Sistema Multiagente de Controlo Inteligente de Trânsito

Universidade do Minho, 4710-057 Braga, Portugal,
Mestrado em Engenharia Informática, Escola de Engenharia

Abstract. Este trabalho apresenta o desenvolvimento de um sistema multiagente hierárquico para controle inteligente de tráfego urbano. A arquitetura proposta distribui a tomada de decisão em três níveis principais: Gestor Regional, Gestor de Cruzamento e agentes especializados (Ambiente CityFlow e Controlador RL). O sistema utiliza o simulador CityFlow como base e implementa algoritmos de aprendizagem por reforço (Reinforcement Learning) para otimizar o controle de semáforos em cruzamentos urbanos complexos. A comunicação entre agentes é realizada através do protocolo XMPP usando o framework SPADE, com mensagens assíncronas e performatives específicas. Os testes realizados demonstraram uma redução de aproximadamente 50% no tempo total de espera dos veículos quando comparado ao controlador padrão, evidenciando o potencial da abordagem. O trabalho também discute possíveis melhorias futuras, como modelagem de múltiplos modos de transporte e desenvolvimento de interfaces visuais para monitorização em tempo real, além de considerações éticas relevantes para implementações reais do sistema.

Palavras-chave Agentes Inteligentes, Sistemas Multiagente, Controlo do Tráfego, Ruas Inteligentes, CityFlow, Reinforcement Learning

Guilherme Rio,
PG57875

César Cardoso,
PG57870

Gonçalo Brandão,
PG57874

LingYun Zhu,
PG57885

1 Proposta Design

Baseando nos no Trabalho de Investigação Realizado Anterirormento o grupo decidui que queria realizar um sistema multiagente hierarquizado semelhante ao do artigo "A Smart Surveillance System of Distributed Smart Multi Cameras Modelled as Agents" [7] que propõe um sistema sistema distribuído e hierárquico de agentes, onde cada nível processa e sintetiza informação de forma incremental, optimizando assim a largura de banda e evitando possíveis *Bottlenecks*. No nível mais baixo, os agentes analisam o volume bruto dos dados, reduzindo-os a informação mais simples para transpor aos níveis superiores. Os níveis Intermédios agregam informação das múltiplas fontes e/ou tomam decisões regionais e por fim informam o nível mais superior que irá realizar ações semelhantes mas num*scope* maior. Achamos esta arquitetura bastante intuitiva, escalável, implementável e testável, pontos cruciais para a nossa implementação.

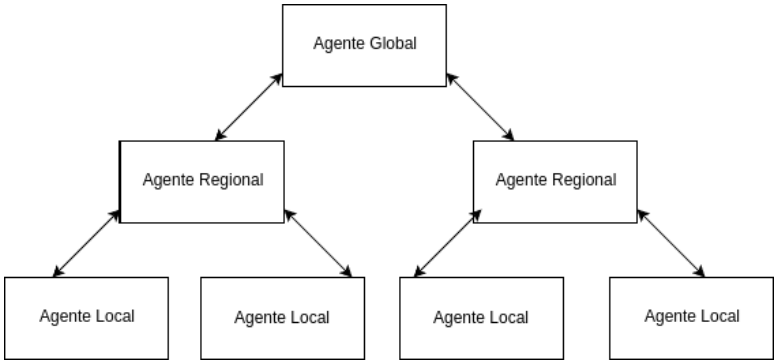

Fig. 1. Arquitetura Pretendida

Com uma base pré definida procuramos ferramentas para nos auxiliar, sendo que estávamos à procura de algo que pudesse ser caracterizado como nível local, com base nos artigos "Definition of a Smart Street as Smart City's Building Element" [5] ou o "A Smart Surveillance System of Distributed Smart Multi Cameras Modelled as Agents" [7] como vimos anteriormente. Também estávamos interessados em trabalhos de otimização de transito e/ou transportes públicos, após a nossa analise do artigo "Dynamic Public Transport in Smart City using Multi-agent system" [6], portanto procuramos por simulações de transito ou *digital twins de cidades*, como o ultimo requeria muito poder computacional, decidimos seguir a direção de simulações de transito, encontrandos algumas ferramentas como Matsim [8], Mars [9], CityFlow [10], Sumo [11], PTV Vissim [13], Flow [14], OpenTrafficSim [15], MOSS [16].

2 CityFlow

Após a analise das varias ferramentas decidimos realizar o nosso trabalho com base o CityFlow[10], uma simulação de cruzamentos complexos, por causa da sua documentação, *docker image* e *base code* publico, uma boa performance e leveza computacional. Também decidimos que era interessante implementar-mos modelos de coordenação com base em modelos de *reinforcement learning* para lidar mos a alteração da cor dos semáforos de forma automática.

