INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Otimização de Redes de Transporte Público
Aprendizagem Reforçada para uma Mobilidade Inteligente

António Luís Barros Mota

Mestrado em Ciência de Dados,

Orientadora:
Diana Elisabeta Diana Mendes (Doutora, Professora Associada)
ISCTE – Instituto Universitário de Lisboa

Setembro, 2024

BUSINESS SCHOOL

TECNOLOGIAS E ARQUITETURA

Otimização de Redes de Transporte Público
Aprendizagem Reforçada para uma Mobilidade Inteligente

António Luís Barros Mota

Mestrado em Ciência de Dados,

Orientadora:
Doutora Diana Elisabeta Diana Mendes (Professora Associada),
ISCTE – Instituto Universitário de Lisboa

Setembro, 2024

Agradecimentos

Este trabalho não poderia ter sido concluido sem a ajuda preciosa da Matilde Caldas, minha parceira de todos os dias. Embora de uma área científica muito diferente desta, soube ouvir-me com atenção, fazendo perguntas pertinentes que ao colocarem o trabalho em causa, o fortaleceram. O seu apoio moral e capacidade, inigualável, de me organizar as ideias só não me espantam porque sempre, sempre, estiveram lá. Agradeço-lhe ainda toda a disponibilidade para rever o texto. Agradeço também ao Professor Nuno R. Faria, do Departamento de Epidemologia e Doenças Infecciosas do *Imperial College of London*) pelo seu tempo e apoio na finalização e por me ajudar a elevar a fasquia deste trabalho. O meu agradecimento especial ao Federico Berto, da *Korea Advanced Institute of Science & Technology* (KAIST), pela generosidade que demonstrou ao apoiar-me nos assuntos mais técnicos com que me deparei. Por fim, o meu agradecimento à professora Diana Mendes, pela orientação positiva e construtiva ao longo de todo o processo, pelo entusiasmo que permitiu que partilhássemos durante conversas e debates que me orientaram na direcção certa.

Resumo

Desenhar uma “boa” rede pública de transportes é um problema complexo (NP-difícil) que envolve, entre outros, a escolha de paragens, dos “melhores” trajectos, a definição de horários e frequências, tendo em conta múltiplos factores em conflito. Dada a complexidade, a busca por soluções óptimas é preterida em favor do uso de métodos heurísticos (regras gerais de decisão) e recurso a conhecimento especializado, que permitem encontrar soluções satisfatórias. Esta dissertação foca-se na otimização da rede de transportes da Carris, em Lisboa, explorando a aplicação de mecanismos de aprendizagem reforçada (AR) para resolver uma parte deste problema: encontrar “melhores” trajectos entre várias paragens, servidas por um número variável de linhas – um problema de roteamento de veículos. Através do algoritmo *Multi-task Vehicle Routing Solver with Mixture-of-Experts*, foram treinados dois modelos. Os resultados foram comparados com os da rede Carris e com os do algoritmo de economias de Clarke e Wright. A AR demonstra potencial para “aprender” boas heurísticas e encontrar melhores soluções que as atuais, tendo o modelo minimizado a distância em linha recta do menor troço da rede. No entanto, a complexidade da mobilidade urbana é ainda um desafio, tendo sido necessário efectuar simplificações para modelar este problema. Apesar das limitações, tais como os baixos recursos computacionais e a natureza estática dos dados, esta análise demonstra que através da integração da informação de trânsito e do desenvolvimento de algoritmos mais abrangentes, a AR tem o potencial de melhorar a eficiência destas redes e construir soluções que se ajustem dinamicamente aos diferentes constrangimentos.

Palavras-chave

Transit Route Network Design and Frequency Setting Problem

Network Design and Frequency Setting Problem; VRP – Problema do Roteamento de Veículos aprendizagem reforçada; otimização combinatória; grafo;

3 a 6 palavrsas chave e 2 códigos de classificação retirados do JEL Classification System (https://www.aeaweb.org/econlit/jelCodes.php).

Abstract

Designing an efficient public transport network is a complex problem (NP-hard) that involves, among other factors, selecting stops, determining the most optimal routes, and defining schedules and frequencies, all while considering multiple conflicting factors. Given this complexity, the pursuit of optimal solutions is often set aside in favor of heuristic methods (general decision rules) and expert knowledge, which allow for the identification of satisfactory solutions. This dissertation focuses on optimizing Lisbon's Carris public transport network by exploring the application of reinforcement learning (RL) mechanisms to address part of this problem: finding more optimal routes between several stops served by a variable number of lines – a vehicle routing problem. Two models were trained using the Multi-task Vehicle Routing Solver with Mixture-of-Experts algorithm. The results were compared with the Carris network and the results given by the Clarke and Wright Savings algorithm. RL shows potential in learning good heuristics and finding better solutions than the current ones, as the model minimized the straight-line distance of the shortest segment of the network. However, the complexity of urban mobility remains a challenge, requiring simplifications to model this problem effectively. Despite limitations such as low computational resources and the static nature of the data, this analysis demonstrates that by integrating traffic information and developing more comprehensive algorithms, RL can improve the efficiency of these networks and create solutions that dynamically adjust to constraints.

Keywords

3 a 6 palavrsas chave e 2 códigos de classificação retirados do JEL Classification System (https://www.aeaweb.org/econlit/jelCodes.php).

Índice

Índice de Figuras

Glossário

Glossário de siglas

AR – Aprendizagem Reforçadaa

CML – Câmara Municipal de Lisboa

CNN – Rede Neuronal Convolucional

CVRP – Problema de Roteamento de Veículos com Capacidade

DQN – *Deep Q-Network*

GAT – Rede de Atenção em Grafos

IAG - Inteligência Artificial Geral

INE – Instituto Nacional de Estatística

MDP – Processo de Decisão de Markov

MVMoE – *Multi-task Vehicle Routing Solver with Mixture-of-Experts*

OP – Problema de Orientação

PCTSP – Problema do Caixeiro Viajante com recolha de prémios

RMSE – Raiz do Erros Quadrático Médio

RNN – Rede Neuronal Recorrente

SVM – Máquinas de Vetores de Suporte

TNDFSP – *Transit Route Network Design and Frequency Setting Problem*

TSP – Problema do Caixeiro Viajante

VRP – Problema de Roteamento de Veículos

VRPPD – Problema de Roteamento de Veículos com entregas e recolhas

VRPTW – Problema de Roteamento de Veículos com Janelas Temporais

Introdução

O desenvolvimento de uma Inteligência Artificial Geral (IAG)[1] é o objetivo último dos estudos em torno da aprendizagem automática e tópico de discussão pelo menos desde meados do século passado[2]. Hoje em dia, o desenvolvimento cada vez mais rápido da tecnologia, a crescente capacidade computacional (velocidade, memória, infraestruturas) e a criação de algoritmos complexos que mimetizam certos aspetos da biologia, mas também – tem que se admitir – algum furor em torno da tecnologia[3] e das suas capacidades têm vindo a reforçar os argumentos que consideram o cérebro humano não mais do que uma máquina complexa, onde sentimentos, emoções e consciência (e moral!, e cultura!) são o subproduto de milhões de interações entre neurónios, sendo, desta forma, características emergentes passíveis de se revelar, também, através da interação entre milhões de "neurónios" artificiais[4].

A verificação (desde os anos 70 do século passado) de que comportamentos complexos podem emergir a partir de regras muito simples e limitadas (isto é, locais)[5] vem dar força a esta linha de raciocínio, já que é possível observar o aparecimento de propriedades não determinísticas (ou, pelo menos, que não conseguimos prever) emergindo da combinação de poucas regras determinísticas. Por outro lado, o grande desconhecimento que ainda existe relativamente aos múltiplos processos que formam a inteligência ou a criatividade e o facto de o funcionamento das redes neuronais artificiais ser muito diferente do cérebro humano[6] faz crer que se pode "estar apenas a criar um avião e não um pássaro" e que a tecnologia de que dispomos atualmente não será capaz de produzir verdadeira inteligência.

Enquanto a busca por uma IAG é tema de um intenso debate filosófico, ético e moral, económico e técnico, o estudo de partes do problema já foi com certeza iniciado e consubstancia-se no uso de ferramentas estatísticas e métodos de aprendizagem supervisionada (através de exemplos), não supervisionada (através de padrões estatísticos na informação) e reforçada (através de tentativa e erro)

[1] Uma inteligência ao nível humano, capaz de realizar múltiplas tarefas claramente distintas.

[2] Veja-se, por exemplo, o famoso artigo "*Computing Machinery and Intelligence*", de Alan Turing, publicado em 1950 e no qual o autor desenvolve o "jogo da imitação" que permitiria determinar se uma máquina pensa ou não.

[3] (Eric Rosenbaum, 2023).

[4] David Deutsch é um dos pensadores a defender esta tese: "[Douglas Hofstadter e Lady Lovelace] partilham a premissa errada de que os passos computacionais de nível inferior não podem de maneira nenhuma ascender a um «eu» de nível mais elevado que afecta tudo" (Deutsch, 2013, p. 208)

[5] Note-se, por exemplo, o comportamento dos 'agentes' no 'jogo da vida' de John Conway ou o desenvolvimento das estruturas fractais.

[6] Repare-se na (muito) maior eficiência do cérebro humano por comparação aos gastos atuais de energia para o treino dos grandes modelos de linguagem ou, ainda, no facto destes modelos necessitarem de enormes quantidades de dados para serem treinados.

para a modelação de diferentes redes neuronais profundas (*deep neural networks*)[7], pretendendo-se mimetizar diferentes aspetos da inteligência e aprendizagem humana. Num exemplo já clássico, através de uma amostra (catalogada) de milhares de imagens de algarismos desenhados de formas diferentes à mão, redes neuronais profundas "aprenderam" a identificar novos exemplos de algarismos desenhados que nunca antes tinham visto; ou, quanto a um tópico mais próximo do tema desta tese, redes neuronais conseguiram aprender a jogar o jogo do Go (com um espaço de configurações ou estados possíveis enorme, muito maior que o do xadrez) vencendo, em 2016, um dos melhores jogadores da época.

As várias arquiteturas desenvolvidas tiveram impacto em domínios tão variados como a classificação de imagens, a deteção de objectos, o reconhecimento de discurso, a tradução de textos, a geração de discursos e imagens, e, também, resultados promissores em áreas como o diagnóstico médico, o estudo de novos produtos farmacêuticos, a condução autónoma, o combate às alterações climáticas, entre muitos outros.

No entanto, apesar de enormes progressos e de haver quem considere que há formas de aprendizagem (por exemplo, através de recompensas) capazes de generalizar as restantes, substituindo-as[8], não só poderá considerar-se que o tipo de aprendizagem que se pretende mimetizar é limitado e não abarca toda a complexidade da aprendizagem humana, como as dificuldades técnicas[9], sociais[10] e de implementação destes algoritmos fazem prever que demorará muito tempo até (poder) haver implementações generalizadas destas ferramentas (substituindo tarefas repetitivas nos locais de trabalho, realizando atividades artísticas, tomando decisões profissionais, jurídicas ou políticas sem qualquer intervenção humana, apenas para dar alguns exemplos).

---

[7] Em particular, várias arquiteturas de redes neuronais, tais como as *Feedforward NN* (FNN), as Redes Neuronais Recorrentes (*Recurrent NN* – RNN) ou as Redes Neuronais Convolucionais (*Convolutional NN* – CNN). Mais recentemente, o *Tranformer*, uma arquitetura baseada em mecanismos de atenção, tem vindo a merecer atenção mediática e, sendo mais versátil (pois é capaz de capturar as dependências existentes a diferentes distâncias em informação sequencial) trouxe progressos e alguma unificação das diferentes arquiteturas.

[8] "*we consider an alternative hypothesis: that the generic objective of maximising reward is enough to drive behaviour* that *exhibits most if not all abilities that are studied in natural and artificial intelligence"* (Silver et al., 2021, p. 1)

[9] Wagstaff, (2010) discorre sobre a dificuldade existente em implementar, fora do setor académico, as técnicas e soluções de aprendizagem automática. Neste artigo considera-se que muita da investigação é ainda puramente académica, feita isoladamente (sem contacto com problemas reais) e focando-se muitas vezes apenas em problemas 'teóricos' ou usando bases de dados 'tipo'. Reflexo disso é o uso frequente de métricas generalistas (RMSE, por exemplo) para avaliar os modelos testados, não relacionadas com o problema concreto que se pretende estudar.

[10] "*Artificial Intelligence (AI) entails a number of potential risks, such as opaque decision-making, gender-based or other* kinds *of discrimination, intrusion in our private lives or being used for criminal purposes."*(Nikolinakos, 2023, p.1)

A discussão encetada em Wagstaff, (2010) é particularmente relevante quando considerada no âmbito da aprendizagem reforçada (*Reinforcement Learning*, AR) – método da aprendizagem automática no qual esta tese se vai focar. Os mecanismos desenvolvidos no âmbito dos modelos de aprendizagem reforçada oferecem dificuldades acrescidas quanto à sua avaliação e implementação (*deployment*), não só no que concerne às questões técnicas próprias desta classe de algoritmos, como também às relacionadas com questões de segurança aquando da interação destes algoritmos com o "mundo real" (Whittlestone et al., 2021). Isto deve-se, em última análise, ao facto de estes serem algoritmos que têm no seu core a tomada de decisão, isto é, algoritmos desenhados para escolher de entre um conjunto de ações possíveis e agir sobre o ambiente para atingir um determinado objetivo mais ou menos complicado. Isto pressupõe, por exemplo, que é possível resumir um objetivo humano a um número, um valor que se pretende maximizar (Silver et al., 2021). Esta é ainda uma questão em aberto, visto que não é fácil traduzir matematicamente, de forma eficaz, desejos ou vontades humanas. Assim, mais do que noutros casos, deve ser dada atenção redobrada às questões de avaliação em teste e segurança da fase de implementação.

Não obstante todas as questões levantadas até agora, a tecnologia tem enormes potencialidades (mesmo quando pensada apenas para aplicações locais). Focado inicialmente na compreensão de imagens, texto e som e em processos tais como a tradução, o campo da aprendizagem automática tem mudado o seu foco, de forma a considerar ideias sobre como tratar a imensidão de dados existentes de uma forma mais significativa, isto é, sobre como transitar da descrição, previsão e geração de dados, para a prescrição de orientações, situação onde a aprendizagem reforçada desempenha um papel muito significativo.

As enormes capacidades instaladas podem auxiliar o ser humano na compreensão de muitos fenómenos, ou no auxílio para a resolução de problemas "pequenos" ou de âmbito mais limitado, tal como é o caso daquele sobre o qual nos debruçaremos nesta dissertação.

Enquadramento e Motivação

O transporte de pessoas e mercadorias é uma realidade cada vez mais presente em todas as escalas temporais e espaciais (Brockmann et al., 2006). O movimento de pessoas e bens tem vindo a aumentar consistentemente nos últimos 20 anos, tendo-se vindo a alterar, também a várias escalas, os padrões de mobilidade (Gkiotsalitis, 2023).

Considerando este cenário, o desenho de "melhores" redes de transporte – e, em particular, de redes de transporte público – tem um impacto cada vez maior, pois quanto mais eficiente (relativamente a algum

fator) for esta rede, mais pessoas poderá servir[11] ou mais mercadorias pode distribuir com menos meios de transporte ou menos quilómetros percorridos. O desenvolvimento destas redes tem, assim, impacto ambiental (redução das emissões de carbono, adoção de combustíveis alternativos), económico (redução dos custos do transporte) e social (diminuição do tempo de viagem, a redução do tráfego e dos acidentes), entre outros.

O aumento das necessidades de mobilidade tem sido necessariamente acompanhado pelo desenvolvimento de infraestruturas de transporte – estradas, portos e paragens de autocarro, mas também, horários, calendários, gestão da frota e vários outros elementos. Dada a utilidade económica e benefícios sociais de tais infraestruturas (Pereira et al., 2022), bem como a dimensão e complexidade do seu *design*[12] (Kepaptsoglou & Karlaftis, 2009), estas redes de transporte têm de ser planeadas.

No entanto, este planeamento não é trivial (Wu et al., 2018). Não só há interesses em conflito – quem faz o transporte quer diminuir os custos, quem o usa quer mais rapidez –, como o tipo de problema que é necessário resolver para encontrar uma boa rede de transportes, muda rapidamente: a digitalização da sociedade moderna, a introdução de sensores em veículos e o desenvolvimento de veículos autónomos, a criação de serviços de partilha e de *leasing*, a massificação dos veículos elétricos e desenvolvimento da infraestrutura de suporte a esses veículos ou a integração dos diferentes modos de mobilidade pública, por exemplo, têm impacto na forma como se pretende desenhar uma rede de transportes pública (Gkiotsalitis, 2023).

Note-se ainda que no desenho de tais redes deve também ter-se em conta as futuras necessidades previstas, pois não há só fatores de mudança a curto prazo, mas tendências de médio e longo prazo: por exemplo, estudos recentes (INE, 2023) indicam uma mudança na mobilidade geral de pessoas, que é hoje em dia mais intermunicipal do que intra-municipal (anteriormente, a principal dinâmica de mudança era dentro da própria freguesia); no caso particular de Lisboa, estes estudos referem um alargamento territorial das deslocações (fruto de maiores deslocações para o trabalho), numa cidade menos polarizadora, tendo o movimento de pessoas deixado de ser tão radial, o que implica uma maior complexidade nos movimentos interurbanos dentro desta área metropolitana.

No que concerne o movimento de pessoas e bens dentro de uma cidade (ou de uma zona mais alargada, tal como a Área Metropolitana de Lisboa), os autocarros têm um papel determinante no desenvolvimento de tais redes, dada a sua maior flexibilidade quando comparada com a dos veículos sobre

[11] O tempo de viagem, o número de transbordos e a falta de ligações directas são algumas das razões apontadas pelos utilizadores de carro para não usarem mais o transporte público (CML, 2021).

[12] Muitos utentes, muitos operadores, muitos objetivos interconectados e contraditórios.

carris. Esta flexibilidade permite acautelar não só as mudanças sazonais (de médio prazo, mais previsíveis), mas também as de curto e longo prazo, que ocorrem na distribuição do movimento populacional.

Problema de Pesquisa e Objetivo

O facto de ser complexo adequar uma rede de transporte às necessidades existentes em cada momento e de estas necessidades de mobilidade estarem em constante mudança[13], torna este um tipo de problema ideal para ser abordado pela aprendizagem automática, visto que existe o potencial destes algoritmos virem a revelar métodos mais rápidos e eficazes que os atualmente existentes.

Esta tese focar-se-á no uso de aprendizagem reforçada para modelar um problema de mobilidade denominado Problema de Roteamento de Veículos (*Vehicle Routing Problem* – VRP), que é um problema da área da logística que envolve determinar a melhor rota possível de um conjunto de veículos para a entrega de bens ou prestação de um serviço, ao mesmo tempo minimizando ou maximizando um determinado fator, com o objectivo de perceber se o uso destes algoritmos é adequado para auxiliar na procura de soluções para esta classe de problemas.

Em particular, abordar-se-á o VRP ao nível de um município[14], através de dados disponíveis sobre a rede de transportes públicos de Lisboa (mais concretamente, a rede de autocarros Carris), tentando avaliar a rede existente e compará-la com a solução encontrada recorrendo a modelos de aprendizagem automática.

Âmbito e Relevância da Pesquisa

A busca por redes de transporte mais eficientes é um problema de otimização combinatória, que é um sub-ramo da otimização matemática[15] e que envolve encontrar uma solução ótima (que maximize ou minimize um determinado objetivo, ou "função objectivo") de entre um conjunto discreto de possíveis

[13] Muitos fatores, como o fenómeno da mobilidade partilhada, o teletrabalho ou a movimentação de pessoas para fora dos centros urbanos (a acontecer no caso de Lisboa, onde tradicionalmente se concentram os trabalhos), contribuíram e contribuem para as alterações nos padrões de mobilidade, gerando novos problemas que têm de ser resolvidos, diariamente, se se pretende ter uma rede de transportes (e, mais geralmente, de distribuição de bens e pessoas) mais eficiente.

[14] Com características próprias relativamente a um VRP para problemas com distâncias maiores (por exemplo, um VRP entre cidades.

[15] "A otimização (matemática), também conhecida por programação matemática, é uma área da matemática que (…) diz respeito [à] caracterização e procura de minimizantes, ou maximizantes, de uma certa função num certo conjunto, geralmente definido por equações e inequações algébricas" (Soares, 2005, p.1). No caso mais geral dos problemas de otimização, a função que se pretende optimizar assume valores no conjunto dos números reais e o conjunto onde a função está definida é contínuo.

soluções (*feasible solutions*) – a otimização combinatória corresponde então à otimização matemática considerada em conjuntos finitos. Esta área do conhecimento teve um rápido desenvolvimento desde os anos 50, motivado, entre outros fatores, pelo desenvolvimento e crescimento de redes de transporte públicas. Este desenvolvimento permitiu a criação de algoritmos capazes de resolver este tipo de problemas, no sentido em que permitam encontrar soluções exatas ou aproximadas para os mesmos. Tradicionalmente, os algoritmos existentes apenas conseguem uma solução exata quando o problema é "pequeno", pois o espaço de soluções deste tipo de problemas cresce exponencialmente com o aumento do número de pontos – são NP-difíceis (Bengio et al., 2021) – o que torna impossível o processo de encontrar soluções exatas.

É hipótese deste trabalho que a aprendizagem automática, nomeadamente os métodos de aprendizagem reforçada (AR), podem ser aplicados para resolver o problema que pretendemos, permitindo explorar e descobrir novas heurísticas de decisão na escolha de "melhores" caminhos para uma instância particular de um problema de otimização combinatória, sem qualquer intervenção de especialistas no tema.

