

DESENVOLVIMENTO DE SOFTWARE E SIMULAÇÃO DE SEMÁFORO INTELIGENTE

Jefferson Silvio Meireles de Matos¹

Herculano Haymussi De Biasi²

RESUMO

Com a atual evolução do mundo, a importância que o transporte tem em nossas vidas nos faz dependentes na utilização de sistemas como o semáforo. O semáforo é um sistema antigo e utilizado em todo o mundo com o mesmo padrão (verde passa, amarelo atenção e vermelho para). O objetivo do sistema proposto neste artigo é otimizar o tempo de espera em um semáforo. Para este fim utiliza técnicas de simulação para comparar o sistema atual com o proposto. A simulação modifica o sistema atual, que é utilizado com tempo fixo, por um sistema dinâmico, que se adapta ao fluxo de veículos. Desenvolveu-se um software em Java utilizando *threads* e após isto utilizou-se o software SUMO para simular e comparar o sistema semaforico atual com o proposto. Para que a simulação apresentasse um resultado próximo ao real foram utilizados dados como quantidade de carros, motos, ônibus, caminhões, fornecidos pelo IBGE, e dados semaforicos (tempo de verde, amarelo, vermelho) colhidos diretamente no local. A simulação foi executada em 86.400 segundos, equivalente a 24 horas. Os resultados mostraram que o sistema proposto é mais eficaz pois teve redução de 58,8% em relação ao tempo de espera do sistema atual, passando de 20h e 48min (sistema atual) para 8h 38min (sistema proposto). Com estes resultados é possível afirmar que o sistema proposto é eficiente na otimização do tempo.

Palavras-Chave: Semáforo; Simulação; Otimização.

1 INTRODUÇÃO

Diante de uma era tão tecnológica, sempre visando resolver problemas antigos como, encontrar curas para doenças, descobrir inteligência extraterrestre, etc. convém observar que todo ser humano já se deparou com um problema peculiar que é “a falta de tempo”. Tempo é um, senão o principal problema em que a humanidade vive hoje. Quem nunca teve que correr

¹ Graduando do Curso de Ciência da Computação da Universidade do Oeste de Santa Catarina de Videira; jefferson.matos@unoesc.edu.br

² Professor do Curso de Ciência da Computação da Universidade do Oeste de Santa Catarina de Videira; herculano.debiasi@unoesc.edu.br

contra o relógio por algum motivo? Não teve tempo de abraçar um ente querido enquanto podia? Como ter tempo para resolver a falta de tempo? Quantas horas um dia precisaria ter? Todas estas perguntas podem ser resumidas em apenas uma. Como otimizar o tempo? Existem várias técnicas e ferramentas disponíveis hoje em dia, mas mais uma vez o tempo seria um empecilho então como mudar ao redor e otimizar o tempo, sem que ninguém precise aprender nada ou fazer algo de diferente?

Imaginem uma encruzilhada onde um semáforo de quatro tempos funciona normalmente em x unidades de tempo para cada lado, nas quais motoristas e pedestres devem respeitar as cores (vermelha, amarelo e verde) que determinam se podem prosseguir ou não. Surge uma simples pergunta “Porque estou parado aqui se não há fluxo em nenhuma das vias?”. Para o motorista a resposta poderia ser “para não ser autuado com multa como prevê o artigo 208 do Código de Trânsito Brasileiro” e para o pedestre simplesmente para se manter seguro. Uma alternativa seria se desviar o semáforo contornando-o, ou talvez em vez disto reduzir o tempo do semáforo ou talvez o retirar do local, por que não? Agora com as informações e padrões do trânsito pode-se elaborar um algoritmo que aprenda com o fluxo e que saiba quais os horários, dias e pontos onde o tráfego é maior.

A proposta deste estudo é analisar o fluxo de tráfego de um ponto estratégico ou uma cidade e coletar dados que mostrem em números o tempo total de parada em um semáforo. Com estes dados levantados, simulações de ambiente foram executadas e um algoritmo foi desenvolvido com intuito de estudar e aprender a otimizar o tempo levando em consideração as informações coletadas (simulações, reais) e também parâmetros como tempo, horário, entre outros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos os softwares de simulação utilizados assim como a linguagem e técnicas de programação aplicadas.

