UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

LEVI JORDÃO MEMÓRIA PAIVA RIBEIRO

APRENDIZAGEM POR REFORÇO EM SISTEMAS MULTIAGENTE APLICADO AO CONTROLE DE GRUPOS DE ELEVADORES

FORTALEZA – CEARÁ

2019

LEVI JORDÃO MEMÓRIA PAIVA RIBEIRO

APRENDIZAGEM POR REFORÇO EM SISTEMAS MULTIAGENTE APLICADO AO CONTROLE DE GRUPOS DE ELEVADORES

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Inteligência Artificial

Orientador: Prof. Dr. José Everardo Bessa Maia

FORTALEZA – CEARÁ

2019

Dados Internacionais de Catalogação na Publicação
Universidade Estadual do Ceará
Sistema de Bibliotecas

Ribeiro, Levi Jordao Memoria Paiva.
Aprendizagem por reforço em sistemas multiagente aplicado ao controle de grupo de elevadores [recurso eletrônico] / Levi Jordao Memoria Paiva Ribeiro. – 2020.
66 f. : il.

Trabalho de Conclusão de Curso (Mestrado acadêmico) – Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Curso de Mestrado Acadêmico em Ciência da Computação, Fortaleza, 2020.

Orientação: Prof. Dr. JOSE EVERARDO BESSA MAIA.

1. Inteligência Artificial. 2. Aprendizagem por reforço. 3. Grupos de elevadores. 4. Sistemas multiagente. 5. Funções heurísticas. I. Título.

LEVI JORDÃO MEMÓRIA PAIVA RIBEIRO

APRENDIZAGEM POR REFORÇO EM SISTEMAS MULTIAGENTE APLICADO AO CONTROLE DE GRUPOS DE ELEVADORES

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Inteligência Artificial

Aprovada em: 21/02/2020

BANCA EXAMINADORA

Prof. Dr. José Everardo Bessa Maia (Orientador)
Universidade Estadual do Ceará - UECE

Prof. Dr. João Paulo Pordeus Gomes
Universidade Federal do Ceará - UFC

Prof. Dr. Gustavo Augusto de Lima Campos
Universidade Estadual do Ceará - UECE

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse, ao longo de minha vida, e não somente nestes anos como universitário, mas que em todos os momentos é o maior mestre que alguém pode conhecer. Obrigado Deus por tudo, desde e sempre por ter me amparado em todos os momentos da minha vida, me ajudando, orientando e incentivando a seguir.

Obrigado meus irmãos e sobrinhos, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente!

A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

Agradeço ao meu orientador Prof. Dr. José Everardo Bessa Maia, para quem não há agradecimentos que cheguem. Sou grato pelas aulas, orientação, incentivo, elaboração, pela paciência, pela liderança, atenção e dedicação oferecidas antes e durante a construção deste trabalho. Muito obrigado, pelo amparo em todos os momentos.

Agradeço a minha namorada Tessele por seu cuidado e dedicação que me deram a força para seguir em frente. Por ter aceitado se privar de minha companhia pelos estudos, concedendo a mim a oportunidade de me realizar ainda mais. Obrigado pelo constante amor e incentivo.

À minha mãe Cíntia e ao meu pai Fábio deixo um agradecimento especial, por todas as lições de amor, companheirismo, amizade, caridade, dedicação, abnegação, compreensão e perdão que foi me dado ao longo dos anos.

Agradeço também aos meus amigos dos seguintes grupos que me apoiaram e incentivaram dia-a-dia: meus amigos da graduação (IFCE); meus amigos de infância; e principalmente meus amigos do mestrado.

A todos obrigado por permitirem que esta dissertação seja uma realidade.

"É melhor lançar-se à luta em busca do triunfo mesmo expondo-se ao insucesso, que formar fila com os pobres de espírito, que nem gozam muito nem sofrem muito; E vivem nessa penumbra cinzenta sem conhecer nem vitoria nem derrota."

(Franklin Roosevelt)

RESUMO

Neste trabalho, uma modelagem e algoritmo baseado em aprendizado por reforço multiagente são desenvolvidos para o problema do despacho do grupo de elevadores. A principal vantagem é que, juntamente com a aproximação da função, esta solução multiagente leva a uma síntese do espaço de estados, permitindo que estados complexos sejam endereçados com uma função de avaliação. Cada elevador é considerado um agente que tem que decidir sobre duas ações: responder ou ignorar o novo chamado. Um estado neste ambiente possui características tais como: posição dos elevadores (assumindo o número do andar atual); direção dos elevadores (subindo, descendo ou parado); a intenção de pegar pessoa nos andares; e a intenção de deixar pessoas nos andares dentre outros fatores que compõem um estado. Como função recompensa tem-se um mecanismo que recompensa o agente-elevador quando o mesmo decide atender o chamado sendo o mais próximo do novo chamado ou decide ignorar o chamado no caso contrário. Analogamente, o função de recompensa pune o agente elevador que decide atender o chamado não sendo o mais próximo ou ignora o chamado no caso contrário. A proximidade entre os elevadores do novo chamado é feita através da distância heurística D, proposta neste trabalho. Com algumas iterações, os agentes aprendem os pesos de uma função de avaliação que aproximam a função de valor estado-ação. O desempenho da solução (Tempo Médio de Serviço - TMS), mostrado variando o padrão de tráfego, fluxo de pessoas, número de elevadores e número de andares, é comparável a outras propostas relatadas na literatura. O primeiro experimento é feito sendo: fixado o número de elevadores e número de andares; e variando o fluxo de pessoas e o padrão de tráfego. Dois algoritmos clássicos são comparados a solução proposta e esta apresenta uma estratégia quase dominante. O segundo experimento: fixa o fluxo de pessoas e padrão de tráfego; e varia o número de andares e número de elevadores sendo investigado o impacto dessas variáveis no TMS. O terceiro experimento compara os resultados de outras 3 solução publicadas com este trabalho onde o mesmo apresenta resultados competitivos.

Palavras-chave: Aprendizagem por reforço. Sistemas Multiagente. Controle de elevadores. Heurística de distância.

ABSTRACT

In this work, a modeling and algorithm based on multi-agent reinforcement learning are developed for the elevator group dispatch problem. The main advantage is that, together with the approximation of the function, this multiagent solution leads to a synthesis of the state space, allowing complex states to be addressed with an evaluation function. Each elevator is considered an agent who has to decide on two actions: answer or ignore the new call. A state in this environment has characteristics such as: position of the elevators (assuming the current floor number); direction of the elevators (going up, down or stopped); the intention to catch people on the floors; and the intention to leave people on the floors among other factors that make up a state. As a reward function, there is a mechanism that rewards the agent-lift when he decides to answer the call being the closest to the new call or decides to ignore the call otherwise. Similarly, the reward function punishes the elevator agent who decides to answer the call not being the closest one or ignores the call otherwise. The proximity between the elevators of the new call is made through the heuristic distance D, proposed in this work. With a few iterations, agents learn the weights of an assessment function that approximate the state-action value function. The performance of the solution (Average Waiting Time - AWT), shown by varying the traffic pattern, flow of people, number of elevators and number of floors, is comparable to other proposals reported in the literature. The first experiment is done by: fixing the number of elevators and the number of floors; and varying the flow of people and the traffic pattern. Two classic algorithms are compared to the proposed solution and it presents an almost dominant strategy. The second experiment: fixes the flow of people and traffic pattern; and the number of floors and number of elevators varies, investigating the impact of these variables on the AWT. The third experiment compares the results of 3 other published solutions with this work where it presents competitive results.

Keywords: Reinforcement Learning. Multi Agent Systems. Elevators Control. Heuristic of distance.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmos Genéticos
AR	Aprendizagem por reforço
AWT	Average Waiting Time
GA	Genetic Algorithms
HAMMQ	Heuristically-Accelerated Minimax-Q
HAMQ(λ)	Heuristically-Accelerated Minimax-Q(λ)
HAMQS	Heuristically-Accelerated Minimax-QS
HAMRL	Heuristically-Accelerated Multiagent Reinforcement Learning
HAMS	Heuristically-Accelerated Minimax-SARSA
IA	Inteligência Artificial
IC	Inteligência Computacional
MAS	Multi Agent Systems
MDP	Markovian Decison Process
MESA	Agent-based modeling in Python3+
RL	Reinforcement Learning
RNA	Redes Neurais Artificiais
SMA	Sistemas Multiagente
SMA-RL	Sistemas Multiagente utilizando Aprendizagem por Reforço
SZ	Static Zoning
TME	Tempo Médio de Espera
TMJ	Tempo Médio de Jornada
TMS	Tempo Médio de Serviço
ZE	Zoneamento Estático
ZSMG	Zero Sum Markovian Games

LISTA DE SÍMBOLOS

s	Estado
a	Ação
$Q(s,a)$	Valor para o agente de se utilizar a ação a no estado s.
$f(r)$	Função recompensa
Dis	Distância em andares da chamada
R	Tempo de serviço da chamada específica
R_{wt}	Tempo total dos passageiros fora dos elevadores
R_{jny}	Tempo total dos passageiros dentro dos elevadores
R_{stp}	Número de paradas de todos os carros
N	Número de elevadores
M	Número de andares
n	Elevador específico, tal que $n \in [1,N]$
m	Andar específico, tal que $m \in [1,M]$
θ_0	Peso 1 que compõe a função de avaliação
θ_1	Peso 2 que compõe a função de avaliação
D	Valor que informa se o elevador em questão é o mais próximo do chamado atual, sendo atribuído 1 em caso verdadeiro e -1 caso contrário
A	Valor que utiliza a ação escolhida pelo agente, sendo atribuído 1 para "atender o chamado" e -1 para "ignorar o chamado"
s'	Estado atual do agente
a'	Todas as possíveis ações a serem tomadas em s'
α	Taxa de aprendizado
r	Recompensa fornecida ao agente pelo estado atual
γ	Desconto dado a recompensas futuras
$Q'(s,a)$	Valor para o agente de se utilizar a ação a no estado s, baseado na função de valor estado-ação.
λ	Número esperado de pessoas que chegam para utilizar os elevadores em um dado intervalo de tempo (segundo)

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de elevadores em edifícios de escritórios, instalações comerciais e apartamentos têm evoluído rapidamente. Isso se deve, entre outros fatores, à concentração da população em grandes cidades e a sua consequente verticalização. Consequentemente, uma parte relevante dos edifícios se tonaram inevitavelmente altos e em grande escala. Elevadores em tais edifícios são indispensáveis a fim de utilizar seu espaço de forma eficiente e melhorar a conveniência de movimentação em prédios e construções. A fim de assegurar a capacidade de transporte suficiente em um sistema de elevadores nos dias de hoje, é necessário aumentar o número de elevadores em um mesmo local, formando um grupo de elevadores, que devem operar cooperativamente para maior economia e eficácia. Em consequência disso, vários algoritmos de controle de grupo de elevadores foram desenvolvidos até agora, usando programação matemática, Inteligência Artificial, Redes Neurais e Fuzzy, e Algoritmos Populacionais tais como Algoritmos Genéticos (Cao; Tian; Zhang, 2008; Cao; Zhou; Yang, 2008; VALDIVIELSO; MIYAMOTO; KUMAGAI, 2008; IKEDA *et al.*, 2008; TAKATA *et al.*, 2010; Valdivielso; Miyamoto, 2011; XUE, 2002).

Aprendizagem por reforço é uma outra técnica já utilizada para obter controladores ótimos ou quase ótimos para o controle de grupos de elevadores (Liu *et al.*, 2013). A aprendizagem por reforço é um tipo de aprendizagem que não depende de sinais de um "professor", mas obtém regras de ação ótimas por meio de tentativa e erro (SUTTON; BARTO, 2018). Na aprendizagem por reforço, os agentes observam o ambiente, determinam sua ação de acordo com o estado e realizam a ação. O estado do ambiente muda a partir da ação executada. O agente obtém uma recompensa de acordo com o resultado da ação. Com base na recompensa, o agente aprende o valor de avaliação para o estado e o comportamento do agente. Espera-se que os agentes obtenham um mecanismo de tomada de decisão (correspondência entre ações e estados) que maximize o retorno calculado das recompensas. Desta forma, o aprendizado por reforço pode adquirir automaticamente a política que maximiza o retorno esperado apenas definindo o espaço de estados, as ações e as recompensas para problemas a serem resolvidos usando o agente.

Uma questão crítica quando modelando o controle do despacho de grupos de elevadores para aprendizagem por reforço é a explosão do espaço de estados. Modelar o grupo de elevadores como uma Sistema Multi-Agente (SMA) tende a aumentar mais ainda o conjunto de estados (BARRIOS-ARANIBAR; GONÇALVES, 2007). Entretanto, ao invés de utilizar uma

simples tabela que armazena o valor de se utilizar a ação *a* no estado *s*, uma função de valor estado-ação será utilizada e essa abordagem resulta em reduzir o armazenamento (SUTTON; BARTO, 2018), substituindo uma grande matriz por alguns parâmetros.

Neste trabalho, um sistema multiagente (SMA) que utiliza aprendizagem por reforço (AR) para o controle de grupos de elevadores é proposto. O método de AR escolhido foi a aprendizagem Q (WATKINS; DAYAN, 1992), onde o agente associa valores a pares de estado-ação *Q(s,a)*. Na solução proposta, cada elevador é abstraído como um agente que deve escolher entre duas ações (atender o chamado ou ignorar o chamado) e a junção destes agentes formam o SMA. Nessa proposta procura-se equilibrar o compromisso entre o tamanho do espaço de estados e a curva de aprendizado propondo uma função de aproximação e uma heurística para aprender os parâmetros da função.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver e avaliar uma modelagem de aprendizagem por reforço multiagente (SMA-RL) para o projeto do controlador de despacho de grupos de elevadores competitiva com o estado da arte publicado.

A principal métrica de avaliação deste trabalho consiste em observar 3 variáveis: Tempo Médio de Espera (TME), Tempo Médio de Jornada (TMJ) e Tempo Médio de Serviço (TMS). TME representa o tempo que uma pessoa gasta esperando o elevador chegar. TMJ representa o tempo que uma pessoa gasta dentro de um elevador esperando seu respectivo andar de destino chegar. E finalmente:

$$TMS = TME + TMJ, \tag{1.1}$$

simplesmente representando a soma das duas primeiras variáveis. Essas três variáveis serão utilizadas para validar a proposta quanto aos objetivos gerais.

1.1.2 Objetivos Específicos

a) Caracterizar o problema do controle do despacho de grupos de elevadores, as soluções clássicas e as configurações de teste de validação.

b) Elaborar uma revisão conceitual em aprendizagem por reforço e aprendizagem por reforço em sistemas multiagente.
c) Revisar a literatura sobre técnicas de Inteligência Artificial (IA)/Inteligência Computacional (IC), Aprendizagem por reforço (AR) e Sistemas Multiagente utilizando Aprendizagem por Reforço (SMA-RL) aplicadas ao Controle de grupos de elevadores.
d) Propor e avaliar uma abordagem satisfatória (competitiva) para o projeto do controlador de grupos de elevadores baseada em SMA-RL e comparar o desempenho com: agentes RL; SMAs de agentes independentes e de agentes cooperativos; e com o estado da arte publicado para controle de elevadores.
e) Aplicar e avaliar a solução proposta em grupos de elevadores que apresentam novos chamados com indicação de direção e andar atual.

1.2 Organização do documento

No Capítulo 2 são apresentados conhecimentos necessários ao entendimento da solução proposta. No Capítulo 3, são apresentados alguns trabalhos que propõem uma nova forma de controle de grupo de elevadores também utilizando aprendizagem Q, além de heurísticas de aceleração de aprendizagem. No Capítulo 4 a solução deste trabalho é descrita em detalhes, a qual consiste em uma heurística para aprender uma função de aproximação da função de valor estado-ação. No Capítulo 5 são feitos experimentos que avaliam o desempenho da solução SMA proposta e no Capítulo 6 a conclusão e os trabalhos futuros desta pesquisa são apresentados.

2 FUNDAMENTAÇÃO TEÓRICA

Existem, principalmente, dois temas necessários para se entender este trabalho: Aprendizagem por reforço (mais especificamente aprendizagem Q) e Sistemas multiagente. Por questões didáticas eles abordados separadamente, entretanto uma substancial quantidade de estudos apresentam pesquisas sobre aprendizado por reforço multiagente (PANAIT; LUKE, 2005; TAN, 1998; STONE; VELOSO, 2000; YANG; GU, 2004; Busoniu; Babuska; De Schutter, 2008; BOWLING; VELOSO, 2000).