É no contexto de problemas em aberto para os quais é preciso encontrar resposta diariamente, que o desenvolvimento de algoritmos "mais abrangentes", com recurso a técnicas de aprendizagem automática, se torna relevante, pois têm o potencial de responder a uma maior diversidade de problemas, ou, mais especificamente, a instâncias novas do mesmo problema, e também de encontrar novos algoritmos que possam ser mais eficazes na procura de soluções mais próximas da solução ótima – ou melhores estratégias para encontrar melhores soluções.

Estrutura da Tese

Além desta introdução, apresentar-se-á em seguida o Estado da Arte, onde se faz uma análise geral do VRP e dos tipos de solução, bem como as aplicações de AR e de que forma este método tem sido usado para encontrar soluções para o problema em questão, justificando-se ainda o seu uso por oposição a outros métodos de aprendizagem; a Metodologia, onde se aborda a recolha dos dados e se formula e concretiza o problema, e, subsequentemente, se decide sobre que algoritmo específico de aprendizagem reforçada vai ser testado; a Configuração Experimental onde se concretizam o ambiente e o agente, e se definem os parâmetros e hiperparâmetros da experiência; os Resultados e a Discussão onde se avalia o modelo treinado quanto a métricas de avaliação, bem como quanto à sua capacidade para resolver um problema concreto e se reflete sobre todo o processo; e, finalmente a Conclusão.

Revisão da Literatura

Sendo a Ciência de Dados uma área fundamentalmente transdisciplinar, qualquer estado da arte deve incidir não só sobre o ponto em que se encontra a investigação acerca dos algoritmos e métodos analíticos a ser usados, como também sobre o campo específico que se quer estudar.

O facto de a Ciência de Dados ser uma área relativamente recente, que incorpora conhecimento vindo de áreas como a matemática, a probabilidade, a estatística e a computação, que se entrecruzam e influenciam[16], torna necessário sedimentar os conceitos de base, para que permitam explorações mais complexas sem nos perdermos no mar de informação.

Neste sentido, a leitura atenta e demorada de livros seminais sobre os diversos temas foi fundamental para se poder prosseguir, após estarem bem ancorados os conceitos base.

Refiram-se a este propósito os livros "Introdução à Probabilidade e à Estatística" (Pestana & Velosa, 2008) e "*The elements of Statistical Learning*" (Hastie et al., 2009), essenciais para uma revisão mais técnica dos princípios estatísticos que subjazem a toda a Ciência de Dados; em particular, métodos estatísticos, tais como a regressão, a classificação ou os processos de decisão de Markov e várias técnicas estatísticas, tais como regressão estatística, máquinas de vetores de suporte (*support vector machines* – SVM) ou a análise de componentes principais puderam aí ser explorados e aprofundados.

Para além destes, os livros "*Reinforcement Learning: An Introduction*" (Sutton & Barto, 2015), "*Deep Reinforcement Learning*" (Plaat, 2022) e "*Algorithms for Reinforcement Learning*" (Szepesvári, 2010), foram necessários para consolidar o formalismo matemático dos algoritmos de aprendizagem reforçada, nomeadamente quanto à definição rigorosa dos vários conceitos que lhe são próprios, como: política (*policy*), recompensa (*reward signal*) , função-valor (*value function*), ambiente (*environment*), entre muitos outros. Contribuíram também para o desenvolvimento da intuição necessária à compreensão da forma de funcionamento destes algoritmos, nomeadamente quanto à sua conceptualização matemática na forma de processos de decisão de Markov e sua resolução através da equação de Bellman.

Por fim, o artigo "*Human-level control through deep reinforcement learning*" (Mnih et al., 2015) – que marca um progresso importante na área da AR[17] – permitiu uma primeira viagem à forma de operacionalização das abordagens de aprendizagem reforçada profunda, isto é, sobre como é possível

[16] Muitas vezes, são os problemas levantados numa área de estudo e as perguntas que aí se fazem a determinarem o caminho da investigação feito noutra área de estudo (Kuhn & Hacking, 2012).

[17] Nomeadamente, a criação das primeiras *deep Q-networks*, que combinam redes neuronais profundas (*deep neural networks*) e AR e que permitiram, pela primeira vez, que o mesmo algoritmo (isto é, igual arquitetura e mesmos hiperparâmetros) fosse capaz de realizar várias tarefas distintas, nomeadamente, jogando vários jogos distintos de Atari (Mnih et al., 2015).

conjugar uma vasta quantidade de técnicas diferentes, em arquitecturas diferentes, para produzir resultados diferentes. Permitiu ainda uma introdução aos desafios técnicos que se colocam no desenvolvimento destes algoritmos e sua aplicação na prática. O artigo evidencia uma área de estudo com muitas potencialidades de desenvolvimento e desafios.

A primeira incursão à vasta bibliografia que existe sobre estes tópicos foi feita considerando que, apesar de recente, este campo de estudo tem merecido muita atenção nos últimos anos. Numa primeira análise ao número de artigos publicados, verificou-se que há milhares de artigos sobre AR – num total de 79.000 artigos publicados no Scopus desde 2014. Uma busca com as palavras "*reinforcement learning*" e "*combinatorial optimization*" devolveu 538 entradas. O foco foi assim colocado em encontrar artigos mais específicos. Fazendo a busca com as palavras "*transit network*" (que têm a ver com o problema que pretendemos estudar) e "*reinforcement learning*", foram encontrados 75 artigos desde 2014; destes, 11 foram considerados e analisados, visto que estavam mais próximos do tema de partida – o desenho de uma rede de transporte público. Foram também ponderados os artigos encontrados através da pesquisa com as palavras "*Transit Route Network Design Problem*", num total de 24.

É de realçar, no entanto, que mais do que uma pesquisa exaustiva e seleção mais ou menos aleatória de artigos para ler, e considerando que sobre o problema específico não foi encontrada muita literatura, o foco foi colocado na leitura em profundidade de alguns artigos considerados relevantes em otimização combinatória, aprendizagem reforçada e redes neuronais de grafos (*graph neural networks*), artigos esses que foram escolhidos tendo em conta algumas recomendações e orientações de especialistas no tema, o número de citações no Scopus, referência em documentos oficiais produzidos pela União Europeia e, acima de tudo, através de referências na bibliografia dos artigos lidos, numa leitura mais vertical que horizontal.

O Problema de Roteamento de Veículos

Tanto a otimização combinatória como a aprendizagem automática são áreas de estudo muito vastas, com vários problemas em aberto, o que se revela um desafio. Por um lado, muitos dos problemas abordados pela otimização combinatória são problemas complexos, NP-difíceis (*NP-hard*), para os quais se considera, em geral, não ser possível encontrar uma solução algébrica (exata), visto que os problemas aí abordados envolvem, muitas vezes, espaços de solução enormes, para os quais uma busca em árvore não é possível. Por outro lado, o desenvolvimento de métodos de aprendizagem automática para a resolução deste tipo de problemas ainda está nos seus primórdios, reportando-se a 2009 os primeiros desenvolvimentos de

redes neuronais com arquitecturas especificas para trabalhar com grafos e a 2017 o desenvolvimento de *Grapth Attention Networks*, que melhoram os modelos anteriores (Veličković et al., 2017)[18].

A otimização combinatória estuda problemas que têm algum tipo de estrutura combinatória, sendo a estrutura de grafo[19] um dos arquétipos da representação matemática (formal) deste tipo de problemas. Com efeito, muitos dos problemas da otimização combinatória lidam com conceitos que aparecem tipicamente quando se fala de grafos, tais como caminho, fluxo, direção, corte, entre outros. Assim, problemas em áreas científicas muito diferentes[20] – mas também muitos problemas "mundanos", que se colocam no dia-a-dia[21], são problemas de otimização combinatória e podem ser modelados por grafos (a cujos vértices e arestas são acrescentadas características ou dimensões) que generalizam bem a sua estrutura: "*Instances of the same type of problem are solved again and again on a regular basis, maintaining the same combinatorial structure, but differing mainly in their data*" (Dai et al., 2017, p.1). O estudo desta estrutura poderá, assim, ser frutuoso em diversos domínios, tal como aquele sobre o qual esta tese se debruça.

A construção de uma infraestrutura pública de serviços de transporte, eficaz e eficiente, isto é, uma infraestrutura capaz de transportar as pessoas entre sítios, no menor tempo possível, usando a quantidade mínima possível de recursos (em particular, com o menor número de veículos a operar) é um problema de otimização combinatória – em particular, um problema de roteamento[22] – e tema complexo, em debate desde os anos 60 (Darwish et al, 2022). O problema do *design* de uma rede de transportes que sirva toda a população envolve a atenção a múltiplos atores, com interesses e prioridades muito distintos. Os algoritmos desenvolvidos para o desenho deste tipo de redes têm de ser pensados de forma a equilibrar

[18] Note-se, ainda, que quanto comparada com o campo de estudo mais geral da aprendizagem automática, a aprendizagem reforçada é também um tópico relativamente novo e que apenas começou a ganhar tração nos últimos anos.

[19] Conjunto de vértices (nós) e arestas (ligações) que os unem.

[20] Por exemplo, a Química (desenho de novas moléculas) ou a Sociologia (estudo de redes), mas também, gestão de telecomunicações ou de redes de transporte, a definição de calendarizações e horários, a modelação do movimento de passageiros, etc.

[21] A melhor rota, o caminho mais curto, a construção de cronogramas, a distribuição de tarefas, etc. Esta área do conhecimento pretende responder a questões ancoradas em problemas concretos, que surgem no dia-a-dia e que estão intrinsecamente relacionados com a organização económica da sociedade. As questões que se levantam no seio desta disciplina são fáceis de descrever – "qual é o caminho mais curto?", "como distribuir pessoas por horários disponíveis?", "que carga posso recolher sem ultrapassar a capacidade do meu veículo?", "que caminhos devem os carteiros percorrer?", "como devo distribuir as paragens de autocarro numa cidade?".

[22] Eldrandaly et al. (2008) referem que: "*Routing problems can be classified into two main categories based on their objectives and complexity. The first category, path finding problems, includes a set of problems whose main objective is to get the shortest path. These problems are actually simple compared to other types of routing* problems*. The second category, tour construction problems, includes a set of problems that aim to build a complete tour in a given network*" (p. 51).

estes objetivos[23]. Para além disso, note-se que os diferentes objetivos que se pretende atingir estão muitas vezes em conflito (Darwish et al, 2022): basta pensar que os operadores pretendem sempre reduzir o custo das operações (nomeadamente, reduzindo o número de veículos a operar em cada instante) o que implica necessariamente um possível aumento dos tempos de viagem para o utilizador. Por outro lado, a complexidade advém também do número de possibilidades no espaço de decisão/ação no que concerne o desenho da rede (em particular, a determinação das rotas e a definição das frequências).

O problema que pretendemos abordar – o problema do *design* de tais redes – é referido na literatura como "*Transit Route Network Design Problem*", "*Network Design and Frequency Setting Problem*" ou como uma combinação destes dois nomes (TNDFSP). Para uma primeira abordagem a este problema, foi essencial a leitura dos artigos "*Transit Route Network Design Problem: Review*" (Kepaptsoglou & Karlaftis, 2009) e "*Optimising Public Bus Transit Networks Using Deep Reinforcement Learning*" (Darwish et al, 2020), bem como outros artigos complementares, referidos em seguida.

O artigo de Kepaptsoglou & Karlaftis, (2009) permitiu uma introdução inicial ao problema – caracterizando-o – e distinguindo e classificando as diferentes soluções encontradas até à data. O TNDFSP divide-se, em geral, em 5 etapas:

I. desenho de rotas;
II. definição das frequências de cada rota;
III. desenvolvimento de horários (*timetables*);
IV. agendamento de cada veículo; e
V. agendamento de cada condutor.

Muitos dos estudos feitos consideram cada uma destas etapas isoladamente, focando-se, assim, no desenvolvimento de soluções para partes do problema mais geral. Isto porque cada uma destas etapas corresponde a um problema de otimização combinatória específico, de complexidade computacional NP-difícil (*NP-hard*)[24] (Darwish et al., 2020). Com efeito, em (Kepaptsoglou & Karlaftis, 2009) pode ler-se:

> *"the selection of an optimal route structure for a transit network of realistic size is a combinatorial problem of astronomical proportions. (…) while the selection of optimal frequencies in an existing network is, in general, a convex optimization problem, (…) the choice of routes is generally a*

[23] São os diferentes objetivos considerados no *design* destas redes que servem de referência à otimização dos algoritmos desenvolvidos – no sentido em que os algoritmos têm de minimizar ou maximizar valores para cada um dos objetivos, ou de uma métrica que os englobe a todos.

[24] O que significa que, provavelmente, não existe um procedimento para encontrar as melhores rotas com exatidão (Sutton & Barto, 2015, p. 295).

nonconvex (even concave) optimization problem for which no simple procedure exists short of direct comparison of various local minima" (p. 3)[25]

Os VRPs podem ser enquadrados no problema mais geral do TNDFSP, sendo uma das suas etapas, e são uma classe particular de problemas de otimização combinatória, que envolve o desenho de um conjunto de rotas entre paragens, para servir um certo número de clientes, ao mesmo tempo que se optimiza um determinado objetivo, isto é, ao mesmo tempo que se minimiza ou maximiza o custo de uma qualquer variável, relativamente à qual o sistema se diz optimizado.

Considerando a sua variedade, Kepaptsoglou & Karlaftis (2009) fazem ainda uma meta-análise aos estudos já realizados acerca do VRP, sistematizando os diferentes tipos de análise possível. Em particular, o artigo propõe a classificação do tipo de análise, considerando três dimensões: os objetivos, os parâmetros e a metodologia. Assim, tal como todos os problemas de otimização combinatória, os VRP são compostos pelos seguintes elementos ou dimensões: o objectivo, que é a variável que se pretende optimizar (poderá ser a distância percorrida, o número máximo de paragens, a distância máxima entre paragens, o lucro do operador, etc.); a dimensão dos parâmetros, que divide os estudos realizados quanto às variáveis que se pretendem ver calculadas pelo algoritmo desenvolvido (tais como a definição de rotas ou a determinação das frequências de cada meio de transporte) e quanto às restrições existentes no sistema, que são restrições no conjunto de possíveis soluções (por exemplo, um autocarro que serve utentes tem uma capacidade máxima que não pode ser excedida ao longo do percurso, ou, poderemos querer que todos os percursos entre A e B passem por C)[26]; com a dimensão da metodologia pretende-se caracterizar os estudos realizados quanto ao tipo de algoritmos e ferramentas usadas para resolver o problema concreto. Fazem ainda parte de qualquer VRP, a definição de uma "função objectivo", que é a função (a fórmula) que, para cada solução possível calcula o valor do objectivo (por exemplo, a soma da cada uma das viagens parciais de uma rota maior).

Com base nestes conceitos pode dizer-se que uma solução viável (*feasible solution*) é uma solução que satisfaz todos as restrições do problema e que uma solução ótima é uma solução viável em que o valor da função objectivo é o melhor possível (um máximo ou mínimo globais) (Google OR-Tools, 2024).

[25] Citando G. F. Newell: Institute of Transportation Studies, University of California.

[26] Estas são variáveis operacionais e ambientais que afetam e condicionam a rede e que estão frequentemente predefinidas à partida. São restrições comuns a este tipo de problemas o número de rotas, o comprimento máximo de cada rota, o número de meios de transporte disponíveis, bem como a sua capacidade, e também as frequências ou horários permitidos; pela sua importância, algumas destas restrições conduzem a problemas que têm sido tratados separadamente, tal como por exemplo o VRP com capacidade ("*capacitated vehicle routing problem*" – CVRP) ou o VRP com janelas temporais ("*vehicle routing problems with time windows*" – CVRPTW).

Abordagens ao VRP

Ao longo dos anos têm existido diversas abordagens a este problema. Dai et al. (2017) referem que, em geral, há três paradigmas ou abordagens através das quais se tenta resolver o tipo de problemas que se pretende enfrentar, nomeadamente, algoritmos exatos, algoritmos aproximados e heurísticas.

Por um lado, foram desenvolvidos vários algoritmos (por exemplo, *Branch and Bound*, *Branch and Cut*, *Branch and Price*) que permitem encontrar soluções exatas para este problema; no entanto, estes algoritmos são apenas viáveis para instâncias muito pequenas – ou casos particulares muito simples – do problema. Por outro lado, existem também algoritmos aproximados, que garantem que o problema pode ser resolvido em tempo polinomial[27], mas que nem sempre podem ser aplicados para o problema em questão nem garantem uma "optimalidade desejada" (*weak optimality guarantees*), em muitas situações. Por fim, foram desenvolvidas abordagens heurísticas que funcionam no caso em que o espaço de soluções é demasiado grande para se poder encontrar uma solução exata[28]; estas abordagens não garantem que a solução encontrada seja ótima, mas, ao trocarem exatidão por velocidade e recursos computacionais, permitem encontrar soluções satisfatórias num tempo reduzido, o que é muitas vezes suficiente para aplicações práticas. Os métodos heurísticos são baseados em regras de escolha de um determinado caminho em detrimento de outro (por exemplo, escolher sempre o vértice mais próximo) e permitem reduzir o espaço das soluções viáveis e, assim, realizar uma busca em árvore em espaços mais limitados. Existem também métodos meta-heurísticos que envolvem a construção de uma solução possível e o melhoramento dessa solução através de buscas locais (em particular, evitando ou saindo de mínimos não globais) (Nazari et al., 2018), tais como algoritmos genéticos ou a otimização por enxame de partículas.

Note-se que, quanto à metodologia, muitas das abordagens incluem ainda decisões de natureza heurística e regras de decisão de especialistas ou que requerem grande conhecimento sobre o sistema de transporte a ser optimizado (Darwish et al., 2020), pelo que dificilmente são generalizáveis e escaláveis, pois tendem a necessitar de grande suporte de conhecimento humano sobre o problema em questão, altamente específico[29], não sendo ainda suficientemente robustos (isto é, os modelos podem ser afetados por pequenas mudanças).

---

[27] Um algoritmo diz-se polinomial se o tempo que demora a encontrar uma solução para um *input* de tamanho *n* é menor que um polinómio de grau *n* (isto é, menor que Cn^k, para C e k constantes).

[28] Um dos primeiros a desenvolver métodos heurísticos ["desenhados por especialístas" (*handcrafted*)] foi Christoph Mandel, que os aplicou a uma situação real, na Suiça. Esta rede é considerada uma rede *benchmark* para avaliar algoritmos. (Darwish et al., 2020)

[29] Isto é uma limitação de facto, visto que cada número de instâncias do problema (o número de configurações possível de vértices e arestas de um grafo) é muito vasto.

Assim, mais recentemente têm sido desenvolvidos métodos baseados em aprendizagem automática, onde se pretende que o algoritmo "aprenda" melhores regras de decisão. Na literatura podem ser encontradas tentativas de usar aprendizagem supervisionada para resolver este problema; no entanto o uso de aprendizagem supervisionada levanta desafios: por um lado necessita de problemas que já tenham sido resolvidos (problemas de VRP catalogados) – o que é dispendioso –, por outro lado esta abordagem parece encerrar em si mesma uma contradição visto que resolver o problema é precisamente o que se está a tentar fazer. Berto et al. (2023) salientam esta mesma ideia: "Although supervised learning methods are shown to be effective in [Neural Combinatorial Optimization], they require the availability of high-quality solutions, which is unrealistic for large instances or theoretically hard problems" (p.2).

Assim, uma abordagem com AR parece vir a ser mais frutuosa: podendo ser interpretadas como políticas de decisão, as regras de decisão sobre qual o melhor caminho a tomar podem, eventualmente, ser aprendidas por uma rede neuronal profunda, usando mecanismos de AR. A ideia de que estas regras podem ser apreendidas sem qualquer intervenção humana, foi anteriormente defendida por Kool et al. (2018):

> *A decade ago, computer vision algorithms used hand-crafted features but today they are learned end-to-end by Deep Neural Networks (DNNs) (…) Heuristics are typically expressed in the form of rules, which can be interpreted as policies to make decisions. We believe that these policies can be parameterized using DNNs, and be trained to obtain new and stronger algorithms for many different combinatorial optimization problems, similar to the way DNNs have boosted performance in the applications mentioned before.* (p. 1)

Introdução à Aprendizagem Reforçada

A AR, é, tal como referido, um tópico cuja aplicação na prática é relativamente recente e que apenas começou a ganhar tração nos últimos anos. Note-se a Figura 1, onde se mostra que o número de artigos acerca deste tópico cresceu claramente nos últimos anos:

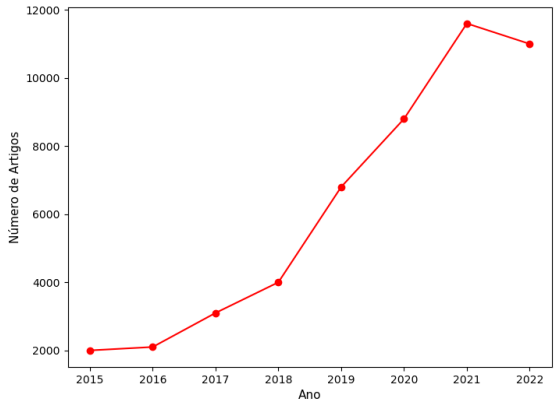


Figura 1: Número de entradas no Scopus com as palavras "*Reinforcement Learning*"

Embora já date de meados do século passado o aparecimento das primeiras ideias matemáticas e conceitos estatísticos – tal como a "Equação de Bellman" e os "Processos de Decisão de Markov" – que tornaram possível o desenvolvimento da AR, o início do desenvolvimento de algoritmos específicos de AR iniciou-se apenas em finais do século passado[30] e só mais recentemente foi possível aplicar com sucesso alguns destes conceitos e algoritmos[31].