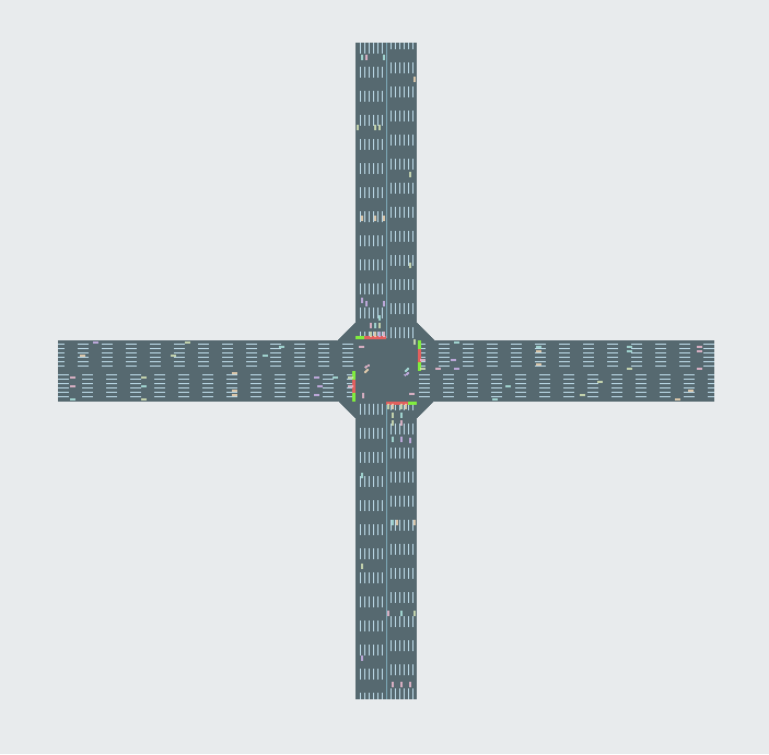

Fig. 2. Exemplo Simples CityFlow

O CityFlow também nos permite uma escalabilidade de complexidade, podendo analisar cruzamentos simples como na imagem em cima ou modelos mais complexos com *grid* de 2 por 2.

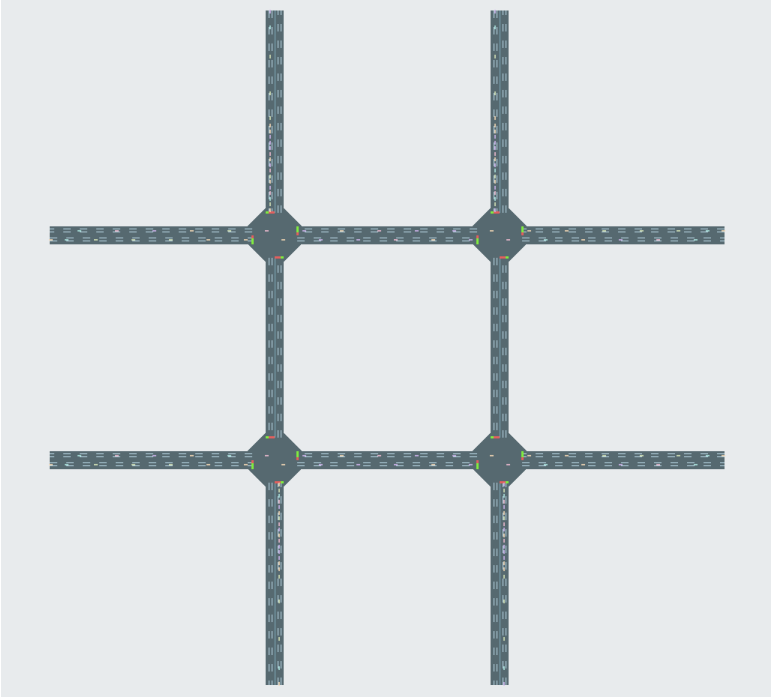

Fig. 3. Exemplo *grid* 2 x 2

O CityFlow também era bastante adaptável à arquitetura imaginada anteriormente. Pois poderíamos lidar com cada cruzamento com um agente local especifico e com um conjunto de cruzamentos com um agente regional, evoluindo o projeto desta maneira, aumento o numero de cruzamentos e regiões em analise.