2.1 SUMO

O SUMO é uma suíte de simulação de tráfego livre e aberta que está disponível desde 2001 (DLR, 2018). Permitindo a modelagem de sistemas de tráfego intermodal, incluindo veículos rodoviários, transporte público e pedestres. Incluído no pacote estão ferramentas de suporte que lidam com tarefas como localização de rotas, visualização, importação de rede e

cálculo de emissões de poluentes. O SUMO pode ser aprimorado com modelos personalizados e fornece várias APIs para controlar remotamente a simulação.

2.1.1 NETCONVERT

NETCONVERT é uma ferramenta do pacote SUMO usada para importar redes de estradas digitais de diferentes fontes e gerar redes rodoviárias, que podem ser interpretadas por outras ferramentas do pacote (DLR, 2018). O Quadro 1 mostra as opções.

Quadro 1: Opções utilizadas na ferramenta NETCONVERT

Opção	Descrição	Valor
--osm-files	Informa uma entrada OSM XML.	nome_arquivo.osm
-o	Informa a saída em NET XML.	nome_arquivo.net.xml
--output.street-names	Inclui os nomes das ruas.	true
--output.original-names	Mantem os nomes originais das ruas.	true
--no-turnarounds	Desabilita a criação de retornos.	
--tls.join	Agrupar nós próximos em um mesmo semáforo.	
--junctions.join	Agrupar junções próximas.	
--tls.guess-signals	Controla semáforos próximos em um mesmo controle.	
--osm.elevation	Leva em consideração se existe elevações.	

Fonte: (DLR, 2018)

2.1.2 POLYCONVERT

POLYCONVERT é uma ferramenta do pacote SUMO usada para importar formas geométricas (polígonos ou pontos de interesse) de diferentes fontes, convertendo-as em uma representação que pode ser visualizada usando SUMO-GUI (DLR, 2018). O Quadro 2 mostra as opções utilizadas.

Quadro 2: Opções utilizadas na ferramenta POLYCONVERT

Opção	Descrição	Valor
--net-files	Informa uma entrada NET XML	nome_arquivo.net.xml
-osm-files	Informa uma entrada OSM XML	nome_arquivo.osm
--type-file	Inclui os nomes das ruas	typemap.xml
-o	Informa a saída em POLY XML	nome_arquivo.poly.xml

Fonte: (DLR, 2018)

2.1.3 RANDONTRIPS

RANDONTRIPS é uma ferramenta do pacote SUMO usada para gerar viagens aleatórias (DLR, 2018). O Quadro 3 exibe as opções utilizadas.

Quadro 3: Opções utilizadas na ferramenta RANDONTRIPS

Opção	Descrição	Valor
-n	Informa uma entrada NET XML	nome_arquivo.net.xml
-e	Tempo em Segundos	86400
-l	Probabilidade de borda de peso por comprimento	

Fonte: (DLR, 2018)

2.1.5 SUMO-GUI

SUMO-GUI é uma ferramenta do pacote SUMO que fornece uma interface gráfica para o usuário (DLR, 2018).

2.1.6 Traffic Lights

Normalmente, a ferramenta NETCONVERT gera semáforos e programas de ações para junções durante o cálculo das redes. Ainda assim, esses programas computados muitas vezes diferem daqueles encontrados na realidade. Para alimentar a simulação com programas reais de semáforos, é possível executar o SUMO / SUMO-GUI com definições adicionais do arquivo TLS (DLR, 2018).

Um arquivo TLS é composto pelos ciclos semaforicos (tempos, padrões, etc.). Para alterar a estrutura de um arquivo TLS pode-se carregar novas definições para semáforos como parte de um arquivo adicional. Opções de TLS (Quadro 4):

Cada caractere dentro do estado de uma fase descreve o estado de um sinal do semáforo. Uma única faixa pode conter vários sinais - por exemplo, um para veículos virando à esquerda e outro para veículos que se movem em linha reta. Isso significa que um sinal não controla pistas, mas *links* - cada um conectando uma pista que está entrando em uma junção e uma que está saindo desta mesma junção (DLR, 2018).

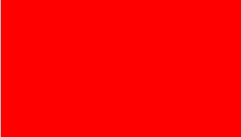



Quadro 4: Composição do arquivo TLS

Elemento	Atributo	Tipo de valor	Descrição
tlLogic	id	id (string)	Esse deve ser um código de semáforo existente no arquivo .net.xml. Normalmente, o id de um semáforo é idêntico ao id da <i>junction</i> .
	type	enum (static, actuated, delay_based)	O tipo de semáforo (durações de fase fixa (<i>static</i>), prolongamento de fase com base em intervalos de tempo entre veículos (<i>actuated</i>) ou na perda de tempo acumulada de veículos na fila (<i>delay_based</i>)).
	programID	id (string)	Este deve ser um novo nome de programa para o id do semáforo.
	offset	int	O deslocamento inicial do tempo do programa.
phase	duration	time (int)	A duração da fase.
	State	Lista de signal states	Os sinais de semáforo para esta fase.
	minDur	time (int)	A duração mínima da fase ao usar o tipo <i>actuated</i> .
	maxDur	time (int)	A duração mínima da fase ao usar o tipo <i>actuated</i> .