Estes desenvolvimentos – de técnicas, métodos e algoritmos de AR – permitem abrir novas possibilidades de aplicação da aprendizagem automática a outros problemas, nomeadamente a problemas onde não é possível programar a "infinidade" de casos possíveis (por exemplo, movimento robótico) ou onde não podem ser "mostrados" exemplos significativos suficientes (ou seja, onde não é possível utilizar técnicas de aprendizagem supervisionada).

A AR[32] corresponde a uma abordagem computacional ao problema da tomada de decisão em ambiente de incerteza, pretendendo-se que estes algoritmos aprendam a tomar decisões e a agir sozinhos, sem supervisão ou conhecimento anterior humano. Pretende-se mimetizar certas facetas da aprendizagem humana – tais como, por exemplo, a capacidade de modificar ações perante os erros cometidos ou a capacidade de repetir ações que tenham sido benéficas no passado – traduzindo

[30] São exemplos destes algoritmos, o Q-learning (1989) ou o REINFORCE (1992); note-se também que (Sutton & Barto, 2015) é considerado um dos livros seminais sobre AR e cuja primeira edição data de 1998.

[31] A aplicação de AR ao controlo de helicópteros, ao controlo da refrigeração de *data centers* ou na vitória do jogo do Go. Mais recentemente, têm sido desenvolvidas aplicações em áreas tão diferentes como a logística e a gestão de inventários de bens perecíveis (Selukar et al., 2022), a economia e a formação de preços, a sociologia, a gestão de redes electricas, os problemas de *optimal-control* em estufas ou cadeias de abastecimento, a procura por algoritmos mais rápidos para a multiplicação de matrizes, o desenho de fármacos, entre muitos outros exemplos (Fawzi et al., 2022; Ha & Jeong, 2022; Nakabi & Toivanen, 2021; Popova et al., 2018; Zhang et al., 2022; Q. Zhou et al., 2022).

[32] Mais especificamente, Aprendizagem Reforçada Profunda (*Deep Reinforcement Learning*).

algoritmicamente a sua forma de funcionamento[33]. Esta é, assim, uma abordagem marcadamente transdisciplinar, ao incorporar conhecimentos vindos de áreas como a matemática, a computação e a engenharia e de ciências que estudam variados aspetos do comportamento humano como a psicologia e sociologia, a economia e as neurociências (Silver, 2016).

Muitos são os aspetos que distinguem a aprendizagem reforçada das aprendizagens supervisionada e não supervisionada. A AR dá-se através de um sinal de recompensa (*feed-back*) dado pelo ambiente em que o algoritmo se encontra (por oposição ao uso de dados pré-categorizados – *labelled*), sinal esse usado na computação da decisão sobre a próxima ação a tomar[34] através de mecanismos dinâmicos de tentativa e erro, e de acções que equilibram fases de exploração e fases de consumo[35] (*exploration versus exploitation*) (Plaat, 2022).

Por agirem sobre o ambiente os algoritmos de aprendizagem reforçada são denominados "agentes"; em Whittlestone et al. (2021) pode ler-se:

> *RL systems are often referred to as "agents", because they act autonomously in their environment. This does not mean that an RL agent is responsible or aware of their actions, just that the human designer is one step removed from the action selection process.* (p. 1006)

Assim, enquanto as aprendizagens supervisionada e não supervisionada usam, à partida, um conjunto de dados de *input*, categorizados ou não, e produzem como *output* classificações, regressões ou a descrição de associações a padrões entre esses dados, as técnicas de AR não têm necessariamente de ter como *input* um conjunto de dados, mas apenas a definição clara da tarefa que têm de cumprir, e produzem como *output* uma tomada de decisão consubstanciada na realização de uma ação (o algoritmo tem acesso a um ambiente com que interage e a partir do qual vai recolhendo a informação necessária e constrói conhecimento). Estes algoritmos implicam problemas técnicos e de computação específicos, pois a sua construção tem de ter em conta que os dados chegam por ordem sequencial, que estes dados (as observações do agente) estão muitas vezes altamente correlacionados[36], que o agente influencia o

[33] Especula-se, inclusive, que a noção de recompensa, positiva ou negativa, é suficiente para sumarizar todas as formas de aprendizagem e que a maximização de uma recompensa pode ser tudo o que é necessário para desenvolver capacidades aparentemente inteligentes: "*(…) we consider an alternative hypothesis: that the generic objective of maximising reward is enough to drive behaviour that exhibits most if not all abilities that are studied in natural and artificial intelligence*" (Silver et al., 2021, p. 1).

[34] O conjunto de acções a tomar pode não ser discreto (Lillicrap et al., 2015).

[35] Exploração refere-se a qualquer acção que seja seguida para conhecer algo não familiar (mesmo que isso implique uma recompensa menor). Consumo refere-se ao conjunto de acções em que o algoritmo decide seguir o caminho que já conhece considerando que o retorno desse caminho é bom.

[36] Ao contrário de muitos problemas tradicionais, onde a informação está (ou é considerada como) identicamente distribuída.

ambiente que o rodeia e, consequentemente, o que "vê" e, por fim, que a recompensa de uma acção pode não ser – e muitas vezes, não é – imediata.

Fundamentalmente, os algoritmos de aprendizagem reforçada são usados para responder a problemas com características sequenciais, em que o algoritmo "aprende" interagindo com o meio ambiente que o rodeia, recebendo recompensas pelas suas ações, de forma a realizar tarefas com uma finalidade pré-determinada: é esta característica sequencial que os torna ideais para modelar situações da "vida real", isto é, situações em que é necessário ir fazendo constantes e pequenos melhoramentos de decisão (com base na recompensa – ou, sinal – dada pelo ambiente) para, através da ação sobre o meio ambiente e do planeamento continuado do futuro, realizar uma qualquer tarefa.

Assim, enquanto as aprendizagens supervisionada e não supervisionada respondem, essencialmente, a problemas estáticos, a aprendizagem reforçada responde a problemas dinâmicos, onde a ação do algoritmo interage com o meio ambiente, alterando-o, e modificando com isso a informação que o próprio vai ter disponível para recolher e analisar no futuro. O problema é dinâmico, visto que a cada novo passo temporal o algoritmo precisa de se adaptar a uma situação diferente.

De realçar ainda que esta categoria de algoritmos mostra-se adequada para lidar com problemas em que há múltiplos atores em jogo, possivelmente com motivações diferentes, situações em que é necessário balançar múltiplos objetivos, possivelmente contraditórios entre si (Darwish et al., 2020).

O artigo "*Human-level control through deep reinforcement learning*" (Mnih et al., 2015) marca, tal como já foi referido, um desenvolvimento importante na área da aprendizagem reforçada. O desenvolvimento de que falamos corresponde à criação das primeiras "*deep Q-networks*" (DQN), que combinam redes neuronais profundas[37] e aprendizagem reforçada, fundando o campo da aprendizagem reforçada profunda (*deep reinforcement learning*), campo que permitiu a expansão das possíveis aplicações de AR:

> *Because neural networks can approximate any function, they can be used to encode the policy of an agent. This allows RL to extend to problems with state or action spaces that are too large for the tabular approach—which includes most interesting modern applications* (Whittlestone et al., 2021, p. 1007)

As DQN permitiram que pela primeira vez o mesmo algoritmo – o mesmo agente, com a mesma arquitectura e os mesmos hiperparâmetros – fosse capaz de realizar várias tarefas distintas, nomeadamente, jogando vários jogos de Atari. No caso das DQN, as redes neuronais profundas foram

[37] Nomeadamente, redes neuronais convolucionais.

usadas para aproximar o valor da função de valor de acção (*action-value function*, definida em seguida) (Lillicrap et al., 2015). A junção de aprendizagem reforçada com redes neuronais profundas veio aumentar o alcance das soluções vindas deste tipo de algoritmos, visto que esta nova abordagem permitiu ultrapassar as dificuldades anteriores relacionados com a possibilidade destes algoritmos funcionarem apenas em ambientes de "fraca resolução" e espaços de ação discretos. Com efeito, segundo Mnih et al. (2015), "*This work bridges the divide between high-dimensional sensory inputs and actions, resulting in the first artificial agent that is capable of learning to excel at a diverse array of challenging tasks*" (p. 529). No entanto, as DQN eram ainda limitadas, pois, embora conseguissem lidar com problemas que envolviam espaços de observação muito grandes, os espaços de acção tinham de ser discretos, de dimensão relativamente pequena (Lillicrap et al., 2015).

Considera-se aprendizagem reforçada profunda quando redes neuronais são utilizadas para aproximar qualquer uma das funções da aprendizagem reforçada: a política (*policy*), a recompensa (*reward function*), o modelo do ambiente e qualquer uma das funções de valor (Selukar et al., 2022).

Sistematizando, então, refira-se que os algoritmos de aprendizagem reforçada têm como missão selecionar e executar, a cada passo, uma ação, avaliando de seguida o novo estado do ambiente em que se encontram fruto da sua ação e recolhendo uma recompensa fornecida pelo ambiente (ver Figura 2). Essa recompensa é, no fundo, uma informação que indica ao algoritmo se a ação que tomou o aproximou ou não do seu objetivo[38]; "*reinforcement learning (RL) is a learning framework that improves a policy in terms of a given objective through interaction with an environment where an agent perceives the state of that environment"* (Matsuo et al., 2022, p. 270).

[38] Esta noção de recompensa é mais complexa do que se imagina à primeira vista; não só a recompensa de uma determinada acção pode apenas ser percebida muitos passos temporais depois da acção (ou seja, percebida apenas retrospetivamente), como os algoritmos de aprendizagem reforçada podem "planear", decidindo ter, no momento presente, uma recompensa negativa perante uma determinada acção de forma a ter uma recompensa maior no futuro.

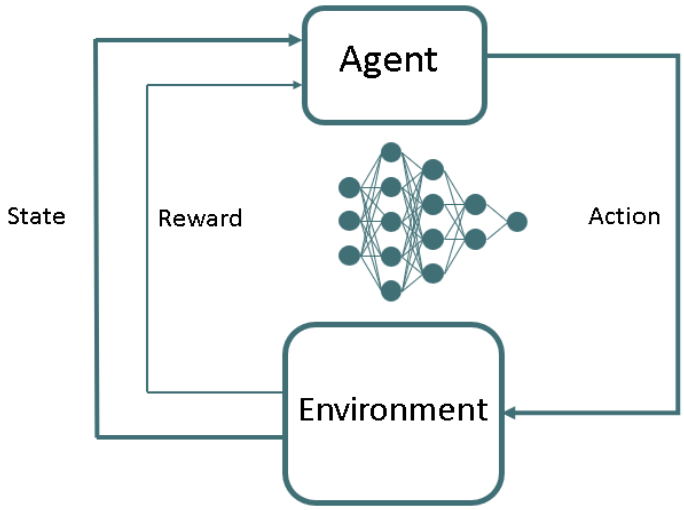


Figura 2: Relação entre Agente e Ambiente[39]

A secção seguinte, onde irão ser definidos alguns conceitos fundamentais para a compreensão desta classe de algoritmos, suporta-se maioritariamente em Sutton & Barto (2015), Mazyavkina et al. (2020) Kirk et al. (2021) e Plaat (2022):

a. ***State* (Estado, S):** Informação usada pelo agente para determinar a acção seguinte; é uma representação interna que o agente faz da História (aquilo de que "se lembra" ou não), isto é, da situação corrente do ambiente, num determinado momento; é um subconjunto da história

b. ***Action* (Acção, A)**: Uma decisão ou movimento que afecta o estado do ambiente

c. ***Reward* (Recompensa, R)**: Um valor escalar; é o *feed-back* (imediato) dado ao agente, dependente do estado do ambiente e da acção tomada: $\mathbf{R_S = E[R_{T+1} \mid S_T = S, A_T = a]}$

d. **História (H)**: Sequência de observações, acções e recompensas

e. **Agente**: Um mapa (função) que atribui à história observada uma acção seguinte. O agente observa os estados, decide e executa acções e recolhe as recompensas. O objectivo do agente é maximizar a soma de todas as recompensas (o retorno)

f. **Estado do ambiente**: É a informação privada do ambiente, que determina, do ponto de vista do ambiente, o que acontece após uma determinada acção; esta informação não é, na maioria das vezes, acessível ao agente

[39] Imagem inspirada em (Bengio et al., 2021).

g. **Retorno (G)**: Corresponde à recompensa descontada total a partir do tempo t: $\mathbf{G_T = R_{T+1} + \gamma R_{T+2} + \gamma^2 R_{T+3} + \gamma^3 R_{T+4} + \ldots}$ onde $\gamma \in [0,1]$ é um factor de desconto que implica, como regra geral, que as recompensas presentes são mais valorizadas que as recompensas futuras

h. **Política (π)**: Determina o comportamento do agente – como é que o agente escolhe as acções; é um mapa (função) que atribui a cada estado uma acção; pode ser determinística ou estocástica, correspondendo, neste último caso, a uma distribuição de acções dado um estado específico: $\mathbf{\pi(a \mid s) = P[A_T = a \mid S_T = s]}$

i. **Função de Valor de Estado (V)**: Função que prevê as recompensas futuras de um determinado estado (isto é, quão "bom" ou "mau" é um determinado estado) e seguindo uma política **π** especificada; pode ser estimada com amostras. $\mathbf{V_\pi(S) = E_\pi[R_T + \gamma R_{T+1} + \gamma^2 R_{T+2} + \ldots \mid S_T = s] = E_\pi[G_T \mid S_T = s]}$ onde $\gamma \in [0,1]$ é o factor de desconto já referido

j. **Função de Valor de Acção (Q)**: Corresponde ao retorno expectável a partir do estado **s**, seguindo a acção **a** e depois a política **π**: $\mathbf{Q_\pi(s,a) = E_\pi[G_T \mid S_T = s , A_T = a]}$

k. **Modelo**: Um modelo prevê o que é que o ambiente vai fazer em seguida; no que concerne a AR, o modelo do ambiente pode ser:

   i. **Transaccional (*transactional model*)**: o modelo prevê o próximo estado do ambiente (isto é, o modelo prevê a dinâmica do ambiente)**:** $\mathbf{P_{SS'} = P[S' = s' \mid S = s , A=a]}$

   ii. **De recompensa (*reward model*)**: o modelo prevê a recompensa imediata: $\mathbf{R_S = E[R \mid S = s , A = a]}$

l. **Processo de Markov**: É um processo aleatório "sem memória", isto é, uma sequência de estados com a propriedade de Markov: a probabilidade de um acontecimento futuro apenas depende do estado presente e não dos estados passados: $\mathbf{P[S_{T+1} \mid S_T] = P[S_{T+1} \mid S_1,\ldots,S_T]}$

m. **Matriz [de Probabilidade] de Transição de Estados (P)**: É uma matriz que define a probabilidade de transição entre cada par de estados, para todos os estados possíveis do ambiente

n. **Processo de Decisão de Markov (MDP)**: "*Markov Decision Processes are a tool for modeling sequential decision-making problems where a decision maker interacts with a system in a sequential fashion*" (Szepesvári, 2010). Os MDP são uma extensão dos processos de Markov

aos quais se acrescenta a possibilidade de diferentes acções (para maximizar o retorno). Matematicamente, correspondem a um vector – **(S,A,P,R,γ)** – composto pelos seguintes elementos:

i. **S** é um conjunto (finito) de estados
ii. **A** é um conjunto (finito) de acções
iii. **P** é a matriz de transição de estados
iv. **R** é a função recompensa
v. **γ** é o factor de desconto

o. **Equação de Bellman**: A equação de Bellmann permite encontrar as funções de valor de um sistema, em alguns casos de forma linear. Esta equação pode ser usada para decompor tanto a função de valor de estado como a função de valor de acção. Note-se que a função de valor pode ser decomposta em duas partes: a recompensa imediata e as recompensas descontadas dos estados seguintes. Esta estratégia de decomposição permite estabelecer uma relação recursiva para $\mathbf{V_\pi(S)}$ e $\mathbf{Q_\pi(s,a)}$. Assim:

$$\begin{aligned}
\mathbf{V(S)} &= \mathbf{E[G_T \mid S_T = S]} \\
&= \mathbf{E[R_{T+1} + \gamma R_{T+2} + \gamma^2 R_{T+3} + \ldots \mid S_T = S]} \\
&= \mathbf{E[R_{T+1} + \gamma(R_{T+2} + \gamma R_{T+3} + \ldots) \mid S_T = S]} \\
&= \mathbf{E[R_{T+1} + \gamma G_{T+1} \mid S_T = S]} \\
&= \mathbf{E[R_{T+1} + \gamma V(S_{T+1}) \mid S_T = S]}
\end{aligned}$$

$\therefore$ $\mathbf{V(S) = E[R_{T+1} + \gamma V(S_{T+1}) \mid S_T = S]}$ é a equação de Bellman **(1)**

Resolvendo algebricamente a equação **(1)**:

$$\begin{aligned}
\mathbf{V(S)} &= \mathbf{E[R_{T+1} \mid S_T = S] + E[\gamma V(S_{T+1}) \mid S_T = S]} \\
&= \mathbf{R_S + \gamma E[V(S_{T+1}) \mid S_T = S]} \\
&= \mathbf{R_S + \gamma \sum P_{SS'} V(S')},
\end{aligned}$$

onde $\mathbf{P_{SS'}}$ é a probabilidade de transição de **S** para **S'**

Simplificando a notação, tem-se que:

$\mathbf{V = R + \gamma PV}$, onde **P** é a matriz de Probabilidade de Transição de Estados **(2)**

De **(2)** pode concluir-se que:

$$\mathbf{V - \gamma PV = R \Rightarrow}$$

$$\mathbf{(1 - \gamma P)V = R \Rightarrow}$$

$$\mathbf{V = (1 - \gamma P)^{-1}R} \qquad \mathbf{(3)}$$

De **(3)** conclui-se que é necessário inverter a matriz de probabilidades de transição de estados para resolver a equação de Bellman

p. Note-se que a inversão da matrix **P** em **(3)** é computacionalmente complexa, pelo que a resolução da equação de Bellman só pode ser feita directamente nos casos mais simples. Em geral é necessário usar diferentes técnicas, tais como:

   a. Programação Dinâmica
   b. Simulação de Monte Carlo
   c. Aprendizagem por Diferença Temporal

Estas diferentes técnicas permitem caracterizar o tipo de algoritmo de AR que se está a usar. Assim, por exemplo, a Programação Dinâmica necessita de total conhecimento do sistema (isto é, conhecimento da matriz de probabilidades de transição entre estados e da função recompensa); a Avaliação de Monte Carlo estima a função de valor de estado com base na média do retorno de amostras de episódios e necessita, portanto, que os episódio terminem; a Aprendizagem por Diferença Temporal combina as duas técnicas anteriores, aprendendo (isto é, aproximando-se da verdadeira função de valor de estado) antes do fim de cada episódio (não necessita que os episódios terminem)

Considerando as definições apresentadas, pode dizer-se que os algoritmos de AR podem ser classificados quanto à estratégia de aprendizagem e de representação, em:

a. Algoritmos baseados em valor (*value-based*): quando o agente possui uma função de valor, que pode ser baseada no valor do estado (função de valor de estado, **V(S)**) ou no valor da acção (função de valor de acção, $\mathbf{Q_\pi(s,a)}$);
b. Algoritmos baseados em política (*policy-based*): quando o agente possui uma representação explícita da política a seguir; e
c. Actor-Crítico (*actor-critic*): nestes casos, o agente possui uma política definida, bem como o valor que cada estado representa.

Podem também ser classificados quanto à origem dos dados que o agente usa para aprender, em:

a. Métodos *on-policy*: o agente aprende com base na política que está a usar, isto é, a política usada para decidir o comportamento do agente é a mesma que é atualizada; e
b. Métodos *off-policy*: o agente pode usar informação gerada por outra política que não a que está a usar no momento.

Aplicações de AR em problemas de Optimização Combinatória

A hipótese de que a AR pode representar uma das vias possíveis para resolver problemas de otimização combinatória é defendida em Bengio et al. (2021) e Berto et al. (2023). Não obstante, considerando que uma abordagem ao TNDFSP através de AR se encontra ainda nos seus primórdios[40] e que este é um problema extremamente complexo[41], o estudo de procedimentos e soluções para instâncias mais especificas do TNDFSP, tais como o "problema do caixeiro-viajante" (*travelling salesman problem*, TSP), o VRP ou o "problema de orientação" (*orienteering problem* – OP)[42], entre outros, poderá sugerir pistas para a resolução do TNDFSP.

