Otimização da Programação Semafórica em Tempo Real com Base em Modelagem Matemática

Eric W. L. Gonzaga
PPGEE
Universidade Federal de Minas Gerais
Belo Horizonte, MG – Brasil
ericgonzaga@ufmg.br

Paulo E. M. Almeida
Dep. Computação
Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte, MG – Brasil
pema@lsi.cefetmg.br

Eduardo G. Carrano
Dep. Engenharia Elétrica / PPGEE
Universidade Federal de Minas Gerais
Belo Horizonte, MG – Brasil
egcarrano@ufmg.br

***Resumo* - O atual congestionamento de veículos nas regiões urbanas traz a necessidade de ampliação da capacidade das redes viárias nas médias e grandes cidades. No entanto, a ampliação significativa dessas vias é, na maior parte dos casos, inviável devido a restrições econômicas ou físicas. Portanto, cabe a engenharia de tráfego encontrar alternativas menos onerosas para atenuar o impacto do aumento da frota de veículos e pedestres. A programação semafórica se mostra uma das estratégias mais interessantes para alcançar tal objetivo, tendo em vista seu potencial de resultados e seu baixo custo de implantação. Este trabalho propõe uma nova metodologia de otimização que utiliza um modelo matemático para avaliação das soluções candidatas. Essa abordagem se diferencia das clássicas por não depender do uso de simuladores de tráfego durante a execução do algoritmo, o que a torna sensivelmente mais rápida. Espera-se com esse ganho de tempo tornar a ferramenta viável para a programação semafórica em tempo real, o que não é possível com os algoritmos baseados em simulação. Testes preliminares sugerem que a ferramenta proposta é promissora para este tipo de problema.**

***Palavras-Chave* – Algoritmos genéticos, programação semafórica, otimização em tempo real, simulação de tráfego.**

I. INTRODUÇÃO

Médias e grandes cidades hoje convivem com sérios problemas de congestionamento de veículos em suas vias urbanas, especialmente nos horários de pico. Esse problema vem se agravando ainda mais com o tempo, devido ao crescente número de veículos e pedestres. Um relatório criado pela Companhia de Engenharia de Tráfego de São Paulo (CET-SP) [1] mostrou que o fluxo de veículos na cidade aumentou cerca de 40% entre 1997 e 2009. Assim, o problema de congestionamento traz a necessidade de expansão das vias, o que é, muitas vezes, inviável. Para diminuir tal impacto, uma engenharia de tráfego eficiente se torna essencial. Dentre as estratégias que podem ser adotadas, a programação semafórica se mostra relevante [2] e já é aplicada por diversas empresas de tráfego urbano [3].

Nesse contexto, [2] e [4] apresentam abordagens que têm por intuito ajustar a programação semafórica de forma a maximizar a velocidade média dos veículos dentro da rede de interesse. Entretanto, ambos os trabalhos dependem do uso de um simulador de tráfego para calcular a velocidade média dos veículos nos movimentos e, assim, avaliar as soluções candidatas. Uma vez que o simulador é computacionalmente dispendioso, a avaliação das soluções se torna um gargalo para a ferramenta de otimização, que pode levar horas para encontrar bons resultados. Essa característica torna impossível a obtenção de programações semafóricas em tempo real, o que pode ser interessante em cidades onde existe grande variação de fluxo ao longo do dia ou ao longo dos dias da semana.

Este trabalho propõe uma nova metodologia para lidar com esse problema. Nela, o simulador de tráfego é utilizado apenas uma vez, no início da execução, para construir um cenário para os fluxos de veículos em cada movimento. Estes fluxos são utilizados para construir o modelo matemático proposto em [5] que, por sua vez, é utilizado na avaliação dos indivíduos no processo de otimização. Com isso, a nova abordagem permite a avaliação de novas soluções candidatas em tempo computacional consideravelmente inferior, o que deve tornar possível a avaliação das condições de tráfego em tempo real ou a modelagem da incerteza relacionada à variabilidade das informações.