Fonte: (DLR, 2018).

As seguintes cores de sinal são usadas (Quadro 5),

Quadro 5: Cores de sinal

Personagem	Cor da GUI	Descrição
r		Os veículos devem parar.
y		Os veículos começarão a desacelerar se estiverem longe da junção, caso contrário eles passarão.
g		Sem prioridade - os veículos podem passar a junção se nenhum veículo usar um maior fluxo antecipado, caso contrário eles desaceleram para deixar passar.
G		Prioridade - veículos podem passar a junção.

Fonte: (DLR, 2018).

No SUMO, uma dependência de um para n entre sinais e *links* é implementada, isso significa que cada sinal pode controlar mais de um *link* - embora as redes geradas pelo NETCONVERT usem normalmente um sinal por *link*. Um semáforo também pode controlar as faixas que entram em diferentes cruzamentos. As informações sobre qual *link* é controlado por qual sinal de semáforo podem ser obtidos com a opção "*show link tls index*" nas configurações de visualização do SUMO-GUI ou a partir do atributo *linkIndex* dos elementos *<connection>* no arquivo .net.xml (DLR, 2018).

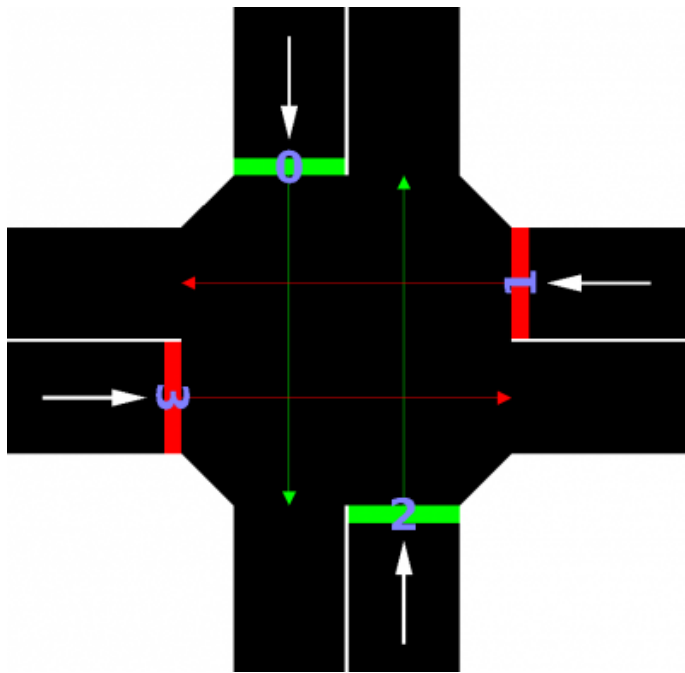


Figura 2: Exemplo de semáforo com estado atual "GrGr"

Fonte: (DLR, 2018).

A Figura 2 mostra um exemplo de semáforo com estado atual "GrGr". A letra mais à esquerda "G" codifica a luz verde para o *link* 0, seguida de "r" para o *link* 1, "G" para o *link* 2 e "r" para o *link* 3. Os números de *link* são ativados via configurações SUMO-GUI ativando a opção "*show link tls index*" (DLR, 2018).

2.2 JAVA

A linguagem de programação JavaTM é uma linguagem orientada a objetos concorrente, fortemente tipada e de uso geral. Normalmente, é compilada para um código intermediário *bytecode* e no formato binário definidos na especificação da *Java Virtual Machine* (ORACLE, 2018).

2.2.1 Processos e *Threads*

Na programação concorrente existem duas unidades básicas de execução: processos e *threads*. Na linguagem de programação Java, a programação concorrente está principalmente relacionada a *threads*. No entanto, os processos também são importantes.

