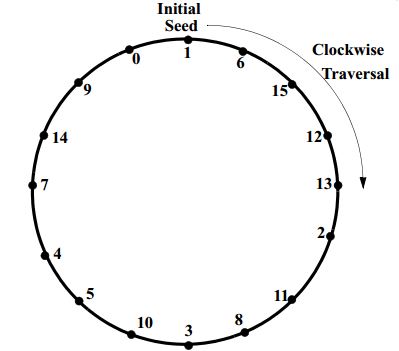
DES memiliki sifat stokastik dan dinamis, stokastik artinya simulasi berjalan berdasarkan parameter yang memiliki peluang atau probabilitas, misalnya input simulasi adalah variabel random yang dihasilkan dengan probabilitas tertentu, sedangkan dinamis artinya simulasi terus berubah mengikuti waktu. Karena sifatnya yang stokastik, DES bergantung pada parameter masukkan yang dimunculkan dari *random variable* atau variabel acak, salah satu metode yang dapat digunakan untuk membangkitkan *random variable* adalah *Linear Congruential Generators* (LCG).

LCG merupakan metode untuk membangkitkan *random variable* yang bersifat *pseudo-random*, artinya angka yang dimunculkan tidak benar-benar acak, namun memiliki pola tertentu, LCG dapat direpresentasikan dengan persamaan sebagai berikut[8]:

Xn+1 = aXn + c (mod m) (2.1)

Dimana *Xn* adalah bilangan random *X* yang dibangkitkan pada urutan ke-*n*, *a*, *c*, dan *m* adalah konstanta yang ditentukan secara manual.

Pada LCG nilai yang pertama kali dibangkitkan secara manual adalah *seed* atau X0 yang akan menentukan nilai yang dibangkitkan selanjutnya.



**Gambar 2‑4. Contoh sekuens pembangkitan random variable (seed) dengan LCG(a=5, c=1, m=16, dan X0=1).**

* 1. ***Green Light District Simulator***

*Green Light District* (GLD) *Simulator* adalah *simulator* untuk *Intelligent Traffic Light Control*, GLD bersifat *open source*, GLD banyak digunakan untuk penelitian pengembangan algoritma *Traffic Light Control*.

GLD memiliki beberapa *package* yang masing-masing memiliki fungsi yang berbeda, *package* utama yang menjadi fokus untuk digunakan dalam penelitian serta pengembangan algoritma adalah package *gld.algo*, walaupun pada beberapa kondisi tertentu dalam implementasinya perlu melakukan modifikasi pada *package* lainnya.

Pada GLD semua aksi/fungsi serta algoritma yang digunakan dalam *simulator* dilakukan berdasarkan *time step*, yaitu satuan waktu yang digunakan dalam GLD, *time step* berupa *thread* dengan *delay* yang dapat diatur, secara *default* memiliki *delay* selama 1000 *milisecond*, *spawn* kendaraan, *switch* warna lampu, pergerakkan kendaraan, semua dilakukan berdasarkan *time step*, transisi/perpindahan *time step* disebut *cycle*, setiap *cycle* secara *default* jaraknya adalah 1 *time step*.

GLD menggunakan sebuah variabel bernama *gain* untuk menentukan konfigurasi lampu pada setiap persimpangan jalan/*node*, lampu yang memiliki nilai *gain* yang paling tinggi diantara lampu lainnya dalam satu *node* akan diberikan rambu jalan/warna lampu *hijau*, sedangkan sisanya akan diberi warna lampu merah/berhenti. Semua algoritma yang diciptakan pada package *gld.algo* memiliki nilai akhir/*output* berupa *gain*, sehingga dapat memanipulasi konfigurasi lampu lalu lintas pada setiap *node*.

GLD memiliki beberapa algoritma *basic traffic light control* yang sudah diimplementasikan sebelumnya yang dapat digunakan untuk perbandingan performansi, beberapa algoritma *basic* pada GLD diantaranya :

* Random TLC : Memberikan nilai *gain* tertinggi pada salah satu lampu secara *random*.
* Longest Que TLC : Memberikan nilai *gain* tertinggi pada lampu yang memiliki kepadatan jalur tertinggi.
* Relative Longest Que TLC : Memberikan nilai *gain* pada setiap lampu sesuai dengan kepadatan jalur yang dimiliki.
* Dan lain lain.
  1. ***Reinforcement Learning***

*Reinforcement Learning* (RL) pada intinya adalah bagaimana cara memetakan keadaan menjadi aksi, pembelajar atau yang disebut dengan *agent* tidak diberi arahan untuk memilih aksi yang cocok untuk kasus tertentu, tapi dibiarkan berinteraksi dengan lingkungan sistem dan memilih aksi terbaik berdasarkan *value* atau *feedback* yang didapatkan dengan cara mencoba aksi-aksi yang tersedia. Dengan *trial and error*, RL dapat mempelajari aturan-aturan yang terdapat dalam sebuah sistem [4].

RL berbeda dengan *supervised learning* (salah satu metode *learning*), *supervised learning* mempelajari suatu masalah melalui contoh dengan bantuan atau arahan dari *supervisor*, dalam hal ini adalah manusia, karena sifatnya tersebut *supervised* *learning* tidak dapat mempelajari bagaimana interaksi dalam sistem [4].

RL memiliki empat elemen penting lainnya selain *agent* dan lingkungan sistem, yaitu *policy*/aturan, fungsi *reward* dan *value*, serta model dari lingkungan sistem [4]. *Policy* adalah pemetaan dari *state* lingkungan menjadi *action* yang tersedia pada *state* tersebut. Fungsi *reward* menentukan tujuan dari RL, memetakan *action* yang diambil dari *state* menjadi sebuah nilai *reward*, sedangkan fungsi *value* adalah fungsi yang memetakan nilai *reward* untuk jangka waktu yang lebih lama, mempertimbangkan masa yang akan datang, tujuan dari *agent* dalam RL sendiri adalah untuk memaksimalkan nilai *reward* untuk keseluruhan, artinya mencari *value* terbaik.



### *Model Based Reinforcement Learning*

Pada awalnya, sebuah kasus *Reinforcement* *Learning* perlu untuk mempelajari keadaan *environment* nya terlebih dahulu, dalam hal ini yang perlu didapatkan dari *environment* agar bisa dibentuk sebuah model adalah *transition probability* dan *immediate reward*, dimana *transition probability* adalah kemungkinan berpindahnya *state* *i* ke *state* *j* setelah melakukan suatu *action a*, atau dapat dinotasikan dengan *Pij*(*a*), sedangkan *immediate reward* adalah *reward* yang diberikan ketika memasuki suatu *state* *i* jika berpindah dari suatu *state j* [4].

Untuk mendapatkan model dari *transition probability*, perlu dilakukan observasi pada *environment*, yaitu mengumpulkan data berapa kali transisi dari *state i* ke *state j* ketika melakukan *action* *a* untuk semua *state*, dan berapa kali *state i* melakukan *action a*. Perhitungan mencari *transition probability* dapat dilihat pada persamaan (2.2) :

(2.2)

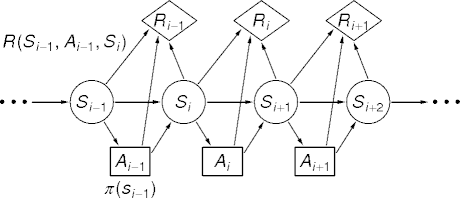
Dimana adalah jumlah transisi *state i* ke *state j* jika melakukan *action a*, dan adalah jumlah *state i* melakukan *action a*.

