5
Avaliação Experimental

Com o objetivo de validar a modelagem proposta, foi desenvolvido um código que gerencia um ambiente de treinamento. Neste ambiente são controladas as condições de execução do treinamento, como quantidade de páginas da base, quantidade de páginas visitáveis por intervalo de tempo, duração do intervalo de tempo, taxa de aprendizagem, horizonte de tempo do treinamento, entre outros. Além disso, nesse ambiente é possível criar simulações de páginas com comportamento definido por um processo de Poisson, e também acessar os dados de edições sobre um conjunto de páginas da Wikipédia.

Neste capítulo são apresentados os conjuntos de dados utilizados nos testes, aspectos de implementação de nosso sistema, parâmetros de controle definidos nos ambientes e os resultados dos treinamentos do algoritmo de aprendizado por reforço. Por fim é apresentada uma discussão sobre os resultados alcançados.

5.1.
Conjunto de Dados

Dois conjuntos de dados foram criados para realização dos experimentos: um com simulações de páginas, com comportamento respeitando um processo de Poisson, e o outro com informações históricas sobre um conjunto de páginas da Wikipédia.

5.1.1.
Criação de conjunto de páginas de Poisson

Com o objetivo de validar o algoritmo de aprendizado por reforço, os primeiros testes da modelagem proposta foram realizados utilizando um conjunto de páginas simuladas, que têm taxas de edição fixas e bem definidas. Os valores dessas taxas são atribuídos aleatoriamente pelo programa de treinamento no

momento da criação de cada uma das páginas. Dessa forma o algoritmo tem acesso ao valor exato das taxas durante toda a execução do treinamento.

O restante do comportamento do sistema foi implementado de maneira a respeitar a modelagem proposta: distribuição das páginas em grupos, em função da taxa de edição e do tempo decorrido desde a última atualização. Como as páginas de Poisson apresentam taxa de edição constante, não acontecem mudanças de grupo como consequência de mudança de taxa de edição, somente ocorrem mudanças em função do tempo desde a última atualização.

É possível controlar quantos grupos o sistema terá, bem como o tamanho dos intervalos entre atualizações, a quantidade de páginas atualizada em cada oportunidade, a quantidade total de páginas, o intervalo de tempo para o qual deseja-se que o sistema apresente valor máximo de freshness.

5.1.2. Base de páginas da Wikipédia

Para realizar os testes comparativos entre as políticas clássicas de manutenção de atualidade de base de dados e a política proposta no trabalho, é necessário obter um conjunto de páginas que sofra edições periódicas e simular o comportamento dos algoritmos de revisitação atuando sobre uma cópia desse conjunto. Obter as páginas não é complicado; obter os seus registros de instante de edição pode ser mais difícil, pois exigiria um monitoramento constante desse conjunto, para posterior utilização em simulação.

O Projeto Wikipédia [WIK] oferece um repositório de artigos que contém, além das páginas que compõem a enciclopédia online, o histórico de suas modificações. Dessa maneira, uma única requisição ao servidor da Wikipédia resulta em informações sobre as últimas 500 modificações de um artigo, facilitando assim a obtenção do registro de edições.

À época da construção do repositório utilizado nesse trabalho, foram obtidas as páginas em inglês da Wikipédia disponíveis em junho de 2008, totalizando

5.453.838 artigos, e as informações de suas modificações realizadas em 2007. Cada artigo possui uma página contendo seu histórico de modificações. Por exemplo, podemos encontrar as edições realizadas sobre o artigo "Computer Science" na página [SZA]:

http://en.wikipedia.org/w/index.php?title=Computer_science&action=history

As informações sobre os artigos da Wikipédia e seus instantes de modificações foram armazenadas em um banco de dados MySQL, sobre o qual se realizam consultas que simulam o comportamento da cópia do conjunto de páginas no decorrer do tempo.

5.2. Aspectos de Implementação

É importante salientar que, diferentemente de outros métodos de aprendizado de máquina, não é comum encontrarmos bibliotecas de funções que sejam diretamente aplicáveis à modelagem do problema. Dessa forma, foi necessário criar todo o código para as simulações realizadas: representações das páginas, dos algoritmos, até mesmo das estruturas de dados utilizadas para organizar as páginas nos grupos.

As duas implementações de páginas, de Poisson (comportamento controlado) e da Wikipédia (de taxa de edição verificada no banco de dados), são especializações de uma classe abstrata, *Pagina*, que apresenta os métodos e atributos necessários para ter seu comportamento tratado pelo algoritmo de aprendizado por reforço. Nas páginas de Poisson, o parâmetro utilizado para separar os conjuntos foi o coeficiente de edição de páginas. Já para as páginas da Wikipédia, o valor médio dos últimos dez intervalos entre edições, μ, foi utilizado como parâmetro para agrupar as páginas.

Conforme foi apresentado na seção 5.1.1, para cada página de Poisson criada é atribuído um valor de taxa de edição λ. Esse valor é um número gerado aleatoriamente no intervalo (0,1) e dividido por 100, no instante em que a nova página é instanciada.

Sempre que for necessário determinar qual o próximo instante de edição de uma página, invoca-se um método da classe *PaginaPoisson*, que adiciona ao instante da última edição conhecida o valor do intervalo entre edições μ, calculado como $\mu = \frac{-\ln(1-k)}{\lambda}$. O fator k é um valor aleatório, entre *0* e *1*, que representa o valor acumulado da distribuição exponencial negativa que deve ser alcançado para que seja considerada a existência de um evento. Ao visitar uma página (e atualizar a cópia armazenada), é interessante determinar quando será o momento que esta receberá sua próxima edição (instante no qual a cópia ficará desatualizada). Para isso, invoca-se o método que calcula os instantes de edição da página, e obtém-se a informação do novo instante de tempo no qual a página passará a estar desatualizada.

Implementamos um *heap binário* para encontrar, de modo eficiente, quais páginas, num determinado momento, devem ser removidas de um grupo e posicionadas num grupo que contenha páginas visitadas há mais tempo.

Durante a fase de aprendizado, para determinar o valor de Q(s) para cada ação possível, o código precisa simular a tomada das ações sobre todo o conjunto de páginas. Isso é feito realizando-se as ações sobre *snapshots* do conjunto, e retornando ao conjunto original para a tomada da ação determinada pelo algoritmo. Esses *snapshots* são realizados criando-se uma cópia de todos os grupos e páginas, com as informações instantâneas de *freshness* de cada uma, para simulação da tomada de ações.

5.3. Testes com as páginas de Poisson

Os experimentos com as páginas de Poisson tiveram seus resultados comparados com um limite teórico, apresentado na próxima seção. Além disso, foram implementadas as políticas clássicas apresentadas na seção 2.5, que visitavam a mesma quantidade de páginas por intervalo de tempo que limita o *politeness* do algoritmo de aprendizado por reforço.

5.3.1.
Cálculo do freshness médio ótimo

O objetivo final do algoritmo aqui proposto é conseguir obter um alto valor de freshness médio para o conjunto de cópias de páginas web. Para saber quão bom é o resultado alcançado pelo algoritmo, é necessário saber qual o limite teórico para o freshness médio do conjunto.

Em [WNG], um estudo sobre a obsolescência e entrega de páginas web, nos é demonstrado como calcular um limite superior para páginas que se modificam de acordo com um processo de Poisson. Uma página é dita obsoleta se a cópia armazenada já está diferente da original, que foi atualizada.