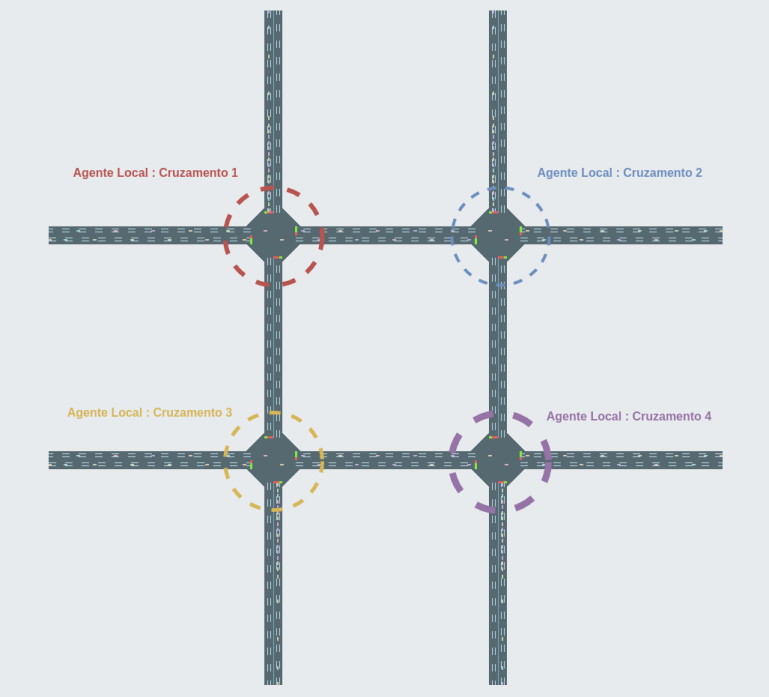

Fig. 4. Representação da Implementação de um Sistema Multiagente numa *grid* 2 x 2

Também será importante compreender a complexidade dos cruzamentos a serem analisados, que são poucos comuns existirem em ambientes reais, ja que sao propricios a acidentes, transito e dificuldade de escuamento natural do transito

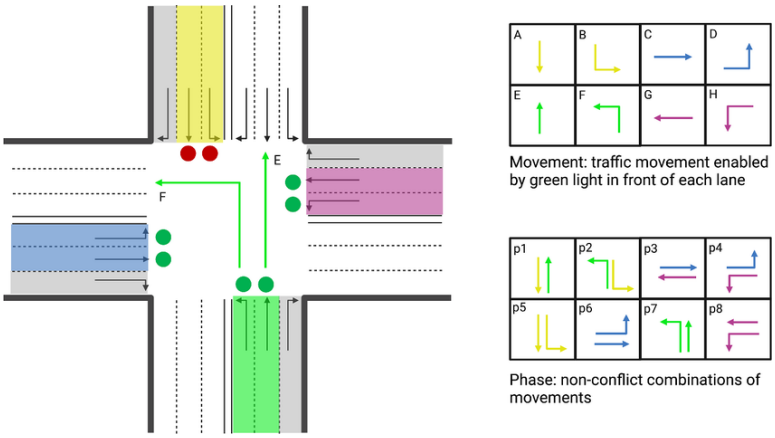

Fig. 5. Complexidade de um Cruzamento

2.1 Integração do CityFlow com Aprendizagem por Reforço Profundo

Este trabalho utiliza o *CityFlow*, um simulador de tráfego rodoviário de alto desempenho, em conjunto com uma rede de Aprendizagem por Reforço Profundo (Deep Reinforcement Learning) para optimizar o controlo dos semáforos em cruzamentos urbanos.

A classe *CityFlowEnv* implementa uma interface compatível com o *OpenAI Gym*, permitindo a utilização de algoritmos padrão de aprendizagem por reforço. O ambiente simula o tráfego urbano, onde cada acção corresponde à selecção de uma fase de semáforo para cada intersecção, e a observação consiste no número de veículos em espera nas vias de entrada. A recompensa é definida como o valor negativo do total de veículos em fila de espera, acrescido de uma penalização adicional para veículos parados dentro da zona de intersecção, promovendo assim o escoamento eficiente do tráfego.

Durante a fase de treino, é utilizado o algoritmo *PPO* (Proximal Policy Optimization), da biblioteca *Stable-Baselines3*, com uma política do tipo *MlpPolicy*, baseada em redes neuronais totalmente conectadas. A interacção decorre da seguinte forma:

1. A simulação é executada durante *MAX_STEPS* passos por episódio.
2. A rede neuronal observa o estado actual (número de carros em cada via) e escolhe uma acção (configuração dos semáforos).
3. O ambiente devolve uma nova observação, a recompensa e a indicação de fim do episódio.
4. O agente actualiza a sua política com base nas recompensas obtidas, aprendendo a minimizar o tempo de espera total e a evitar bloqueios nas intersecções.

No final, a política treinada é aplicada para controlar os semáforos numa nova simulação, sendo comparada com um cenário baseline, onde não é aplicada qualquer optimização. Esta comparação permite avaliar os ganhos obtidos com a utilização da aprendizagem por reforço.

Este modelo demonstra como os sistemas de transporte inteligentes podem beneficiar de técnicas modernas de inteligência artificial para melhorar a fluidez do tráfego e reduzir congestionamentos em ambientes urbanos complexos.