Setelah mendapatkan model transisi/*transition probability* dari observasi pada *environment* dan melakukan pemberian nilai *immediate reward* pada setiap *state* yang dapat dijangkau, maka model MDP sudah dapat untuk dibawa pada tahap selanjutnya, yaitu *training* dengan *Dynamic Programming*.

### *Markov Decision Process*

*Markov Decision Process* (MDP) adalah sebuah model interaksi antara *agent* dan lingkungannya yang merupakan sebuah *task* dari *reinforcement learning* yang memenuhi bentuk *Markov Property*, secara khususnya, MDP yang memiliki jumlah *state* yang terbatas adalah *finite MDP* [9]. *Markov Property* adalah kumpulan *state* yang berhasil/memuat cukup banyak informasi yang relevan bagi *agent*.

*Finite* MDP terdiri dari kumpulan *state* yang terbatas yaitu *S=s1,s2,….si* dan aksi *A=a1,a2…..ai* yang dapat dilakukan oleh *agent*, probabilitas *P(s,a,s’)* untuk berpindah ke suatu *state s’* dari *state s* berdasarkan aksi *a*, kemudian sebuah nilai *reward* *R(s,a,s’)* yang akan diperoleh *agent* ketika mengambil aksi *a* di *state* *s* yang bertransisi menjadi state *s’* [10].



**Gambar 2‑5. Contoh representasi MDP dengan state Si, aksi Ai, dan nilai reward Ri[http://bluehawk.monmouth.edu/rclayton/web-pages/f11-520/mdpf01.png].**

*Agent* akan menentukan sebuah *policy* *π* : *S → A*, dimana pada waktu *t* mengambil aksi *at* pada state *st* kemudian akan menghitung kumulatif nilai *reward* atau dengan kata lain mendapatkan nilai dari fungsi *value* dengan persamaan sebagai berikut[10]:

(2.3)

Dimana Rt adalah nilai *reward* pada waktu *t* dan γadalah *discount factor* yang akan menurunkan nilai *reward* pada *t* selanjutnya.

### *Dynamic Programming*

*Dynamic Programming* adalah kumpulan algoritma yang digunakan untuk mencari penyelesaian dan mencari nilai optimal *policy*/peraturan dari *perfect model* seperti MDP [9]. Tugas dari *agent* adalah untuk memilih aksi yang mengoptimalkan nilai kumulatif dari *reward*. Misalkan nilai *Vπ(s)* adalah prediksi dari nilai kumulatif *reward* apabila *agent* sedang berada pada *state* *s* dan *policy π* akan digunakan [4]:

(2.4)

Operator *E* digunakan untuk mencari rata-rata dari semua kemungkinan perpindahan *state*.

Kemudian terdapat sebuah fungsi *Qπ(i,a)* yaitu *Q-function* yang berfungsi untuk mengevaluasi aksi yang sudah diambil yang menghasilkan nilai ekspektsi pada tahap selanjutnya dari nilai *reward* yang sudah dihitung dengan *discount factor* apabila mengambil aksi *a* pada state *i* dan mengeksekusi *policy* π [4]:

(2.5)

Dimana dicari nilai maksimalnya dengan persamaan :

(2.)

(2.)

Kemudian untuk semua *state i*, dicari nilai *policy* maksimalnya dengan iterasi, kemudian akan didapatkan *Bellman optimality equation* dimana *Q\** dan *V\** adalah nilai optimal [4]:

(2.)

### *Value Iteration*

*Value Iteration* adalah salah satu algoritma dari *dynamic programming* yang digunakan untuk mencari *policy* optimal dari MDP. *Value iteration* melakukan iterasi terhadap *Bellman Equation* hingga mencapai titik konvergen [9].

Berikut adalah langkah-langkah yang dilakukan dalam *value* *iteration* untuk mencapai *policy* yang optimal/konvergen [4]:

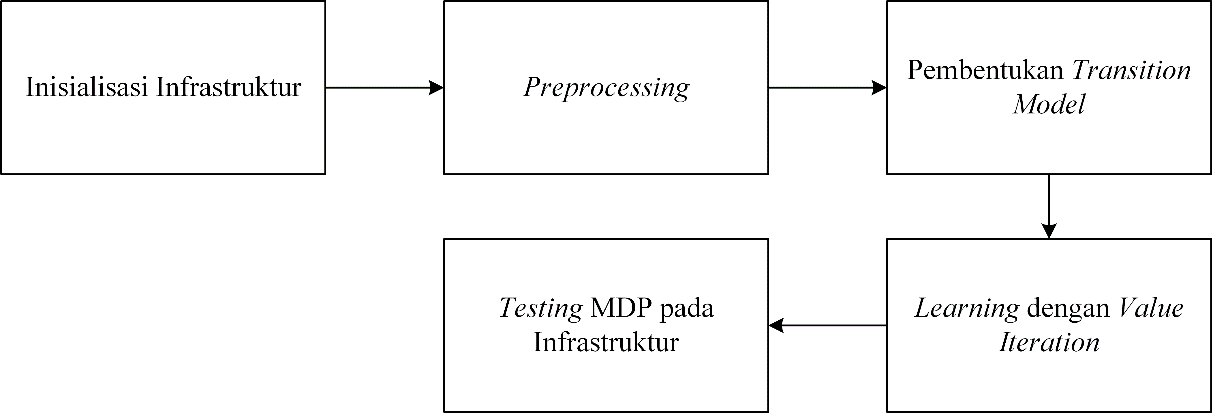
1. Inisialisasi *state utility* (*Q* dan *V*) terlebih dahulu, misalnya di-*set­* 0.
2. Ulangi *step* 3-5 apabila kriteria berhenti iterasi belum terpenuhi, misalnya kriteria berhenti iterasi yang digunakan adalah *delta*, dimana *delta* adalah nilai positif yang sangat kecil (mendekati 0), jika *delta>max*(|*V(i)-V(i-1)|*), maka kriteria berhenti terpenuhi.
3. *Update* nilai *Q* dengan fungsi :

1. Hitung nilai fungsi *V* yang baru :
2. Gunakan *policy* untuk setiap state berdasarkan nilai *V(i)* :

# Metodologi dan Desain Sistem

* 1. **Gambaran Umum Sistem**

Secara umum, skema sistem yang digunakan untuk penelitian ini mennggunakan skema *general* untuk metode-metode *Reinforcement Learning*, meliputi inisialisasi *environment* yang akan digunakan untuk penelitian, yaitu pembentukan infrastruktur; *preprocessing*, yaitu transformasi keadaan infrastruktur/*environment* dari persimpangan jalan menjadi *model* MDP, menentukan persentase kepadatan jalur menjadi *state* serta penentuan *action* yang diambil dari warna lampu semua jalur di persimpangan; proses pengambilan data untuk menentukan *transition model/transition probability matrix*, yaitu dengan melakukan observasi perpindahan antar *state* serta *action* yang dilakukan ketika berpindah *state*;proses *training/learning* yaitu mencari *optimal policy* dengan melakukan *learning* pada *value iteration* menggunakan data *transition probability* yang telah diambil sebelumnya; *testing*/pengujian menggunakan infrastruktur yang sama dengan yang digunakan untuk pengambilan data serta *training* serta pengambilan hasil *testing*, blok diagram pada ***Gambar 3-1*** menunjukan tahapan kerja sistem yang digunakan.