Bello et al. (2017) tinham anteriormente desenvolvido um primeiro esforço nesse sentido ao equacionar a resolução do TSP, sem recurso a qualquer tipo de conhecimento prévio[43]. Os seus resultados, embora ainda não satisfatórios mostram caminhos possíveis para a aplicação da AR profunda em problemas de otimização. Neste artigo os autores usam a noção de "negativo da distância do percurso" como sinal de recompensa para o agente; o problema é traduzido para linguagem matemática[44] e a solução é redefinida como a tentativa de encontrar uma permutação dos vértices do grafo que satisfaça as condições pretendidas (nomeadamente, uma permutação que seja um ciclo completo dos pontos do grafo). Os autores utilizam uma *pointer network* treinada de forma diferente da originalmente desenvolvida por Vinyals et al. (2015)[45]: em vez de ser treinada com base em aprendizagem

[40] O primeiro artigo especificamente dedicado a uma solução do TNDFSP com AR parece ser (Darwish et al., 2020).
[41] "*sources of complexity for the TRNDP: its combinatorial and multiobjective nature, the difficulties in formulating the problem and formally defining acceptable spatial route layouts, and the nonlinearities and nonconvexities characterizing the problem.*" (Baaj and Mahmassani, 1991, as cited in Kepaptsoglou & Karlaftis, 2009)
[42] O TPS é um problema de otimização combinatória clássico: dada uma lista de cidades e a distância entre cada par de cidades, qual é o caminho mais curto que retorna à origem após visitar cada cidade uma e uma só vez. O OP envolve fazer uma escolha, constrangida por determinados fatores, de um subconjunto de localidades com o objectivo de maximizar o valor nelas recolhido.
[43] O problema abordado em particular é o do TSP num espaço euclidiano 2D (plano euclidiano).
[44] Definição da distância entre dois pontos e do objetivo como o mínimo da distância, definição de caminho como uma permutação de pontos, factorização da probabilidade total de um caminho em probabilidades condicionais (regra da cadeia).
[45] *Pointer Networks:* Redes neuronais especificamente construídas para prever uma permutação de elementos que são *tokens* discretos numa determinada posição da sequência de *input*; o uso de RNN para resolução de

supervisionada[46] e função de perda correspondente a uma medida de entropia[47] – tal como fizeram os autores originais –, em (Bello et al., 2017) a *pointer network* é modificada ao ser treinada com uma abordagem *policy-based* (em particular, *actor-critic*) usando um algoritmo denominado REINFORCE (Williams, 1992, como citado em Bello et al., 2017).

Em Kool et al. (2018) os autores propõem uma arquitectura de rede do tipo *Encoder-Decoder*[48] e um modelo baseado em "mecanismos de atenção" (*attention layers*)[49] e AR, originalmente usado para encontrar soluções para o TSP e também para variações do VRP[50]. O *encoder* aí usado é denominado *Transformer*, uma arquitectura de rede originalmente desenvolvida por Vaswani et al. (2017) e testada com sucesso[51] em problemas de tradução, que tem a particularidade de não usar RNN na estrutura de *encoder-decoder*, mas apenas "mecanismos de atenção", de forma semelhante às *pointer networks*. Kool et al. (2018) reforçam que os modelos baseados somente em "mecanismos de atenção" têm benefícios relativamente à arquitectura de Vinyals et al. (2015) (onde o treino é feito de forma supervisionada).

No que diz respeito aos trabalhos de Vinyals et al. (2015) e Bello et al. (2017) – trabalhos considerados seminais – Dai et al. (2017) referem que, embora estes trabalhos permitam já lidar com grafos de diferentes tamanhos (propriedade que é desejável com vista à generalização dos algoritmos encontrados) apenas conseguem fazê-lo através de algum tipo de engenharia *ad-hoc* (por exemplo, através da adição

problemas de otimização combinatória tem limitações, visto que as RNN necessitam, em geral, que os dados de *input* e de *output* tenham o mesmo tamanho, o que não é o caso em muitos problemas deste género. As *pointer networks* representam uma tentativa de contornar esta limitação (Vinyals et al., 2015). Representam, ainda, uma abordagem supervisionada a problemas de otimização combinatória (Berto et al., 2023).

[46] Isto é, baseada em dados anotados: para gerarem dados anotados, Vinyals et al. (2015) usam diversos algoritmos (de natureza heurística) para resolver sub-optimamente o TSP.

[47] "*Entropy is a measure of disorder that can be applied to a set (…). Disorder corresponds to how mixed (impure) the segment is with respect to these properties of interest (…) we can define information gain to measure how much an attribute improves (decreases) entropy over the whole segmentation it creates*" (Provost & Fawcett, 2013, pp. 51-52)

[48] Neste tipo de redes os dados são inicialmente codificados (*embedded*) em vetores de grandes dimensões pelo *encoder* (no caso particular deste artigo, pontos de 3 dimensões espaciais são codificados em vectores de 128 dimensões).

[49] "*Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences*" (Vaswani et al., 2017, p. 2);

[50] Tais como o OP e o TSP com recolha de prémios (*prize collecting TSP – PCTSP*).

[51] Aqui, a definição de sucesso é mais técnica, pois o objetivo dos autores é mostrar que redes mais simples, baseadas em mecanismos de atenção (e não baseadas em RNN, tipicamente usadas para problemas sequenciais) são mais paralelizáveis e, consequentemente, mais rápidas; em todo o caso, esta arquitectura de rede está na origem do primeiro modelo de linguagem conhecido pelo público em geral – o GPT.

de zeros – *padding zeros* – às matrizes de adjacência[52], para lidar com grafos de tamanhos diferentes); referem ainda que estas abordagens adotam estratégias que não têm em consideração a estrutura do grafo, sendo "*graph-agnostic sequence-to-sequence mapping*" (p. 2).

Como já houve oportunidade de referir, a estrutura que melhor descreve o TNDFSP é uma estrutura em grafo, direccionado ou não, em que tanto aos vértices como às arestas podem ser acrescentadas diversas caraterísticas, sendo a matriz de adjacência (*adjacency matrix*) uma das formas primordiais de representar grafos em formato tabular.

Os problemas de otimização combinatória sobre grafos têm merecido considerável atenção nos últimos anos, tendo estas estruturas complexas capacidade para generalizar aspectos muito diversos da realidade humana, aspetos esses que são representados por dados diferentes, mas que mantém a mesma estrutura combinatória (Dai et al., 2017). Assim, a tentativa de aprendizagem automática de heurísticas gerais que funcionem em muitas instâncias diferentes do mesmo problema pode ser promissora. Dai et al. (2017) aplicam esta ideia a problemas de otimização combinatória como o *Minimum Vertex Cover*, o *Maximum Cut* e o TSP combinando AR com uma forma de codificação (*embedding*) do grafo denominada "Structure2Vec".

Veličković et al. (2017) argumentam que existem limitações quanto ao uso de redes neuronais convolucionais[53] para a resolução de problemas de otimização combinatória:

> *Convolutional Neural Networks (CNNs) have been successfully applied to tackle problems such as image classification (…), semantic segmentation (…) or machine translation (…), where the underlying data representation has a grid-like structure (…) However, many interesting tasks involve data that can not be represented in a grid-like structure and that instead lies in an irregular domain. This is the case of 3D meshes, social networks, telecommunication networks, biological networks or brain connectomes. Such data can usually be represented in the form of graphs.* (p. 1)

Também já antes foi possível encontrar argumentos contra o uso de RNN para tratar problemas de otimização combinatória: com efeito, Vinyals et al. (2015) referem que:

---

[52] "*The adjacency matrix, sometimes also called the connection matrix, of a simple labeled graph is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in (…) according to whether (…) [the respective nodes] are adjacent or not. For a simple graph with no self-loops, the adjacency matrix must have 0s on the diagonal. For an undirected graph, the adjacency matrix is symmetric*" (Wolfram, 2023)

[53] Note-se que uma imagem é um caso muito particular de grafo em que cada nó tem sempre o mesmo número de ligações. As técnicas tradicionais de convolução (tal como as usadas nas DQN) permitem apenas *inputs* de tamanho regular, porque cada pixel da imagem tem sempre o mesmo número de vizinhos.

> *[RNN-based] methods still require the size of the output dictionary to be fixed a priori. Because of this constraint we cannot directly apply this framework [RNN] to combinatorial problems where the size of the output dictionary depends on the length of the input sequence.* (p. 1)

Assim, considerando estes argumentos, Veličković et al. (2017) desenvolvem uma arquitectura de rede a que chamaram Redes de Atenção em Grafos (*Graph Attention Networks* – GATs). Esta arquitectura vem melhorar anteriores implementações da modelação de grafos através de mecanismos de *deep learning*, ao contribuir para o trabalho de generalização de processos de convolução ao domínio dos grafos, nomeadamente através do uso de mecanismos de atenção (já referidos anteriormente); o artigo descreve pormenorizadamente a forma como os mecanismos de atenção funcionam, referindo-se ainda que têm diversas vantagens relativamente às implementações anteriores:

- operações mais eficientes porque são paralelizáveis
- podem ser aplicadas a todos os vértices de um grafo, independentemente do seu grau[54]
- maior capacidade de generalização para grafos não vistos (*out-of-sample data*)

Estudos Relevantes sobre Soluções para o VRP baseadas em AR

O uso de instrumentos desenvolvidos para problemas que não o próprio TNDFSP é também o procedimento adoptado em Darwish et al. (2020), onde os autores usam a arquitectura desenvolvida em Kool et al. (2018) (que tinha sido desenvolvida e aplicada em problemas mais simples, como o TSP, OP e PCTSP), para uma tentativa de resolução do próprio TNDFSP. Apoiando-se no facto de Kool et al. (2018) usarem mecanismos de atenção com resultados promissores e referirem que a sua arquitectura pode ser aplicada noutros problemas desde que para isso se altere a forma como os pontos visitados são mascarados (ou não) pelo algoritmo, (Darwish et al., 2020) atingem resultados bastante interessantes, especialmente se se considerar que modelam aspectos muito particulares de uma rede de transportes públicos como é o caso das mudanças de linha e, inclusive, uma forma de contabilizar o tempo de espera nas paragens (a atenção a estes aspectos não foi vista em nenhum outro trabalho). Também Yoo et al. (2023) referem conseguir modelar situações bastante mais complexas que noutros artigos estudados: neste caso, os autores dizem ter conseguido desenvolver uma arquitectura que modela (e optimiza) simultaneamente o número de rotas, o seu desenho e a frequência dos transportes. No entanto, o seu artigo é bastante vago quanto ao tipo de algoritmos usado. Infelizmente, não estando disponível *online* o

[54] O grau de um vértice num grafo não dirigido corresponde ao seu número de arestas.

código usado no âmbito de qualquer destes artigos, não nos foi possível reproduzir os resultados alcançados aqui.

Nazari et al., (2018) desenvolvem um modelo para a resolução do Problema de Roteamento de Veículos com Capacidade (*Capacitated Vehicle Routing Problem* – CVRP) que tem a vantagem de conseguir ser aplicado não só a instâncias "estáticas", onde todas as condições são conhecidas à partida, mas também a situações "dinâmicas" onde perturbações externas (tal como o aparecimento, a meio do processo, de novos utentes que alteram os requisitos de uma determinada localização) fazem com que a função de transição de Markov seja desconhecida; desenvolvem, então, um modelo que pode ser aplicado a situações estocásticas. No seu modelo substituem o *RNN enconder* usado por Bello et al. (2017) argumentando que, ao contrário das tarefas de tradução onde a sequência do *input* fornece informação relevante, as RNN não são necessárias em problemas de otimização combinatória onde a ordem com que o *input* é dado ao modelo não tem significado. No seu artigo comparam os seus resultados com o de outros *solvers*, com muito bons resultados especialmente para instâncias maiores do problema (50 pontos e mais). Utilizam duas técnicas de *decoding* – *greedy*, em que a cada passo é escolhido sempre o ponto com maior probabilidade e *beam search* que escolhe o caminho que corresponde à rota com menor comprimento de entre os pontos com maior probabilidade. Embora obtenham resultados melhores com o segundo *decoder*, a distância entre os resultados esbate-se com o aumento das instâncias.

Peng et al. (2020) partem do modelo idealizado por Kool et al. (2018), modificando-o de forma a que as características de cada vértice sejam alteradas à medida que os caminhos vão sendo construídos, de forma incremental, pelo agente. Pretendem com isto ultrapassar aquilo que consideram ser uma limitação do método de Kool et al. (2018) pois nesse caso as características de cada vértice são definidas no início do processo, não se alterando posteriormente[55].

Mais recentemente, têm sido desenvolvidas abordagens com racionalidades diferentes (isto é, onde não se pretende apenas resolver um VRP). A pensar na capacidade de generalização destes algoritmos, não só a instâncias ainda não vistas pelo agente (generalização "clássica") mas também à capacidade do algoritmos em resolver várias variantes distintas (com condicionamentos diferentes) do VRP (e considerando o facto de as abordagens até ao momento se focarem, essencialmente, na solução a uma das variantes possíveis), Zhou et al. (2024) desenvolvem um algoritmo, a que chamam *Multi-task Vehicle Routing Solver with Mixture-of-Experts* (MVMoE), que consegue lidar com várias variantes do VRP

[55] O que não deveria ser o caso, pelo menos em teoria, já que as características de um nó dependem das características dos seus vizinhos e das dos vizinhos dos vizinhos (numa cadeia tão mais profunda quanto se queira) e se estes se vão alterando, também as características iniciais deveriam alterar-se.

simultaneamente. Esta rede neuronal é treinada numa instância diferente do problema por cada época de treino; o mecanismo de MoE consiste num conjunto de "especialistas", que são redes de propagação directa (*feed forward networks*) com parâmetros independentes e treináveis e uma rede de modulação (*gating network*) parametrizável, responsável por decidir que aspectos do *input* são distriuídos para cada "especialista". Também na mesma direcção seguem os trabalhos de Liu et al. (2024) e Berto et al. (2024) que propõem um modelo que resolve um VRP com combinações de condicionamentos diferentes das vistas pelo modelo durante a fase de treino; fazem-no através de uma redefinição do próprio problema do VRP, generalizando-o como uma combinação de diferentes atributos cuja diferente mistura cria uma variação diferente.

Metodologia

Formulação do VRP em Lisboa

Pretende-se com esta tese aprofundar o conhecimento acerca do funcionamento da AR, bem como das suas potencialidades de aplicação à resolução de um problema de otimização combinatória. Assim, através de um estudo de caso, aplicar-se-á um dos algoritmos de AR já descritos a uma situação concreta. A motivação é dupla: por um lado, pretende-se avaliar o funcionamento destes algoritmos quando aplicados em dados reais – e não só em dados de *benchmark*, como acontece em muitos casos; por outro lado, é pertinente tentar aplicar estes algoritmos a problemas de cariz marcadamente social, tal como o de o melhoramento de uma rede de transportes públicos (evidentemente, com diversos impactos positivos numa cidade). No seguimento destas motivações, pretende-se incorporar nesta discussão e na avaliação dos algoritmos algum tipo de métrica sobre o impacto da solução encontrada[56].

A escolha do estudo de caso recaiu sobre a cidade de Lisboa e, em particular, sobre a rede pública de autocarros da Carris. Esta rede representa um bom ponto de partida visto que, além de não ser necessário definir os pontos de paragem[57] (os vértices do grafo), as redes de autocarro (por oposição às de metro) são mais flexíveis, pelo que, na prática e caso fosse necessário, seria mais fácil alterar os locais de paragem, e também as linhas que os unem e as frequências dos autocarros.

No que concerne a rede da Carris, Lisboa encontra-se dividida em cinco zonas geográficas (Centro, Ocidental, Noroeste, Norte e Oriental), havendo ainda uma zona funcional (Circulares) que não corresponde a nenhuma zona geográfica específica, dizendo respeito a linhas que atravessam todas ou parte das restantes zonas e que permitem a ligação entre zonas não adjacentes (ver Figura 3).

[56] A eficiência de uma rede de transportes pode ser medida através de vários indicadores, tais como por exemplo, o tempo máximo que cada meio de transporte demora a percorrer determinado trajecto, o número de paragens que efectua, a distância total que o transporte percorre, o número de pessoas que consegue transportar, a distância máxima entre cada paragem, o custo de manter a frota existente, o número necessário de condutores, o tempo de viagem médio de cada utente, etc.

[57] Tal como já referido, na grande maioria dos casos a escolha do local das paragens é um problema tratado separadamente. Assumir-se-á, aqui, que esta escolha é ótima, considerando uma determinada racionalidade por parte da Carris.

Figura 3: Diagrama estilizado da Rede Carris

As quatro zonas exteriores (isto é, todas menos a zona Centro e a zona Circulares) são compostas por linhas com um movimento mais radial, que confluem em direcção ao centro, presumivelmente, porque é esse o movimento mais comum da população, que vai trabalhar de manhã para o centro da cidade e que volta para casa ao final do dia; a zona centro é composta por linhas sem direcção definida e que cumprem objectivos mais complicados, nomeadamente, não só o de permitir deslocações com motivações variadas (além de casa-trabalho) como o de permitir interconexões com outros meios de transporte, como o metro e o comboio.

Em geral, cada linha pertence apenas a uma zona, sendo a excepção as linhas da zona Circulares. Estas zonas não são totalmente estanques, pois cada uma está conectada, através da sua fronteira permeável, às zonas adjacentes; assim, algumas das linhas de cada zona entram ligeiramente na zona adjacente para permitir a comunicação entre zonas, na fronteira.

Pode notar-se ainda que cada zona tem dois tipos de linhas: as linhas grandes, que atravessam a zona e cujos autocarros andam nas duas direcções da linha, e as linhas pequenas (compostas por autocarros pequenos, também denominadas linhas de bairro), que são ciclos com direcção, isto é, que começam e terminam no mesmo sítio e cujos autocarros andam sempre na mesma direcção circular (no sentido das horas ou contrário).

A rede da Carris tem 1701 paragens; no entanto, o total de paragens (somando o número de paragens de cada linha) é de 2935. Se se pretendesse complexificar (aproximando a análise da realidade), seria necessário entender esta rede não só como um conjunto de pontos ligados entre si por diversas linhas, mas também as conexões entre essas linhas, o que corresponderia a atribuir três dimensões a cada ponto – duas dimensões geográficas e uma terceira que indicasse o número de linhas que passam nesse ponto – e a considerar que uma mudança de linha corresponde a uma parte da viagem onde se consome tempo (de espera) mas não distância.

Considerando, então, o tamanho e complexidade da rede, como trabalho exploratório escolheram-se duas paragens para cada linha – a primeira e última. Desenhou-se em seguida o grafo correspondente, obtendo-se uma imagem muito grosseira da rede actual, num total de 218 pontos conectados por 109 linhas[58] (Anexo I), onde já é possível observar algumas das características da cidade e, consequentemente, da rede Carris: espaços vazios ou sem linhas (correspondentes a zonas como Monsanto ou o aeroporto de Lisboa, locais onde há poucas estradas ou onde não há necessidade de transporte) e uma estrutura de transportes tão mais concentrada (número de paragens) e conectada (número de linhas) quanto mais próxima do centro; é possível observar também pontos de maior conectividade (tais como o Marquês de Pombal, Sete Rios e Cais do Sodré) que correspondem a sítios onde passam mais linhas ou que correspondem ao início de várias linhas ou sítios de conexão com outros meios de transporte. Pretendia-se ir complexificando esta construção, construindo-se grafos com cada vez mais pontos e cuja representação se aproximaria cada vez mais da rede real, com todas as suas conexões, cruzamentos e circularidades (Figura 4), ganhando, assim, conhecimento sobre a rede.

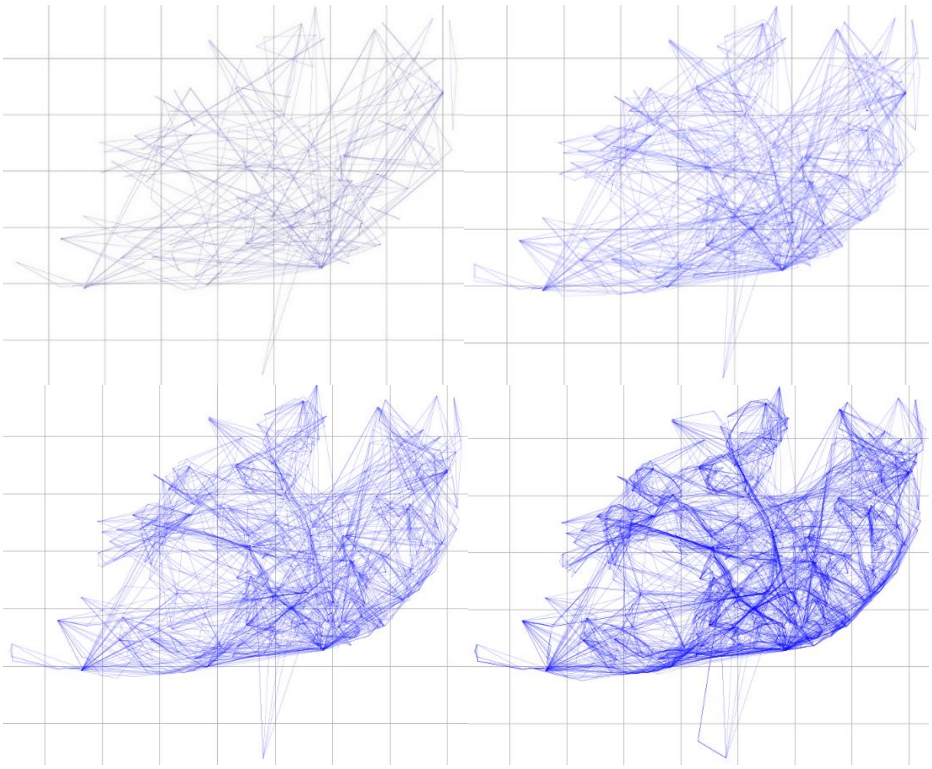


Figura 4: Evolução fractal da rede Carris

[58] A informação de teor quantitativo, referida ao longo deste trabalho, diz respeito aos dados recolhidos sobre a Rede entre junho e julho de 2023.

Inicialmente pensou-se analisar – e encontrar uma solução – para o conjunto das paragens de Lisboa, que representavam um total de 1701 paragens (distribuídas por 109 linhas, ver Figura 5).

