A solution for the Elevators Group Dispatch by Multiagent Reinforcement Learning

Jordão Memória, José E. B. Maia

[1]CCT - Universidade Estadual do Ceará (UECE)
60914-703 - Fortaleza - Ceará - Brasil

jordaomemoria@gmail.com,jose.maia@uece.br

Abstract. *In this work, a modeling and algorithm based on multiagent reinforcement learning is developed for the problem of elevator group dispatch. The main advantage is that, along with the function approximation, this multi-agent solution leads to reduction of the state space, allowing complex states to be addressed with a synthesizing evaluation function. Each elevator is considered an agent that have to decide about two actions: answer or ignore the new call. With some iterations, the agents learn the weights of an evaluation function which approximate the state-action value function. The performance of solution (average waiting time - AWT), shown varying the traffic pattern, flow of people, number of elevators and number of floors, is comparable to other current proposals reported in the literature.*

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 et al. 2008a, Cao et al. 2008b, Valdivielso et al. 2008, Ikeda et al. 2008, Takata et al. 2010, Valdivielso and 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 [R. S. Sutton and A. G. Barto 2000]. 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 esperado 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. Um caminho trilhado para reduzir esta dificuldade é modelar o grupo de elevadores como uma Sistema Multi-Agente (SMA) [Barrios-Aranibar and Gonçalves 2007]. Essa abordagem geralmente resulta em reduzir o armazenamento (espaço de estados) para cada agente, porém sacrificando em muito a curva velocidade de aprendizado [R. S. Sutton and A. G. Barto 2000].

Neste trabalho, um sistema multiagente (SMA) que utiliza aprendizagem por reforço (AR) é proposto. O método de AR escolhido foi a aprendizagem Q [Watkins and 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 forma 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.

Na seção 2, são apresentados alguns trabalhos que propõem uma nova forma de controle de grupo de elevadores também utilizando aprendizagem Q. Na seção 3 a solução deste trabalho é descrita em detalhes, a qual consiste em uma heurística para aprender uma função de aproximação da função valor estado-ação. Na seção 4 são feitos experimentos que avaliam o desempenho da solução SMA proposta e na Seção 5 a conclusão desta pesquisa é apresentada.

2. Trabalhos Relacionados

Em [Liu et al. 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 (AST) calculados com a chamada atual e a AST prevista calculada sem a chamada atual. A estratégia de zona existe porque [Liu et al. 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 às 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 $r = \frac{D}{R}$, onde *D* é a distância em andares da chamada e *R* é o tempo de serviço da chamada específica. Por exemplo, uma pessoa decide se locomover do andar 2 ao 5 e passou 30 segundos para utilizar os elevadores. Logo r = 0,1. Nos resultados, o tempo de serviço médio 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 na seção 4.

Em [Ikuta 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 artigo utilizou exatamente o mesmo estado de [Ikuta et al. 2013], então para uma descrição detalhada deste estado a subseção 3.2 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 $R = \sqrt{R_{wt} + R_{jny} + R_{stp}}$, 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 influência 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 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 [Ikuta et al. 2013], que utiliza *N* redes neurais recorrente para retornar o valor de usar a ação a no estado s (uma rede para cada elevador). 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.

3. Modelagem

3.1. 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. A seguir cada módulo do problema será explicado para a compreensão da solução proposta.

3.2. 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
- Uma lista de tamanho *N* contendo as direções dos *N* elevadores (subindo, parado ou descendo)
- 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*)

- 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]*
- 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]*

Pode-se notar que o estado é global para todos os agentes da modelagem, onde cada agente influência no estado atual. Isso aumenta exponencialmente a quantidade de estados com a entrada de *N* elevadores e *M* andares. 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.

3.3. 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 tomada com base na 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.

3.4. Função de avaliação

A função de avaliação é inserida para atribuir um valor a se 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 and Dayan 1992]. Ela é definida por:

$$Q(s,a) = \theta_0 + \theta_1 DA \tag{1}$$

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.