Sendo N a quantidade de páginas que pode ser atualizada a cada intervalo de tempo de duração I, M o total de páginas constante no conjunto de cópias, μ_i o tempo médio entre edições de uma página *i*, a obsolescência ótima de uma página é dada por:

$$\frac{e^{-\frac{MI}{N\mu_i}} + \frac{MI}{N\mu_i} - 1}{\frac{MI}{N\mu_i}} \qquad \text{(Eq 22)}$$

Assim, sendo o freshness médio definido como a fração do tempo em que a página não está obsoleta, o freshness ótimo da página é dado por:

$$1 - \left(\frac{e^{-\frac{MI}{N\mu_i}} + \frac{MI}{N\mu_i} - 1}{\frac{MI}{N\mu_i}} \right) \qquad \text{(Eq 23)}$$

5.3.2.
Resultado dos testes com as páginas de Poisson

Utilizando a equação 23 podemos calcular qual seria o valor do freshness ótimo para os testes com as páginas de Poisson. Define-se os valores de controle como:

- Quantidade de páginas atualizáveis por intervalo de tempo (N): 250
- Duração do intervalo de tempo ($\Delta\tau$): 1 segundo

- Total de páginas no treinamento (M): 10000 páginas
- Horizonte de tempo utilizado (T): 1 hora

As taxas de edição são determinados como κ/100, sendo κ um real gerado aleatoriamente entre 0 e 1. O tempo médio entre edições, μ_i, é igual à média da distribuição exponencial, que é o inverso da taxa de edição.

Para estimar o valor ótimo esperado para o conjunto de 10000 páginas, foram gerados 10000 valores aleatórios de κ, entre 0 e 1. Os valores de tempo entre edições μ_i foram calculados como o inverso dessa taxa κ, e utilizados na equação 23.

Foi então calculada a média entre os 10000 valores obtidos, e chegou-se ao resultado de 0.906, valor teórico para o freshness médio ótimo.

Para os testes, foram realizados treinamentos com diferentes valores de taxa de aprendizagem (α) e de taxa de exploração (ε). Em todos os casos, o conjunto de treinamento era iniciado com freshness instantâneo ao redor de 0.2. Devido ao fato do algoritmo ser aleatório, dependendo da sequência de decisões de aprendizado tomadas, pode variar o tempo necessário para convergência ao valor máximo possível de freshness. Por esse motivo foram então realizados três experimentos com cada combinação de valores α e ε. Tabela 2 mostra a evolução do freshness dos conjuntos de treinamento, nos três experimentos realizados com cada combinação (α,ε). O eixo das abscissas representa o tempo (em segundos), e o das ordenadas, o freshness do conjunto.

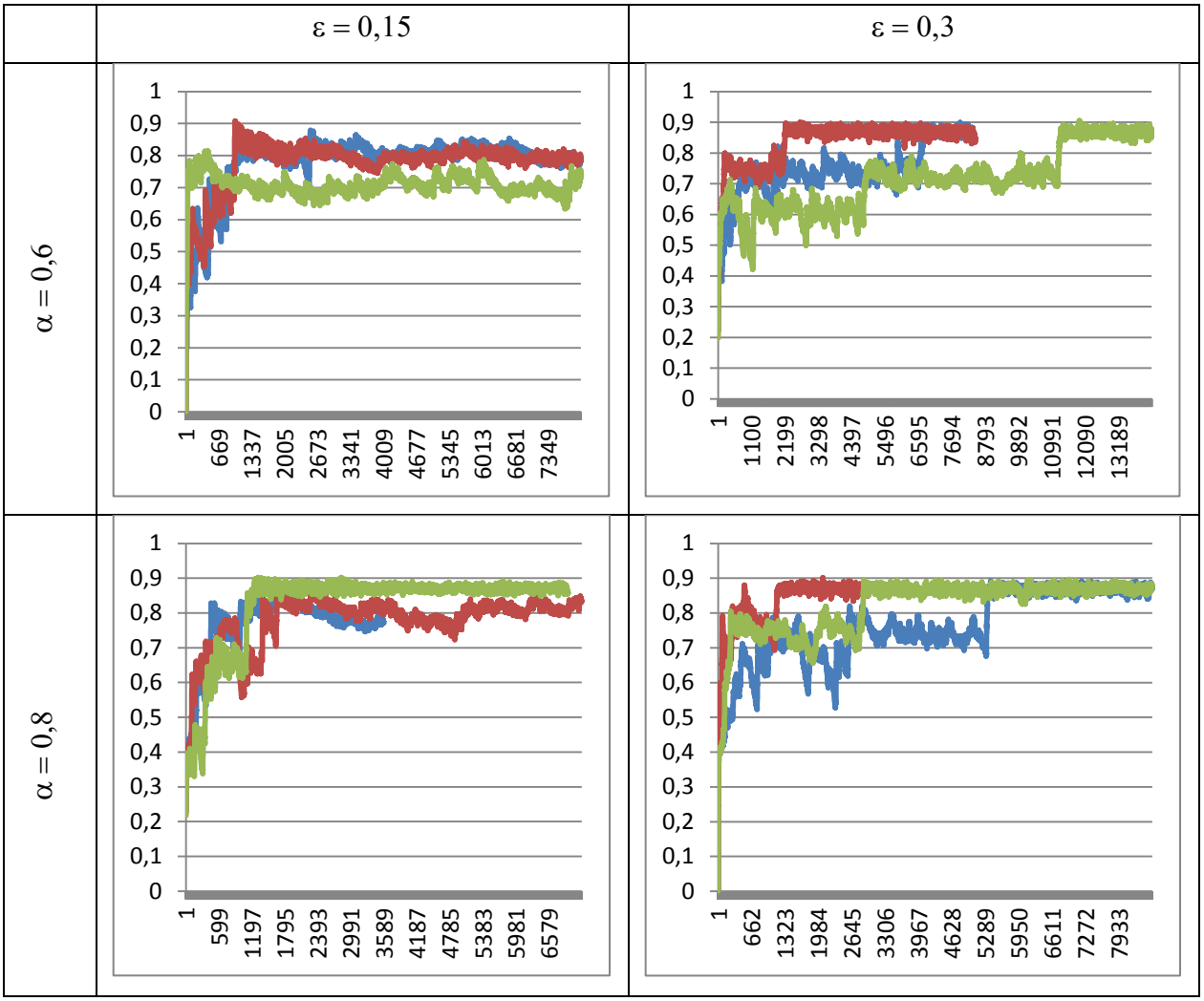

Tabela 5 – Gráficos da evolução do freshness do conjunto de treinamento para diferentes valores de α e ε, em função do tempo

Observando os gráficos do freshness do conjunto de testes durante o treinamento, podemos verificar que a combinação de $\alpha = 0.6$ com $\varepsilon = 0.15$ não produziu resultados interessantes em nenhuma dos três testes. Alterando o valor de α para 0.8, em um dos treinamentos o freshness do conjunto aproximou-se de 0.9. A alteração que causa melhor impacto no treinamento é a utilização do valor de taxa de exploração (ε) igual a 0.3, o que é comprovado nos dois gráficos da direita. Com essa taxa de exploração, todos os treinamentos conseguiam alcançar um freshness médio próximo de 0.9.

Após o treinamento foram realizados testes sobre novos conjuntos de dados. Para cada par (α,ε) foi escolhida a política aprendida no treinamento que alcançou

mais rapidamente o nível de freshness próximo ao ótimo (0.9). Nestes testes, também foram utilizados os mesmos valores de controle do treinamento, e o freshness inicial do conjunto também era próximo de 0.2.

A Figura 3 apresenta a comparação do comportamento das políticas aprendidas por cada um dos pares (α,ε).