O restante deste documento está estruturado como segue: A fundamentação teórica deste trabalho é apresentada na Seção II. A metodologia proposta e o modelo matemático são descritos na Seção III. Os resultados pelo algoritmo desenvolvido obtidos em testes preliminares são exibidos na Seção IV. Por fim, conclusões e propostas de continuidade são discutidas nas seções V e VI respectivamente.

II. FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados os conceitos de programação semafórica e de movimentos, fundamentais para o entendimento do problema tratado.

A. Programação Semafórica

A programação semafórica consiste em determinar os tempos de ciclo, tempos de verde e defasagens dos conjuntos de semáforos instalados nas interseções entre vias da rede [6]. O tempo de ciclo é definido como o somatório dos tempos de verde e amarelo de cada conjunto de semáforos. Em geral, essa programação é feita com o intuito de minimizar o tempo médio (*delay*) que os veículos gastam para percorrer o trecho em

análise. Assim, o plano semafórico deve conter as seguintes configurações:

- Tempo total dos ciclos.
- Número de fases dos semáforos, que controlam um ou mais movimentos.
- Sequência e dimensão dos estágios, correspondentes aos tempos de verde, amarelo e vermelho de um semáforo, para cada fase.
- Sincronização entre os ciclos (defasagens), quando for necessário.

Dentre as principais referências utilizadas na engenharia de tráfego têm-se: o Manual de Semáforos [6], o Highway Capacity Manual 2010 (HCM-2010) [7] e o Manual Brasileiro de Sinalização de Trânsito [8]. Uma descrição extensiva do problema abordado pode ser encontrada nesses documentos.

B. *Movimentos e Fluxos de Veículos*

O termo movimento é usado para identificar a origem e o destino de veículos ou pedestres. Estes são graficamente representados por setas e são definidos de acordo com os trajetos permitidos em um cruzamento ou uma bifurcação. Por exemplo, em um cruzamento entre duas ruas de sentido único, existem quatro movimentos possíveis para os veículos que chegam a ele, conforme mostra a Fig. 1.

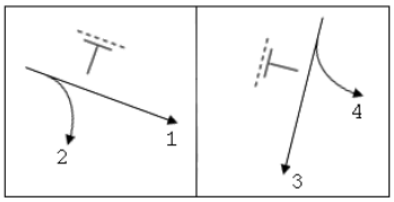

Fig. 1. Exemplo de movimentos em um cruzamento.

O fluxo de veículos, ou volume de tráfego, é o número de veículos que passa por uma determinada via, ou movimento, em um determinado período de tempo [8]. Na prática, os fluxos de veículos são dados a partir da contagem efetuada pelas empresas de tráfego em determinados períodos de tempo. No entanto, simuladores de trafego podem ser utilizados para gerar estes fluxos, seja para avaliar ferramentas de programação, estimar cenários em redes a serem instaladas/expandidas ou gerar cenários alternativos para o fluxo de veículos em um determinado trecho. Como exemplo, a aplicação de um simulador no cruzamento da Fig. 1 deve retornar quantos veículos realizam cada um dos quatro movimentos permitidos naquele trecho. A quantidade de veículos deve ser condizente com o total de veículos que entra no trecho e atender todas as restrições de balanceamento (todos os veículos que chegam ao cruzamento pela avenida horizontal devem seguir 1 ou 2 e todos os veículos que chegam pela avenida vertical devem seguir 3 ou 4).

Os dados de fluxos, sejam eles reais ou simulados, são utilizados como parâmetros de entrada para o problema de otimização da programação semafórica.

III. METODOLOGIA

Nesta seção é apresentada a arquitetura proposta para otimização das programações semafóricas, o algoritmo de otimização desenvolvido para os testes preliminares e o modelo matemático proposto em [5] para calcular os *delays* médios.