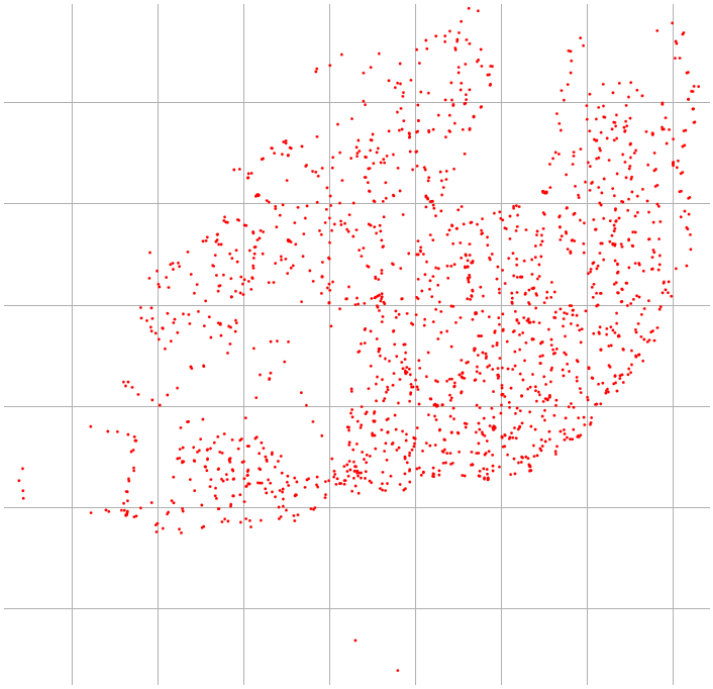


Figura 5: Localização das paragens – rede Carris

Verificou-se, no entanto, que a dimensão do que estava a ser proposto – analisar com algumas das ferramentas disponíveis um grafo que representasse a cidade de Lisboa – era inviável, mesmo considerando todas as simplificações possíveis[59]. Assim, e tendo em conta que a própria Carris divide a cidade em diferentes zonas, foi decidido concentrar a análise apenas numa das zonas. Atente-se a Tabela 1, com os valores do número de linhas e paragens (não únicas), por zona:

[59] Apenas para dar uma ideia desta dificuldade, o melhor modelo treinado por um dos algoritmos de AR experimentados (Peng et al., 2020), cujos autores referem poder competir com o *solver* da Google (OR Tools, que não tem inteligência artificial na sua concepção), tinha sido treinado para responder a problemas com um máximo de 50 vértices; também alguns testes exploratórios simples feitos com o OR Tools revelaram fragilidades quando o número de vértices era superior a 100 – por exemplo, para um teste feito com 150 vértices e 15 linhas de forma a aproximar os números da zona que se pretendia modelar, este *solver* devolvia soluções em que muitos dos transportes não ligavam qualquer vértice.

Tabela 1: Número de linhas e paragens por zona

Zona	cor da zona	# linhas	# paragens
Bairro	amarela	28	586
Noroeste	azul	14	382
Nocturna	azul escura	6	276
Circulares	cinzenta	11	423
Centro	laranja	19	310
Ocidental	rosa	9	252
Norte	verde	7	176
Oriental	vermelha	15	457

A linha Nocturna e a linha de Bairro não foram consideradas na selecção da zona a analisar: uma, por ser categoria temporal e não geográfica; outra, porque é composta por linhas circulares, o que levantava problemas no tratamento ("qual a distância entre dois pontos num percurso circular?") e análise dos dados (os ciclos representam uma dificuldade acrescida na modelação destas situações[60]). Considerando estes fatores, foi seleccionada a zona Norte, uma zona com relativamente menos linhas e paragens – apenas 7 linhas e 176 paragens, que correspondem a 144 locais de paragem únicos. O conjunto de pontos analisados pode ser visto no Anexo II.

Extracção e Tratamento dos Dados

A extração de dados, automatizada e com qualidade, revela-se determinante para conduzir uma análise consequente e estatisticamente significativa. A informação foi recolhida principalmente dos sítios do GoogleMaps[61] e da Carris[62] durante junho 2023, tendo sido utilizadas técnicas de *scraping* (com recurso às bibliotecas *beautifulsoup* e *selenium*) para automatizar a recolha dos dados. Além destas técnicas, foi utilizada também a API da Google – através da qual muita informação é disponibilizada – nomeadamente para o cálculo da "matriz distância" e da "matriz duração"[63] – informação essencial para conduzir a análise pretendida.

[60] A existência de ciclos pode implicar um sistema menos eficiente, porque estes representam uma redundância: num ciclo todos os pontos estão ligados por dois caminhos diferentes. Muitos algoritmos não admitem a existência de "ciclos" – durante o treino o algoritmo esconde ("mascara") todas os pontos já percorridos pelo agente.

[61] https://www.google.com/maps

[62] https://carris.pt/viaje/carreiras/ e https://carris.pt/viaje/mapas/

[63] Ambas as matrizes pretendem descrever um grafo. Neste caso em particular, a matriz distância fornece a informação sobre a distância entre cada dois pontos (dependendo do problema, em linha recta ou seguindo as estradas existentes); a matriz duração fornece a informação sobre o tempo que demora se demora a ir de um ponto ao outro entre cada dois pontos (esta informação já tem em conta, por exemplo, o tráfego existente à hora a que se recolheu a informação)

Foram recolhidos dados sobre cada uma das linhas de autocarro existentes na cidade, nomeadamente, as paragens de cada linha, a localização geográfica (latitude e longitude) de cada paragem, a sua identificação unívoca (através de um *id* único disponibilizado pela Google) e o URL associado, o seu nome (no sítio da Carris cada paragem tem um nome, que não é único), o tipo de paragem e a situação em que se encontra (no sítio do GoogleMaps é possível encontrar informação sobre o tipo de paragem – estação, terminal, garagem, etc. – e sobre a sua situação – activa ou inactiva) e os horários de cada um dos autocarros em cada paragem. Foi também calculada a distância em linha recta, a distância por estrada e a duração da viagem entre cada par de paragens (construiu-se, assim, um grafo completo (Kool et al., 2018)).

Após a recolha da informação, deu-se seguimento à sua compilação em diversos ficheiros Excel, de forma a poder ser observada com mais facilidade e proceder à sua limpeza – o trabalho de recolha de dados através de técnicas de *scraping* gera muitas vezes erros, relacionados com a falta de informação pretendida em determinada página, ou relacionados com dificuldades de acesso na altura da recolha da informação (por exemplo, devido à alteração da estrutura do HTML ao ir seleccionando automaticamente páginas diferentes do mesmo sítio). A limpeza dos dados consistiu na eliminação de duplicados e na correção de algumas referências *id* das paragens, que apresentavam algumas incongruências (ao solicitar valores de *id* através da API da Google, alguns valores devolvidos pela API eram claramente não aleatórios, pelo que foi necessário proceder a nova recolha dessa informação).

Para o caso particular deste trabalho, a limpeza também requereu a eliminação das paragens muito próximas umas das outras (que correspondem, ou a um mesmo sítio que tem várias paragens, como é o caso dos locais com maior conectividade – Sapadores, Cais do Sodré, Marquês de Pombal, Estação do Oriente, etc. –, ou a estações que se encontram na mesma rua, em direcções diferentes). Embora não seja absolutamente preciso, foi usada a distância em linha recta para eliminar paragens muito próximas umas das outras, tendo-se escolhido, de entre as paragens a menos de 50 metros umas das outras, apenas uma.

Após a limpeza dos dados, calcularam-se algumas métricas sobre a rede, tais como o número de linhas, ou o número de paragens por linha e por zona[64], métricas essas que permitiram não só ter uma visão mais geral da estrutura de rede como também fazer as escolhas subsequentes já mencionadas, quanto a que zona escolher para continuar a análise.

[64] Uma das informações consideradas relevantes nesta fase foi o número de autocarros a circular numa determinada hora do dia, por cada linha. Infelizmente, por motivos de complexidade técnica na análise da informação disponível *online*, não foi possível calcular esses valores.

Bibliotecas Usadas

Considerou-se utilizar algumas das implementações práticas desenvolvidas nos artigos previamente mencionados (e disponíveis *online*). Assim, de forma a tentar perceber se seria possível adaptar código desenhado para a modelação de um problema diferente, mas semelhante, exploraram-se os algoritmos (e respectivo código) desenvolvimentos por Kool et al. (2018), Nazari et al. (2018), Peng et al. (2020) e Berto et al. (2023)[65].

Os modelos desenvolvidos por Kool et al. (2018) e Nazari et al. (2018) que estão disponíveis na plataforma *GitHub*[66] estão atualmente desatualizados. No primeiro caso, vários dos diferentes ficheiros não foram alterados nos últimos cinco anos, no caso dos restantes dois desenvolvimentos, os modelos estão dependentes, por exemplo, da versão Python 3.6 e Pytorch 0.4.1.

Relativamente ao trabalho desenvolvido por Peng et al. (2020), que tal como já referido partem do trabalho de Kool et al. (2018) tornando dinâmico o *encoding* das localizações, não tendo sido possível treinar novos modelos com o código disponibilizado pelos autores[67], foram feitas várias experiências com os modelos pré-treinados. No entanto, relativamente a esta abordagem, notem-se dois aspectos: por um lado, os modelos pré-treinados tinham-no sido para ambientes com o máximo de 50 pontos e, embora os autores referissem que os modelos pré-treinados poderiam funcionar para mais pontos, os resultados obtidos para cerca de 150 pontos (um número mais próximo da situação real, que se queria modelar) foram desanimadores. Foi levantada (mas não testada) a hipótese de que estes modelos pré-treinados apresentaram bons resultados para distribuições uniformes de pontos (isto é, em ambiente semelhante ao de treino), mas que poderão não estar a generalizar bem para situações em que a distribuição não é uniforme (tal como é o caso da distribuição de paragens de autocarro em Lisboa). Assim, esta abordagem foi preterida.

Considerando tudo o que já foi referido, foi escolhida, por fim, a arquitectura desenvolvida por Berto et al. (2023) – arquitectura essa que corresponde a uma biblioteca denominada RL4CO (*reinforcement learning for combinatorial optimization*), disponível no *GitHub*[68]. Foi escolhida esta arquitectura de trabalho, não só por estar facilmente disponível, atualizada, ser a implementação mais recente e estar activa (no sentido em que tem utilizadores que a promovem, monitorizam, testam e desenvolvem), mas

[65] Além das implementações que aqui se referem, outras houve que se pretendia explorar, ou por apresentarem resultados mais promissores ou por dizerem respeito a modelos treinados em ambientes mais complexos e por isso mais interessantes; no entanto, alguns autores, como é o caso, por exemplo, de Darwish et al. (2020) não disponibilizam *online* o código que deu origem ao respectivo artigo.

[66] https://github.com/wouterkool/attention-learn-to-route; https://github.com/OptMLGroup/VRP-RL

[67] https://github.com/d-eremeev/ADM-VRP

[68] https://github.com/ai4co/rl4co/tree/main

também, e principalmente, porque representa, ao contrário das outras referidas, a tentativa de construir um *framework* que possa ser usado e desenvolvido por todos (*open-source*), em vários problemas distintos, de aplicação o mais geral possível.

Ou seja, ao contrário das outras implementações, cujo código foi desenvolvido para responder a um problema específico (VRP, neste caso) e que se tentou adaptar, sem sucesso, ao âmbito desta tese, a biblioteca RL4CO permitiu construir código de raiz e realizar com relativa facilidade[69] várias experiências com modelos e hiperparâmetros diferentes, que puderam ser comparadas. Com efeito, esta biblioteca, que é um *wrapper* à volta de Pytorch, permite a criação de ambientes diferentes e o controlo de vários aspetos do processo de treino e de teste. Nas palavras dos autores:

> *RL4CO [is] a unified and extensive benchmark with in-depth library coverage of 23 state-of-the-art methods and more than 20 CO problems. (…) RL4CO allows researchers to seamlessly navigate existing successes and develop their unique designs, facilitating the entire research process by decoupling science from heavy engineering*. (Berto et al., 2023, p.1)

Esta biblioteca tem ainda a vantagem de poder ser paralelizável, já que é baseada em TorchRL, o que permite a execução de ambientes e algoritmos com recurso a GPU (ao contrário dos ambientes que têm como base o Open AI Gym).

Algoritmos de AR

Por tudo o que já foi referido, e, nomeadamente, notando que a experiência de Kool et al. (2018) é uma das mais citadas, tendo sido das primeiras a usar mecanismos de atenção para resolver este tipo de problemas, foi decidido que se usaria uma arquitectura semelhante à construida no artigo (*encoder-decoder* com mecanismos de atenção) com uma *baseline* diferente[70]; em particular, irá ser usada a arquitectura denominada MVMoE – *Multi-task Vehicle Routing Solver with Mixture-of-Experts*, proposta por J. Zhou et al, (2024) que incorpora uma mistura de especialistas nos mecanismos de atenção e que permite melhorar a eficiência do modelo sem o aumento da capacidade computacional; para construir a solução usar-se-á *greedy decoding* (no seguimento das experiências realizadas por Nazari et al. (2018), que não revelaram diferenças significativas entre *greedy decoding* e *beam search* quando o número de pontos do problema aumentava).

---

[69] A facilidade é de facto relativa: nas palavras de Frederico Berto este *framework* tem uma curva de aprendizagem inclinada, mas que se torna rapidamente mais plana; essa é também a experiência tida com este trabalho.

[70] *Baseline*: corresponde a um valor de referência usado para estabilizar o processo de aprendizagem, ao permitir reduzir a variância no processo de estimação do *policy gradient*, melhorando a convergência com a política ótima. A *baseline* usada em Kool et al. (2018), que vai variando ao longo do tempo, é mais complexa do que a que foi usada nas experiências descritas nos próximos capítulos – que corresponde a uma *baseline* partilhada.

Assim, foram realizadas duas experiências com o MVMoE, um algoritmo *policy-based* (visto que se foca directamente na otimização de uma política e não na função valor), *on-policy* (através do uso de *baselines*, a política executada é a política avaliada) com avaliação de Monte Carlo (em que se espera que os episódios terminem para fazer a avaliação).

Configuração Experimental

Neste capítulo, irá ser detalhada a configuração experimental utilizada para validar os métodos e modelos propostos. Abordar-se-á o ambiente de simulação, incluindo a descrição do ambiente gerado para a aprendizagem do agente, bem como as métricas de avaliação escolhidas por forma a comparar os modelos. Além disso, serão descritos os parâmetros de configuração dos algoritmos, bem como as ferramentas e tecnologias utilizadas no desenvolvimento e na execução das experiências. O desenho adequado destes componentes é essencial já que tem impacto directo na eficácia e eficiência do algoritmo ao longo do processo de aprendizagem.

O treino dos modelos foi realizado na plataforma Google Colab; realizaram-se alguns testes exploratórios para decidir que GPU usar, não se tendo encontrado diferença significativa no tempo de treino ao fazer o GPU variar. Assim, todas as experiências foram conduzidas com recurso a um acelerador de hardware NVIDIA T4 Tensor Core GPU. Quanto ao software, todo o código foi escrito em Python, versão 3.10; o registo das experiências foi feito com recurso à plataforma Weights & Biases e à ferramenta TensorBoard.

Foram realizadas duas experiências onde se fez variar o número de épocas de treino – quatro épocas na primeira experiência e dez na segunda; assim, a configuração dos diferentes estados que o ambiente poderia assumir, as diferentes acções que o agente poderia executar e o cálculo da recompensa por cada época não variaram entre experiências. Foram utilizadas, em ambas as experiências, 10.000 instâncias (problemas de VRP, gerados aleatoriamente) de treino, 10.000 instâncias de teste e 1.000 instâncias para validação, o que proporcionou uma base sólida para avaliar a performance do algoritmo proposto.

Dinâmica de Decisão

A resolução de um VRP aplicado à rede de transporte público de Lisboa envolve a modelação de vários fatores, tais como a recolha e entrega de passageiros, a definição das localizações, a capacidade dos veículos, entre outros parâmetros. Para este estudo, foi utilizado o ambiente fornecido pela biblioteca RL4CO, adaptado ao problema de transporte de passageiros. Abaixo são descritos os principais componentes e a configuração do ambiente de simulação.

Estados do Ambiente de Simulação

Para realizar as experiências foi escolhido o ambiente gerados para o VRP com entregas e recolhas (*Vehicle Routing Problem with Pickups and Deliveries* – VRPPD), já que este foi o ambiente disponível na

biblioteca que melhor se adaptava ao objectivo deste trabalho – a modelação do transporte de passageiros em ambiente urbano – e onde se poderia incluir a definição de constrangimentos quanto à carga[71], onde cada rota pode ser "fechada" ou "aberta" (isto é, que retorna ou não ao ponto de partida) e incluir uma mistura de entregas e recolhas (em qualquer ordem que respeite a regra de que os passageiros têm de ser recolhidos antes de ser entregues) e a definição de janelas temporais (que no caso deste trabalho não foram usadas por motivos que serão sucintamente expostos). Estas especificações adequam-se à realidade que se pretende modelar: os autocarros da Carris precisam de recolher passageiros em várias localizações e entregá-los em múltiplas outras, os autocarros têm capacidade limitada e há um horário diário, por localização, que cada veículo deve cumprir.

A configuração das experiências seguiu as seguintes definições[72]:

- *Número de localizações*: determina quantos pontos devem ser visitados pelos veículos. Por simplificar o processo de aplicação dos algoritmos aos dados reais[73], as experiências foram realizadas para 144 localizações – correspondentes ao número de localizações únicas na zona Norte de Lisboa.
- *Localização dos pontos*: informação que é gerada aleatoriamente no início do treino; por motivos de eficiência e uniformidade no processo de treino, todos os pontos são gerados com distribuição uniforme no quadrado unitário $[0, 1]^2$.
- *Posição dos veículos*: informação dinâmica, que vai sendo atualizada à medida que o agente interage com o ambiente, isto é, à medida que o agente escolhe onde ir a seguir.
- *Capacidade dos veículos*: Configurados de forma estática, com capacidade máxima de 100 pessoas, valor com que se pretende refletir o cenário do transporte público em Lisboa.
- *Capacidade restante dos veículos*: informação dinâmica, atualizada ao longo do percurso, indicando a quantidade de pessoas que os veículos ainda podem levar.
- *Procura de serviço*: o número de pessoas que o veículo tem que levar em cada localização, bem como o seu local de destino são informações geradas aleatoriamente para cada ponto, com limite entre um mínimo (que foi pré-definido como sendo igual a 1) e um máximo (pré-definido para 20).

[71] Podem ser: limites de peso, limites de volume, restrições quanto à ordem em que os bens são carregados ou descarregados, compatibilidade entre produtos, entre outros.

[72] Ao nível do código, o estado está codificado num *TensorDict*, geralmente denominado "td".

[73] A aplicação destes algoritmos a situações concretas, do dia-a-dia, implica necessariamente que estes possam ser usados em situações com um número de localizações variável e, consequentemente, em situações com um número de pontos diferente daquele para o qual foram treinados. No entanto, abordar esta questão gera desafios técnicos cuja resolução está fora do âmbito deste trabalho.

- *Matriz Distância* ou *Matriz Duração*: no caso presente, estas matrizes não fizeram parte da definição do ambiente; no entanto, a definição de uma destas matrizes é muito comum neste tipo de problemas; aliás, muitos dos algoritmos não incluem a definição das localizações geográficas (latitude e longitude), mas apenas as distâncias entre os pontos.
- *Pontos já visitados*: informação dinâmica, que vai sendo atualizada ao longo do processo de treino; os pontos já visitados são mascarados pelo algoritmo de forma a não poderem ser escolhidos novamente.
- *Janelas temporais*: correspondem ao constrangimento de tempo para cada localização. Foram realizadas algumas experiências exploratórias através da definição de janelas temporais para cada localização, tentando mimetizar o que acontece na realidade, onde cada veículo chega a cada uma das localizações a uma determinada hora; no entanto, percebeu-se que não era possível estabelecer estes horários sem influenciar enormemente as escolhas do agente ao longo do processo de treino[74], pelo que no caso das experiências efectuadas as janelas temporais não foram definidas.
- *Velocidade dos veículos*: por simplicidade, a velocidade foi definida como "1" para todos os veículos.

As localizações dos pontos, a localização do *depot,* a quantidade de passageiros em cada local e o ponto para onde se dirigem foram gerados aleatoriamente para criar um conjunto diversificado de cenários.

Espaço de Acção

O espaço de acção refere-se às possíveis escolhas que o agente pode fazer em cada passo. No presente caso, o espaço de acção é bastante pequeno, o que reduz a complexidade computacional do problema. Neste caso, o agente deve selecionar uma das localizações não visitadas (exceto o *depot*). Em ambientes com janelas ou limites temporais, o agente poderia também optar por esperar numa localização específica.

Função de Recompensa

[74] A tendência é para definir janelas temporais muito próximas e assim constranger artificialmente as escolhas do agente. Por exemplo, se o veículo tiver que estar no ponto A às 17:00 e no ponto B às 17:03 o algoritmo vai seguir naturalmente esse caminho, mesmo que esse não seja o caminho mais eficiente, que é o que interessa aqui descobrir.

A definição de um objectivo é central para a aprendizagem. Assim, quanto à função de recompensa, note-se que é necessário definir o que é "a melhor rota". Se o objectivo do algoritmo fosse apenas encontrar a rota com a menor distância total, sem qualquer outro constrangimento, a solução ótima corresponderia a usar apenas um veículo que unisse numa única rota todas as localizações, reduzindo-se o problema ao TSP. Uma forma comum de definir o objectivo no caso do VRP é que o agente minimize o comprimento da rota mais longa de entre todas as rotas, cumprindo as restrições apresentadas. Foi esta a função objectivo definida, considerando-se então a recompensa como o negativo da distância, o que penaliza rotas mais longas e incentiva uma distribuição mais equilibrada entre os veículos.