**Gambar 3‑1. Blok Diagram garis besar sistem.**

* 1. **Deskripsi Kebutuhan Sistem**

Tugas Akhir ini dalam pelaksanaan serta implementasinya menggunakan beberapa perangkat lunak dan keras, serta memiliki beberapa kebutuhan fungsional, berikut adalah spesifikasinya :

* + 1. **Kebutuhan Fungsional**

Berikut ini adalah kebutuhan fungsional dalam sistem yang digunakan dalam tugas akhir ini :

1. Dapat mengubah keadaan infrastruktur persimpangan jalan pada *simulator* menjadi model matematis MDP (*preprocessing*).
2. Dapat mengambil data kepadatan persimpangan pada simulator dan disimpan sehingga dapat digunakan untuk *training* MDP.
3. Dapat melakukan *training* MDP menggunakan data yang sudah diambil dan disimpan serta mampu dilakukan perubahan parameter *training*.
4. Dapat menerapkan hasil *training* kembali ke *simulator*, serta dapat menampilkan hasil *testing*.
   * 1. **Spesifikasi Perangkat Keras**

Berikut adalah spesifikasi Perangkat Keras/*Hardware* yang digunakan pada Tugas Akhir ini :

1. Processor Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz.
2. 16GB RAM.
3. 128GB SSD dan 2.5TB HDD.
   * 1. **Spesifikasi Perangkat Lunak**

Berikut adalah spesifikasi Perangkat Lunak/*Software* yang digunakan pada Tugas Akhir ini :

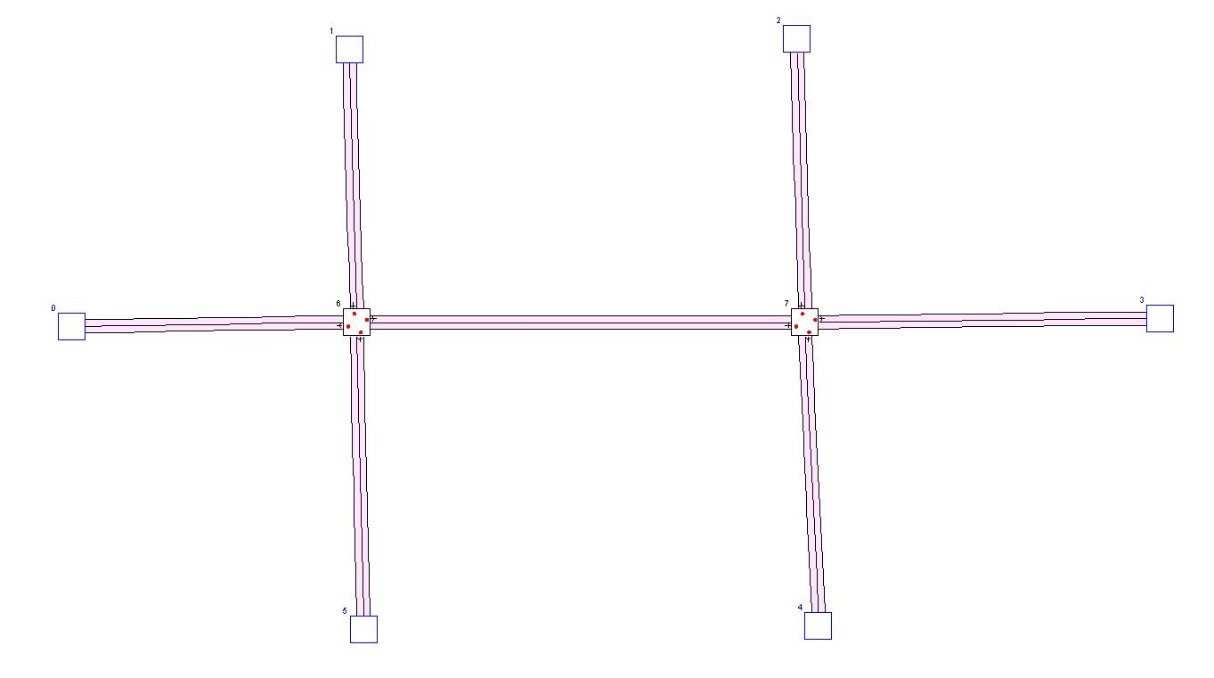
1. Sistem Operasi Windows 7 64 bit.
2. Netbeans IDE versi 8.0.2.
3. JDK versi 1.8.
4. *Green Light District Simulator*.
5. SourceTree 1.8.3.0 dengan Github VCS.
6. Microsoft Office Excel 2013.
   1. **Desain Sistem**

Berdasarkan langkah/tahapan umum yang dilakukan sistem, maka dapat dijabarkan detail desain sistem yang diimplementasikan dan digunakan dalam penelitian.



### Infrastruktur/*environment*

Infrastruktur yang digunakan bersifat tetap untuk semua tahap dan skenario. Infrastruktur yang digunakan terdiri dari 2 buah *junction*/persimpangan, dimana masing-masing *junction* memiliki 4 lampu, 4 jalur masuk dan 4 jalur keluar, sehingga terdapat 4 jalan (*road*), dengan masing-masing jalan memiliki 2 jalur yang masing-masing berlawanan arah, serta terdapat 6 buah *edge node* tempat *spawn* mobil di titik akhir jalan, seperti pada ***Gambar (3-2).***



**Gambar 3‑2. Infrastruktur yang digunakan untuk penelitian.**

Kendaraan yang dimunculkan*/spawn* bertipe *car*, jumlah kendaraan muncul dari masing-masing *edge node* berdasarkan *spawn rate* per *time step* yang ditentukan pada *edge node* tersebut, pada setiap *junction* masing-masing kendaraan dapat menyebrang jalan melewati *junction* dengan 3 kemungkinan, yaitu belok kiri, kanan dan lurus, kemudian secara *default* setiap kendaraan memiliki tujuan akhir *edge node* yang di-*generate* secara *random* oleh *simulator*, pola *random* yang digunakan bersifat tetap untuk semua skenario khusus untuk *spawn* kendaraan, karena sebuah angka *seed* digunakan untuk mengatur pola random.

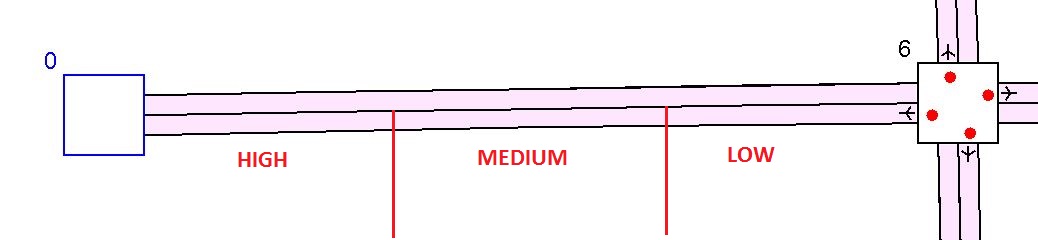
### Desain MDP

MDP yang digunakan memiliki desain yang dibuat berdasarkan keadaan infrastruktur, *state* diambil berdasarkan kepadatan jalan, *action* yang berupa konfigurasi lampu hijau di salah satu lampu pada satu junction per *time step*, *agent* berupa *junction* sehingga terdapat 2 *transition matrix*, kemudian *immediate reward* yang secara *default* diberikan berdasarkan *state*.

