Projeto de Controle de Trânsito Baseado em Aprendizagem por Reforço

Marcos Yamasaki[1], Alysson J. M. de Freitas[1], Sibilla B. da L. França[1]
[1] UFPR, Curitiba, Brasil
marcos.yamasaki@ufpr.br, alyssonmalko@ufpr.br, sibilla@eletrica.ufpr.br

Resumo*—O congestionamento do trânsito e as emissões de gases poluentes são os maiores problemas enfrentados nos dias atuais para a mobilidade urbana. O desenvolvimento de projetos de semáforos inteligentes visa minimizar o congestionamento de veículos em intersecções, diminuindo o tempo de espera do motorista e consequentemente aumentando o fluxo de carros. O sistema atua de modo inteligente, pois está sempre aprendendo a melhorar o trânsito. Este projeto consiste em desenvolver e implementar em hardware o controle de trânsito baseado em aprendizagem por reforço. Para execução deste projeto utilizou-se a ferramenta MATLAB para o desenvolvimento do algoritmo e simulações em software, além de sua extensão HDL Coder para geração do código VHDL e implementação em hardware na FPGA Virtex 5 VLX50T. O resultado apresentou baixo consumo de hardware e validou a metodologia de implementação utilizada.

I. INTRODUÇÃO

A mobilidade urbana é um dos temas com foco permanente de discussões no setor público e fora dele dado o caos que vem se transformando o trânsito de cargas e pessoas nas grandes cidades brasileiras. Os semáforos de trânsito atuais que normalmente são encontrados nas ruas funcionam de maneira limitada, pois são programados antecipadamente apenas baseados na análise histórica de tráfego do local. São utilizadas fórmulas matemáticas baseadas em fatores como o fluxo de carros por minuto para descobrir o ciclo ideal de cada cruzamento. Cada ciclo corresponde ao tempo que leva para completar os sinais de verde, amarelo e vermelho. O desenvolvimento de projetos de semáforos inteligentes é relevante, pois tem como objetivo minimizar o congestionamento de veículos em interseções, diminuindo o tempo de espera do motorista e por consequência, aumentando o fluxo de carros, desafogando a cidade de maneira inteligente, visto que o sistema está sempre aprendendo a melhorar o trânsito. Entre as técnicas para minimizar o tempo de espera no trânsito se destacam as que baseiam nos algoritmos de inteligência artificial. Entre as soluções com algoritmos de inteligência artificial, o método de aprendizado por reforço *Q-Learning* apresenta diversas referências com uma maior eficiência em comparação a outras soluções [1]. O método de aprendizagem por reforço *Q-Learning* aplicado à um sistema de controle de trânsito apresenta alta relevância econômica, social e ambiental, pois a diminuição do tempo de espera no trânsito resulta na redução de gastos públicos e nas emissões de gases poluentes dos veículos.

II. APRENDIZADO POR REFORÇO

O método de *Reinforcement Learning* (RL), aprendizado pelo reforço, é uma das teorias de aprendizado para máquinas focadas em interações com o meio. O RL pode ser descrito como um processo de aprendizado por interação com o ambiente, mapeando situações e ações, e com objetivo de maximizar um sinal numérico de recompensa. O RL é definido não pela caracterização dos seus métodos de aprendizado, mas por caracterizar um problema de aprendizado [1].

Formalmente um agente de RL encontra um problema de decisão de Markov (PDM), que possui quatro componentes básicos: estados, ações, distribuições de transição e de recompensa. Como este método é dinâmico terá um aumento exponencial de ações durante o tempo de execução [2]. O RL pode ser adaptado em quatro elementos: política, função de recompensa, função de valor e modelo de ambiente [1].

A política define a maneira do agente de aprendizagem se comportar em um determinado ambiente. A política é um mapeamento de estados encontrados no ambiente para ações a serem tomadas nesses estados. Em alguns casos, a política pode ser uma função ou pesquisa de tabela simples, enquanto que em outros, pode envolver extensa computação, como um processo de busca. A política é o núcleo de um agente de aprendizagem, no sentido de que só ele é suficiente para determinar o comportamento.

Uma função de recompensa define a meta em um problema de RL. Ou seja, mapeia cada estado observado do ambiente para um único número, uma recompensa, indicando o desejo intrínseco daquele estado. O único objetivo de um agente de aprendizagem é maximizar a recompensa total, que ele recebe em longo prazo. A função de recompensa define quais são os bons e maus eventos para o agente. A função de recompensa não deve ser alterada pelo agente. No entanto, pode servir como uma

base para a alteração da política. Por exemplo, se uma ação selecionada pela política resulta em uma baixa recompensa, no futuro a política pode ser alterada para selecionar outra ação para melhorar a recompensa obtida.

