UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

PROCESSOS DE DECISÃO DE *MARKOV*

PEDRO CORREIA FAGUNDES

JUIZ DE FORA
Dezembro, 2011

PROCESSOS DE DECISÃO DE *MARKOV*

PEDRO CORREIA FAGUNDES

Monografia de conclusão de curso apresentada ao Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora.
Orientador: Saul de Castro Leite

JUIZ DE FORA

Dezembro, 2011

PROCESSOS DE DECISÃO DE *MARKOV*

PEDRO CORREIA FAGUNDES

Monografia de conclusão de curso apresentada ao Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora, na área de otimização, como requisito à obtenção do título de Bacharel em Ciência da Computação.

Aprovado em: __/__/____

_____
Prof. Dr. Sc. Saul de Castro Leite(Orientador) – UFJF

_____
Prof. Dr. Sc. Carlos Cristiano Hasenclever Borges – UFJF

_____
Prof. Dr. Sc. Raul Fonseca Neto – UFJF

Dedico este trabalho aos meus pais, Fernando e Lídia, responsáveis por esta conquista. A Hortência por toda a ajuda, amor, apoio, carinho e compreensão.

AGRADECIMENTO

Agradeço aos meus irmãos, Guilherme e Fernando, pelo carinho e apoio incondicionais.

Agradeço aos meus professores por terem contribuído vastamente para minha formação acadêmica.

Agradeço aos meus amigos, pelo companheirismo durante esses anos de faculdade e também por terem ajudado na construção desse trabalho.

Agradeço ao Prof. Orientador Saul Leite pelos conhecimentos transmitidos e também pela paciente e dedicada orientação.

RESUMO

Este projeto tem como finalidade apresentar o estudo realizado sobre os processos de decisão de *Markov* que permitem a resolução de problemas em ambientes de incerteza de uma forma a maximizar a recompensa esperada ou minimizar o custo esperado. A motivação para a escolha do tema é oriunda de problemas de otimização em sistemas de filas, onde foi realizado um estudo em que processos de decisão de *Markov* foram utilizados em sua resolução. Além disso, o tema abordado mostrou-se muito útil em uma grande diversidade de áreas, podendo ser aplicado para a resolução de problemas de controle, selecionando as ações a serem tomadas nesses ambientes. Espera-se que, com esse trabalho, sejam mostrados diversos exemplos de problemas em diferentes áreas em que se podem aplicar os processos de decisão de *Markov* juntamente com os resultados obtidos através da utilização desses processos, demonstrando-se então a grande utilidade destes processos.

Palavras-chave: Processos de Decisão de Markov.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

Tomar decisões em sequência é uma atividade que pode ser de grande importância, principalmente se o ambiente onde essas decisões estão sendo tomadas é de incerteza.

Um exemplo deste tipo de ambiente seria o sistema de navegação de um robô onde este conta somente com sensores imprecisos para decidir a qual direção deve ir (SMITH AND SIMMONS, 2005). Outros exemplos são sistemas de ajuda online, onde ao mesmo tempo em que o sistema tenta ajudar o usuário passo a passo na tomada de decisões, ele tenta traçar o perfil do usuário e determinar o que ele realmente quer (HUI AND BOUTILIER, 2006); a mudança no preço de serviços ou produtos comercializados por uma empresa, assim como a decisão a respeito de promoções desses produtos (PINEAU AND THRUN, 2001).

Esses são exemplos de situações onde as decisões são tomadas e não há certeza sobre o resultado de cada ação realizada.

Este projeto objetiva estudar os processos de decisão de *Markov*, que são processos utilizados para encontrar a solução, ou uma sequência de ações ideal, de situações onde é necessário executar ações em sequência em ambientes de incerteza. Essa incerteza se encontra no resultado das ações executadas nesse ambiente.

Um sistema que pode ser modelado como um processo de decisão de *Markov* é composto por vários estados, ações, que podem modificar o estado atual do sistema, e a possibilidade de perceber o resultado de cada ação executada. Resolver esse problema significa encontrar uma política ótima que maximize a recompensa esperada ou minimize o custo esperado em cada execução de uma ação.

O interesse pelo estudo dos processos de decisão de *Markov* surgiu a partir de estudos de métodos numéricos para a resolução de problema de controle de modelos de sistemas de fila que operam no limite de sua saturação. Esse estudo foi desenvolvido através de um projeto intitulado “Métodos Numéricos para a Solução de Problemas de Controle Ótimo Estocástico a Tempo Contínuo”. Nesse projeto, procurou-se construir controles ótimos utilizando o Método da Cadeia de *Markov* Aproximada. Esse método tem como base uma discretização do espaço de estados para tratar o problema de controle como um processo de decisão de *Markov*.

A partir do estudo realizado nesse projeto, surgiu o interesse maior pelos processos de decisão de *Markov*, ou seja, a capacidade de resolver diversos tipos de

problemas, como os já citados anteriormente, além de sistemas de comunicação de alto desempenho, sistemas de controle de estoque, etc, que demonstram a grande área de atuação em que as soluções geradas pelos processos de decisão de *Markov* podem alcançar.

O presente trabalho tem como objetivo desenvolver algoritmos que solucionam problemas encontrados em ambientes onde há incerteza durante a evolução do sistema. Alguns problemas de temas diferentes serão resolvidos ao longo do trabalho, mostrando o potencial de resolução e a amplitude de atuação dos processos de decisão de *Markov.*

O estudo será dividido em três partes, onde a primeira parte fará uma breve introdução sobre os conceitos de cadeias de *Markov* e introduzirá os processos de decisão de *Markov*. A segunda parte tratará sobre processos de decisão de *Markov* em ambientes com horizontes finitos e a terceira parte trabalhará com ambientes com horizontes infinitos.

Serão abordados temas como a teoria, algoritmos e alguns problemas que podem ser resolvidos utilizando os processos de decisão de *Markov* para ajudar na tomada das decisões, maximizando a recompensa esperada ou reduzindo o custo esperado nas ações executadas em cada estado do ambiente modelado.

Para alcançar o objetivo geral desse trabalho serão desenvolvidos os algoritmos de programação dinâmica para tratar os ambientes com horizonte finito e os algoritmos de Iteração de Valores e Iteração de Políticas, que são algoritmos utilizados para a resolução de problemas onde o horizonte de estados é infinito nos processos de decisão de *Markov*.

É esperado como resultado neste trabalho que as soluções dos problemas estudados sejam encontradas, além de mostrar a amplitude do campo de estudo onde os processos de decisão de *Markov* podem atuar solucionando problemas e encontrando uma política ótima maximizando a recompensa esperada ou diminuindo o custo esperado.

2. CADEIAS DE *MARKOV*

Processos estocásticos são modelos para sistemas que envolvem com o tempo de forma probabilística. Um processo estocástico é definido como uma coleção de variáveis aleatórias *(X(t))* indexadas por parâmetro *t* pertencente a um conjunto *T*. *X(t)* pode ser uma quantidade de produto em um armazém ao longo do ano *t*, ou o número de pedidos de impressão para uma impressora ao longo do dia, por exemplo.

Os processos estocásticos podem ser classificados de acordo com o estado ou de acordo com o tempo.

São ditos de estado discreto (cadeia) quando *X(t)* é definido sobre um conjunto enumerável ou finito e são ditos de estado contínuo (sequência) quando *X(t)* é definido sobre um conjunto não enumerável.

De acordo com o tempo os processos estocásticos podem ser definidos como de tempo discreto, quando *t* é finito ou enumerável, e de tempo contínuo, quando *t* não é enumerável ou é infinito.

Seja *T* um conjunto discreto e X_t toma valor em um conjunto discreto *E* um processo estocástico possui a propriedade de *Markov* se

$$P(X_{t+1} = j | X_t = i, X_0 = i_0, X_1 = i_1, ..., X_{t-1} = i_{t-1}) = P(X_{t+1} = j | X_t = i)$$

para todo $j, i, i_0, ..., i_{t-1} \in E$ e tempo *t*. Ou seja, um processo estocástico é dito ser um Processo *Markoviano* se o estado futuro depende apenas do estado presente e não dos estados passados.

As probabilidades de transição representam a probabilidade do estado X_t ser *j* no instante *t* dado que o estado X_{t-1} é *i* no instante *t - 1*.

$$P(X_t = j | X_{t-1} = i) = p_t(j|i)$$

É comum as probabilidades de transição serem escritas de forma matricial:

$$P_t = \begin{pmatrix} p_t(0|0) & p_t(1|0) & p_t(2|0) & p_t(3|0) & \dots \\ p_t(1|0) & p_t(1|1) & p_t(2|1) & p_t(3|1) & \dots \\ p_t(0|2) & p_t(1|2) & p_t(2|2) & p_t(3|2) & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots \end{pmatrix}$$

A matriz de transição também pode ser dada por um diagrama de transições de estados.

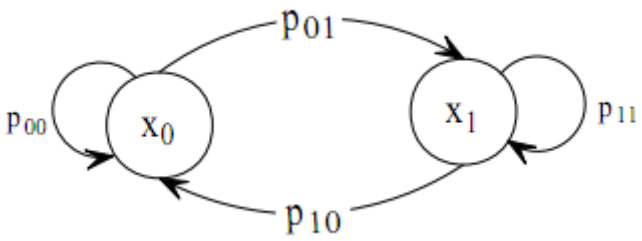

Figura 1. Diagrama da matriz de transições de estados de uma cadeia de Markov de dois estados.

Um processo *Markoviano* é dito ser uma Cadeia de *Markov* quando as variáveis aleatórias *X(t)* estão definidas em um espaço de estados discreto. Walrand (1988) define cadeia de *Markov* como: uma cadeia de *Markov* $x = \{x_t, t \geq 0\}$ descreve a evolução aleatória de uma partícula "amnésia" num conjunto contável. Isso significa que o processo *x* chegou ao presente estado x_i esquecendo o modo pelo qual chegou a esse estado e a chegada aos próximos estados também será feita sem recorrer aos estados anteriores.

2.1 EXEMPLO DE CADEIA DE *MARKOV*

Um jogo de apostas pode ser modelado como uma cadeia de *Markov* da seguinte forma:

- Jogador joga uma moeda e aposta em cara ou coroa;
- $P(X = 1) = p$ (probabilidade de acertar o lado da moeda);
- $P(X = 0) = 1 - p$ (probabilidade de errar o lado da moeda);
- O jogador ganha R$ 1 (1 real) ao acertar e perde 1 real ao errar o lado mostrado da moeda;
- O jogo encerra quando o jogador não tiver mais dinheiro ou quando tiver uma quantia igual à 3 reais;
- $E = \{0, 1, 2, 3\}$(espaço de estados da cadeia).

