Uma Contribuição à Solução do Problema dos k-Servos Usando Aprendizagem por Reforço

Manoel Leandro L. Junior, Adrião D. Dória Neto, Jorge D. Melo
Departamento de Engenharia de Computação e Automação
Universidade Federal do Rio Grande do Norte
Natal, Brasil
E-mail: mleandro@dca.ufrn.br, adriao@dca.ufrn.br, jdmelo@dca.ufrn.br

***Resumo*— Este trabalho mostra um algoritmo original, baseado na aprendizagem por reforço, para resolver o Problema dos k-Servos (PKS). O PKS foi modelado como um processo de decisão em múltiplas etapas, e em seguida foi utilizado o algoritmo *Q-Learning* como método de solução. Vários experimentos foram realizados no intuito de verificar a adequabilidade e desempenho da solução proposta. Os resultados obtidos mostram a eficiência do algoritmo sugerido quando comparado com outros métodos de solução do PKS difundidos na literatura, e cujas taxas de competitividade já conseguiram ser provadas.**

***Palavras-Chave*: Sistemas Inteligentes, Aprendizado não-supervisionado, Aprendizagem por Reforço, Problemas dos k-Servos, Otimização.**

I. INTRODUÇÃO

A solução de problemas de computação *online* tem sido objeto de estudo há bastante tempo [1]. Esses problemas podem ser caracterizados como segue: dada uma seqüência de solicitações $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, um algoritmo *online* A deve atender cada uma dessas solicitações *online*, ou seja, sem o conhecimento prévio das solicitações subseqüentes. Desde que o atendimento das solicitações implica em custos, o objetivo a ser alcançado é a minimização desses custos. Tal problema pode ser visto como um processo de decisão em múltiplas etapas, onde a cada instante t_i $(i = 1, 2, \ldots, m)$ considerado, uma decisão deve ser tomada sobre como atender a solicitação σ_i. Esse processo é também *markoviano*, uma vez que a decisão a ser tomada no instante t_i depende apenas das informações disponíveis nesse instante. Para a solução de problemas de decisão markovianos, uma abordagem bastante eficiente é a aprendizagem por reforço [3]. Trata-se de um processo de aprendizagem não-supervisionada baseado em iterações entre um agente e o seu ambiente, conforme apresentado na Figura 1.

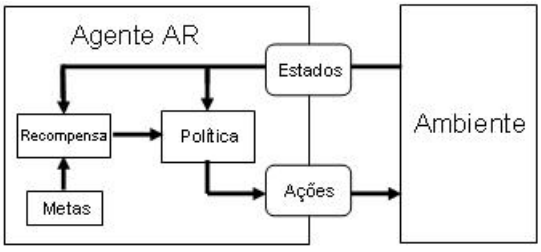

Fig. 1. Sistema de aprendizagem por reforço.

A cada passo de tempo, o agente observa o estado s_t do ambiente e seleciona uma ação a_t apropriada. A execução de uma ação produz uma mudança no estado do ambiente para um novo estado s_{t+1}, e uma avaliação desta ação, na forma de punição ou recompensa denotada por $r_{t+1}(s_t, a_t)$, é apresentada ao agente pelo ambiente. O processo de aprendizagem tem por finalidade orientar o agente a tomar as ações que venham a maximizar (minimizar) as recompensas (punições) recebidas. Deve-se levar em conta que uma ação tomada em um dado instante tem influência não apenas sobre a avaliação imediata, mas, também, sobre todas as outras ações que serão efetuadas a partir de então. Trata-se, portanto, do problema de como mapear situações do ambiente em ações do agente, de forma a maximizar (minimizar) um dado retorno.

O problema a ser resolvido pode ser estabelecido da seguinte forma: dado um estado incial $s_0 = s$ qual deve ser a *política* $\pi(s_t)$ empregada na escolha das ações a_t $(t = 0, 1, \ldots)$, tal que o retorno obtido $R(s, \pi) = \sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t))$, onde γ é um fator de desconto, $0 < \gamma < 1$, é ótimo em determinado sentido. Na solução deste problema, associa-se uma função de valor $V(s)$ ($Q(s, a)$) a cada estado (ou par estado-ação), que fornece uma indicação de quão bom é estar no estado s (ou estar em s e escolher a ação a), de forma a obter o melhor retorno R. Baseado em resultados da programação dinâmica ([7]), pode-se desenvolver algoritmos para a estimação da função de valor ótima e, conseqüentemente, da política ótima a ser empregada. Entretanto, a aprendizagem por reforço está limitada pela maldição da dimensionalidade inerente ao método, uma vez que torna-se necessário armazenar os valores de $V(s)$ para cada estado $s \in S$, o que é impossível se o número de estados é elevado.