3.5. 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 and Norvig 2009]:

$$Q'(s,a) = Q(s,a) + \alpha(r + \gamma max_{a'} Q(s',a') - Q(s,a)) \tag{2}$$

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 (2) são obtidos através da equação (1). Com $Q'(s,a)$ obtido, o aprendizado ocorre através da atualização dos pesos, baseado na regra de Widrow-Hoff [Russell and Norvig 2009], ou regra delta, da seguinte forma:

$$\theta_0 = \theta_0 + \alpha(Q'(s,a) - Q(s,a)) \tag{3}$$

$$\theta_1 = \theta_1 + \alpha(Q'(s,a) - Q(s,a))DA \tag{4}$$

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.

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 (1)

Algoritmo 1: getD(s, idButtom, agent)

Entrada: recebe o estado atual; o id do botão apertado; e o agente
Saída: retorna 1 ou -1

```
1  início
2      ds ←lista inicializada com N zeros;
3      andarChamada e sentidoChamada são inicializados com o id do botão
         apertado;
4      para cada a ∈ Agentes faça
5          dAndares ← 0;
6          dParadas ← 0;
7          contabiliza o número de andares e número de paradas até
             andarChamada;
8          dParadas ← dParadas + 1;
9          contabiliza o número de andares e número de paradas até o
             primeiro/último andar baseado em sentidoChamada;
10         ds[a.id] = dAndares + 2dParada;
11     fim
12     se min(ds) = ds[agent.id] então
13         retorna 1
14     fim
15     senão
16         retorna -1
17     fim
18 fim
```

Figura 1. Pseudocódigo descrevendo a heurística D

3.6. Distância heurística de D

O algoritmo 1 mostra o pseudocódigo da distância heurística do cálculo 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* com a quantidade de paradas que serão realizadas. Nesta heurística, parar uma vez possui a mesma distância que se movimentar por dois andares, logo

$$d = dAndares + 2dParadas \tag{5}$$

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.

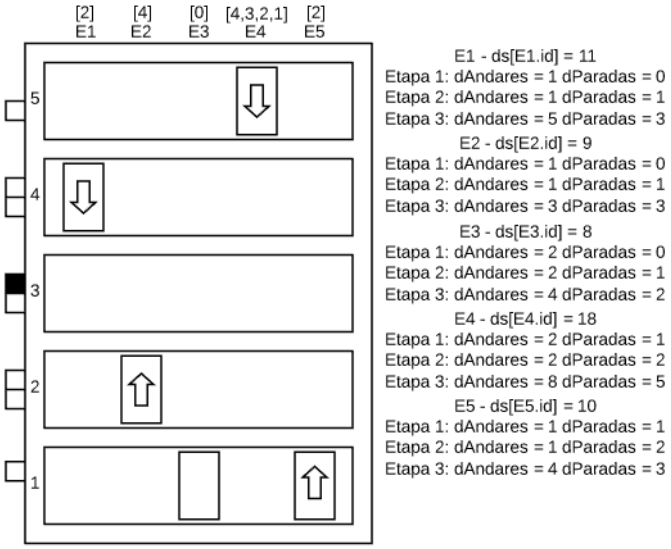

Figura 2. Diagrama de um estado exemplo com um novo chamado no andar 3 subindo, contendo as distâncias do 5 elevadores

Um exemplo concreto está mostrado na figura 2. Suponha um sistema com 5 elevadores atuando em 5 andares. O elevador E1, que está no andar 4, está indo pegar alguém no andar 2. E2, que está no andar 2, está indo deixar alguém no andar 4. E3 está ocioso no andar 1. E4 está no andar 5 e deve pegar/deixar pessoas no andar 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 2 mostra um diagrama que sintetiza o que foi descrito. O próximo passo é calcular a distância de cada elevador para o novo chamado. E1 move 1 andar e finaliza a primeira etapa. A segunda etapa sempre aumenta uma parada. Na terceira etapa, como E1 está indo pegar alguém no

segundo andar, acontece o seguinte: anda 1, para 1, anda 3 e para 1 (no andar 5). Assim, E1 finaliza com 3 paradas e 5 andares totalizando 11 de acordo com a equação (5). O mesmo é feito para os outros elevadores e a figura 2 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 D, utilizada na equação 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 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.

3.7. 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 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. Ressalte-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. Resultados e Experimentos

Esta seção apresenta os experimentos realizados para avaliar o desempenho da solução proposta. Ela contem 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 ordinário 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". Para avaliar o TMS, existem 4 atributos que serão variados ao longo desta seção: 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 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 1 mostra os parâmetros de ambiente referentes aos elevadores e a construção.