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1-p & 0 & p & 0 \\ 0 & 1-p & 0 & p \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

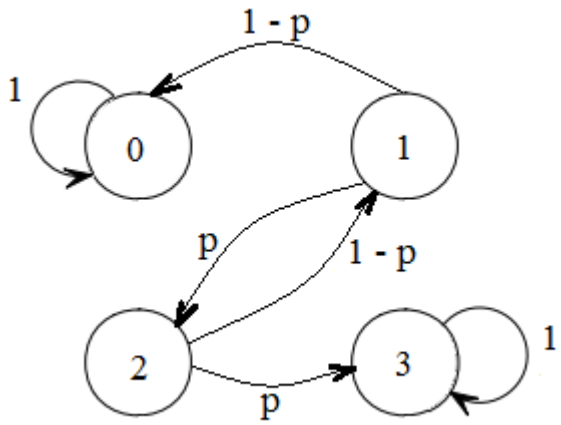

Figura 2. Diagrama da matriz de transições de estados de uma cadeia de Markov do exemplo citado anteriormente.

3. PROCESSOS DE DECISÃO DE *MARKOV*

Processos de decisão de *Markov* (MDP – *Markov Decision Process*) são utilizados para modelar e solucionar situações onde uma sequência de ações será executada em um ambiente onde não há certeza sobre o estado atual e o resultado da ação aplicada sobre esse estado.

O MDP auxilia na tomada de decisões em ambientes com incerteza e a sequência de ações realizadas nesse tipo de ambiente pode ser considerada de grande importância em algumas áreas.

Um exemplo prático seria um diagnóstico médico onde o profissional de saúde deve escolher uma ação conhecendo apenas os sintomas do paciente. Dentre essas ações estão exames, medicação, alta, internação, etc. Após a tomada de decisão não há como garantir a eficiência do tratamento, pois em alguns casos o tratamento escolhido resulta em cura, mas em outros casos, o resultado pode não ser o satisfatório. Somente após a tomada da decisão a influência do tratamento, positiva ou negativa, será conhecida pelo médico. (PELLEGRINI AND WAINER, 2003) (HAUSKRECHT, 2000).

O MDP procura determinar uma decisão que chegue mais próxima do resultado esperado sem a certeza do que vai acontecer após a tomada da decisão. No exemplo descrito esse resultado seria a cura do paciente.

O MDP trata problemas desta natureza onde o resultado de uma decisão tomada não depende de como a sequência de decisões chegou ao estado atual, e sim do estado atual e da ação escolhida.

Processos onde as transições entre estados são probabilísticas, há a possibilidade de observar em que estado o processo está e é possível interferir de forma periódica no processo executando ações, podem ser modelados utilizando MDP.

Dependendo do estado em que o processo se encontra, a ação realizada pode gerar uma recompensa, ou um custo. Essa recompensa, ou custo, pode ser definida também para os estados, independente da ação executada.

Dada a descrição de um problema que pode ser modelado como um processo de decisão de *Markov*, a resolução desse problema reside no fato de encontrar uma política ótima onde a cada momento ou estado, será determinada uma ação que torna máxima a recompensa esperada ou mínimo o custo esperado (PUTERMAN, 1994) (BERTSEKAS, 2001).

3.1 DEFINIÇÃO DOS PROCESSOS DE DECISÃO DE *MARKOV*

Um MDP é uma tupla (S, A, p_t, r_t), onde:

- S é um conjunto de estados em que o processo pode estar;
- A é um conjunto de ações que podem ser executadas em diferentes épocas de decisão;
- $p_t : S \times A \times S \rightarrow [0 ; 1]$ é uma função que dá a probabilidade de o sistema passar para um estado $s' \in S$, dado que o processo estava em um estado $s \in S$ e o agente decidiu executar uma ação $a \in A$ (denotada $p_t(s' \mid s, a)$);
- $r_t : S \times A \rightarrow R$ é uma função que dá o custo (ou recompensa) por tomar uma decisão $a \in A$ quando o processo está em um estado $s \in S$.

Os conjuntos de estados (S) e ações (A) podem ser finitos ou infinitos.

3.2 DEFINIÇÃO DE HORIZONTE EM PROCESSOS DE DECISÃO DE *MARKOV*

O número de épocas de decisão disponíveis para a tomada de decisões de um MDP é chamado horizonte. O horizonte pode ser finito, infinito ou indefinido, dependendo do número de decisões a tomar.

Caso o número de decisões a tomar seja fixo, então o horizonte é finito. Se não existe possibilidade de parada então o horizonte é infinito. Se o processo para apenas alcançando algum estado que foi definido como estado final do processo, então o horizonte é também infinito.

3.3 POLÍTICA PARA UM PROCESSO DE DECISÃO DE *MARKOV*

Um ambiente que pode ser modelado como um processo de decisão de *Markov* pode ser descrito da seguinte forma: a cada estado em que se encontra o sistema, o tomador de decisões verifica esse estado consultando uma política e após essa consulta executa uma ação. A ação pode modificar o estado atual. A partir dessa ação, o ambiente encontra-se em um novo estado. Esse novo estado é verificado novamente pelo tomador de decisões para que a próxima decisão seja tomada.

Uma regra de decisão deve ser utilizada pelo tomador de decisão a cada época de decisão. Todas as regras de decisão, juntas, formam uma política π. Uma regra de decisão é

uma função $d_k : S \rightarrow A$, que irá determinar a ação que será executada em um estado do sistema, onde k é uma época de decisão. A política é formada por uma sequência de regras de decisão $\pi = \{d_0, d_1, ..., d_{N-1}\}$, para cada época de decisão, onde N é o horizonte.

O objetivo principal é encontrar uma política que seja ótima de acordo com o critério de desempenho das decisões.

A classificação da política de acordo com as épocas de decisão pode ser definida de acordo com a dependência da ação com a época de decisão, ou seja, se a ação recomendada não depender da época de decisão então a política é estacionária, caso contrário a política é não-estacionária.

Em uma política estacionária a política pode ser usada como um sinônimo de decisão. As políticas estacionárias ou não-estacionárias podem ser utilizadas para horizonte finito ou infinito. O uso comum é de políticas estacionárias para horizonte infinito e de políticas não-estacionárias para horizonte finito.

As políticas podem ser classificadas também como determinísticas, quando cada estado é mapeado em uma única ação, e não-determinística quando um estado é mapeado em um conjunto de ações, sendo que cada ação tem uma probabilidade de ser escolhida.

Caso a escolha da ação dependa apenas do estado atual, então a política é *Markoviana*. Se a ação depende de todo o histórico de ações e estados do sistema até o momento da tomada de decisão então ela é não-*Markoviana*.

Em cada época de decisão o tomador de decisões receberá uma recompensa na execução de uma política. Essa recompensa pode ser utilizada para comparar duas políticas. Os critérios mais utilizados para essa comparação são:

- A recompensa média por época de decisão;
- A recompensa esperada total;
- A recompensa esperada descontada.

O objetivo de um problema de tomada de decisão sequencial é encontrar a política π que maximize as recompensas esperadas ou minimize os custos esperados.

3.4 PROGRAMAÇÃO DINÂMICA

A programação dinâmica (*Dynamic Programming – DP*) é amplamente reconhecida como a única solução para problemas de tomada de decisão sequencial que é ao mesmo tempo bem fundamentada teoricamente e viável computacionalmente (BARRETO, 2008). A DP

oferece um algoritmo mais barato que a simples enumeração de todas as possibilidades de um problema combinatório, o que, em alguns casos, se torna impraticável ou extremamente caro.

O início da DP, data dos anos 50, época da publicação dos primeiros artigos de Bellman, que introduziram essa teoria. Desde então a DP torna-se uma disciplina de interesse maior em diversas áreas como a matemática aplicada, investigação operacional, engenharia, economia, gestão, entre muitas outras.

A programação dinâmica se baseia fortemente no conceito de função de valor, que reflete o desempenho de uma política de decisão em um horizonte de tempo potencialmente infinito (BARRETO, 2008).

O conceito de uma função de valor é um conceito absolutamente imprescindível para a teoria da programação dinâmica. A cada estado s_i do espaço S a função de valor de uma política π associa o valor esperado da série de recompensas recolhidas por π a partir desse estado: $V^{\pi} : S \rightarrow R$.

4. PROCESSOS DE DECISÃO DE *MARKOV* COM HORIZONTE FINITO

Neste capítulo serão apresentados alguns itens sobre os processos de decisão de *Markov* com horizonte finito. Dentre os assuntos discutidos terão conceitos básicos, algoritmos e problemas resolvidos com este modelo.

4.1 CRITÉRIOS DE OTIMALIDADE

Como resultado da escolha e implementação de uma política, o tomador de decisões recebe recompensas em períodos 1, ..., N. O tomador de decisão deverá escolher uma política que tenha a melhor sequência possível de recompensas e, para isso, ele necessita de um método de comparação para as sequências de recompensas que, antes de implementar a política, são desconhecidas para o tomador de decisão. Algumas abordagens para comparação de recompensas serão discutidas.

Um dos modelos de comparações existentes leva em conta a utilidade esperada. Para MDP de horizonte finito e espaço discreto a utilidade esperada da política π pode ser representada por:

$$E^{\pi}[\psi(R)] = \sum_{(p_1,\dots,p_N)\epsilon\, IR} \psi(p_1,\dots,p_N) P^{\pi}\{(p_1,\dots,p_N)\}.$$

Onde π é uma política, P^{π} são as probabilidades de ocorrência de uma certa sequência utilizando a política π, N é o espaço de tempo onde as decisões serão tomadas e $\psi(R)$ é a utilidade de um vetor de recompensas R.

Sob esse critério o tomador de decisão prefere a política π sobre a política v se:

$$E^{\pi}[\psi(R_1,\dots,R_N)] > E^{v}[\psi(R_1,\dots,R_N)].$$

E é equivalente quando:

$$E^{\pi}[\psi(R_1,\dots,R_N)] = E^{v}[\psi(R_1,\dots,R_N)].$$

Pode-se também considerar preferências temporais incluindo um fator de desconto nesta expressão.