Para contornar este problema, técnicas de parametrização da função de valor ou uso de decomposição hierárquica do problema têm sido apresentadas na literatura, visando melhorar a eficiência dos algoritmos de aprendizagem por reforço e possibilitar a aplicação destes métodos a uma ampla gama de problemas reais, como citado no início desta Seção. Em razão disso, será apresentada apresentada uma solução hierárquica baseada na aprendizagem por reforço para superar o problema do dimensionamento.

Este artigo está organizado da seguinte forma: na Seção 2 são apresentadas noções gerais sobre Computação *Online* e

Análise Competitiva, com ênfase no Problema dos k-Servos. Na Seção 3 é apresentado o *Q-Learning*, um dos métodos de solução mais importantes da aprendizagem por reforço e que foi utilizado neste trabalho. Na Seção 4 é apresentada a modelagem do Problema dos k-Servos segundo a aprendizagem por reforço. Os resultados obtidos e a análise comparativa para problemas que envolvam um número reduzido de nós são apresentados na Seção 5. Na Seção 6, é apresentado o problema do dimensionamento inerente ao método da aprendizagem por reforço e suas implicações. A solução proposta para contornar este problema, bem como uma comparação do desempenho obtido por esta solução será visto na Seção 7. As conclusões encontram-se na Seção 8.

II. Computação Online e Análise Competitiva

Conforme citado anteriormente, os problemas de computação *online* podem ser caracterizados como segue: dada uma seqüência de solicitações $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, um algoritmo *online* A deve atender cada uma sem o conhecimento prévio das solicitações subseqüentes. Assim, quando se está servindo uma demanda σ_t, com $1 \leq t \leq m$, as demandas futuras em um dado instante t' ainda não são conhecidas, isto é, $\sigma_{t'}$ é desconhecido se $t' > t$. Desde que o atendimento das solicitações implica em custos, o objetivo buscado é a minimização desses custos.

Formalmente, seja $G = (N, A)$ um grafo ponderado, onde N é o conjunto de nós, com $|N| = n$, e A o conjunto de arestas (i, j). A cada aresta (i, j) é associada uma ponderação $d(i, j)$, obedecendo a desigualdade triangular $d(i, l) \leq d(i, j) + d(j, l)$, dado que existe um caminho entre i e l passando por j.

Seja K um conjunto de facilidades (servos), com $|K| = k$, localizados em k nós de G não necessariamente distintos, e seja $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ uma seqüência de solicitações de serviço, as quais podem surgir em um nó qualquer do grafo G, apresentadas ao conjunto de servos. Para atender uma solicitação em um dado instante t_i ($i = 1, 2, \ldots, m$), um dos servos deve ser movido de sua posição atual para o nó σ_i em G. Associado ao deslocamento do servo de um nó i para um nó j, existe um custo de atendimento proporcional à distância percorrida $d(i, j)$.

O Problema dos k-Servos (PKS), introduzido por [5], pode ser definido da seguinte forma: dada uma configuração inicial dos servos em G e uma seqüência σ, encontre uma política π que indique os movimentos dos servos de modo que a distância total $\sum_{l=1}^{m} d_l(i, j)$ percorrida pelos mesmos seja a menor possível. O PKS serve como abstração para uma ampla gama de problemas *online* ([5]), tais como o problema de paginação de memória ([2]), otimização do gerenciamento de sondas de produção terrestre ([12]), disco de leitura/escrita ([13]), entre outros.

Uma solução ótima para o problema é normalmente possível se a seqüência inteira de solicitações é conhecida *a priori*. Nesse caso, o algoritmo de solução do problema é dito ser *offline*.

Um algoritmo *online* é dito ser c-competitivo se existirem constantes c e α, tal que $C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + \alpha$ para qualquer seqüência σ. Nessa expressão $C_A(\sigma)$ e $C_{OPT}(\sigma)$ correspondem aos custos associados ao algoritmo A e ao algoritmo ótimo *offline*, respectivamente. A constante c é chamada *razão de competitividade* de A, considerando que A é determinístico. O estabelecimento desses limites superiores e/ou inferiores de desempenho para um algoritmo *online*, seja ele determinístico ou randômico, fazem apelo a um conjunto de técnicas, tais como funções potenciais [2] e o princípio minimax de Yao [6], que não serão abordadas neste artigo.