Considerando que uma função de recompensa indica o que é bom imediatamente, uma função valor especifica indica o que é bom no longo prazo. De maneira geral, o valor de um estado é a quantidade total de recompensa que um agente pode acumular no futuro. As recompensas determinam o desejo imediato, intrínseco de estados do ambiente, já valores indicam o desejo de longo prazo dos estados, tendo em conta os estados que são propensos a seguir, e as recompensas disponíveis nesses estados. Por exemplo, um estado pode sempre produzir uma baixa recompensa, mas ainda tem um alto valor, pois é regularmente seguido por outros estados que produzem altas recompensas. Ou o inverso pode ser verdadeiro. Recompensas estão em primeiro lugar, enquanto que os valores, como previsões de recompensas, estão em segundo. Sem recompensas não poderia haver valores, e o único objetivo de estimar valores é conseguir mais recompensas. As opções de ações são feitas com base em juízos de valor. Geralmente, buscam-se ações que provocam estados de maior valor, não mais alta recompensa, porque essas ações vão obter a maior quantidade de recompensa no longo prazo. Na tomada de decisão e planejamento a quantidade de valor é aquela com a qual deve-se ficar mais preocupado. Recompensas são basicamente, dadas diretamente pelo ambiente, mas os valores devem ser estimados e reestimados a partir das sequências de observações que um agente faz ao longo de toda sua vida útil. É possível dizer que o componente mais importante dos algoritmos de aprendizagem de reforço é um método eficiente para estimar valores.

O quarto e último elemento é o modelo de ambiente. Pode ser entendido como algo que imita o comportamento do ambiente. Por exemplo, dado um estado e ação, o modelo pode prever o próximo estado resultante e próxima recompensa.

A. Método de aprendizagem Q-Learning

Um dos avanços importantes no aprendizado por reforço foi o desenvolvimento de um algoritmo de controle conhecido como *Q-Learning* [2]. A sua forma mais simples, um episódio de *Q-Learning*, é definido pela equação (1), em que a cada episódio o algoritmo incrementa valores em uma tabela de recompensas que servirá para a tomada de decisão do agente de aprendizado. Em outras palavras, toda vez que o agente de aprendizado toma uma decisão escolhendo uma ação a ser tomada, é levado em consideração o valor da recompensa e o novo estado que vai depender da ação escolhida atualmente e da anterior, assim o valor de Q é atualizado:

$$Q(s,a)_i \leftarrow Q(1-\alpha).Q(s,a)_{i-1} + \alpha[R(s,a)_i + \gamma \underbrace{max}_{a} Q(s',a')] \quad (1)$$

Onde, s = Estado atual; a = ação tomada no estado atual; s' = próximo estado; a' = ação tomada no próximo estado; α = taxa de aprendizado; γ = fator de desconto.

III. CONTROLE DE TRÂNSITO BASEADO EM APRENDIZADO POR REFORÇO

Um método simples e já bem difundido utilizando o *Q-Learning* é o QLC (*Q-Learning Control*). O QLC pode também ser definido como método tabular de *Q-Learning* e existe um número limitado de conjunto de ações neste controle. Neste método um conjunto de ações é uma combinação de tempos verdes no semáforo em cada fase. O tempo cíclico é variável e baseado na demanda do tráfego. Estados são formados a partir da média do tamanho das filas. [3].

O processo de interação entre o QLC e um simulador de tráfego pode ser vista na Fig.1. Os tamanhos das filas que formam o ambiente são enviados ao QLC e é proposto um tempo verde para cada semáforo. Os tempos propostos de semáforos verdes são selecionados pela lista de ações pré-definidas do método *Q-Learning*.