2.1 Aprendizado por reforço

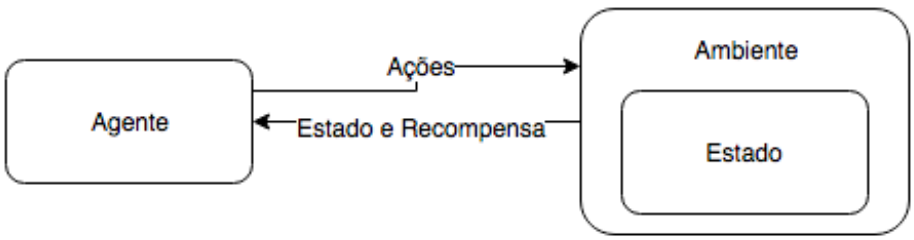
Considere, por exemplo, o problema de aprender a jogar xadrez. Um agente de aprendizagem supervisionada precisa ser informado da jogada correta para cada posição que encontra, mas tal realimentação raramente está disponível. Na ausência da realimentação de um professor, um agente pode aprender um modelo de transição para seus próprios movimentos e talvez possa aprender a prever as jogadas do adversário, mas sem alguma realimentação sobre o que é bom e o que é ruim, o agente não terá nenhuma base para decidir que movimento executar. O agente precisa saber que algo de bom aconteceu quando (acidentalmente) dá o xeque-mate no oponente ou vice versa, se for um jogo de xadrez suicida. Essa espécie de realimentação é chamada recompensa ou reforço. Em jogos como o xadrez, o reforço é recebido apenas no fim do jogo. Em outros ambientes, as recompensas vêm com maior frequência. No jogo de pingue-pongue, cada ponto marcado pode ser considerado uma recompensa; quando se aprende a engatinhar, qualquer movimento para a frente é uma realização. A estrutura animal para agentes considera a recompensa como uma parte da percepção de entrada, mas o agente deve ser "fisicamente programado" para reconhecer essa parte como uma recompensa, e não apenas como outra entrada sensória. Desse modo, os animais parecem estar programados para reconhecer dor e fome como recompensas negativas e também prazer e ingestão de alimentos como recompensas positivas. O reforço foi cuidadosamente investigado por estudiosos da psicologia animal por mais de 60 anos (RUSSELL; NORVIG, 2009).

Uma política ótima é uma política que maximiza a recompensa total esperada. A tarefa da aprendizagem por reforço consiste em usar recompensas observadas para aprender uma política ótima (ou quase ótima) para o ambiente. Na aprendizagem por reforço não supõem se nenhum conhecimento anterior do modelo ou da função de recompensa. Imagine disputar um novo jogo cujas regras não são conhecidas; depois de aproximadamente uma centena de

movimentos, o oponente anuncia: “O agente perdeu.” Em resumo, isso é a aprendizagem por reforço (RUSSELL; NORVIG, 2009).

Em muitos domínios complexos, a aprendizagem por reforço é o único caminho possível para treinar um programa com desempenho de alto nível. Por exemplo, em jogos, é muito difícil um ser humano fornecer avaliações precisas e consistentes de um grande número de posições, que seriam necessárias para treinar uma função de avaliação diretamente a partir de exemplos. Em vez disso, o programa pode ser informado de quando ganhou ou perdeu, e pode usar essa informação para aprender uma função de avaliação que forneça estimativas razoavelmente precisas da probabilidade de ganhar a partir de qualquer posição dada. De modo semelhante, é extremamente difícil programar um agente para voar em um helicóptero; ainda assim, dadas recompensas negativas por cair, colidir ou se desviar de um curso definido, um agente poderá aprender a voar por si só (RUSSELL; NORVIG, 2009). As técnicas de AR, á medida que se popularizaram, foram aplicadas a uma ampla variedade de problemas de planejamento propositivo e de controle quando nem um modelo analítico e nem um modelo de amostragem estavam disponíveis a priori (FERNáNDEZ; GARCÍA; VELOSO, 2010; MATOS *et al.*, 2011; BOWLING; VELOSO, 2002; DAS *et al.*, 2008). A Figura 1 ilustra um exemplo básico de como um agente AR funciona. O ambiente fornece estados ao agente. Cada estado tem associado a ele uma recompensa. Dada a situação, o agente por sua vez, escolhe uma ação dentre todas as ações possíveis. Essa ação altera o estado do ambiente. O novo estado possui uma nova recompensa. Assim, o ciclo é fechado e iterativamente a aprendizagem por reforço ocorre.

Figura 1 – Estrutura básica da aprendizagem por reforço



Fonte – Elaborado pelo autor

Existem 2 tipos de aprendizagem por reforço: a **aprendizagem passiva** e a **aprendizagem ativa**. Na aprendizagem passiva a política do agente é fixa e a tarefa consiste em aprender as utilidades de estados (ou pares estado-ação); isso também poderia envolver a aprendizagem

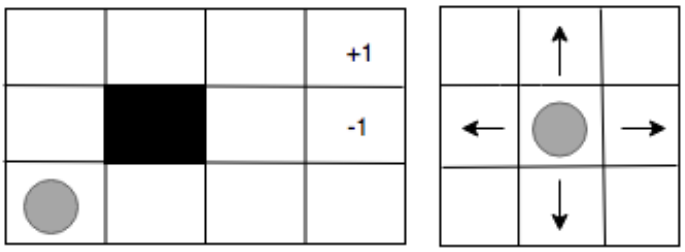
de um modelo do ambiente. O agente não decide o que fazer, mas apenas analisa o quão boa é a sua política atual. Na aprendizagem ativa o agente também deve aprender o que fazer. A principal questão é a exploração: um agente deve experimentar tanto quanto possível do seu ambiente, a fim de aprender como se comportar nele (RUSSELL; NORVIG, 2009). Nesta, seção o foco está em aprendizagem ativa.

Existem agentes AR baseados em modelos e livre de modelo. Enquanto aquele constrói internamente ao longo das experiências um modelo MDP sobre como o ambiente funciona, este se preocupa apenas com os resultados de suas ações nos estados possíveis e não constrói modelo MDP. Nesta seção, como o foco está em um método livre de modelo, o tema Markovian Decison Process (MDP) não será abordado.

2.1.1 Aprendizagem Q

A aprendizagem Q, um método bastante utilizado na literatura, é livre de modelo. Este método consiste em atribuir um valor para cada par estado-ação $Q(s,a)$. A Figura 2 ilustra um exemplo didático de ambiente. Neste, existem 11 estados possíveis, pois o quadrado preto é considerado uma parede. Existem também 4 ações, que representam os movimentos, nos 11 estados: cima; baixo; direita; e esquerda (indicada a direita da Figura 2). A partir dessas ações o agente (circulo cinza) se move no ambiente e acessa estados diferentes. Cada estado terminal fornece uma recompensa de “+1” ou “-1”. Para estados sem indicação explícita, considera-se uma recompensa de -0,04 neste exemplo didático. Esse valor sutilmente negativo estimula o agente a movimentação para que o mesmo possa encontrar o estado “+1”.

Figura 2 – Exemplo didático de ambiente



Fonte – Elaborado pelo autor

O primeiro passo para aplicar a aprendizagem Q é criar uma matriz 11x4, indicando $Q(s,a)$, onde cada linha é um estado e cada coluna uma ação e todos os valores são inicialmente 0. O método se inicia com o ambiente fornecendo o estado s do agente juntamente com a

recompensa r correspondente ao estado. Na primeira iteração o agente escolhe uma ação a partir da **função de exploração**, que será aprofundada na próxima subseção. A partir da segunda iteração, e considerando o novo estado do agente fornecido pelo ambiente s', o par s e a, ou seja $Q(s,a)$, tem seu valor atualizado através da Equação 4.2. As iterações continuam enquanto for pertinente para a aplicação, entretanto usualmente a simulação costuma acabar quando o agente chega a um estado terminal. À medida que as iterações passam, os valores de Q são atualizados e isso representa diretamente o aprendizado do agente. O algoritmo 1 mostra o pseudocódigo do que foi descrito.

Algoritmo 1: Algoritmo de aprendizagemQ(s', r')

Entrada: o estado atual s' e a recompensa atual r'
Saída: retorna a
Dados: Q, uma matriz de valores de ações indexada por estado e ação, inicializada com zero;
s, a, r, estado, ação e recompensa anteriores, inicialmente nulos;
início
  se *s.TERMINAL?* **então**
    $Q(s,a) = r$';
  fim
  se *s é não nulo* **então**
    $Q'(s,a) = Q(s,a) + \alpha(r + \gamma max_{a'} Q(s',a') - Q(s,a))$;
  fim
  $s = s'$;
  $r = r'$;
  $a =$ funçãoDeExploração(Q);
  retorna a
fim

2.1.2 Função de exploração

A função de exploração deve ser modelada considerando duas situações: se por um lado o agente deve explorar todo ambiente a fim de garantir que encontrou a melhor política possível, por outro lado o agente deve ter noção de quando parar a exploração e começar a utilizar o conhecimento adquirido, visando maximizar sua recompensa. Existem várias alternativas para balancear a função de exploração entre explorar e maximizar a recompensa. A Equação 2.1

mostra um exemplo de função de exploração:

$$f(Q,N) = \begin{cases} R^{+} & \text{Se } N(s,a) < N_{min}, \\ Q(s,a) & \text{Caso contrário} \end{cases} \tag{2.1}$$

onde *N(s, a)* é número de vezes que o agente já executou a ação a no estado s, R^{+} é a maior recompensa possível que o ambiente pode oferecer de tal forma que o agente considera que vai muito bem recompensado com a exploração, e N_{min} é o número mínimo de vezes que o agente deve escutar cada ação em cada estado. Assim, o agente procurar visitar todas as ações em disponíveis em todos os estados disponíveis N_{min} vezes e então vai paulatinamente considerando o real valor da matriz Q, que representa o conhecimento adquirido, o aprendizado. Para utilizar a Equação 2.1 como função de exploração naturalmente se faz necessário guardar também a matriz N (além de Q) ao longo das iterações. Um outro exemplo de função de exploração pode ser visto na Equação 3.1, onde o agente explora em uma porcentagem p das vezes e utiliza o conhecimento adquirido em uma porcentagem $1-p$. O coeficiente p costuma variar de 0.1 a 0.3.

2.1.3 Generalização da Aprendizagem Q

Até agora, presumiu-se que a função de valor estado-ação aprendida pelos agentes é representada em forma tabular com um valor de saída para cada tupla de entrada. Tal abordagem funciona razoavelmente bem para pequenos espaços de estados, mas o tempo para convergência e o tempo por iteração aumentam com rapidez à medida que o espaço de estados fica maior. O xadrez e o gamão são minúsculos subconjuntos do mundo real, ainda que seus espaços de estados contenham um número de estados da ordem de 10^{20} e 10^{40} estados, respectivamente. Seria absurdo supor que o agente tivesse de visitar todos esses estados a fim de aprender a jogar.

Um modo de manipular tais problemas é usar uma função de aproximação, que simplesmente significa usar qualquer tipo de representação para a função de valor estado-ação (Q) que não seja uma tabela de consulta. A representação é visualizada como aproximada porque talvez função de valor estado-ação não possa ser representada na forma escolhida. Sintetizando, a aprendizagem Q consiste em aprender uma função valor da ação a no estado s. Caso seja definida a função a priori não haverá aprendizagem. Entretanto, se a função for conhecida imprecisamente, pequenos ajustes poderão ser feitos para aproximar a função da melhor política possível no ambiente.

Um das principais contribuições deste trabalho consiste em propor um função de aproximação específica para o controle de elevadores visando a diminuição do tempo médio de serviço dos usuários, e fornecendo resultados competitivos com os já publicados na literatura. Uma significativa parte desta seção foi escrita com o auxílio de (RUSSELL; NORVIG, 2009)

2.2 Sistemas multiagente

Várias técnicas computacionais são inspiradas pela realidade humana, desenvolvidas com o objetivo de modelar sistemas de forma eficiente. Os sistemas multiagente são influenciados pela sociologia, com suas principais características funcionais definidas a partir de análises realizadas sobre as propriedades da sociedade e as propriedades dos indivíduos.

Pode-se descrever uma sociedade como um grupo formado por indivíduos interagindo entre eles de tal forma que cada um busca cumprir seus objetivos traçados. A interação social ajuda a superar limitações tanto em termos físicos como em termos cognitivos dos indivíduos e a conformação de organizações e grupos facilita a solução de problemas (MATARIC, 1995). Neste sentido, os sistemas multiagente tem por objeto de estudo a coletividade onde o foco é a interação e organização dos indivíduos que no caso são chamados de agentes.

2.2.1 Definições de sistemas multiagente

Sistema multiagente (MAS) não tem uma definição universal. Segundo (HOEN *et al.*, 2006) são definidos como um área da inteligência artificial que enfatiza o comportamento conjunto dos agentes, com algum grau de autonomia e com certa complexidade decorrente de suas interações. Já, (PANAIT; LUKE, 2005) definem MAS como o ambiente onde existe mais de um agente, onde os agentes interagem uns com outros e além disso existem certas restrições no ambiente tais como que alguns agentes não conhecem nada do ambiente enquanto outros agentes conhecem.

No presente trabalho, definimos um sistema multiagente (MAS - Multiagent System) como um software no qual dois ou mais agentes interagem num mesmo ambiente de forma a cooperarem e/ou competirem na solução de problemas. Os agentes devem ter capacidades básicas de percepção e de ação sobre o ambiente. Esses sistemas têm duas propriedades fundamentais: a autonomia dos agentes e sua organização.

2.2.2 Exemplos de sistemas multiagente

O uso de sistemas multiagente tornou-se comum na solução de problemas computacionais como por exemplo comércio eletrônico (Pan *et al.*, 2008), agendamento de rotas de caminhões (MES; HEIJDEN; HARTEN, 2007), modelagem da demanda de energia em cidades ou países (TOKSARI, 2007), recuperação de imagens baseada em conteúdo (DIMITRIADIS; MARIAS; ORPHANOUDAKIS, 2007), entre outros, e na solução de problemas envolvendo robôs como envio de cartas utilizando robôs (CARRASCOSA *et al.*, 2008), missões de resgate de vítimas de desastres (ROOKER; BIRK, 2005), geração de mapas de ambientes estruturados (Rocha; Dias; Carvalho, 2005) entre outros.

2.2.3 Tipos de aprendizagem por reforço em MAS

As interações em sistemas multiagente podem ser de natureza cooperativa, competitiva ou mista. Baseado na aplicação deste trabalho, a natureza cooperativa será enfatizada.

Os algoritmos de aprendizado por reforço foram introduzidos inicialmente para aprendizado de um único agente. Eles são também aplicados em sistemas multiagente segundo diferentes modos. A primeira maneira é modelar o time de agentes como um agente só, o que neste trabalho é denominado de *aprender como um time*. Um outro modo é aplicar os algoritmos sem nenhuma modificação, mas de forma independente para cada agente o que é denominado de *aprender independentemente*. A terceira e última forma, até o momento realizada, é a de *aprender ações conjuntas*. Isto quer dizer que cada agente aprende a executar uma ação pensando em combiná-la com as ações que os outros agentes irão executar. Na solução proposta neste trabalho, ainda que os agentes influenciem no estado global através de suas ações, os mesmos *aprendem independentemente*. Mas detalhes serão apresentados no capítulo 4.

2.2.4 Aprendendo como um time (TL)

O paradigma de aprendizado em sistemas multi-agente onde os agentes aprendem como um time baseia-se na modelagem do time todo como um agente único. A grande vantagem deste paradigma é que os algoritmos não precisam ser modificados. Porém, em aplicações reais como em robótica podem ser muito difíceis de implementar pelo fato de que eles precisam ter todo o processamento e a informação centralizados.

Um exemplo deste paradigma usando aprendizado por reforço é o trabalho de Kok e

Vlasis, que modelam o problema de colaboração em sistemas multi-agente como um processo de decisão markoviano (KOK; VLASSIS, 2004). Neste e outros trabalhos similares o principal problema é o fato que a aplicabilidade dos mesmos torna-se inviável quando o número de agentes aumenta, já que o número de estados e ações também aumenta de forma exponencial.

Os algoritmos genéticos, por sua vez, não são muito afetados pelo crescimento exponencial no número de estados e ações (SEN; SEKARAN, 1996; SAŁUSTOWICZ; WIERING; SCHMIDHUBER, 1998; IBA, 1999). Porém, a necessidade de ter o processamento e a informação centralizados ainda é um problema impossível de ser resolvido neste paradigma, sendo uma de suas desvantagens.