Tabela 1. 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 da construção	
Altura dos andares	3 metros

Cada simulação correu por uma hora (3600 segundos). A tabela 2 mostra as variáveis relevantes adotadas na implementação da solução.

Tabela 2. 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
Desconto da função de valor estado ação (γ)	0,3
Política de exploração (ϵ)	0,2
Duração do processo de Poisson	3600 segs

4.1. 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 ordinário 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. A figura 3 mostra o resultado do que acabou de ser explicado. Em 3(a), com o tráfego ordinário, o SMA fornece um TMS menor para todos os λs. Em 3(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 3(a), 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 3(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 3(c), com exceção dos menores valores de λ, o SMA proposto vence drasticamente o algoritmo ordinário. 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 - 3(c)
- chegada aleatória e saída aleatória resulta em resultados medianos - 3(a)
- chegada aleatória e saída específica resulta em resultados ruins - 3(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 na curva da figura 3. 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.

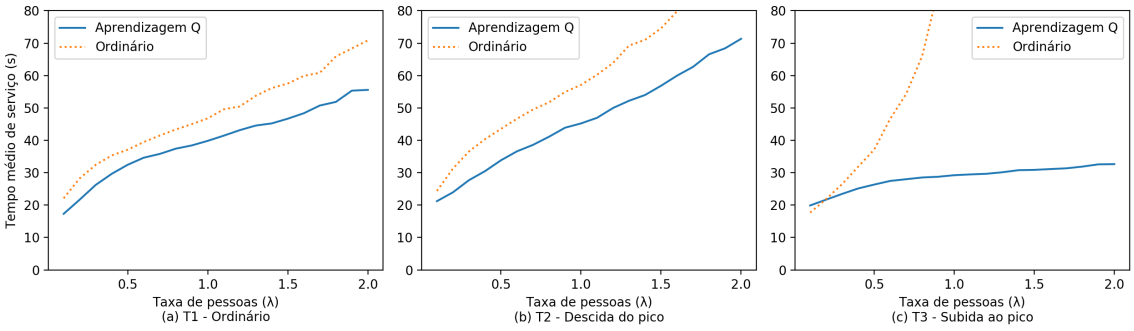

Figura 3. TMS dos três padrões de tráfego com λ variando de 0.1 a 2 para o algoritmo ordinário e a solução proposta

4.2. 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. Em seguida, o número N de elevadores varia de 2 a 10 e o número M de andares

varia de 2 a 50. Para cada combinação de N e M foi feita uma simulação e obtido o TMS. Assim, foi construído um gráfico 3D com esses números visando avaliar o comportamento do SMA. A figura 4 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 4, que mesmo variando os elevadores, para andares abaixo de 10, o TMS se mantém aproximadamente o mesmo.

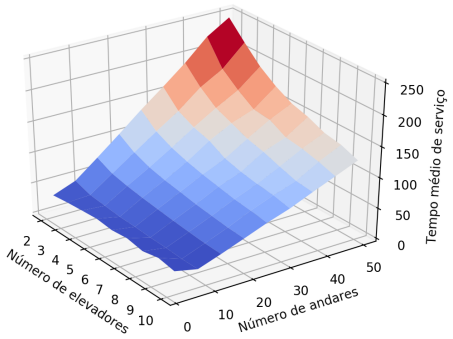

Figura 4. Tempo médio de serviço obtido com tráfego ordinário e $\lambda = 1$ variando o número de elevadores e de andares

4.3. 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 et al. 2013], baseada em redes neurais recorrentes e aprendizagem por reforço, a mesma comentada na seção 2; GA [Xue 2002] baseada em algoritmos genéticos; e SZ baseada em zoneamento estático. A tabela 3 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 3. Fluxo de chegada de pessoas customizado

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

A tabela 4 mostra o tempo médio de espera em segundos que significa o tempo que uma pessoa passa em média esperando o elevador chegar. Além disso, mostra tempo médio de jornada em segundos que representa o tempo médio de uma pessoa dentro de

um elevador. 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. Quando sob 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 espeta e jornada, SMARL mostra melhor desempenho além de menor porcentagem de aglomeração. Isso somado aos resultados da subseção 4.1 confirma que SMARL apresenta resultados especialmente satisfatórios para o tráfego de subida ao pico. Sob TF2, o tempo de espera médio continua o melhor, enquanto que nas outras duas medidas ele apresenta resultados competitivos. Caso ambos os tempo médios sejam somados, SMARL mostra um tempo superior apenas de SZ, por 0,57 segundos de diferença. E sob T3, SMARL mostra seu pior resultado ficando pior que todos os outros.