III. O Algoritmo Q-Learning

O problema a ser resolvido é de controle ótimo em processos de decisão markovianos. O ambiente é representado por um conjunto finito de estados S, cujos elementos s_t representam os estados tomados no instante de tempo discreto t. Para cada estado, está associado um conjunto $A(s_t)$ finito de ações a_t. O sistema evolui dinamicamente de acordo com as suas probabilidades de transição

$$P_{s,s'}(a) = Pr\{s_{t+1} = s', r_{t+1} = r | s_t = s, a_t = a\} \quad (1)$$

que podem ser conhecidas (existe um modelo para o sistema) ou não. Usando a abordagem da aprendizagem por reforço, dado o processo markoviano, busca-se inicialmente estimar a função de valor estado-ação ótima $Q(s, a)$ associada ao seu respectivo par estado-ação (s, a). Essa função associa a cada par considerado o retorno médio esperado obtido quando uma ação particular é tomada em um dado estado e uma política $\pi(s, a)$ é seguida daí em diante.

Antes de detalhar o que vem a ser a função de valor estado-ação, apresentar-se-á algumas definições básicas associadas ao problema da aprendizagem por reforço [3].

Uma política $\pi_t(s, a)$ associada ao problema é um mapeamento das representações dos estados em probabilidades (no caso da política estocástica) de seleção de cada uma das ações possíveis, ou seja:

$$\pi_t(s, a) = Pr\{a_t = a | s_t = s\} \quad (2)$$

O retorno esperado, que corresponde ao valor esperado de todas as recompensas e/ou punições colhidas pela política empregada, é dado por

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_T \quad (3)$$

onde r_{t+i} é a recompensa/punição obtida no i-ésimo passo de tempo, e T é o horizonte de tempo. No caso de $T \to \infty$, o retorno esperado é dado por

$$R_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \quad (4)$$

onde γ é o fator de desconto, $0 \leq \gamma \leq 1$, utilizado para garantir que R_t seja finito.

Com base nas definições apresentadas, pode-se descrever formalmente o que é a função de valor estado-ação, associada

a uma dada política $\pi(s,a)$, através da equação:

$$Q^{\pi}(s,a) = E_{\pi}\{\sum_{k=0}^{\infty} | s_t = s, a_t = a\} \quad (5)$$

A questão central da aprendizagem por reforço pode então ser colocada:

- Dada uma política $\pi(s,a)$, qual a melhor forma de estimar $Q(s,a)$?
- Conhecendo-se uma resposta afirmativa para a questão anterior, de que forma essa política pode ser modificada tal que $Q(s,a)$ aproxime o valor ótimo dessa função e a conseqüente política ótima correspondente possa ser obtida?

Vários são os resultados encontrados na literatura que apontam uma resposta a estas questões, notadamente aqueles baseados em Programação Dinâmica [7], métodos de Monte Carlo e Diferenças Temporais [3]. Neste trabalho optou-se por utilizar o algoritmo *Q-Learning* desenvolvido por Watkins [8]. Dentre as vantagens desse algoritmo está o fato de ela aproximar diretamente o valor ótimo de $Q(s,a)$, independentemente da política utilizada. Os valores de $Q(s,a)$ são atualizados segundo a equação

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \cdot max_{a'}\ Q(s',a') - Q(s,a)] \quad (6)$$

onde α é a taxa de aprendizagem. O algoritmo que implementa o *Q-Learning* está mostrado na Tabela I:

TABELA I
DESCRIÇÃO DO ALGORITMO *Q-Learning*.

```
1   Initialize Q(s,a) randomicamente;
2   Para cada episódio
3       Initialize s;
4       Repita para cada passo do episódio
5           Escolha a para s usando a política π (ε-gulosa, por ex.);
6           Dado a ação a, observe r e s';
7           Q(s,a) ← Q(s,a) + α[r + γ · max_a' Q(s',a') − Q(s,a)];
8           s ← s';
10  até a condição de parada estabelecida
```

Dado que a convergência do algoritmo só é garantida se todos os pares estado-ação são visitados infinitas vezes, a escolha da política a ser utilizada no *Q-Learning* deve garantir que todos os pares tenham uma probabilidade não nula de serem visitados. Isto pode ser alcançado utilizando-se uma política ϵ-gulosa, definida por:

$$\pi(s,a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|}, & \text{se} \quad a = arg\ max_a\ Q(s,a) \\ \\ \frac{\epsilon}{|A(s)|}, & \text{se} \quad a \neq arg\ max_a\ Q(s,a) \end{cases}$$