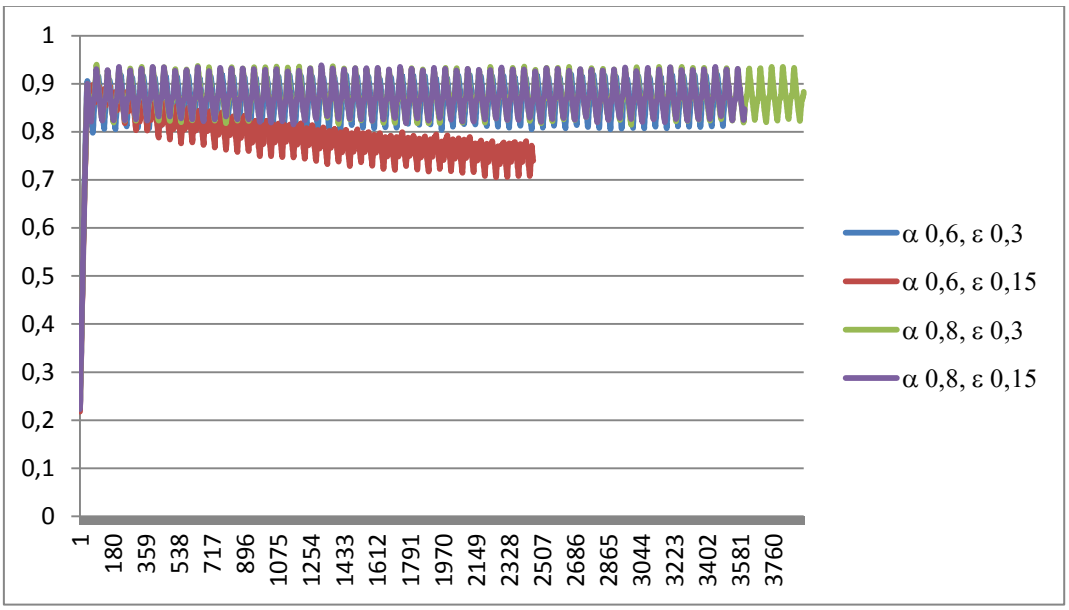

Figura 7 – Evolução do freshness do conjunto de páginas, seguindo as políticas aprendidas com diferentes valores de α e ε

É bastante nítida a semelhança da evolução de todas as combinações, exceto a que utiliza simultaneamente valores mais baixos de taxa de exploração e aprendizado. As políticas aprendidas nas demais configurações de taxas α e ε conseguem manter o freshness médio do conjunto de páginas muito próximo do valor ótimo calculado, 0.906.

Na Figura 8 é apresentada uma comparação do freshness alcançado pelas políticas aprendidas pelas combinações α = 0.8 e ε = 0.3 e α = 0.6 e ε = 0.15 com os valores alcançados, sobre o mesmo conjunto de testes, pelas políticas clássicas.

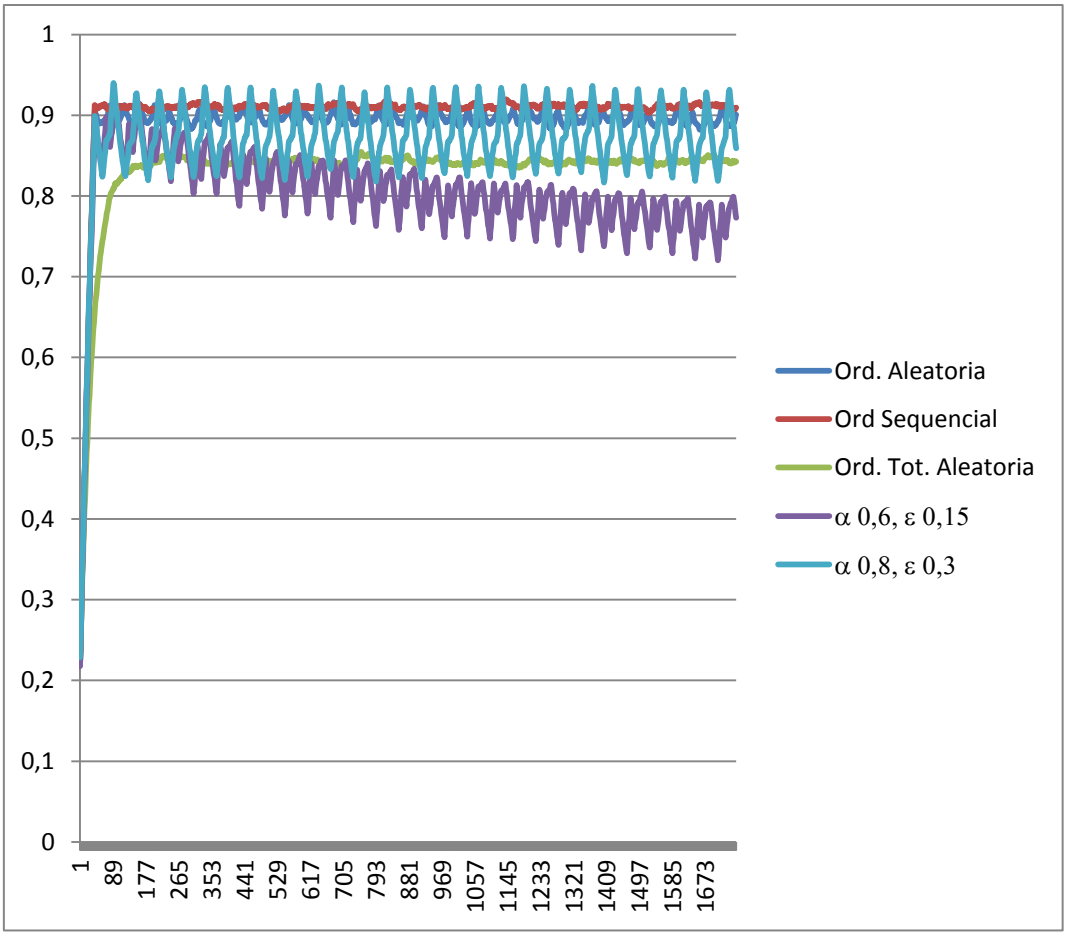

Figura 8 –Resultados de testes com páginas de Poisson - RL e políticas clássicas (freshness x tempo)

O freshness obtido pela política aprendida pelo algoritmo de RL, com valores maiores de α e ε, mantém-se com valores próximos aos obtidos pelas políticas clássicas. Os valores menores de α e ε fizeram com que o algoritmo aprendesse uma política que não consegue manter o freshness do conjunto compatível com o alcançado pelas políticas clássicas. Nas duas políticas aprendidas pelo algoritmo, há um certo padrão repetido periodicamente. Essa repetição pode ser explicada com o sistema alcançando repetidamente estados parecidos e, consequentemente, tomando ações equivalentes em intervalos regulares de tempo.

5.4. Testes sobre a base de páginas da Wikipédia

Nos testes realizados com as páginas da Wikipédia, o valor utilizado para agrupar as páginas foi o tempo médio entre edições, estimado pela média dos últimos dez intervalos entre edições da página.

Assim como nas páginas de Poisson, também existe na classe *PaginaWikipedia* um método que determina o instante em que a página sofre uma nova edição. Enquanto nas páginas de Poisson esse instante é calculado com uma fórmula, nas páginas da Wikipédia essa informação é obtida com uma consulta ao banco de dados, que informa o instante em que está registrada a ocorrência da próxima edição à página. O restante do processo de treinamento assemelha-se ao realizado com as páginas de Poisson.

A Figura 9 apresenta o histograma (em escala logarítmica) da quantidade de páginas em cada faixa de quantidade de edições recebidas.