A. *Arquitetura Proposta*

As referências [2] e [4] utilizam o simulador de tráfego GISSIM, construído sobre o OpenJUMP [9], para gerar os fluxos de veículos em cada movimento e também para medir os tempos gastos pelos veículos para percorrer o trecho sob estudo (*delay*). Com isso, o algoritmo de otimização empregado executa $n * m$ vezes o simulador de tráfego, sendo n o total de iterações e m o tamanho da população, conforme o diagrama da Fig. 2. Como o tempo de simulação é relativamente alto, os algoritmos de otimização se tornam lentos e podem levar horas para oferecer bons resultados.

Fig. 2. Arquitetura do algoritmo de otimização em [2].

A proposta deste trabalho é alterar esta arquitetura para a apresentada na Fig. 3, onde o simulador de tráfego é executado apenas uma vez, no início do processo de otimização, com o intuito de gerar os fluxos de veículos nos movimentos. Nessa arquitetura, a avaliação dos *delays* é feita a partir do modelo matemático proposto em [5], que possui um custo computacional muito menor, tornando o processo de otimização muito mais rápido. Para a geração inicial do cenário foi adotado o Simulation of Urban MObility (SUMO) [10], que é um simulador de tráfego gratuito. No entanto, vale destacar que este poderia ser substituído por qualquer outro simulador ou, ainda, por dados reais de contagem de veículos nos movimentos, caso estes estivessem disponíveis.

Fig. 3. Arquitetura do algoritmo de otimização proposto.

B. *Algoritmo de Otimização*

Para realizar o processo de otimização foi desenvolvido neste trabalho um algoritmo genético (AG) que recebe do simulador de tráfego um cenário com os fluxos de entrada de veículos em cada movimento e, ao final, tem como resposta a melhor programação semafórica encontrada por ele para a rede.

O AG foi implementado tendo indivíduos modelados como vetores de números inteiros, onde cada posição representa o tempo de verde de um semáforo, enquanto os ciclos são dispostos sequencialmente, conforme exemplo na

Fig. 4 para um trecho com dois ciclos, cada um com quatro semáforos.

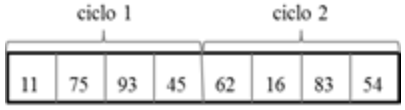

Fig. 4. Exemplo de indivíduo no AG com 2 ciclos, cada um com 4 semáforos.

A primeira população é composta por indivíduos gerados aleatoriamente, onde os tempos de verde dos semáforos devem apenas estar dentro do intervalo de verde permitido (I_{verde}). O processo evolutivo do GA é, então, composto por quatro etapas: seleção, cruzamento, mutação e avaliação.

A seleção define quais indivíduos serão utilizados como pais para as próximas populações. Esta é realizada utilizando o operador *Stochastic Universal Sampling* (SUS) [11].

A população de indivíduos selecionada é dividida em pares e o cruzamento entre eles ocorre conforme uma taxa de cruzamento T_{cross}. Para tal, foram utilizados dois operadores: o cruzamento de dois pontos (*two point crossover*) e o cruzamento uniforme (*uniform crossover*) [12]. A escolha entre qual método utilizar em cada cruzamento é feita de forma aleatória, seguindo distribuição uniforme.

Após o cruzamento, é avaliada a probabilidade de cada indivíduo da população sofrer mutação, com um taxa T_{mut}. Uma vez que o indivíduo é escolhido para mutação, são escolhidas posições aleatórias neste indivíduo e, para cada posição, é escolhido aleatoriamente um novo valor inteiro no intervalo $[v_{atual} - \Delta; v_{atual} + \Delta]$, onde v_{atual} é o valor atual da variável e Δ é a máxima perturbação admissível pela mutação. Caso necessário, estes intervalos saturam tanto no limite inferior quanto no limite superior da variável. Além disso, as escolhas relacionadas à posição e novo valor são realizadas com base em distribuições uniformes.