IV. APRENDIZAGEM POR REFORÇO APLICADA AO PROBLEMA DOS k-SERVOS

Diferentemente da modelagem formal do Problema dos k-Servos apresentada anteriormente, onde os servos podem estar localizados em quaisquer nós, não necessariamente distintos, no modelo utilizado pela aprendizagem por reforço, puramente por razões de simplicidade, será considerado que os servos estarão localizados em nós necessariamente distintos, não sendo permitido que dois servos ocupem o mesmo nó em um dado instante de tempo. Esta restrição não significa uma limitação da abordagem da aprendizagem por reforço, sendo considerada somente para a simplificação na manipulação e geração dos estados apresentados. A modelagem pode ser estendida, sem dificuldades teóricas, para casos onde diversos servos ocupem o mesmo nó em um dado instante de tempo.

Do ponto de vista da aprendizagem por reforço, o problema pode ser modelado como segue: o estado do ambiente é representado por uma configuração possível dos servos, ou seja, por k-tuplos do tipo $s = \{no_1, no_2, \ldots, no_k\}$. O número total de estados possíveis é dado pela expressão:

$$C_{n,k} = \frac{n!}{(n-k)!(k)!} \quad (7)$$

As ações correspondem aos movimentos permitidos dos servos em um estado válido. Cada ação representa o movimento de um servo de um nó i para um nó j, no intuito de atender a requisição σ_j. Neste trabalho, só será considerado o atendimento de uma requisição por vez, deixando a análise de múltiplos servos para trabalhos futuros. Em cada estado, e considerando o surgimento de uma demanda σ_j em um dos n nós de um grafo G, um dos k servos será deslocado. Deste modo, o número de *ações permitidas* na ocasião é igual a k, todas do tipo mover servo localizado no nó i para o nó j, de forma a atender a solicitação σ_j. Como n demandas podem surgir por estado, o número total de *ações possíveis* é igual a $k \cdot n$.

A distinção deve ser notada entre os conceitos de ações *permitidas* e *possíveis*. Ações permitidas são as k ações que podem ser tomadas quando do surgimento de uma dada requisição, ocasionando o deslocamento de um dos k servos para atender a mesma. Ações possíveis são todas as ações que podem ser tomadas quando ainda não se conhece qual o nó onde irá surgir a próxima requisição, podendo a mesma, portanto, surgir em qualquer um dos n nós que compõem o grafo G. Conseqüentemente, qualquer um dos k servos pode ser deslocado para atender esta demanda, totalizando $k \cdot n$ ações possíveis.

O sinal de reforço corresponde à distância percorrida pelo servo k_i localizado no nó i para atender à demanda σ_j localizada no nó j, representado por $d(k_i, \sigma_j)$.

Pelo exposto, infere-se que para armazenar os valores da função de estado-ação Q, uma estrutura de dimensão $C_{n,k} \cdot k \cdot n$, onde $C_{n,k}$ representa o total de estados válidos e $k \cdot n$ o total de ações possíveis por estado. Logo, a complexidade em espaço do algoritmo *Q-Learning* é $\mathcal{O}(k \cdot n^{k+1})$.

Na solução da equação do *Q-Learning* apresentada acima, uma questão importante diz respeito ao cálculo do termo $max_{a'}\ Q(s',a')$. O diagrama de backup, correspondente ao problema de aprendizagem por reforço, que representa as transições em uma etapa e permite o cálculo da atualização de $Q(s,a)$ de acordo com o algoritmo *Q-Learning* está representado na Figura 2.

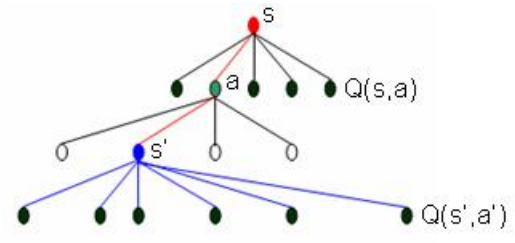

Fig. 2. Diagram de Backup do Algoritmo *Q-Learning*.

Na figura 2, os estados estão representados por círculos branco, vermelho (estado s) e azul (estado s'). As ações estão representadas por círculos preto e verde (ação a). Uma vez conhecidos s e a, tem-se o estado s'. O valor do termo $max_{a'}\ Q(s', a')$ é então tomado entre os valores de $Q(s', a')$ de todas as $k \cdot n$ ações possíveis de serem tomadas a partir de s'. Deve ser observado que não existe a necessidade de se conhecer qual ação será tomada em s', e sim quais são todas as ações possíveis de serem tomadas neste estado.