Ainda, é importante observar que este paradigma foge do conceito de sistema multiagente, já que, em sistemas multi-agente cada agente deve ser autônomo, não podendo existir um controle global dos mesmos (Shen; Wu, 2008). Fato pelo qual este paradigma não foi utilizado para a construção desta solução.

2.2.5 Aprendendo independentemente (IL)

Os problemas relatados no paradigma de aprendizado como um time podem ser resolvidos ao implementar o algoritmo de aprendizado de forma independente em cada agente. Vários trabalhos mostram resultados promissores ao aplicar este paradigma (SEN *et al.*, 1994; KAPETANAKIS; KUDENKO, 2002; TUMER; AGOGINO; WOLPERT, 2002; BARRIOS-ARANIBAR; ALSINA, 2006).

A pergunta que logo aparece ao se descrever este paradigma é se os algoritmos tradicionalmente introduzidos para tratar problemas com um agente único terão resultados ótimos ou pelo menos condizentes com a solução do problema. Os trabalhos de Sen (SEN *et al.*, 1994) e de Barrios-Aranibar (BARRIOS-ARANIBAR; ALSINA, 2006) mostram que estes algoritmos são aplicáveis em sistemas multiagente. Claus e Bouitilier (CLAUS; BOUTILIER, 1998) exploram o uso de aprendizes independentes em jogos repetitivos, mostrando empiricamente que a proposta têm condições de atingir um equilíbrio de Nash mas não o equilíbrio ótimo.

Estes resultados são importantes se os analisarmos no tocante à natureza dos algoritmos utilizados. Pode ser observado que os algoritmos de aprendizado por reforço visam levar o agente a executar um conjunto de ações que lhe proporcionem a maior utilidade (maiores recompensas). Segue que, em problemas envolvendo vários agentes, existe a possibilidade que a combinação de estratégias individuais ótimas não representem necessariamente uma estratégia

do time também ótima.

Na tentativa de resolver este problema, diversos trabalhos foram desenvolvidos. Estes trabalhos modificam a forma de calcular as recompensas para os agentes individualmente, já que o paradigma trabalha com independência dos agentes e não é possível incluir informações dos colegas. Assim, Um exemplo destas tentativas é o trabalho de Kapetanakis e Kudenko (KAPETANAKIS; KUDENKO, 2002), que propõem uma nova heurística para calcular os valores de recompensa para as ações baseada na frequência com que uma ação teve sua máxima recompensa. Eles mostram, empiricamente, que a heurística tem capacidade de convergir para o equilíbrio de Nash ótimo em jogos repetitivos com dois agentes. Ainda no trabalho em questão, a heurística é testada com até quatro agentes em jogos repetitivos, onde só um deles utiliza a heurística de frequência, mostrando que a capacidade de convergir para o equilíbrio ótimo aumenta, porém não é garantida (KAPETANAKIS; KUDENKO, 2005).

Um outro trabalho que explora modificações nas escolhas de recompensas é o de Tumer et. al. onde é abordado o problema de escolha das recompensas corretas em aprendizado independente (TUMER; AGOGINO; WOLPERT, 2002). Os autores propõem um algoritmo que utiliza conceitos de inteligência coletiva para obter melhores resultados que os obtidos ao aplicar os algoritmos sem modificação alguma.

Um outro exemplo nesta linha é o trabalho de Mataric (MATARIĆ, 1997), o qual propõe uma metodologia que envolve minimização do espaço de aprendizado através do uso de comportamentos e condições, e tratando diretamente o problema de atribuição de recompensas usando funções de recompensa heterogêneas e estimadores de progresso.

Mesmo obtendo bons resultados em problemas simples como os jogos repetitivos ou jogos estocásticos com poucos agentes, um problema neste paradigma, que ocorre à medida que o número de agentes aumenta, é que os algoritmos tradicionais foram criados para casos em que o ambiente não muda, quer dizer para ambientes onde as recompensas são estáticas. Porém, em sistemas multiagente, as recompensas podem mudar com o tempo, uma vez que as ações dos outros agentes irão influenciar nas mesmas.

2.2.6 Aprendendo ações conjuntas (JAL)

Uma das formas de resolver o problema do paradigma anterior é aprender a melhor resposta às ações dos outros agentes. Neste contexto, aparece o paradigma de aprender ações conjuntas, onde cada agente deverá aprender qual o melhor valor ao executar suas ações em

combinação com as ações dos outros (ação conjunta). Ao intuir um modelo dos outros agentes, ele deverá calcular qual a melhor ação às supostas ações a serem executadas pelos colegas e/ou adversários (KAPETANAKIS; KUDENKO; STRENS, 2003; CHALKIADAKIS; BOUTILIER, 2003; GUO *et al.*, 2007).

Claus e Bouitilier exploram o uso deste paradigma em jogos repetitivos (CLAUS; BOUTILIER, 1998) mostrando empiricamente que a forma básica do algoritmo converge para um equilíbrio de Nash mas não garante a convergência para o equilíbrio ótimo. Porém, os autores indicam que, diferentemente dos algoritmos aplicados para aprender de forma independente, este paradigma pode ser melhorado se forem melhorados os modelos dos outros agentes.

Outros exemplos de algoritmos baseados neste paradigma incluem o trabalho de Suematsu e Hayashi que propõem um algoritmo para aprender ações conjuntas que garante a convergência para um equilíbrio de Nash (SUEMATSU; HAYASHI, 2002). O trabalho de Banerjee e Sen (BANERJEE; SEN, 2007) propõe um algoritmo de aprendizado de ações conjuntas condicional, no qual os agentes aprendem a probabilidade condicional de uma ação executada pelo oponente ser ótima, dadas as suas próprias ações e usa esta probabilidade para escolher suas ações futuras.

Outra forma encontrada deste paradigma é o aprendizado implícito de ações conjuntas, proposto por Lauer e Riedmiller (Lauer; Riedmiller, 2004). Nesta proposta os autores armazenam os valores Q das ações conjuntas, porém não as definem explicitamente como nos trabalhos anteriores.

O principal problema deste paradigma é dado pela quantidade de combinações de estados e ações que cresce exponencialmente a medida que número de estados, ações e/ou agentes crescem. A Tabela 1 mostra um resumo das características das três abordagens introduzidas.

Tabela 1 – Comparação tipos de aprendizagem introduzidos: Team Learning, Independent Learning e Join Actions Learning

	TL	**IL**	**JAL**
Processamento	Centralizado	Distribuido	Distribuido
Informação do colega?	-	Não	Sim
Comunicação?	-	Não	Sim
Auto-organização?	-	Não	Não
Ambiente	Estático	Estático	Estático
Exige sincronia?	Sim	Não	Sim
Mais agentes			
Ações aumentam?	Exponencial	Não	Exponencial
Armazenamento cresce?	Exponencial	Não	Exponencial
Processamento cresce?	Exponencial	Não	Exponencial
Mais ações			
Armazenamento cresce?	Exponencial	Linear	Exponencial
Processamento cresce?	Exponencial	Linear	Exponencial

Fonte – (ARANIBAR, 2009)

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentadas algumas soluções que utilizam aprendizagem por reforço com objetivo de controlar grupos de elevadores, sistemas multiagente e heurísticas.

3.1 Heurísticas para acelerar o aprendizado por reforço multiagente

Em (Bianchi *et al.*, 2014), é proposto uma classe de algoritmos que visa acelerar o aprendizado por reforço multiagente através de heurísticas contidas na função de exploração. A classe HAMRL (Heuristically-Accelerated Multiagent Reinforcement Learning) é testada em jogos markovianos de soma zero (ZSMG), e é subdividida em 4 algoritmos:

- O Heuristically-Accelerated Minimax-Q (HAMMQ), que é aplicado ao algoritmo Minimax-Q.
- O Heuristically-Accelerated Minimax-Q(λ) (HAMQ(λ)), que é aplicado ao algoritmo Minimax-Q(λ).
- O Heuristically-Accelerated Minimax-QS (HAMQS), que é aplicado ao algoritmo Minimax-QS.
- O Heuristically-Accelerated Minimax-SARSA (HAMS), que é aplicado ao algoritmo Minimax-SARSA.

A heurística funciona essencialmente da mesma maneira. A diferença consiste apenas em ser utilizada em algoritmos de ZSMG diferentes, que são variações do algoritmo minimax.

Como já dito, todo agente de aprendizagem por reforço utiliza uma função de exploração em sua regra de seleção ações e uma possibilidade frequentemente utilizada é a seguinte:

$$\pi(s) = \begin{cases} arg\ max_a \min_o Q(s,a,o) & \text{Se } p \geq \varepsilon, \\ a_{aleatório} & \text{Caso contrário} \end{cases} \tag{3.1}$$

onde $p \in [0,1]$ e $\varepsilon \in [0,1]$, ou seja, com probabilidade $p \geq \varepsilon$ o agente escolhe a ação a de maior valor considerando o estado s e a melhor ação o utilizada pelo oponente; e caso contrário, escolhe uma ação aleatória representando a exploração do agente. Em (Bianchi *et al.*, 2014), a seguinte

função de exploração é proposta:

$$\pi(s) = \begin{cases} arg\ max_a \min_o [Q(s,a,o) + \xi H_t(s,a,o)^\beta] & \text{Se } p \geq \varepsilon, \\ a_{aleatório} & \text{Caso contrário} \end{cases} \tag{3.2}$$

muito similar a Equação 3.1, com a diferença de que a heurística H_t influencia na decisão final da ação. ξ e β são variáveis que controlam o peso da heurística. O t subscrito indica que a heurística pode ser não estacionária. Neste trabalho, não é proposta exatamente uma heurística fixa. A heurística pode ser construída por experiência prévia em domínios distintos, por observações ao longo das experiências dos agentes, ou mesmo por um conjunto de regras condição-ação.

Para obter resultados, é escolhido o domínio de futebol de robôs e é comparado as variações dos algoritmos minimax com os respectivos algoritmos acelerados heuristicamente, mostrando que o saldo médio de gols é maior ou igual com a classe de algoritmos HAMRL.

Comparando (Bianchi *et al.*, 2014) com este trabalho, a diferença é que enquanto a heurística está na função de exploração em (Bianchi *et al.*, 2014), neste trabalho a heurística proposta é: específica para o controle de elevadores; e está diretamente ligada a função de avaliação *Q(s,a)*. Um heurística sendo utilizada na função de exploração sugere quais ações são mais valiosas que outras, acelerando assim o aprendizado do agente. Um heurística sendo utilizada na função de avaliação interfere diretamente no aprendizado do agente, não necessariamente acelerando, mas "ensinando" quais ações trazem maior recompensa. Aquela visa acelerar o aprendizado, enquanto esta aspira a influenciar no aprendizado em si.

É perfeitamente factível que haja uma segunda heurística na função de exploração utilizando as contribuições de (Bianchi *et al.*, 2014), e esta na verdade, é uma das tarefas que serão realizadas futuramente.

3.2 Aprendizagem Q aplicada a grupos de elevadores Multi-Car

Em (Ikuta; Takahashi; Inaba, 2013), é proposto um sistema com um único agente onde existem 4 modos de operação: estratégia de transporte, estratégia de passageiros, estratégia de zona e estratégia de diferença. A primeira estratégia, estratégia de transporte, atribui chamadas ao elevador com o menor tempo de transporte previsto. A segunda estratégia, estratégia de passageiros, atribui chamadas ao elevador em que o número de passageiros é o menor. A terceira estratégia é a estratégia de zona; na estratégia, o alcance de cada elevador é especificado antecipadamente. As chamadas são atribuídas ao elevador que pode transportar chamadas para

andares de destino em seu alcance. Finalmente, a quarta estratégia é a estratégia de diferença; na estratégia, a chamada atual é atribuída a um elevador com a menor diferença de tempo entre os tempos médios de serviço previstos (TMS) calculados com a chamada atual e a TMS prevista calculada sem a chamada atual. A estratégia de zona existe porque (Ikuta; Takahashi; Inaba, 2013) aborda o problema de múltiplos elevadores em um único canal, incentivando assim o uso desta estratégia. O sistema funciona da seguinte maneira: dado o estado atual, o agente deve decidir (aprender) sob qual estratégia operar, ou seja, existem 4 ações disponíveis para o agente que corresponde as 4 estratégias. Existem 2 variáveis que determinam o estado atual: X e Y.

- X é o número total de chamadas ativas no sistema, contado com as chamadas de pessoas que desejam entrar nos elevadores e com as chamadas de pessoas que desejam sair dos elevadores.
- Y é a distância em andares da nova chamada, ou seja, a variação entre o andar de chegada e o andar de saída.

X e Y são discretizados diminuindo assim o espaço de estados do ambiente e possibilitando o armazenamento de um tabela com os valores dos pares estado-ação.

A função recompensa é:

$$f(r) = \frac{Dis}{R} \tag{3.3}$$

onde Dis é a distância em andares da chamada e R é o tempo de serviço da chamada específica. Por exemplo, para uma pessoa que decide se locomover do andar 2 ao 5 e passou 30 segundos para utilizar os elevadores, logo $f(r) = 0{,}1$.

Nos resultados, o Tempo Médio de Serviço (TMS) é obtido sendo variado: o fluxo de pessoas que utilizam o sistema; e o padrão de tráfego dos chamados. O padrão de tráfego possui 3 possibilidades: ordinário, descida do pico e subida ao pico. Esses padrões serão explicados no Capítulo 5.

3.3 Aprendizagem Q e RNA recorrentes aplicada a grupos de elevadores

Em (Liu *et al.*, 2013), é proposto um sistema com um único agente onde dado um novo chamado, o agente deve decidir (aprender) qual elevador dos N elevadores disponíveis enviar para atender o chamado. A modelagem deste sistema é notavelmente mais complexa pois o espaço de estado é grande demais para que *Q(s,a)* seja armazenado em uma tabela, tornando-se necessário a existência de uma função de avaliação. A solução contida neste trabalho utilizou exatamente o mesmo estado de (Liu *et al.*, 2013), então para uma descrição detalhada deste

estado a Seção 4.2 deve ser consultada. O conjunto de ações, como já intuído, consiste em enviar um dos N elevadores para atender o chamado. A função de recompensa é:

$$f(r) = \sqrt{R_{wt} + R_{jny} + R_{stp}} \tag{3.4}$$

onde R_{wt} representa o tempo total dos passageiros fora dos elevadores, R_{jny} representa o tempo total dos passageiros dentro dos elevadores e R_{stp} representa o número de paradas de todos os carros. A variação de tempo influencia diretamente no cálculo de R_{wt}, R_{jny} e R_{stp}, de tal forma que se os passageiros são atendidos rapidamente a recompensa diminuí e se os passageiros demoram para para deixar os elevadores, então a recompensa aumenta. Dessa forma, o agente é programado para escolher as ações com o menor valor de $Q(s,a)$.

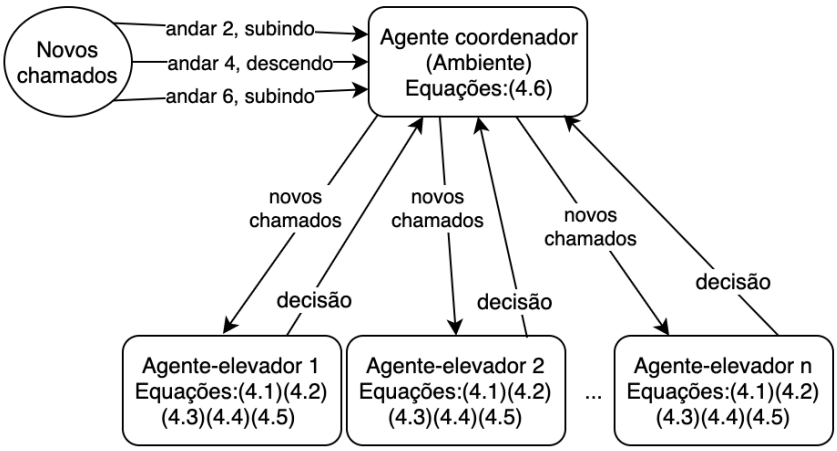
É relevante comentar sobre a função de avaliação de (Liu *et al.*, 2013), que utiliza N redes neurais recorrentes, uma para cada elevador, para retornar o valor de usar a ação a no estado s. Como resultado, a solução é comparada com outros dois algoritmos de controle de elevadores já publicados na literatura, mostrando desempenho comparável e na maior parte das vezes superior.

