RELATÓRIO DO PROJETO INTEGRADO A

Sistema de acesso rápido ao RU via código de barras, sistema de créditos e plataforma *WEB* de gerenciamento de dados

|  |  |
| --- | --- |
| **Disciplina:** | **Projeto Integrado A** |
| **Aluno(s):** | **ERIK N. O. DOS SANTOS**  **PEDRO M. ANTUNES** |

CURITIBA  
2016

**Sumário**

[1. Introdução 4](#_Toc454731851)

[1.1. Objetivos 5](#_Toc454731852)

[1.1.1. Objetivo Geral 5](#_Toc454731853)

[1.1.2. Objetivos Específicos 5](#_Toc454731854)

[2. Revisão bibliográfica 6](#_Toc454731855)

[2.1. controle de tráfego existente 6](#_Toc454731856)

[2.2. O aprendizado de máquina 10](#_Toc454731857)

[2.3. O método de Reinforcement Learning: 11](#_Toc454731858)

[2.4. Desafios do método: 12](#_Toc454731859)

[2.5. Elementos básicos do Reinforcement Learning: 13](#_Toc454731860)

[2.5.1. A política: 14](#_Toc454731861)

[2.5.2. A função de recompensa: 14](#_Toc454731862)

[2.5.3. A função de valor: 14](#_Toc454731863)

[2.5.4. O modelo de ambiente: 15](#_Toc454731864)

[2.6. Aprendizagem de diferença temporal 15](#_Toc454731865)

[2.7. Método de aprendizagem *Q-Learning* 15](#_Toc454731866)

[2.8. Aplicações do Q-Learning para controle de semáforos 17](#_Toc454731867)

[2.9. Projeto de um QLC 20](#_Toc454731868)

[3. Desenvolvimento utilizando Q-Learning 22](#_Toc454731869)

[3.1. Exemplo de utilização do q-learnig 22](#_Toc454731870)

[3.2. Adaptação do método Q-learning para controle de trânsito 28](#_Toc454731871)

[3.3. definição do ambiente de simulação 28](#_Toc454731872)

[3.3.1. o espaço do ambiente de simulação 28](#_Toc454731873)

[3.3.2. a física do ambiente de simulação 29](#_Toc454731874)

[3.4. Aplicação do algoritmo 31](#_Toc454731875)

[4. Resultados 33](#_Toc454731876)

[5. Conclusão 36](#_Toc454731877)

[6. Referências 38](#_Toc454731878)

[7. Anexos 41](#_Toc454731879)

[7.1. Anexo A 41](#_Toc454731880)

[7.2. Anexo B 43](#_Toc454731881)

# Introdução

O Restaurante Universitário da UFPR, mas conhecido entre a comunidade como “RU”, é um importante serviço para discentes, docentes e técnicos administrativos da univerdade, pois proporciona refeições de qualidade, nutricionalmente adequadas e por um custo acessível a todos que o utilizam.

O Restaurante Universitário teve sua origem em 05/08/1966, administrado por membros do Diretório Central dos Estudantes (DCE). A partir de 1980 o Restaurante passou a ter administração da UFPR, quando recebeu, então, a denominação “Restaurante Universitário” (PRÓ-REITORIA DE ADMINISTRAÇÃO DA UFPR, 2016).

Após longo período utilizando sistema convencional de distribuição/servimento “porcionado” – onde os usuários são servidos de porções pré-estabelecidas de alimentos – em 1995, o RU Central adotou o sistema “self-service”, sendo porcionadas apenas o prato protéico (carnes) e a sobremesa, o que proporcionou maior conforto ao usuário em determinar as quantidades desejadas (PRÓ-REITORIA DE ADMINISTRAÇÃO DA UFPR, 2016).

Recentemente, o RU completou 50 anos de existência. De lá para cá, muitas mudanças ocorreram tanto na universidade, quanto no corpo discente/docente que o frequentam. Como é de se esperar, a comunidade acadêmica da UFPR aumentou e a tendência é de que continue a crescer.

Atualmente, o RU conta com 4 unidades em Curitiba e outras 4 nos campis do interior. Na conjuntura atual, é impossível não deparar-se com grandes filas e verdadeiros “congestionamentos” nos horários de pico, seja na sede do RU Politécnico, Central, Agrárias ou Jardim Botânico. Outro problema grave e decorrente são as fraudes. Alunos que obtem por mais de uma vez a mesma refeição, ou até mesmo pessoas de fora da comunidade que acessam o Restaurante Universitário como membros da UFPR.

O sistema atual de acesso ao RU funciona da seguinte maneira:

* Primeiramente, na entrada do restaurante, deve-se apresentar um documento que comprove seu vínculo com a universidade e então recebe-se uma ficha indicando qual é esse vínculo. Ex: “ALUNO”, “SERVIDOR”.
* Com a ficha em mãos, você deve dirigir-se ao caixa (cuja quantidade varia de restaurante para restaurante) e então efetuar o pagamento.
* O indivíduo então recebe um ticket fiscal que comprova o pagamento da taxa adequada para a respectiva refeição (café da manhã, almoço ou jantar).
* O ticket deve ser entregue a um terceiro funcionário, e só então a pessoa tem acesso ao buffet.

Evidentemente, tal processo gera um certo retardo no acesso ao RU, além de ser pouco eficiente em termos de controle e agilidade. Dificuldades com o troco são igualmente comuns e o manuseio de cédulas de dinheiro e moedas antes da refeição pode ser desagradável.

Sendo assim, o desenvolvimento de um sistema de acesso simples e ágil, mas que garanta a segurança e a comodidade dos que o utilizam torna-se pertinente. Além de gerar maior organização no acesso e diminuição no tamanho das filas, por consequência, estudantes, professores e técnicos ganharão mais tempo e poderão despendê-lo de forma mais proveitosa em atividades acadêmicas, ao invés de permanecer numa longa fila de acesso aos restaurantes universitários.

### Objetivo Geral

Desenvolver um projeto de controle de acesso aos Restaurantes Universitários, através de leitores de código de barras, sistema de créditos e uma plataforma *WEB* de gerenciamento.

### Objetivos Específicos

Dentre os principais objetivos específicos destacam-se:

* desenvolver a implementação e simulação em software MATLAB do algoritmo de controle inteligente;
* analisar e concluir a viabilidade do algoritmo em aplicação real;
* refinar o algoritmo com parâmetros de aprendizagem buscando melhorias na eficiência;
* introduzir um sistema de agentes múltiplos de aprendizado.

# Revisão bibliográfica

## controle de tráfego existente

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. Outros fatores considerados são: largura da rua, número de carros estacionados e presença de lombadas ou valas (PREFEITURA DE MOGI DAS CRUZES, 2015).

Os sistemas são reprogramados conforme a necessidade, sendo que alguns locais podem variar de três a cinco anos para a manutenção. Os ajustes são feitos nos tempos de sinal verde e sinal vermelho, dependendo do tráfego do local. Em corredores, vias que percorrem grandes distâncias com diversos cruzamentos, busca-se a “onda verde”, que idealmente o veículo percorre todo o percurso sem parar em nenhum sinaleiro (PREFEITURA DE MOGI DAS CRUZES, 2015).

No Brasil, há sistemas de trânsito sendo desenvolvidos para otimizar o fluxo de veículos nas cidades, como aplicado Mogi das Cruzes no estado de São Paulo, que em certas interseções do município há monitoramento do tráfego local por câmeras instaladas nos semáforos, detectando o volume de carros. Com as informações obtidas pelas câmeras, o semáforo inteligente adapta os tempos de sinal verde e vermelho, reduzindo o tempo de espera do motorista (PREFEITURA DE MOGI DAS CRUZES, 2015).

Na cidade de Curitiba, foi inaugurado em 2012 o CCO (Centro de Controle Operacional) que faz parte da empresa URBS (Urbanização de Curitiba) que controla o sistema de transporte público na cidade, tem como objetivo monitorar o trânsito intervindo quando necessário dependendo das circunstâncias. De acordo com o site da URBS, o sistema utiliza CFTVs (Circuitos Fechados de TV), PMVs (Painéis de Mensagens Variáveis), conectividade por fibra ótica e outros equipamentos e softwares específicos que estão instalados em diversos pontos da cidade.

No entanto a intervenção nos semáforos é feita por pessoas, funcionários da empresa, que tem capacidade limitada em relação ao aprendizado de máquina (Inteligência Artificial).