V. Análise Comparativa – Resultados para Problemas de Menor Porte

Inúmeros algoritmos *online* foram propostos para solucionar o Problema dos k-Servos. Dois deles se destacaram por causa do seu desempenho e passaram a ser bastante estudados na literatura: os algoritmos *Harmonic* e *Work Function* (WFA). Nos diversos estudos realizados sobre estes algoritmos, alguns autores conseguiram provar a competitividade de ambos em relação ao algoritmo ótimo *offline*, para algumas instâncias do problema ([9] e [10]).

O objetivo deste trabalho é verificar o desempenho obtido pela solução baseada na aprendizagem por reforço e compará-lo com o dos algoritmos *Harmonic* e *Work Function*, já que a competitividade de ambos foi provada, tornando-os parâmetros de comparação significativos. Um algoritmo proposto que apresente desempenho eficiente em relação aos mesmos tende a ser competitivo também, devendo, obviamente, se provar o referido fato. Tal prova, porém, foge aos objetivos deste texto.

Para testar o algoritmo desenvolvido neste trabalho, experimentos foram realizados no intuito de observar o desempenho da aprendizagem por reforço em comparação com os algoritmos mencionados. Foram realizados 4 experimentos, mantendo-se constante a quantidades de servos e variando-se a de nós. O grafo deve obedecer a desigualdade triangular [1], restrição imposta pelo *Work Function*. Em cada experimento, 100 sequências $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_{100}\}$ foram geradas. Uma seqüência contém 500 demandas σ_j. Para cada seqüência de requisições, registrou-se a distância percorrida ao se deslocar os servos, segundo os critérios de seleção estabelecidos por cada algoritmo. No final dos experimentos, anotou-se a menor e a maior distância percorrida para atender a seqüência de requisições, o valor médio das distâncias e quantas vezes cada algoritmo foi o melhor em comparação aos demais, vale dizer, quantas vezes conseguiu obter a menor distância percorrida para atender a seqüência de requisições. O treinamento da aprendizagem por reforço foi feito a partir de uma seqüência de 300.000 requisições apresentadas ao mesmo, considerando $\alpha = 0.1$, $\gamma = 0.1$ and $\epsilon = 0.1$. Os dados dos experimentos são mostrados na Tabela II para diferentes quantidades de nós e de servos.

TABELA II
Resultados obtidos nos testes.

Algoritmos	$n = 10\ k = 2\ iter = 3e05$			
	Menor	**Maior**	μ	**Vit.**
Q-Learning	7454	8751	8016.30	98
Harmonic	9823	11733	10806.45	0
Work Function	7601	9200	8313.65	2
	$n = 12\ k = 2\ iter = 3e05$			
	Menor	**Maior**	μ	**Vit.**
Q-Learning	7711	9342	8417.08	100
Harmonic	10191	12860	11678.44	0
Work Function	8089	9770	8818.73	0
	$n = 15\ k = 2\ iter = 3e05$			
	Menor	**Maior**	μ	**Vit.**
Q-Learning	8097	9730	8779.07	79
Harmonic	8190	9698	8941.41	0
Work Function	9951	11712	10796.38	21
	$n = 20\ k = 2\ iter = 3e05$			
	Menor	**Maior**	μ	**Vit.**
Q-Learning	9090	10862	9923.28	92
Harmonic	9375	11228	10164.30	1
Work Function	11212	13034	12043.72	8
Legenda	μ - Média Menor – Menor Distância Percorrida Maior – Maior Distância Percorrida Vit. – Vitórias Obtidas			

Pela análise dos resultados constatou-se que o desempenho do algoritmo proposto neste trabalho obteve, de modo geral, um desempenho mais eficiente nos testes quando comparado com os demais algoritmos. Em seu melhor desempenho, obteve 100% das vitórias nos testes realizados. Em seu pior, obteve 79% das vitórias. Em todos os experimentos, conseguiu obter mais vitórias do que os outros algoritmos.

Garante-se, pela teoria exposta anteriormente, que a política converge para valores ótimos se todos os pares estado-ação forem visitados infinitas vezes. Claramente, esta condição só pode ser obtida teoricamente. Na prática, pode-se inferir que quanto maior o número de exemplos fornecidos durante o treinamento, melhores serão as respostas obtidas pelo algoritmo, até o limite do ótimo, obviamente.