Tabela 4. 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)**	**Média de aglomeração (%)**
TF1	RL	50,19	45,82	68,09
	GA	75,16	53,89	89,13
	SZ	31,78	38,26	68,87
	SMA-RL	**9,33**	**41,08**	**52,12**
TF2	RL	33,48	40,56	68,91
	GA	64,81	52,85	88,27
	SZ	25,56	39,95	66,20
	SMA-RL	**18,94**	**47,14**	**88,12**
TF3	RL	11,95	15,38	11,47
	GA	7,35	17,53	15,60
	SZ	20,20	21,81	26,53
	SMA-RL	**15,49**	**33,75**	**69,90**

5. Conclusão

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 (Tabela 4) 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 porém mais eficiente em processamento. A solução se mostrou especialmente satisfatória quando o tráfego assumia pico de subida, o que provavelmente está relacionada com a forma de cálculo da distância heurística D.

Essa nova abordagem pode ser aperfeiçoada em vários pontos no futuro. Pretende-se investigar detalhadamente o comportamento do SMA proposto com outros padrões de tráfego, além de rever a função de avaliação e a regra de atualização dos pesos visando obter melhores índices de desempenho.

Referências

Aranibar, D. B. (2009). *Aprendizado por Reforço com Valores de Influência em Sistemas Multi-Agente*. PhD thesis, Universidade Federal do Rio Grande do Norte.

Barrios-Aranibar, D. and Gonçalves, L. M. G. (2007). Learning coordination in multi-agent systems using influence value reinforcement learning. In *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*, pages 471–478. IEEE.

Cao, L., Tian, J., and Zhang, Z. (2008a). Elevator group control system based on information fusion technology. In *2008 3rd International Conference on Innovative Computing Information and Control*, pages 473–473.

Cao, L., Zhou, S., and Yang, S. (2008b). Elevator group dynamic dispatching system based on artificial intelligent theory. In *2008 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, volume 1, pages 183–186.

Ikeda, K., Suzuki, H., Kita, H., and Markon, S. (2008). Exemplar-based control of multi-car elevators and its multi-objective optimization using genetic algorithm. In *The 23rd International Technical Conference on Circuits/Systems, Computers and Communications*, page 701–704.

Ikuta, M., Takahashi, K., and Inaba, M. (2013). Strategy selection by reinforcement learning for multi-car elevator systems. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2479–2484.

Liu, W., Liu, N., Sun, H., Xing, G., Dong, Y., and Chen, H. (2013). Dispatching algorithm design for elevator group control system with q-learning based on a recurrent neural network. In *2013 25th Chinese Control and Decision Conference (CCDC)*, pages 3397–3402.

R. S. Sutton and A. G. Barto (2000). reinforcement learning. In *Tokyo: Morikita Publiching Co.*

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Takata, Y., Mikura, Y., Ueda, H., and Takahashi, K. (2010). Cooperative learning of bdi elevator agents. In *ICAART 2010 - 2nd International Conference on Agents and Artificial Intelligence, Proceedings*, volume 2, pages 172–177.

Valdivielso, A. and Miyamoto, T. (2011). Multicar-elevator group control algorithm for interference prevention and optimal call allocation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(2):311–322.

Valdivielso, A., Miyamoto, T., and Kumagai, S. (2008). Multi-car elevator group control: Schedule completion time optimization algorithm with synchronized schedule direction and service zone coverage oriented parking strategies. In *The 23rd International Technnical Conference on Circuits/Systems, Computers and Communications, Session H5 Elevator Control Systems*, pages 689–692.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Xue, L. H. (2002). Fuzzy neural network based elevator group control method with genetic algorithm. Master's thesis, Tianjin University.