4 METODOLOGIA

4.1 Definição do problema

Para que qualquer algoritmo de AR seja utilizado, o ambiente precisa ser modelado adequadamente. Especificamente, é necessário definir um **conjunto de estados**, um **conjunto de ações** e uma **função recompensa** que guie o aprendizado do agente. Nas seções subsequentes cada módulo do problema será explicado para a compreensão da solução proposta. A Figura 3 introduz a solução proposta a nível de comunicação e coordenação entre as 3 principais entidades do sistema: os chamados executados por cada pessoa; o ambiente atuando como agente coordenador; e os agentes-elevador.

Figura 3 – Diagrama de comunicação e coordenação entre as principais entidades do sistema



Fonte – Elaborado pelo autor

A Figura 3 explicita as equações que serão utilizadas em cada entidade, e indica que todos os novos chamados são enviados a todos os agentes-elevador, onde cada um tem autonomia para decidir que ação utilizar a cada novo chamado. As equações serão explicadas ao longo deste capítulo.

4.2 Conjunto de estados

Um estado neste ambiente, considerando N elevadores e M andares, é composto por:

- Uma lista de tamanho N contendo as posições dos N elevadores, que significa M^N possibilidades de estados

- Uma lista de tamanho *N* contendo as direções dos *N* elevadores (subindo, parado ou descendo), que significa 3^N possibilidades de estados
- Uma lista de tamanho *2xM - 2* que indica os botões apertados nos *M* andares (seta para cima e para baixo no andar *2 a M-1*; seta para cima no andar *1*; e seta para baixo no andar *M*) que significa 2^{2M-2} possibilidades de estados
- Uma matriz *MxN* com valores binários, onde 1 signfica que o elevador *n* pretende **pegar** alguma(s) pessoa(s) no andar *m*, com *n ∈[1,N] e m ∈[1,M]*, que significa 2^{MN} possibilidades de estados
- Uma matriz *MxN* com valores binários, onde 1 signfica que o elevador *n* pretende **deixar** alguma(s) pessoa(s) no andar *m*, com *n ∈ [1,N] e m ∈ [1,M]*, que significa 2^{MN} possibilidades de estados

Considerando os tipos de aprendizagem multiagente descritos na subseção 2.2.3, cada agente desta solução multiagente aprende independentemente. Entretanto, pode-se notar que o estado é global para todos os agentes da modelagem, onde cada agente influência no estado atual. Logo, o estado global funciona como canal de comunicação indireta entre os agente ainda que eles aprendam independentemente. O fato da solução proposta utilizar um estado global aumenta exponencialmente a quantidade de estados com a entrada de *N* elevadores e *M* andares. Mais especificamente, considerando um ambiente com *N* elevadores e *M* andares, temos que o número total de estados é:

$$\begin{aligned} S &= M^N 3^N 2^{2M-2} 2^{MN} 2^{MN} \\ &= M^N 3^N 2^{2M-2} 2^{2MN} \\ &= M^N 3^N 2^{2M-2+2MN} \\ &= M^N 3^N 2^{2(M+MN-1)} \end{aligned}$$

Tomando como exemplo um ambiente de 6 andares e 3 elevadores o número de estados possíveis é de aproximadamente 4.10^{17}. Assim, se torna inviável armazenar uma tabela de valores com pares estado-ação sendo esse um cenário adequado para se utilizar uma função de avaliação.

4.3 Conjunto de ações

Neste trabalho, cada elevador é considerado um agente, e os mesmos juntamente com o ambiente compõem o sistema multiagente, onde o aprendizado de cada agente é independente

(ARANIBAR, 2009). Dado que um novo botão foi apertado, cada agente deve decidir entre as seguintes duas ações:

- Atender o chamado
- Ignorar o chamado

A decisão será escolhida pela função de avaliação descrita a seguir. Caso o botão seja ignorado por todos os agentes, o mesmo botão é reenviado aos agente até que ao menos um decida atender o chamado. Isso necessariamente acontecerá com a mudança de estado do sistema por outras ações e chamadas.

4.4 Função de avaliação

A função de avaliação é inserida para atribuir um valor a utilizar a ação a no estado s. Aqui definiu-se uma função linear nos parâmetros, simples. Funções de avaliação lineares são frequentemente utilizadas em aprendizagem Q (WATKINS; DAYAN, 1992). Ela é definida por:

$$Q(s,a) = \theta_0 + \theta_1 DA \tag{4.1}$$

onde θ_0 e θ_1 são os pesos que serão iterativamente aprendidos, A assume valor 1 para "atender o chamado" e -1 para "ignorar o chamado", e D assume valor 1 caso o elevador em questão seja o mais próximo do chamado atual, e -1 caso contrário. O chamado atual é identificado pelo botão apertado, que informa o andar e a direção que a pessoa deseja ir. D utiliza uma distância heurística que será explicada em uma seção adiante.

4.4.1 Origem da função de avaliação adotada

Todas as vezes que um agente-elevador deve decidir entre *atender o chamado* ou *ignorar o chamado* a função de avaliação é consultada. O objetivo da função de avaliação proposta neste trabalho, que é indicada na Equação 4.1, é atribuir valores maiores a pares de estado-ação ($Q(s,a)$) considerados interessantes ao agente-elevador do que a pares de estado-ação ($Q(s,a)$) considerados repulsivos ao agente-elevador. Para isso, foi adotada a ideia de forçar tanto A como D assumir unicamente os valores 1 e -1, pois assim:

- Se a ação escolhida for *atender o chamado* e o agente-elevador em questão for o mais próximo do chamado atual de acordo com a função heurística D, então $A = 1$ e $D = 1$, assim $Q(s,a) = \theta_0 + \theta_1$

- Se a ação escolhida for *ignorar o chamado* e o agente-elevador em questão for o mais próximo do chamado atual, então $A = -1$ e $D = 1$, assim $Q(s,a) = \theta_0 - \theta_1$
- Se a ação escolhida for *atender o chamado* e o agente-elevador em questão **não** for o mais próximo do chamado atual, então $A = 1$ e $D = -1$, assim $Q(s,a) = \theta_0 - \theta_1$
- Se a ação escolhida for *ignorar o chamado* e o agente-elevador em questão **não** for o mais próximo do chamado atual, então $A = -1$ e $D = -1$, assim $Q(s,a) = \theta_0 + \theta_1$

Dessa forma, espera-se que θ_0 e θ_1 iterativamente se aproximem de valores que reflitam o ambiente real, considerando que ambos sempre assumirão valores positivos, já que, após testes, tais valores positivos sempre foram ratificados. Pode-se notar que nos dois casos em que A e D assumem valores diferente, $Q(s,a) = \theta_0 - \theta_1$, um valor necessariamente menor que $Q(s,a) = \theta_0 + \theta_1$, para valores positivos de θ_0 e θ_1. Assim, pares de estado-ação com a política correta tendem a possuir valores maiores que pares de estado-ação com a política incorreta.

4.5 Aprendizagem Q

A aprendizagem Q foi utilizada como algoritmo de Aprendizagem por Reforço. Com isso a solução proposta possui a seguinte função de valor estado-ação (RUSSELL; NORVIG, 2009):

$$Q'(s,a) = Q(s,a) + \alpha(r + \gamma max_{a'} Q(s',a') - Q(s,a)) \tag{4.2}$$

onde s é o último estado do agente, a é a última ação do agente, r é a recompensa fornecida pela ação a ter sido escolhida no estado s, s' é o estado atual do agente, a' assume todas as possíveis ações a serem tomadas no estado s', α é a taxa de aprendizado e γ é o desconto dado a recompensas futuras.

Para encontrar $Q'(s,a)$, todos $Q(x,y)$ da Equação (4.2) são obtidos através da Equação (4.1). Com $Q'(s,a)$ obtido, o aprendizado ocorre através da atualização dos pesos, baseado na regra de Widrow-Hoff (RUSSELL; NORVIG, 2009), ou regra delta, da seguinte forma:

$$\theta_0 = \{\theta_0 + \alpha[Q'(s,a) - Q(s,a)]\} \; mod \; 2 \tag{4.3}$$

$$\theta_1 = \theta_1 + \alpha[Q'(s,a) - Q(s,a)]DA \tag{4.4}$$

onde α é a taxa de aprendizado.

Assim, à medida que os botões dos elevadores são apertados, os chamados ocorrem, os agentes decidem atender ou ignorar os chamados, os pesos são atualizados e os agentes convergem para uma política ótima de acordo com a função de avaliação e as recompensas, que serão explicadas mais a frente.

Para que os agentes possam convergir é necessário que nas equações (4.1) e (4.2) aconteçam em somas e subtrações, de tal forma que essas somas e subtrações eventualmente consigam convergir. Na equação (4.1) o resultado de "DT" assumi valores positivos e negativos ao longo das iterações, garantindo assim somas e subtrações. No entanto, na equação (4.2), não existe nenhum multiplicador que assume valores positivos e negativos. Assim, foi adotada a estratégia de se utilizar mod 2 no segundo membro inteiro de forma a limitar o crescimento indefinido de θ_0. Sem o mod 2, ou seja, com θ_0 crescendo indefinidamente, a solução contida neste trabalho não apresentava bons resultados. Os autores acreditam que isso se dá pelo fato de θ_0 atuar como estabilizador θ_1 enquanto este converge.

Em aprendizagem por reforço uma questão importante é a função de exploração. Ela permite que soluções sejam buscadas fora do caminho guloso e assim explorar melhor o espaço de soluções. Neste trabalho adotou-se, após testes, a seguinte política de exploração:

- Em 20% do tempo o agente escolhe uma das duas ações ao acaso
- Em 80% do tempo o agente escolhe a ação de maior valor baseado na Equação (4.1)

4.6 Distância heurística de D

O Algoritmo 2 mostra o pseudocódigo da distância heurística de D. Toda vez que o mesmo é chamado, uma lista com a distância heurística de todos os agente para o determinado andar e sentido é calculada. *D* será 1 se o respectivo agente possui a menor distância em comparação com os outros agentes e será -1 caso contrário. O agente com menor distância tem o D retornado como 1. Essa distância heurística depende de duas variáveis: *dAndares* e *dParadas*. *dAndares* está relacionado a quantidade andares que serão percorridos e *dParadas* a quantidade de paradas que serão realizadas. Nesta heurística, parar uma vez possui a mesma distância que se movimentar por dois andares, logo

$$d = dAndares + 2dParadas \tag{4.5}$$

Algoritmo 2: Cálculo de distância heurística getD(s, idButtom, agent)

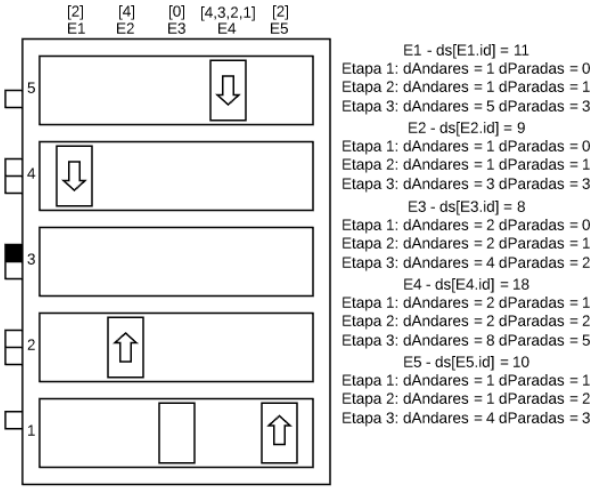
```
Entrada: recebe o estado atual; o id do botão apertado; e o agente
Saída: retorna 1 ou -1
início
    ds ←lista inicializada com N zeros;
    andarChamada e sentidoChamada são inicializados com o id do botão apertado;
    para cada a ∈ Agentes faça
        dAndares ← 0;
        dParadas ← 0;
        contabiliza o número de andares e número de paradas até andarChamada;
        dParadas ← dParadas + 1;
        contabiliza o número de andares e número de paradas até o primeiro/último andar
          baseado em sentidoChamada;
        ds[a.id] = dAndares + 2dParada;
    fim
    se min(ds) = ds[agent.id] então
        retorna 1
    fim
    senão
        retorna -1
    fim
fim
```

Existem três etapas que acumulam essas duas variáveis. A primeira é quando o agente pretende chegar no andar do chamado; a segunda simplesmente soma mais uma parada, que é parada referente ao chamado; e a terceira utiliza o sentido do chamado para supor aonde o agente deverá ir depois que pegar a(s) pessoa(s) do chamado. Se a direção do chamado é subindo, então a heurística supõem que o chamado será finalizado no último andar. Se a direção do chamado é descendo, então a heurística supõem que o chamado será finalizado no primeiro andar.

Um exemplo concreto está mostrado na Figura 4. Supondo um sistema com 5 elevadores atuando em 5 andares. O elevador E1, que está no andar 4, está indo pegar alguém no andar 2, como indicado no topo da Figura 4. E2, que está no andar 2, está indo deixar alguém no andar 4, como indicado no topo da Figura 4. E3 está ocioso no andar 1. E4 está no andar 5 e deve pegar/deixar pessoas nos andares 4, 3, 2 e 1. E5 está no andar 1 indo deixar uma pessoa no andar 2. Então, dado este estado, um novo chamado acontece no terceiro andar com a direção subindo. A Figura 4 mostra uma imagem que sintetiza o que foi descrito.

O próximo passo é calcular a distância de cada elevador para o novo chamado, de acordo com a função heurística de D.

Figura 4 – Diagrama de um estado exemplo com um novo chamado no andar 3 subindo, contendo as distâncias dos 5 elevadores



Fonte – Elaborado pelo autor

4.6.1 Elevador 1

E1 move 1 andar (do andar 4 para o andar 3) e finaliza a primeira etapa, chegando ao andar inicial do chamado (*dAndares* = 1 e *dParadas* = 0).

A segunda etapa sempre soma mais 1 a "*dParadas*", que representa a abertura e o fechamento das portas no andar 3 para que a pessoa possa entrar no elevador (*dAndares* = 1 e *dParadas* = 1).

Na terceira etapa, considerando que o sentido do chamado é subindo, E1 considera que o andar de destino do novo chamado é o andar 5 (pior caso) e agora existem dois andares para ir, que é o andar 2 e o andar 5. Como antes do novo chamado acontecer E1 já possuía a intenção de pegar alguém no andar 2, o mesmo dá prioridade ao andar 2. Assim, são simulados os seguintes passos com E1:

- Se desloca do andar 3 para o andar 2 (*dAndares* = 2 e *dParadas* = 1)
- Realiza uma parada para pegar alguém no andar 2 (*dAndares* = 2 e *dParadas* = 2)
- Se desloca do andar 2 para o andar 5 (*dAndares* = 5 e *dParadas* = 2)
- Realiza uma parada no andar 5 para deixar a pessoa que foi pegada no andar 3 (*dAndares* = 5 e *dParadas* = 3)

Ao fim da simulação, considerando a Equação (4.5), e as variáveis *dAndares* = 5 e *dParadas* = 3, E1 possui distância heurística de $d = 11$.

4.6.2 Elevador 2

E2 move 1 andar (do andar 2 para o andar 3) e finaliza a primeira etapa, chegando ao andar inicial do chamado (*dAndares* = 1 e *dParadas* = 0).

A segunda etapa sempre soma mais 1 a "*dParadas*", que representa a abertura e o fechamento das portas no andar 3 para que a pessoa possa entrar no elevador (*dAndares* = 1 e *dParadas* = 1).

Na terceira etapa, considerando que o sentido do chamado é subindo, E2 considera que o andar de destino do novo chamado é o andar 5 (pior caso) e agora existem dois andares para ir, que é o andar 4 e o andar 5. Diferente de E1, ir ao andar 4 e ao andar 5 possui a mesma direção diminuindo assim a distância heurística. Assim, são simulados os seguintes passos com E2:

- Se desloca do andar 3 para o andar 4 (*dAndares* = 2 e *dParadas* = 1)
- Realiza uma parada para pegar alguém no andar 4 (*dAndares* = 2 e *dParadas* = 2)
- Se desloca do andar 4 para o andar 5 (*dAndares* = 3 e *dParadas* = 2)
- Realiza uma parada no andar 5 para deixar a pessoa que foi pegada no andar 3 (*dAndares* = 3 e *dParadas* = 3)

Ao fim da simulação, considerando a Equação (4.5), e as variáveis *dAndares* = 3 e *dParadas* = 3, E2 possui distância heurística de $d = 9$.

