Faculdade de Engenharia da Universidade do Porto

T02_03: Tráfego Numa Cidade

André Rodrigues Barros - up201303567@fe.up.pt
José Miguel Botelho Mendes - up2013004828@fe.up.pt
Pedro Miguel Vieira da Silva - up201306032@fe.up.pt

Trabalho realizada(o) no âmbito do
Mestrado Integrado em Engenharia Informática de Computação
Agentes de Inteligência Artificial Distribuída

16 de Dezembro de 2016

Índice

Objetivo

No âmbito desta Unidade Curricular o grupo propôs-se a desenvolver um sistema de simulação baseado em agentes que visa estudar o tráfego automóvel numa determinada cidade.

Este problema surge pois é bastante frequente nas grandes cidades haver bastante tráfego automóvel durante certos momentos do dia, situação esta que impede, em certos casos, os automobilistas de chegarem no tempo previsto ao local pretendido. Esta situação, aliada ao facto de ser bastante desconfortável para os condutores estarem muito tempo parados numa fila, leva à necessidade de estudar o tráfego automóvel nas cidades de forma a encontrar soluções que ajudem pelo menos atenuar este problema para os automobilistas já que a sua solução parece neste momento complicada de encontrar.

O objetivo deste trabalho é construir uma simulação cujo cenário é constituído pelas diversas ruas e estradas numa cidade. Os semáforos em cada cruzamento representam agentes, bem como os automóveis em circulação. Os semáforos têm uma determinada temporização, ou seja, um determinado tempo num estado. A mudança do seu estado pode ser automática ou atualizada pelo próprio agente. Por seu lado, cada automóvel tem como objetivo chegar a um determinado ponto da cidade, e pode tomar decisões de acordo com a informação que lhe chega ao longo da viagem.

Especificação

Caracterização dos Agentes

Para a realização deste trabalho, foram usados como agentes os veículos e semáforos, sendo estes criados através da ferramenta *SUMO* (Simulation of Urban MObility). Para além dos agentes, a estrada presente na interface gráfica é gerada por um ficheiro .xml, tal como os diferentes estados dos semáforos. Apesar de serem usados veículos como agentes, o comportamento que foi explorado e alterado neste trabalho foi o dos semáforos. Na ferramenta *SUMO* cada semáforo é constituído por uma intersecção de ruas em que podem existir vários sinais luminosos, ou seja, para cada objecto semáforo da ferramenta podem corresponder vários semáforos na interface gráfica. Para além disso, o *SUMO* gera, para cada semáforo um conjunto de fases pré-definida, isto é, um conjunto de estados e respectivas durações, ainda que nunca menor que 4 *ticks,* de forma a evitar as colisões das viaturas nas intercepções. Assim, cada semáforo repete as suas fases em ciclo, repetindo os estados. Deste modo, foram criados os seguintes agentes:

Agente básico

Este agente é utilizado como “grupo de controlo”, visto que os agentes apenas seguem a configuração pré-definida pelo *SUMO*.

Agente “*Intersection*”

O agente “*intersection*” já apresenta uma diferenciação para o básico. Resumidamente, este agente irá analisar, a cada *tick* e em todos os semáforos, o número de carros parados (velocidade < 0.3 é considerado como parado) e o estado dos semáforos. Com estes dados o agente irá verificar se existe um número de carros parados num semáforo vermelho suficiente (neste caso este número é 4) para se justificar a alteração do estado do semáforo, considerando que não existe nenhum veículo a circular na rua que se encontra com o semáforo a verde. Ora, se esta condição se verificar o agente do semáforo irá mudar o seu estado para o próximo do seu conjunto de fases.