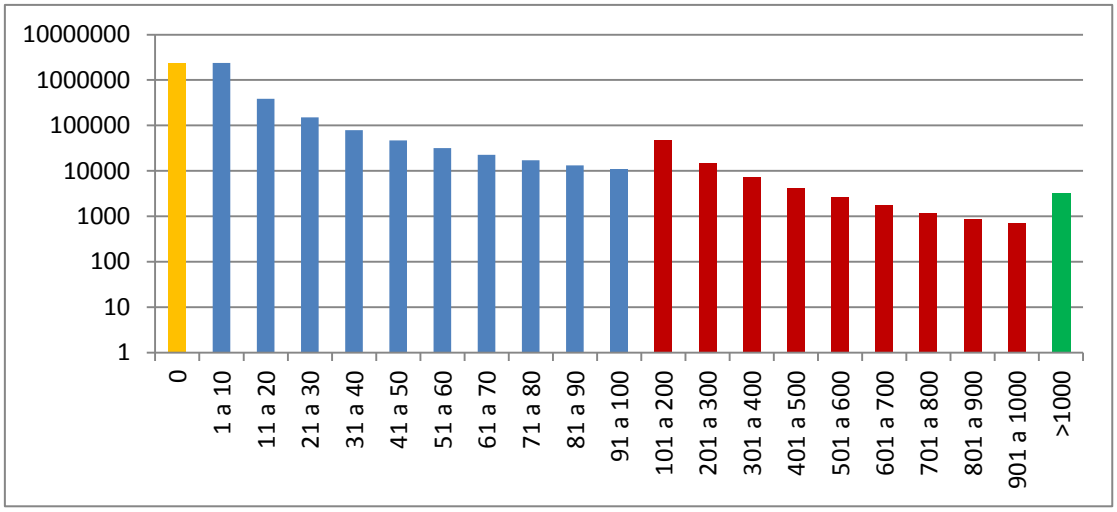

Figura 9 – Número de páginas por grupo de quantidade de edições em um ano

Das mais de 5000000 disponíveis no treinamento, 3205188 recebem ao menos uma edição, enquanto 2248557 páginas não recebem edições no período considerado. Além disso, na Figura 9 vemos que a quantidade de páginas que

recebem poucas edições é consideravelmente maior que a quantidade de páginas que são frequentemente editadas.

O valores utilizados para os testes foram:

- Quantidade de páginas atualizáveis por intervalo de tempo (N): 250
- Duração do intervalo de tempo (I): 10000 segundos
- Total de páginas no treinamento (M): 10000
- Horizonte de tempo utilizado (T): 6 meses

As 10000 páginas utilizadas para o treinamento foram selecionadas dentre as que possuíam ao menos 100 edições no período considerado (um ano). As páginas com menor quantidade de edições foram preteridas por dificultarem o cálculo de tempo médio entre edições bem como não causarem obsolescência nas cópias que devem ser mantidas atualizadas. Esse último efeito levaria a um freshness médio alto, mesmo quando o algoritmo estivesse fazendo escolhas ruins, já que dificilmente as cópias ficariam obsoletas.

Assim como nos testes realizados sobre as páginas de Poisson, foram estudadas diferentes combinações de taxa de aprendizado (α) e de exploração (ε). Os valores utilizados também foram os mesmos, 0.6 e 0.8 para a taxa de aprendizado, e 0.15 e 0.3 para a taxa de exploração.

Os treinamentos foram realizados sobre as informações de edições registradas no primeiro semestre de 2007. Os testes, tanto com as políticas clássicas quanto com as políticas aprendidas pelo algoritmo de RL foram realizados utilizando os dados de edições do segundo semestre de 2007.

A Figura 10 apresenta a evolução do freshness do conjunto de páginas, quando atualizadas segundo as políticas clássicas. A política que segue ordem totalmente aleatória foi a que apresentou pior resultado, oscilando o freshness do conjunto em torno de 0.48. A política de ordem aleatória teve freshness oscilando em torno de 0.55 e a ordem sequencial ficou em torno de 0.58.

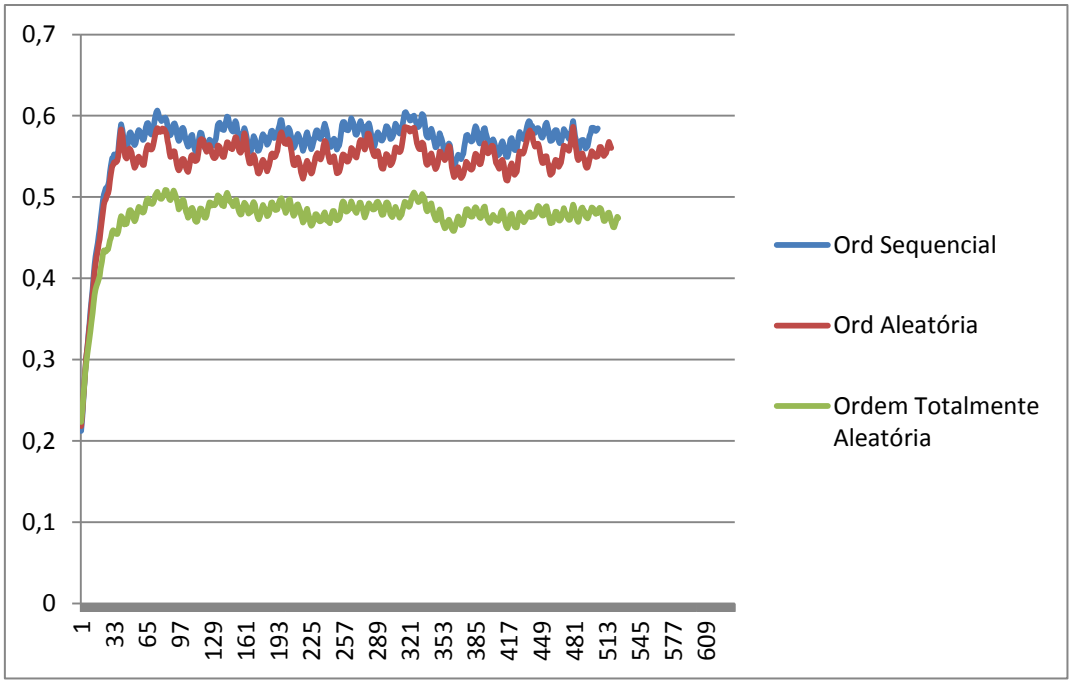

Figura 10 – Evolução do freshness do conjunto de páginas da Wikipédia, seguindo as políticas clássicas

Sobre as informações de edição de páginas no primeiro semestre de 2007, foram realizados treinamentos utilizando as quatro combinações de taxas de aprendizado e exploração, e a evolução dos treinamentos pode ser comparada na Tabela 3. Levando em consideração o fato do algoritmo de aprendizado ser aleatório, e que diferentes execuções podem levar a diferentes resultados, foram realizados três experimentos com cada combinação, para aumentar a chance do algoritmo convergir para uma boa política.

Durante o aprendizado, as duas combinações que utilizam taxa de aprendizado igual a 0.6 conseguiram atingir níveis de freshness próximos a 0.6. Também as combinações que utilizam taxa de aprendizado 0.8 conseguiram alcançar, durante o aprendizado, valores de freshness melhores que os atingidos pelas políticas clássicas.

Para a fase de validação, de cada combinação de α e ε foi utilizada uma das políticas aprendidas no treinamento. A seleção foi feita comparando-se os níveis de freshness atingidos por elas no treinamento, e escolhendo-se as que atingiram

valores mais estáveis de freshness do conjunto. Quando não havia grande diferença entre esses valores nas políticas, uma delas foi escolhida aleatoriamente.