#### *State*

Representasi *state* yang digunakan berupa kombinasi *segment* dari setiap jalur masuk pada *junction*, karena terdapat 4 jalur masuk setiap *junction*, maka satu *state* merupakan gabungan 4 *segment* jalur, *segment* tersebut dibagi berdasarkan kepadatan pada jalur yang dimaksud, misalnya apabila 1 jalur dibagi menjadi 3 *segment* dengan kriteria sebagai berikut :

* Kepadatan Jalur ≤ 33%, *segment* = LOW (L).
* Kepadatan Jalur > 33% dan ≤ 66%, *segment* = MEDIUM (M).
* Kepadatan Jalur > 66%, *segment* = HIGH (H).



**Gambar 3‑3. ilustrasi pembagian segment pada satu jalur.**

*State* yang dihasilkan berdasarkan konfigurasi tersebut dapat dijabarkan sebagai berikut :

* *State* per *junction* merupakan susunan *segmen* (*Low, Medium, High*) dari masing-masing jalur, sehingga menjadi kumpulan *segment* 4 jalur, diantaranya *LLLL* (*Low, Low, Low, Low*), *LMHM* (*Low, Medium, High, Medium*)*, HLHL* (*High, Low, High, Low*), dan lain lain.
* Berdasarkan *state* diatas, maka dapat dihitung kemungkinan total *state* yang *reachable*, per *junction* memiliki total 4 jalur masuk dimana masing-masing jalur memiliki 3 *segment*, total kemungkinan *state* yang *reachable* adalah *jumlah\_segmentjalur* yaitu sebanyak 34 = 81 *states*.

#### *Action*

*Action* per *junction* merupakan lampu yang di-*switch* hijau di salah satu jalur, setiap *action* diatur sehingga terjadi setiap 5 *time step*, sedangkan *cycle spawn* kendaraan tetap pada 1 *time step*, hal ini dilakukan agar pergantian warna lampu tidak terlalu cepat. *Action* direpresentasikan dengan angka (0-3), dimana *action* 0 adalah lampu pada jalur nomor 0 (*lane\_*0) yang dihijaukan sedangkan pada jalur nomor 1 hingga 3 lampu dimerahkan, sehingga jumlah kemungkinan *action* yang dapat terjadi pada setiap transisi *state* pada MDP ini adalah 4 *action* .

#### *Transition Matrix*

*Transition Matrix* atau Model Transisi dibentuk melalui observasi perpindahan *state* pada infrastruktur, saat setiap terjadinya *action* per 5 *time step*, akan dilihat *initial state* serta *next state* nya serta dihitung jumlah terjadinya, kemudian akan dibagi dengan jumlah keseluruhan *initial state* melakukan *action* tersebut seperti pada teori **bab 2.5.1**, setiap *junction* memiliki satu *transition matrix*, sehingga untuk infrastruktur ini seperti pada **bab 3.3.1** terdapat 2 *transition matrix,* contoh potongan salah satu tabel transisi yang dapat dibentuk berdasarkan sebuah *initial state* yaitu pada **Tabel 3-1**.

**Tabel 3‑1. Potongan Tabel Transisi dari initial state LLLL pada model 3 segment**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Initial State*** | ***Action*** | ***Transition Probability*** | ***Next State*** |
| LLLL | 0 | 0.775 | LLLL |
| LLLL | 1 | 0.804511278 | LLLL |
| LLLL | 2 | 0.808823529 | LLLL |
| LLLL | 3 | 0.783216783 | LLLL |
| LLLL | 0 | 0.016666667 | LLML |
| LLLL | 1 | 0.060150376 | LLML |
| LLLL | 2 | 0 | LLML |
| LLLL | 3 | 0.06993007 | LLML |
| LLLL | 0 | 0 | LLHL |
| LLLL | 1 | 0 | LLHL |
| LLLL | 2 | 0 | LLHL |
| LLLL | 3 | 0 | LLHL |
| LLLL | 0 | 0 | LLHM |
| LLLL | 1 | 0 | LLHM |
| LLLL | 2 | 0 | LLHM |
| LLLL | 3 | 0 | LLHM |
| …… | …… | …………. | …….. |

#### *Immediate Reward*

*Immediate Reward* yang digunakan untuk semua skenario MDP adalah sama, yaitu ditentukan berdasarkan kombinasi *segment* jalan pada satu *state*, dengan *rule* sebagai berikut :

* Semua *immediate reward* diberikan pada semua *state* baik yang *reachable* maupun yang *unreachable*.
* Masing-masing *state* mendapatkan nilai *immediate reward* sesuai dengan kombinasi *segment state* yang dimilikinya.
* Untuk *segment Low* (*L*), *immediate reward* = +20;
* *Segment Medium* (*M*) = -10.
* *Segment High* (*H*) = -20.

Misalnya apabila sebuah *state* terdiri dari kombinasi *segment LLLL*, maka jumlah *immediate reward* yang diperoleh *state* tersebut adalah 20x4=80 *reward*. Semakin rendah kepadatan *segment* maka *reward* semakin tinggi diberikan, hal ini dilakukan agar *final policy* yang didapatkan dari MDP mampu meminimalisasi kepadatan jalur, sehingga *waiting time* yang diperoleh dapat optimal.

#### *Value Iteration*

*Value Iteration* dilakukan untuk mendapatkan *final policy*, yaitu *best action* yang dapat dilakukan oleh suatu *state* untuk semua *state space* yang ada pada MDP, langkah-langkah yang dilakukan *value iteration* adalah langkah pada **bab 2.5.4**, diawali dengan menentukan beberapa parameter untuk *value iteration*, yaitu *threshold* kriteria berhenti iterasi serta *gamma*/*discount factor*, kemudian iterasi dilakukan hingga *state utility* konvergen atau dengan kata lain memiliki *delta* dibawah *threshold*. *Final policy* yang dihasilkan oleh MDP bersifat per *junction* karena observasi dilakukan terhadap masing-masing *junction* dan setiap *junction* memiliki *transition matrix* nya sendiri, Salah satu contoh potongan tabel *final policy* yang dihasilkan dari *value iteration* pada MDP ini terdapat pada **Tabel 3-2**.