O sistema mais avançado de semáforos de trânsito em grande escala se localiza no estado de Utah nos Estados Unidos segundo Keith Barry, do site Citylab (2014). Os engenheiros de tráfego utilizam dados históricos de tráfego do estado para criar planos de sinalização para otimizar o “tempo de sinal verde” melhorando a fluidez do trânsito, com sistemas sofisticados usando diferentes estratégias para diferentes momentos, dias ou sensores para detectar veículos em espera no sinaleiro. Enquanto em sistemas atuais de muitos lugares, os ajustes são revisados entre três a cinco anos, no *Utah Department of Transportation* (UDOT) os ajustes podem ser feitos em qualquer semáforo em todo o estado dentro de trinta segundos.

A chance de encontrar um sinal vermelho em Utah, segundo Lee Davidson do jornal The Salt Lake Tribune (2013), é de uma em quatro, precisamente, 28 por cento, ou seja, 72 por cento de chance de não parar em um sinaleiro (BARRY, 2014).

A comunicação dos semáforos é feita por uma rede de fibra ótica instaladas por empresas parceiras. Os sensores utilizados que foram informados, são câmeras de circuito fechado (BARRY, 2014).

Toda esta infraestrutura tem um custo alto, no entanto, o investimento tem uma razão de retorno de quarenta por um – não é surpreendente considerando que os gastos com congestionamentos custem mais de 120 bilhões de dólares por ano para os EUA, segundo Keith Barry do portal Citylab (2014).

Basicamente, o sistema utilizado em Utah, coleta dados do trânsito ao longo do tempo e otimiza dinamicamente o tempo de sinal verde e vermelho. Os ajustes no tempo são feitos em tempo real, o próprio sistema se adapta para melhorar a fluidez do tráfego de veículos.

É possível verificar no site do UDOT *Traffic* (*Utah Department of Transportation*) as condições do tráfego no estado, navegando no mapa. É possível visualizar as imagens das câmeras instaladas nas vias, verificar acidentes, sinalizações, construções e condições do tempo. Também está aberto a comunidade, os dados de trânsito no estado, podendo escolher no mapa a interseção desejada e visualizar uma série de informações obtidas pelo sistema inteligente de semáforos de Utah, como volume de veículos, tempo de espera dos veículos, tempo de espera dos pedestres, velocidade, entre outros, ilustrado nas Figuras 1 e 2.

|  |
| --- |
| Figura 1 – Site de UDOT *Traffic* (*Utah Department of Transportation)* |
| figura1 |
| Fonte: <http://udottraffic.utah.gov/> (Acesso em 02/04/2016). |
| Figura 2 – Site de UDOT *Traffic* (*Utah Department of Transportation) Signal Performance Metrics* |
| figura2 |
| Fonte: <http://udottraffic.utah.gov/signalperformancemetrics/> (Acesso em 02/04/2016). |

Inclusive, é possível gerar os gráficos dos dados de tráfego da interseção desejada, podendo escolher o horário e o dia. As informações são apresentadas em tempo real, como mostra Figura 3.

|  |
| --- |
| Figura 3 – Gráfico gerado na interseção *3500 South (SR-171) I-215 West SB Ramps/2400 W* verificando o tempo de atraso |
| figura3 |
| Fonte: <http://udottraffic.utah.gov/signalperformancemetrics/> (Acesso em 02/04/2016). |

Uma alternativa mais barata, é a solução de Stephen Smith, professor do *Robotics Institute da Carnegie Mellon University* emPittsburgh. A solução utiliza-se de uma rede de semáforos de trânsito inteligentes. (SURTRAC; FINAL REPORT, 2014).

Ao invés de utilizar pessoas para monitorar e atuar no trânsito, os novos semáforos usam radares e câmeras para detectar o tráfego, e algoritmos sofisticados para aplicar instantaneamente ajustes no sistema baseado nas condições do ambiente em tempo real. Segundo Smith (2014), a cada interseção é construído um plano para otimizar o tráfego local, e quando isso é feito, ele se comunica com os outros semáforos seguintes. O sistema conta com 49 *smart* *signals* (PITTSBURGH POST-GAZETTE, 2014).

A redução do tempo ao longo de corredores foi de 26 por cento, o tempo de espera caiu para 41 por cento e as emissões dos veículos caíram 21 por cento (SURTRAC; FINAL REPORT, 2014).

Já em Cambridge, Carolina Osorio do MIT (*Massachusetts Institute of Technology*), professora assistente de engenharia ambiental e civil, criou um elaborado modelo de computador de tráfego urbano que prediz como os motoristas irão se comportar, incluindo como eles irão reagir com as mudanças dos padrões do semáforo (BARRY; CITYLAB, 2014).

O algoritmo de Osorio pode programar automaticamente os sinais do semáforo, em particulares situações que a cidade prioriza, como por exemplo, aumento de fluxo de pedestres.

No Brasil, ainda não há sistemas de otimização de tráfego em grande escala sendo implantados. As possíveis soluções ainda estão em fase experimental, como em São Paulo e Mogi das Cruzes. Em Curitiba, não há oficialmente semáforos inteligentes que se adaptam automaticamente dependendo das condições de trânsito, o que motiva uma implantação de um sistema inteligente experimental.

## O aprendizado de máquina

O aprendizado é “qualquer processo no qual um sistema melhora seu desempenho através da experiência” (SIMON, 1954). Na natureza do aprendizado, pode-se inicialmente definir que o ser humano aprende interagindo com o seu ambiente. Quando uma criança brinca, mexe seus braços, ou pensa sobre o que está fazendo, não existe alguém que a esteja ensinando isso de maneira explicita, mas existe uma conexão sensorial e motora dela com o seu ambiente. Treinar este tipo de conexão gera um perfeito conceito sobre causa e efeito, as consequências de suas atitudes, e o que fazer, na ordem correta, para atingir algum objetivo. Quando alguém tenta aprender a dirigir ou aprende como estabelecer uma conversa com outra pessoa, fica claro como seu ambiente responde ao que é feito, e o ser humano sempre busca influenciar o que acontece através do seu comportamento. Aprender através de interações é uma ideia fundamental que permeia todos as teorias de aprendizado e inteligência (SUTTON, 1998).

O aprendizado de máquina é uma das áreas da inteligência artificial mais relevantes e que já conta com diversos algoritmos computacionais elaborados ao longo dos últimos anos. O objetivo do aprendizado de máquina pode ser descrito como programar computadores para aprender um determinado padrão de agir, um comportamento, dado exemplos ou observações (MITCHELL, 1997).

Uma das maneiras que um animal adquire comportamentos mais complexos é pelo aprendizado buscando obter recompensas e evitando punições. A teoria pelo aprendizado por reforço (*Reinforcement Learning*) é um modelo computacional deste tipo de aprendizado (WATKINS, 1989).

## O método de Reinforcement Learning:

O método de *Reinforcement Learning (RL)*, aprendizado pelo reforço, é uma das teorias de aprendizado para máquinas mais focadas em interações com o meio do que outras abordagens (BARTO,1998).

O RL pode ser descrito como aprender o que fazer, como mapear situações para ações, de modo a maximizar um sinal numérico de recompensa. Não é dito para aquele que estará aprendendo quais ações tomar, como na maioria das formas de aprendizados de máquina, ao invés disto ele deve descobrir quais ações geram maior recompensa ao tenta-las. Nos mais desafiadores casos, ações podem afetar não só a recompensa imediata, mas podem interferir na próxima situação e assim todas as próximas recompensas (BARTO,1998). Essas duas características, tentativa-e-erro e recompensa atrasada são duas das mais importantes características do método de RL, e mostram como isto pode melhorar o desempenho do problema proposto, controle de tráfego em tempo real.

*O RL* é definido não pela caracterização dos seus métodos de aprendizado, mas por caracterizar um problema de aprendizado. (BARTO,1998).

Qualquer método que se encaixa bem para resolver um problema, pode ser considerado um método de *RL*, a ideia básica é simplesmente capturar os aspectos mais importantes do problema real diante de um agente que está aprendendo a interagir com o seu ambiente para alcançar um objetivo (BARTO,1998).

O agente que está aprendendo deve ser capaz de saber o estado do ambiente em que se encontra e conseguir, através de ações, alterar o seu ambiente. O agente também deve ter um objetivo ou objetivos que tenham relação com o estado do ambiente. A formulação tende a incluir estes três aspectos: sensação, ação e objetivo, em suas mais simples formas (BARTO,1998).