Por fim, a *fitness* de cada um dos indivíduos da nova população é avaliada utilizando o modelo matemático proposto em [5]. Caso o melhor indivíduo encontrado até o momento não esteja presente na população da geração, este substitui o pior indivíduo desta e a próxima iteração se inicia. O processo termina quando o número máximo de gerações é atingido.

C. Modelo Matemático para Cálculo do Delay

Como citado anteriormente, o uso do simulador de tráfego para medir os *delays* dos veículos em um trajeto torna o processo de otimização de programação semafórica lento, impossibilitando seu uso em alguns casos, como na programação semafórica em tempo real.

Assim, foi utilizado o modelo matemático proposto em [5] para estimar os *delays* e, por consequência, avaliar cada solução candidata obtida pelo algoritmo genético. Este modelo é apresentado em (1) e (2).

$$t = \sum_{m \in M} t_m \qquad (1)$$

$t \rightarrow$ *delay* médio total dos veículos no trecho (s).
$M \rightarrow$ conjunto de movimentos possíveis no trecho.

$$t_m = t_{0_m} + 0.25 * T_f * \left[z + \left(z^2 + \frac{8 * J_m * x}{Q * T_f} \right)^{0.5} \right] \qquad (2)$$

$t_m \rightarrow$ *delay* médio dos veículos no movimento (s).
$t_{0_m} \rightarrow$ *delay* mínimo no movimento (s).
$J_m \rightarrow$ parâmetro de qualidade do movimento.
$x \rightarrow$ grau de saturação do movimento.
$z = x - 1$.
$T_f \rightarrow$ tempo total de simulação (s).

O modelo matemático depende do *delay* mínimo de viagem do movimento t_{0_m}, que é o tempo gasto pelo veículo para trafegar o movimento na velocidade máxima permitida, sem qualquer impedimento; o parâmetro de qualidade do movimento J_m, que determina fatores como interrupções na via e dependências entre os semáforos; e o grau de saturação do movimento x, que é calculado conforme (3) [3].

$$x = (q * T_{ciclo}) / (Q * T_{verde}) \qquad (3)$$

$x \rightarrow$ grau de saturação do movimento.
$q \rightarrow$ fluxo de veículos do movimento (V/h).
$Q \rightarrow$ fluxo de saturação no movimento (V/h).
$T_{verde} \rightarrow$ tempo de verde dos semáforos (s).
$T_{ciclo} \rightarrow$ somatório dos tempos de verde e amarelo dos semáforos presentes no ciclo (s).

O valor da *fitness* atribuído a cada indivíduo é exatamente igual ao valor de função objetivo calculado por meio de (1). A partir do modelo matemático apresentado é possível, então, avaliar os indivíduos sem a necessidade do simulador de tráfego. Dessa forma, o tempo de execução do método se torna bem menor que em trabalhos anteriores.

IV. Experimentos Realizados

Para testar o método proposto foram modelados dois trechos e obtidos os fluxos de entrada de veículos q em cada movimento. O AG foi executado 21 vezes por trecho, com o intuito de analisar a estabilidade da resposta oferecida. Cada execução contou com 1000 iterações e 100 indivíduos na população. Os parâmetros utilizados pelo algoritmo, obtidos de forma empírica, e pelo modelo matemático, escolhidos com base em [2], são descritos nas Tabelas 1 e 2 respectivamente.

TABELA 1
Parâmetros de Execução do AG

Parâmetro	Valor
Execuções	21
Iterações	1000
População	100
Taxa de Cruzamento (T_{cross})	1.0
Taxa de Mutação (T_{mut})	0.6
Delta de Mutação (Δ)	5s

TABELA 2
Parâmetros do Modelo

Parâmetro	Valor
Tempo de Amarelo ($T_{amarelo}$)	3s
Intervalo de Verde (I_{verde})	10 a 100s

A. *Trecho 1*

No primeiro experimento foi modelado um trecho com 4 ruas de sentido único em torno de um quarteirão ou de uma praça, conforme apresentado na Fig. 5. Foi considerado um horizonte de análise de 1h. Seguindo o sugerido em [5], foi adotado $J_m = 4$ para todos os movimentos, devido aos mesmos possuírem semáforos com dependências cíclicas. Além disso, o fluxo de saturação das vias foi definido como $Q = 1800V/h$, conforme sugerido em [7]. Os parâmetros de entrada do modelo são apresentados na Tabela 3.