Selecção de Parâmetros e Hiperparâmetros

Verificou-se que quando se treinava um modelo para um número de localizações já na ordem das centenas (por exemplo, 150) com um *batch size* maior (por exemplo, de 128 ou 256) o GPU não suportava o número de cálculos que era necessário efectuar; assim, foram sendo realizadas experiências sucessivas, baixando cada vez mais este hiperparâmetro até se conseguir que o processo de treino terminasse. Foi então necessário baixar significativamente este hiperparâmeto, tendo o processo estabilizado para um um *batch size*[75] de oito instâncias, bastante mais baixo que o visto em várias outras experiências. A definição deste hiperparâmetro implicou a redução do número de épocas[76] de treino, visto que quanto mais baixo era o *batch size*, mais tempo demorava o treino. Assim, foram treinados dois modelos, tendo-se feito variar o número de épocas de treino: o primeiro modelo foi treinado durante 4 épocas e demorou cerca de 4 horas a ser treinado e o segundo foi treinado em 10 épocas, tendo o seu treino demorado cerca de 9 horas. O número de épocas pode ser considerado baixo, tendo em conta outros exemplos observados e o tamanho do espaço de soluções.

[75] No caso a que nos referimos, *batch size* corresponde ao número de episódios guardados antes de atualizar a política ou a função valor. Um episódio corresponde a uma sequência completa de interacções agente – ambiente (isto é, uma sequência de passos em que cada passo envolve escolher um ponto, deslocar-se para esse ponto, receber uma recompensa e transitar para um novo estado), do estado inicial do ambiente (nenhum ponto visitado, veículo na localização *depot*) até ao seu estado final (todos os pontos visitados, veículo no *depot* ou no último ponto visitado). Note-se que em muitos algoritmos, o *batch size* diz respeito ao número de amostras (vetores estado-acção-recompensa) usadas num *gradient descent* durante o processo de otimização da política ou função valor.

[76] Época é o conceito usado para estruturar o treino. Cada época é um ciclo de treino por parte do agente; corresponde, geralmente, a um conjunto de episódios sempre realizado com a mesma política. Após completar uma época a experiência acumulada é usada para atualizar a política do modelo (ou, noutros casos, a sua função valor) (Sutton & Barto, 2015).

Foi sempre usado o *Adam optimizer*, com taxa de aprendizagem (*learning rate*)[77] de 0.0001 e *weight* decay[78] de 1e-06.

Infraestrutura Computacional e Treino

Modelo composto pelo ambiente (já descrito anteriormente) que providencia a interface com que o agente interage através da sua política, que é a estrutura responsável por decidir que acções tomar considerando o estado do ambiente.

A política da nossa experiência é uma rede neuronal composta por duas partes: um *encoder* e um *decoder*. O *encoder* produz codificações (ou vectores de representação, *embeddings*) para cada ponto; o *decoder* tem como *input* o *output* do *enconder* e é a rede neuronal responsável por produzir a solução na forma de uma sequência de pontos.

No Anexo III encontra-se uma descrição exaustiva desta rede neuronal; dessa descrição fez-se um resumo dos aspectos mais significativos, que pode ser lido em baixo.

Encoder

O *encoder* é composto por duas camadas.

A camada inicial, totalmente conectada (*fully connected layer*), que trata da codificação das localizações para um espaço de maior dimensão; cada localização (que tem, inicialmente, 7 dimensões) é transformada num vetor de representação com 128 dimensões (o que corresponde a 7*128 + 128 = 1152 parâmetros); o *depot* (que tem apenas 2 dimensões espaciais) é também transformado por uma camada diferente num vetor de representação com 128 dimensões (o que corresponde a 2*128 + 128 = 384 parâmetros). Assim, a partir de *inputs* de dimensão reduzida o *encoder* calcula uma representação de muito maior dimensão através de uma projecção linear aprendida.

A segunda camada, que é uma camada sequencial, é uma Rede de Atenção em Grafos (*Graph Attention Network*), que processa e atualiza o *output* da camada anterior usando 5 camadas de atenção com múltiplas cabeças (*multi-head attention layers*). Cada uma destas camadas de atenção é composta por uma sequência de quatro sub-camadas: *SkipConnection -> Normalization -> SkipConnection ->*

[77] A taxa de aprendizagem é um hiperparâmetro usado no processo de treino para determinar o tamanho da deslocação do valor de *input*, de forma a optimizar a função de erro (*loss function*) em métodos que usam gradient *descent*; taxas de aprendizagem maiores implicam adaptações dos pesos mais rápidas, o que pode provocar instabilidade no treino. (Goodfellow et al., 2016)

[78] Hiperparâmetro usado no processo de treino para controlar a tendência de um modelo para se ajustar excecivamente (*overfitting*) ou insuficientemente (*underfitting*). (Goodfellow et al., 2016)

Normalization. Na primeira *SkipConnection* está implementado o módulo *multi-head attention layer*; na segunda *SkipConnection* está implementado o módulo *Mixture of Experts*[79], um módulo específico para o tipo de ambiente que se usou nesta experiência, que permite um aumento da capacidade do modelo sem aumento dos níveis de computação; este módulo substitui a mais tradicional camada linear final dos mecanismos de atenção (Berto et al., 2023, p.35). Além desta perspectiva geral sobre o modelo, podem ainda ser referidas outras camadas e subcamadas (porventura, secundárias) tais como de normalização e de activação e funções, tais como *softmax* e *softplus*.

Decoder

O *decoder* possui uma camada inicial, linear, que codifica a informação sobre o contexto do ambiente num vector de representação com 128 dimensões; possui também uma segunda camada que tem como objectivo codificar as características dinâmicas (isto é, aquelas que vão sendo alteradas ao longo do treino), tal como a capacidade restante do veículo ou a informação sobre os pontos já visitados, num total de 5 dimensões. Cada uma destas 133 dimensões é posteriormente transformadas em vetores de 128 dimensões (o que corresponde a um total de 133*128 = 17024 parâmetros).

Além destas duas camadas, o *decoder* tem implementado na sua estrutura um mecanismo *pointer*, importante para tarefas de geração em sequência (tal como a de escolher o ponto seguinte do caminho) e, novamente, mais um módulo *Mixture of Experts* que corresponde, neste caso, a 4 camadas lineares que não alteram a dimensão do *input* (mapeiam vetores de 128 dimensões em vetores de 128 dimensões); este módulo possui duas funções: uma função de activação, *softplus* e uma outra denominada *softmax*. Esta função *softmax* aplicada aqui garante que o *output* seja uma distribuição de probabilidades sobre o conjunto de pontos seguintes a serem seleccionados.

Resumindo, o modelo da experiência tinha as seguintes componentes:

- *Encoder*: cerca de 3.000.000 de parâmetros
  - Usa *Graph Attention Network* com múltiplas *Multi-Head Attention Layers*.
  - Cada camada usa *skip connections*, *normalization* (InstanceNorm1d), e um mecanismo de *mixture of experts* (MoE).
- *Decoder*: cerca de 150.000 parâmetros
  - A representação do contexto e o *pointer attention* usam MoE para seleccionar a rede especialista para o *decoding*.

[79] Este módulo é incorporado apenas neste modelo específico

- *Baseline*:
    - Foi utilizada uma *baseline* partilhada, ou seja, um valor que não foi alterado ao longo de todo o processo de treino.

Métricas de avaliação e Modelos de Referência para Comparação

De forma a ser possível avaliar as soluções e ter uma medida de comparação, analisou-se a possibilidade de aplicar outros métodos ao problema em questão: nomeadamente, foi examinada a hipótese de usar implementações *open-source* tais como o OR Tools e o PyVRP. No entanto, considerou-se que utilizar este *software* não fazia parte dos objectivos desta tese, pelo que se decidiu por abordagem mais simples.

Usou-se, assim, o algoritmo denominado "algoritmo de economias de Clarke e Wright"[80] (*Clarke and Wright Savings algorithm* – algoritmo Savings), comparando a solução encontrada por este algoritmo com a solução encontrada pelo modelo AR treinado. Para o efeito, foi usada a implementação do algoritmo feita em Peng et al. (2020), cujo código foi disponibilizado pelos autores na plataforma *GitHub*. Este algoritmo não incorpora qualquer mecanismo de aprendizagem automática e corresponde a uma abordagem heurística para a resolução do VRP, no caso de o número de transportes não estar fixo, pelo que é um método distinto do que foi experimentado nesta dissertação.

Darwish et al. (2020) definem várias métricas que têm a vantagem de permitir uma avaliação dos modelos relativamente a vários interesses, muitas vezes contraditórios, tendo essas métricas a capacidade de tornar transparente e ponderar os fatores que se consideram mais importantes ao construir qualquer modelo. Considera-se que o cálculo dessas métricas relativamente aos modelos que foram aqui treinados está fora do âmbito da tese (porque a função recompensa usada no caso do treino presente é muito mais simples que a que os autores usaram), mas pelo seu interesse vale a pena referi-las:

- Tempo médio de viagem para todos os passageiros satisfeitos (em que incluem uma penalização por cada mudança de veículo;
- Percentagem de viagens que conseguem ser completadas com um máximo de 2 mudanças de veículo;
- Tamanho da frota necessária para a rede.

Tentando realizar uma avaliação mais concreta e encontrar métricas mais directamente relacionadas com o problema que se pretende estudar, os diferentes modelos forem depois comparados quanto ao

[80] Originalmente desenvolvido por (Clarke & Wright, 1964).

número de linhas (rotas) que a solução encontrou, a distância total da rede, a distância do maior troço, e a duração do percurso da rede.

Resultados e Discussão

Além de todas as simplificações tornadas explícitas nos capítulos anteriores – que dizem respeito à construção do modelo – foi necessário também proceder a simplificações no momento de aplicação do modelo: para alcançar um resultado final e poder ter matéria com que comparar a realidade da rede Carris com as soluções encontradas, foi ainda necessário definir o ponto de partida – o *depot* – dos veículos. A simplificação não tem a ver com a definição de um ponto de partida, porque ele existe na realidade (os veículos retornam a um sítio à noite, donde partem novamente no dia seguinte) e, embora não tenha sido possível confirmar esta informação, não é exagerado supor que por cada zona da cidade deverá haver algum sítio onde os veículos ficam guardados durante a noite. A assunção feita é de que o *depot* é um ponto central relativamente a todos os pontos da zona, isto é, que está na localização média dos pontos da zona – latitude e longitude correspondente à média das latitudes e longitudes de todos os pontos. Assim, não foram consideradas as distâncias entre o *depot* e os restantes pontos ao comparar os resultados alcançados com os do algoritmo *Savings* ou os da rede Carris. Ou seja, visto que para calcular o comprimento de cada linha da rede Carris não se considerou a distância entre o *depot* e a primeira e a última paragem de cada linha, essa distância também não foi considerada nos modelos, tendo esse troço sido ignorado tanto nos cálculos como nas figuras apresentadas.

Também não foi possível modelar convenientemente a procura e oferta de cada localização (isto é, quantas pessoas esperavam o veículo em cada paragem e para que sítios eram levadas); foi assumido, por simplicidade, que metade das localizações eram sítios de onde as pessoas partiam e a restante metade, sítios para onde as pessoas iam; esta assunção implica que também se assumiu que se há pessoas a ir da paragem A para a paragem B, não há pessoas a ir de B para A. Como é evidente, estas simplificações afastam-se da realidade e tornam os modelos menos capazes de resolver problemas concretos. Apesar de ser fácil pensar numa solução para este problema (por exemplo, assumindo – como de facto acontece – que há paragens em diferentes lados da rua e que devem ser ligados por linhas diferentes) já não é tão linear pensar de que forma é que é possível mascarar os pontos que se encontram do outro lado da rua (para poder aplicar estes algoritmos a situações concretas, será sempre necessário mascarar essas localizações, visto que do ponto de vista do algoritmo fará sentido ligar essas localizações com a mesma linha, já que estão tão próximas, o que evidentemente não faz sentido). Teria que ser necessário, por exemplo, mascarar não só os pontos já visitados, mas também os que estão extremamente próximos (por exemplo, a menos de 100 metros) do local onde se encontra o veículo.

Quanto às experiências efectuadas, obtiveram-se os seguintes resultados:

- *Experiência 1 (Modelo 1, 4 épocas)*: obteve-se, em treino, um retorno (*reward*) médio de -16.42 por episódio e, em teste, um retorno médio de -15.36 por episódio.
- *Experiência 2 (Modelo 2, 10 épocas)*: obteve-se, em treino, um retorno (*reward*) médio de -14.70 por episódio e, em teste, um retorno médio de -14.35 por episódio.

O facto de ambos os modelos terem um resultado muito semelhante em teste e em treino indica que não houve *overfitting*, pelo que os modelos poderão generalizar bem a problemas novos. Note-se ainda que o Modelo 2, treinado durante mais épocas, teve um resultado melhor quanto ao retorno, isto é, conseguiu em média encontrar caminhos mais curtos.

Quanto à comparação entre os resultados obtidos com estas experiências e a solução encontrada com recurso ao algoritmo *Savings* (que foi implementado com base no código de Peng et al. (2020) e cuja imagem de rede pode ser consultada no Anexo IV), pode afirmar-se que ambos os modelos treinados são significativamente melhores que este algoritmo quando avaliados de acordo com as métricas definidas anteriormente: no que concerne a distância total da rede em linha recta o algoritmo *Savings* devolveu uma solução com cerca de 240 km, muito distante da solução de ambos os modelos que foi de cerca de 60 km; por fim, a distância do maior troço em linha recta dada pela solução *Savings* foi de cerca 9 km enquanto aquela dada pelos modelos treinados foi de cerca 2 km. Verifica-se, então, que o algoritmo *Savings* teve uma performance abaixo do esperado e com valores muito piores (isto é, mais altos) que os da rede Carris ou o das soluções encontradas pelos modelos treinados.

Embora não tenha sida analisada a causa de resultados tão discrepantes, os maus resultados alcançados com este algoritmo poderão ter a ver com o facto do problema que se está a tentar resolver ser relativamente grande, ou com o facto da distribuição de pontos não ser uniforme (dado que a qualidade de solução para este algoritmo dependen um pouco das condições iniciais) ou ainda com o facto de as rotas que vão ser fundidas estarem previamente fixas com base na lista de economias, o que pode ser restritivo (Laporte & Semet, 2002).

Apesar disso, o algoritmo *Savings* apresentou uma vantagem relativamente às experiências: foi possível, com este algoritmo, encontrar uma solução com o mesmo número de linhas que a rede Carris. Infelizmente, a questão da definição do número de linhas afigurou-se difícil aquando da concretização das experiências, o que teve a ver com o facto de não haver ambientes em que esse parâmetro pudesse ser facilmente pré-definido. Com o algoritmo *Savings* podia definir-se o número de linhas pretendido.

Pode consultar-se em baixo a Tabela 2, onde se apresentam estes resultados (página 51).

Aplicação do Modelo à Rede Carris

Muitos dos algoritmos encontrados ao longo da pesquisa efectuada foram testados em "situações perfeitas", como é o caso da rede de Mandl's[81], cuja avaliação serve de *benchmark* para muitos modelos; considerou-se neste trabalho – e seguindo alguma da bibliografia lida – que a aplicação do modelo e a tentativa de encontrar boas soluções para problemas concretos é mais valiosa do que ter bons resultados em situações idealizadas. Assim, apresentam-se em seguida os resultados alcançados pelo modelo a partir de uma situação concreta, aproximada da realidade da rede da Carris, em Lisboa, e a avaliação desses resultados com base nas métricas específicas definidas anteriormente.

O conjunto de rotas definidas pela Carris (rede Carris) bem como as localizações das paragens para a zona Norte de Lisboa podem ser vistas na Figura 6.

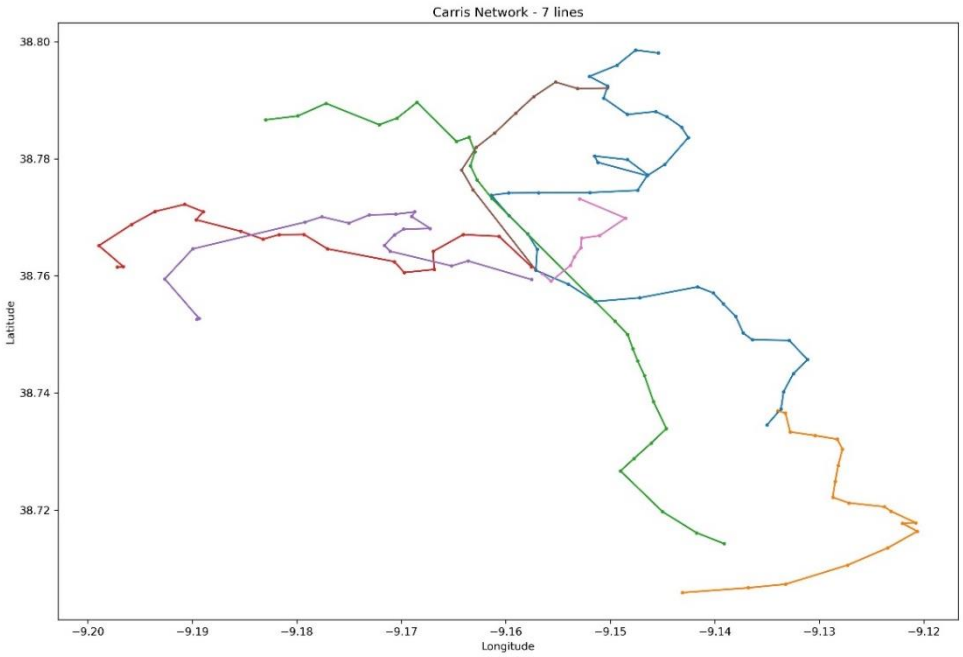


Figura 6: Linhas Carris – Zona Norte

Já na Figura 7 podem ver-se as 11 linhas geradas pelo Modelo 1. Note-se o contraste com a Figura 6: os trajectos gerados pelo Modelo 1 são muito segmentados e algumas das rotas são muito curtas, o que está directamente relacionado com facto de o modelo ter gerado mais linhas do que as que existem na rede Carris. Além disso, é fácil observar zonas onde o algoritmo poderia ter escolhido caminhos mais curtos.

[81] Um conjunto de dados para 20 cidades na Suíça.

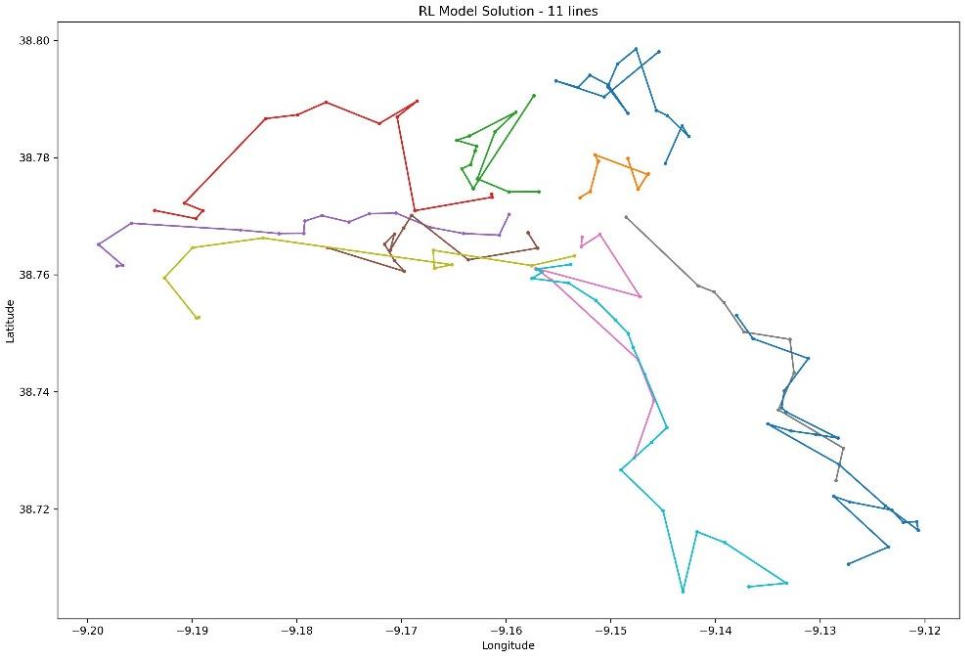


Figura 7: Linhas geradas pelo Modelo 1

Os resultados foram significativamente melhores com a experiência 2: neste caso, a solução encontrada tinha apenas 8 linhas, tendo o Modelo 2 construído uma rede cujo desenho (Figura 8) se aproxima mais do da rede Carris, com trajectos mais longos e contínuos, e em que o o comprimento do trajecto mais curto continua pequeno.