Agente "*Learning"*

O Agente "Learning" aplica o algoritmo *Q-Learning*, que se trata de um algoritmo de reforço que selecciona ações num estado dependendo da qualidade Q do par ação-consequência. O algoritmo converge para as melhores combinações de pares ação-consequência depois de terem sido experimentadas vezes suficientes. A escolha da Função de Reforço (recompensa) deve estar de acordo com os objetivos do Agente. A recompensa é uma medida imediatista. Estimam-se Funções de Valor que determinam a qualidade do estado ou a qualidade de executar uma ação num determinado estado. O Valor associado ao estado (ou estado-ação) é uma medida a longo prazo: Q(s,a) = Q(s,a) + a[r + y*max Q(s',a')-Q(s,a)] onde 0<=a<=1 é a taxa de aprendizagem, r é o reforço obtido pela execução de a em s, 0<=y<=1 é o factor de desconto representando a importância dos valores Q nos estados futuros para o valor a actualizar. No nosso caso, em específico, o valor da taxa de aprendizagem, o "a", decidimos a 1 pois é um modelo determinístico. O nosso fator de desconto, o "b", fixamos em 0.3, de forma a ter um impacto mais imediato no algoritmo. Usamos 9 estados na tabela Q, pois decidimos que o tempo do semáforo iria variar entre 20 segundos e 60, com um *step* de 5 segundos (60-20/5 = 8).

Agente Manager

Este agente é utilizado para gerir os dados sobre todos os agentes, sendo responsável pela sua inicialização, e algumas atualizações de estados do próprio SUMO.

Protocolos de interação

Considerando uma aproximação ao mundo real, foi decidido que não deveria haver comunicação entre veículos e semáforos, devido, principalmente, à velocidade da comunicação e a segurança. Assim sendo, faltava estabelecer um protocolo entre os agentes de semáforos que iriam comunicar, para tal, foi concebido um protocolo baseado em duas interações. O objectivo deste protocolo é um semáforo comunicar com os seus vizinhos de modo a receber a melhor ação a tomar. Esta comunicação acontece a cada vez que um semáforo completa um ciclo, ou seja, cada vez que ele passa pelos sequência de estados: verde, amarelo, vermelho, verde e amarelo. Assim, é possível aplicar o algoritmo de learning e fazer update à tabela de decisões de cada semáforo de maneira a obter a melhor ação a tomar no próximo ciclo.

Desenvolvimento

Plataformas/Ferramentas Utilizadas

Nesta aplicação são usadas 3 ferramentas que facilitam a implementação da mesma.

SUMO (**S**imulation Of **U**rban **Mo**bility)

SUMO é um programa *open-source* de simulação de tráfego automóvel. O seu modelo de simulação é microscópico, ou seja, cada veículo é modelado explicitamente e tem as suas próprias rotas definidas individualmente. O seu valor advém da existência de diferentes tipos de veículos, estradas com várias faixas, semáforos e uma interface gráfica para visualizar a rede criada e as entidades que estão a ser simuladas. Além disso, oferece também uma API chamada TraCI, que trata da interoperabilidade com outras aplicações em tempo real. Finalmente, a interoperabilidade com outras aplicações permite que cada agente esteja ligado a uma entidade no SUMO, ou seja, alterações nas dinâmicas de semáforos, podem ser visualizadas na interface gráfica.

JADE (**J**AVA **A**gent **De**velopment Framework)

JADE é uma framework desenvolvida em Java que simplifica a implementação de sistemas de multi-agentes (MAS) através de um *middleware* que cumpre com as especificações do FIPA () e um conjunto de ferramentas gráficas que suportam as fases de *debug* e *deployment.* Além disso, esta plataforma de agentes corre em qualquer máquina, independentemente do seu sistema operativo. Entre todas as vantagens oferecidas pelo JADE, as mais úteis são a possibilidade de visualizar e controlar as interações entre agentes em tempo real, o transporte de mensagens, codificação e *parsing* de mensagens ou até mesmo o tempo de vida de um agente.