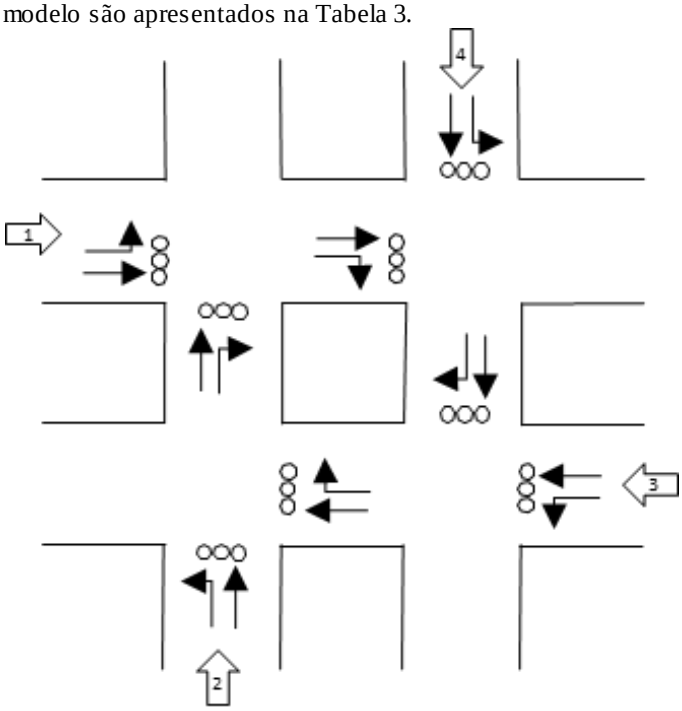

Fig. 5. Trecho 1 modelado para os testes.

TABELA 3
PARÂMETROS DO MODELO - TRECHO 1

Parâmetro	Valor
T_f	3600s
T_{0_m}	0s
J_m	4
Q	1800V/h

Após finalizado o experimento, o melhor *delay* obtido foi de 194s, como exibido na Tabela 4. Este valor foi encontrado em 20 das 21 execuções.

TABELA 4
RESULTADOS OBTIDOS – TRECHO 1

Resultado	*Delay* (s)
Melhor	194
Pior	195
Média	194.05
Mediana	194
Desvio Padrão	0.22

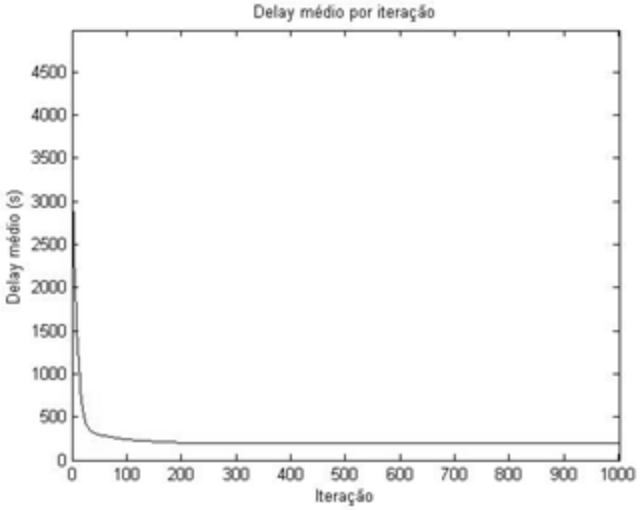

Fig. 6. Curva de convergência média para o Trecho 1.