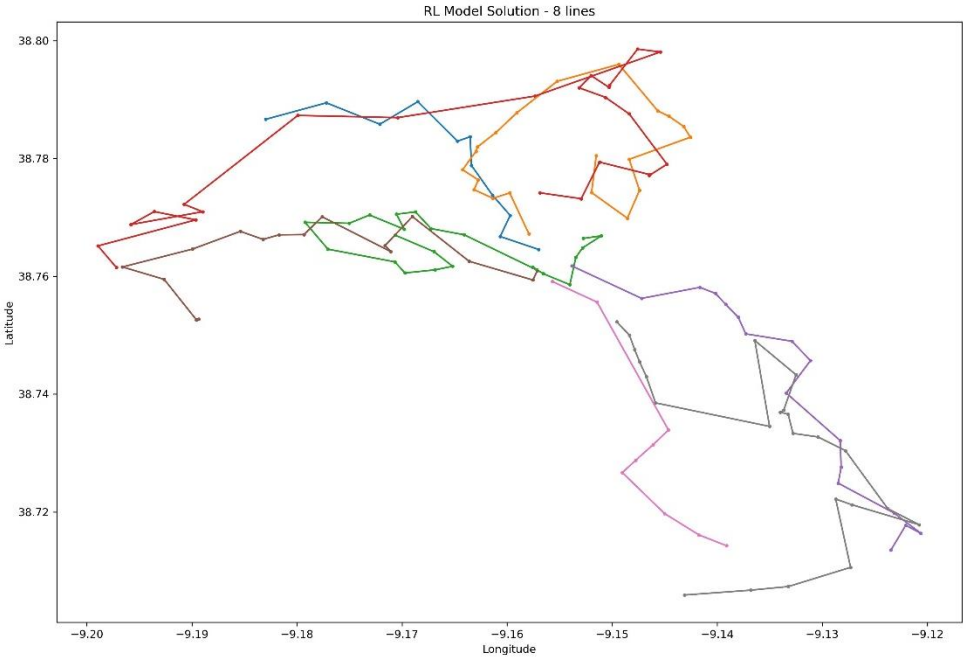


Figura 8: Linhas geradas pelo Modelo 2

Atente-se a Tabela 2, onde se resumem os principias resultados, bem como os valores de algumas métricas também para a rede Carris.

Métrica	Rede Carris	Modelo AR (4 épocas)	Modelo AR (10 épocas)	Solução Savings
Número de linhas	7	11	8	7
Distância total da rede (linha recta - Km)	47.41	66.86	58.51	237.82
Distância do maior troço (linha recta - Km)	2.54	1.84	2.49	8.56
Distância total da rede (por estrada - Km)	60.88	147.48	125.41	406.67
Duração do percurso da rede (por estrada - min)	207.48	452.45	377.77	952.62

Tabela 2: Resultados alcançados – modelos AR e algoritmo *Savings*

Note-se, então, que os modelos treinados conseguiram minimizar a distância máxima entre dois pontos da rede – no caso da rede da Carris a distância máxima é de 2.54 Km, enquanto no caso da rede encontrada pelos modelos das experiências foi de 1.84 Km (Modelo 1) e 2.49 km (Modelo 2). Este pode ser considerado um bom resultado visto que a distância máxima entre dois pontos da rede foi o objectivo que o algoritmo tentou minimizar (e não, por exemplo, a distância total da rede).

No entanto, as distâncias totais encontradas pelos modelos 1 e 2 são francamente piores que as da rede actual da Carris: a rede da Carris tem 47 Km e a rede da solução encontrada pelo modelo 2 tem 58 Km, o que representa uma rede 19% mais longa que a da Carris. As soluções encontradas teriam beneficiado de métodos posteriores de melhoramento, que atuam localmente sobre a rede, tal como o método heurístico Lin–Kernighan[82] visto que, pelo menos no caso da Modelo 1, pode facilmente ver-se através da observação da figura correspondente que há locais onde o algoritmo escolhe percursos piores do que os possíveis.

Por fim, note-se que a distância por estrada encontrada pelo modelo é muitíssimo divergente da existente atualmente (61 Km na rede Carris versus 125 Km com o Modelo 2), o que se deve ao facto de terem sido usadas as coordenadas geográficas e a distância entre as mesmas em linha recta para treinar o modelo, e não a distância por estrada ou a duração da viagem entre as localizações.

Esta foi, aliás, uma das limitações destes algoritmos: o ambiente permitia apenas definir pontos através das suas coordenadas, não sendo possível incluir a distância (por estrada) entre os mesmos. A este

[82] Este método é baseado numa generalização do método de "transformação por troca". A ideia central deste método consiste na troca de um número não fixado de arestas de uma solução previamente encontrada, arestas essas que são reorganizadas para criar uma nova solução; é um algoritmo de busca local com que se pretende melhorar progressivamente uma solução inicial, evitando mínimos locais. (Lin & Kernighan, 1973)

respeito, refira-se que pela forma como a maioria dos modelos trata os dados – nomeadamente, escalando as coordenadas de localização para o quadrado unitário, de forma a homogeneizar e facilitar o processo de treino – conclui-se que a questão da escala destes problemas é ignorada no processo de treino: como se pensar este problema entre cidades fosse idêntico a pensá-lo dentro de uma cidade, ou pensar este problema para um carro fosse idêntico a pensá-lo a pé ou de autocarro[83].

No entanto, a diferença entre os percursos entre cidades e os percursos dentro de uma cidade não é apenas de escala, isto é, uma não é apenas a ampliação da outra: são qualitativamente diferentes. O percurso entre cidades assemelha-se a uma "linha recta", em que a distância por estrada entre dois pontos é sensivelmente igual à distância em linha recta (e, de facto, quanto mais distantes estiverem as cidades, mais o percurso se assemelha a uma linha recta à escala a que é medido) – nestes casos, é indiferente ter como *input* a localização ou a matriz distância; já o percurso entre pontos de uma cidade é muitas vezes sinuoso, sem linhas rectas, fractal, em que a distância entre dois pontos raramente corresponde à distância em linha recta e, mais importante, em que a "distância" entre dois pontos não dependente apenas da distância euclidiana mas também do tipo de sítio de cidade em que essas coordenadas se encontram. Mais, não é linear a relação entre a distância por estrada e o tempo de viagem entre dois pontos.

No caso de Lisboa, os percursos podem ser extremamente sinuosos. Embora fora do âmbito desta tese, poder-se-ia, em princípio, pensar cada uma das linhas como um caminho aleatório, formando um padrão fractal cuja dimensão – representativa das irregularidades da rede – pode ser analisada e calculada, contribuindo para o estudo da rede como um todo e, eventualmente, para a sua modelação, servindo de hiperparâmetro para o treino do modelo, mais tarde.

Para além destas considerações, note-se que a aplicação destes algoritmos para modelar o tráfego dentro de cidades (e não, entre cidades) traz desafios extra ao problema, já que pensar o VRP dentro de uma cidade tem duas particularidades que tornam o problema substancialmente mais complexo: a definição dos horários, porque é preciso passar várias vezes em cada paragem e existem muitas interconexões (ao aplicar estes algoritmos entre cidades a questão das interconexões não é tão relevante), as motivações variadas para a deslocação e a consequente maior complexidade de movimentos, a existência de muitos mais pontos que na generalidade dos problemas, pontos esses que mesmo estando muito próximos não têm, necessariamente, que estar ligados (as pessoas podem mudar de transporte a pé) e a questão do trânsito e de ruas com apenas uma direção (ou seja, o caminho de A para B pode não ser nada parecido com o caminho de B para A), o que faz com que as matrizes distância e duração não

[83] Além desta questão, há ainda dúvidas quanto ao processo de normalização das localizações já que, aparentemente, algumas das formas de normalização alteram as distâncias relativas entre pontos.

sejam nada simétricas, ao contrário do que se assume em vários estudos (como por exemplo, (Darwish et al., 2020)).

Conclui-se então que as ferramentas atuais e, em particular, os algoritmos disponíveis na biblioteca RL4CO ainda só permitem definir ambientes muito simplificados quando comparados com a realidade.

Apesar de todas as simplificações feitas, os resultados foram apenas ligeiramente melhores quando comparados com a rede Carris, no que concerne a distância do maior troço, não tendo nenhum dos modelos encontrado rotas mais curtas (em linha recta) que a da Carris.

Em todo o caso, no entanto, para obter os resultados alcançados não foi necessário conhecimento prévio sobre a cidade ou o trânsito ou mesmo sobre problemas de roteamento, o que pode ser considerada uma vantagem da abordagem experimentada nesta tese. Para além disso, foi possível perceber que algoritmos treinados para um determinado número de pontos podem ser generalizados para situações com menos pontos, podendo o modelo ser facilmente generalizado a outras situações.

Conclusão

Resumo das Descobertas

Com esta dissertação explorou-se a aplicação de mecanismos de aprendizagem reforçada para resolver um problema de roteamento de veículos focado na otimização da rede de transportes públicos de Lisboa, em particular numa parte da rede Carris.

A partir do uso da arquitectura denominada *Multi-task Vehicle Routing Solver with Mixture-of-Experts* – MVMoE foi possível obter resultados satisfatórios no que concerne a descoberta, de forma eficiente, de rotas mais curtas que as da rede Carris. O modelo treinado conseguiu encontrar rotas em que foi minimizada a distância do troço mais comprido, sem incorporação de qualquer conhecimento anterior sobre resolução de problemas de otimização matemática ou conhecimento especializado sobre o ambiente urbano complexo da cidade de Lisboa. O modelo conseguiu fazê-lo após tempo aceitável, em particular quando se considera que este era um problema de roteamento relativamente grande. Assim, esta abordagem poderá contribuir para a construção de uma alternativa aos métodos tradicionais como os métodos matemáticos exatos ou os métodos meta-heurísticos.

A pesquisa efetuada indica que a Aprendizagem Reforçada pode contribuir para melhorar a eficiência das redes de transporte público, ao permitir a otimização dinâmica das mesmas: embora o tempo de treino possa ser longo, após esta fase o modelo consegue devolver soluções para uma multiplicidade de casos, podendo resolver situações dinâmicas e adaptar a solução a desafios novos, tais como interrupções momentâneas da via pública ou constrangimentos de tráfego.

O aprofundamento da pesquisa sobre modelos de aprendizagem reforçada pode contribuir para a redução das rotas, ao mesmo tempo se que mantém o nível de serviço e, assim, ter impacto para operadores (com redução de custos) e utilizadores (com redução do tempo de viagem e fiabilidade do serviço).

Desafios e Limitações

Apesar do relativo sucesso da aplicação de métodos de aprendizagem reforçada ao VRP, foram encontrados vários desafios. A complexidade da rede da Carris e a grande quantidade de paragens impuseram dificuldades computacionais, o que exigiu a simplificação da rede para um subconjunto de paragens. Além disso, os modelos de AR, embora eficazes, dependem de extensos recursos computacionais, especialmente para problemas de grande escala. Adicionalmente, no caso particular desta dissertação, a natureza estática do conjunto de dados utilizado poderá limitar a capacidade do

modelo em lidar com mudanças dinâmicas em tempo real, como as condições de tráfego e a evolução dos padrões de deslocação.

No que concerne a generalidade dos algoritmos de AR usados para resolver esta classe de problemas, a pesquisa efectuada permitiu concluir que ainda é necessário muito desenvolvimento para que estes algoritmos devolvam resultados mais satisfatórios e possam ser aplicados a casos reais, em grande escala. Veja-se, por exemplo, o que refere Kevin Tierney da Universidade de Bielefeld[84], que considera que as abordagens de aprendizagem automática aos problemas de otimização matemática ainda se encontram na sua infância, não devolvendo ainda a melhor solução na maioria das vezes, apenas conseguindo resolver problemas muito simples e não sendo facilmente escaláveis ou generalizáveis (isto é, se os constrangimentos do problema mudam um pouco, tem de ser treinado um novo modelo). No mesmo sentido, Berto et al. (2024) referem que "*despite the recent progress made in learning to solve individual VRP variants, there is a lack of a unified approach that can effectively tackle a wide range of tasks, which is crucial for real-world impact*" (p. 1).

Estas considerações vão de encontro aos resultados obtidos nesta dissertação, onde, apesar das simplificações feitas e de ter sido possível minimizar a distância do maior troço, a distância total das rotas encontradas está longe do desejável e não é inferior à da rede Carris.

Recomendações Práticas para a Carris

Para melhorar a eficiência operacional da Carris, recomenda-se que a rede considere implementar otimização baseada em AR em programas-piloto. O modelo proposto poderia ser utilizado para reavaliar as rotas de autocarros existentes e sugerir melhorias para linhas com baixo desempenho, com base nos padrões de procura identificados. Além disso, os algoritmos de AR poderiam ser empregues para ajustar dinamicamente as frequências dos autocarros, garantindo um serviço mais adaptável que se alinhe com a procura de passageiros, em tempo real. Com grande probabilidade, estas ações levariam à redução dos custos operacionais e à melhoria da satisfação dos passageiros, minimizando os tempos de espera e otimizando a alocação dos autocarros.

Para que os autocarros possam ser bem alocados e as redes mais bem definidas, considera-se necessário, também, fazer um estudo sobre o padrão de deslocação dos passageiros. Este estudo é necessário visto que, atualmente, a empresa Carris não sabe exatamente para onde transporta as pessoas.

[84] A palestra onde este investigador discute vários limites à abordagem de AR a problemas de otimização, "Search Heuristics for Solving Routing Problems with Deep Reinforcement Learning" pode ser vista em https://www.youtube.com/watch?v=nqAubq2K_Ug. (acedido em dezembro 2023)

Da forma como a rede é feita, a Carris tem informação sobre onde é que cada passageiro entra, mas não onde é que cada passageiro sai, já que os passageiros "picam" o seu bilhete à entrada, mas não à saída. Não é certo, também, que a Carris tenha informação sobre o número de pessoas que não entra nos autocarros por estes estarem cheios. Os estudos de mobilidade feitos por parte do INE são importantes, mas a Carris beneficiaria de informação mais fina, feita ao nível de cada bairro.

Foi notícia recente que os autocarros da Carris nunca foram tão lentos como em 2024 – uma média de 14 Km/hora[85]. Evidentemente, muito se deve a fatores externos (caso do aumento do número de automóveis na cidade ou de grandes obras em Lisboa); no entanto, a melhoria das rotas, bem como a sua alteração com base em situações previsíveis é um fator a ser tido em conta de forma a melhorar o serviço que a Carris presta. Além disto, a partir de estudos realizados pelo INE[86], conclui-se que uma das razões apresentadas pela população para não usar tanto os autocarros tem que ver precisamente com a imprevisibilidade desse transporte e o tempo de espera, que são fatores que poderiam ser melhorados com a ajuda de métodos de AR. O aprofundamento das abordagens que foram aqui exploradas seria, então, uma mais-valia nestas situações.

Pesquisas Futuras

Ao longo desta dissertação foi possível refletir sobre vários aspetos que podem ser aprofundados. De um ponto de vista muito geral, continuar a desenvolver os modelos implicará necessariamente a expansão do modelo de AR para incorporar variáveis mais dinâmicas, como dados de tráfego em tempo real ou densidades de passageiros flutuantes, e a integração de camadas adicionais de complexidade, como opções de transporte multimodal (metro, comboio e sistemas de partilha de bicicletas). Estes desenvolvimentos poderiam fornecer uma solução mais holística para os desafios de mobilidade urbana.

Outra área de desenvolvimento, neste caso de um ponto de vista mais técnico, diz respeito à escalabilidade dos algoritmos propostos para um maior número de localizações[87], o que poderia fornecer uma validação adicional da eficácia destes modelos.

---

[85] Ver a notícia do Público "Em mais de 15 anos, autocarros de Lisboa e Porto nunca estiveram tão lentos", em https://www.publico.pt/2024/06/20/local/noticia/15-anos-autocarros-lisboa-porto-tao-lentos-2094651. (acedido em junho 2024)

[86] (INE - Instituto Nacional de Estatística, 2017)

[87] Ao longo da pesquisa bibliográfica econtraram-se algumas formas usadas actualmente para resolver instâncias muito maiores do VRP, tais como métodos que utilizam estratégias "dividir para reinar", em que se divide um problema grande em vários mais pequenos (eventualmente, com recurso a AR) para depois os resolver com heurísticas que funcionam muito bem para problemas menores.

No que concerne os algoritmos usados e, em particular, a biblioteca RL4CO, seria interessante, no futuro, alterar a função que mascara as localizações, de forma que possa haver localizações visitadas várias vezes (que corresponderiam a pontos onde passam várias linhas e onde é possível trocar de veículo).

A incorporação de informação sobre tráfego não foi efetuada em nenhuma das experiências, mas a forma de incluir essa informação nos modelos foi pensada. É necessário perceber como pode ser feita a inclusão desta informação para os casos em que os algoritmos não permitem a inclusão de uma matriz distância – ou de matriz equivalente, tal como uma "matriz tempo" com informação sobre o tempo de viagem entre cada duas localizações, que já incorpora naturalmente a informação sobre o tráfego. Mesmo que não seja possível incluir esta informação diretamente nos modelos (tal como acontece com a biblioteca RL4CO), esta pode ser incluída na fase de aplicação do modelo aos dados reais, através da atribuição de um peso a cada aresta (isto é, aos caminhos entre cada dois pontos): aquando a normalização dos pontos para o quadrado unitário todas as localizações são reescalonadas com direção paralela ao vetor *localização-depot* e magnitude proporcional à razão entre a distância (localização - depot) e a duração (localização - depot). Assim, conseguir-se-ia incluir a noção de trânsito no resultado final.

Já foi possível refletir sobre as limitações actuais dos modelos aqui experimentados. Assim, um dos desenvolvimentos necessários tem que ver com a necessidade de complexificar os problemas que se estão a tentar resolver – isto é, deixar de resolver "apenas" VRPs com alguns constrangimentos e passar a tentar abordar todo o problema do *design* de tais redes: a construção das "melhores" rotas possíveis é um problema muito mais abrangente que a mera resolução de um VRP: na prática, antes de encontrar estas rotas é necessário também definir os pontos de paragem (pelo menos no caso de rotas dentro de cidades) e, depois de encontrá-las, é necessário definir, por exemplo, diferentes calendarizações ou horários. O conjunto destes problemas denomina-se, tal como já referido, *Transit Network Design and Frequency Setting Problem* (TNDFSP). Hoje-em-dia cada um destes problemas é ainda tratado separadamente Darwish et al., (2020) recorrendo-se a *data,* técnicas e algoritmos diferentes. Por exemplo, a escolha do local das paragens pode depender de paradigmas teóricos relacionados com o urbanismo – do que se quer ao pensar uma cidade – e ser feita com recurso a métodos supervisionados, a definição dos "melhores" trajetos pode significar encontrar trajetos mais rápidos ou trajetos mais conectados e estes trajetos podem ser encontrados recorrendo a métodos de aprendizagem reforçada e, por fim, a definição dos horários e das frequências pode (ou, deve) depender das necessidades de viagens dos utentes bem como das interconexões com outros meios de transporte (o que é especialmente importante quando se pensa em desenhar uma rede pública de transportes) e ser encontrada com base na experiência de peritos. Ora, uma direcção de pesquisa futura corresponde ao desenvolvimento de algoritmos que possam não só

encontrar rotas otimizadas relativamente a vários parâmetros diferentes, como também, a montante, escolher o local das paragens e, a jusante, definir os horários e frequências dos meios de transporte.

As pesquisas e ações futuras serão seguidas, com coerência, mas sem certeza de sucesso, pois a escolha de “o que fazer a seguir” é ela própria um problema de otimização, difícil de resolver.

Disponibilização de Dados

Ao longo desta dissertação foram produzidos vários materias, nos quais se incluem:

- Dissertação escrita em português.
- Tradução da dissertação para inglês.
- Código fonte adaptado ao problema em questão (biblioteca RL4CO).
- Código em python versão 3.10 usado para retirar toda a informação necessária à dissertação.
- Ficheiros com o modelo, através do qual se podem repetir as experiências efectuadas.
- Ficheiros excel com informação variada sobre as paragens de Lisboa (inclusive latitude e longitude).

Referências Bibliográficas

Bello, I., Pham, H., Le, Q. V, Norouzi, M., Bengio, S., & Brain, G. (2017). *Workshop track-ICLR 2017 Neural Combinatorial Optimization With Reinforcement Learning*.

Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, *290*(2), 405–421. https://doi.org/10.1016/j.ejor.2020.07.063

Berto, F., Hua, C., Park, J., Luttmann, L., Ma, Y., Bu, F., Wang, J., Ye, H., Kim, M., Choi, S., Zepeda, N. G., Hottung, A., Zhou, J., Bi, J., Hu, Y., Liu, F., Kim, H., Son, J., Kim, H., … Park, J. (2023). *RL4CO: an Extensive Reinforcement Learning for Combinatorial Optimization Benchmark*.

Berto, F., Hua, C., Zepeda, N. G., Hottung, A., Wouda, N., Lan, L., Tierney, K., & Park, J. (2024). *RouteFinder: Towards Foundation Models for Vehicle Routing Problems*.

Brockmann, D., Hufnagel, L., & Geisel, T. (2006). The scaling laws of human travel. *Nature*, *439*(7075), 462–465. https://doi.org/10.1038/nature04292

Clarke, G., & Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, *12*(4), 568–581. https://doi.org/10.1287/opre.12.4.568

CML. (2021). *Inquérito à Moblidade Lisboa*.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., & Song, L. (2017). *Learning Combinatorial Optimization Algorithms over Graphs*.

Darwish, A., Khalil, M., & Badawi, K. (2020). Optimising Public Bus Transit Networks Using Deep Reinforcement Learning. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC 2020*. https://doi.org/10.1109/ITSC45102.2020.9294710

Deutsch, D. (2013). *O Início do Infinito - Explicações Que Transformam o Mundo* (1st ed.).

Eldrandaly, K. A., Ahmed A. H. N., & AbdAll A. F. (2008). The 43rd Annual Conference on Statistics, Computer Sciences and Operations Research 22-25 Dec 2008. In Institute of Statistical Studies and Research - Cairo University (Ed.), *Routing Problems: A Survey* (pp. 51–70).

Eric Rosenbaum. (2023, February 11). The ChatGPT AI hype cycle is peaking, but even tech skeptics don't expect a bust. *CNBC - Technology Executive Coucil*. https://www.cnbc.com/2023/02/11/chatgpt-ai-hype-cycle-is-peaking-but-even-tech-skeptics-doubt-a-bust.html

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R. Ruiz, F. J., Schrittwieser, J., Swirszcz, G., Silver, D., Hassabis, D., & Kohli, P. (2022). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, *610*(7930), 47–53. https://doi.org/10.1038/s41586-022-05172-4

Gkiotsalitis, K. (2023). Public Transport Optimization. In *Public Transport Optimization*. Springer International Publishing. https://doi.org/10.1007/978-3-031-12444-0

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*.