4.6.3 Elevador 3

E3 move 2 andares (do andar 1 para o andar 3) e finaliza a primeira etapa, chegando ao andar inicial do chamado (*dAndares* = 2 e *dParadas* = 0).

A segunda etapa sempre soma mais 1 a "*dParadas*", que representa a abertura e o fechamento das portas no andar 3 para que a pessoa possa entrar no elevador (*dAndares* = 2 e *dParadas* = 1).

Na terceira etapa, considerando que o sentido do chamado é subindo, E3 considera que o andar de destino do novo chamado é o andar 5 (pior caso). Caso o chamado possuísse a direção "descendo", E3 iria considerar que o andar de destino seria o andar 1. É relevante ressaltar que pelo fato de E3 não possui nenhuma outra tarefa de pegar ou deixar pessoas em outros andares a distância heurística será bastante reduzida. Assim, são simulados os seguintes passos com E3:

- Se desloca do andar 3 para o andar 5 (*dAndares* = 4 e *dParadas* = 1)

- Realiza uma parada no andar 5 para deixar a pessoa que foi pegada no andar 3 (*dAndares* = 4 e *dParadas* = 2)

Ao fim da simulação, considerando a Equação (4.5), e as variáveis *dAndares* = 4 e *dParadas* = 2, E3 possui distância heurística de $d = 8$.

4.6.4 Elevador 4

E4 possui tarefas nos andares 4, 3, 2 e 1. Isso irá aumentar bastante sua distância heurística considerando o número de paradas que será necessário até que E4 consiga finalizar completamente o novo chamado. Na primeira etapa o seguinte é simulado com E4:

- Se desloca do andar 5 para o andar 4 (*dAndares* = 1 e *dParadas* = 0)
- Realiza uma parada no andar 4 (*dAndares* = 1 e *dParadas* = 1)
- Se desloca do andar 4 para o andar 3 (*dAndares* = 2 e *dParadas* = 1)

A segunda etapa sempre soma mais 1 a "*dParadas*", que representa a abertura e o fechamento das portas no andar 3 para que a pessoa possa entrar no elevador. No caso específico de E4, a parada também foi feita para deixar pessoas no andar 3 (*dAndares* = 2 e *dParadas* = 2).

Na terceira etapa, considerando que o sentido do chamado é subindo, E4 considera que o andar de destino do novo chamado é o andar 5 (pior caso). Entretanto E4 precisa realizar paradas tanto no andar 2 como no andar 1 e a prioridade são de ambos andares pois os mesmo já estavam na lista de tarefas de E4 antes do novo chamado no andar 3. Assim, são simulados os seguintes passos com E4:

- Se desloca do andar 3 para o andar 2 (*dAndares* = 3 e *dParadas* = 2)
- Realiza uma parada no andar 2 (*dAndares* = 3 e *dParadas* = 3)
- Se desloca do andar 2 para o andar 1 (*dAndares* = 4 e *dParadas* = 3)
- Realiza uma parada no andar 1 (*dAndares* = 4 e *dParadas* = 4)
- Se desloca do andar 1 para o andar 5 (*dAndares* = 8 e *dParadas* = 4)
- Realiza uma parada no andar 5 finalizando a simulação do novo chamado (*dAndares* = 8 e *dParadas* = 5)

Ao fim da simulação, considerando a Equação (4.5), e as variáveis *dAndares* = 8 e *dParadas* = 5, E4 possui distância heurística de $d = 18$.

4.6.5 Elevador 5

E5 possui uma tarefa no andar 2. Na primeira etapa o seguinte é simulado com E5:

- Se desloca do andar 1 para o andar 2 (*dAndares* = 1 e *dParadas* = 0)
- Realiza uma parada no andar 2 (*dAndares* = 1 e *dParadas* = 1)
- Se desloca do andar 2 para o andar 3 (*dAndares* = 2 e *dParadas* = 1)

A segunda etapa sempre soma mais 1 a "*dParadas*", que representa a abertura e o fechamento das portas no andar 3 para que a pessoa possa entrar no elevador (*dAndares* = 2 e *dParadas* = 2).

Na terceira etapa, considerando que o sentido do chamado é subindo, E5 considera que o andar de destino do novo chamado é o andar 5 (pior caso). Assim, são simulados os seguintes passos com E5:

- Se desloca do andar 3 para o andar 5 (*dAndares* = 4 e *dParadas* = 2)
- Realiza uma parada no andar 5 finalizando a simulação do novo chamado (*dAndares* = 4 e *dParadas* = 3)

Ao fim da simulação, considerando a Equação (4.5), e as variáveis *dAndares* = 4 e *dParadas* = 3, E5 possui distância heurística de $d = 10$.

A Figura 4 mostra o acúmulo das variáveis ao longo das etapas para todos os elevadores. Finalmente, depois de calcular as distâncias pode-se notar que a melhor possibilidade é que E3 decida atender o chamado, enquanto os outro deveriam ignorar o chamado.

A distância heurística de D, utilizada na Equação 4.1, reduz drasticamente o espaço de estados possíveis na modelagem deste problema e isso justifica abordar o problema de despacho de elevadores com sistemas multiagente e aprendizagem por reforço. A Equação 4.1 funciona satisfatoriamente porque, dado que θ_0 e θ_1 assuma valores positivos, se D e A são iguais, então o valor de $Q(s,a)$ será positivo. Entretanto, se D e A são diferentes, significa que o agente, ou resolveu atender um chamado não sendo o mais próximo, ou decidiu ignorar um chamado sendo o mais próximo. Como D e A são diferentes, $\theta_1 DA$ resultará em um valor negativo, tornando $Q(s,a)$ no mínimo "desconfortável" ao agente, já que este escolhe ações com o maior valor ($Q(s,a)$) em 80% do tempo.

4.7 Função de recompensa

Para completar a heurística proposta, se faz necessário descrever a função de recompensa. Ela funciona da seguinte forma:

- A função recompensa recebe como entrada o estado s, a ação a e o agente que a invocou.

- D é calculado a partir do agente que invocou a função recompensa.
- Se D e A são iguais, significa que o agente ou atendeu o chamado sendo o mais próximo, ou ignorou o chamado não sendo o mais próximo, sendo recompensado com 1.
- Se D e A são diferentes, significa que o agente ou resolveu atender um chamado não sendo o mais próximo, ou decidiu ignorar um chamado sendo o mais próximo, sendo punido com -1.

A função recompensa pode ser definida por:

$$f(A) = \begin{cases} 1 & \text{Se } DA > 0, \\ -1 & \text{Se } DA < 0 \end{cases} \tag{4.6}$$

onde D é calculado considerando o agente que chamou a função.

A função recompensa foi programada dessa forma porque observou-se que os agentes só agiam quando havia novos chamados, de tal forma que tornou-se difícil recompensar os agentes futuramente, e principalmente saber a qual par estado-ação deveria ser associado a nova recompensa. Sendo assim, foi utilizada uma recompensa instantânea para as ações corretas.

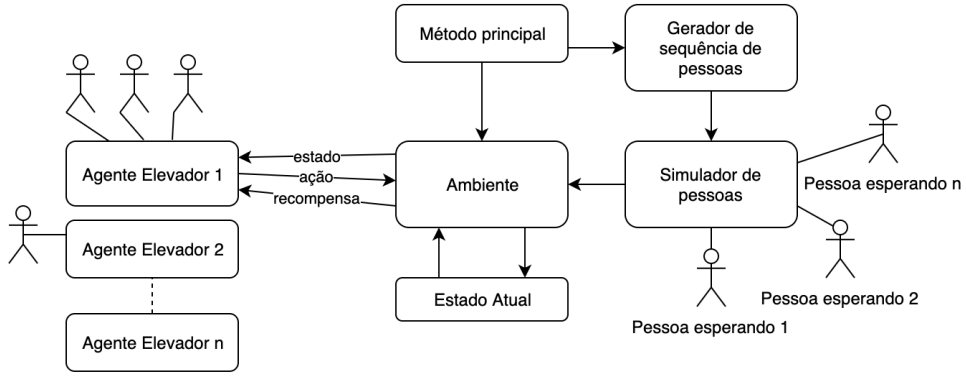
Com isso, todos os componentes relevantes da solução heurística proposta foram definidos. Ressalta-se que todas as etapas da heurística foram construídas sobre suposições razoáveis. Por exemplo, supor que em todos os movimentos ascendentes o elevador irá até o último piso e que nos movimentos descendentes ele irá até o térreo não viola os princípios mais básicos dos algoritmos clássicos para elevadores. Assim também acontece quando a heurística privilegia um chamado ser atendido pelo elevador mais próximo.

4.8 Procedimento de simulação

A Figura 5 apresenta a estrutura dos módulos implementados para validar este trabalho e a forma como eles se relacionam.

Para auxiliar na simulação do sistema multiagente foi utilizado o framework MESA (KAZIL; MASAD, 2018), que é um ambiente de simulação de SMA orientado a tempo, ou seja, a simulação avança em tic cuja duração pode ser definido pelo usuário. Para gerar uma simulação no MESA é necessário que seja configurando um ambiente e que este possua um ou mais agentes. A principal função do MESA neste trabalho é controlar a unidade de tempo, que no caso desta implementação está em segundos.

Figura 5 – Estrutura dos módulos de simulação



Fonte – Elaborado pelo autor

- O **Método principal** possui a função de: iniciar a simulação; finalizar a simulação; controlar a passagem do tempo da simulação; e coletar dados necessários aos resultados.
- As **Pessoas** são objetos que transitam no SMA de acordo com sua localização física, podendo estar esperando um elevador ou dentro de um elevador.
- O **Gerador de sequência de pessoas** retorna uma lista indicando quantas pessoas irão chegar a cada segundo.
- O **Simulador de pessoas** alerta outros módulos da chegada de novas pessoas e inicializam um andar de chegada e um andar de destino para as respectivas pessoas, de acordo com os tráfegos, que serão explicados no capítulo 5.
- O **Agente-elevador** possui toda a implementação referente ao aprendizado do mesmo, como foi explicado neste capítulo.
- O **Ambiente** modifica o estado atual, altera as posições dos elevadores à medida que os segundos passam, além de abrir e fechar portas. De fato, o ambiente simula toda a dinâmica do sistema multiagente que não envolve a tomada de decisão entre atender ou ignorar o novo chamado. O ambiente fornece também recompensas aos Agentes-elevador.

O **Método principal** inicialmente solicita do **Gerador de sequência de pessoas** uma lista com a quantidade de pessoas que irá chegar a cada segundo. Essa lista é criada ou a partir de um processo de Poisson ou a partir de um padrão de tráfego personalizado (subida ao pico, descida do pico, combinação de ambos). Com esta, o **Simulador de pessoas** é instanciado, e o mesmo passa a se comunicar com o **Ambiente** notificando as pessoas que estão esperando para utilizar o sistema, junto com o andar onde cada uma está e a direção para onde desejam se

locomover (subir ou descer). À medida que as pessoas chegam no **Ambiente**, este informa aos **Agentes-elevador**, que decidem atender ou ignorar ao chamado baseado no que foi explicado nas seções anteriores. As ações dos **Agentes-elevador**, o passar dos segundos e a chegada de pessoas (ou o apertar de botões) alteram o **Estado atual** através da comunicação com o ambiente.

O controle do tempo se mantém no **Método principal**, que através de um laço avança segundo a segundo dando andamento na simulação. A simulação acaba quando: a lista com a quantidade de pessoas que irá chegar a cada segundo é totalmente esvaziada; e todas as pessoas instanciadas no sistema são deixadas nos seus respectivos andares de destino. Cada pessoa possui:

- **Id** para controle do MESA
- **Andar atual**
- **Andar de destino**
- **Tempo de espera** para contar o tempo que a pessoa passa esperando o elevador chegar
- **Tempo de jornada** para contar o tempo que a pessoa passa dentro de um elevador até chegar ao seu destino
- **nElevador** que é uma flag de controle caso todos os elevadores decidam ignorar o chamado

Uma pessoa ou está contida no **Simulador de pessoas** esperando um elevador chegar ou está contida em um **Agente-elevador** esperando seu andar de destino.

Algoritmo 3: Método principal de simulação multiagente métodoPrincipal(λ, s, t, N, M)

Entrada: coeficiente Poisson λ; tempo da sequência s; padrão de tráfego t; número de elevadores N; e número de andares M
Saída: Uma lista com todas as pessoas que utilizaram o sistema multiagente
início
    gsp = GeradorDeSequênciaDePessoas(λ, s, t);
    amb = Ambiente(N, M, gsp);
    pessoas = [];
    while *gsp.tamanho > 0 ou amb.simuladorDePessoas.pessoas.tamanho > 0 ou amb.elevadores.pessoas.tamanho > 0* **do**
        pessoaQueUtilizaramOSistema = *amb*.próximoSegundo();
        pessoas.concatenar(*pessoaQueUtilizaramOSistema*);
    end
    retorna *pessoas*
fim

O Algoritmo 3 mostra o pseudocódigo do método principal controlando assim todos

os outros módulos do sistema, sendo que o mesmo foi implicitamente explicado no parágrafo anterior. No fim da simulação, a lista de pessoas é utilizada para a construção dos resultados através do tempo de espera e tempo de jornada das pessoas. Toda a implementação foi codificada em Python 3.7.

5 RESULTADOS E EXPERIMENTOS

Este capítulo apresenta os experimentos realizados para avaliar o desempenho da solução proposta. Ela contém experimentos próprios e uma subseção de comparação com outro trabalho publicado. O fluxo de chegada de pessoas é simulado por um processo de Poisson. O primeiro experimento compara a solução proposta com um algoritmo clássico típico descrito a seguir. A cada pessoa *p* que chega para utilizar o SMA, com 70% de chances *p* aperta todos os botões de mesmo sentido em todos os N elevadores dispostos. Com 30% de chances *p* aperta um único botão. Se todos os botões de mesmo sentido forem apertados, então todos os elevadores "atendem o chamado". Se um único botão for apertado, então apenas o elevador referente ao botão "atende o chamado". O Algoritmo 4 mostra um pseudocódigo do que foi descrito.

Algoritmo 4: Algoritmo clássico para o controle de grupos de elevadores

```
Entrada: uma lista de pessoas P; uma lista de elevadores E
Saída: uma lista de pessoas P sem a primeira pessoa da lista
início
    pes = P.primeiro;
    P = P.removaPrimeiro();
    n = sorteie(0,100);
    se n ≤ 30 então
        ele = sorteieUm(E);
        ele.novoChamado(pes);
    fim
    senão
        para cada ele ∈ E faça
            ele.novoChamado(pes)
        fim
    fim
    retorna P
fim
```

Além do Algoritmo 4, o algoritmo baseline também foi utilizado para atribuir maior confiabilidade a validação da solução proposta (SMA-RL). O algoritmo baseline será explicado a seguir. A cada pessoa *p* que chega para utilizar os elevadores e aperta o botão referente ao destino (sobe ou desce) o elevador cujo andar estiver mais próximo geograficamente é escolhido como o que irá "atender o chamado". O Algoritmo 5 mostra um pseudocódigo do que foi descrito.

Para avaliar o TMS, existem 4 atributos que serão variados ao longo deste capítulo: número de elevadores, número de andares, fluxo de chegada de pessoas (λ) e o padrão de tráfego. Existem 3 padrões de tráfego que são adotados: ordinário, descida do pico e subida ao pico. No

Algoritmo 5: Algoritmo baseline para o controle de grupos de elevadores

```
Entrada: uma lista de pessoas P; uma lista de elevadores E
Saída: uma lista de pessoas P sem a primeira pessoa da lista
início
    pes = P.primeiro;
    P = P.removaPrimeiro();
    elevadorEscolhido = E.primeiro;
    distanciaMenor = modulo(pes.andarAtual - elevadorEscolhido.andarAtual);
    para cada ele ∈ E faça
        se modulo(pes.andarAtual - ele.andarAtual) < distanciaMenor então
            elevadorEscolhido = ele;
            distanciaMenor = modulo(pes.andarAtual - ele.andarAtual);
        fim
    fim
    elevadorEscolhido.novoChamado(pes);
    retorna P
fim
```

ordinário, cada pessoa que chega para utilizar o sistema tem o seu andar de chegada e andar de saída sorteado, com a restrição óbvia do andar de chegada ser diferente do andar de saída. Esse padrão acontece em shoppings, por exemplo. Na descida do pico, cada pessoa que chega para utilizar o sistema tem o seu andar de chegada sorteado entre o segundo e último andar e o seu andar de saída é necessariamente o primeiro. Simula, por exemplo, o fim de expediente de um prédio comercial. Na subida ao pico, o andar de chegada é necessariamente o primeiro, e o andar de saída é sorteado entre o segundo e último andar. Simula, por exemplo, o início de expediente de um prédio comercial. A Tabela 2 mostra os parâmetros de ambiente referentes aos elevadores e a construção.