A curva de convergência média das 21 execuções é apresentada na Fig. 6. A partir dela é possível perceber que o algoritmo consegue obter resultados bastante próximos da melhor solução encontrada já em poucas iterações. Isso sugere ser possível refinar o algoritmo de forma a acelerar o tempo total de convergência, o que pode ser feito com a inclusão de operadores baseados em busca local.

O algoritmo gastou, em média, cerca de 0,16s por iteração e 160s para o total de 1000 iterações, tendo as simulações sido realizadas em apenas um núcleo de um computador PC com processador Intel Core i5, 2,27GHz e 3GB de memória RAM, e os algoritmos implementados em Python 3.4.2 no Windows 7. Esses tempos foram consideravelmente inferiores aos observados nas abordagens baseadas em simulação ([2] e [4]). Nesses casos a avaliação de cada indivíduo demorava, em média, 0,2s, o que levava a um tempo de 20s por iteração para avaliar a população com 100 indivíduos, além do tempo gasto pelo algoritmo de otimização em si.

B. *Trecho 2*

Para o segundo experimento foi modelado um trecho com uma via principal e 3 vias secundárias, representando parte de uma grande avenida/corredor (Fig. 7). Também foi considerado um período de análise de 1h e o fluxo de saturação das vias $Q = 1800V/h$. O parâmetro de qualidade dos movimentos foi ajustado para $J_m = 1,2$, como sugerido em [5] para trechos que possuem semáforos sincronizados. Os parâmetros de entrada do modelo são exibidos na Tabela 5.