Um sistema de computador normalmente tem muitos processos e *threads* ativos. Isso é válido mesmo em sistemas que possuem apenas um único núcleo de execução e, portanto, possuem apenas uma *thread* em execução em um determinado momento. O tempo de processamento de um único núcleo é compartilhado entre processos e o tempo em que cada processo/*thread* for executado é chamado *time slice* (fatia de tempo) (ORACLE, 2017).

3 DESENVOLVIMENTO

O projeto se baseia no desenvolvimento de um software que otimize o tempo de espera em um semáforo, e o compare através de simulação com o método atual. Os passos utilizados para desenvolvimento do sistema podem ser divididos em três etapas.

- Modelagem do sistema;
- Desenvolvimento do software de otimização;
- Execução das simulações.

3.1 MODELAGEM DO SISTEMA

A modelagem do sistema foi construída utilizando a ferramenta SysML que é uma linguagem de modelagem gráfica de propósito geral usada para especificar, analisar, projetar e verificar sistemas complexos que podem incluir hardware, software, informações, procedimentos e instalações (OMG, 2019).

Para representar o comportamento do semáforo foi utilizado o diagrama de estados também conhecido como diagrama de máquina de estados ilustrado na Figura 3. Ele tem como objetivo principal mostrar os estados e seus comportamentos mostrando os possíveis estados de um objeto e as transações responsáveis pelas suas mudanças de estado.

O processo é iniciado com a solicitação de permissão. Quando concedida é avaliado se o sensor está ocioso ou não. Se o sensor está ativo, passa para o processo de **siga**. Em seguida passa para o processo de **atenção** e finalmente para o processo de **pare** permanecendo em um ciclo.