O *RL* é diferente de um método supervisionado (*supervised learning)*, onde as informações dos estados do ambiente são fornecidas ao agente de aprendizado, como reconhecimento de padrões estatísticos e redes neurais artificiais (BARTO,1998). O aprendizado supervisionado pode ser entendido como aprender por exemplos providos de um supervisor externo. Este é um importante tipo de aprendizado, mas sozinho não é adequado para o aprendizado pela interação com o ambiente, e não se aplicaria adequadamente a este projeto. Em um problema de interação muitas vezes não se consegue adquirir exemplos do comportamento desejado, que mostre quais as situações que o agente que está aprendendo deve interagir. Em um território desconhecido o agente que está aprendendo deve ser capaz de aprender pela sua própria experiência, e o método de RF oferece as ferramentas necessárias para este objetivo (SUTTON, 1998).

## Desafios do método:

Um dos maiores desafios existentes no *RL*, e não em outros métodos de aprendizado é o equilíbrio entre exploração e aproveitamento. Para obter muita recompensa, o agente que está aprendendo deve ter preferência por ações que ele experimentou no passado e que foram efetivas produzindo recompensas. Mas para descobrir essas ações ele precisa experimentar ações que ainda não testou antes. O agente deve aproveitar o que ele já conhece para conseguir recompensas, mas também deve explorar para fazer melhores ações no futuro. O dilema é este, nem exploração ou aproveitamento podem ser usadas exclusivamente, sem que haja uma falha na tarefa (BARTO,1998).

O agente deve tentar uma variedade de ações e progressivamente favorecer aquelas que parecem a melhor opção. Na área da estocástica, cada ação deve ser tentada muitas vezes para obter uma estimativa confiável e a recompensa esperada. É interessante destacar que esse problema de balancear exploração e aproveitamento não aparece no método de aprendizado supervisionado (BARTO,1998).

O RL considera explicitamente todo o problema que um agente vai enfrentar para alcançar um certo objetivo interagindo com um ambiente desconhecido. Isto contrasta com a muitos outros métodos, que consideram seus subproblemas sem se importar como isso se encaixa em uma escala maior. O método de RL começa com um ambiente completo, interativo e um agente que sempre busca algum objetivo. Todo o agente de aprendizado tem um objetivo claro, podem sentir aspectos do seu ambiente, e escolher ações que influenciam o seu ambiente. Além disso, é normalmente assumido desde o princípio que o agente vai operar, apesar da incerteza significativa do ambiente que enfrenta (BARTO,1998). Quando o RF envolve planejamento, ele tem que lidar com a interação entre planejamento e seleção de ação em tempo real, bem como a questão de como modelos ambientais são adquiridos e melhorados, definido bem o problema de tráfego de automóveis, que está em constante mudança e necessita de ações de correção em tempo real.

## Elementos básicos do Reinforcement Learning:

Formalmente um agente de RL encontra um problema de decisão de Markov (PDM), que possui quatro componentes básicos: estados, ações, e distribuições de transição e de recompensa. Como este método é dinâmico, ou seja, terá um aumento exponencial de ações durante o tempo (BELMANN, 1957), ele pode ser adaptado em quatro elementos: *a policy* (a política), *a reward function* (função de recompensa), *a value function* (função de valor), e opcionalmente, um modelo de ambiente (BARTO,1998).

### A política:

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 (BARTO,1998).

### A função de recompensa:

Uma função de recompensa define a meta em um problema de *RL*. Ou seja, ela 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. (BARTO,1998).

### A função de valor:

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. 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 (BARTO,1998).

### O modelo de ambiente:

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 (BARTO,1998).

## Aprendizagem de diferença temporal

O método de diferença temporal de aprendizagem (TD) é uma combinação de ideias de Monte Carlo e de programação dinâmica ideias (DP). Como métodos de Monte Carlo, o método TD pode aprender diretamente de experiência crua, sem um modelo de dinâmica do ambiente (SUTTON, 1998).

## Método de aprendizagem *Q-Learning*

Um dos avanços mais importantes no aprendizado por reforço foi o desenvolvimento de um algoritmo de controle conhecido como *Q-Learning* (WATKINS, 1989). A sua forma mais simples, um episódio de *Q-Learning*, é definido pela equação 3.1, a cada episódio o algoritmo incrementa uma tabela de recompensas que servirá para a tomada de decisão do agente de aprendizado.

(3.1)

Onde:  
;  
;  
.

Para facilitar o entendimento a forma procedural do algoritmo de *Q-learning* é mostrada no quadro 1:

|  |
| --- |
| Quadro 1 – Forma procedural algoritmo Q-Learning Fonte: SUTTON, Richard S.; BARTO, **Andrew G. Reinforcement Learning: An Introduction**. (1998) p. 28. |

A essência básica de *Q-Learning* é que o agente de aprendizado tem uma representação dos estados ***s*** do ambiente e possíveis ações nesses estados ***a***, e aprende o valor de cada uma dessas ações em cada um desses estados. Intuitivamente, este valor, *Q*, é referido como o valor de estado-ação. Assim, em *Q-Learning* é necessário começar definindo todos os seus valores de estado de ação para zero e o agente vai ao redor explorar o espaço de estado-ação. Depois de tentar uma ação em um estado, é avaliado o estado que levou a ação ***a***. Se esta ação levou a um resultado indesejável, é reduzido o valor *Q* (ou peso) para que outras ações tenham um valor maior e sejam escolhidas na próxima vez que o agente estiver nesse estado. Da mesma forma, se há uma recompensa por tomar uma ação particular, o peso dessa ação para esse estado é aumentado, assim o agente de aprendizado é mais propenso a escolhê-la novamente na próxima vez que você estiver nesse estado (BARTO,1998).

## Aplicações do Q-Learning para controle de semáforos

A maioria dos métodos de controle de tráfego precisam de um modelo pré-definido de fluxo de tráfego para que tenham um curto tempo de resposta para predizer futuras condições. No método *Q-Learning* não é necessário nenhum tipo de modelo do ambiente e uma relação entre as possíveis ações, estados, e ambientes são aprendidos por interações com o ambiente. Neste contexto, foram Thorpe e Anderson (1996) que estudaram pela primeira vez métodos de *Reinforcement Learning* para controle de tráfego (WIERING, VREEKEN, VAN VEENEN, & KOOPMAN, 2004). Thorpe (1996) aplicou o método de SARSA (SUTTON, 1996) em um problema de controle de tráfego e avaliou o desempenho em três diferentes representações de um estado específico, ele usou algoritmos de redes neurais para estimar o valor de reforço. No seu estudo, estados eram definidos pelo número e posição dos veículos em todas as direções terminando em uma interseção. Uma ação para cada estado foi definida para mudar as cores de um semáforo de vermelho para verde e vice-versa. Essas características foram combinadas de três diferentes maneiras. A primeira delas foi chamada de “*Vehicle count*”. Nesta abordagem Thorpe (1996) construiu dez partições baseadas no número de carros considerando todos os pares de combinação destas dez partições nas direções de leste a oeste e de norte a sul e também considerando dois possíveis modos para os semáforos, 200 (10x10x2) estados eram entradas para o agente de aprendizado. Na segunda representação ou “distância fixa”, Thorpe (1996) dividiu cada faixa em intervalos de 34 metros, o que gerou quatro partições em cada faixa. Um bit “ocupado” é definido para mostrar a ausência ou presença de veículos em cada partição, o que causa a existência de oito componentes para toda faixa oeste-leste ou norte-sul e um componente para a luz do semáforo. No total, existia um vetor de nove componentes para a entrada da rede neural. A terceira representação era a de “distância variável”, de maneira similar a segunda representação, mas com uma distância variável para cada partição. As distâncias eram definidas em 15, 34, 67 e 122 metros, e novamente existem quatro partições em cada faixa e para o semáforo. O agente de aprendizado nesta representação tem nove componentes de entrada como na distância fixa. Thorpe (1996) definiu a recompensa para r = -1 em cada passo no percurso parra atingir o objetivo. A avaliação é feita em uma rede 4x4, e durante a avaliação o melhor resultado para simular os passos para limpar os veículos do ambiente pertenciam as partições variáveis, e para o caso do tempo mínimo de viagem que teve o melhor resultado.

Wiering (2000), propôs uma transição de modelos para estimar o tempo de espera para os semáforos verdes e vermelhos em cada interseção. Ele aplicou um agente múltiplo de aprendizado para controlar os semáforos. Seu método era centrado no veículo, cada carro estima seu próprio tempo de espera e se comunica com o semáforo mais próximo. Para a definição de estado ele considerou a orientação e o posicionamento do veículo em uma fila, e o seu endereço de destino. A ação foi definida para mudar entre a luz vermelha e verde, a para a função de recompensa, se o carro permanece na mesma posição r = 1 e de outra maneira r = 0. Neste sistema o objetivo era minimizar o tempo total de espera, e a aprender a função de atribuição para estimar o tempo médio de todos os veículos. Durante seus experimentos, ambas as comunicações, local e global, eram levadas em consideração para chegar em uma melhor decisão no controle de semáforos.