3 Sistema Implementado

3.1 Tipos de Agentes

– **Gestor Regional** : O Gestor Regional é um agente central na arquitetura multiagente do sistema, será o agente responsável por gerir uma região, isto é um conjunto de cruzamentos como na Figura 4. Para tal acontecer o gestor regional tem a responsabilidade de dividir os vários cruzamentos em cruzamentos simples e transpor a responsabilidade dos mesmos para os agentes Gestores de Cruzamentos. Também é responsável por receber e armazenar os relatórios de atividade dos gestores de rua e tomar decisões do que devem fazer.

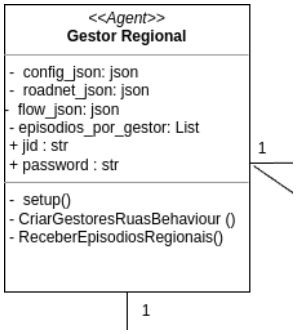

Fig. 6. Classe Gestor Regional

– **Gestor de Cruzamento** : O gestor de cruzamentos está responsável de lidar com um cruzamento único, estes enviam os ficheiros para a simulação de um cruzamento para um ambiente cityflow e devem receber um report completo do que aconteceu no final da simulação

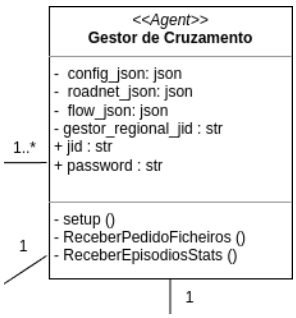

Fig. 7. Classe Gestor de Cruzamento

– **Ambiente CityFlow** : Os Ambientes Cityflow realizam uma simulação de um cruzamento e do seu tráfego, estes devem implementar as sugestões do modelo de Reinforcement Learning pré treinado e reportar ao agentes gestor do cruzamento o que aconteceu, ou está a acontecer no mesmo.

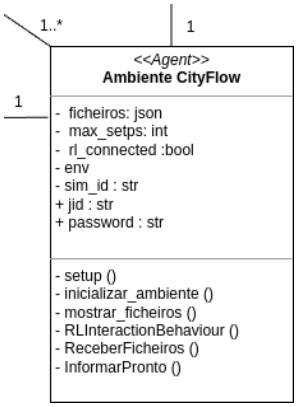

Fig. 8. Classe Ambiente CityFlow

– **Controlador RL** : O Controlador RL possui um modelo de Reinforcement Learning pre treinado, de modo a tomar decisões quando questionado pelo ambiente cityflow do que fazer.

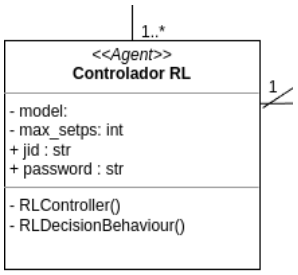

Fig. 9. Classe Controlador RL

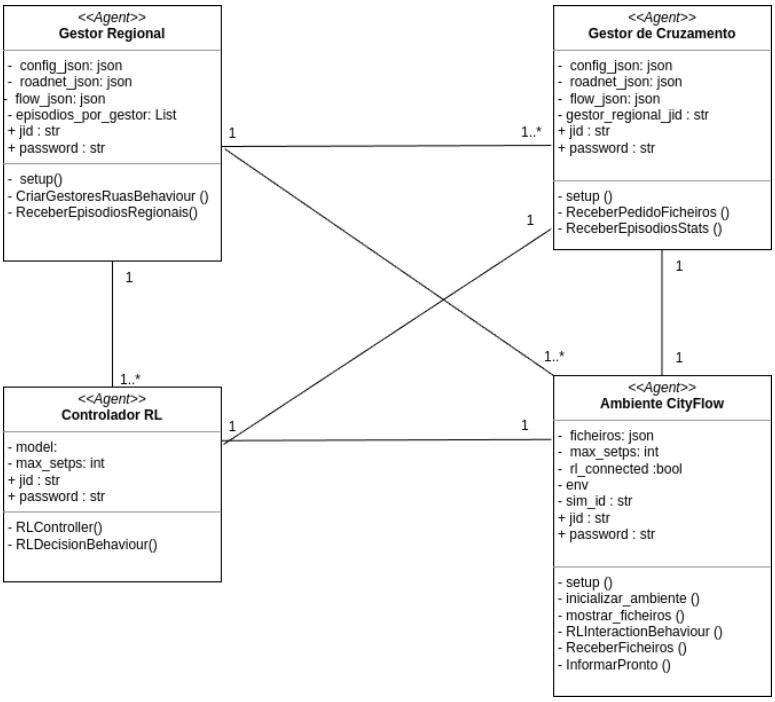

Fig. 10. Diagrama de Classes

Analisando o diagramas de classes percebemos que um Gestor Regional tem relacação de 1 para muitos, enquantos os agentes que estão a lidar com um cruzamento tem relação de um para um.