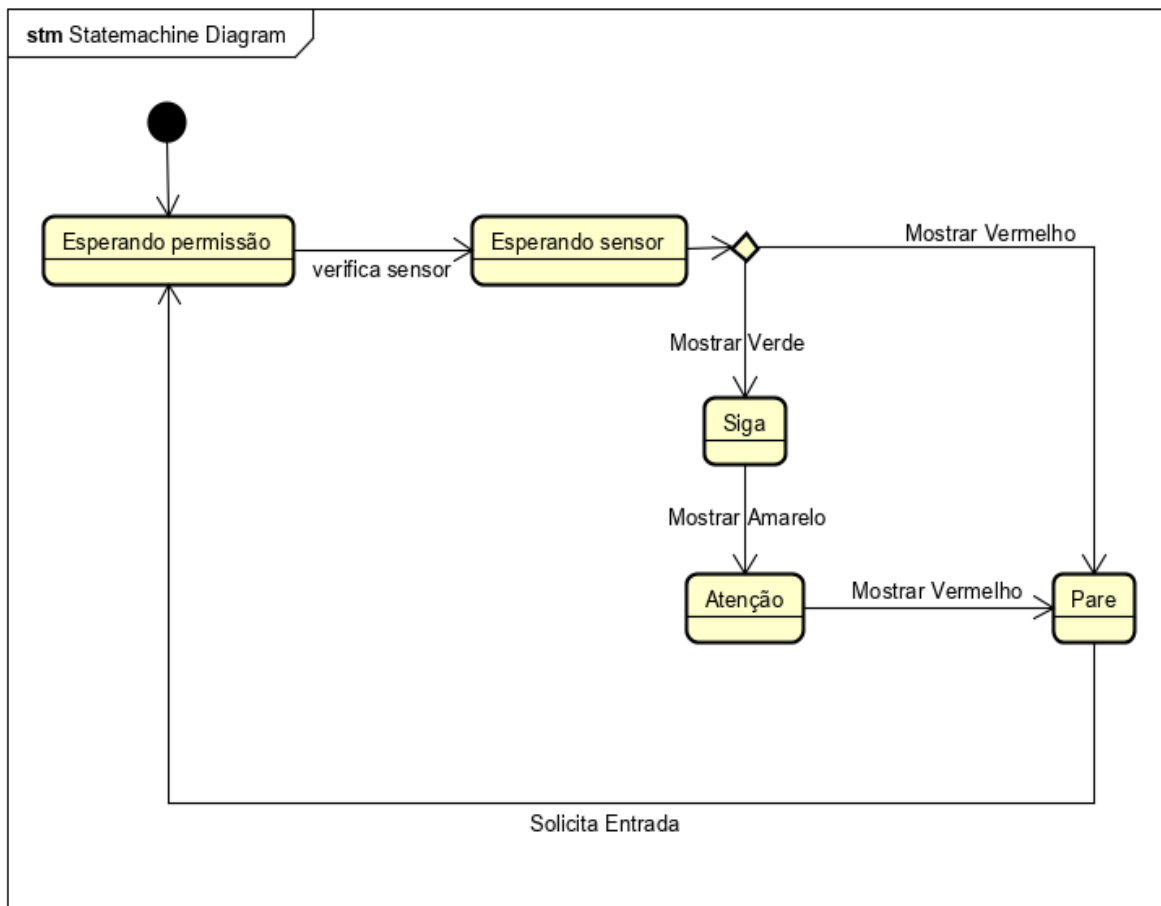


Figura 3: Diagrama de atividade

Fonte: Autor

3.2 SIMULAÇÃO

Para a etapa de simulação foi escolhida a ferramenta SUMO, cujas características atendem aos requisitos elencados. Para que fosse possível realizar a simulação as seguintes etapas tiveram de ser executadas:

- Definição de uma rede rodoviária (NETCONVERT);
- Criação de formas geométricas (POLYCONVERT);
- Desenvolvimento de um conjunto de rotas (RANDOMTRIPS).

Para que a simulação gerasse informações mais detalhadas sobre o semáforo em questão, foi criado um arquivo adicional denominado de `tls_rep.xml`, a saída deste arquivo contém todas as informações do semáforo.

Por fim a ferramenta SUMO-GUI é utilizada passando para ela o arquivo `cacador.sumocfg` Figura 4.

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.xsd">

  <input>
    <net-file value="cacador.net.xml"/>
    <route-files value="cacador.passenger.trips.xml,cacador.truck.trips.xml,
      cacador.bus.trips.xml,cacador.motorcycle.trips.xml"/>
    <additional-files value="cacador.poly.xml,cacador_stops.add.xml,tls_rep.xml"/>
  </input>