TraSMAPI (**Tra**ffic **S**imulation **M**anager **A**pplication **P**rogramming **I**nterface)

TraSMAPI é uma API para tráfego que permite comunicação em tempo real entre agentes no âmbito de tráfego numa cidade (veículos e semáforos) e o ambiente criado por vários simuladores, entre eles o SUMO.

Esta API tem um alto nível de abstração, ou seja, a solução é independente do simulador microscópico que se use. Existem três módulos principais: um módulo de comunicação com a possibilidade de vários simuladores microscópicos, um módulo que gera estatísticas e um módulo para gerir o MAS. No entanto, apenas o módulo de comunicação é importante para o âmbito da aplicação.

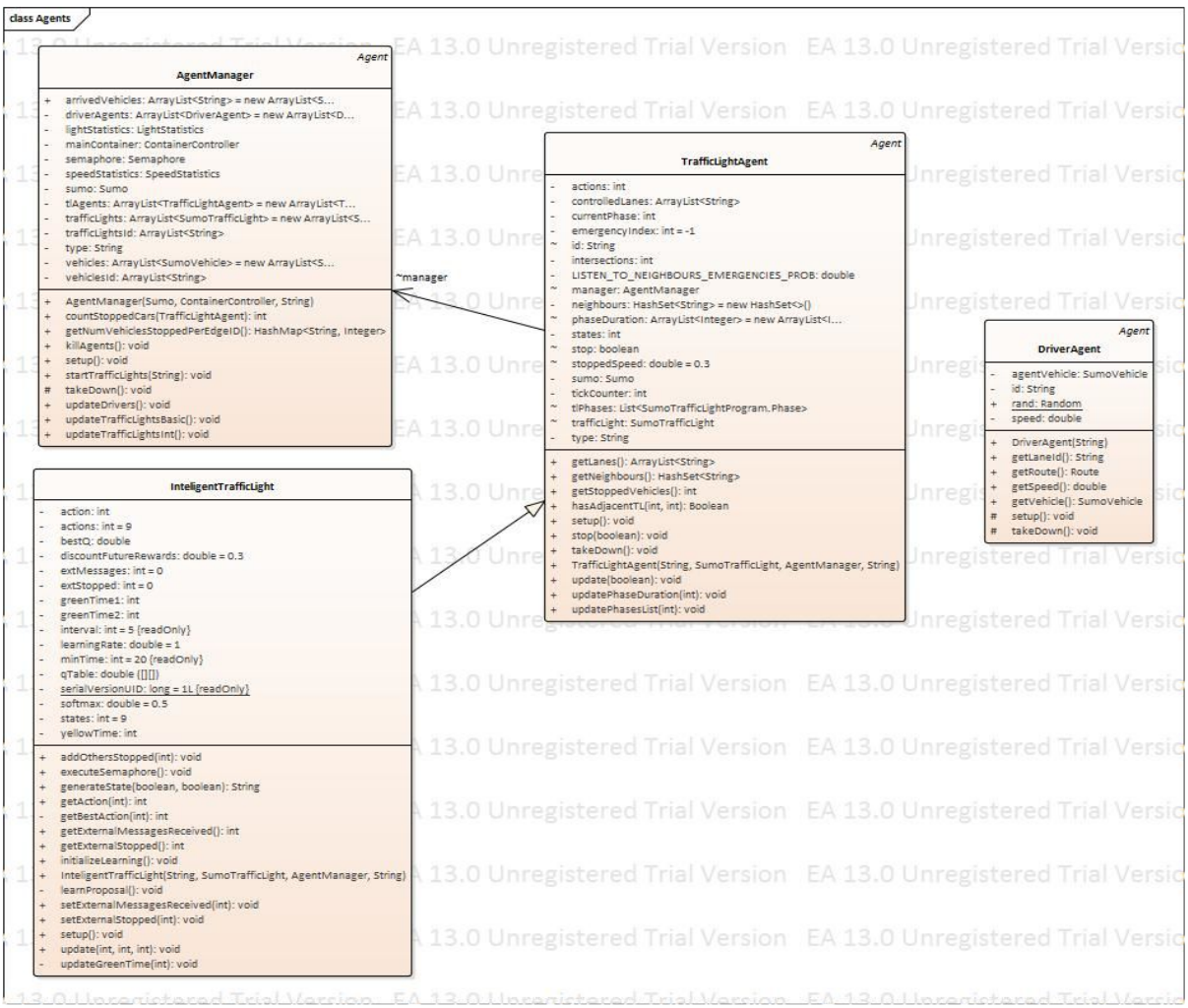
Ambiente de Desenvolvimento