Gestor de Região :

- **Behaviours** :
  - **CriarGestoresRuasBehaviour (OneShotBehaviour)** : Este behaviour tem a função de iniciar todos os agentes de um nível hierárquico inferior (Gestores de Cruzamento, Ambientes CityFlow, Controladores RL) a partir dos ficheiros de configuração.
  - **ReceberEpisodiosRegionais (CyclicBehaviour)** : Este behaviour é responsável por receber e armazenar os relatórios de funcionamento de cada simulação enviados pelo Gestor de Cruzamento.
- **Performatives** :
  - **episodio_stats** : Performative utilizada nas mensagens entre os vários Gestores de Cruzamento e o Gestor Regional, esta performative recebe as informações na forma da Classe Rua_Stats.

Gestor de Cruzamento :

- **Behaviours** :
  - **ReceberPedidoFicheiros (CyclicBehaviour)** : Behaviour cíclico que fica à espera de uma mensagem de confirmação do ambiente cityflow que vai realizar a simulação do cruzamento. Ao receber a confirmação o gestor de Cruzamento envia os ficheiros serializados ao agente correspondente para poder iniciar a simulação.
  - **ReceberEpisodioStats (CyclicBehaviour)** : Behaviour ciclico que recebe o relatório de funcionamento da simulação e encaminha o mesmo para o Gestor Regional, através da performative "episodio_stats".
- **Performatives** :
  - **ambiente_ready**: Performative que indica que o ambiente CityFlow se encontra disponível para receber os ficheiros de configuração e começar a simulação.
  - **envio_ficheiros**: Performative utilizada para enviar os ficheiros de configuração do Gestor de Cruzamento para o Ambiente CityFlow.
  - **episodio_stats**: Performative utilizada nas mensagens entre os vários Gestores de Cruzamento e o Gestor Regional, esta performative recebe as informações na forma da Classe Rua_Stats.

Ambiente CityFlow :

- **Behaviours** :
  - **InformarPronto (OneShotBehaviour)** : Este behaviour tem a função de notificar o Gestor de Cruzamento que o ambiente está pronto para receber os ficheiros de configuração e iniciar a simulação, através da performative "ambiente_ready"-
  - **ReceberFicheiros (CyclicBehaviour)** : Behaviour ciclico que fica à espera dos ficheiros de configuração que serão enviados pelo Gestor de Cruzamento, através da performance "envio_ficheiros". Após receber os ficheiros, o ambiente de simulação é iniciado e o controlador RL informado com a perfomative "ambiente_ready" que o agente Ambiente Cityflow está pronto.
  - **RLInteractionBehaviour (CyclicBehaviour)** : Este Behaviour ciclico lida com a interação entre o controlador RL e o ambiente cityflow durante a simulação. Após receber a performative "controlador_ready" os dois agentes vão trocando informações para realizar a simulação, o behaviour envia a perfomative "obs" com os dados atuais da simulação e recebe a performative "action", para tomar as decisões do proximo passo. No final do episódio simulado, através da performative "episodio_stats" envia um relatório para o Gestor de Cruzamento.
  - **FixedControllerBehaviour (CyclicBehaviour)** : Caso a flag de fazer uma simulação base, isto é, sem as tomadas de decisão serem realizadas pelo controlador RL mas sim de forma estática, este comportamento é ativado, para fins de criar um relatório base para questões de comparação e estabelecimento de um grupo de controlo.
- **Performatives** :
  - **ambiente_ready**: Performative enviada ao Gestor de Rua e Controlador RL em momentos separados para informar os agentes que o ambiente se encontra pronto para interagir,
  - **envio_ficheiros** : Performative utilizada no sentido Gestor de Rua para o ambiente CityFlow de maneira a este receber os ficheiros de configuração e iniciar posteriormente uma simulação de um cruzamento.
  - **controlador_ready**: Performative enviada do Controlador RL para o Ambiente CityFlow para confirmar que o controlador se encontra pronto para começar a simulação.
  - **obs** : Performative enviada do Ambiente CityFlow para o Controlador RL de forma a informá-lo do estado atual da simulação.

- **action** : Performative envida do Controlador RL para o Ambiente CityFlow de forma a informá-lo de que ação deve tomar no próximo passo da simulação.
- **episodio_stats**: Performative utilizada nas mensagens entre os vários Gestores de Cruzamento e o Ambiente CityFlow, estas performative envia as informações na forma da Classe Rua_Stats.

Controlador RL :

– **Behaviour** :
  - **RLController (CyclicBehaviour)** : Behaviour cíclico que fica à espera de uma mensagem do Ambiente Cityflow de "ambiente_ready" e devolve a performative "controlador_ready" informando assim o agente que o controlador RL está pronto para iniciar a simulação.
  - **RLDecisionBehaviour (CyclicBehaviour)** : Behaviour que tomas decisões as decisões de ação durante a simulação do CityFlow, este fica à espera da performative "obs", contendo a fase atual da simulação, toma uma decisão com o seu modelo e devolve com a performative "action" a próxima decisão da simulação.