  <processing>
    <ignore-route-errors value="true"/>
  </processing>

  <routing>
    <device.rerouting.adaptation-steps value="180"/>
  </routing>

  <output>
    <tripinfo-output value="C:\Users\Jefferson\Desktop\SUMO\Tripnfo1.xml"/>
  </output>
</configuration>
```

Figura 41: Arquivo `cacador.sumocfg`

Fonte: Autor.

4 RESULTADOS

Neste capítulo é apresentada a simulação com os dados produzidos pelo *software*.

4.1 SIMULAÇÃO COM SISTEMA ATUAL

A simulação com o sistema semaforico atual utiliza os valores de vinte segundos para verde, três segundos para amarelo, com tipo *static* utilizando o padrão de oito fases. Após executar uma simulação de 86.400 segundos, obtiveram-se os seguintes resultados: em 20h 48min (87%) do tempo o semáforo executou o estado de verde e em 3h 07min (13%) o estado amarelo.

4.2 SIMULAÇÃO COM SISTEMA PROPOSTO

Para a simulação com o sistema semafórico proposto utilizou-se os valores de vinte segundos para verde com duração mínima de 5 e máxima de 60, cinco segundos para amarelo, com tipo *actuated* utilizando o padrão de oito. Após executar uma simulação de 86.400 segundos, equivalente a 24 horas, obteve-se os seguintes resultados, em 8h 38min (50%) do tempo o semáforo atual em estado de verde e nos 8h 42min (50%) em amarelo.

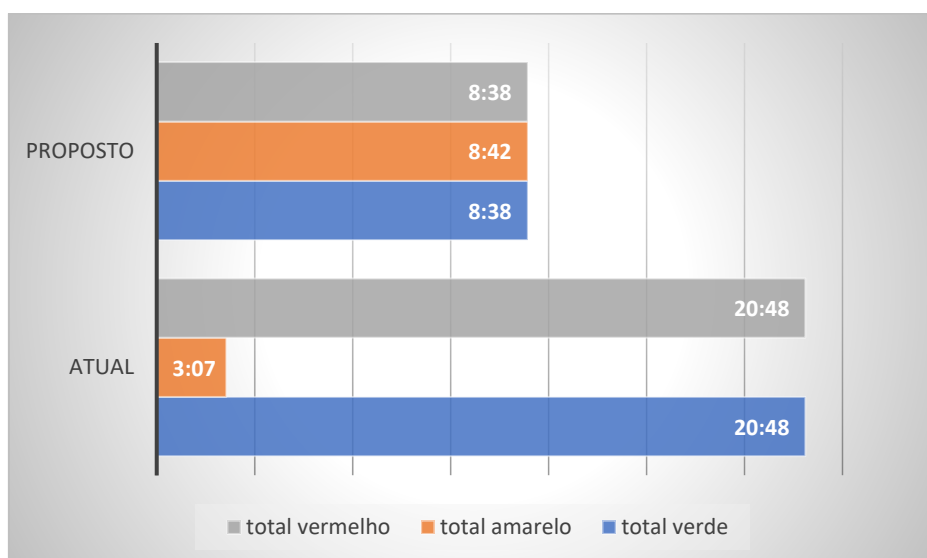


Figura 5: Comparação dos sistemas

Fonte: Autor

Através das simulações realizadas comparou-se os resultados e chegou-se às seguintes conclusões: como ilustra a Figura 5, o sistema proposto teve redução significativa no tempo de espera (estado vermelho) em relação ao sistema atual. O sistema atual fornece tempo de verde em excesso, o que implica em espera excessiva e desnecessária das demais partes.

5 CONCLUSÃO

Tendo em mãos os dados conclui-se que a principal vantagem em adotar o sistema proposto se dá na otimização do tempo do usuário. Sabendo que o sistema atual é antigo e defasado, analisando seus dados verifica-se um tempo de verde alto, porém o tempo de vermelho é o mesmo, o que significa tempo de verde sem uso e veículos em espera. No sistema proposto o valor chega a 58,88% de diferença ao sistema atual, significando menos tempo de

espera enquanto que no sistema atual tem-se 20h 48min de tempo parado, no proposto totalizou apenas 8h 38min. O sistema proposto atende ao requisito de otimização.

6 TRABALHOS FUTUROS

Como ações em potenciais, para trabalhos futuros pode-se apontar o aprimoramento no software para estender a utilização.

Outro fator importante a ser mencionado é referente a aplicação de visão computacional e possivelmente tratando de um sistema independente que gerencie e tome decisões como quanto ao sentido das vias, e comunicação com os veículos.

Por fim, outro trabalho futuro seria o desenvolvimento de uma interface mais amigável com possibilidade de análise em tempo real com sugestões de novas implementações nas vias urbanas como ruas, semáforos entre outras.

SOFTWARE DEVELOPMENT AND INTELLIGENT SEMAPHORE SIMULATION

ABSTRACT

With the current evolution of the world the importance that transport has in our lives makes us dependent on the use of systems such as traffic lights, the traffic light is old system and used all over the world with the same pattern (green to go, yellow to attention and red to stop), the objective of this system is to optimize the waiting time in a traffic light using simulation techniques to compare the current system with the proposed one, modifying the current system that is used with fixed time by a system that adapts to the flow of vehicles. The software was developed using threads so that the system obtained the optimization of the time and after the software SUMO was used to simulate and compare the current system with the proposed, so that the simulation presented a result close to the real one were used data such as (quantity of cars, motorcycles, buses, trucks), provided by IBGE and semaphore data collected directly in the place (time of green, yellow, red). The simulation was performed in 86.400 seconds, equivalent to 24 hours. The results showed that the proposed system is more efficient because it had a reduction of 58.8% in relation to the waiting time of the current system, going from 20h 48min (current system) to 8h 38min (proposed system), with this information it is possible to affirm that the proposed system is efficient in time optimization.

Key-words: Traffic light. Simulation. Optimization.

REFERÊNCIAS

- DLR. **Institute of Transportation Systems**, 2018. Disponível em: <https://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/>. Acesso em: 18 ago. 2018.
- DLR, S. SUMO. **SUMO SIMULATION OF URBAN MOBILITY**, 2018. Disponível em: <http://sumo.dlr.de/wiki/SUMO_User_Documentation>. Acesso em: 18 ago. 2018.
- IBGE. **IBGE. IBGE**, 2016. Disponível em: <<https://cidades.ibge.gov.br/brasil/sc/cacador/pesquisa/22/28120>>. Acesso em: 30 maio 2019.
- OMG. **OMG SysML. OMG Systems Modeling Language**, 2019. Disponível em: <<http://www.omgsysml.org/what-is-sysml.htm>>. Acesso em: 07 maio 2019.
- ORACLE. **THE JAVA TUTORIALS. ORACLE JAVA DOCUMENTATION**, 2017. Disponível em: <<https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html>>. Acesso em: 02 abr. 2019.
- ORACLE. **ORACLE. JAVA PROGRAMING LANGUAGE**, 2018. Disponível em: <<https://docs.oracle.com/javase/7/docs/technotes/guides/language/>>. Acesso em: 02 abr. 2019.