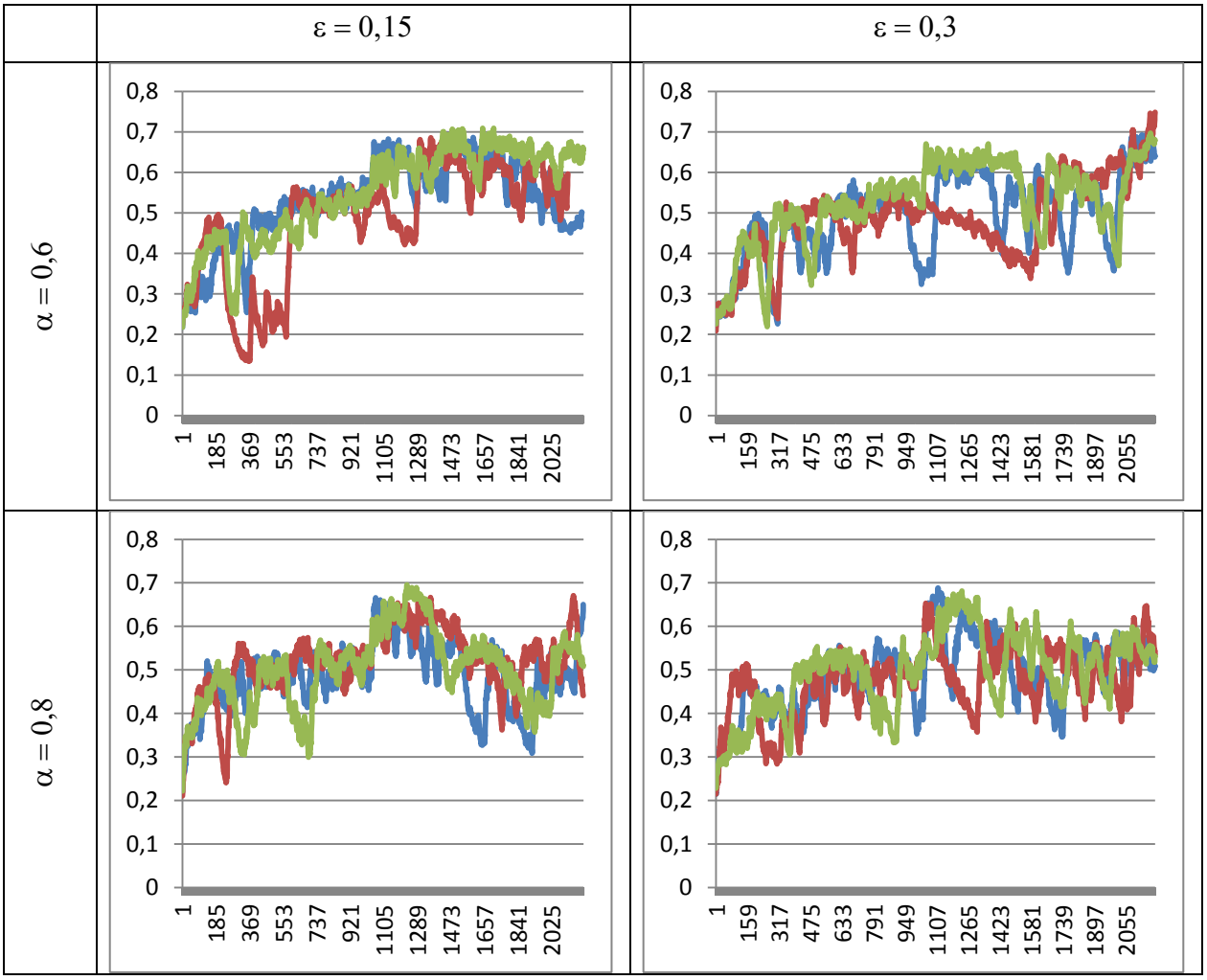

Tabela 6 – Gráficos da evolução do freshness do conjunto de treinamento (Wikipédia) para diferentes valores de α e ε, em função do tempo

As políticas foram então aplicadas sobre as páginas, considerando-se o período do segundo semestre de 2007. Os resultados podem ser observados na Figura 11.

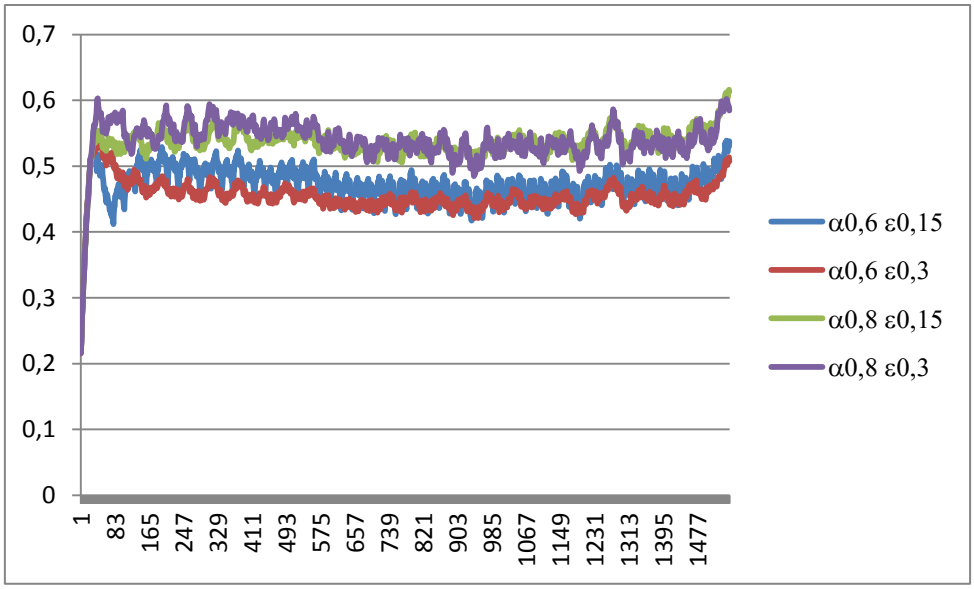

Figura 11 – Evolução do freshness do conjunto de testes (2º Sem. 2007 – Páginas Wikipédia) para diferentes valores de α e ε

Enquanto no treinamento as políticas que utilizaram taxa de aprendizado menor (0.6) apresentaram os melhores valores de freshness, durante os testes com aplicação das políticas aprendidas, os melhores resultados foram os obtidos pelas políticas que no treinamento utilizaram taxa de aprendizado 0.8. O freshness mantido por essas políticas oscilou na faixa de 0.55, comparável àquele mantido pelas políticas clássicas.

Com o objetivo de verificar se seria possível obter valores de freshness ainda melhores, foi realizada uma nova etapa de treinamento. As políticas aprendidas no primeiro treinamento foram utilizadas no início de um novo treinamento, ainda sobre o período do primeiro semestre de 2007, buscando aprimorar o aprendizado. A Figura 12 apresenta a evolução do freshness do conjunto durante o treinamento das quatro políticas selecionadas para validação. Esse treinamento foi iniciado com políticas aprendidas anteriormente.

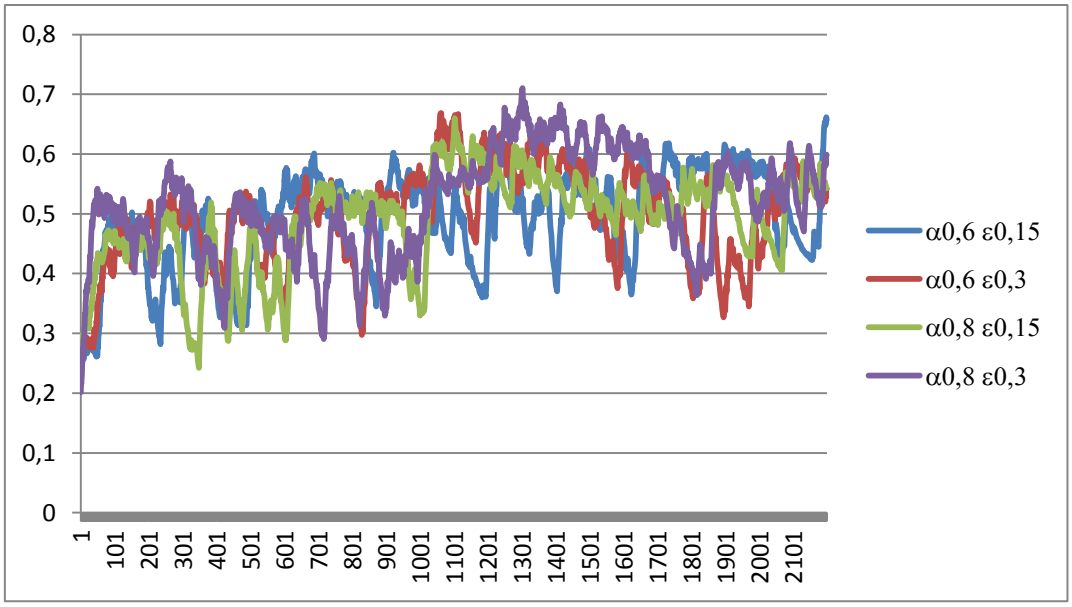

Figura 12 – Evolução do freshness do conjunto de treinamento, durante a segunda passagem de treinamento

Essa aproximação do comportamento das políticas, com diferentes valores para as taxas de aprendizado e exploração, sugere que é possível que todas elas venham a convergir para um valor máximo, dependendo do tempo de treinamento e das escolhas tomadas durante o aprendizado.