Abdulhai (2003), aplicou *Q-Learning* como um controlador de semáforos de trânsito. Ele executou um experimento para uma interseção isolada, mas com alguns traços do método de agente múltiplo. Neste caso de interseção única, estados são comprimentos de filas em quatro abordagens fazendo uma conexão com a interseção e o tempo decorrido. Ações são definidas como uma extensão das luzes verdes e vermelhas ou uma mudança para a próxima a ação. Neste caso a recompensa é considerada como uma penalidade e é o tempo total de atraso entre duas decisões sucessivas pelos veículos na fila formadas atrás da luz de parada de quatro abordagens conectando uma interseção. Além disso, uma função de potência foi usada para aproximar o equilíbrio do tamanho da fila alterando a recompensa, que é diretamente proporcional ao tamanho da fila em cada passo “*s*”. Isto é útil para que o agente de aprendizado não fique indiferente em uma fila muito grande ou muito pequena ou ainda filas de tamanho igual. Para o caso de múltiplas interseções alguns outros estados como a divisão entre duas intercessões podem ser adicionados, e a recompensa seria ponderada pela soma de todas as interseções, considerando a maior recompensa para a estrada principal. Abdulhai (2003) mostrou que o *Reinforcement Learning* e especificamente o *Q-Learning* são uma promissora abordagem para um controle de semáforos. O seu resultado para uma interseção única mostrou que o *Q-Learning* superou o controle de variáveis de trafego pré-definidas, e superou por pouco ou se iguala ao controlador pré-programado para situações de fluxo constante e uniformes.

Em Wunderlich, Liu, Elhanany, and Urbanik (2008), estabeleceram uma nova visão, o LQF (*Longest Queue First*), com um algoritmo de semáforos por agendamento para uma interseção isolada. O algoritmo LQF foi desenvolvido para um problema de controle de sinais e os conceitos foram empregados a partir do campo de comutação de pacotes em redes de computadores. Este método utiliza um algoritmo de união dos pesos máximos para minimizar o tamanho das filas em cada passo e levava ao menor atraso médio de um veículo através do cruzamento. Foi constatado que o LQF era estável e com um bom desempenho em variados cenários de controle de tráfego. Eles decidiram aplicar o LQF em uma rede de interseção múltipla em seu próximo estudo (AREL, LIU, URBANIK, & KOHLS, 2010). Em uma rede de interseção múltipla uma decisão de agendamento de fase em uma única interseção afetaria muito as condições de tráfego em sua vizinhança de interseção e aplicando o LQF a tarefa se torna ainda mais difícil. Nesta pesquisa de *Reinforcement Learning*, é usada para dar a capacidade de ter controle distribuído conforme necessário para a programação de múltiplos cruzamentos. Na verdade, eles introduziram um novo uso do sistema de múltiplo agente e a estrutura do *Reinforcement Learning* para obter uma política de controle de tráfego eficiente.

Alguns outros trabalhos que Abdulhai teve contribuição que falam sobre o controle de tráfego são Abdi, Moshiri, Abdulhai e Sedigh (2012, 2013), Tantawy, Abdulhai e Abdelgawad (2013). Em Tantawy (2013), foi proposto um controle adaptativo de tráfego que emprega uma abordagem de aprendizado por reforço em múltiplos agentes. Cada controlador (agente) era responsável pelo controle do tempo de um semáforo em uma única junção de tráfego. Ele propôs dois modelos distintos: modo independente (1), onde cada controlador de interseção trabalha independentemente de outros agentes; e o modo integrado (2) onde cada controlador coordenava as ações com interseções vizinhas. Ele testou um modelo com uma rede de 59 interseções em uma parte da cidade de Toronto, Canadá, na parte da manhã em um horário de grande movimento. Os resultados mostraram uma redução no tempo de espera médio de 27% no modo 1, e de 39% no modo 2.

O grande desafio de todos os controles com *Q-Learning Control* (QLC) é administrar o enorme número de espaços de estado-ações. Uma das soluções para reduzir o número de espaços é categorizando possíveis estados em grupos. Além disso, esta abordagem aumenta a taxa de aprendizado, limitando o número de estados para o número de grupos diminuindo a precisão do sistema. A maioria das propostas de QLC atribuem a uma ação o tempo verde do semáforo. Geralmente o tempo de espera é um período fixado, que pode se repetir até chegar em um *treshhold* máximo. Este período fixo pode diminuir a eficiência do sistema. Preparar informação suficiente para treinar o sistema pode gerar vários problemas para o QLC. *Q-Learning* sem treino suficiente não conseguem convergir para um resultado otimizado. No entanto, o *Q-Learning* é um método eficaz para adquirir um aprendizado em tempo real e é possível melhorar o seu desempenho conforme adapta-se a novas situações.

## Projeto de um QLC

Um modo de projetar um QLC pode ser feito em tabelas (ARAGHI, 2015). Estados são formados a partir da média do tamanho das filas. Todos os tamanhos das filas são categorizados em três alcances: baixo, médio e alto. Diferentes estados são formados por diferentes combinações destes valores.

O QLC é método tabular de *Q-Learning* e existe um número limitado de conjunto de ações neste controle. Um conjunto de ações são uma combinação de tempos verdes no semáforo em cada fase. O tempo cíclico é variável e baseado na demanda do tráfego.

|  |
| --- |
| Figura 4 – O processo de interação entre o QLC e um Simulador de tráfego |
| figura4 |
| Fonte: ARAGHI, S., KHOSRAVI, A., CREIGHTON D. A review on computational intelligence methods for controlling traffic signal timing, **Expert Systems with Applications** (2015). P. 1538–1550. |

A recompensa é 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.

O processo de interação entre o QLC e um Simulador de tráfego pode ser vista na Figura 4. 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*.

# Desenvolvimento utilizando Q-Learning

## Exemplo de utilização do q-learnig

Nesta etapa do projeto foi realizada uma implementação no software MATLAB, para solução de um problema de “melhor caminho”, com o uso do método de Q-Learning. O problema consiste em sair de um ponto de um ambiente (A) e chegar a outro ponto objetivo (C), respeitando as regras e limitações impostas pelo ambiente, assim definindo a política de funcionamento do sistema como pode ser visto na figura 5.

|  |
| --- |
| Figura 5 – Problema para encontrar o melhor caminho |
| Objetivo  Inicio |
| Fonte: do autor |

O problema consiste em sair do ponto A e encontrar o melhor caminho até o ponto C, seguindo as normas impostas em cada ponto, ou seja, de A apenas pode-se ir para B e D, de B pode-se ir para E, C e A, e assim por diante.

Este problema foi escolhido pois demonstra um problema básico de aprendizado de máquina e que ajuda no entendimento do método de Q-Learning demonstrando todos os pontos básicos importantes para implementação de um projeto mais complexo.

Para começar definiu-se valores de recompensa para cada ação em cada estado, os estados serão os pontos A, B, C, D, E e F, e as ações serão os caminhos que os agente pode tomar, por exemplo, de A ir para B. Os valores das ações estão descritos na tabela 1:

Tabela 1 – Estados e ações do problema proposto

|  |  |  |
| --- | --- | --- |
| **Estado** | **Ações** | **Recompensas** |
| A | Ir para B | 0 |
| Ir para D | 0 |
| B | Ir para A | 0 |
| Ir para C | 100 |
| Ir para E | 0 |
| D | Ir para A | 0 |
| Ir para E | 0 |
| E | Ir para B | 0 |
| Ir para D | 0 |
| Ir para F | 0 |
| F | Ir para C | 100 |
| Ir para E | 0 |

Apenas serão dados valores de recompensa altos quando a próxima ação for ir para C. Isto significa que o agente escolherá caminhos que forneçam maiores recompensas, e estas informações são guardadas em uma matriz que irá sendo “treinada” conforme o número de iterações aumenta.

O coeficiente de aprendizado define a velocidade de aprendizado e convergência de um valor alto recompensa e o fator de desconto define o quanto o agente de aprendizado irá explorar o ambiente.