– **Performatives** :
  - **ambiente_ready**: Performative recebida do Ambiente CityFlow para informar o agente que o ambiente se encontra pronto para interagir.
  - **controlador_ready**: Performative enviada do Controlador RL para o Ambiente CityFlow para confirmar que o controlador se encontra pronto para começar a simulação.
  - **obs** : Performative enviada do Ambiente CityFlow para o Controlador RL de forma a informá-lo do estado atual da simulação.
  - **action** : Performative envida do Controlador RL para o Ambiente CityFlow de forma a informá-lo de que ação deve tomar no próximo passo da simulação.

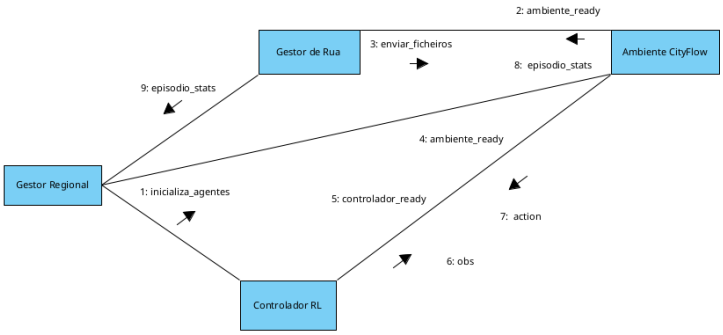

Fig. 11. Diagrama de Colaboração

3.2 Arquitetura

O sistema implementa uma arquitetura hierárquica distribuída, organizada em camadas especializadas que trabalham de forma coordenada. No topo da hierarquia, encontra-se o Gestor Regional , responsável por dividir uma região urbana complexa em cruzamentos gerenciáveis e coordenar os agentes subordinados. Cada cruzamento é administrado por um Gestor de Cruzamento , que atua como intermediário entre o Gestor Regional e dois agentes especializados: o Ambiente CityFlow , que encapsula a simulação do tráfego e o Controlador RL , que utiliza algoritmos de aprendizado por reforço para otimizar o controle de semáforos. Esta hierarquia implementada representa fielmente a hierarquia proposta no inicio do trabalho