**Tabel 3‑2. Potongan Final Policy hasil Value Iteration pada salah satu junction dengan model 3 segment**

|  |  |  |
| --- | --- | --- |
| ***State*** | ***State's Utility*** | ***Best Action*** |
| LLLL | 717.9390439 | 2 |
| LLML | 681.487652 | 2 |
| LLHL | 617.2691233 | 2 |
| LLHM | 481.5778086 | 2 |
| LMHL | 479.6334292 | 2 |
| LHHL | 347.8532014 | 2 |
| LHHM | 170.2146193 | 3 |
| MHHL | 176.4912432 | 2 |
| MMHM | 103.1931574 | 0 |
| HMHM | -54.87728675 | 1 |
| HHHL | -47.76314791 | 1 |
| LLLM | 676.8445801 | 3 |
| MLML | 633.3388868 | 0 |
| LMLL | 664.4563576 | 1 |
| LMLM | 584.4558223 | 3 |
| LMML | 595.9604343 | 1 |
| …….. | …………… | ……… |

* 1. **Skenario Pengujian**

Pengujian dilakukan diawali dengan melakukan klasifikasi kepadatan jalur berdasarkan *spawn rate* pada setiap *edge node*, untuk memisahkan skenario keadaan lalu lintas sepi, sedang, atau padat. Kemudian selanjutnya dilakukan pengujian untuk algoritma TLC dasar yang terdapat pada GLD yang akan digunakan sebagai pembanding untuk TLC yang menggunakan MDP. Selanjutnya dilakukan pencarian parameter MDP yang paling baik dalam membuat *final policy* untuk lalu lintas, setelah itu akan dibandingkan hasil performansi MDP yang memiliki parameter terbaik dengan algoritma dasar TLC. Semua pengujian untuk satu parameter dilakukan sebanyak 30 kali, hal ini karena dalam pengujian mengandung sifat *randomness*.

Parameter yang digunakan dalam pengujian merupakan sebuah parameter *waiting time* dari GLD, yaitu *average junction waiting time* (AJWT), karena belum jelas bagaimana persamaan dasar yang digunakan untuk menghitung AJWT pada GLD, maka digunakan perbandingan dengan beberapa algoritma TLC lainnya untuk melihat performa MDP dan menentukan berada di posisi manakah MDP dibandingkan algoritma lainnya.



### Klasifikasi Kepadatan Jalur

Klasifikasi kepadatan jalan dilakukan untuk memisahkan skenario kepadatan lalu lintas, yaitu sepi, sedang, dan padat, klasifikasi dilakukan berdasarkan *spawn rate* dari semua *edge node*, disini diberi batasan *edge node* memiliki nilai *spawn rate* yang statis serta semua *edge node* mendapatkan nilai *spawn rate* yang sama, *spawn rate* diatur dengan *seed* untuk *random* sehingga semua kemunculan *random* memiliki pola yang sama untuk setiap skenario, *seed* yang digunakan berupa nilai bertipe *long*, **Tabel 3-3** menunjukkan *spawn rate* dan *seed* yang digunakan untuk klasifikasi.

**Tabel 3‑3. Parameter Spawn Rate dan Seed untuk semua skenario pengujian**

|  |  |  |
| --- | --- | --- |
| **No** | **Spawn Rate** | **Seed** |
| 1 | 0.135 | 15032016 |
| 2 | 0.25 | 15032016 |
| 3 | 0.4 | 15032016 |

Semua skenario pengujian akan menggunakan parameter *spawn rate* dan *seed* seperti pada **Tabel 3-3**.

### Algoritma *basic* TLC

Algoritma *basic* TLC *built-in* dari GLD *simulator* yang digunakan untuk perbandingan dengan MDP adalah *Random* TLC, *Longest Que* TLC, dengan tambahan algoritma *basic non-*yang diimplementasi untuk penelitian ini yaitu *Fixed Cycle* TLC. *Fixed Cycle* TLC adalah konfigurasi lampu lalu lintas statik normal yang biasa digunakan saat ini, yang *action* pergantian lampu nya berurutan, dan dengan pola yang tetap sepanjang waktu, semua algoritma melakukan perpindahan warna lampu setiap 5 *time step*.

### Optimasi parameter MDP

Dilakukan pengujian untuk mencari MDP dengan parameter terbaik, parameter yang dicari adalah *segment* per jalur dan *gamma* atau *discount factor*, dimana *segment* akan mempengaruhi *state space* dan jumlah *state* yang *reachable* sehingga jumlah *action* pada *final policy* yang diciptakan juga semakin besar atau kecil berbanding lurus dengan jumlah *segment* yang ditentukan, sedangkan *discount factor* akan mempengaruhi seberapa cepat MDP konvergen, kecepatan konvergen dihitung berdasarkan jumlah iterasi yang dilakukan pada *value iteration*, kecepatan konvergen dari *value iteration* juga akan mempengaruhi *final policy* atau kumpulan *best action* yang dapat diciptakan karena nilai *state utility* yang berbeda pada akhir iterasi. Beberapa tipe MDP yang dibentuk berdasarkan parameternya terdapat pada **Tabel 3-4**.

**Tabel 3‑4. MDP dengan masing-masing parameter pembeda yaitu Segment dan Gamma**

|  |  |  |
| --- | --- | --- |
| **MDP** | ***Segment* per jalur** | ***Gamma (discount factor)*** |
| MDP-1 | 3 segmen | 0.9 |
| MDP-2 | 5 segmen | 0.9 |
| MDP-3 | 5 segmen | 0.5 |
| MDP-4 | 5 segmen | 0.2 |

### Perbandingan MDP dengan *basic TLC*

Setelah mendapatkan MDP dengan parameter terbaik pada tahap pengujian sebelumnya, akan dibandingkan performa MDP dengan semua algoritma *basic* TLC dan dilihat posisi/kedudukan MDP berdasarkan AJWT-nya. Semua parameter pengujian kepadatan lalu lintas digunakan juga untuk skenario ini.

# Pengujian dan Analisis

* 1. **Pengujian Sistem**

Pengujian sistem dilakukan untuk mengetahui performa algoritma MDP untuk TLC, selain itu melalui skenario pengujian yang berbeda dapat dapat diambil hasilnya dan melakukan analisis terhadap semua hasil kemudian mampu diambil kesimpulan pada kondisi/parameter yang seperti apakah sistem dengan algoritma MDP untuk TLC memiliki performansi optimal serta dapat melihat juga dimanakah posisinya dibandingkan algoritma lainnya.

* 1. **Tujuan Pengujian**

Adapun tujuan pengujian yang dilakukan pada penelitian ini adalah sebagai berikut :

* Mengetahui performansi algoritma MDP dalam menentukan konfigurasi lampu lalu lintas pada infrastruktur yang sudah ditentukan.
* Mencari parameter/kondisi yang dapat mengoptimalkan algoritma MDP dalam menentukan konfigurasi lampu lalu lintas.
* Mengetahui posisi performansi algoritma MDP apabila dibandingkan dengan algoritma TLC lainnya/konfigurasi standar lalu lintas.
  1. **Skenario Pengujian**

Skenario pengujian yang dilakukan berdasarkan tujuan pengujian dan **bab 3.4** yang telah dipaparkan adalah sebagai berikut :

* Penentuan kepadatan lalu lintas/klasifikasi kepadatan jalur
* Pengujian Algoritma *basic* TLC.
* Pencarian parameter optimal MDP untuk TLC.
* Perbandingan dan analisis performansi Algoritma MDP dengan Algoritma *basic* TLC.
  1. **Analisis Hasil Pengujian**