A figura 7 mostra o resultado de uma simulação com o treino de 500 iterações, onde o agente deveria sair do ponto A e ir ao ponto C em cada iteração, e a cada vez que fizer o processo atualizar a tabela de recompensas (tabela Q). É possível ver o ganho de recompensa do agente ao longo das iterações, e é interessante observar que mesmo quando o sistema identifica um “melhor caminho”, ou seja quando o gráfico tende a estabilizar com um valor máximo de recompensa, ele decai devido ao fator de desconto, ou fator de exploração aplicado. Por isto o que se tem após aproximadamente 100 iterações é uma faixa de melhores recompensas para aquele sistema de conjunto de estados e ações. Assim fica fácil notar porque este método é eficiente com dados que são atualizados em tempo real, visto que ele sempre explora novos caminhos buscando uma nova melhor solução.

|  |
| --- |
| Figura 7 – Gráfico de aprendizado *Q-Learning* para o melhor caminho |
| C:\Users\Marke\Desktop\PROJETO INTEGRADO B\Qlearning\graph_qlearning_a_0.1_g_0.9.png |
| Fonte: do autor |

O gráfico de aprendizado, na Figura 7, está com a seguinte configuração:

Nesta configuração, o gráfico de aprendizado se comportou adequadamente, convergindo o aprendizado em torno de 100 iterações. Nas Figuras 8 e 9 e possível ver melhor desempenho na curva, na relação Recompensa versus Iterações.

Alterar o valor do coeficiente de aprendizado faz com que o sistema estabilize mais rapidamente, o que nesta aplicação onde o ambiente não muda com frequência, tende a ser mais perto do valor unitário para melhores resultados. Já o fator de desconto estimula a exploração e conforme ele diminuiu mais instável fica o valor de recompensa do agente de aprendizado do sistema.

O gráfico de aprendizado, na Figura 8, está com a seguinte configuração:

|  |
| --- |
| Figura 8 – Gráfico de aprendizado *Q-Learning* para o melhor caminho |
| C:\Users\Marke\Desktop\PROJETO INTEGRADO B\Qlearning\graph_qlearning_a_0.5_g_0.9.png |
| Fonte: do autor |

O gráfico de aprendizado, na Figura 9, está com a seguinte configuração:

|  |
| --- |
| Figura 9 – Gráfico de aprendizado *Q-Learning* para o melhor caminho |
| C:\Users\Marke\Desktop\PROJETO INTEGRADO B\Qlearning\graph_qlearning_a_0.9_g_0.9.png |
| Fonte: do autor |

O gráfico de aprendizado, na Figura 10, está com a seguinte configuração:

|  |
| --- |
| Figura 10 – Gráfico de aprendizado *Q-Learning* para o melhor caminho |
| C:\Users\Marke\Desktop\PROJETO INTEGRADO B\Qlearning\graph_qlearning_a_0.1_g_0.1.png |
| Fonte: do autor |

O gráfico de aprendizado, na Figura 11, está com a seguinte configuração:

|  |
| --- |
| Figura 11 – Gráfico de aprendizado *Q-Learning* para o melhor caminho |
| C:\Users\Marke\Desktop\PROJETO INTEGRADO B\Qlearning\graph_qlearning_a_0.1_g_0.5.png |
| Fonte: do autor |

O gráfico de aprendizado, na Figura 12, está com a seguinte configuração:

|  |
| --- |
| Figura 12 – Gráfico de aprendizado *Q-Learning* para o melhor caminho |
| C:\Users\Marke\Desktop\PROJETO INTEGRADO B\Qlearning\graph_qlearning_a_0.1_g_0.75.png |
| Fonte: do autor |

O código completo encontra-se no Anexo A.

## Adaptação do método Q-learning para controle de trânsito

O primeiro passo para implementação do Q-Learning em um controle de trânsito é a definição dos parâmetros básicos do algoritmo.

A função de recompensa foi definida dependendo do tamanho da fila de veículos a cada ciclo do semáforo. Cada ciclo representa a soma do tempo que um semáforo permanece verde com o tempo que ele permanece vermelho.

Os estados são o número total de veículos em cada via da esquina e o ciclo do semáforo.

As ações foram definidas como a mudança de tempo em que um semáforo permanece em verde e vermelho, dentro de um ciclo.

## definição do ambiente de simulação

Foi necessário a criação de um ambiente de simulação para testes, criado no software MATLAB, sendo necessário definir todos os parâmetros do ambiente onde o *Q-Learning* será implementado, como número de semáforos, tamanho da quadra, direção onde os veículos podem seguir, o tamanho médio de um veículo, a distância média entre dois veículos e a aceleração média do veículo, quantidade de veículos que entram em cada via em um ciclo do semáforo.

### o espaço do ambiente de simulação

A simulação foi feita com apenas uma esquina, para simplificar a implementação, e apenas dois semáforos, a figura 13 mostra de forma gráfica uma representação desta esquina, indicando os possíveis sentidos que os carros em cada via podem seguir. Cada quadra tem um tamanho de 100 metros por 100 metros.

O modelo contém duas vias, na via A os veículos podem apenas continuar em frente ou virar para esquerda, e na via B os veículos podem seguir em frente ou virar para esquerda. As duas vias contêm apenas uma mão, ou seja, um único sentido de tráfego.

|  |
| --- |
| Figura 13 – Ambiente de simulação |
| Via A  Via B |
| Fonte: do autor |

### a física do ambiente de simulação

Embora simples, este ambiente deve obedecer algumas regras físicas, para que os dados adquiridos na simulação tenham certa validade quando comparados a um modelo real.

A tabela 2 mostra as informações necessárias para a simulação baseado na posição em que o veículo está na fila de espera em uma das vias. O tamanho médio dos veículos foi definido através de uma média de uma lista de veículos mais populares que circulam no Brasil. A aceleração foi definida pela média de aceleração destes veículos consultando as informações técnicas fornecidas pelos fabricantes

A construção dos dados da tabela 2 foi construída a partir das equações básicas de mecânica clássica newtoniana.

Nesta simulação a velocidade máxima permitida será de 40 km/h, velocidade comum em vias urbanas no Brasil. E o tempo de um ciclo do semáforo (tempo de sinal verde mais o tempo de sinal vermelho será de 60 segundos), o que significa que quando um dos semáforos estiver com 35 segundos verde o outro estará com 25 segundos, e assim por diante.

Tabela 2 – Dados necessários para a simulação

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **POSIÇÃO** | **DISTANCIA DO SEMÁFORO (m)** | **TEMPO ATÉ ATINGIR 40km/h (s)** | **TEMPO ATÉ O SEMÁFORO (s)** | **DISTÂNCIA PERCORRIDA ACELERANDO** | **ATRASO (s)** | **DISTANCIA RESTANTE APÓS 40km/h** | **TEMPO ATÉ O SEMÁFORO a 40km/h(s)** | **TEMPO ACUMULADO (s)** |
| 1 | 6.34 | 5.6 | 2.53 | 6.34 | 1.5 | 0 | 0 | 4.03 |
| 2 | 12.68 | 3.37 | 12.68 | 3 | 0 | 0 | 6.37 |
| 3 | 19.02 | 4.21 | 19.02 | 4.5 | 0 | 0 | 8.71 |
| 4 | 25.36 | 4.91 | 25.36 | 6 | 0 | 0 | 10.91 |
| 5 | 31.7 | 5.52 | 31.70 | 7.5 | 0 | 0 | 13.02 |
| 6 | 38.04 | 6.07 | 31.11 | 9 | 6.933 | 6.224 | 21.30 |
| 7 | 44.38 | 6.58 | 31.11 | 10.5 | 13.273 | 6.794 | 23.87 |
| 8 | 50.72 | 7.05 | 31.11 | 12 | 19.613 | 7.365 | 26.41 |
| 9 | 57.06 | 7.49 | 31.11 | 13.5 | 25.953 | 7.936 | 28.92 |
| 10 | 63.4 | 7.90 | 31.11 | 15 | 32.293 | 8.506 | 31.41 |
| 11 | 69.74 | 8.30 | 31.11 | 16.5 | 38.633 | 9.077 | 33.87 |
| 12 | 76.08 | 8.67 | 31.11 | 18 | 44.973 | 9.648 | 36.32 |
| 13 | 82.42 | 9.04 | 31.11 | 19.5 | 51.313 | 10.218 | 38.75 |
| 14 | 88.76 | 9.38 | 31.11 | 21 | 57.653 | 10.789 | 41.17 |
| 15 | 95.1 | 9.72 | 31.11 | 22.5 | 63.993 | 11.360 | 43.58 |

Com estes dados é possível definir quantos carros passam em um determinado tempo que o semáforo permaneça verde, levando em consideração sua posição na fila, sua aceleração até atingir 40 km/h, e o tempo de atraso dado a posição na fila. A última coluna da tabela 2 mostra os tempos finais necessários para um carro em determinada posição ultrapassar o semáforo, por exemplo, com 40 segundos de tempo no sinal verde é possível atravessar até 13 carros. Uma simulação mais precisa teria que levar em consideração a aceleração variada de cada veículo, que alteraria o fluxo de veículos em cada ciclo. E número de veículos que entram em uma via será adquirido no futuro através de processamento de imagens, mas para esta simulação estes valores foram definidos previamente.