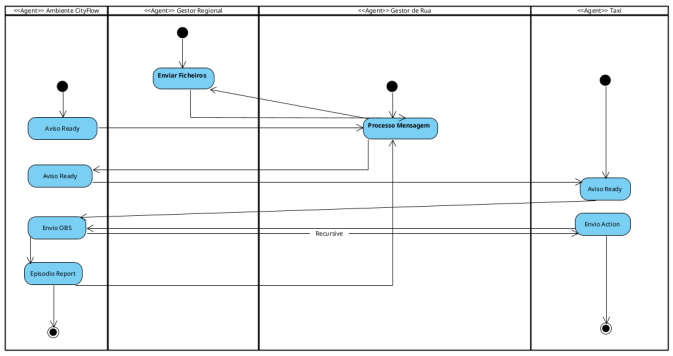

Fig. 12. Diagrama de Atividades

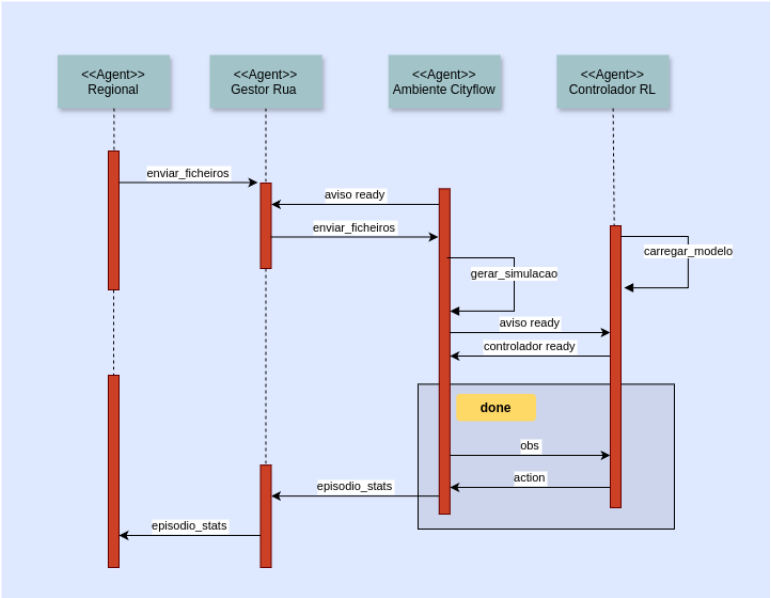

Fig. 13. Diagrama de Sequencia

3.3 Comunicação

A comunicação entre os diversos agentes do sistema é implementada através do protocolo XMPP, utilizando o framework SPADE, que proporciona um mecanismo de troca de mensagens assíncronas e robusto. Cada agente possui um identificador único (JID) no formato nome_agente@servidor, permitindo o endereçamento preciso das mensagens. O sistema emprega performatives específicas como "ambiente_ready", "episodio_stats", "action" e "obs" para indicar a intenção e o contexto da comunicação, enquanto o conteúdo das mensagens é serializado utilizando jsonpickle, possibilitando a transferência de estruturas de dados complexas como observações do ambiente, ações de controle e estatísticas de desempenho. A arquitetura de comunicação segue um padrão hierárquico, onde o Gestor Regional coordena a comunicação de alto nível, os Gestores de Cruzmaneto atuam como intermediários para seus respectivos cruzamentos, e os agentes especializados (Ambiente CityFlow e Controlador RL) trocam mensagens em tempo real durante a simulação, resultando em um sistema de comunicação flexível que suporta tanto interações verticais (entre níveis hierárquicos) quanto horizontais (entre agentes de mesmo nível).

3.4 Funcionalidades

O sistema desenvolvido combina simulação de tráfego com controlo inteligente de semáforos através de aprendizagem por reforço. As principais funcionalidades implementadas são as seguintes:

Integração com o Simulador CityFlow. A classe CityFlowEnv actua como um ambiente compatível com o OpenAI Gym, encapsulando o simulador CityFlow. Isto permite:

- Inicializar, controlar e reiniciar a simulação.
- Aceder a métricas detalhadas como número de veículos em espera, posições e velocidades.
- Aplicar acções correspondentes à escolha das fases dos semáforos em todas as intersecções.

Definição de Espaços de Acções e Observações

- O espaço de acções (action_space) é composto por uma acção discreta para cada intersecção, correspondente à fase de semáforo seleccionada.
- O espaço de observações (observation_space) consiste num vetor com os números de veículos em espera em cada estrada de entrada da rede.

Função de Recompensa Personalizada A função de recompensa foi desenhada para:

- Penalizar o número total de veículos em espera nas faixas.
- Penalizar adicionalmente os veículos parados perto das intersecções, incentivando o movimento contínuo e evitando bloqueios (gridlock).

Pré-processamento da Rede Durante a inicialização, o sistema:

- Carrega a configuração da rede e calcula o comprimento real de cada estrada, com base nas coordenadas dos pontos que a compõem.
- Garante que apenas as estradas relevantes são consideradas para observação e controlo.

Treino com Aprendizagem por Reforço Profundo

- O agente é treinado utilizando o algoritmo PPO, com uma política baseada em redes neuronais (MlpPolicy).
- O treino decorre ao longo de múltiplos episódios, com o agente a aprender a tomar decisões eficazes com base no estado do tráfego.

Comparação entre Simulação Base e Simulação Otimizada O código executa duas simulações:

- **Simulação baseline**: sem controlo inteligente, serve de referência.
- **Simulação RL**: controlada pelo modelo treinado, permite avaliar os ganhos de desempenho.

Geração de Registos de Simulação Durante a simulação, são gerados ficheiros de registo (replayLogFile, roadnetLogFile) que podem ser usados para visualização posterior do tráfego e das decisões do agente.

Extensibilidade e Modularidade A estrutura modular do código permite:

- Alterar facilmente a configuração da rede ou os parâmetros da função de recompensa.
- Substituir o algoritmo de aprendizagem, se necessário.
- Adaptar o sistema a diferentes topologias urbanas.

3.5 Sumários dos resultados obtidos e respetiva análise crítica

Os testes em pequena escala demonstraram um impacto significativo do modelo de aprendizagem por reforço no desempenho do sistema de controle de tráfego. Especificamente, o tempo total de espera de todos os veículos foi reduzido de **57 090 segundos** (com o controlador padrão do CityFlow) para **28 266 segundos** com o agente treinado. Esta redução de aproximadamente **50%** evidencia a eficácia do modelo PPO em ambientes mesmo com um único cruzamento, desde que haja um número razoável de faixas e veículos. No entanto, cada sessão de treino do modelo levou cerca de **2 horas**, o que impõe um limite prático ao número de iterações que podem ser feitas para ajuste fino. Com mais tempo de computação e hardware mais potente, seria possível realizar treinos mais longos e mais testes em rede, o que certamente levaria a uma melhoria ainda maior do modelo.