Tabela 2 – Parâmetros dos elevadores e da construção

Parâmetros dos elevadores	
Tempo de abertura de portas	2 segundos
Tempo de fechamento de portas	2 segundos
Tempo médio de transferência de passageiros	1 segundo
Velocidade (m/s)	1,5
Aceleração (m/s^2)	1,5
Freio (m/s^2)	1,5
Parâmetro do edifício	
Altura dos andares	3 metros

Fonte – Elaborado pelo autor

Cada simulação correu por uma hora (3600 segundos). A Tabela 3 mostra todas as variáveis mais relevantes adotadas na implementação da solução.

Tabela 3 – Variáveis mais relevantes utilizadas na implementação do SMA

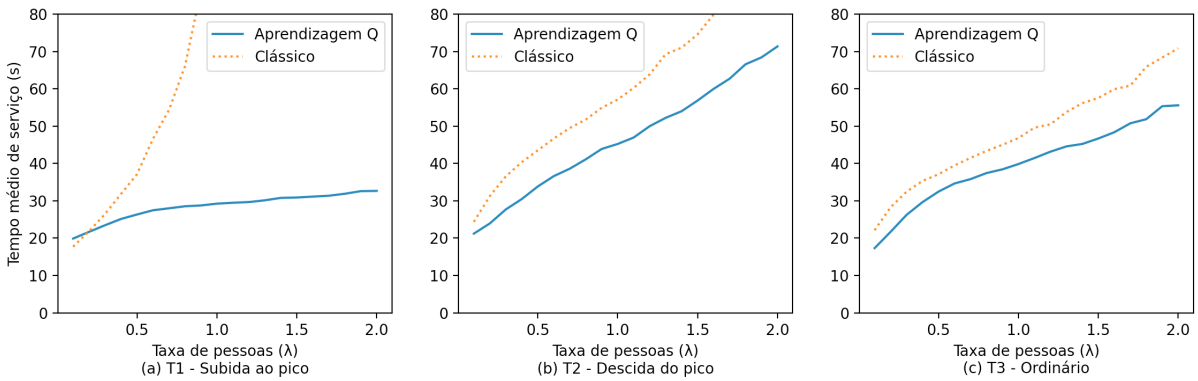
Variável	Valor
Taxa de aprendizado da função de valor estado-ação (α)	0,9
Taxa de aprendizado da atualização dos pesos	0,4
θ_0 inicial	0
θ_1 inicial	1
Desconto da função de valor estado ação (γ)	0,3
Política de exploração (ε)	0,2
Duração do processo de Poisson	3600 segs

Fonte – Elaborado pelo autor

5.1 Taxa de pessoas e Padrão de tráfego

Como primeiro experimento, λ foi variado de 0,1 a 2 em intervalos de 0,1. Para cada λ foram feitas 10 simulações resultando em 10 TMSs, onde o TMS associado ao λ é a média dos 10 TMSs. Assim, foi gerada uma curva que avalia o TMS para cada λ. Foram geradas duas curvas: uma para solução proposta e outra para o algoritmo clássico com o intuito de validar um desempenho aceitável para o SMA proposto. Neste experimento o SMA possui 3 elevadores e 6 andares. Este experimento foi realizado 3 vezes variando os três padrões de tráfego apresentados.

Figura 6 – TMS dos três padrões de tráfego com λ variando de 0.1 a 2 para o algoritmo clássico e a solução proposta, para 3 elevadores e 6 andares



Fonte – Elaborado pelo autor

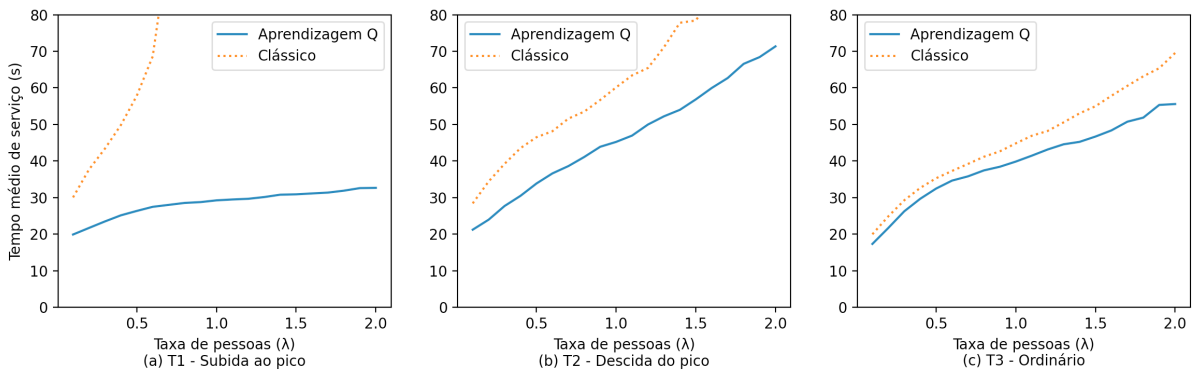
A Figura 6 mostra o resultado do que acabou de ser explicado. Em 6(c), com o

tráfego ordinário, o SMA fornece um TMS menor para todos os λs. Em 6(b), com o tráfego descida do pico, o mesmo acontece. É natural que o TMS suba à medida que λ suba, pois com mais pessoas, mas os elevadores ficam congestionados. O objetivo, entretanto, é que suba o mínimo possível. Considerando 6(c), para $\lambda = 2$, tem-se que o TMS do SMA é aproximadamente 55 segundos, que é um valor bastante razoável para um sistema que recebe 2 pessoas por segundo. Em 6(b), para $\lambda = 2$, o TMS do SMA é aproximadamente 72 segundos. Os autores acreditam que os TMSs são relativamente altos porque todos os elevadores precisam sempre **deixar** as pessoas em um mesmo andar específico (andar 1). Em 6(a), com exceção dos menores valores de λ, o SMA proposto domina drasticamente o algoritmo clássico. Além disso, no padrão subida ao pico obtém-se os melhores TMSs deste experimento. Os autores acreditam que os TMSs são relativamente baixos porque todos os elevadores precisam sempre **pegar** as pessoas em um mesmo andar específico (andar 1). É válido perceber que, relativamente, para:

- chegada específica e saída aleatória resulta em resultados bons - 6(a)
- chegada aleatória e saída aleatória resulta em resultados medianos - 6(c)
- chegada aleatória e saída específica resulta em resultados ruins - 6(b)

Ainda sim, isso não é o suficiente para afirmar algo, já que o andar específico em questão (andar 1) está em uma extremidade, podendo ser um fator impactante nas curvas da Figura 6(a) e 6(b). Contudo, o último ponto enfatizado sugere que existe uma relação entre o SMA, a distância heurística de D e o tráfego de subida ao pico pois mesmo com o $\lambda = 2$ o TMS se mantém aproximadamente 30 segundos, que é um tempo bastante satisfatório. Com exceção dos menores valores de λ em 6(a), a solução proposta apresenta uma estratégia dominante em todos os casos frente ao algoritmo clássico.

Figura 7 – TMS dos três padrões de tráfego com λ variando de 0.1 a 2 para o algoritmo baseline e a solução proposta, para 3 elevadores e 6 andares



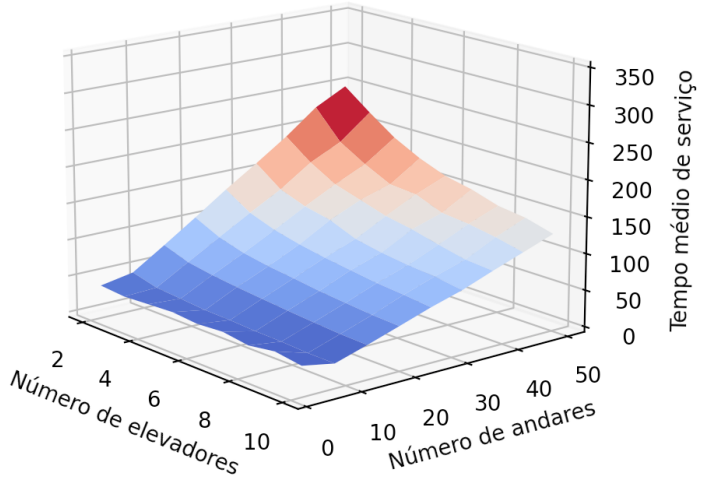
Fonte – Elaborado pelo autor

A Figura 7 mostra os resultados do mesmo procedimento explicado no início desta seção. Entretanto, na Figura 7 são comparados o algoritmo baseline (Algoritmo 5) e a solução proposta (SMA-RL). Pode-se notar que a solução proposta apresenta uma estratégia dominante frente ao algoritmo baseline nos 3 padrões de tráfego, sem exceções. Além disso, considerando a Figura 6(a) e a Figura 7(a), tanto o algoritmo clássico como o algoritmo baseline apresentam resultados ruins no padrão de tráfego T1 (subida ao pico) ao passo que a solução proposta (SMA-RL) apresenta seus melhores resultados, ou seja, um tempo de serviço bastante satisfatório mesmo com o aumento da taxa de pessoas.

5.2 Número de elevadores e número de andares

O objetivo deste experimento é avaliar se o SMA proposto resulta em valores de TMSs razoáveis em condições extremas. Neste experimento, fixaram-se $\lambda = 1$ e tráfego ordinário. Os N elevadores variam de 2 a 10 e os M andares variam de 2 a 50. Para cada combinação de N e M foi feita uma simulação e obtido 1 TMS. Assim, foi construído um gráfico 3D com esses números visando avaliar o comportamento do SMA. A Figura 8 mostra o que foi descrito. O ponto a se enfatizar é que com o SMA deste trabalho, utilizar muitos elevadores com poucos andares não impacta no TMS. Para perceber isso, basta observar na Figura 8, que mesmo variando os elevadores, para andares abaixo de 10, o TMS se mantém aproximadamente o mesmo.

Figura 8 – Tempo médio de serviço obtido com tráfego ordinário e $\lambda = 1$ variando o número de elevadores e de andares para a solução proposta (SMA-RL)

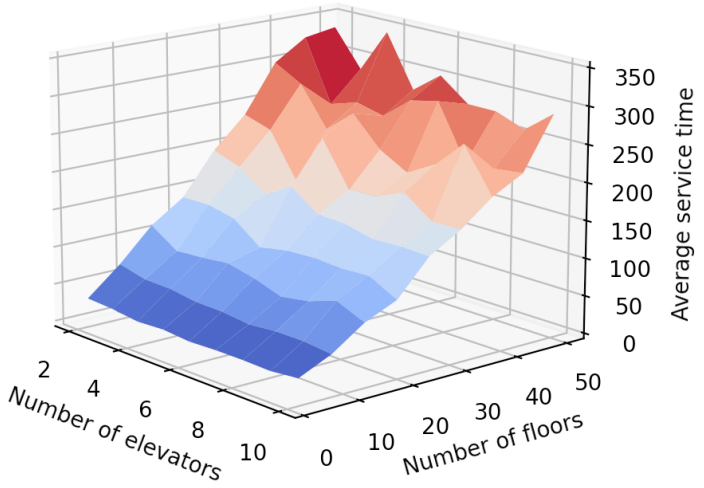


Fonte – Elaborado pelo autor

A Figura 9 mostra, sob as mesmas circunstâncias descritas nesta seção, o resultado

do tempo médio de serviço obtido para o algoritmo clássico.

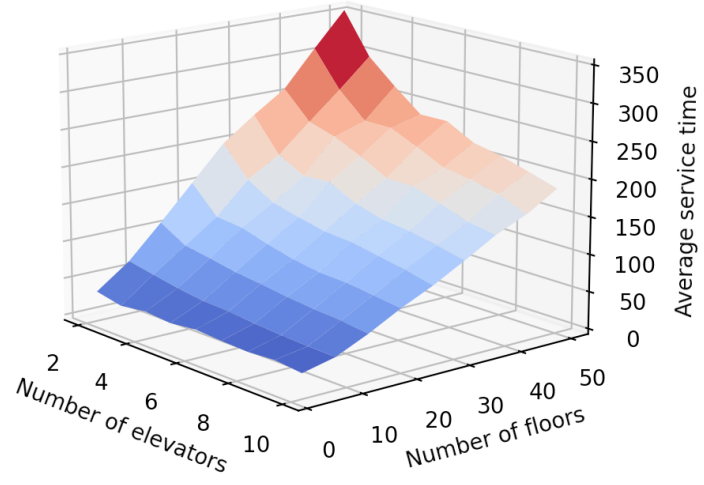
Figura 9 – Tempo médio de serviço obtido com tráfego ordinário e $\lambda = 1$ variando o número de elevadores e de andares para o algoritmo clássico



Fonte – Elaborado pelo autor

A Figura 10 mostra, sob as mesmas circunstâncias descritas nesta seção, o resultado do tempo médio de serviço obtido para o algoritmo baseline.

Figura 10 – Tempo médio de serviço obtido com tráfego ordinário e $\lambda = 1$ variando o número de elevadores e de andares para o algoritmo baseline



Fonte – Elaborado pelo autor

5.3 Comparação com outras soluções

O objetivo deste experimento é avaliar se o SMA proposto possui desempenho competitivo com outras soluções anteriormente publicadas na literatura. São elas: RL (Ikuta; Takahashi; Inaba, 2013), baseada em redes neurais recorrentes e aprendizagem por reforço, a mesma comentada na Seção 3.3; GA (XUE, 2002) baseada em Algoritmos Genéticos; e SZ baseada em Zoneamento Estático. A Tabela 4 mostra o número de pessoas que chegam como função do tempo (em minutos) para os três padrões de tráfegos adotados. Esse trabalho foi escolhido para comparação por dois motivos: porque seus resultados de desempenho são comparáveis a outros publicados e porque os autores fornecem dados suficientes para que a reprodução dos experimentos possa ser realizada. Isso nem sempre acontece com outros trabalhos.

Tabela 4 – Fluxos de chegada de pessoas para os três tipos de tráfego

Tempo(min)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TF1: Subida ao pico															
Pessoas	18	11	15	8	8	34	46	40	34	46	9	10	6	6	9
TF2: Descida do pico															
Pessoas	17	17	13	13	10	34	31	26	41	38	13	12	14	8	12
TF3: Descida do pico e leve subida ao pico															
P. subindo	0	1	2	2	0	2	2	0	2	0	1	2	1	0	0
P. descendo	7	10	7	9	12	10	8	8	10	18	11	6	12	2	15

Fonte – Elaborado pelo autor

A Tabela 5 mostra o Tempo Médio de Espera (TME) em segundos que significa o tempo que uma pessoa passa em média esperando o elevador chegar. Além disso, mostra Tempo Médio de Jornada (TMJ) em segundos que representa o tempo médio de uma pessoa dentro de um elevador. Para facilitar a comparação, mostra também o Tempo Médio de Serviço (TMS), que é simplesmente a soma de TME mais TMJ. E por último mostra a porcentagem de tempo que o sistema de elevadores fica aglomerado, onde é considerado aglomerado se ao menos uma pessoa estiver esperando para adentrar em algum elevador. Este teste utiliza 4 elevadores e 16 andares para ser executado. Se tratando de TF1, a solução proposta apresentou melhores resultado que quase todas as outras 3 soluções, perdendo apenas para o tempo médio de jornada de SZ. Entretanto, caso seja somado o tempos médio de espera e jornada, SMA-RL mostra melhor desempenho além de menor porcentagem de aglomeração. Isso somado aos

resultados da Seção 5.1 confirma que SMA-RL apresenta resultados especialmente satisfatórios para o tráfego de subida ao pico. Assim, para TF1, a solução proposta aprensenta uma estratégia dominante em comparação com as outras 3 previamente publicadas na literatura. Em TF2, o tempo médio de espera continua o melhor, enquanto nas outras duas medidas apresenta resultados competitivos. Caso ambos os tempo médios sejam somados, SMA-RL mostra um tempo inferior apenas de SZ, por 0,57 segundos de diferença. Assim, para TF2, SMA-RL apresenta uma estrégia dominada por SZ, e dominante em relação a GA e RL. Em T3, SMA-RL mostra seu pior resultado apresentando uma estratégia dominada por todos os outros 3 algoritmos. Esse resultado ruim acontece provavelmente por que o tráfego de pessoas se concentra no andar 1, ou seja, as pessoas subindo iniciam no andar 1 enquanto as pessoas descendo finalizam no andar 1. Essa concentração de tráfego dificulta a otimização do algoritmo, considerando o posicionamento atual de cada elevador, ao passo que os outros algoritmos da literatura não enfrentam essa dificuldade.