Toda a aplicação foi desenvolvida em Windows 10 e com o IntellIJ (IDE). A versão usada de SUMO foi a 0.28.0, a versão de JADE foi a 4.4.0 e o TraSMAPI foi a versão mais recente do Github.

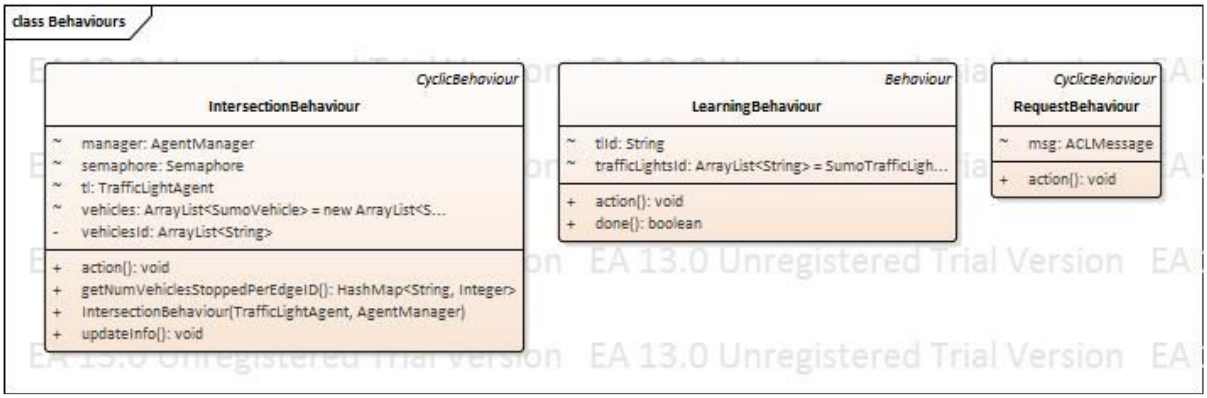
Estrutura da aplicação

A aplicação está, presentemente, dividida em 5 pacotes:

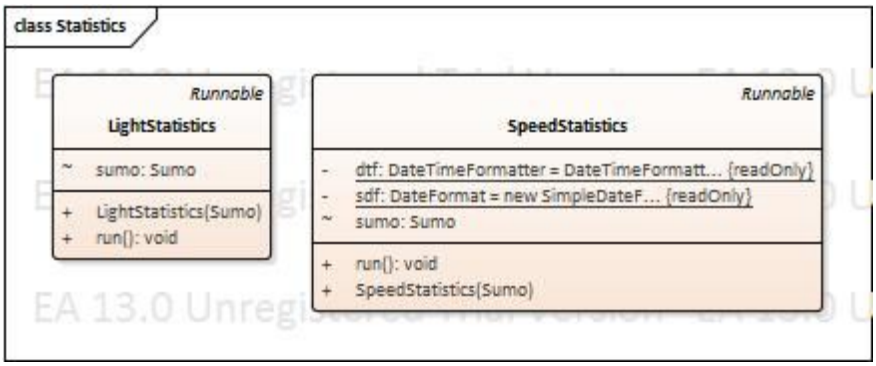
- Agents: Este *package* contém 4 classes, todas elas agentes, extendendo a classe *Agent* do JADE. *DriverAgent*, o veículo agente, *TrafficLightAgent*, o semáforo como agente e *IntelligentTrafficLight* que além de ser um agente, tem implementado o *Learning*. Além disso, a classe *AgentManager* possui informação de todos os agentes do programa, estando encarregue da sua inicialização.