Como uma alternativa aos critérios de utilidade esperada, o tomador de decisão pode usar um critério que leva em conta explicitamente tanto o valor total esperado quando a variância da sequência de recompensas. Como consequência do uso de tal critério, ele pode renunciar a política com a maior recompensa esperada total porque a sua variância é muito grande.

De acordo com o critério de recompensa total esperada, seja $v_N^{\pi}(s)$ a representação da recompensa total esperada sobre o horizonte de tomada de decisões se a política π é usada e o sistema está no estado *s* na primeira época de decisão, $\pi \epsilon \Pi$ é definido por:

$$v_N^{\pi}(s) \equiv E_s^{\pi} \left\{ \sum_{t=1}^{N-1} r_t(X_t, Y_t) + r_N(X_N) \right\}.$$

Para contabilizar o tempo das recompensas, muitas vezes é introduzido um fator de desconto. Fator de desconto é um valor escalar λ, $0 < \lambda < 1$, que mede o valor no tempo *n* de uma recompensa recebida no tempo $n + 1$. Uma recompensa que soma valores de períodos *t* no futuro tem o valor de λ presente.

Teorias e algoritmos de MDP para horizonte finito se preocupam principalmente em encontrar uma política que tenha a maior recompensa total esperada. A política encontrada que retorna o maior valor para uma recompensa esperada é chamada de política ótima.

Um problema de decisão de *Markov* é utilizado juntamente com um critério de otimalidade para um processo de decisão de *Markov*. Os problemas abordados nesse capítulo são de tempo discreto e horizonte finito e o critério de otimalidade utilizado é o da recompensa total esperada.

4.2 AVALIAÇÃO DE POLÍTICAS

Os problemas modelados como processos de decisão de *Markov* para horizontes finitos são resolvidos através da programação dinâmica utilizando uma indução de trás para frente para avaliar recursivamente a recompensa total esperada.

Seja $\pi = (d_1, d_2, ..., d_{N-1})$ uma sequência de decisões tomadas, ou seja, uma política. $u_t^{\pi}: H_t \rightarrow R$ representa a recompensa total esperada usando a política π nas épocas de decisão

$t, t+1, ..., N-1$, onde H_t é o espaço das histórias do processo até o tempo t. Se o histórico de decisões na época t é $h_t \in H_t$, então u_t^π para $t < N$ é definida por:

$$u_t^\pi(h_t) = E_{h_t}^\pi\left\{\sum_{n=t}^{N-1} r_n(X_n, Y_n) + r_N(X_N)\right\}, \qquad (4.2.1)$$

onde u_t^π inclui recompensas sobre todas as épocas futuras.

Para mostrar como avaliar indutivamente u_t^π utilizando o algoritmo de avaliação de políticas para horizonte finito, vamos supor que $\pi \in \Pi$ seja determinístico e que S seja discreto.

O algoritmo de avaliação de políticas para horizonte finito é descrito a seguir:

1. Considere $t = N$ e $u_N^\pi(h_N) = r_N(s_N)$ para todo $h_N = (h_{N-1}, a_{N-1}, s_N) \in H_N$.
2. Se $t = 1$, pare, senão vá para o passo 3.
3. Substitua $t - 1$ por t e calcule $u_t^\pi(h_t)$ para cada $h_t = (h_{t-1}, a_{t-1}, s_t) \in H_t$ com:

$$u_t^\pi(h_t) = r_t\big(s_t, d_t(h_t)\big) + \sum_{j \in S} p_t\big(j \big| s_t, d_t(h_t)\big) u_{t+1}^\pi(h_t, d_t(h_t), j), \qquad (4.2.2)$$

para cada $(h_t, d_t(h_t), j) \in H_{t+1}$.

4. Retorne para o passo 2.

O valor esperado da política π ao longo dos períodos $t, t+1, ..., N$, quando o histórico de recompensas na época t é h_t, é dado por $r_t(s_t, d_t(h_t))$ mais a recompensa esperada ao longo dos períodos restantes. O segundo termo da equação (4.2.2) contém o produto da probabilidade de estar no estado j na época de decisão $t+1$ usando a ação $d_t(h_t)$ e a recompensa total esperada obtida usando a política π ao longo dos períodos $t+1, ..., N$, quando o histórico de recompensas na época $t+1$ é $h_{t+1} = (h_t, d_t(h_t), j)$. A soma sobre todos os j possíveis dá a expectativa desejada expressa em termos de u_{t+1}^π, ou seja:

$$u_t^\pi(h_t) = r_t\big(s_t, d_t(h_t)\big) + E_{h_t}^\pi\{u_{t+1}^\pi(h_t, d_t(h_t), X_{t+1})\}.$$

Este esquema indutivo reduz o problema de computação das recompensas esperadas totais ao longo de N períodos a uma sequência de $N - 1$ passos sobre um período

(PUTERMAN, 2005). Essa redução de cálculos na computação é a essência da programação dinâmica.

"Problemas de múltiplos estágios podem ser resolvidos através de uma análise de sequência simples de fases únicas de problemas definidas indutivamente." (PUTERMAN, 2005).

Ao utilizar essa essência da programação dinâmica a pesada tarefa de determinar explicitamente a distribuição de probabilidade conjunta de histórias em π é evitada. Um procedimento para encontrar políticas ótimas que generaliza esses cálculos é descrito a seguir.

A prova de que esse algoritmo faz a otimização proposta por ele é feita através de induções. Contrariamente à convenção matemática usual, a indução aplicada é para trás, ou seja, realizada do final para o inicio. O desenvolvimento da prova abaixo foi retirado de (PUTERMAN, 2005) página 81.

Considerando t o índice de induções, π e supondo que u_t^π, $t \leq N$, foi gerado pelo algoritmo de avaliação de política. Então, para todo $t \leq N$, a equação (4.2.1) é verificada e $v_N^\pi(s) = u_1^\pi(s)$ para todo $s \in S$.

O resultado é obviamente verdadeiro quando $t = N$. Suponhamos agora que essa afirmação seja verdadeira para $t + 1, t + 2, \ldots, N$. Em seguida, usando

$$u_t^\pi(h_t) = r_t\big(s_t, d_t(h_t)\big) + E_{h_t}^\pi\{u_{t+1}^\pi(h_t, d_t(h_t), X_{t+1})\}$$

e a hipótese de indução

$$u_t^\pi(h_t) = r_t\big(s_t, d_t(h_t)\big) + E_{h_t}^\pi\left[E_{h_{t+1}}^\pi\left\{\sum_{n=t+1}^{N-1} r_n(X_n, Y_n) + r_N(X_n)\right\}\right]$$
$$= r_t\big(s_t, d_t(h_t)\big) + E_{h_t}^\pi\left\{\sum_{n=t+1}^{N-1} r_n(X_n, Y_n) + r_N(X_n)\right\}.$$

Deste s_t e h_t são conhecidos na época de decisão t, $X_t = s_t$, de modo que o primeiro termo da equação anterior pode ser incluído dentro da expectativa de trazer os resultados esperados.

Quando se toma uma expectativa condicional sobre a história até o momento $t + 1$, uma X_t é conhecida, mas quando se toma expectativa condicional em relação à história até o tempo t, uma X_t é aleatória. Assim, uma vez que o algoritmo avalia $u_{t+1}^\pi(h_{t+1})$ para todo h_{t+1}, antes de avaliar u_t^π a expectativa sobre X_{t+1} pode ser computada.

Uma característica chave deste algoritmo é que a indução é para trás. Isto é necessário devido à natureza probabilística do MDP. Se o modelo for determinístico, políticas poderiam ser avaliadas por qualquer indução, tanto para frente quanto para trás, porque as expectativas não precisam ser computadas. Para avaliar uma expectativa condicional da decisão diante da época *t*, é exigida a expectativa condicional da época de decisão *t + 1* em diante para cada possibilidade de história surgida a partir da época de decisão *t + 1*. Proceder de forma indutiva para frente exigiria uma enumeração de todos os históricos de decisão, e derivar a distribuição de probabilidade de cada política inferior de π.

4.3 EQUAÇÕES DE OTIMALIDADE E O PRINCÍPIO DA OTIMALIDADE

Nesta seção será mostrado que as equações de otimalidade correspondem às funções de valor ótimo e que elas também fornecem uma base para determinar políticas ótimas.

Considere que

$$u_t^*(h_t) = \sup_{\pi \in \Pi} u_t^{\pi}(h_t)$$

denota o supremo sobre todas as políticas da recompensa total esperada de decisão diante da época *t*, quando o histórico de decisões até o instante *t* é h_t. Uma vez que h_t é conhecido, será considerado apenas porções de políticas de decisão diante da época *t*, ou seja, é necessário apenas o mais supremo $(d_t, d_{t+1}, ..., d_{N-1}) \in D_t \times D_{t+1} \times ... \times D_{N-1}$. Quando deseja-se minimizar o custo em vez de maximizar a recompensa, u_t^* é referido como uma função *cost-to-go*.

As equações de otimalidade são dadas por

$$u_t^*(h_t) = \sup_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(h_t, a, j) \right\}$$

para $t = 1, ..., N-1$ e $h_t = (h_{t-1}, a_{t-1}, s_t) \in H_t$. Para $t = N$ a condição $u_N(h_N) = r_N(s_N)$ é adicionada para $h_N = (h_{N-1}, a_{N-1}, s_N) \in H_N$.

As operações de "sup" são implementadas para avaliar a quantidade entre chaves para cada $a \in A_{s_t}$ e escolhendo o supremo sobre todos esses valores. Quando A_{s_t} é contínuo, o

supremo pode ser encontrado analiticamente. Se o supremo é atingido, por exemplo, quando A_{s_t} é finito, ele pode ser substituído por "max" e torna-se a seguinte equação:

$$u_t(h_t) = \max_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}(h_t, a, j) \right\}.$$

As equações de otimalidade são ferramentas fundamentais na teoria da decisão de *Markov*, tendo as seguintes propriedades importantes e úteis:

a. Soluções de equações de otimalidade são retornos ótimos a partir do período *t* progressivamente para cada *t*.
b. Elas fornecem um método para determinar se uma política é a ideal. Se a recompensa esperada total da política π a partir do período *t* satisfaz este sistema de equações para *t = 1, ..., N*, então essa política é ótima.
c. Elas são a base para um procedimento eficiente para o calculo de políticas ótimas.
d. Elas podem ser usadas para determinar as propriedades estruturais das políticas ótimas e as funções de retorno.