Pode-se treinar exaustivamente o *Q-Learning* até que suas respostas convirjam para valores ótimos antes de submetê-lo aos experimentos. Ou, por outro lado, o treinamento pode ocorrer simultaneamente aos experimentos. Nos experimentos, somente quando se chega uma solicitação *online* é que se recorre à política do *Q-Learning* para indicar as melhores ações a serem tomadas. Portanto, nada impede que durante o intervalo entre a chegada de solicitações *online*, o treinamento do *Q-Learning* aconteça. Obviamente, só faz sentido executar o treinamento até que os valores da política ainda não tenham atingido o ótimo.

A escolha das ações segundo a política do *Q-Learning* ocorre de forma imediata. É só consultar a entrada correspondente ao estado e à demanda na tabela e selecionar a ação

ótima.

O algoritmo *Harmonic* também indica a ação a ser executada imediatamente. Entretanto, como a escolha das ações é feita de maneira probabilística em função da distância do servo à demanda, não se garante que as melhores ações estão sendo selecionadas, fazendo com que o desempenho deste algoritmo seja significativamente inferior ao *Q-Learning*.

O algoritmo *Work Function*, por utilizar conceitos de programação dinâmica na escolha do deslocamento dos servos, pode apresentar retardos não-desprezíveis no tempo de apresentação da resposta do servo a ser deslocado, pois o cálculo para a escolha dos servos só ocorre quando se chega uma requisição. Trata-se de um fator primordial a ser considerado quando o tempo de resposta for crítico. Portanto, tanto sob o aspecto de desempenho quanto de tempo de resposta o *Q-Learning* apresenta-se mais vantajoso do que o *Work Function*

VI. Apresentação da Solução Hierárquica

Como pôde ser visto anteriormente, a dimensão da estrutura de armazenamento utilizada pela aprendizagem por reforço cresce em função do número de nós e de servos. Ao se analisar esse crescimento, percebe-se que o mesmo ocorre de maneira exponencial. Este problema, denominado *Maldição do Dimensionamento* (ou *Problema do Dimensionamento*), foi introduzido por Belmann [7] e significa a impossibilidade de execução de um algoritmo para certas instâncias de um problema pelo esgotamento de recursos computacionais para obtenção de sua saída.

Infere-se, conseqüentemente, que para que se possa aplicar a solução baseada no *Q-Learning* a uma ampla gama de aplicações, algum mecanismo deve ser criado para contornar o problema do dimensionamento inerente ao método da aprendizagem por reforço.

Para superar o problema supracitado, uma solução hierárquica baseada na aprendizagem por reforço e no método guloso foi proposta. A idéia geral é aplicar a aprendizagem por reforço a um número reduzido de nós, selecionados seguindo um critério específico, do conjunto de nós do problema e tentar generalizar o aprendizado obtido neste treinamento para outros pares estado-ação não visitados. Quando a generalização não for possível, utiliza-se o critério guloso para selecionar o servo a ser deslocado.

Os passos algoritmo que implementa a solução hierárquica são os seguintes:

- Divida o conjunto de nós em grupamentos de proximidade;
- Para cada grupamento formado
  - Escolha um nó para representar o grupo – nó-centro;
  - Execute o algoritmo da aprendizagem por reforço no conjunto de nós que compõem o grupo;
- Execute o algoritmo da aprendizagem por reforço nos nós escolhidos no passo anterior – nós-centro;
- Se o par estado-ação não foi visitado no passo anterior e a generalização não puder ser feita, utilize o critério guloso para escolher o servo a ser deslocado.

Para realizar a divisão do conjunto de n nós do grafo em x grupos ($x \leq n$), utilizou-se o *Algoritmo de Boruvka* [11], e foi feita a partir da *Árvore Geradora Mínima* [11] do grafo. O número de grupos formados é fornecido como parâmetro.

O critério de escolha dos nós que representam cada grupo, os quais denominou-se *nós-centro*, é a média de sua distância em relação aos demais nós do seu grupo. Em outras palavras, o nó selecionado será aquele que possuir, em média, a menor distância em relação a todos os outros nós que compõem seu grupo.

Nos primeiros passos do algoritmo hierárquico, o conjunto de n nós do problema foi dividido em x grupamentos de proximidade, e para cada grupamento um nó-centro foi selecionado, totalizando x nós-centro. A aprendizagem por reforço será aplicada no conjunto de x nós-centro e em cada conjunto de nós que compõem os x grupamentos, mantendo-se constante o total de k servos. A execução da aprendizagem por reforço, neste conjunto reduzido de nós, mantém as mesmas características da execução se a mesma ocorresse no conjunto de n nós do problema. Porém, deve-se observar que os servos e os possíveis locais de demanda estarão localizados somente nos nós selecionados para cada execução.