Tabela 5 – Tempo médio de espera(s), tempo médio de jornada(s) e tempo de aglomeração considerando outras 3 soluções para 4 elevadores e 16 andares

		Tempo médio de espera (s)	**Tempo médio de jornada (s)**	**Tempo médio de serviço (s)**	**Média de aglomeração (%)**
TF1	RL	50,19	45,82	96,01	68,09
	GA	75,16	53,89	129,05	89,13
	SZ	31,78	**38,26**	70,04	68,87
	SMA-RL	**9,33**	41,08	**50,41**	**52,12**
TF2	RL	33,48	40,56	74,04	68,91
	GA	64,81	52,85	117,66	88,27
	SZ	25,56	**39,95**	**65,51**	**66,20**
	SMA-RL	**18,94**	47,14	66,08	88,12
TF3	RL	11,95	**15,38**	27,33	**11,47**
	GA	**7,35**	17,53	**24,88**	15,60
	SZ	20,20	21,81	42,01	26,53
	SMA-RL	15,49	33,75	49,24	69,90

Fonte – Elaborado pelo autor

5.4 Análise do comportamento do algoritmo

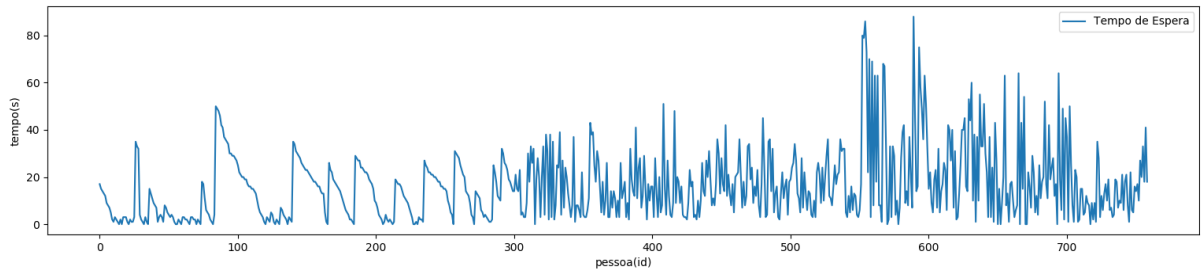
Esta seção se destina a analisar o comportamento dos parâmetros do algoritmo ao longo de uma simulação para fazer uma verificação empírica da estabilidade da solução. As seguintes variáveis serão observadas: θ_0; θ_1; TME; e TMJ. A simulação para observar essas variáveis foi feita da seguinte forma: os 3 padrões de tráfegos da Tabela 4 foram enfileirados totalizando 45 minutos de simulação utilizando a sequência TF1, TF3 e TF2. Assim, inicialmente o sistema lida apenas com pessoas subindo, em seguida um misto de pessoas descendo e subindo e por fim apenas pessoas descendo. Um fator relevante a se constar, é que θ_0 e θ_1 não são reinicializados ao longo dos 45 minutos de simulação, buscando observar e validar a versatilidade da solução com a variação dos padrões de tráfegos. Considerando toda a simulação aproximadamente 800 pessoas utilizaram o grupo de elevadores. Nesta simulação foram utilizados 4 elevadores e 16 andares.

5.4.1 Tempo de Espera e Tempo de Jornada

A Figuras 11 e 12 mostram o tempo de espera e o tempo de jornada de cada pessoa à medida que esta é deixada em seu andar de destino. É importante perceber que as figuras 11 e 12 apresentam resultados muito parecidos com a Figura 6. Nos primeiros 15 minutos (subida ao pico) é onde o algoritmo mostra os menores TE e TJ, assim como na Figura 6(a), com diferença de que esta apresenta TMS que é a soma de TMJ mais TME. Em seguida, se for comparado TF3 da Tabela 4 com o tráfego ordinário da Figura 6(c), o algoritmo apresenta resultados inferiores a TF1, com TJ relativamente igual e TE com tempo superiores. E por último, nos últimos 15 minutos, é onde a solução apresenta os piores resultados (descida do pico), assim como na Figura 6(b). À medida que os agentes-elevador aprendem, o sistema necessariamente se torna mais eficiente, entretanto fatores como o fluxo de pessoas congestionando o sistema e a variação dos padrões de tráfego, que são fatores comuns a um teste real, dificultam essa percepção ao ponto de que essa eficiência não é percebida nitidamente nas figuras 11 e 12.

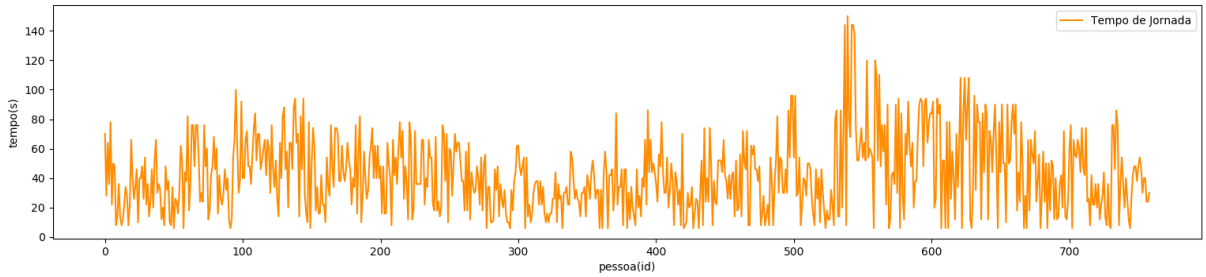
Como forma alternativa de visualização, foi produzido um histograma que visa estimar o tempo de espera, tempo de jornada e tempo de serviço que cada pessoa gastou para utilizar o sistema de elevadores considerando a mesma simulação desta seção. A Figura 13 enfatiza o tempo de espera. A Figura 13(a) fornece uma noção do número de pessoas que utilizou o sistema de elevadores no eixo y e quantos segundos foram gastos aproximadamente com o tempo de espera, ou seja, o tempo que a mesma espera para o elevador chegar ao andar dela e

Figura 11 – Tempos de espera de cada pessoa para TF1, TF3 e TF2



Fonte – Elaborado pelo autor

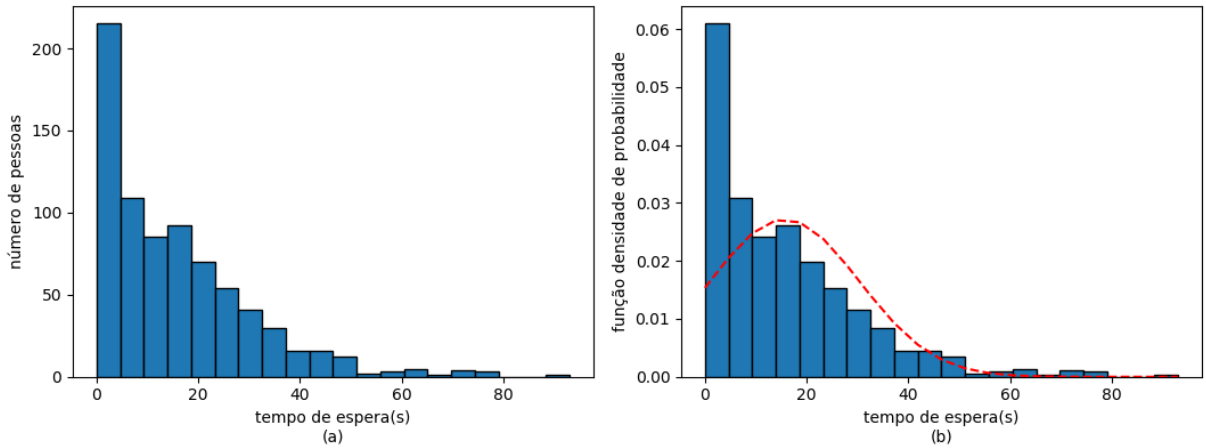
Figura 12 – Tempos de jornada de cada pessoa para TF1, TF3 e TF2



Fonte – Elaborado pelo autor

abrir a porta para que ela possa entrar. A Figura 13(b) fornece uma noção de probabilidade onde o somatório das barras compõem a função massa de probabilidade discretizada pelos intervalos de tempo no eixo x e a linha vermelha tracejada indica a função densidade de probabilidade onde o eixo y contém a probabilidade do *tempo de espera correspondente no eixo x* acontecer. A função densidade de probabilidade da Figura 13(b) possui média de 15,88 segundos e desvio padrão de 14,66 segundos.

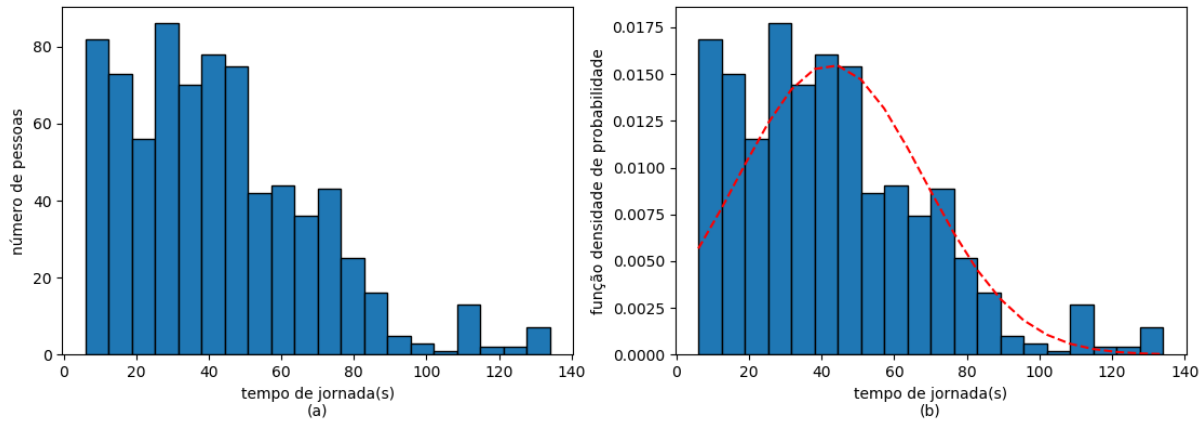
Figura 13 – Histograma do tempo de espera em segundos para TF1, TF3 e TF2 com média de 15,68 segundos e desvio padrão de 14,66 segundos



Fonte – Elaborado pelo autor

A Figura 14 enfatiza o tempo de jornada. A Figura 14(a) fornece uma noção do número de pessoas que utilizou o sistema de elevadores no eixo y e quantos segundos foram gastos aproximadamente com o tempo de jornada, ou seja, o tempo que a mesma utiliza dentro de um elevador enquanto espera seu andar de destino chegar. A Figura 14(b) fornece uma noção de probabilidade onde o somatório das barras compõem a função massa de probabilidade discretizada pelos intervalos de tempo no eixo x e a linha vermelha tracejada indica a função densidade de probabilidade onde o eixo y contém a probabilidade do *tempo de jornada correspondente no eixo x* acontecer. A função densidade de probabilidade da Figura 14(b) possui média de 42,35 segundos e desvio padrão de 22,55 segundos.

Figura 14 – Histograma do tempo de jornada em segundos para TF1, TF3 e TF2 com média de 42,35 segundos e desvio padrão de 22,55 segundos

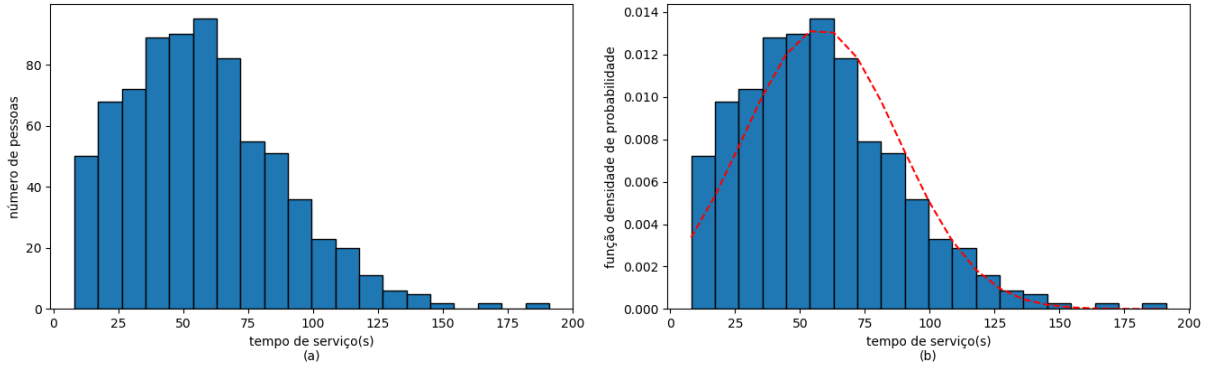


Fonte – Elaborado pelo autor

A Figura 15 enfatiza o tempo de serviço. A Figura 15(a) fornece uma noção do número de pessoas que utilizou o sistema de elevadores no eixo y e quantos segundos foram gastos aproximadamente com o tempo de serviço, ou seja, a soma do tempo de espera mais o tempo de jornada. A Figura 15(b) fornece uma noção de probabilidade onde o somatório das barras compõem a função massa de probabilidade discretizada pelos intervalos de tempo no eixo x e a linha vermelha tracejada indica a função densidade de probabilidade onde o eixo y contém a probabilidade do *tempo de serviço correspondente no eixo x* acontecer. A função densidade de probabilidade da Figura 15(b) possui média de 57,87 segundos e desvio padrão de 30,19 segundos.

Observando a média e o desvio padrão do tempo de serviço (Figura 15), pode-se chegar a conclusão que uma pessoa que chega para utilizar o sistema de elevadores, independente do tráfego (TF1, TF2 ou TF3), leva, em média, aproximadamente 60 segundos para chegar ao seu

Figura 15 – Histograma do tempo de serviço em segundos para TF1, TF3 e TF2 com média de 57,87 segundos e desvio padrão de 30,19 segundos



Fonte – Elaborado pelo autor

andar de destino, podendo variar de 30 segundos a 1,5 minutos, um valor bastante satisfatório. É válido ressaltar que esses números foram encontrados a partir do fluxo de pessoas considerado na Tabela 4. Assim, o tempo de serviço é diretamente proporcional ao fluxo de pessoas como demonstrado nas figuras 6 e 7.

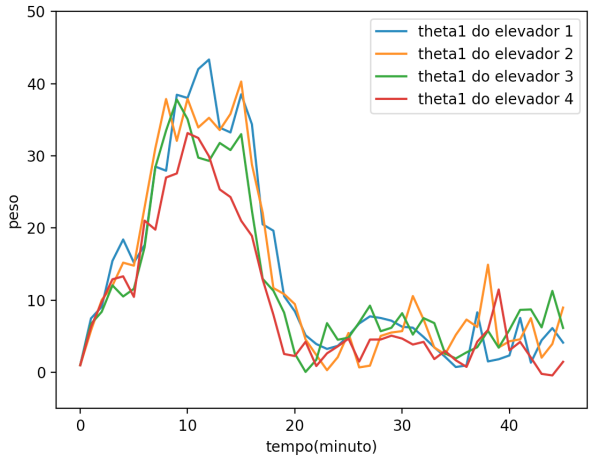
5.4.2 O aprendizado do algoritmo

De fato, para uma aplicação que trocou uma tabela Q por uma função de avaliação, o aprendizado é representado pela atualização dos pesos, neste caso θ_0 e θ_1. A Figura 17 mostra os 4 valores de θ_1 para cada um dos 4 elevadores ao longo de cada minuto da simulação. Ainda que o aprendizado seja independente em cada agente-elevador, a Figura 17 mostra que ele acontece de forma bastante similar, onde a partir dos 20 minutos todos os θ_1 convergem para uma região previsível: entre 0 e 10. O fato de cada θ_1 convergir de forma previsível valida o aprendizado dos agentes-elevador na solução proposta.

A Figura 17 mostra os 4 valores de θ_0 para cada um dos 4 elevadores ao longo de cada minuto da simulação. Na implementação do algoritmo, foi percebido que sem θ_0 eventualmente θ_1 não convergia e as simulações nunca acabavam, ou apresentavam resultados muito ruins. Com θ_0 incluído na função de valor estado-ação, tais falhas eventuais cessaram.