Algoritmos de programação dinâmica são obtidos através de modificações sobre equações de Bellman, ou seja, partem de regras de atualização para melhorar aproximações das funções valor desejadas. Eles podem ser vistos como uma coevolução entre uma função de valor e uma política, que competem localmente, mas cooperam uma com a outra em uma escala maior.

O objetivo da otimização procurada com a utilização da programação dinâmica é determinar a política ótima que otimize a função objetivo global do sistema nas suas *n* etapas. Em cada estágio de decisão é possível definir o estado da solução. O estado é o ponto de situação em que se pode estar como consequência da decisão tomada no estágio anterior. Em cada estágio decide-se, para cada estado, qual o estado do estágio seguinte que oferece melhor retorno para a solução do problema.

"Para um dado estado do sistema, a política ótima para os restantes estados é independente da política de decisão adotada em estados anteriores." (BELLMAN, 1959).

A partir do conceito de função de valor que Bellman estabeleceu uma relação recursiva entre os estados de um processo de decisão de *Markov* que ficou conhecida como a *equação de Bellman.*

A equação de Bellman é derivada do princípio de otimalidade também formulado por Bellman: “De qualquer ponto de uma trajetória ótima, a trajetória remanescente é ótima para o problema correspondente que se inicia naquele ponto e termina no mesmo ponto do problema anterior.” (BELLMAN, 1959). Ela constitui o cerne dos algoritmos de programação dinâmica. É a equação de Bellman que torna a busca pela política ótima de um processo de decisão de *Markov* um processo computacionalmente viável.

A partir das equações de otimalidade conseguimos utilizar o princípio da otimalidade que é um resultado fundamental da programação dinâmica. De acordo com Bellman (1957) “Uma política ótima possui a propriedade que qualquer que seja o estado inicial e a decisão inicial, as decisões restantes devem constituir uma política ótima no que se refere ao estado resultante da primeira decisão.”.

4.4 INDUÇÃO RETROATIVA

A indução retroativa, ou indução para trás, fornece um método eficiente para resolver MDPs de horizonte finito e tempo discreto. Para problemas estocásticos, a enumeração e avaliação de todas as políticas é a única alternativa, mas a indução para frente e métodos de alcance fornecem métodos de soluções alternativos para sistemas determinísticos. Os termos “indução retroativa” e “programação dinâmica” são sinônimos, embora a expressão “programação dinâmica”, muitas vezes refere-se a todos os resultados e métodos para o processo de decisão sequencial. Esta seção apresenta o algoritmo de indução retroativa e mostra como usá-lo para encontrar políticas ótimas e as funções de valor. Esse algoritmo resolve equações de otimalidade.

O algoritmo de Indução Retroativa:

1. Faça $t = N$ e $u_N^*(s_N) = r_N(s_N)$ para todo $s_N \in S$,
2. Substitua $t - 1$ por t e compute $u_t^*(s_t)$ para cada $s_t \in S$ por

$$u_t^*(s_t) = \max_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\}.$$

Atribua

$$A^*_{s_t,t} = \underset{a \in A_{s_t}}{\arg max} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u^*_{t+1}(j) \right\}.$$

3. Se $t = 1$, pare. Caso contrário, retorne ao passo 2.

O seguinte teorema representa uma declaração formal das propriedades do algoritmo de indução retroativa.

Teorema: Suponha que u^*_t, $t = 1, ..., N$ e $A^*_{s_t,t}$ satisfazem a primeira e a segunda equação do passo 2 do algoritmo de indução retroativa, então,

a. Para $t = 1, ..., N$ e $h_t = (h_{t-1}, a_{t-1}, s_t)$

$$u^*_t(s_t) = \sup_{\pi \in \Pi} u^\pi_t(h_t),$$

$s_t \epsilon S$.

b. Seja $d^*_t(s_t) \epsilon A^*_{s_t,t}$ para todo $s_t \epsilon S$, $t = 1, ..., N - 1$, e seja $\pi^* = (d^*_1, ..., d^*_{N-1})$. Então $\pi^* \epsilon \Pi$ é ótimo e satisfaz

$$v^{\pi^*}_N(s) = \sup_{\pi \in \Pi} v^\pi_N(s),$$

$s \epsilon S$ e

$$u^{\pi^*}_t(s_t) = u^*_t(s_t),$$

$s_t \epsilon$ S para $t = 1, ..., N$.

As propriedades do algoritmo de indução retroativa são:

a. Para $t = 1, 2, ..., N - 1$, que encontra conjuntos $A^*_{s_t,t}$ que contém todas as ações em A_{s_t} que atinge o máximo em

$$u^*_t(s_t) = \max_{a \in A_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u^*_{t+1}(j) \right\}.$$

b. Ele avalia qualquer política que seleciona uma ação em $A^*_{s_t,t}$ para cada $s_t \epsilon S$ para todo $t = 1, 2, ..., N - 1$.

c. Ele calcula a recompensa total esperada para o horizonte de tomada de decisão inteiro, e de cada período para o final do horizonte de qualquer política ideal.

4.5 PROBLEMAS

Nesta seção o algoritmo de indução retroativa será utilizado para encontrar políticas ótimas para alguns problemas de decisão de *Markov* com horizontes finitos. Lembrando que u_t^*, *t = 1, 2, ..., N* denotam soluções de equações de otimalidade.

4.5.1 O PROBLEMA DO MODELO DE INVENTÁRIO ESTOCÁSTICO

Nesta seção, será considerado um problema de controle de estoque retirado de (PUTTERMAN, 2005).

A cada mês, o gerente de um armazém determina o inventário atual (estoque atual) de um único produto. Com base nessas informações, ele decide se quer ou não fazer um pedido a um fornecedor para adicionar produtos ao estoque. Ao fazer esse pedido, ele se depara com uma troca entre o custo associado à realização do inventário e as vendas perdidas ou penalidades associadas com a incapacidade de satisfazer a demanda do cliente para o produto. O objetivo do gerente é maximizar alguma medida de lucro (receita de vendas menos os custos da exploração e ordenação do estoque) ao longo do horizonte de decisão de *Markov*. A demanda para o produto é aleatória com uma distribuição de probabilidade conhecida.

Um modelo foi formulado sob o seguinte conjunto de hipóteses:

1. A decisão de ordenar ações de adição de produtos é feita no início de cada mês e a entrega ocorre instantaneamente.
2. A demanda para o produto chega ao longo do mês, mas todos os pedidos são preenchidos no último dia do mês.
3. Se a procura exceder o estoque, o cliente vai em outros lugares para comprar o produto, isto é, não há registro de volta (*backlogging*) de ordens não preenchidas para que a demanda em excesso seja perdida.
4. As receitas, custos, e a distribuição da demanda não variam de mês para mês.
5. O produto é vendido apenas em unidades inteiras.
6. O armazém tem capacidade para *M* unidades do produto.

Seja s_t o estoque atual no início do mês t, a_t o número de unidades encomendadas pelo gerente de estoque no mês t e D_t a demanda aleatória no mês t. A demanda tem uma conhecida distribuição de probabilidade de tempo homogêneo $p_j = P\{D_t = j\}, j = 0, 1, 2, ...$ O estoque na época de decisão $t + 1$, s_{t+1}, está relacionado com o estoque na época de decisão t, s_t, através da seguinte equação:

$$s_{t+1} = \max\{s_t + a_t - D_t, 0\} \equiv [s_t + a_t - D_t]^+.$$

Porque *backlogging* não é permitido, já que o nível de estoque não pode ser negativo. Assim, sempre que $s_t + a_t - D_t < 0$, o nível do estoque na época de decisão posterior é 0.

Valores, em dinheiro, serão expressos no início dos meses. Esses valores serão referidos como valores presentes. O valor presente do custo de ordenar unidades u em qualquer mês é $O(u)$. Ele é composto por um custo fixo $K > 0$ para encomendas e um custo variável $c(u)$, que aumenta de acordo com a quantidade encomendada. Daí:

$$O(u) = \begin{cases} K + c(u) & se\ u > 0 \\ 0 & se\ u = 0 \end{cases}.$$

O valor presente do custo de manter um estoque de unidades u por um mês (entre o recebimento da ordem e a liberação do estoque para atender a demanda) é representado pela função não decrescente $h(u)$. Em problemas de horizonte finito, o estoque remanescente na última época de decisão tem valor $g(u)$. Finalmente, se a demanda for j unidades e tiver estoque disponível para atender a demanda, o gerente recebe a receita com o valor presente $f(j)$. Assumindo que $f(0) = 0$.

Neste modelo a recompensa depende do estado do sistema na época de decisão posterior, ou seja,

$$r_t(s_t, a_t, s_{t+1}) = -O(a_t) - h(s_t + a_t) + f(s_t + a_t - s_{t+1}).$$

Para fazer com que a recompensa dependa apenas de s_t e a_t a função $F_t(u)$ é definida. $F_t(u)$ computa o valor presente esperado (no início do mês t) da receita recebida no mês t, quando o estoque antes do recebimento de pedidos de clientes é u unidades. Se o estoque u excede a demanda j, o valor presente da receita é $f(j)$. Isso ocorre com probabilidade p_j. Se a procura excede o estoque, o valor presente da receita é igual a $f(u)$. Isso ocorre com probabilidade $q_u = \Sigma_{j=u}^{\infty} p_j$. Assim

$$F(u) = \sum_{j=0}^{u-1} f(j)p_j + f(u)q_u.$$

A formulação do processo de decisão de *Markov* segue:

1. Épocas de decisão: $t = \{1, 2, ..., N\}$, $N \leq \infty$.
2. Estados (a quantidade de estoque atual no início de um mês): $S = \{0, 1, 2, ..., M\}$.
3. Ações (a quantidade de pedidos para ordenar o estoque no mês t): $A_s = \{0, 1, 2, ..., M - s\}$.
4. Recompensas esperadas (receita esperada menos custo de reposição do estoque e despesas):

$$r_t(s, a) = F(s + a) - O(a) - h(s + a),$$

$t = 1, 2, ..., N - 1$ (o valor terminal do estoque) $r_N(s) = g(s)$, $t = N$.

5. Probabilidades de transição:

$$p_t(j|s, a) = \begin{cases} 0 & se\ M \geq j > s + a \\ p_{s+a-j} & se\ M \geq s + a \geq j > 0 \\ q_{s+a} & se\ M \geq s + a\ e\ j = 0, \end{cases}$$

onde q_{s+a} é definida abaixo.