Quando os servos e as demandas pertencerem a grupos distintos, todos eles, e os mesmos não estiverem nos nós-centrais dos seus respectivos grupos, serão considerados como se estivessem. A partir desta suposição, utiliza-se o conhecimento obtido durante a execução da aprendizagem por reforço para escolha do servo a ser deslocado, já que o aprendizado com os servos e as demandas localizados nos respectivos nós-centros dos seus grupos já foi realizado. Esta transposição da posição dos servos ou da requisição faz com que o aprendizado possa ser utilizado em pares estado-ação que não foram visitados, generalizando o conhecimento obtido durante o aprendizado.

Quando o par estado-ação não foi visitado pela aprendizagem por reforço e a generalização do conhecimento não puder ser feita, o critério para o deslocamento dos servos será o guloso, ou seja, o servo a ser deslocado será o que estiver mais próximo à demanda.

A redução da dimensão da estrutura usada pela aprendizagem por reforço para armazenar os valores da função Q ocorrerá proporcionalmente a um *fator de redução* δ. Matematicamente, $\mathcal{C}_{\frac{n}{\delta},k} \cdot k \cdot \frac{n}{\delta}$, onde:

$$\mathcal{C}_{\frac{n}{\delta},k} = \frac{\left(\frac{n}{\delta}!\right)}{(\frac{n}{\delta} - k)! \cdot k!} \tag{8}$$

De posse destas informações e fazendo-se as manipulações necessárias, obter-se-á $\mathcal{O} = \frac{1}{(\delta)^k} \cdot n^k$ a complexidade em espaço da aprendizagem por reforço Hierárquica, sendo a redução da complexidade em relação ao *Q-Learning* da ordem de $\frac{1}{(\delta)^k}$.

VII. Análise Comparativa – Resultados para Problemas de Maior Porte

O objetivo desta seção é verificar a aptidão do algoritmo hierárquico em solucionar o PKS para problemas de maior

porte, comparando o seu desempenho com o algoritmo *Harmonic*, já que o mesmo não é afetado pelo problema do dimensionamento. Em função disso, resolveu-se um problema com um conjunto de nós de tamanho 100 e com 3 servos. Variou-se a quantidade de grupamentos formados para verificar se existia alguma relação com o desempenho obtido. Em cada experimento, 100 sequências $\sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_{500}\}$ foram geradas. Uma seqüência contém 500 demandas σ_j. Registrou-se a distância percorrida ao se deslocar os servos, segundo os critérios de seleção estabelecidos pelos algoritmos. No final dos experimentos, anotou-se a menor e a maior distância percorrida para atender a seqüência de requisições, o valor médio das distâncias e quantas vezes cada algoritmo foi o melhor em comparação ao outro, vale dizer, quantas vezes conseguiu obter a menor distância percorrida para atender uma seqüência de requisições. O treinamento da aprendizagem por reforço foi feito a partir de uma seqüência de 300.000 requisições apresentadas ao mesmo, considerando $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.1$. Os dados registrados nos experimentos são mostrados na Tabela III.

TABELA III
RESULTADOS OBTIDOS NOS TESTES.

	$n = 100\ k = 3\ grupos = 18\ iter = 3e05$			
Algoritmos	**Menor**	**Maior**	μ	**Vit.**
Hierárquico	6010	6794	6421.53	65
Harmonic	6045	6879	6465.77	35
	$n = 100\ k = 3\ grupos = 20\ iter = 3e05$			
Algoritmos	**Menor**	**Maior**	μ	**Vit.**
Hierárquico	6096	6867	6403.64	80
Harmonic	6176	6884	6498.21	20
	$n = 100\ k = 3\ grupos = 24\ iter = 3e05$			
Algorithms	**Menor**	**Maior**	μ	**Vit.**
Hierárquico	2055	3104	2517.07	75
Harmonic	2370	3699	2980.03	25
Legenda	μ - Média Menor – Menor Distância Percorrida Maior – Maior Distância Percorrida Vit. - Vitórias Obtidas			

Pela análise dos resultados apresentados, constatou-se que o algoritmo hierárquico foi mais eficiente do que o do *Harmonic*. Em seu melhor caso, obteve 80% das vitórias. Em seu pior caso, 65%. Em todos os casos, obteve mais vitórias do que o outro algoritmo.