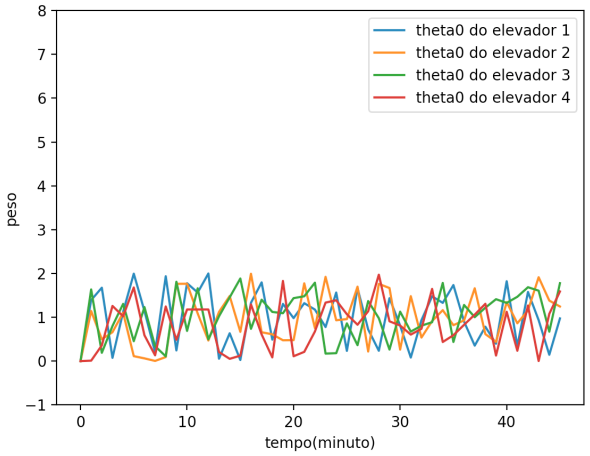
Nesta simulação o elevador 1 foi utilizado por 20.42% das pessoas, o elevador 2 foi utilizado por 28.19% das pessoas, o elevador 3 foi utilizado por 22.79% das pessoas, e o elevador 4 foi utilizado por 28.59% das pessoas, mostrando que uma quantidade aproximada de pessoas utiliza os elevadores de forma relativamente simétrica. É possível que em algumas simulações essas porcentagens não sejam necessariamente simétricas. Entretanto, após algumas simulações,

Figura 16 – Valores de θ_1 para cada elevador ao longo dos 45 minutos



Fonte – Elaborado pelo autor

Figura 17 – Valores de θ_0 para cada elevador ao longo dos 45 minutos



Fonte – Elaborado pelo autor

na maior parte das vezes nenhum elevador foi utilizado por mais de 30% das pessoas.

6 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho propôs uma solução para o problema do despacho de elevadores que utiliza sistemas multiagente e aprendizagem por reforço (aprendizagem Q). Além disso, foi mostrado que a solução fornece métricas aceitáveis para os diversos padrões de tráfego utilizados em avaliação de desempenho. A proposta consiste em uma função aproximação simples com uma heurística de aprendizagem associada. O objetivo foi encontrar um compromisso vantajoso entre crescimento do espaço de estados e velocidade de aprendizagem. Os resultados mostraram que o desempenho é comparável com o estado da arte. A solução se mostrou especialmente satisfatória quando o tráfego assumia o pico de subida, o que provavelmente está relacionada a distância heurística D.

6.1 Trabalhos futuros

Essa nova abordagem pode ser aperfeiçoada em vários pontos no futuro. Pretende-se investigar detalhadamente o comportamento do SMA proposto com outros padrões de tráfegos, além de rever a função de avaliação e a regra de atualização dos pesos visando obter melhores índices de desempenho. Outra tarefa a se fazer é implementar uma segunda heurística a partir de conhecimento prévio do domínio do controle de elevadores e utilizá-la na função de exploração com a finalidade de acelerar o aprendizado dos agentes-elevador.

REFERÊNCIAS

ARANIBAR, Dennis Barrios. **Aprendizado por Reforço com Valores de Influência em Sistemas Multi-Agente**. 2009. 125 f. Tese (Doutorado em Ciência da Computação) - Universidade Federal do Rio Grande do Norte, Rio Grande do Norte, 2009.

ARANIBAR, Dennis Barrios; ALSINA, Pablo J. **LEARNING STRATEGIES FOR COORDINATION OF MULTI ROBOT SYSTEMS**: a robot soccer application. A ROBOT SOCCER APPLICATION. 2016. Disponível em: https://pdfs.semanticscholar.org/510d/1bffae456a9dd0cb04e2408bf794df08f97a.pdf. Acesso em: 2 jul. 2019.

ARANIBAR, Dennis Barrios; GONCALVES, Luiz Marcos Garcia. Learning Coordination in Multi-Agent Systems Using Influence Value Reinforcement Learning. **Seventh International Conference On Intelligent Systems Design And Applications (Isda 2007)**, Kiev, p. 471-478, out. 2007.

BANERJEE, Dipyaman; SEN, Sandip. Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. **Autonomous Agents And Multi-Agent Systems**, [S.L.], v. 15, n. 1, p. 91-108, 30 abr. 2007.

BIANCHI, Reinaldo A. C.; MARTINS, Murilo F.; RIBEIRO, Carlos H. C.; COSTA, Anna H. R.. Heuristically-Accelerated Multiagent Reinforcement Learning. **Ieee Transactions On Cybernetics**, [S.L.], v. 44, n. 2, p. 252-265, fev. 2014.

BOWLING, Michael; VELOSO, Manuela. Multiagent learning using a variable learning rate. **Artificial Intelligence**, [S.L.], v. 136, n. 2, p. 215-250, abr. 2002.

BOWLING, Michael H.; VELOSO, Manuela M.. **An Analysis of Stochastic Game Theory for Multiagent Reinforcement Learning**. 2000. Disponível em: https://www.semanticscholar.org/paper/An-Analysis-of-Stochastic-Game-Theory-for-Learning-Bowling-Veloso/1355de09f6cc7ca8ec59f6f5e2f65acc3996b516. Acesso em: 2 jun. 2019.

BUSONIU, Lucian; BABUSKA, Robert; SCHUTTER, Bart de. A Comprehensive Survey of Multiagent Reinforcement Learning. **Ieee Transactions On Systems, Man, And Cybernetics, Part C (Applications And Reviews)**, [S.L.], v. 38, n. 2, p. 156-172, mar. 2008.

CAO, Liting; TIAN, Jingwen; ZHANG, Zhaoli. Elevator Group Control System Based on Information Fusion Technology. **2008 3Rd International Conference On Innovative Computing Information And Control**, [S.L.], p. 473-473, 2008.

CAO, Liting; ZHOU, Shiru; YANG, Shuo. Elevator Group Dynamic Dispatching System Based on Artificial Intelligent Theory. **2008 International Conference On Intelligent Computation Technology And Automation (Icicta)**, [S.L.], p. 183-186, out. 2008.

CARRASCOSA, C.; BAJO, J.; JULIAN, V.; CORCHADO, J.M.; BOTTI, V.. Hybrid multi-agent architecture as a real-time problem-solving model. **Expert Systems With Applications**, [S.L.], v. 34, n. 1, p. 2-17, jan. 2008.

CHALKIADAKIS, Georgios; BOUTILIER, Craig. Coordination in multiagent reinforcement learning. **Proceedings Of The Second International Joint Conference On Autonomous Agents And Multiagent Systems - Aamas '03**, [S.L.], p. 709-716, 2003.

CLAUS, Caroline; BOUTILIER, Craig. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In: CLAUS, Caroline; BOUTILIER, Craig. **Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence**. Madison: American Association For Artificial Intelligence, 1998. p. 746-752.

DAS, Rajarshi; KEPHART, Jeffrey; LEFURGY, Charles; TESAURO, Gerald; LEVINE, David; CHAN, Hoi. Autonomic multiagent management of power and performance in data centers. **Proceedings Of The 7Th International Joint Conference On Autonomous Agents And Multiagent Systems: Industrial Track.** Amsterdã, p. 107-114. 9 jan. 2008.

DIMITRIADIS, Socrates; MARIAS, Kostas; ORPHANOUDAKIS, Stelios C.. A multi-agent platform for content-based image retrieval. **Multimedia Tools And Applications**, [S.L.], v. 33, n. 1, p. 57-72, 7 fev. 2007.

FERNÁNDEZ, Fernando; GARCÍA, Javier; VELOSO, Manuela. Probabilistic Policy Reuse for inter-task transfer learning. **Robotics And Autonomous Systems**, [S.L.], v. 58, n. 7, p. 866-871, jul. 2010.

GUO, R.; WU, M.; PENG, J.; CAO, W.. New q learning algorithm for multi-agent systems. **Zidonghua Xuebao/acta Automatica Sinica.** Pequin, p. 367-372. 10 fev. 2007.

HOEN, Pieter Jan 'T; TUYLS, Karl; PANAIT, Liviu; LUKE, Sean; LAPOUTRÉ, J. A.. An Overview of Cooperative and Competitive Multiagent Learning. **Learning And Adaption In Multi-Agent Systems**, [S.L.], p. 1-46, 2006.

IBA, Hitoshi. Evolving Multiple Agents by Genetic Programming. In: IBA, Hitoshi. **Advances in Genetic Programming**: volume 3. Cambridge: Mit Press, 1999. p. 447-466.

IKEDA, Kokolo; SUZUKI, Hiromichi; KITA, Hajime; MARKON, Sandor. Exemplar-based Control of Multi-Car Elevators and its Multi-Objective Optimization using Genetic Algorithm. **The 23Rd International Technical Conference On Circuits/systems, Computers And Communications**, Tokyo, p. 701-704, 2008.

IKUTA, Masaki; TAKAHASHI, Kenichi; INABA, Michimasa. Strategy Selection by Reinforcement Learning for Multi-car Elevator Systems. **2013 Ieee International Conference On Systems, Man, And Cybernetics**, [S.L.], p. 2479-2484, out. 2013.

KAPETANAKIS, Spiros; KUDENKO, Daniel. Reinforcement Learning of Coordination in Cooperative Multi-agent Systems. **Eighteenth National Conference On Artificial Intelligence**, Menlo Park, p. 326-331, 2002.

KAPETANAKIS, Spiros; KUDENKO, Daniel. Reinforcement Learning of Coordination in Heterogeneous Cooperative Multi-agent Systems. **Adaptive Agents And Multi-Agent Systems II**, [S.L.], p. 119-131, 2005.

KAZIL, Jackie; MASAD, David. **Mesa: Agent-based modeling in Python 3+**. 2018. Disponível em: https://mesa.readthedocs.io/en/master/. Acesso em: 3 mar. 2019.

KOK, Jelle R.; VLASSIS, Nikos. Sparse cooperative Q-learning. **Twenty-First International Conference On Machine Learning - Icml '04**, [S.L.], p. 319-326, set. 2004.

LAUER, M; RIEDMILLER, M. Reinforcement learning for stochastic cooperative multi-agent-systems. **Proceedings Of The Third International Joint Conference On Autonomous Agents And Multiagent Systems**, Lisboa, p. 1516-1517, jul. 2004.

LIU, Weipeng; LIU, Ning; SUN, Hexu; XING, Guansheng; DONG, Yan; CHEN, Haiyong. Dispatching algorithm design for elevator group control system with Q-learning based on a recurrent neural network. **2013 25Th Chinese Control And Decision Conference (Ccdc)**, [S.L.], p. 3397-3402, maio 2013.

MATARIć, Maja J. Designing and Understanding Adaptive Group Behavior. **Adaptive Behavior**, [S.L.], v. 4, n. 1, p. 51-80, set. 1995.

MATARIć, Maja J.. Reinforcement Learning in the Multi-Robot Domain. **Autonomous Robots**, [S.L.], v. 4, n. 1, p. 73-83, 1997.

MATOS, Tiago; BERGAMO, Yannick P; SILVA, Valdinei da; COZMAN, Fabio G; COSTA, Anna. **Simultaneous Abstract and Concrete Reinforcement Learning**. 2011. Disponível em: http://www.each.usp.br/valdinei/Papers/sara2011.pdf. Acesso em: 15 jan. 2019.

MES, Martijn; HEIJDEN, Matthieu van Der; VAN HARTEN, Aart. Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. **European Journal Of Operational Research**, [S.L.], v. 181, n. 1, p. 59-75, ago. 2007.

PAN, Wei; CAI, Zixing; LIU, Limei; CHEN, Baifan. An Approach to Cooperative Multi-Robot Map Building in Complex Environments. **2008 The 9Th International Conference For Young Computer Scientists**, [S.L.], p. 1733-1737, nov. 2008.

PANAIT, Liviu; LUKE, Sean. Cooperative Multi-Agent Learning: the state of the art. **Autonomous Agents And Multi-Agent Systems**, [S.L.], v. 11, n. 3, p. 387-434, nov. 2005.

ROCHA, R.; DIAS, J.; CARVALHO, A.. Cooperative Multi-Robot Systems A study of Vision-based 3-D Mapping using Information Theory. **Proceedings Of The 2005 Ieee International Conference On Robotics And Automation**, [S.L.], p. 384-389, 2005.

ROOKER, Martijn N.; BIRK, Andreas. Combining Exploration and Ad-Hoc Networking in RoboCup Rescue. **Robocup 2004**: Robot Soccer World Cup VIII, [S.L.], p. 236-246, 2005.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 3. ed. Upper Saddle River: Prentice Hall Press, 2009.

SAłUSTOWICZ, Rafał P.; WIERING, Marco A.; SCHMIDHUBER, Jürgen. Learning Team Strategies: soccer case studies. **Machine Learning**, [S.L.], v. 33, n. 2/3, p. 263-282, 1998.

SEN, Sandip; SEKARAN, Mahendra. Multiagent coordination with learning classifier systems. **Lecture Notes In Computer Science**, [S.L.], p. 218-233, 1996.

SEN, Sandip; SEN, Ip; SEKARAN, Mahendra; HALE, John. Learning to Coordinate Without Sharing Information. **In Proceedings Of The Twelfth National Conference On Artificial Intelligence.** Colorado Springs, p. 426-431. mar. 1994.

SHEN, Jiquan; WU, Zhiqiang. Service-Oriented Organization Management and Coordination Control of MAS. **2008 International Conference On Intelligent Computation Technology And Automation (Icicta)**, [S.L.], p. 479-483, out. 2008.

STONE, Peter; VELOSO, Manuela. Multiagent Systems: A Survey from a Machine Learning Perspective. **Autonomous Robots**, [S.L.], v. 8, n. 3, p. 345-383, 2000.

SUEMATSU, Nobuo; HAYASHI, Akira. A multiagent reinforcement learning algorithm using extended optimal response. **Proceedings Of The First International Joint Conference On Autonomous Agents And Multiagent Systems Part 1 - Aamas '02**, [S.L.], p. 370-377, 2002.

SUTTON, Richard s; BARTO, Andrew G. **Reinforcement learning: An introduction**. Cambridge: Mit Press, 2018.

TAKATA, Yuya; MIKURA, Yuki; UEDA, Hiroaki; TAKAHASHI, Kenichi. Cooperative Learning of BDI Elevator Agents. **Icaart 2010 - 2Nd International Conference On Agents And Artificial Intelligence**, Hiroshima, p. 172-177, abr. 2010.

TAN, Ming. Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents. In: TAN, Ming. **Readings in Agents**. San Francisco: Morgan Kaufmann Publishers Inc, 1998. p. 487-494.

TOKSARđ, M. Duran. Ant colony optimization approach to estimate energy demand of Turkey. **Energy Policy**, [S.L.], v. 35, n. 8, p. 3984-3990, ago. 2007.

TUMER, Kagan; AGOGINO, Adrian K.; WOLPERT, David H.. Learning sequences of actions in collectives of autonomous agents. **Proceedings Of The First International Joint Conference On Autonomous Agents And Multiagent Systems Part 1 - Aamas '02**, Bologna, p. 378-385, 2002.

VALDIVIELSO, Alex; MIYAMOTO, Toshiyuki. Multicar-Elevator Group Control Algorithm for Interference Prevention and Optimal Call Allocation. **Ieee Transactions On Systems, Man, And Cybernetics - Part A**: Systems and Humans, [S.L.], v. 41, n. 2, p. 311-322, mar. 2011.

VALDIVIELSO, Alex; MIYAMOTO, Toshiyuki; KUMAGAI, S. Multi-Car Elevator Group Control: Schedule Completion Time Optimization Algorithm with Synchronized Schedule Direction and Service Zone Coverage Oriented Parking Strategies. **The 23Rd International Technnical Conference On Circuits/systems, Computers And Communications, Session H5 Elevator Control Systems.** Himeji, p. 689-692. jul. 2008.

WATKINS, Christopher J. C. H.; DAYAN, Peter. Q-learning. **Machine Learning**, [S.L.], v. 8, n. 3-4, p. 279-292, maio 1992.

XUE, L. H.. **Fuzzy neural network based elevator group control method with genetic algorithm**. 2002. 137 f. Dissertação (Mestrado em Ciência da Computação) - Tianjin University, Tianjin, 2002.

YANG, Erfu; GU, Dongbing. **Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey**. 2004. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.1602. Acesso em: 3 fev. 2019.