Se o estoque atual no início do período t é s unidades e um pedido é feito por a unidades, o estoque antes de a demanda externa é $s + a$ unidades. Um nível de estoque $j > 0$ no início do período $t + 1$ exige uma demanda de unidades de $s + a_j$ no período t. Isso ocorre com probabilidade p_{s+a-j}. Porque o *backlogging* não é permitido, se a demanda no período t excede $s + a$ unidades, então o estoque no início do período $t + 1$ é 0 unidades. Isso ocorre com probabilidade q_{s+a}. A probabilidade do nível de estoque exceder $s + a$ unidades é 0, pois a demanda é não-negativa. O estoque tem um nível sempre menor ou igual a M.

O estoque ao longo de um mês é $s + a$, de modo que o custo de manutenção mensal total é $h(s + a)$.

As regras de decisão atribuem a quantidade de estoque a ser encomendada a cada mês para cada nível inicial de estoque possível.

Valores foram fornecidos para a resolução desse problema: *K = 4*, *c(u) = 2u*, *g(u) = 0*, *h(u) = u*, *M = 3*, *N = 4*, *f(u) = 8u*, e

$$p_j = \begin{cases} \frac{1}{4} & se\ j = 0 \\ \frac{1}{2} & se\ j = 1 \\ \frac{1}{4} & se\ j = 2. \end{cases}$$

O modelo pode ser interpretado da seguinte forma. O estoque é limitado em 3 unidades. Para cada unidade pedida o custo é de 2 por unidade, cada unidade em estoque tem custo 1 e cada unidade vendida gera uma receita de 8. A receita esperada quando *u* unidades estão em estoque antes do recebimento de um pedido é: Se *u = 0* então *F(u) = 0*, se *u = 1* então *F(u) = 6*, se *u = 2* então *F(u) = 8* e se *u = 3* então *F(u) = 8*.

A política $u_t^*(s,a)$ é definida por:

$$u_t^*(s,a) = r(s,a) + \sum_{j \in S} p(j|s,a) u_{t+1}^*(j).$$

O algoritmo de indução retroativa foi implementado da seguinte forma:

1. Atribua *t = 4* e $u_4^*(s) = r_4(s) = 0$, *s = 0, 1, 2, 3*.
2. Atribua *t = 3* e

$$u_3^*(s) = \max_{a \in A_s} \left\{ r(s,a) + \sum_{j \in S} p(j|s,a) u_4^*(j) \right\}, para\ s = 0,1,2,3$$

$$= \max_{a \in A_s} \{r(s,a)\}$$

Inspecionando os valores de *r(s,a)* mostra que em cada estado a maximização é 0, ou seja, não há pedidos. Temos

s	$u_3^*(s)$	$A_{s,3}^*$
0	0	0
1	5	0

2	6	0
3	5	0

3. Atribua *t* = *2* e

$$u_2^*(s) = \max_{a \in A_s}\{u_2^*(s,a)\},$$

onde, por exemplo

$$u_2^*(0,0) = r(0,0) + p(0|0,0)u_3^*(0) + p(1|0,0)u_3^*(1) + p(2|0,0)u_3^*(2) + p(3|0,0)u_3^*$$

$$= 0 + 1 \times 0 + 0 \times 5 + 0 \times 6 + 0 \times 5 = 0$$

$$u_2^*(0,1) = r(0,1) + p(0|0,1)u_3^*(0) + p(1|0,1)u_3^*(1) + p(2|0,1)u_3^*(2) + p(3|0,1)u_3^*$$

$$= -1 + \left(\frac{3}{4}\right) \times 0 + \left(\frac{1}{4}\right) \times 5 + 0 \times 6 + 0 \times 5 = \frac{1}{4}$$

$$u_2^*(0,2) = r(0,2) + p(0|0,2)u_3^*(0) + p(1|0,2)u_3^*(1) + p(2|0,2)u_3^*(2) + p(3|0,2)u_3^*$$

$$= -2 + \left(\frac{1}{4}\right) \times 0 + \left(\frac{1}{2}\right) \times 5 + \left(\frac{1}{4}\right) \times 6 + 0 \times 5 = 2$$

$$u_2^*(0,3) = r(0,3) + p(0|0,3)u_3^*(0) + p(1|0,3)u_3^*(1) + p(2|0,3)u_3^*(2) + p(3|0,3)u_3^*$$

$$= -5 + 0 \times 0 + \left(\frac{1}{4}\right) \times 5 + \left(\frac{1}{2}\right) \times 6 + \left(\frac{1}{4}\right) \times 5 = \frac{1}{2}$$

As quantidades $u_2^*(s,a)$, $u_2^*(s)$ e $A_{s,2}^*$ são resumidos na seguinte tabela denotando como X as ações inviáveis.

	$u_2^*(s,a)$					
s	a = 0	a = 1	a = 2	a = 3	$u_2^*(s)$	$A_{s,2}^*$
0	0	$\frac{1}{4}$	2	$\frac{1}{2}$	2	2
1	$\frac{25}{4}$	4	$\frac{5}{2}$	X	$\frac{25}{4}$	0
2	10	$\frac{9}{2}$	X	X	10	0
3	$\frac{21}{2}$	X	X	X	$\frac{21}{2}$	0

Figura 3. Valores do MDP.

4. Atribua *t = 1* e

$$u_1^*(s) = \max_{a \in A_s}\{u_1^*(s,a)\}.$$

As quantidades $u_1^*(s,a)$, $u_1^*(s)$, e $A_{s,1}^*$ são resumidos na seguinte tabela.

	$u_1^*(s,a)$					
s	a = 0	a = 1	a = 2	a = 3	$u_1^*(s)$	$A_{s,1}^*$
0	2	$\frac{33}{16}$	$\frac{66}{16}$	$\frac{67}{16}$	$\frac{67}{16}$	3
1	$\frac{129}{16}$	$\frac{98}{16}$	$\frac{99}{16}$	X	$\frac{129}{16}$	0
2	$\frac{194}{16}$	$\frac{131}{16}$	X	X	$\frac{194}{16}$	0
3	$\frac{227}{16}$	X	X	X	$\frac{227}{16}$	0

Figura 4. Valores do MDP.

5. Se *t = 1*, pare.

Este algoritmo produz a total recompensa ótima esperada com a função $v_4^*(s)$ e a política ótima $\pi^* = (d_1^*(s), d_2^*(s), d_3^*(s))$ que é mostrada abaixo:

s	$d_1^*(s)$	$d_2^*(s)$	$d_3^*(s)$	$v_4^*(s)$
0	3	2	0	$\frac{67}{16}$
1	0	0	0	$\frac{129}{16}$
2	0	0	0	$\frac{194}{16}$
3	0	0	0	$\frac{227}{16}$

Figura 5. Política.

4.5.2 O PROBLEMA DO ROTEAMENTO

Nesta seção, foi retirado um exemplo de (PUTERMAN, 2005), onde o algoritmo de indução retroativa será aplicado para encontrar o caminho mais longo no modelo da figura seguinte. Neste modelo os estados correspondem a nós e conjuntos de ações correspondem ao conjunto de arcos de origem em cada nó. A ação *(i, j)* corresponde a escolher o arco entre os nós *i* e *j*.

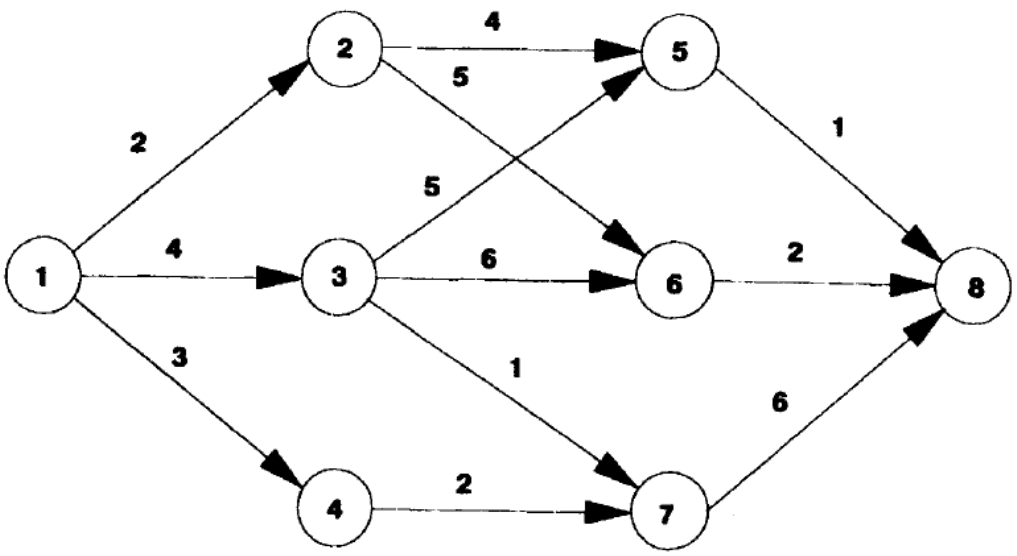

Figura 6. Rotas.

O algoritmo utilizado é descrito a seguir:

1. Atribua *t* = *4* e $u_4^*(8) = 0$.
2. Atribua *t* = *3* e

$$u_3^*(5) = 1 + u_4^*(8) = 1, u_3^*(6) = 2 + u_4^*(8) = 2,$$

$$u_3^*(7) = 6 + u_4^*(8) = 6.$$

Como não há apenas uma ação em cada estado,

$$A_{5,3}^* = (5,8), \qquad A_{6,3}^* = (6,8), \qquad A_{7,3}^* = (7,8).$$

3. Atribua *t = 2* e

$$u_2^*(2) = \max\{4 + u_3^*(5), 5 + u_3^*(6)\} = \max\{5,7\} = 7,$$

$$u_2^*(3) = \max\{5 + u_3^*(5), 6 + u_3^*(6), 1 + u_3^*(7)\} = \max\{6,8,7\} = 8,$$

$$u_2^*(4) = 2 + u_3^*(7) = 8.$$

As ações ideais seriam

$$A_{2,2}^* = (2,6), A_{3,2}^* = (3,6), A_{4,2}^* = (4,7).$$

4. Atribua *t = 1* e

$$u_1^*(1) = \max\{2 + u_2^*(2), 4 + u_2^*(3), 3 + u_2^*(4)\} = \max\{9,12,11\} = 12.$$

A ação ótima seria

$$A_{1,1}^* = (1,3).$$

5. Se *t = 1*, pare.

Uma vez que a ação ótima fixada em cada estado é única, existe uma política única ideal $\pi^* = (d_1^*, d_2^*, d_3^*)$ onde

$$d_1^*(1) = (1,3),$$

$$d_2^*(2) = (2,6), \qquad d_2^*(3) = (3,6), \qquad d_2^*(4) = (4,7),$$

$$d_3^*(5) = (5,8), \qquad d_3^*(6) = (6,8), \qquad d_3^*(7) = (7,8).$$

Esta política seleciona qual arco deve ser usado para atravessar cada nó, a fim de percorrer o caminho mais longo a partir desse nó ao nó 8. A partir do nó 1, o caminho ideal é 1→3→6→8.