As novas políticas aprendidas foram novamente submetidas à validação sobre o conjunto de testes, e a evolução do freshness pode ser observada na Figura 13.

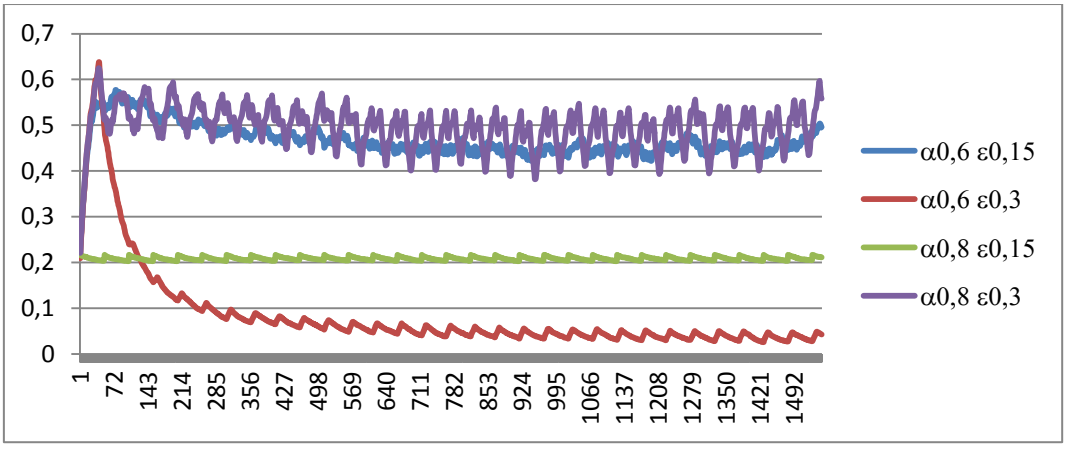

Figura 13 – Evolução do freshness do conjunto de testes, após segunda rodada de treinamento

As políticas aprendidas com os pares de taxa de aprendizado e exploração 0,6 e 0,3 e 0,8 e 0,15 apresentam, na figura 13, comportamento diferente do esperado. Como o experimento tem componentes aleatórios, podem ter sido aprendidas políticas que escolhem sempre um mesmo grupo de páginas para atualizar, e isto faz com que todas as outras páginas do conjunto fiquem permanentemente desatualizadas. Isso explica porque uma política mantém constante o valor de freshness inicial de 0,2, e outra política toma decisões adequadas no começo do experimento, levando o freshness instantâneo para acima de 0,6, mas depois opta por atualizar sempre páginas de um conjunto inadequado, mantendo assim o freshness do conjunto abaixo de 0,1.

Separando e comparando os desempenhos das políticas aprendidas em um ou dois treinamentos, podemos verificar o exposto nas Figuras 14 e 15.

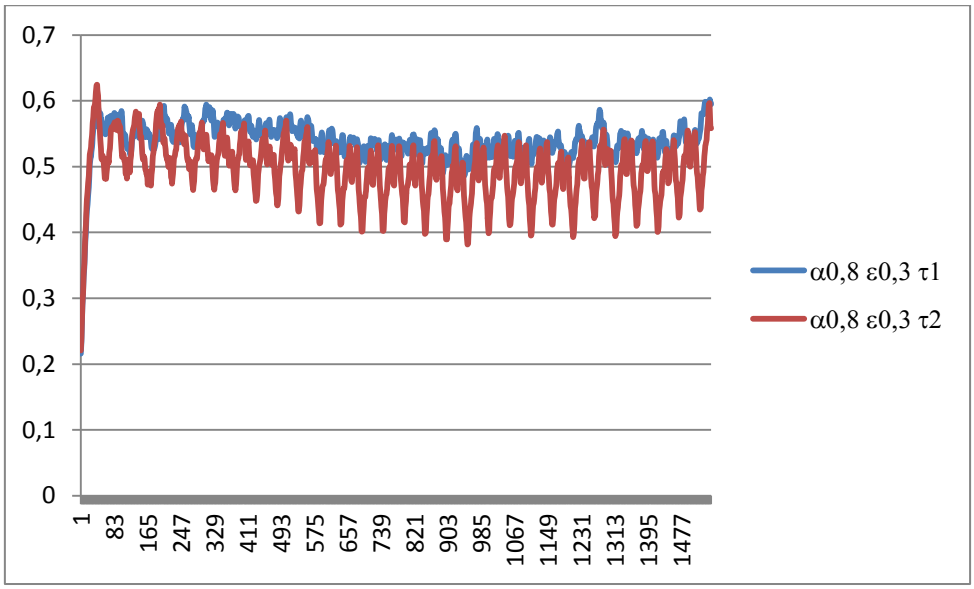

Figura 14 – Comparação de desempenho das políticas aprendidas em um treinamento (τ1) e em dois treinamentos (τ2), com os mesmos valores de α e ε

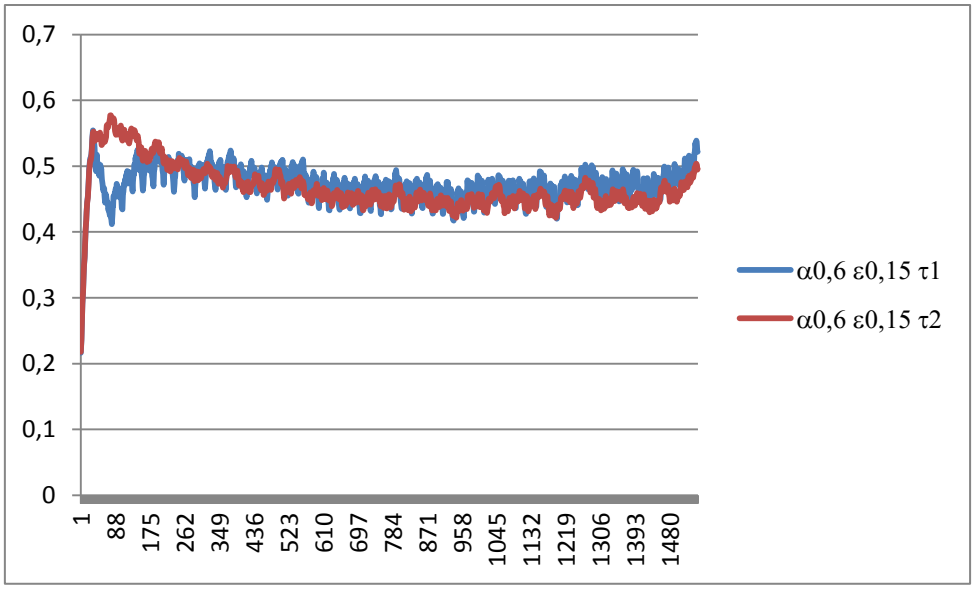

Figura 15 – Comparação de desempenho das políticas aprendidas em um treinamento (τ1) e em dois treinamentos (τ2), com os mesmos valores de α e ε

O segundo treinamento não deixou as políticas mais eficientes – a aprendida com $\alpha = 0.8$ e $\varepsilon = 0.3$ passou permitir maiores oscilações do valor de freshness, enquanto a aprendida com $\alpha = 0.6$ e $\varepsilon = 0.15$ inicialmente atingiu maiores valores de freshness, mas no longo prazo manteve níveis inferiores aos obtidos pela política aprendida com somente um treinamento.