### Penentuan Kepadatan Lalu Lintas

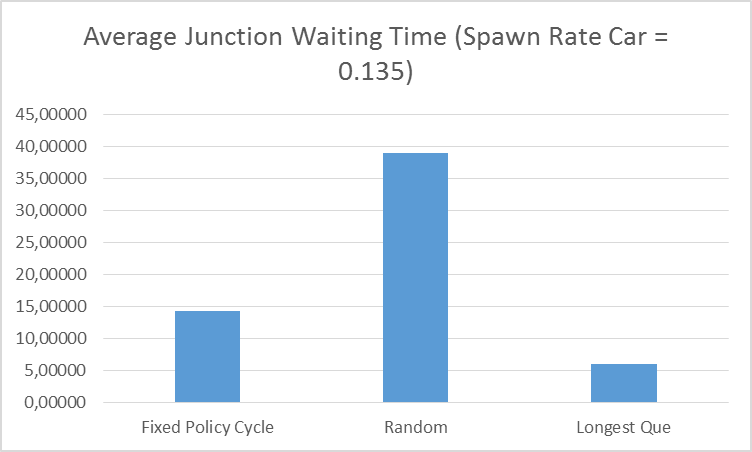
Hasil dari pengujian klasifikasi kepadatan lalu lintas terdapat pada **Tabel 4-1**, dapat dilihat bahwa *spawn rate* dari semua *edge node* yang terdapat pada infrastruktur adalah satu-satunya parameter yang mempengaruhi kepadatan lalu lintas di *simulator*, disini hasil ditentukan berdasarkan observasi seberapa panjang antrian kendaraan pada jalur yang diberi warna lampu merah tanpa ada kriteria parameter lainnya.

**Tabel 4‑1. Klasifikasi Kepadatan Lalu Lintas**

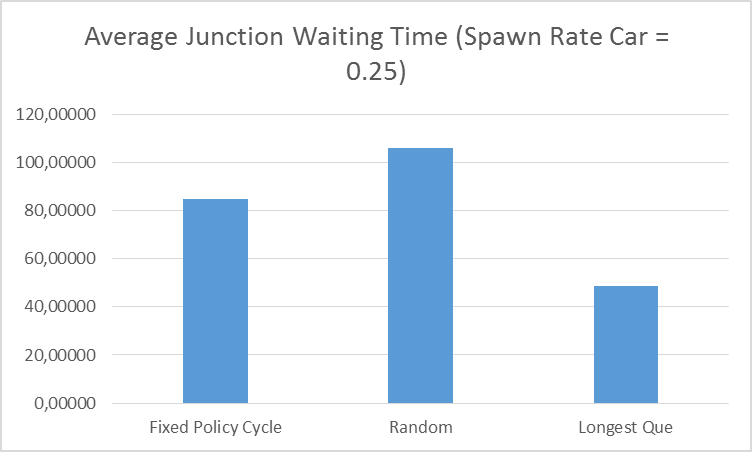
|  |  |  |
| --- | --- | --- |
| **No** | **Spawn Rate** | **Kepadatan Lalu Lintas** |
| 1 | 0.135 | Rendah |
| 2 | 0.25 | Sedang |
| 3 | 0.4 | Tinggi |

### Pengujian Algoritma *basic* TLC

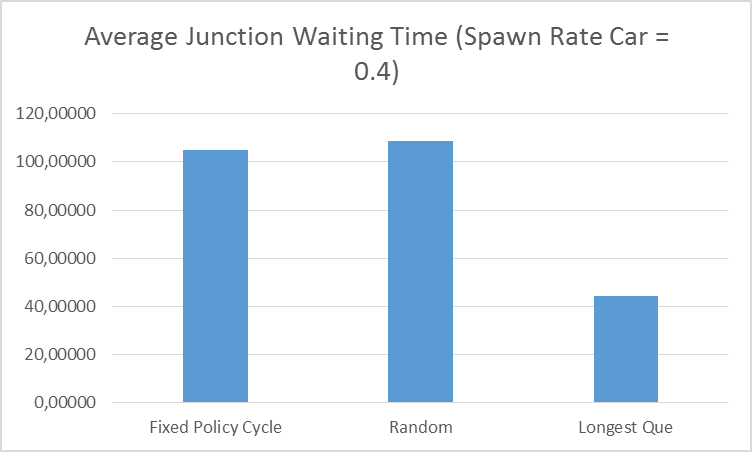
Pada pengujian ini dibandingkan beberapa algoritma *basic* TLC seperti yang sudah dijelaskan pada **bab 3.4.2**. Hasil pengujian algoritma *basic* TLC ini dapat dilihat pada **Gambar 4-1, Gambar 4-2, dan Gambar 4-3**, dapat dilihat bahwa diantara *basic* TLC *algorithm* yang lain, *Longest Que* TLC unggul pada semua skenario kepadatan lalu lintas dengan selisih AJWT yang sangat jauh dibandingkan algoritma lainnya, hal ini dikarenakan *Longest Que* TLC dapat meminimalkan AJWT dengan cara menghijaukan jalur yang paling padat, artinya kepadatan rata-rata dari semua jalur selalu minimal.



**Gambar 4‑1. AJWT basic TLC pada spawn rate (0,135).**



**Gambar 4‑2. AJWT basic TLC dengan spawn rate (0,25).**



**Gambar 4‑3. AJWT basic TLC dengan spawn rate (0,4).**

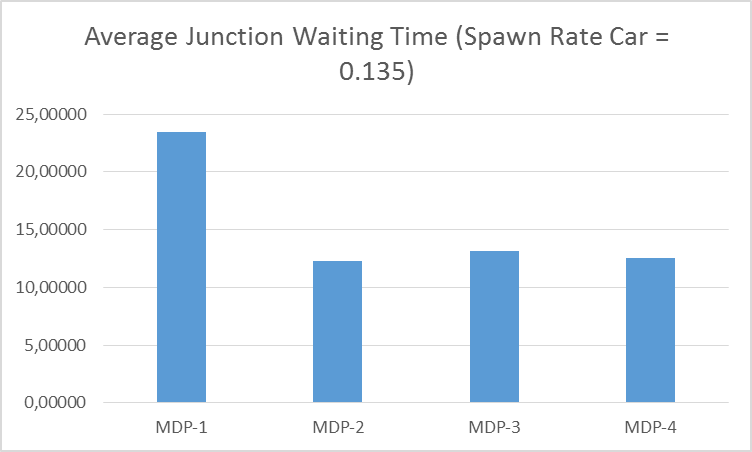
### Pencarian Parameter Optimal MDP

Pada pengujian ini, yang parameter MDP yang dioptimalkan adalah *segment* dan *discount factor*, skenario pengujian yang digunakan bisa dilihat pada **bab 3.4.3**. Hasil pengujian bisa dilihat pada **Gambar 4-4, Gambar 4-5, dan Gambar 4-6**.