Este exemplo ilustra ambas as declarações do Princípio da Otimalidade. A partir do nó 1 e usando a política ótima o sistema move-se para o nó 3, depois do nó 3 para o 6 e, por último, do nó 6 para o nó 8. Entretanto, se o sistema iniciar no nó 3, a política ótima seria ir para o nó 6 e então para o nó 8. Isso é idêntico à política ideal para todo o problema como afirmado no Princípio da Otimalidade. Isso acontece porque a política π^* dá o caminho ideal a partir de cada nó começando do último nó, 8. A segunda declaração do Princípio da Otimalidade também está satisfeito.

5. PROCESSOS DE DECISÃO DE *MARKOV* COM HORIZONTE INFINITO

Neste capítulo serão apresentados alguns itens sobre os processos de decisão de *Markov* com horizonte infinito. Dentre os assuntos discutidos terão algoritmos e problemas resolvidos com este modelo.

5.1 CRITÉRIOS DE OTIMALIDADE

Neste capítulo serão considerados problemas de decisão de *Markov* onde o número de épocas de decisão é infinito $T = \{1,2,3,...\}$. A teoria para este tipo de problema se concentra, em grande parte, nos casos estacionários, ou seja:

$p(s'|s,a)$ *e* $r(s,a)$ não dependem de t.

Essa hipótese de estacionalidade acaba fazendo políticas estacionárias do tipo $\Pi = (d,d,d,...) \equiv d^{\infty}$ tomar um papel fundamental.

Para medir o valor de uma política, um critério será definido para ajudar a escolher a melhor política dentre as possíveis. O critério utilizado será o do valor do custo total esperado descontado, cuja equação é dada a seguir:

$$V_{\lambda}^{\pi}(s) = \lim_{N\to\infty} E_s^{\pi}\left\{\sum_{t=1}^{N} \lambda^{t-1} r(x_t, d_t(x_t))\right\}.$$

O limite sempre existe quando $\lambda \in [0,1)$ e $|r(s,a)| \leq M < \infty$ para todo $s \in S$ e $a \in A_s$. Neste caso é possível trocar o limite com a esperança e com isso é a equação pode ser reescrita da seguinte forma:

$$V_{\lambda}^{\pi}(s) = E_s^{\pi}\left\{\sum_{t=1}^{\infty} \lambda^{t-1} r(x_t, d_t(x_t))\right\}.$$

Para garantir a convergência do valor da recompensa total esperada em um horizonte infinito utiliza-se um fator de desconto $\lambda \in [0, 1)$.

5.2 AVALIAÇÃO DE POLÍTICAS

Assim como no capítulo referente a MDP com horizonte finito, a discussão será iniciada mostrando como calcular V_λ^π para uma política determinística e estacionária $\pi \in \Pi$.

A fim de facilitar o desenvolvimento da teoria para MDP com horizonte infinito, será utilizado uma notação vetorial. Assumindo que S é um conjunto contável ou finito e seja V o espaço de funções $v : S \rightarrow R$ podemos supor que v é um vetor onde cada componente possui o valor de v em um estado s. Desta forma, para uma regra de decisão d determinística e *Markoviana* qualquer podemos definir o vetor r_d, onde cada componente possui o valor de $r_d(s) = r(s,s(s))$. E, de forma similar, para uma regra de decisão d será definida a matriz de probabilidades de transição $P_d \in R^{|S|x|S|}$ como sendo a matriz com índices $(P_d)_{ij} = p(j|i,d(i))$.

Considerando que $\pi = (d_1,d_2,d_3,...)$ então temos, utilizando notação vetorial, que:

$$V_\lambda^\pi = r_{d_1} + \lambda P_{d_1} r_{d_2} + \lambda^2 P_{d_1} P_{d_2} r_{d_3} + \cdots$$

$$= r_{d_1} + \lambda P_{d_1} (r_{d_2} + \lambda P_{d_2} r_{d_3} + \cdots)$$

$$V_\lambda^\pi = r_{d_1} + \lambda V_\lambda^{\pi'}$$

onde $\pi' = (d_2,d_3,d_4,...)$.

Repetindo o que foi feito acima para a política $d^\infty = (d,d,d,...)$ temos:

$$V_\lambda^{d^\infty} = r_d + \lambda P_d V_\lambda^{d^\infty}.$$

Ou seja, $V_\lambda^{d^\infty}$ é uma solução do sistema linear:

$$V = r_d + \lambda P_d V$$

$$(1 - \lambda P_d)V = r_d$$

Com o resultado encontrado a seguir é possível mostrar que existe uma única solução para o sistema linear citado acima.

Teorema: Seja $Q \in R^{|S|x|S|}$ uma matriz qualquer tal que

$$||Q|| \equiv \sup_{i \epsilon S} \sum_{j \epsilon S} |Q_{ij}| \leq 1 \; então:$$

1. $(1 - Q)^{-1}$ existe;
2. $$(1 - Q)^{-1} = \sum_{n=0}^{\infty} Q^n.$$

Para mostrar que a solução de $(1 - \lambda P_d)V = r_d$ é única basta mostrar que $(1 - \lambda P_d)$ é não singular.

Usando o teorema acima, $(1 - \lambda P_d)$ é não singular se $||\lambda P_d|| < 1$:

$$||\lambda P_d|| = \sup_{i \epsilon S} \sum_{j \epsilon S} |\lambda (P_d)_{ij}| = \sup_{i \epsilon S} \lambda = \lambda < 1.$$

Portanto $(1 - \lambda P_d)^{-1}$ existe e temos a solução $V = (1 - \lambda P_d)^{-1} r_d$. Usando ainda o resultado encontrado acima, notamos que:

$$V = \left(\sum_{n=0}^{\infty} \lambda^n P_d^n \right) r_d = \sum_{n-1}^{\infty} \lambda^{n-1} P_d^{n-1} r_d$$

$$= r_d + \lambda P_d r_d + \lambda^2 P_d P_d r_d + \cdots$$

$$= V_{\lambda}^{d^{\infty}}.$$

Logo $V_{\lambda}^{d^{\infty}}$ é a única solução de $V = r_d + \lambda P_d V$.

5.3 ALGORITMOS

Neste trabalho serão apresentados dois algoritmos, dentre vários existentes, para a solução de MDPs com horizontes infinitos. Um trabalha diretamente com política, enquanto que o outro trabalha com funções valor.

5.3.1 ITERAÇÃO DE VALORES

Sendo *V* uma função valor para um MDP *M = (S, A, p, r)*, o algoritmo de iteração de valores usa programação dinâmica para determinar o valor *V*(s)* de cada estado $s \in S$ do MDP, em cada época de decisão.

Seja $h : S \times A \times V \rightarrow R$, tal que *h(s, a, V)* dá o valor resultante da ação *a* no estado *s* e segue uma dada política, derivada de uma função valor *V* após a realização desta ação:

$$h(s,a,V) = r(s,a) + \lambda \sum_{s' \in S} p(s'|s,a)V(s').$$

Um operador $H : V \rightarrow V$ de melhoria de política será utilizado. Ele determinará a melhor ação em um estado usando os valores V dos estados de uma época de decisão anterior:

$$HV_k(s) = \max_{a \in A} h(s,a,V_{k+1}).$$

A partir da época de decisão anterior, a equação de otimalidade para MDPs determina uma função valor em uma época de decisão utilizando o operador *H*:

$$V_k(s) = HV_{k+1}(s) = \max_{a \in A} \left[r(s,a) + \lambda \sum_{s' \in S} p(s'|s,a)\, V_{k+1}(s') \right].$$

Esse algoritmo recebe como entrada um MDP *(S, A, p, r)* e tem como saída um valor *V* para o MDP de entrada. Seu funcionamento é descrito a seguir:

para cada s ∈ S faça
$V_0(s) \longleftarrow max_{a \in A} R(s,a)$;
fim
i ⟵ 1;
enquanto o critério de parada não for satisfeito faça
para cada s ∈ S faça
$V_i(s) \longleftarrow max_{a \in A}[\, r(s,a) + \lambda \Sigma_{s' \in S}\, p(s' \mid s,a) V_{i-1}(s')]$;
fim
i ⟵ i + 1;
fim
retorne V;

Figura 7. Iteração de Valores.

Esse algoritmo pode ser usado para resolver MDPs com horizonte infinito, porque converge para a função valor ótima.

Caso o número de iterações seja igual ao horizonte do problema, então este é o critério de parada no caso de problemas com horizonte finito. Caso o horizonte seja infinito, o algoritmo para quando os valores para cada estado tiverem convergido.

5.3.2 ITERAÇÃO DE POLÍTICAS

No caso de horizonte infinito (e política estacionária) pode-se também usar um algoritmo chamado de iteração de políticas, que faz uma busca gulosa no espaço de políticas. Este algoritmo é mais eficiente que o de iteração de valores (PELLEGRINI AND WAINER, 2007).

Partindo de uma política aleatória, o algoritmo determina o valor dessa política e, de maneira gulosa, melhora essa política buscando modificar as ações recomendadas para cada estado.

O algoritmo de iteração de políticas se baseia no Teorema de Melhoria da política e no de Otimalidade da política.