- Behaviours: Este *package* é formado por 3 classes. *IntersectionBehaviour*, que constrói um *Behaviour* em que sempre que existem 5 carros parados num semáforo, pede ao semáforo para trocar o seu sinal para verde. O *RequestBehaviour* trata da comunicação dos agentes Semáforo e, por fim, o *LearningBehaviour* implementa o algoritmo *Q-Learning* aplicado aos semáforos.



- Statistics: Este *package* trata de escrever para um ficheiro estatísticas da aplicação, após ser corrida. Possui 2 classes, *LightStatistics*, que calcula o tempo médio de espera dos carros em simulação e a classe *SpeedStatistics*, que faz pedidos ao SUMO para saber a velocidade de cada veículo.



- trasmapi: Este pacote é a API usada para fazer pedidos ao SUMO.
- Main: Este pacote apenas contém a classe *Main* que inicializa a aplicação.

Detalhes relevantes da implementação

Durante a nossa implementação, as maiores dificuldades prenderam-se ao facto da dificuldade sentida ao usar *TraSMAPI+SUMO.* Concretamente, uma das funções disponibilizadas para alterar as fases de um semáforo ("*SumoTrafficLight.setProgram()*") não funcionava corretamente, sendo que por este motivo, tornou-se imperativo realizar as mudanças de estado de todos os semáforos manualmente. Para tal, a cada *tick* do nosso programa é usanda a função "*SumoTrafficLight.setState()",* que altera diretamente a String de estado do semáforo. Outro detalhe relevante prende-se com a forma de operar do simulador *SUMO*, uma vez que os agentes não são instanciados automaticamente no seu momento de criação. Assim, a nossa ferramenta necessita de, em cada iteração da simulação, "pedir" ao *SUMO* as listas de agentes ativos e inativos, atuando com essas informações.

Experiências

Objetivo de cada experiência

Considerando os vários tipos de agentes já descritos, foram realizadas várias experiências de modo a poder comparar os seus desempenhos. Assim, são em baixo apresentados os resultados das diversas experiências, apresentando a média de tempo de paragem dos veículos (em segundos) das múltiplas execuções para cada tipo de agentes .

Resultados

Tipo de agente	Básico	Intersection	Learning
Núm. veículos	200	200	200
Média de tempo de paragem	2.677	3.0	2.1

Conclusões

Análise dos resultados

Como foi anteriormente referido, o resultado da experiência com o agente básico serve como "grupo de controlo" para as restantes experiências, logo não há informações relevantes a retirar deste.

No que diz respeito ao agente "Intersection" pode-se notar que tem um desempenho inferior ao básico no que diz respeito ao tempo médio de espera. Este resultado é um pouco surpreendente visto que se trata de uma abordagem reativa, sendo que o comportamento por predefinição é seguido, alterando a fase do semáforo apenas quando existem demasiados carros parados no sinal vermelho, retornando de seguida para o comportamento normal.

Analisando, agora, o agente "Learning" verifica-se que este tempo médio de espera é reduzido comparativamente ao comportamento básico. Isto deve-se ao facto de existir comunicação entre os vários semáforos, de modo a saberem qual a melhor ação a tomar, aplicando o algoritmo *Q-Learning* que mantém uma tabela com os pares ação-consequência de cada semáforo, em que a cada vez que é preciso tomar uma ação esta é escolhida tendo em conta o melhor valor qualidadeQ de cada um destes pares.