Por fim, a Figura 16 apresenta uma comparação do desempenho das políticas aprendidas com uma execução do treinamento com as políticas clássicas.

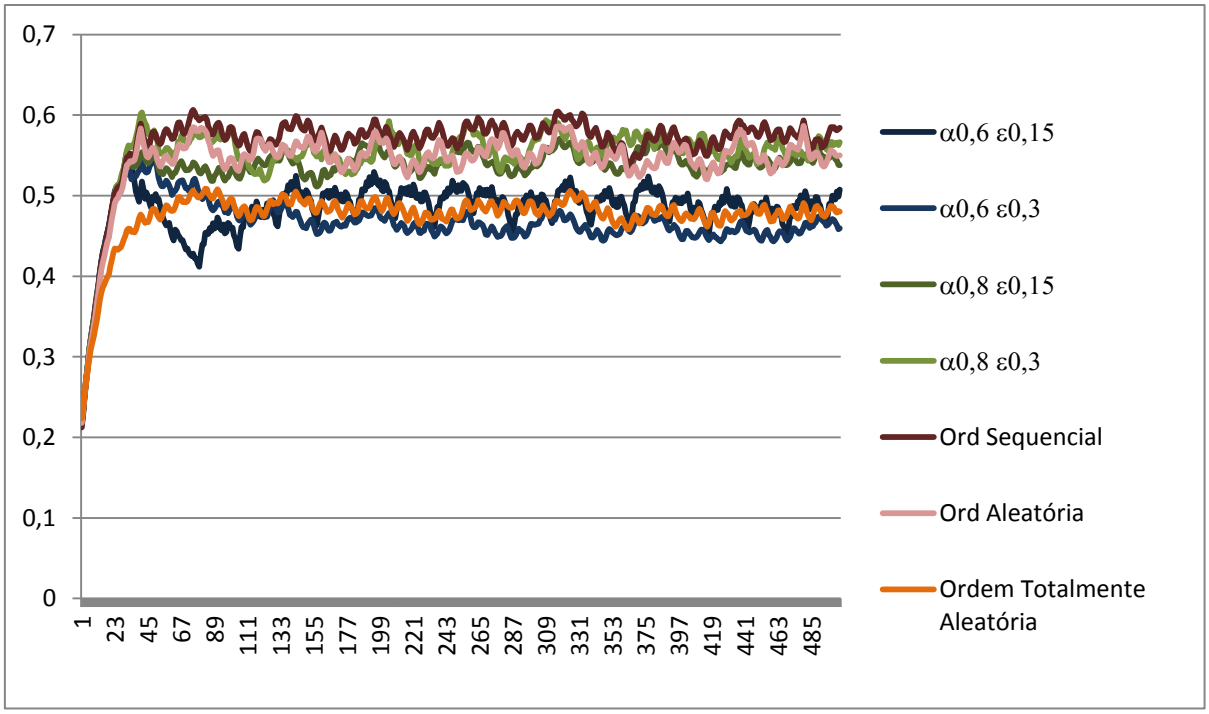

Figura 16 – Comparativo das políticas clássicas com as políticas aprendidas por RL

É interessante notar que as políticas aprendidas com taxa de aprendizado 0.6 alcançaram desempenho semelhante ao obtido pela política de Ordem Totalmente Aleatória, enquanto as políticas aprendidas com taxa de aprendizado 0.8 alcançaram desempenho equivalente ao das políticas de Ordem Sequencial e de Ordem Aleatória.

Novos treinamentos foram realizados, oferecendo ao algoritmo a oportunidade de aprimorar o treinamento com um maior número de passagens. Com as mesmas combinações de α e ε, foram executadas 5 passagens de treinamento. O resultado da aplicação das políticas aprendidas nestes treinamentos sobre o conjunto de testes é exibido na Figura 17. Mais uma vez, uma das políticas apresenta comportamento diferente do demonstrado pelas demais. Novamente o fator aleatório do treinamento pode ter levado o algoritmo a tomar decisões atrasadas para a atualização das páginas, levando assim aos valores menores de freshness alcançados pela política.

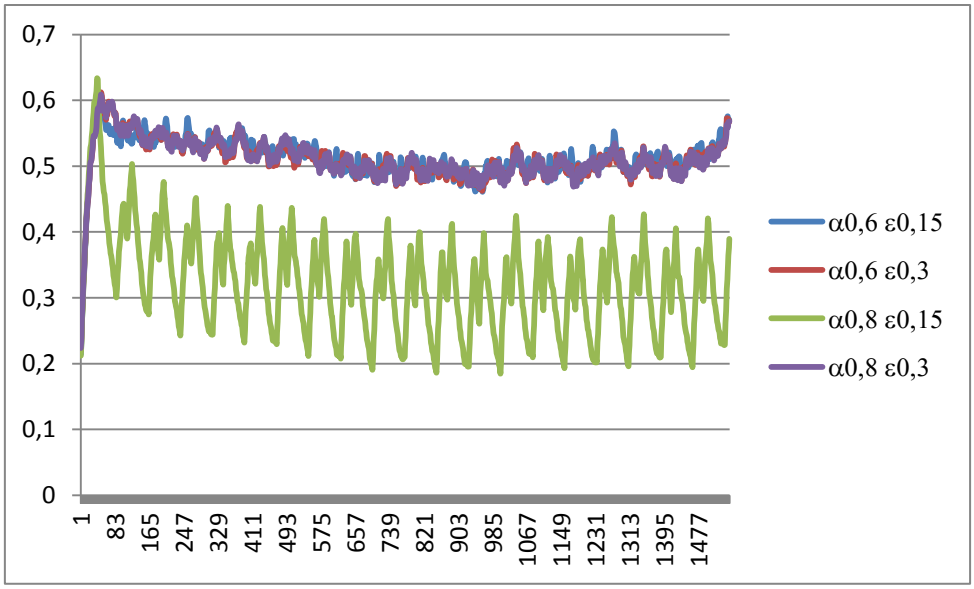

Figura 17 – Freshness do conjunto de testes, utilizando as políticas aprendidas com 5 passagens de treinamento

Na Figura 12 foi ilustrada a evolução do freshness com a utilização das políticas aprendidas em 2 passagens sobre o conjunto de treinamento. Comparando tais resultados com o desempenho obtido após 5 treinamentos, percebe-se que, de maneira geral, todas as combinações de α e ε acabaram por aprender políticas parecidas, e todas elas melhores que as aprendidas com 2 treinamentos. Mesmo a política com α 0.8 e ε 0.15, que nos 5 treinamentos não alcançou os mesmos níveis das demais, melhorou em comparação com a política aprendida em 2 treinamentos.

Um treinamento alternativo foi realizado, executando-se cinco passagens sobre o conjunto de treinamento, com os valores de α e ε diminuindo a cada passagem. Na primeira passagem, α valia 1.0 e ε 0.5. A cada passagem, o valor de α diminuía 0.2 e o de ε 0.1. Os valores mais altos, no início, sugerem uma maior predisposição à exploração e à aprendizagem; nos treinamentos onde esses valores são menores, espera-se que o algoritmo comece a valorizar a experiência adquirida nas fases anteriores de treinamento.

Após o treinamento, foi realizada a experimentação sobre o conjunto de testes, e o resultado é exibido na Figura 18.