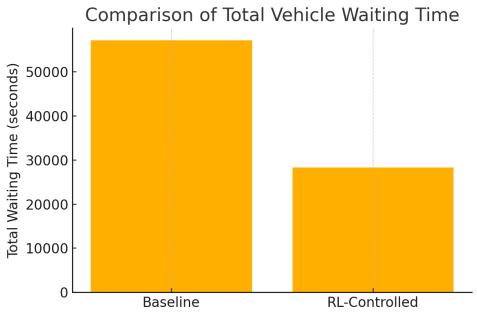

Fig. 14. Melhoria com o modelo de RL

4 Sugestões e Recomendações para Melhoria do Sistema

O sistema desenvolvido poderia ser expandido para cenários urbanos reais, por exemplo, através da implementação de interfaces com OpenStreetMap, permitindo a simulação de qualquer cidade real em vez de cruzamentos predefinidos. Este avanço deveria ser complementado pela adoção de um modelo de aprendizado por reforço hierárquico no Gestor Regional, sendo possível enviar sugetões contextuais aos gestores locais, como "priorizar fluxo norte-sul" ou "reduzir congestionamento no centro".

Outra abordagem seria, incorporar múltiplos modos de transporte, pedestres, ciclistas, transporte público, simulação de incidentes acidentes ou bloqueios. Tambem podiria existtir uma integração com sistemas externos como veículos conectados e dados meteorológicos. Estas seriam algumas melhorias e implementacões futuras para um trabalho deste gereno, também a implementacão de com interfaces de visuais como dashboards em tempo real e heatmaps de transito, transformariam a solução atual em uma plataforma abrangente para gerenciamento inteligente de tráfego urbano adaptável a contextos reais.

5 Ética no Contexto do Regulamento do Transio

Ao realizar trabalhos de análise e planejamento e regulamento do transito, é fundamental considerar diversos aspectos éticos. Primeiramente, deve-se garantir a transparência e a imparcialidade na recolha e análise de dados, evitando manipulações que possam favorecer determinados interesses. É essencial respeitar garantir que as comunidades locais não iram sofrer qualquer perde de qualiade de vida pelas decisões do sistema desenvolvido, favorecendo interesses alheios. A confidencialidade de dados sensíveis deve ser mantida, particularmente quando se trabalha com informações socioeconômicas. Além disso, é importante reconhecer e declarar possíveis conflitos de interesse que possam influenciar as análises e recomendações. Por fim, os projetos devem ser conduzidos com responsabilidade social, visando contribuir para um desenvolvimento regional sustentável e equitativo, que considere não apenas os aspectos econômicos, mas também os sociais e ambientais.

References

1. Wooldridge, M.: An Introduction to Multiagent Systems. John Wiley & Sons (2002)
2. d'Inverno, M. & Luck, M.: Understanding Agent Systems. Springer (2003)
3. Wooldridge, M. & Jennings, N.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 10(2), 115–152 (1995)
4. Kravari, K. & Bassiliades, N.: A Survey of Agent Platforms. Journal of Artificial Societies and Social Simulation, 18(1), 11 (2015)
5. Iordache, D. & Popescu, D. C.: Definition of a Smart Street as Smart City's Building Element. Procedia Engineering, 178, 210–221 (2017)
6. Horaždovský, P., Kozhevnikov, S., & Svítek, M.: Dynamic Public Transport in Smart City using Multi-agent System. In *Smart Cities Symposium Prague 2019*, IEEE, 2019.
7. Eigenraam, D. and Rothkrantz, L.J.M.: A Smart Surveillance System of Distributed Smart Multi Cameras Modelled as Agents. In: Smart Cities Symposium, Prague, IEEE (2016).
8. MATSim – Multi-Agent Transport Simulation. Disponível em: `https://matsim.org/`. Acesso em: 6 maio 2025.
9. MARS – Multi-Agent Research and Simulation. Introdução ao tutorial. Disponível em: `https://www.mars-group.org/docs/tutorial/intro/`. Acesso em: 6 maio 2025.
10. CityFlow – Large-Scale City Traffic Simulator. Disponível em: `https://cityflow-project.github.io/`. Acesso em: 6 maio 2025.
11. Eclipse SUMO – Simulation of Urban MObility. Disponível em: `https://eclipse.dev/sumo/`. Acesso em: 6 maio 2025.
12. H. Mei, X. Lei, L. Da, B. Shi, e H. Wei, "LibSignal: An Open Library for Traffic Signal Control," *Machine Learning*, vol. 113, pp. 1–37, Nov. 2023. DOI: 10.1007/s10994-023-06412-y.
13. PTV Group. *PTV Vissim – Multimodal Traffic Simulation Software*. Disponível em: `https://www.ptvgroup.com/en/products/ptv-vissim`. Acesso em: 6 maio 2025.
14. UC Berkeley – Mobile Sensing Lab. *Flow: Traffic Control Benchmarking Framework*. Disponível em: `https://flow-project.github.io/`. Acesso em: 6 maio 2025.
15. OpenTrafficSim. *OpenTrafficSim Manual*. Disponível em: `https://opentrafficsim.org/manual/`. Acesso em: 6 maio 2025.
16. FIBLAB – Tsinghua University. *MOSS: Mobility Simulation System*. Disponível em: `https://moss.fiblab.net/`. Acesso em: 6 maio 2025.