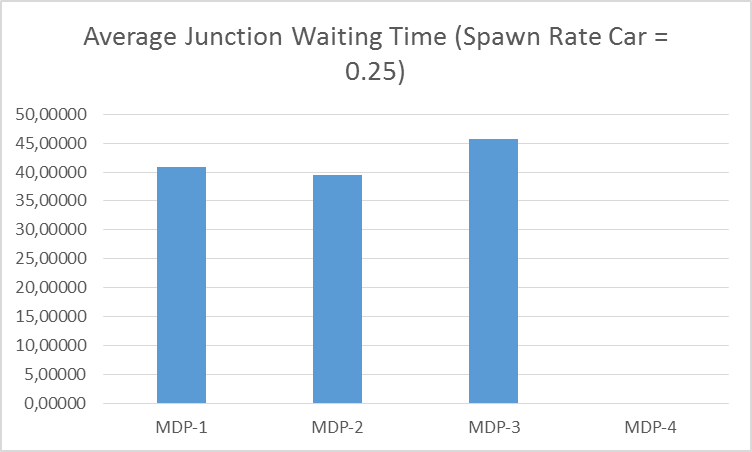
Untuk parameter *segment*, bisa disimpulkan bahwa MDP dengan 5 *segment* kepadatan per jalur mengungguli MDP dengan 3 *segment* kepadatan per jalur pada semua skenario, terutama pada skenario *spawn rate* 0.135 dan 0.4 atau kepadatan lalu lintas rendah dan tinggi, hal ini dikarenakan distribusi kepadatan kendaraan pada 5 *segment* memiliki *segment* dengan distribusi yang lebih tersebar dibandingkan 3 *segment*, misalnya, apabila pada 3 *segment* kepadatan dibawah 33% itu dianggap *Low*, maka pada 5 *segment* kepadatan dibawah 33% dibagi menjadi 2 *segment* kepadatan yaitu *Very Low* dan *Low*, sehingga MDP dengan 5 *segment* memiliki jumlah *action* yang lebih banyak dan tidak perlu menunggu hingga kepadatan mencapai 33% pada satu jalur untuk berubah *state*, maka pergantian *action* dapat lebih cepat.

Untuk parameter *discount factor*, yang paling unggul adalah *discount factor* dengan nilai 0,9 untuk semua kondisi kepadatan lalu lintas. *Discount factor* 0,9 unggul karena MDP dengan *discount factor* yang memiliki nilai dibawah 0,9 konvergen terlalu cepat/selesai iterasi sebelum waktunya ketika *value iteration*, artinya *threshold delta* sudah tercapai padahal *final policy* masih belum matang, atau dengan kata lain kurang terjadinya eksplorasi sehingga konvergen sebelum waktunya, sementara itu MDP dengan *discount factor* 0,2 mengalami *deadlock* untuk skenario kepadatan sedang dan tinggi, hal ini dikarenakan *final policy* yang dihasilkan sangat sedikit sekali mengambil faktor masa depan atau dengan kata lain tidak matang.

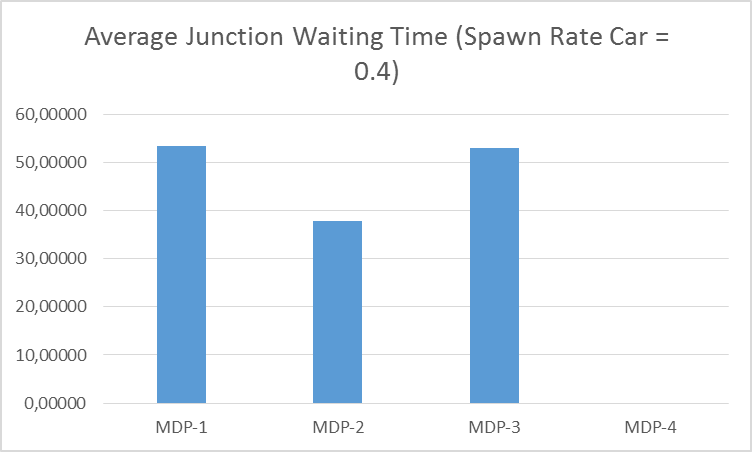
Dapat diambil kesimpulan bahwa MDP-2 adalah MDP dengan parameter terbaik apabila berdasarkan AJWT yang dihasilkan dari semua skenario kepadatan lalu lintas.



**Gambar 4‑4. AJWT MDP pada spawn rate (0,135).**



**Gambar 4‑5. AJWT MDP pada spawn rate (0,25).**



**Gambar 4‑6. AJWT MDP pada spawn rate (0,4).**

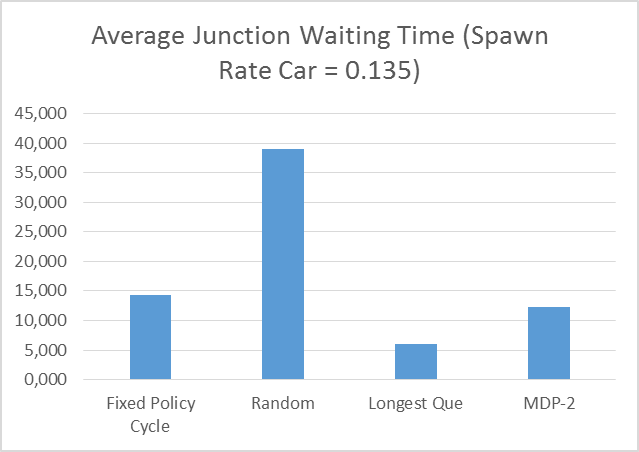
### Perbandingan *best* MDP dengan *basic TLC*

Berdasarkan pengujian pada **bab 4.4.3**, didapatkan MDP dengan parameter terbaik berdasarkan skenario pengujian yang telah ditentukan yaitu MDP dengan parameter 5 *segment per lane* dan *discount factor* 0,9, kemudian MDP terbaik ini dibandingkan dengan algoritma *basic* TLC lainnya untuk mengetahui posisi/kedudukan MDP berdasarkan performansinya, hasil perbandingannya dapat dilihat di **Gambar 4-7, Gambar 4-8,** dan **Gambar 4-9.**

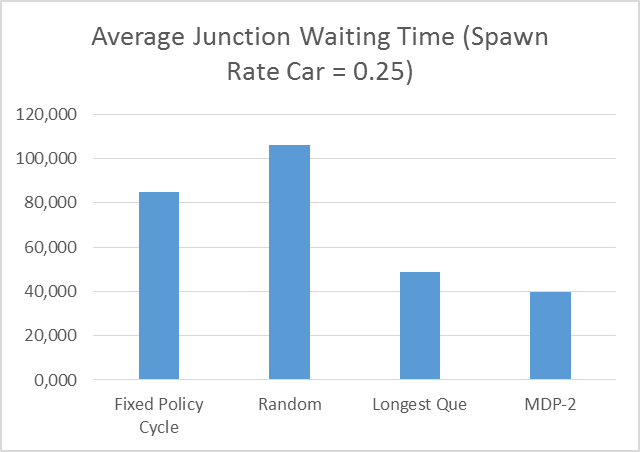
Pada skenario lalu lintas sepi, *Longest Que* unggul dengan AJWT terendah dibandingkan yang lainnya, MDP dikalahkan *Longest Que* pada lalu lintas rendah karena distribusi kepadatan MDP untuk kategori lalu lintas rendah belum mampu mengalahkan *Longest Que* yang sifatnya langsung menghijaukan jalur terpadat, sedangkan MDP harus menunggu untuk berpindah *state* dalam melakukan perubahan *action* dimana *state* dipengaruhi distribusi kepadatan jalur secara langsung, sehingga lebih cepat *Longest Que* dalam mengambil keputusan untuk keadaan lalu lintas rendah, apabila pembagian *segment* lebih banyak maka MDP akan dapat mengalahkan *Longest Que* pada keadaan lalu lintas rendah karena pengambilan *acion* yang lebih cepat, hal ini terbukti pada pengujian **bab 4.4.3** antara MDP dengan 3 *segment* dan 5 *segment*, MDP dengan 5 *segment* per jalur memiliki kecepatan mengambil *action* yang lebih cepat ketimbang MDP dengan 3 *segment* sehingga memiliki AJWT yang lebih rendah.