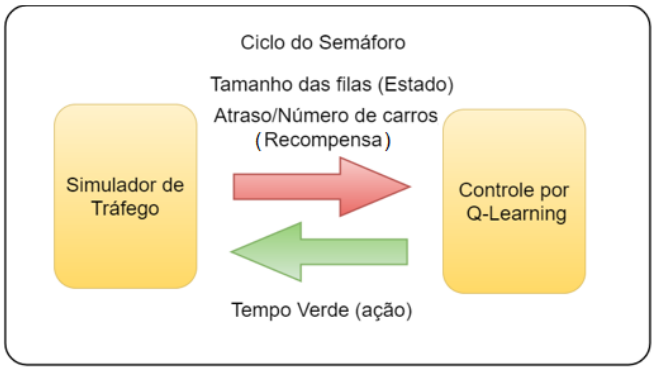

Fig. 1.- *O processo de interação entre o QLC e um simulador de tráfego [3].*

A recompensa pode ser definida como inversamente proporcional ao tempo de atraso médio ao final de cada ciclo para todas as vias que se interceptam. Isso significa que existe um valor maior para casos com um menor tempo médio de atraso.

Para obter os resultados de comparação com o controle de tempo fixo foi necessário antes especificar e melhorar a função de recompensa do algoritmo.

A função de recompensa foi dividida em duas partes, a primeira analisa a razão de veículos que passaram por cada via, e fornece um valor numérico de recompensa dependendo da razão de carros que saiu em cada via. A segunda parte analisa a melhor ação possível a ser tomada naquele estado, dando prioridade para a fila com maior tamanho.

Para complementar esta função foi adicionado um complemento para que o sistema receba recompensa

quando tomar a melhor decisão possível daquele estado, dando prioridade para a fila de maior tamanho.

Com a função de recompensa definida foram realizados os testes de comparação com tempo fixo. O ambiente para comparação foi montado utilizando o software SUMO [4] com os seguintes parâmetros:

- Tamanho médio dos veículos: 5m;
- Aceleração média dos veículos: 1 metro por segundo;
- Tamanho da via: 100 metros;
- Número máximo de veículos em cada via: 15;
- Tempo de simulação (simulador): 8 horas;
- Tempo de simulação (real): 22 minutos.

A distribuição de entrada de veículos foi feita da seguinte forma: foram definidas probabilidades de entrada de veículo por segundo, ou seja, se esta probabilidade estiver definida em 20%, por exemplo, significa que há 20% de chance de entrar um veículo a cada segundo nas vias, conforme a Tabela 1.

TABELA 1. PROBABILIDADES DE ENTRADA DE VEÍCULOS.

Horas	Distribuição Via1	Distribuição Via 2
1	20%	10%
2	10%	20%
3	5%	20%
4	5%	5%
5	18%	20%
6	6%	18%
7	2%	5%
8	16%	10%

Foram aplicados dois métodos de tempo fixo, um fixado em 20 segundos de tempo semafórico verde para cada via, fechando um ciclo de 40 segundos, e outro com tempos com variação determinada de acordo com a definição da modelagem do ambiente, que se baseia na probabilidade da demanda de veículos de cada via. Já no método *Q-Learning* foi utilizada uma tabela Q, treinada com uma política baseada no algoritmo de prioridade para a fila mais longa ao longo de 20 horas de tempo real, equivalente a 12 dias dentro do ambiente de simulação com probabilidades distintas de entradas de veículos. O gráfico de aprendizado pode ser visto na Fig. 2, nota-se que o ponto de convergência ocorre aproximadamente em 1000s, ou 25 ciclos do semáforo. No entanto, isso não representa que a tabela Q foi completada com todas as ações, mas que o valor da recompensa já está saturado.

Para esta comparação o algoritmo foi ajustado para aproveitar apenas os valores da tabela Q, sem realizar nenhuma exploração. Para realizar uma comparação com tempo fixo foi necessário utilizar o mesmo ambiente, e não levar em consideração o tempo de aprendizado e ações tomadas durante este tempo.

Os resultados da comparação podem ser vistos na Fig.3. Houve diminuição do tempo de espera em 7335% (6,6 minutos) no tempo fixo e 3178% (2,8 minutos) no tempo variado. Isso significa que um motorista teria um tempo de espera médio de 6,7 minutos no tempo fixo e 2,9 minutos no tempo variado, resultando em oito e quatro ciclos de semáforo respectivamente, enquanto no semáforo controlado pelo método QLC o tempo médio de espera seria de 5,5 segundos, então o motorista esperaria em média um ciclo de semáforo para sair da intersecção.