Seguindo o Teorema de Melhoria da política, sejam duas políticas estacionárias, π e π', tais que

$$\forall s \in S, V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s))$$

onde V^{π} é a função valor de uma política π para um MDP $M = (S, A, p, r)$ que dá o valor esperado da recompensa para esta política de acordo com o critério de otimalidade e Q^{π} a função que dá o valor da ação a no estado s considerando a recompensa imediata de a e a recompensa esperada após a nas outras épocas de decisão desde que as ações tomadas após a sejam determinadas pela política π (PELLEGRINI AND WAINER, 2007), então π' é uniformemente melhor que π, ou seja:

$$\forall s \in S, V^{\pi}(s) \leq V^{\pi'}(s).$$

O resultado desse teorema pode ser utilizado para melhorar uma política que não seja ótima, já que ele diz que π' é uniformemente melhor que π.

Seguindo o Teorema de Otimalidade da política, seja uma política para um MDP M e uma e uma função valor V^{π} associada a π. Se π não puder ser melhorada usando o teorema anterior, ou seja, se

$$\forall s \in S\, V^{\pi}(s) = \max_{a \in A} Q^{\pi'}(s, a)$$

então π é ótima para M.

O algoritmo de iteração de políticas determina qual é a melhor ação a tomar para cada estado s:

$$\forall s \in S, \pi(s) = arg \max_{a \in A} Q^{\pi'}(s, a).$$

Esse algoritmo recebe como entrada um MDP (S, A, p, r) e tem como saída uma política π^* ótima para o MDP de entrada. Seu funcionamento é descrito a seguir:

π deverá ser inicializado aleatoriamente
repita
    π' recebe o valor de π;
    O sistema linear deverá ser resolvido para avaliar a política atual:
    $\forall s \in S, V(s) = \mathrm{r}(s, \pi'(s)) + \lambda \Sigma_{s' \in S} \mathrm{p}(s' \mid s, \pi'(s)) V(s')$;
    para cada s ∈ S faça
        Melhorar a política:
        $\pi(s) \longleftarrow argmax_{a \in A} Q^{\pi'}(s, a)$;
    fim
até que $\pi = \pi'$;
retorne π

Figura 8. Iteração de Políticas.

5.4 PROBLEMAS

Nesta seção iremos considerar dois problemas de MDP com horizonte infinito. Aqui serão utilizados os algoritmos de Iteração de Valores e o de Iteração de Políticas para encontrar políticas ótimas para esses problemas.

5.4.1 PROBLEMA DO APOSTADOR (*GAMBLER'S PROBLEM*)

Um jogador tem a oportunidade de fazer apostas sobre os resultados de uma sequência de lançamentos de moeda.

Se a moeda dá cara, ele ganha tantos dólares quanto ele tem apostou nesse arremesso. Se der coroa, ele perde a aposta.

O jogo termina quando o jogador atingir seu objetivo de U$100, ou perder e ficar sem dinheiro.

Em cada jogada, o jogador deve decidir quanto ele deve apostar, em números inteiros de dólares.

O estado é representado pela quantidade de capital do jogador, $s \in \{1, 2, ..., 99\}$ e as ações são os valores das apostas, $a \in \{0, 1, ..., min(s, 100 – s)\}$. A recompensa é zero em todas

as transições, exceto aquelas em que o jogador atinge o seu objetivo, quando a recompensa é *+1*.

A função de valor de estado dá a probabilidade de ganhar em cada estado. A política é um mapeamento de níveis de risco de quantidade de capital apostado por aposta. A política ótima maximiza a probabilidade de alcançar a meta.

P denota a probabilidade da moeda dar cara, ou seja, o jogador ganhar a aposta. Se *P* é conhecido, então todo o problema é conhecido e pode ser resolvido utilizando, por exemplo, o algoritmo de Iteração de Valor.

Utilizando $P = 0.4$ e um código utilizando a linguagem de programação *C* para resolver o problema do apostador foi criado se baseando no algoritmo de Iteração de Valor.

A política gerada pelo algoritmo é mostrada na seguinte figura:

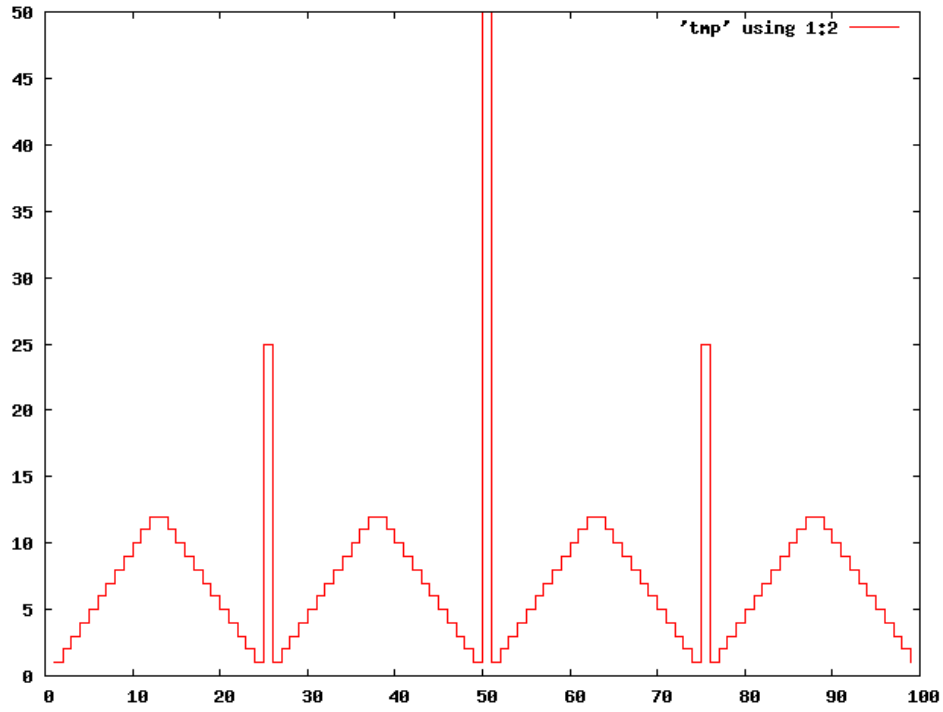

Figura 9. Política ótima do problema do apostador gerada pelo algoritmo Iteração de Valores.

A política resultante mostra a quantidade de dinheiro que deve ser apostada em relação à quantidade de dinheiro em posse do apostador.

Considerando os mesmos valores para a probabilidade *P* de vitória na aposta, o mesmo problema foi resolvido utilizando o algoritmo de Iteração de Política. O resultado

encontrado foi exatamente o mesmo descrito na no gráfico de resultados gerados pelo algoritmo de Iteração de Valores.

Conclui-se com os resultados obtidos que ambos os algoritmos podem ser usados para resolver problemas de MDP com horizontes infinitos e que consegue-se obter o mesmo resultando com qualquer um dos algoritmos utilizados.

5.4.2 PROBLEMA DE INSTABILIDADE DO *SLOTTED ALOHA*

Slotted Aloha é o nome dado a um protocolo de múltiplo acesso ao meio desenvolvido para a "Rede Aloha".

A Rede Aloha é uma rede de radiodifusão via satélite que começou a operar em 1970. O objetivo dessa rede era interligar o computador do centro de computação da Universidade do Havaí aos terminais localizados na mesma linha ou em outras linhas.

A instabilidade deste protocolo se dá quando vários usuários lutam por acesso a um único canal de link de comunicação via satélite com o objetivo de transmitir mensagens. Duas ou mais mensagens no ar geram congestionamento e podem não ser transmitidas com êxito. Neste protocolo os usuários são capazes de detectar colisões desse tipo e tentam retransmitir as mensagens quando isso ocorre. Nem sempre ocorre cooperação entre os usuários nos pedidos de retransmissão das mensagens, ou seja, os usuários tentam retransmitir as suas mensagens ao mesmo tempo.

O protocolo slotted Aloha impõe aos utilizadores as seguintes regras:

- Transmissões e retransmissões de mensagens podem começar apenas em momentos espaçados; o intervalo entre duas (re)transmissões consecutivas é chamada de slot; a duração de um slot é sempre maior do que a de qualquer mensagem.
- Todas as mensagens que tentaram, sem sucesso, pelo menos uma vez passar pelo link são chamadas *backlogged* e requerem a retransmissão, independentes de outras mensagens, com probabilidade $v \in (0, 1)$ em cada slot. Política de retransmissão de Bernoulli.
- As mensagens *fresh* (novas), as que se apresentam pela primeira vez, tentam passar imediatamente.

A ideia básica deste protocolo é que cada nó não *backlogged* simplesmente transmite um pacote recém-chegado no primeiro *slot* após a chegada do pacote, assim arriscando colisões ocasionais, mas consegue um atraso muito pequeno se as colisões são raras.

Quando ocorre uma colisão no *slotted* Aloha, um dos pacotes, o que não teve a transferência realizada com sucesso torna *backlogged*.

Se cada nó *backlogged* simplesmente retransmitir no próximo *slot*, certamente ocorrerá outra colisão, e, por isso, estes nós esperam por algum número aleatório de *slots* antes de retransmitir.

Seja X_n o número de mensagens *backlogged* no início do *slot n*. Se houver $X_n = k$ mensagens *backlogged*, a probabilidade que *i* entre as mensagens tentarão retransmitir é:

$$b_i(k) = \binom{k}{i} v^i (1-v)^{k-1}.$$

Seja A_n o número de novas solicitações para a transmissão no *slot n*. A sequência $\{A_n\}_{n\geq 0}$ é dada com a distribuição:

$$P\,(A_n = j) = a_j.$$

A matriz de transições é mostrada abaixo:

$$p_{ij} = \begin{cases} b_1(i)a_0 & se\ j = i-1, \\ [1-b_1(i)]a_0 + b_0(i)a_1 & se\ j = 1, \\ [1-b_0(i)]a_1 & se\ j = i+1, \\ a_{j-1} & se\ j \geq i+2. \end{cases}$$

A primeira probabilidade indica a chance de que uma entre as *i* mensagens *backlogged* é retransmitida com sucesso. Para isso não deve haver mensagem *fresh* e apenas uma mensagem *backlogged* com permissão para retransmitir.

A segunda probabilidade diz que um dos dois eventos deve ocorrer: "não chegada de mensagem *fresh* e zero ou mais de dois pedidos de retransmissão de mensagens *backlogged*" e "zero requisições de retransmissões de mensagens *backlogged* e uma chegada de mensagem *fresh*".

A terceira probabilidade diz respeito à chance de qualquer número positivo e diferente de zero de requisições de mensagens *backlogged* mais a chegada de uma mensagem *fresh*.