## Aplicação do algoritmo

O método de *Q-Learning* deve aprender e reaprender para fornecer a melhor escolha ao sistema de controle. Deste modo foi necessário implantar uma função de ganho de recompensa que atenda a um objetivo específico, neste caso diminuir a fila de carros em cada via. O algoritmo monta uma tabela com a recompensa ganha em cada escolha que tomou, dado um certo estado e as possíveis ações deste estado. Como foi visto no exemplo para encontrar um melhor caminho, o algoritmo testa caminhos possíveis randomicamente, mas apenas quando alcança um certo objetivo ele ganha recompensa, assim o *Q-Learning* descobre um “melhor caminho” devido a recompensa ganha quando atingiu sua meta. Cabe então a política do algoritmo selecionar este caminho e fornecer ao sistema.

Para o melhorar o fluxo de trânsito o algoritmo deve ser capaz de modificar o tempo dos semáforos para melhor se adaptar ao número de veículos, por exemplo se cada uma das via tem um fluxo de saída máximo de 9 veículos em 30 segundos de tempo verde no semáforo, quando mais veículos entraram em uma via e/ou menos em outra ele deve alterar o tempo para 15 segundos verde com a de menor tráfego e 45 segundos para a de maior, isto deve ser feito em tempo real para que ele aprenda qual é o melhor tempo para cada semáforo. Ou seja, um semáforo de tempo fixo 30 segundos verde e 30 segundos vermelho será eficiente quando as duas vias têm o mesmo fluxo, mas quando ambas se alternam é necessário um aprendizado em tempo real para otimizar o fluxo.

O fluxograma 1 mostra o fluxo do algoritmo, descrevendo as etapas do método.

Fluxograma 1- Q-Learning para tráfego de veículos



# Resultados

Feita a implementação em MATLAB, com a entrada de veículos randômica os parâmetros de aprendizado e fator de desconto devem avaliados. A figura 14 mostra o gráfico de aprendizado com a seguinte configuração:

|  |
| --- |
| Figura 14 – Gráfico de aprendizado *Q-Learning* para trânsito |
|  |
| Fonte: do autor |

É possível ver que nesta configuração o sistema não converge para uma região de estabilidade de recompensa. A figura 15 mostra o gráfico de aprendizado com os seguintes parâmetros:

|  |
| --- |
| Figura 15 – Gráfico de aprendizado *Q-Learning* para trânsito |
| C:\Users\Alysson\Desktop\graph_qlearning_1000ipng\graph_qlearning_a0.9_o0.9.1000ipng.png |
| Fonte: do autor |

Agora o sistema converge cedo para uma recompensa estável, mas está oscilando muito devido ao fator de desconto. A figura 16 mostra o gráfico de aprendizado com os seguintes parâmetros:

|  |
| --- |
| Figura 16 – Gráfico de aprendizado *Q-Learning* para trânsito |
| C:\Users\Alysson\Desktop\graph_qlearning_1000ipng\graph_qlearning_a0.9_o0.99.1000ipng.png |
| Fonte: do autor |

Nesta configuração o ganho de recompensa está mais estável e pode ser usado para um controle efetivo. O código comentado pode ser encontrado no Anexo B.

Ainda restam etapas a serem concluídas para chegar a uma resolução de comparação com um sistema real, como a criação de uma política de aproveitamento das informações providas do *Q-Learning*, mas para isto é necessária uma quantidade de informações do ambiente que se assemelham a realidade, mas ficou claro que uma abordagem de controle de trânsito em tempo real sempre será equivalente ou melhor que uma de tempo fixo.

# Conclusão

A inteligência artificial tem se tornado uma importante ferramenta computacional, e que pode ser aplicada nas mais variadas áreas. Este projeto busca desenvolver um algoritmo de aprendizado de máquina que se encaixe nos problemas atuais de trânsito de nossa região. Diminuir o tempo de espera, melhorar o fluxo no trânsito, diminuir o número de poluentes em nosso meio ambiente, minimizar o custo do transporte coletivo, são algumas das vantagens que esta pesquisa pode proporcionar a sociedade. Conforme revisado neste documento fica claro que o Brasil ainda está atrasado nesta área, mas se for oferecida uma solução de baixo custo e fácil implementação o mercado pode demonstrar interesse. Não existe um consenso geral para definir melhor maneira computacional de controlar semáforos de trânsito, o *Q-Learning* mostra um caminho, já que o agente de aprendizado consegue tomar decisões com base na sua experiência e pode sempre aprender com novas interações com o ambiente.

Como visto na revisão bibliográfica este algoritmo já vem sendo construído e adaptado por mais de 20 anos (WATKINS, 1989) e como foi mostrado já existem algumas implementações no controle de trânsito. Algumas abordagens resultaram em uma forma sólida e eficaz de controle, outras nem tanto.

A complexidade acerca deste tema pode ser um obstáculo, pois este conteúdo não é abordado em disciplinas do curso de graduação de Engenharia Elétrica. Mas tentar outra abordagem mais simples não conduziria a resultados satisfatórios de pesquisa e implementação.

Este projeto mostrou uma maneira simples de implementação do algoritmo de *Q-Learning* no controle do fluxo de veículos, mas a falta de um ambiente com mais complexidade e informação, evitou uma comparação igualitária do método discutido com o método de tempo fixo. Uma possível pesquisa futura, pode ser feita construindo um ambiente mais complexo e uma política completa de controle. O método de *Q-Learning* tem a capacidade de aprender em tempo real, e as mudanças no trânsito acontecem também são em tempo real, por isto ficou claro que ele oferece grandes avanços neste tipo de aplicação.

# Referências

*LIVROS*

SUTTON, Richard S.; BARTO, Andrew G. ***Reinforcement Learning*: An Introduction**. (1998).

THORPE, T. L., & ANDERSON, C. W. **Traffic light control using SARSA with three state representations. Technical Report**. IBM Corporation. (1996).

*ARTIGOS DE PERIÓDICOS*

ABDI, J., MOSHIRI, B., ABDULHAI, B., & SEDIGH, A. Short-term traffic flow forecasting: Parametric and nonparametric approaches via emotional temporal difference learning. Neural Computing and Applications, **Science Direct**, p. 23, 141–159. (2013).

ABDULHAI, B., PRINGLE, R., & KARAKOULAS, G. *Reinforcement Learning* for true adaptive traffic signal control. Journal of Transportation Engineering, **Science Direct,** p. 129, 278–285. (2003).

ARAGHI, S., KHOSRAVI, A., CREIGHTON D. A review on computational intelligence methods for controlling traffic signal timing, **Expert Systems with Applications**, P. 1538–1550. (2015).

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, **Science Direct**, p. 14, 1140–1150. (2013).

WIERING, M. Multi-agent *Reinforcement Learning* for traffic light control. **Science Direct** (2000).

WUNDERLICH, R., LIU, C., ELHANANY, I., & URBANIK, T. A novel signal-scheduling algorithm with quality-of-service provisioning for an isolated intersection. IEEE Transactions on Intelligent Transportation Systems, **Science Direct,** p. 9, 536–547. (2008).

YIT, K. C., NURMIN, B., AROLAND, K., SOO S. Y., KENNETH T. K. T. Q-Learning Based Traffic Optimization in Management of Signal Timing Plan. **INTERNATIONAL JOURNAL OF SIMULATION: SYSTEMS, SCIENCE & TECHNOLOGY** · JUNE (2011).

*NORMAS TÉCNICAS*

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 14724**: Informação e documentação — Trabalhos acadêmicos — Apresentação. 2011.

*DOCUMENTOS CONSULTADOS ONLINE*

JOSÉ RENATO SALATIEL. **Mobilidade urbana: Como solucionar o problema do trânsito nas metrópoles**. UOL – Atualidades, 2012. Disponível em: <<http://vestibular.uol.com.br/resumo-das-disciplinas/atualidades/mobilidade-urbana-como-solucionar-o-problema-do-transito-nas-metropoles.htm>> Acesso em: data (2 abr. 2016).