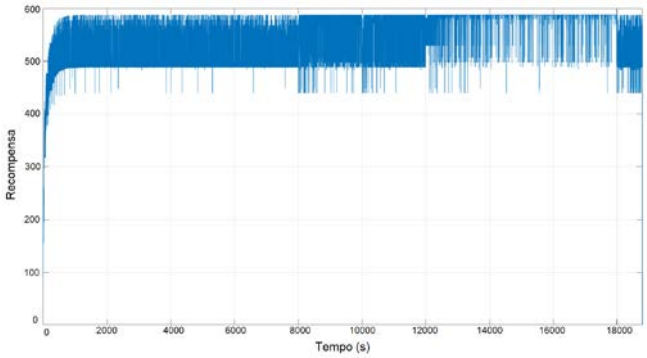

Fig. 2. *Gráfico de aprendizado durante o treinamento.*

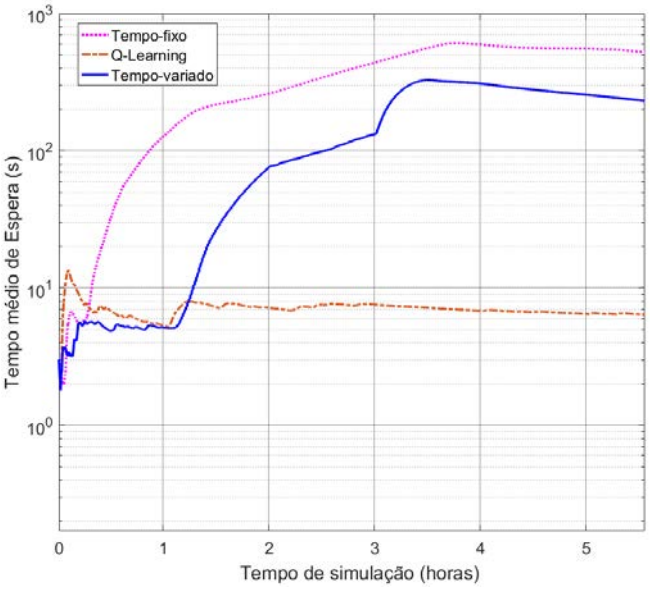

Fig. 3. *Comparação entre tempo-fixo, tempo-variado e q-learning*

Isto representa uma melhora muito significativa, no entanto, algumas observações devem ser levadas em consideração. O ambiente foi modelado de forma a ser dinâmico, caso ele tivesse entradas constantes de veículos iguais nas duas vias, o tempo médio de espera dos dois métodos provavelmente estariam mais próximos. Para uma comparação com a realidade seria necessário modelar um ambiente existente analisando o fluxo médio de veículos durante um período de tempo e também o tempo fixo de controle semafórico utilizado neste ambiente. Como este não era o objetivo desta pesquisa, esta atividade não foi realizada, mas é uma possibilidade de extensão do projeto.

Para simulação do algoritmo em hardware realizou-se uma cossimulação dos blocos do sistema. Desta maneira, o comportamento da FPGA é emulado, permitindo assim analisar os resultados obtidos em hardware dentro do ambiente de simulação. Para isso, foram criados blocos

simplificados, gerados códigos em VHDL e foi realizado o *testbench*.

Para esta etapa, exigiu-se a parametrização para a conversão de dados do tipo flutuante para ponto fixo e a configuração de clock do hardware. A saída do sistema com o simulador rodando junto com o sistema simplificado em cossimulação é mostrada na Fig. 4. Os blocos são divididos por ciclos de processamento, onde o bloco central realiza o controle do algoritmo em geral e o bloco à direita representa uma iteração básica do *Q-Learning*, no qual foi implementado em hardware.

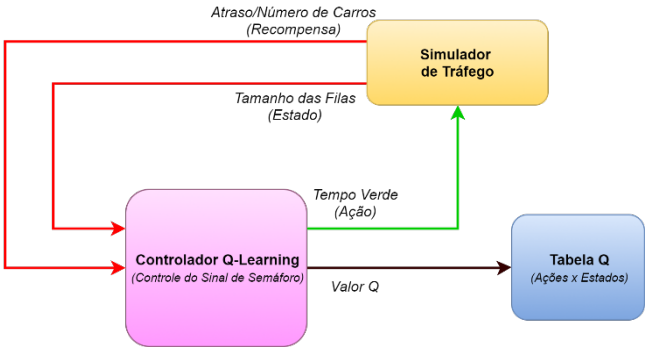

Fig. 4. *Sistema simplificado de cossimulação no Simulink.*

O resultado obtido com a implementação do sistema foi de uma baixa ocupação de hardware, conforme a Tabela 2, o que pode ser explicado pela robustez do modelo de FPGA escolhido e a baixa complexidade do método *Q-Learning* de apenas um agente de aprendizado. A aplicação funcionou em tempo real com o simulador trânsito.