A última probabilidade é a chance de chegar mais de uma mensagem *fresh* ao sistema ao mesmo tempo.

De acordo com essas probabilidades o número de mensagens atrasadas por conta de colisões, pode diminuir em, no máximo, uma unidade por transição, mas pode aumentar por um valor arbitrário.

Se a taxa de chegada é pequena, e poucos pacotes estão envolvidos em colisões então as retransmissões são normalmente bem sucedidas. Por outro lado, se o número de pacotes *backlogged* é muito grande, então colisões ocorrem em quase todos os *slots* sucessivamente.

A partir dessas informações, foi construído um MDP para resolver o problema de colisões do protocolo *slotted Aloha*. Para a resolução desse problema foi utilizado o algoritmo de iteração de valores. O valor do custo imediato para o estado s e ação a utilizado foi: $r(s,a) = \sum_{s' \in S} s' p(s'|s,a).$ Esse algoritmo foi construído utilizando a linguagem de programação C. Na implementação deste algoritmo foi utilizado como probabilidade de chegada de novas mensagens uma distribuição de Poisson e o valor de desconto λ utilizado foi de 0.9. O resultado encontrado é mostrado na seguinte figura:

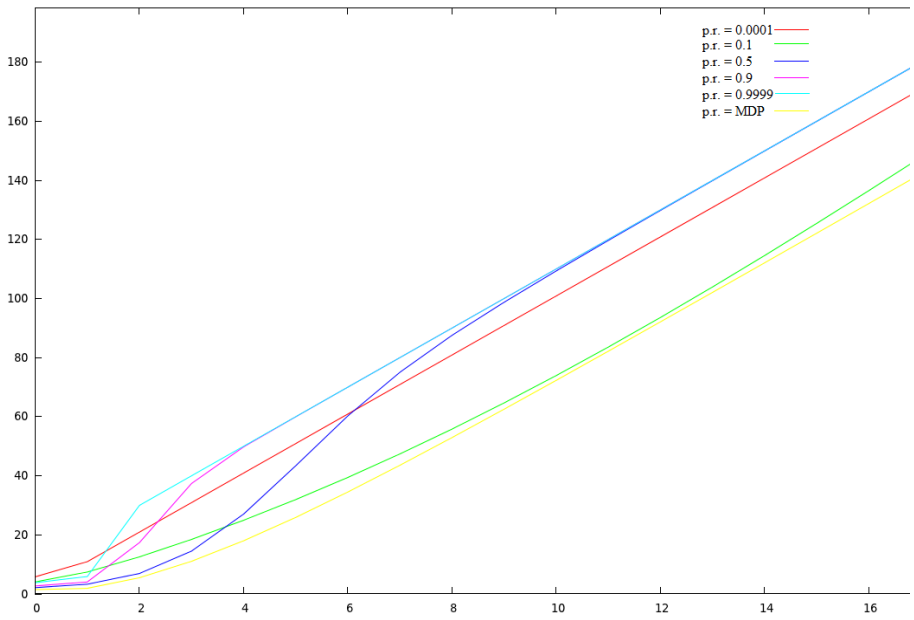

Figura 10. Gráfico de resultados de diferentes políticas para o slotted Aloha.(Parte inicial).

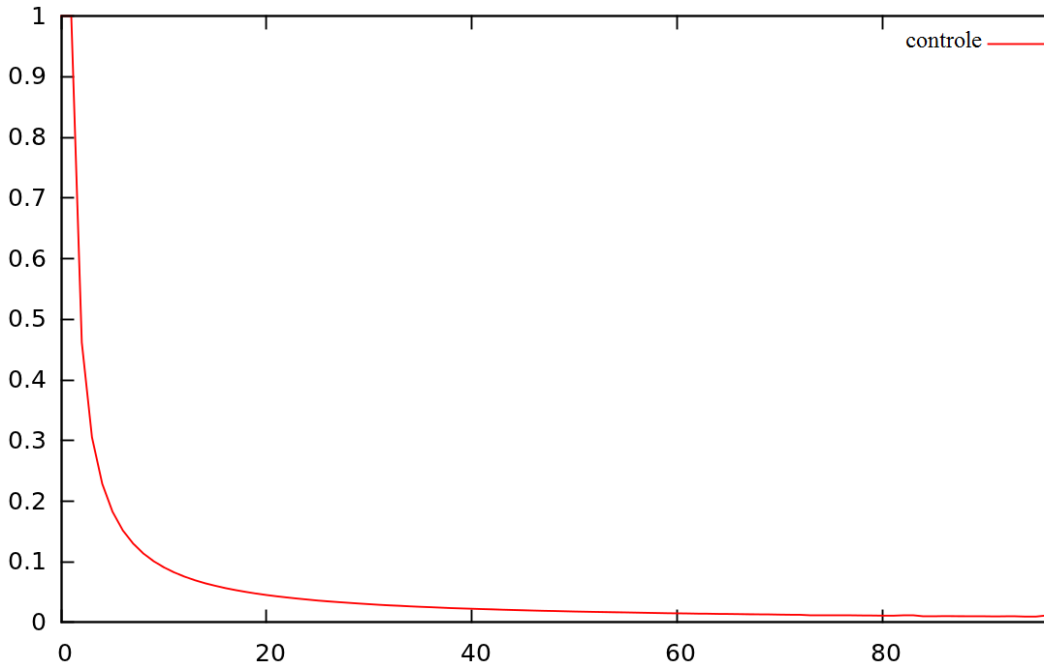

Figura 11. Gráfico de controle do slotted Aloha.

Este gráfico mostrado na figura 10 indica o valor do custo esperado *V(s)* (eixo y) pelo estado *s* (eixo x).

Na figura 10, *P.r.(política de retransmissão)* indica o valor utilizado como probabilidade de retransmissão, variando entre 0,0001 à 0,9999, de mensagens *backlogged* em cada política de retransmissão. Essa probabilidade teve valores fixos em diversas simulações e teve o seu valor selecionado de acordo com a política no processo de decisão de *Markov*. Nota-se que, durante todo o processo, a política mais efetiva foi a política encontrada pelo MDP. A partir do gráfico mostrado na figura 10, conclui-se que para qualquer política de probabilidades fixas de retransmissão de mensagens *backlogged*, a política construída pelo MDP aparece mais eficaz.

O gráfico ilustrado pela figura 11 mostra os valores das probabilidades de retransmissão de mensagens *backlogged*, selecionados pela política determinada pelo MDP, de acordo com o estado *s* em que o sistema se encontra. Percebe-se que para uma quantidade menor de mensagens *backlogged* o MDP seleciona probabilidades maiores para retransmissão, enquanto que com o aumento do número dessas mensagens, a probabilidade de retransmissão diminui.

Através de uma regressão é possível verificar que a regra de decisão descrita na figura 11 é dada por $d(s) = 1/s$. Esta foi a mesma regra de decisão encontrada por (FEINBERG, 1984) utilizando uma metodologia diferente.

6. CONCLUSÃO

Os processos de decisão de *Markov* se mostraram extremamente úteis na resolução de problemas em ambientes de incerteza. Eles conseguem selecionar uma sequência de ações a serem executadas em uma sequência de decisões formando uma política ótima que minimiza o custo esperado ou maximiza a recompensa esperada. Isso se mostra de fundamental importância em diversas áreas, como mostrado no trabalho, pois em muitas delas, é imprescindível que a decisão seja tomada de forma correta, não existindo grandes margens para erro, como em problemas que envolvem uma grande quantidade financeira, por exemplo.

Foi mostrado neste trabalho que problemas tais como, sistemas de redes, sistemas de rotas, ou até mesmo sistemas de aposta podem ter soluções ótimas encontradas a partir da utilização de processos de decisão de *Markov* na resolução desses problemas, o que mostra a grande gama de áreas de possível atuação do tema abordado.

O objetivo estabelecido para este trabalho foi alcançado com êxito, contudo o tempo relativamente curto de um semestre não permitiu que fossem produzidas muitas outras soluções de diversos outros problemas e diferentes áreas, permitindo assim que este trabalho possa ser continuado e melhorado.

6.1 TRABALHOS FUTUROS

Diversas possibilidades surgem a partir desse trabalho. A primeira delas é a continuação da busca por soluções de diversos problemas através da utilização do tema abordado. A segunda seria a realização de um estudo de forma a obter algoritmos mais simples e eficientes para a resolução desses problemas utilizando os processos de decisão de *Markov*.

7. REFERÊNCIAS

Barreto, A. M. S. (2008). Soluções Aproximadas Para Problemas De Tomada De Decisão Sequencial. Tese de Doutorado.

Bellman, R. E. (1957). Dynamic Programming. Princeton University Press.

Bellman, R. E. (1959). On Adaptive Control Processes. Automatic Control, IRE Transactions on.

Bertsekas, D. (2001). Athena scientific. In Dynamic programming and optimal control.

Fainberg, E.A., Kogan, Ya. A. and Smirnov, A. N. (1984). Optimal Control by the Retransmission Probability in Slotted Aloha Systems. Institute of Control Sciences, Profsouznaya ul.65, Moscow GSP-321, USSR.

Hauskrecht, M. (2000). Artificial intelligence in medicine. In Planning treatment of ischemic heart disease with partially observable markov decision processes.

Hui, B. and Boutilier, C. (2006). Who is asking for help? A Bayesian approach to intelligent assistance. *International Conference on Intelligent User Interfaces (IUI)*.

Pellegrini, J. and Wainer, J. (2003). On the use of pomdps to model diagnosis and treatment of diseases. *In On the use of POMDPs to model diagnosis and treatment of diseases*.

Pellegrini, J. and Wainer, J. (2007). Processos de decisão de markov: um tutorial. *Revista de Informática Teórica e Aplicada*.

Pineau, J.and Thrun, S. (2001). Hierarchical pomdp decomposition for a conversational robot. *Workshop on Hierarchy and Memory in Reinforcement Learning. William College, MA, USA.*

Puterman, M. L. (1994). Wiley-interscience. In Markov Decision Process: Discrete Stochastic Dynamic Programming.

Puterman, M. L. (2005). Wiley-interscience. In Markov Decision Process: Discrete Stochastic Dynamic Programming.

Smith, T. and Simmons, R. (2005). Point-based pomdp algorithms: Improved analysis and implementation. *National Conference on Artificial Intelligence (AAAI)*.

Walrand, J. (1988). An Introduction to Queuing Networks. *Prentice Hall*.