TABELA 5
PARÂMETROS DO MODELO - TRECHO 2

Parâmetro	Valor
T_f	3600s
T_{0_m}	0s
J_m	1,2
Q	1800V/h

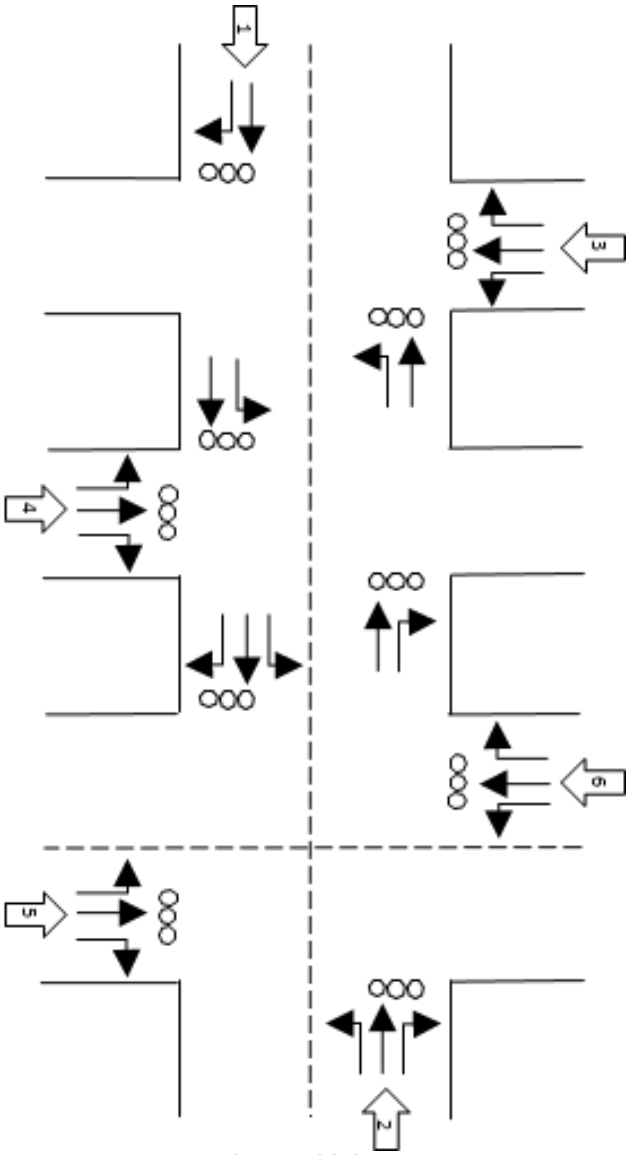

Fig. 7. Trecho 2 modelado para os testes

Neste experimento, o melhor *delay* obtido foi de 206s, como exibido na Tabela 6. Este valor foi encontrado em 9 das 21 execuções. Vale destacar que a mediana das soluções está bem próxima da melhor execução, o que indica que o algoritmo está convergindo significativamente bem na maior parte dos casos.

TABELA 6
RESULTADOS OBTIDOS – TRECHO 2

Resultado	***Delay* (s)**
Melhor	206
Pior	274
Média	221.24
Mediana	210
Desvio Padrão	23.53

A curva média de convergência das 21 execuções, mostrada na Fig. 8, demonstrou comportamento similar ao observado para o Trecho 1.

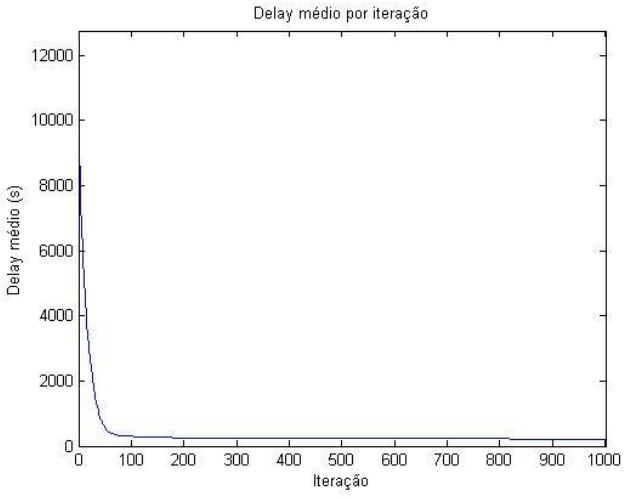

Fig. 8. Curva de convergência média do *delay* médio para o Trecho 2.

Por sua vez, o AG gastou 0,23s por iteração e 230s para 1000 iterações. Esse valor é um pouco superior ao observado no trecho 1 devido ao maior número de semáforos e movimentos. Vale destacar que o algoritmo poderia ser ainda mais rápido se a implementação fosse feita em um ambiente de programação paralela e em linguagem compilada, como C ou C++.

V. CONCLUSÃO

A otimização da programação semafórica é uma das estratégias que podem ser utilizadas para atenuar os problemas de congestionamento em redes viárias de médias e grandes cidades. Neste contexto, foram propostas diversas ferramentas com a finalidade de encontrar boas programações. No entanto, essas abordagens dependem, em geral, de simuladores de tráfego para calcular os *delays* dos veículos e avaliar as soluções candidatas. Como o simulador é lento, o processo de otimização nesses casos pode levar horas, gerando limitações, como, por exemplo, a impossibilidade de se realizar a otimização semafórica em tempo real.

Assim, este trabalho propõe uma nova abordagem para o problema de programação semafórica. Na arquitetura proposta, o simulador de tráfego é executado uma única vez, no início do procedimento, e os fluxos de veículos obtidos pelo simulador são utilizados para construção de um modelo matemático que é usado durante a otimização. Essa opção se mostrou interessante, tendo em vista que o custo computacional de avaliação do modelo é muito inferior ao do simulador. Espera-se com essa estrutura viabilizar, em curto prazo, o uso de ferramentas para otimização robusta em tempo real.

Os resultados obtidos nos testes preliminares para a nova arquitetura constataram o que era esperado em relação ao tempo de execução. Enquanto o simulador gasta em média 0,2s para avaliar cada indivíduo, o AG baseado em modelo matemático conseguiu executar iterações completas nesse mesmo tempo, o que tornou o processo cerca de 100 vezes mais rápido. Dessa forma, as limitações relacionadas a tempo podem ser removidas, o que abre novas perspectivas de tratamento do problema.