Google OR-Tools. (2024, March 11). *What is an optimization problem?* Google OR-Tools. https://developers.google.com/optimization/introduction/python#optimization

Ha, S., & Jeong, H. (2022). *Social learning spontaneously emerges by searching optimal heuristics with deep reinforcement learning*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer New York. https://doi.org/10.1007/978-0-387-84858-7

INE. (2023). *O que nos dizem os Censos sobre dinâmicas territoriais*. https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_publicacoes&PUBLICACOESpub_boui=66320870&PUBLICACOESmodo=2

INE - Instituto Nacional de Estatística. (2017). *IMOB - Inquérito à Mobilidade nas Áreas Metropolitanas de Lisboa e do Porto*. https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_destaques&DESTAQUESdest_boui=334619442&DESTAQUESmodo=2&xlang=pt

Jackie Wiles. (2022, September 15). What's New in Artificial Intelligence from the 2022 Gartner Hype Cycle. *Gartner*. https://www.gartner.com/en/articles/what-s-new-in-artificial-intelligence-from-the-2022-gartner-hype-cycle

Kepaptsoglou, K., & Karlaftis, M. (2009). Transit Route Network Design Problem: Review. *Journal of Transportation Engineering*, *135*(8), 491–505. https://doi.org/10.1061/(ASCE)0733-947X(2009)135:8(491)

Kirk, R., Zhang, A., Grefenstette, E., & Rocktäschel, T. (2021). *A Survey of Generalisation in Deep Reinforcement Learning*. https://doi.org/10.1613/jair.1.14174

Kool, W., van Hoof, H., & Welling, M. (2018). *Attention, Learn to Solve Routing Problems!*

Kuhn, T. S., & Hacking, I. (2012). *The Structure of Scientific Revolutions*. University of Chicago Press. https://doi.org/10.7208/chicago/9780226458144.001.0001

Laporte, G., & Semet, F. (2002). *The Vehicle Routing Problem* (P. Toth & D. Vigo, Eds.). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718515

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). *Continuous control with deep reinforcement learning*.

Lin, S., & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, *21*(2), 498–516. https://doi.org/10.1287/opre.21.2.498

Liu, F., Lin, X., Wang, Z., Zhang, Q., Tong, X., & Yuan, M. (2024). *Multi-Task Learning for Routing Problem with Cross-Problem Zero-Shot Generalization*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Nakabi, T. A., & Toivanen, P. (2021). Deep reinforcement learning for energy management in a microgrid with flexible demand. *Sustainable Energy, Grids and Networks*, *25*, 100413. https://doi.org/10.1016/j.segan.2020.100413

Nazari, M., Oroojlooy, A., Snyder, L. V., & Takáč, M. (2018). *Reinforcement Learning for Solving the Vehicle Routing Problem*. http://arxiv.org/abs/1802.04240

Nikolinakos, N. Th. (2023). *A European Approach to Excellence and Trust: The 2020 White Paper on Artificial Intelligence* (pp. 211–280). https://doi.org/10.1007/978-3-031-27953-9_5

Peng, B., Wang, J., & Zhang, Z. (2020). *A Deep Reinforcement Learning Algorithm Using Dynamic Attention Model for Vehicle Routing Problems*.

Pereira, R. H. M., Andrade, P. R., & Vieira, J. P. B. (2022). Exploring the time geography of public transport networks with the gtfs2gps package. *Journal of Geographical Systems*. https://doi.org/10.1007/s10109-022-00400-x

Pestana, D., & Velosa, S. (2008). *Introdução à Probabilidade e à Estatística* (Fundação Calouste Gulbenkian, Ed.; 3rd ed.).

Plaat, A. (2022). *Deep Reinforcement Learning*. Springer Nature Singapore. https://doi.org/10.1007/978-981-19-0638-1

Popova, M., Isayev, O., & Tropsha, A. (2018). Deep reinforcement learning for de novo drug design. *Science Advances*, *4*(7). https://doi.org/10.1126/sciadv.aap7885

Provost, F., & Fawcett, T. (2013). *Data Science for Business* (O'Reilly, Ed.; 1st ed.).

Selukar, M., Jain, P., & Kumar, T. (2022). Inventory control of multiple perishable goods using deep reinforcement learning for sustainable environment. *Sustainable Energy Technologies and Assessments*, *52*, 102038. https://doi.org/10.1016/j.seta.2022.102038

Silver, D. (2016). *RL Course by David Silver - Lecture 1: Introduction to Reinforcement Learning [Video]. YouTube.* https://www.youtube.com/watch?v=2pWv7GOvuf0&list=PLzuuYNsE1EZAXYR4FJ75jcJseBmo4KQ9-

Silver, D., Singh, S., Precup, D., & Sutton, R. S. (2021). Reward is enough. *Artificial Intelligence*, *299*, 103535. https://doi.org/10.1016/j.artint.2021.103535

Soares, J. L. (2005). *Optimização Matemática*. http://www.mat.uc.pt/˜jsoares/.1

Sutton, R. S., & Barto, A. G. (2015). *Reinforcement Learning: An Introduction* (2º ed (in progress)). https://doi.org/10.1109/TNN.1998.712192

Szepesvári, C. (2010). *Algorithms for Reinforcement Learning*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01551-9

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2017). *Graph Attention Networks*.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). *Pointer Networks*.

Wagstaff, K. L. (2010). *Machine Learning that Matters*.

Whittlestone, J., Arulkumaran, K., & Crosby, M. (2021). The Societal Implications of Deep Reinforcement Learning. *Journal of Artificial Intelligence Research*, *70*, 1003–1030. https://doi.org/10.1613/jair.1.12360

Wolfram, S. (2023). *Wolfram - Computation meets Knowledge*. WolframMathWorld. https://mathworld.wolfram.com/AdjacencyMatrix.html

Wu, G., Li, Y., Bao, J., Zheng, Y., Ye, J., & Luo, J. (2018). Human-Centric Urban Transit Evaluation and Planning. *2018 IEEE International Conference on Data Mining (ICDM)*, 547–556. https://doi.org/10.1109/ICDM.2018.00070

Yoo, S., Lee, J. B., & Han, H. (2023). A Reinforcement Learning approach for bus network design and frequency setting optimisation. *Public Transport*, *15*(2), 503–534. https://doi.org/10.1007/s12469-022-00319-y

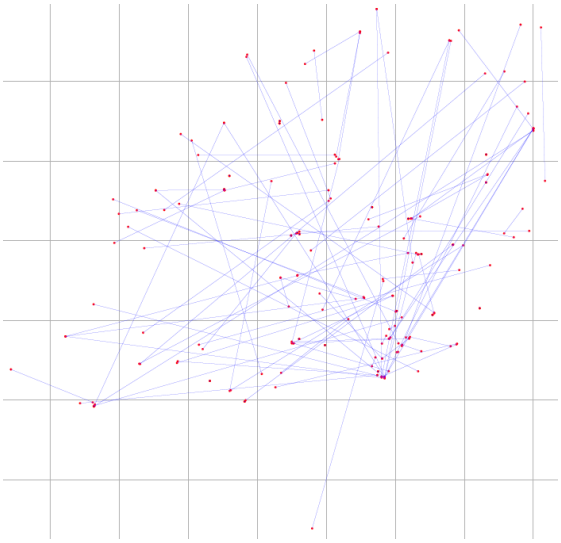
Zhang, J. E., Wu, D., & Boulet, B. (2022). Time Series Anomaly Detection via Reinforcement Learning-Based Model Selection. *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 193–199. https://doi.org/10.1109/CCECE49351.2022.9918216

Zhou, J., Cao, Z., Wu, Y., Song, W., Ma, Y., Zhang, J., & Xu, C. (2024). *MVMoE: Multi-Task Vehicle Routing Solver with Mixture-of-Experts*.

Zhou, Q., Yang, Y., & Fu, S. (2022). Deep reinforcement learning approach for solving joint pricing and inventory problem with reference price effects. *Expert Systems with Applications*, *195*, 116564. https://doi.org/10.1016/j.eswa.2022.116564

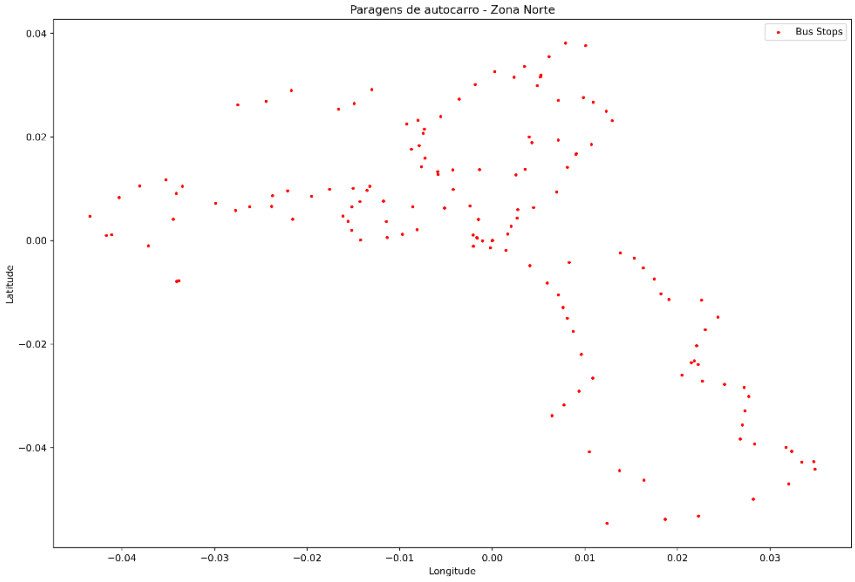
Anexo I

Imagem grosseira da rede Carris (primeira e última paragem)



Anexo II

Conjunto de pontos da zona Norte (144 paragens)



Anexo III

Arquitectura computacional da política do agente

```
MVMoE_POMO(
  (env): MTVRPEnv()
  (policy): AttentionModelPolicy(
    (encoder): AttentionModelEncoder(
      (init_embedding): MTVRPInitEmbedding(
        (init_embed): Linear(in_features=7, out_features=128, bias=True)
        (init_embed_depot): Linear(in_features=2, out_features=128, bias=True)
      )
      (net): GraphAttentionNetwork(
        (layers): Sequential(
          (0): MultiHeadAttentionLayer(
            (0): SkipConnection(
              (module): MultiHeadAttention(
                (Wqkv): Linear(in_features=128, out_features=384, bias=True)
                (out_proj): Linear(in_features=128, out_features=128, bias=True)
              )
            )
            (1): Normalization(
              (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
            )
            (2): SkipConnection(
              (module): MoE(
                (experts): ModuleList(
                  (0-3): 4 x MLP(
                    (hidden_act): ReLU()
                    (out_act): Identity()
                    (lins): ModuleList(
                      (0): Linear(in_features=128, out_features=512, bias=True)
                      (1): Linear(in_features=512, out_features=128, bias=True)
                    )
                    (input_norm): Identity()
                    (output_norm): Identity()
                  )
                )
                (softplus): Softplus(beta=1.0, threshold=20.0)
                (softmax): Softmax(dim=-1)
              )
            )
            (3): Normalization(
              (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
```

```
      )
    )
    (1): MultiHeadAttentionLayer(
      (0): SkipConnection(
        (module): MultiHeadAttention(
          (Wqkv): Linear(in_features=128, out_features=384, bias=True)
          (out_proj): Linear(in_features=128, out_features=128, bias=True)
        )
      )
      (1): Normalization(
        (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
      )
      (2): SkipConnection(
        (module): MoE(
          (experts): ModuleList(
            (0-3): 4 x MLP(
              (hidden_act): ReLU()
              (out_act): Identity()
              (lins): ModuleList(
                (0): Linear(in_features=128, out_features=512, bias=True)
                (1): Linear(in_features=512, out_features=128, bias=True)
              )
              (input_norm): Identity()
              (output_norm): Identity()
            )
          )
          (softplus): Softplus(beta=1.0, threshold=20.0)
          (softmax): Softmax(dim=-1)
        )
      )
      (3): Normalization(
        (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
      )
    )
    (2): MultiHeadAttentionLayer(
      (0): SkipConnection(
        (module): MultiHeadAttention(
          (Wqkv): Linear(in_features=128, out_features=384, bias=True)
          (out_proj): Linear(in_features=128, out_features=128, bias=True)
        )
      )
      (1): Normalization(
        (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
      )
      (2): SkipConnection(
```

```
          (module): MoE(
            (experts): ModuleList(
              (0-3): 4 x MLP(
                (hidden_act): ReLU()
                (out_act): Identity()
                (lins): ModuleList(
                  (0): Linear(in_features=128, out_features=512, bias=True)
                  (1): Linear(in_features=512, out_features=128, bias=True)
                )
                (input_norm): Identity()
                (output_norm): Identity()
              )
            )
            (softplus): Softplus(beta=1.0, threshold=20.0)
            (softmax): Softmax(dim=-1)
          )
        )
        (3): Normalization(
          (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
        )
      )
      (3): MultiHeadAttentionLayer(
        (0): SkipConnection(
          (module): MultiHeadAttention(
            (Wqkv): Linear(in_features=128, out_features=384, bias=True)
            (out_proj): Linear(in_features=128, out_features=128, bias=True)
          )
        )
        (1): Normalization(
          (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
        )
        (2): SkipConnection(
          (module): MoE(
            (experts): ModuleList(
              (0-3): 4 x MLP(
                (hidden_act): ReLU()
                (out_act): Identity()
                (lins): ModuleList(
                  (0): Linear(in_features=128, out_features=512, bias=True)
                  (1): Linear(in_features=512, out_features=128, bias=True)
                )
                (input_norm): Identity()
                (output_norm): Identity()
              )
            )
            (softplus): Softplus(beta=1.0, threshold=20.0)
```

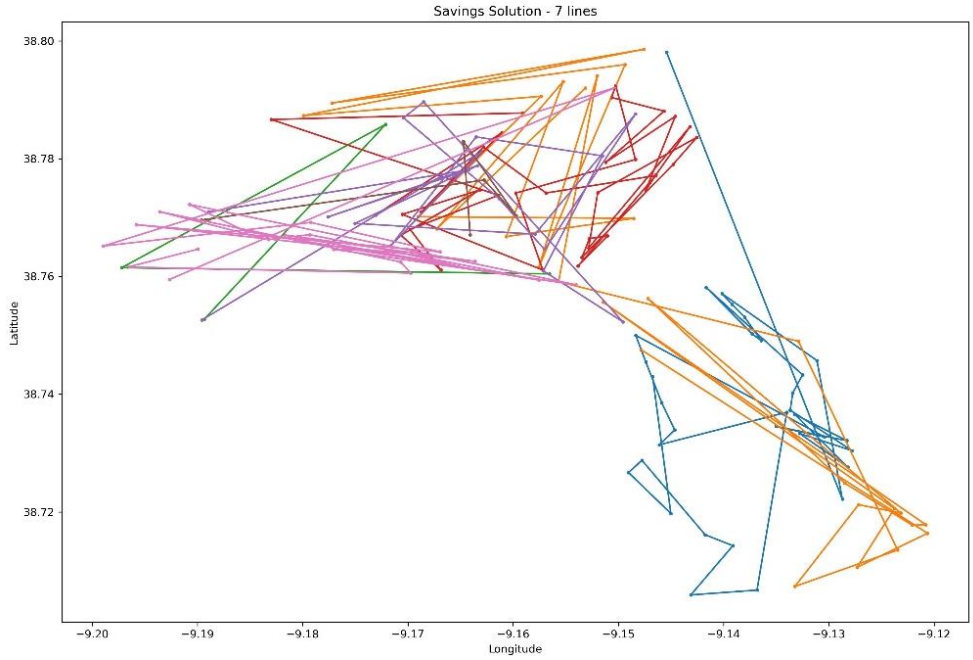
```
          (softmax): Softmax(dim=-1)
        )
      )
      (3): Normalization(
        (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
      )
    )
    (4): MultiHeadAttentionLayer(
      (0): SkipConnection(
        (module): MultiHeadAttention(
          (Wqkv): Linear(in_features=128, out_features=384, bias=True)
          (out_proj): Linear(in_features=128, out_features=128, bias=True)
        )
      )
      (1): Normalization(
        (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
      )
      (2): SkipConnection(
        (module): MoE(
          (experts): ModuleList(
            (0-3): 4 x MLP(
              (hidden_act): ReLU()
              (out_act): Identity()
              (lins): ModuleList(
                (0): Linear(in_features=128, out_features=512, bias=True)
                (1): Linear(in_features=512, out_features=128, bias=True)
              )
              (input_norm): Identity()
              (output_norm): Identity()
            )
          )
          (softplus): Softplus(beta=1.0, threshold=20.0)
          (softmax): Softmax(dim=-1)
        )
      )
      (3): Normalization(
        (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
      )
    )
    (5): MultiHeadAttentionLayer(
      (0): SkipConnection(
        (module): MultiHeadAttention(
          (Wqkv): Linear(in_features=128, out_features=384, bias=True)
          (out_proj): Linear(in_features=128, out_features=128, bias=True)
        )
```

```
        )
        (1): Normalization(
          (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
        )
        (2): SkipConnection(
          (module): MoE(
            (experts): ModuleList(
              (0-3): 4 x MLP(
                (hidden_act): ReLU()
                (out_act): Identity()
                (lins): ModuleList(
                  (0): Linear(in_features=128, out_features=512, bias=True)
                  (1): Linear(in_features=512, out_features=128, bias=True)
                )
                (input_norm): Identity()
                (output_norm): Identity()
              )
            )
            (softplus): Softplus(beta=1.0, threshold=20.0)
            (softmax): Softmax(dim=-1)
          )
        )
        (3): Normalization(
          (normalizer): InstanceNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
        )
      )
    )
  )
  (decoder): AttentionModelDecoder(
    (context_embedding): MTVRPContext(
      (project_context): Linear(in_features=133, out_features=128, bias=False)
    )
    (dynamic_embedding): StaticEmbedding()
    (pointer): PointerAttnMoE(
      (project_out): None
      (project_out_moe): MoE(
        (experts): ModuleList(
          (0-3): 4 x Linear(in_features=128, out_features=128, bias=False)
        )
        (softplus): Softplus(beta=1.0, threshold=20.0)
        (softmax): Softmax(dim=-1)
      )
    )
    (project_node_embeddings): Linear(in_features=128, out_features=384, bias=False)
    (project_fixed_context): Linear(in_features=128, out_features=128, bias=False)
```

```
    )
  )
  (baseline): SharedBaseline()
)
```

Anexo IV

Linhas Geradas pelo algoritmo *Savings* – Zona Norte (Solução *Savings*)



Anexo V

Alterações ao código fonte

Listam-se aqui algumas das alterações efectuadas ao código fonte, nomeadamente o referente à biblioteca RL4CO. Estas alterações poderão resultar na melhoria do código fonte disponível na plataforma *GitHub*.

I. Após instalar devidamente a biblioteca PyVRP (um *solver* com o qual é possível comparar os resultados obtidos pelos modelos de AR), correu-se a seguinte linha:

```
pyvrp_actions, pyvrp_costs = env.solve(instances = td, max_runtime = 5, num_procs = 10, solver="pyvrp")
```

que resultou no erro:

```
ImportError: PyVRP is not installed. Please install it using `pip install -e .[solvers]`.
```

Este erro deveu-se a um *import* que estava a ser mal feito no código fonte, já que indicava incorrectamente o local de uma função. Assim, para o ultrapassar, alterou-se o ficheiro

```
./rl4co/envs/routing/mtvrp/baselines/solve.py
```

onde se lia:

```
import routefinder.baselines.pyvrp as pyvrp
```

passou a ler-se:

```
import pyvrp as pyvrp
```

II. Retirei todos os `max_runtime` dos seguintes ficheiros:

```
/usr/local/lib/python3.10/dist-packages/rl4co/envs/routing/mtvrp/env.py
/usr/lib/python3.10/multiprocessing/pool.py
/usr/local/lib/python3.10/dist-packages/rl4co/envs/routing/mtvrp/baselines/solve.py
/usr/local/lib/python3.10/dist-packages/rl4co/envs/routing/mtvrp/baselines/pyvrp.py
```

III. No ficheiro

```
/usr/local/lib/python3.10/dist-packages/rl4co/envs/routing/mtvrp/baselines/solve.py
```

onde se lia

```
_solve = solvers[solver]
    func = partial(_solve, max_runtime = max_runtime, **kwargs)
```

passou a ler-se

```
_solve = solvers[solver]
    func = partial(_solve, stop = 10, **kwargs)
```

IV. Ainda relacionado com as linhas do ponto 1:

```
pyvrp_actions, pyvrp_costs = env.solve(instances=td, num_procs=10,
solver="pyvrp")
```

surgiu novo erro:

```
AttributeError: 'TensorDict' object has no attribute 'num_locations'
```

Pelo que foi necessário acrescentar essa informação manualmente ao tensor original do ambiente (que não possuia informação directa sobre o número de paragens)

V. Ao correr a linha (no ficheiro TNDFSP_aplicação_V2):

```
with torch.no_grad():
  out = policy(td.clone(), decode_type='greedy', return_actions=True)
```

resultou no erro:

```
AssertionError: Cannot use subsample if variant_preset is not specified.
```

Pelo que foi necessário altar o código fonte:

De `variant_preset=None`, alterou-se para `variant_preset=”cvrp”` e de `subsample=True` alterou-se para `subsample=True`