VIII. CONCLUSÕES

A aprendizagem por reforço mostrou-se uma ferramenta eficaz no desenvolvimento de algoritmos para a solução do Problema dos k-Servos. Para ser utilizada, modelou-se o problema como um processo de decisão em múltiplas etapas e aplicou-se o algoritmo *Q-Learning*. Nos resultados obtidos pelo *Q-Learning* em problemas de menor porte, observou-se que o desempenho do algoritmo proposto foi melhor do que o alcançado pelos algoritmos *Harmonic* e *Work Function*, dois dos métodos de solução do PKS mais difundidos na literatura e cujas taxas de competitividade já foram provadas. Além disso, a restrição da desigualdade triangular imposta pelo algoritmo Work Function pode ser eliminada se for usada a solução apresentada. A abordagem da aprendizagem por reforço pode ser facilmente extendida a problemas correlatos ao k-Servos. A limitação do problema do dimensionamento inerente aos métodos de solução baseados na aprendizagem por reforço foi contornada ao se usar a solução hierárquica proposta neste texto, permitindo a utilização da mesma em um número maior de aplicações. Entretanto, a adequação da solução hierárquica a problemas reais ainda precisa ser verificada. Os resultados dos testes para problemas de maior porte mostraram que a solução hierárquica é mais eficiente do que o *Harmonic*, método bastante utilizado na solução de problemas mais robustos. A utilização do método guloso, sendo este comprovadamente não garantidor de soluções ótimas ([1]), em alguns pares estado-ação em detrimento da aprendizagem por reforço, faz com que a solução hierárquica também não garanta soluções ótimas. Espera-se verificar o desemepenho desta solução ao se incorporar métodos mais eficientes que o guloso, como por exemplo, o *Work Function* ([10]). Busca-se também realizar estudos acerca da convergência da abordagem hierárquica para valores ótimos da sua política, e ainda, estimar um parâmetro comparativo entre a abordagem hierárquica e a do *Q-Learning*. O uso de Redes Neurais para a interpolação dos valores da função Q também precisa ser verificada. As implicações do uso desta estratégia, caso a mesma seja viável, permitiria a interpolação dos valores de Q para pares estado-ação não visitados durante a execução da aprendizagem por reforço.

REFERÊNCIAS

[1] Borodin, A. and El-Yaniv, R., *Online Computation and Competitive Analysis*, Cambridge University Press, 1998.

[2] Albers, S., *Competitive On-Line Algorithms*, BRICS Lecture Series, University of Aarhus, 1996. Disponível em: http://www.uni-paderborn.de/fachbereich/AG/agmadh/WWW/english/scripts.html-Albers.

[3] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction*, The MIT Press, 1998.

[4] Bertsekas, D. P. and Tsitsiklis, J. N., *Neuro-dynamic Programming*, Athena Scientific, 1996.

[5] Manasse, M. S., McGeoch, L. A. and Sleator, D. D., *Competitive Algorithms for On-Line Problems*, In: Proc. 20th Annual ACM Symposium on Theory of Computing, pages 322-33, 1988.

[6] Yao, A C. C., *Probabilistic computations: Towards a unified measure of complexity*, In: Proc. 17th Annual IEEE Symposium on Foundations of Computer Science, pages 222-27, 1977.

[7] Bellman, R., *Dynamic Programming*, Princeton University Press, 1957.

[8] Watkins, C. J. C. H. and Dayan, P. *Q-Learning: Machine Learning*, Kluwer Academic Publishers, pages 279-92, 1992.

[9] Bartal, Y. and Grove, E., *The Harmonic K-Server Algorithm is competitive*. 23rd ACM Symposium on Theory of Computation, pages 260-266, 1991.

[10] Bartal, Y. and Koutsoupias, E., *On the Competitive Ratio of the Work Function Algorithm for the k-Server Problem*. Theor. Comput. Sci. 324(2-3): 337-345, 2004.

[11] Goldbarg, M. C. and Luna, H. P. C., *Otimização Combinatória e Programação Linear: Modelos e Algoritmos*, Editora Campus, 2000.

[12] Júnior, M. L. L., Melo, J. D. and Neto, A. D. D. *Utilização de Sistemas Inteligentes baseados em Aprendizagem por Reforço para a Otimização do Problema do Gerenciamento de Sondas de Produção Terrestre*. 3º Congresso Brasileiro de Pesquisa e Desenvolvimento de Petróleo e Gás, Salvador, Bahia, Brasil, 2005.

[13] Flatto, L., Calderbank, A. R. e Coffman, E. G., *Sequencing Problems in Two Server Systems*, Mathematics of Operation Research, 10, pp. 585-598, 1985.