TABELA 2. SUMÁRIO DE OCUPAÇÃO DE HARDWARE

Sumário de Ocupação de Hardware			
Ocupação de *Slice Logic*	**Utilizado**	**Disponível**	**Ocupação**
Nº de *Slice Registers*	112	28,8	1%
Nº utilizado como *Flip Flops*	112		
Nº of Slice LUTs	48	28,8	1%
Nº utilizado como lógica	48	28,8	1%
Nº utilizando apenas *O6 output*	48		
Nº de *route-thrus*	30		
Nº utilizando apenas *O5 output*	30		
Nº de *Slices* ocupados	32	7,2	1%
Nº de pares utilizados de *LUT Flip Flop*	128		
Nº de *Flip Flop* não-utilizados	16	128	12%
Nº de LUT não-ultilizados	80	128	62%
Nº de pares *LUT-FF* completamente utilizados	32	128	25%
Nº de *unique control sets*	1		
Nº de *slice register sites lost to control set restrictions*	0	28,8	0%
Nº de *DSP48Es*	2	48	4%
Média de *Fanout of Non-Clock Nets*	1.22		

IV. CONCLUSÃO

Este projeto mostrou uma maneira simples de implementação do algoritmo de *Q-Learning* no controle do fluxo de veículos, que mesmo com a falta de um ambiente com mais complexidade e informações, foi possível verificar uma grande melhora se comparado aos métodos convencionais de tempo fixo aplicados. Em um ambiente dinâmico, como o ambiente proposto e simulado, é possível afirmar que o método implementado apresenta reduções significativas no tempo de espera médio dos veículos, e de no máximo um ciclo semafórico de espera na média para cada veículo da via. Foi mostrado ainda uma metodologia que abstrai a maioria dos conhecimentos necessários para realizar uma implementação em hardware diretamente do software de desenvolvimento MATLAB®.

A principal vantagem na implementação em hardware é a possibilidade de utilizar múltiplos agentes de aprendizado, operando de modo concorrentemente. Por consequência, o processo de aprendizagem torna-se muito mais rápido, resolvendo assim um dos grandes problemas dos métodos de aprendizado por reforço, o tempo de convergência do sistema para tomada de decisão com recompensas ótimas.

REFERÊNCIAS

[1] BARTO, A., G., e MAHADEVAN, S. Recent advances in hierarchical reinforcement learning. Discrete Event Dynamic Systems, v.13, n.4, p.341–379, 2003.

[2] WATKINS, C. e DAYAN, P. (1992). Q-Learning. Machine Learning, v.3/4, n.8, p.279-292.

[3] ARAGHI, S.; KHOSRAVI, A.; CREIGHTON, D. A review on computational intelligence methods for controlling traffic signal timing. Expert Systems with Applications, v. 42, n. 3, p. 1538-1550, 2015.

[4] SUMO, Simulation of Urban MObility (2017). Disponível em: <http://sumo.dlr.de/wiki/Simulation_of_Urban_MObility_-_Wiki> Acesso em: data (02 nov. 2017).

[5] ABDULHAI, B., PRINGLE, R., e KARAKOULAS, G. J. Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering, v.129, n.3, p.278–285, 2003.

[6] EL-TANTAWY, S., ABDULHAI, B., & ABDELGAWAD, H. Multiagent Reinforcement Learning for integrated network of adaptive traffic signal controllers (MARLINATSC): Methodology and large-scale application on downtown Toronto. IEEE Transactions on Intelligent Transportation Systems, p. 14, 1140–1150. (2013).

[7] SILVA LUCILEIDE M. D., TORQUATO MATHEUS F., FERNANDES MARCELO A. C., Proposta de arquitetura em hardware para FPGA da técnica Q-Learning de Aprendizagem por Reforço, XIII Encontro Nacional de Inteligência Artificial e Computacional, SBC ENIAC (2016).

[8] SUTTON, R. S. e BARTO, A. G. Introduction to Reinforcement Learning. MIT Press, Cambridge, MA, 1998.

[9] THORPE, T. L. e ANDERSON, C. W. Traffic light control using sarsa with three state representations. Technical report, IBM Corporation, Department of Computer Science Colorado State University Fort Collins, Citeseer, 1996. Disponível em <https://pdfs.semanticscholar.org/c3cb/5362c9b4e0c8ab66feaf6dfbc295cd073d2a.pdf>. Acesso em 20 nov. 2017.