Todas estas experiências devem no entanto ter em conta um detalhe da implementação do SUMO. Por ser um simulador do mundo real e por não querer pôr em causa as regras de trânsito, os veículos não são encorajados a passar em estados amarelos. Aliás com a velocidade de veículos por nós definida, é impossível um veículo passar um semáforo no estado amarelo. Esta é a razão pela qual o resultado dos nossos agentes inteligentes se assemelha tanto ao básico e por vezes até pioram seu performance. Por este motivo os veículos passam bastante tempo parado porque para além de não poderem atravessar o cruzamento enquanto está amarelo, o estado amarelo é o único cujo o estado não é acelerado ou avançado. Torna-se aparente então que um agente que prolongue mais estes estados pode ter melhores resultados na simulação mas isto não se traduz necessariamente ao mundo real.

Sistema Multi-Agente (MAS)

Tal como descrito no enunciado, a abordagem do problema foi considerar semáforos e veículos como agentes, comunicando entre de modo a regular a rede de trânsito com o objetivo de minimizar o tempo médio de paragem de um dado carro num semáforo. Considerando os mecanismos de simulação que o SUMO

oferece, foi possível dar uma maior atenção ao protocolo de troca de mensagens entre os semáforos. Apesar de termos sentido algumas dificuldades a implementar as mudanças de estado dos semáforos, sendo que estas mudanças eram decididas, pelo algoritmo QLearning, conseguimos apresentar um protocolo que demonstra resultados, como referido nas Experiências realizadas.

Desenvolvimento e dificuldades

Ao iniciar o desenvolvimento do projeto, reparamos que surgiram vários problemas. Um deles, o mais fulcral, foi o facto de tentar trabalhar em todas as fases do projeto na mesma altura, ou seja, aprender a utilizar as ferramentas utilizadas, testá-las e desenvolver o nosso próprio projeto. Além disso, existiu imensa dificuldade na comunicação com o SUMO, visto que algumas funções no TraSMAPI não estavam implementadas e, no próprio SUMO, certas funções, aquando da sua chamada, executavam e não alteravam certos parâmetros ou o resultado esperado não acontecia.

Melhoramentos

Olhando para o resultado final, é da nossa opinião que a estrutura do código se encontra satisfatória, apesar de o código não estar tão legível quanto desejado.

Embora certas melhorias de performance pudessem ser consideradas e, assim, reduzir o notório impacto na execução da GUI do SUMO, assumimos também que a falta de documentação do código poderia ser emendada.

Recursos

Bibliografia

- TraSMAPI Documentation - https://github.com/STEMS-group/trasmapi/wiki
- SUMO Documentation - http://sumo.dlr.de/wiki/SUMO_User_Documentation
- Apontamentos das aulas - https://paginas.fe.up.pt/~eol/AIAD/aiad1617.html
- JADE Documentation - http://jade.tilab.com/doc/api/index.html
- JADE, TraSMAPI and SUMO: A tool-chain for simulating traffic light control- https://arxiv.org/pdf/1601.08154v1.pdf
- Reinforcement Learning-based Control of Traffic Lights in Non-stationary Environments: A Case Study in a Microscopic Simulator - http://ceur-ws.org/Vol-223/53.pdf
- Github of Q-Learning Implemented- https://github.com/abethcrane/Q-Learning-Traffic-Lights
- Q-Learning Algorithm - https://en.wikipedia.org/wiki/Q-learning
- Q-Learning Tutorial - http://mnemstudio.org/path-finding-q-learning-tutorial.htm
- JADE Tutorial - http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf

Software

- IntellIJ (IDE) - https://www.jetbrains.com/idea/
- Github - https://github.com/
- TraSMAPI - https://github.com/STEMS-group/TraSMAPI
- JADE - http://jade.tilab.com/
- SUMO - http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/

Elementos do Grupo

Cada membro do grupo trabalhou aproximadamente de forma igual, não havendo distinções a serem indicadas. Houve uma boa comunicação dentro do grupo, de forma a que cada elemento soubesse o que estava a ser feito.