Pada skenario lalu lintas sedang dan tinggi, MDP mengungguli algoritma lainnya, karena MDP dapat mempertimbangkan keadaan kedepannya/*future condition* sehingga selalu memilih *action* terbaik berdasarkan *current state*.

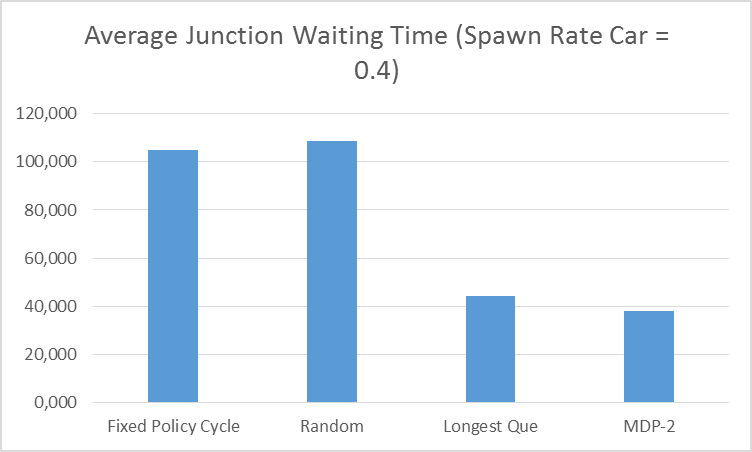
Apabila diambil rata-rata dari semua skenario, MDP mengungguli algoritma lainnya yang terdapat pada skenario pengujian dengan AJWT paling rendah, dapat dilihat pada **Gambar 4-10**.



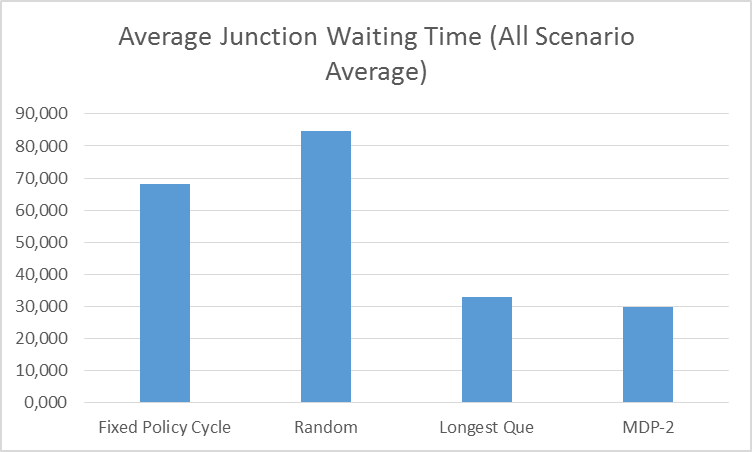
**Gambar 4‑7. AJWT MDP versus basic TLC dengan spawn rate (0,135).**



**Gambar 4‑8. AJWT MDP vs basic TLC dengan spawn rate (0,25).**



**Gambar 4‑9. AJWT MDP versus basic TLC dengan spawn rate (0,4).**



**Gambar 4‑10. AJWT MDP vs basic TLC - all scenario average.**

# Kesimpulan dan Saran

* 1. **Kesimpulan**

Berdasarkan analisis terhadap hasil pengujian dari beberapa skenario yang telah ditentukan, dapat diambil kesimpulan sebagai berikut :

* MDP dapat diimplementasikan dalam sistem TLC dan menghasilkan performansi yang baik, paling unggul diantaraalgoritma *basic* TLC lainnya.
* MDP dengan parameter *segment* per jalur yang semakin banyak maka akan semakin minimal AJWT yang dihasilkan, sedangkan untuk parameter *discount factor*, semakin kecil *discount factor* yang diberikan untuk suatu model MDP, maka MDP akan cenderung konvergen lebih cepat namun memiliki *final policy* yang tidak matang, sehingga semakin besar *discount factor* semakin baik *final policy* yang dihasilkan, selama *discount factor* kurang dari 1.
  1. **Saran**

Berdasarkan penelitian yang dilakukan dalam konteks TLC ini, masih banyak faktor yang dapat dikembangkan untuk memperbaiki atau meningkatkan performa sistem sehingga lebih optimal, berikut adalah saran-saran dalam pengembangan penelitian selanjutnya :

* Dapat dilakukan penelitian untuk MDP-TLC dengan *segment* per jalur yang lebih banyak.
* Melakukan penelitian untuk TLC menggunakan MDP yang representasi *state* nya diambil dari data yang lain.
* Melakukan penelitian untuk MDP dengan *transition matrix* yang dinamis, artinya terus melakukan tahap *learning* selama *running* simulasi.
* Melakukan penelitian untuk MDP dengan menggunakan parameter *spawn rate* yang dinamis atau berubah-ubah sepanjang waktu.
* Melakukan penelitian dengan kondisi/keadaan kendaraan yang lebih banyak atau memperhatikan jenis pengguna jalan lainnya serta memasukkan faktor eksternal penyebab terjadinya kepadatan lalu lintas.

**Daftar Pustaka**

|  |  |
| --- | --- |
|  | Khiang, Kok., Khalid, Marzuki., Yusof, Rubiyah. 1996. “Intelligent Traffic Light Control By Fuzzy Logic”. |
|  | Singh, Leena., Tripathi, Sudhanshu., Arora, Himakshi. 2009. “Time Optimization for Traffic Signal Control Using Genetic Algorithm”. |
|  | Hirankitti, Visit., Krohkaew, Jaturapith., Hogger, Chris. 2007. “A Multi-Agent Approach for Intelligent Traffic-Light Control”. |
|  | Wiering, Marco., Veenen, Jelle., Vreeken, Jilles., Koopman, Arne. 2004. “Intelligent Traffic Light Control”. |
|  | Abdoos, Monireh., Mozayani, Nasser., Bazzan, Ana. 2011. “Traffic Light Control in Non-stationary Environments based on Multi Agent Q-learning”. |
|  | Maria, Anu. 1997. “Introduction to Modeling and Simulation” |
|  | V.P. Singh. 2009. “System Modeling and Simulation”. |
|  | Burns, Pat. 2004. “Linear Congtuential Random Number Generators”. |
|  | Sutton, Richard., Barto, Andrew. 1998. “Reinforcement Learning: An Introduction”. |
|  | Steingrover, Merlijn., Schouten, Roelant., Peelen, Stefan., Nijhuis, Emil.,  Bakker, Bram. “Reinforcement Learning of Traffic Light Controllers Adapting to Traffic Congestion”. |

**Lampiran**

1. Tabel lengkap hasil pengujian sistem