JON SCHMITZ. **City expands use of high-tech traffic signals**. Pittsburgh Post-Gazette (2013). Disponível em: <[http://www.post-gazette.com/news/transportation/2014/05/02/City-expands-use-of-high-tech-traffic-signals/stories/201405020123>](http://www.post-gazette.com/news/transportation/2014/05/02/City-expands-use-of-high-tech-traffic-signals/stories/201405020123%3e%20) Acesso em: data (02 abr. 2016).

KEITH BARRY. **The Traffic Lights of Tomorrow Will Actively Manage Congestion**. Citylab (2014). Disponível em: <[http://www.citylab.com/commute/2014/09/the-traffic-lights-of-tomorrow-will-actively-manage-congestion/379950/>](http://www.citylab.com/commute/2014/09/the-traffic-lights-of-tomorrow-will-actively-manage-congestion/379950/%3e%20) Acesso em: data (1 abr. 2016).

LEE DAVIDSON. **Odds of hitting a red light in Utah? Just 1-in-4**. The Salt Lake Tribune (2013). Disponível em: <<http://archive.sltrib.com/story.php?ref=/sltrib/politics/57276984-90/traffic-percent-system-state.html.csp>> Acesso em: data (1 abr. 2016).

PREFEITURA DE MOGI DAS CRUZES. **“Semáforo inteligente” utiliza tecnologia para garantir fluidez e segurança ao trânsito** (2015). Disponível em: <<http://www.mogidascruzes.sp.gov.br/comunicacao/noticia.php?id=8977>> Acesso em: data (02 abr. 2016).

STEPHEN SMITH, GREG BARLOW, XIAO-FENG XIE, DR. DEBORAH D. STINE, DR. ENES HOSGOR. **SURTRAC** **Smart Traffic Light - Final Report** (2014). Disponível em: <<https://www.cmu.edu/epp/people/faculty/course-reports/SURTRAC%20Final%20Report.pdf>> Acesso em: data (02 abr. 2016).

UN-HABITAT. **State of Latin American and Caribbean cities** (2012). Disponível em: <<http://unhabitat.org/books/state-of-latin-american-and-caribbean-cities-2/>> Acesso em: data (10 mai. 2016).

FIRJAN. “O custo dos deslocamentos nas principais áreas urbanas do Brasil (2015). Disponível em: <http://www.firjan.com.br/lumis/portal/file/fileDownload.jsp?fileId=2C908A8F4F8A7DD3014FB26C8F3D26FE&inline=1> Acesso em: data (10 mai. 2016).

# Anexos

## Anexo A

Código em MATLAB para aplicação do *Q-Learning:*

%% Programa Q-Learning para melhor caminho

% UNIVERSIDADE FEDERAL DO PARANÁ

% CURSO DE GRADUAÇÃO DE ENGENHARIA ELÉTRICA (ÊNFASE EM SIST. EMBARCADOS)

%

% Autores:

% Alysson Jhovert Malko de Freitas

% Marcos Yamasaki

%

% data: 24/05/2016

%

%

clear all

clc

% Estados A,B,C,D,E,F

% |A|B|C|

% |D|E|F|

%

% Inicia-se no Estado A

% Objetivo: Chegar ao Estado C.

% Parâmetros:

% Coeficiente de aprendizagem

alpha **=** 0.1**;**

% Fator de desconto

gamma **=** 0.1**;**

% Representação em valores numéricos para cada Estado

estadoA **=** 1**;**

estadoB **=** 2**;**

estadoC **=** 3**;**

estadoD **=** 4**;**

estadoE **=** 5**;**

estadoF **=** 6**;**

estadosCount **=** 6**;**

estados **=** **{**estadoA**,**estadoB**,**estadoC**,**estadoD**,**estadoE**,**estadoF**};**

% Declaração de matrizes com zeros (R, Q, maxQ)

R **=** **[**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**)];**

Q **=** **[**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**)];**

maxQ **=** **[**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**);**zeros**(**1**,**6**)];**

% Declaração de possíveis ações de cada Estado

acoesdeA **=** **{** estadoB**,** estadoD **};**

acoesdeB **=** **{** estadoA**,** estadoC**,** estadoE **};**

acoesdeC **=** **{** estadoC **};**

acoesdeD **=** **{** estadoA**,** estadoE **};**

acoesdeE **=** **{** estadoB**,** estadoD**,** estadoF **};**

acoesdeF **=** **{** estadoC**,** estadoE **};**

acoes **=** **{** acoesdeA**,** acoesdeB**,** acoesdeC**,** acoesdeD**,** acoesdeE**,** acoesdeF **};**

estadoNames **=** **{** 'A'**,** 'B'**,** 'C'**,** 'D'**,** 'E'**,** 'F' **};**

% Definição de recompensa

R**(**estadoB**,**estadoC**)** **=** 100**;** % estadoB para estadoC

R**(**estadoF**,**estadoC**)** **=** 100**;** % estadoF para estadoC

cont**=**0**;**

somaQ **=** 0**;**

% Definição de Objetivo

objetivo **=** estadoC**;**

%% Iniciar treino

tic % Inicia a contagem de execução

**for** i **=** 1**:**500

estado **=** estadoA**;** % Iniciar no Estado A

**while** estado **~=** objetivo % Não terminar até alcançar o objetivo (Estado C)

% Obter as possíveis ações do Estado atual

acoesDeEstado **=** acoes**{**estado**};**

% Selecionar randomicamente qual caminho seguir (qual ação tomar)

ind **=** randi**(**length**(**acoesDeEstado**));**

acao **=** acoesDeEstado**{**ind**};**

% Definição do próx estado a partir da ação atual

proxEstado **=** acao**;**

% Valor de Q

q **=** Q**(**estado**,**acao**);**

% Cálculo de maxQ

acoesDeEstado **=** acoes**{**proxEstado**};** % Ações do Estado atual

maxQ **=** 10**^-**20**;**

**for** k **=** 1**:**length**(**acoesDeEstado**)**

valor **=** Q**(**proxEstado**,**acoesDeEstado**{**k**});**

**if** **(**valor **>** maxQ**)**

maxQ **=** valor**;**

**end**

**end**

%

% Valor de Recompensa

r **=** R**(**estado**,**acao**);**

% Cálculo do valor Q

valorQ **=** q **+** alpha **\*** **(**r **+** gamma **\*** maxQ **-** q**);**

% Armazena Q

Q**(**estado**,**acao**)** **=** valorQ**;**

% Próximo estado

estado **=** proxEstado**;**

% Soma de Q e contador de ações para o Gráfico

somaQ **=** somaQ **+** valorQ**;**

cont**=**cont**+**1**;**

**end**

% plot(ite,qSalvo)

qSalvo**(**i**)=**somaQ**/**cont**;**

ite**(**i**)=**i**;**

% Limpa variáveis

somaQ **=** 0**;**

cont **=** 0**;**

**end**

% Termina contagem de execução

t2 **=** toc**;**

% Plot do gráfico

plot**(**ite**,**qSalvo**,**'b'**)**

xlabel**(**'Iterações'**)**

ylabel**(**'Recompensa'**)**

title**(**'Gráfico de aprendizado Q-learning para o melhor caminho'**)**

grid on

fprintf**(**'\nTempo de execução: %f \n'**,** t2**);**

## Anexo B

%% Programa Q-Learning para melhor decisão do Semáforo

% UNIVERSIDADE FEDERAL DO PARANÁ

% CURSO DE GRADUAÇÃO DE ENGENHARIA ELÉTRICA (ÊNFASE EM SIST. EMBARCADOS)

%

% Autores:

% Alysson Jhovert Malko de Freitas

% Marcos Yamasaki

%

% data: 25/06/2016

clear all

clc

% Inicia-se no Estado 7 (30s 30s)

% Objetivo: Tomar a melhor decisão dada pela proporção de cada via.

% Parâmetros:

% Coeficiente de aprendizagem

alpha **=** 0.9**;**

% Fator de desconto

gamma **=** 0.98**;**

% Representação em valores numéricos para cada Estado

% Tempo Aberto 1 = 60 55 50 45...

% Tempo Fechado 1= 00 05 10 15...

% Tempo Aberto 2 = 00 05 10 15...