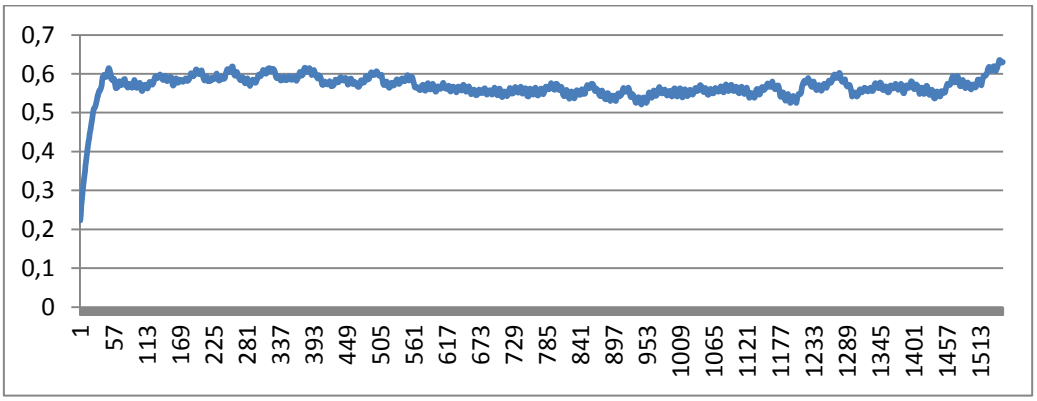

Figura 18 – Evolução do freshness do conjunto de testes, com a política aprendida com α e ε decrescentes

Comparando-se o desempenho desse treinamento com a melhor política aprendida com cinco passagens, mantendo-se α e ε fixos tem-se o que é observado na Figura 19. A política que começa com valores maiores de α e ε e tem esses valores decrescentes com o tempo alcança melhores resultados que as políticas que mantém esses valores fixos.

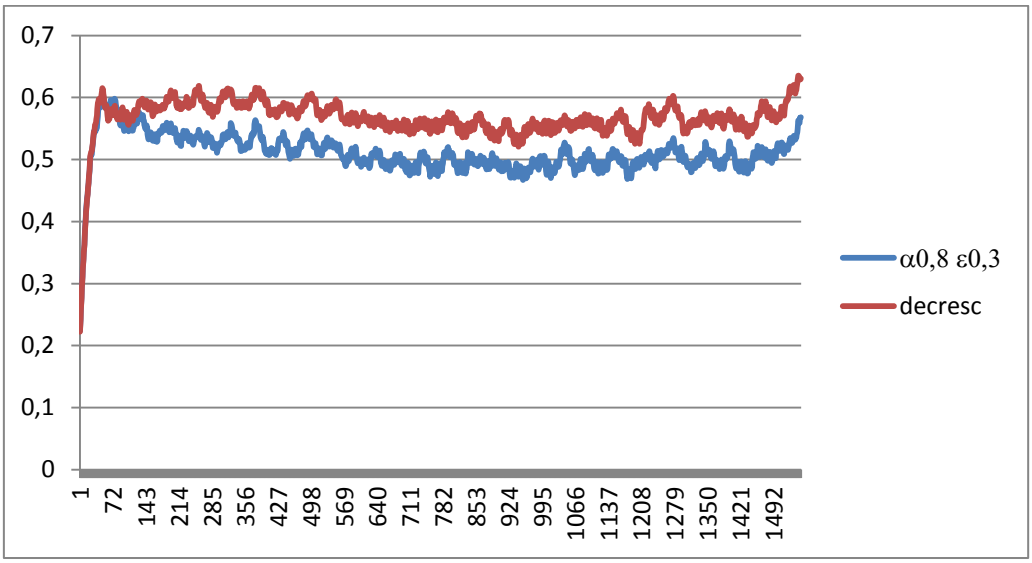

Figura 19 – Comparativo entre políticas aprendidas com 5 treinamentos, uma com valores de α e ε fixos, outra com valores decrescentes no tempo

Ao comparar-se, então, a política aprendida com os valores decrescentes de α e ε com as políticas clássicas, percebe-se que essa abordagem alcança, e consegue manter, valores de freshness melhores que todas as políticas do baseline. Este comparativo é exibido na Figura 20.

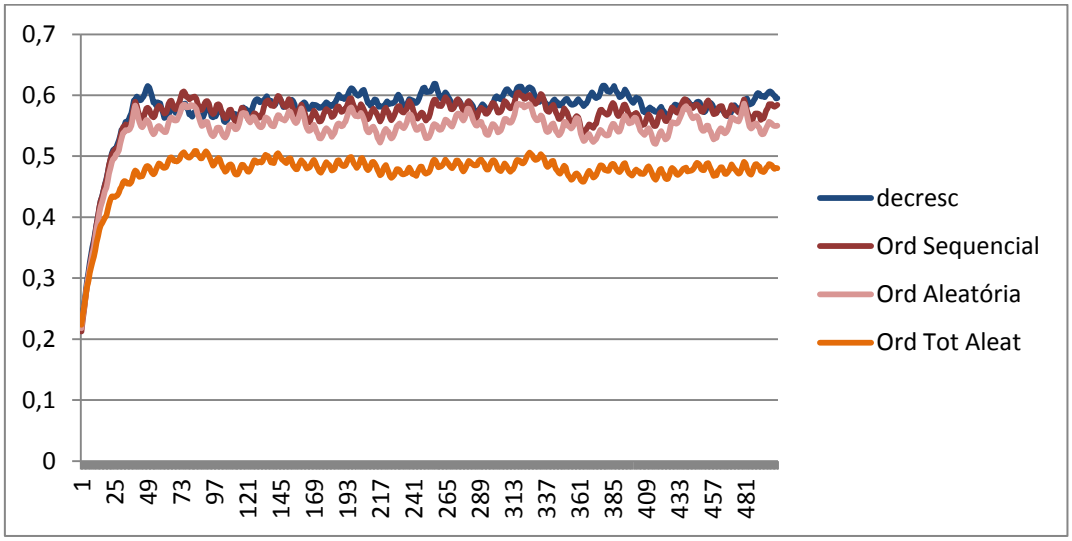

Figura 20 – Comparativo entre a política aprendida com valores decrescentes no tempo de α e ε com as políticas clássicas

Os valores de freshness médios são 0.57714 para a política aprendida com coeficientes decrescente, 0.56536 para a política que segue ordem sequencial de revisitação, 0.54333 para a política de ordem aleatória e 0.47559 para a política de ordem totalmente aleatória. Ao comparar-se esses valores, vemos que a política aprendida pelo algoritmo de aprendizado por reforço, em seu treinamento mais eficiente, obteve ganhos de 2.08%, 6.22% e 21.35%, respectivamente.

5.5. Discussão

Da análise dos resultados obtidos, vemos que para as páginas de Poisson o algoritmo de aprendizado por reforço conseguiu aprender satisfatoriamente bem o comportamento do sistema, atingindo um nível de freshness médio oscilando muito próximo do valor ótimo calculado. Apesar da simplicidade do

comportamento dessas páginas, esses testes confirmam que a modelagem feita do problema representa com eficiência a realidade do sistema.

Os testes com as páginas da Wikipédia mostraram que é possível para o algoritmo de aprendizado por reforço aprender estratégias de atualização equivalente às políticas clássicas. Quando o treinamento é repetido diversas vezes, os diferentes valores utilizados de taxa de aprendizado e exploração acabam por levar o algoritmo a aprender políticas parecidas. O melhor resultado foi percebido quando utilizou-se valores de taxas decrescentes no tempo. Assim, nas primeiras iterações, o algoritmo realiza mais exploração sobre o conjunto de ações possíveis e confere maior relevância aos resultados alcançados. Mais adiante no processo de treinamento, o conhecimento adquirido passa a ser mais valorizado e menos explorações são feitas. A política aprendida com essa abordagem foi a única que conseguiu alcançar e manter valores de freshness do conjunto de páginas melhor que todas as políticas utilizadas como baseline.

Como será comentado no capítulo 6, uma ampliação da capacidade de aprendizado do algoritmo, utilizando mais unidades de aprendizado e uma maior quantidade de *features* (características) para representar o estado do sistema podem levar o algoritmo a atingir resultados mais expressivos.