Já em relação aos *delays* obtidos, os resultados também foram satisfatórios, com boa convergência e estabilidade do algoritmo. Esses resultados sugerem que a ferramenta proposta, uma vez aprimorada, deve ser adequada para realizar a programação semafórica em cenários reais.

VI. TRABALHOS FUTUROS

São sugeridos os seguintes tópicos de estudo para continuidade deste trabalho:

1) Aprimorar o algoritmo proposto, com o estudo de novos operadores de cruzamento e mutação, avaliação mais minuciosa da sintonia do algoritmo e inclusão de mecanismos de busca local para refinamento das soluções obtidas.
2) Testar as programações semafóricas obtidas em um simulador de tráfego, como o SUMO [10], a fim de validar os *delays* encontrados e, por consequência, o modelo matemático proposto em [5].
3) Alterar o foco do otimizador de mono-objetivo para multiobjetivo, incluindo a variância da velocidade dos veículos como um critério de projeto. Essa função é importante para garantir que um movimento com maior fluxo de entrada de veículos ou com maior velocidade máxima permitida não se torne dominante no processo de otimização em detrimento aos demais.
4) Testar as programações semafóricas obtidas em relação à robustez, ou seja, testá-las em diferentes cenários de fluxo de entrada de veículos nos movimentos para averiguar se elas permanecem viáveis caso haja alterações nos fluxos [13]. O grau de robustez das programações semafóricas pode até mesmo se tornar um terceiro critério de decisão no processo multiobjetivo.

AGRADECIMENTOS

Os autores agradecem as agências de fomento CAPES, CNPq e FAPEMIG pelo apoio financeiro.

REFERÊNCIAS

[1] CET-SP. Portaria 061/2010 do Departamento de Operação do Sistema Viário (São Paulo). Maio 2010.
[2] B. C. Costa, P. E. M. Almeida, E. G. Carrano, "Programação Semafórica De Tempo Fixo Em Microrregiões Utilizando Otimização Multiobjetivo E Simulação Microscópica." Dissertação de Mestrado, CEFET-MG, Belo Horizonte, MG, Brasil, 2012.
[3] L. Vilanova, Programação de um semáforo usando o método do grau de saturação. Disponível: http://www.sinaldetransito.com.br/artigos/saturacao.pdf
[4] F. B. Oliveira, P. E. M. Almeida, "Integrating Computational Intelligence, Microsimulation and Semaphore Regulation into Conventional GIS Software." WCTR, pp. 1–18, 2010.
[5] R. Akçelik, "Travel Time Functions for Transport Planning Purposes: Davidson's Function, Its Time-Dependent Form and an Alternative Travel Time Function.", Australian Road Research 21 (3), pp. 49-59, 2000
[6] DENATRAN. *Manual de Semáforos. 2 ed*. Departamento Nacional de Trânsito, Brasília, DF, Brasil, 1984.
[7] TRB. *Highway Capacity Manual 2010*. Transportation Research Board, EUA, 2010.
[8] CONTRAN. *Manual Brasileiro de Sinalização de Trânsito*. Conselho Nacional de Trânsito, Brasília, DF, Brasil, 2012.
[9] Vivid Solutions, OpenJUMP. Disponível: http://www.openjump.org/index.html
[10] Institute of Transportation Systems, SUMO – Simulation of Urban Mobility. Disponível: http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/
[11] J. E. Baker, "Reducing Bias and Inefficiency in the Selection Algorithm." Second International Conference on Genetic Algorithms and their Application, pp. 14–21, 1987.
[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Boston, EUA, 1989.
[13] J. Mulvey, R. Vanderbei, S. Zenios, "Robust Optimization of Large-Scale Systems." Operation Research, v. 43, pp. 264-281, 1995.