% Tempo Fechado 2= 60 55 50 45...

tA1 **=** **(**1**:**13**)\***5 **-** 5

tF1 **=** abs**((**1**:**13**)\***5 **-** 5 **-** 60**)**

tF2 **=** tA1**;**

tA2 **=** tF1**;**

estadoAF1 **=** **[**tA1**;** tF1**]**

estadoAF2 **=** **[**tA2**;** tF2**]**

estadoAF **=** **[**tA1**]**

estadoAFind **=** 1**:**15**;**

estadosCount **=** 13**;**

estados **=** estadoAFind**;**

% Declaração de matrizes com zeros (R, Q, maxQ)

R **=** zeros**(**13**,**13**);**

Q **=** zeros**(**13**,**13**);**

maxQ **=** zeros**(**13**,**13**);**

% Declaração de possíveis ações de cada Estado

acoesdeAF**{**1**}** **=** **{**estadoAFind**(**2**)};**

acoesdeAF**{**13**}** **=** **{**estadoAFind**(**12**)};**

num3 **=** 3**;**

num1 **=** 1**;**

**for** num2 **=** 2**:**12

acoesdeAF**{**num2**}** **=** **{**estadoAFind**(**num1**),** estadoAFind**(**num3**)};**

num3 **=** num3 **+** 1**;**

num1 **=** num1 **+** 1**;**

**end**

acoes **=** **[**acoesdeAF**];**

% Definição de Variáveis para Esquina

carrosVindo1 **=** **(**randi**([**5 15**],**5200**,**1**))';**

carrosVindo2 **=** **(**randi**([**1 5**],**5200**,**1**))';**

qtdDeCarros1 **=** 0**;**

qtdDeCarros2 **=** 0**;**

% Definição para carros vindo, somatório das 2 vias

**for** sumcar **=** 1**:**5200

carrosVindo**(**sumcar**)** **=** carrosVindo1**(**sumcar**)** **+** carrosVindo2**(**sumcar**);**

**end**

% Definição tamanho do comprimento médio de um carro

compCarro **=** 4.34**;**

% Quantidade de carros maxima

% Considerando:

% Velocidade média de 40 km/h = 11,11 m/s;

% Tempo médio de 0 a 100km/h = 14s (Renault-Clio)

% Aceleração média = 100/14 = de 7,1437,143 km/h = 1,984 m/s;

% Tempo médio de 0 a 40km/h = 11,11/1,984 = 5,6s

% Comprimento médio de uma via em um quarteirão = 100 m

% Recomendações do DETRAN para distância entre veículos deve ser que um

% exista espaço suficiente 'de um carro' entre dois carros.

% Para fins de simulação, será desconsiderado isso e utilizado uma

% distância mais próxima do real = 1.5 m.

% Distância média entre os veículos

distCarro **=** 2**;**

% Quantidade máxima de carros em uma via = 100/(4.34+2) = 15 carros

qtdDeCarrosMaxVia **=** fix**(**100**/(**compCarro**+**distCarro**));**

% Tendo 15 carros na via, é possível definir 15 posições de tamanho 6,34 m

% (compCarro+distCarro) cada.

% Tempo médio de atraso para acelerar, após sinal aberto = 1.5s

% Tempo médio acumulado de cada carro, em cada posição para passar o

% semáforo:

tempoCarro **=** **[**4.03 6.37 8.71 10.91 13.02 21.30 23.87 26.41 28.92 31.41 33.87 36.32 38.75 41.17 43.58**];**

% Inicializando variáveis

cont**=**0**;**

somaQ **=** 0**;**

% Definição de Objetivo

qtdDeCarros1Lim **=** 0**;**

qtdDeCarros2Lim **=** 0**;**

% Iniciar no Estado 7 (30s 30s)

estado **=** estadoAFind**(**7**);**

%% Iniciar treino

tic % Inicia a contagem de execução

%carrosTotal = zeros(1,5200);

**for** i **=** 1**:**1000

saiu **=** 0**;**

% Laço para achar a melhor decisão

**while** **(**saiu **<** 1**)**

% Obter as possíveis ações do Estado atual

acoesDeEstado **=** acoes**{**estado**};**

% Selecionar randomicamente qual caminho seguir (qual ação tomar)

ind **=** randi**(**length**(**acoesDeEstado**));**

acao **=** acoesDeEstado**{**ind**};**

% Definição do próx estado a partir da ação atual

proxEstado **=** acao**;**

% Valor de Q

q **=** Q**(**estado**,**acao**);**

% Cálculo de maxQ

acoesDeEstado **=** acoes**{**proxEstado**};** % Ações do Estado atual

maxQ **=** 10**^-**20**;**

**for** k **=** 1**:**length**(**acoesDeEstado**)**

valor **=** Q**(**proxEstado**,**acoesDeEstado**{**k**});**

**if** **(**valor **>** maxQ**)**

maxQ **=** valor**;**

**end**

**end**

% Zera posição dos carros

posCarros **=** zeros**(**1**,**15**);**

% Limite da via para 15 carros.

**if** **(**carrosVindo1**(**i**)** **>** qtdDeCarrosMaxVia**)**

qtdDeCarros1 **=** 15**;**

**end**

**if** **(**carrosVindo2**(**i**)** **>** qtdDeCarrosMaxVia**)**

qtdDeCarros2 **=** 15**;**

**end**

% Definindo tempo aberto do outro semáforo, baseado no ciclo de

% 60s cada

tempoAberto2 **=** 60 **-** estadoAF**(**estado**);**

% Definindo qtd de carros que passará pelo semáforo baseado no

% estado atual (tempo de sinal aberto)

**for** tmpR1 **=** 1**:**15

**if** **(**estadoAF**(**estado**)** **<** tempoCarro**(**tmpR1**))**

qtdDeCarros1Lim **=** tmpR1**-**1**;**

**break;**

**end**

**end**

**for** tmpR2 **=** 1**:**15

**if** **(**tempoAberto2 **<** tempoCarro**(**tmpR2**))**

qtdDeCarros2Lim **=** tmpR2**-**1**;**

**break;**

**end**

**end**

% Quantidade de carros que passou pelo semáforo estimado pelo

% tempo aberto (considerando: velocidade máxima, aceleração média,

% tamanho médio da via, atraso de percepção do motorista,

% comprimento médio de um carro e distância média

% entre os veículos)

**if(**carrosVindo1**(**i**)** **>** qtdDeCarros1Lim**)**

qtdDeCarros1 **=** qtdDeCarros1Lim**;**

**else**

qtdDeCarros1 **=** carrosVindo1**(**i**);**

**end**

**if(**carrosVindo2**(**i**)** **>** qtdDeCarros2Lim**)**

qtdDeCarros2 **=** qtdDeCarros2Lim**;**

**else**

qtdDeCarros2 **=** carrosVindo2**(**i**);**

**end**

% Quantidade total de carros que passou pelo semáforo em 1 ciclo

qtdDeCarros **=** qtdDeCarros1 **+** qtdDeCarros2**;**

% Definição de Recompensa dependendo da quantidade de carros de

% cada via

carrosRazao1 **=** qtdDeCarros1**/**carrosVindo1**(**i**);**

carrosRazao2 **=** qtdDeCarros2**/**carrosVindo2**(**i**);**

**if(**carrosRazao1 **>** carrosRazao2**)**

**if(**acao**>**estado**)**

saiu **=** 1**;**

R **=** 1**;**

**else**

R **=** 0**;**

**end**

**elseif(**carrosRazao1 **==** carrosRazao2**)**

**if(**acao**==**estado**)**

saiu **=** 1**;**

R **=** 1**;**

**else**

R **=** 0**;**

**end**

**else**

**if(**acao**<**estado**)**

saiu **=** 1**;**

R **=** 1**;**

**else**

R **=** 0**;**

**end**

**end**

% Valor de Recompensa

r **=** R**;**

% Cálculo do valor Q

valorQ **=** q **+** alpha **\*** **(**r **+** gamma **\*** maxQ **-** q**);**

% Armazena Q

Q**(**estado**,**acao**)** **=** valorQ**;**

% Próximo estado

estado **=** proxEstado**;**

% Soma de Q e contador de ações para o Gráfico

somaQ **=** somaQ **+** valorQ**;**

cont**=**cont**+**1**;**

**end**

carrosTotal**(**i**)** **=** qtdDeCarros**;**

qSalvo**(**i**)=**somaQ**/**cont**;**

ite**(**i**)=**i**;**

% Limpa variáveis

somaQ **=** 0**;**

cont **=** 0**;**

**end**

% Termina contagem de execução

t2 **=** toc**;**

% Plot do gráfico

plot**(**ite**,**qSalvo**,**'b'**)**

xlabel**(**'Iterações'**)**

ylabel**(**'Recompensa'**)**

title**(**'Gráfico de aprendizado Q-learning para Semáforo'**)**

grid on

fprintf**(**'\nTempo de execução: %f \n'**,** t2**);**