UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

DENIS BENEVOLO PAIS

Abordagens eficientes e aproximadas com políticas estacionárias para $CVaR$
MDP

São Paulo

2020

DENIS BENEVOLO PAIS

**Abordagens eficientes e aproximadas com políticas estacionárias para *CVaR*
MDP**

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Versão corrigida contendo as alterações solicitadas pela comissão julgadora em 05 de dezembro de 2019. A versão original encontra-se em acervo reservado na Biblioteca da EACH-USP e na Biblioteca Digital de Teses e Dissertações da USP (BDTD), de acordo com a Resolução CoPGr 6018, de 13 de outubro de 2011.

Orientador: Prof. Dr. Karina Valdivia Delgado

São Paulo

2020

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

CATALOGAÇÃO-NA-PUBLICAÇÃO

(Universidade de São Paulo. Escola de Artes, Ciências e Humanidades. Biblioteca)
CRB 8 - 4936

Pais, Denis Benevolo

Abordagens eficientes e aproximadas com políticas
estacionárias para CVaR MDP / Denis Benevolo Pais ; orientadora,
Karina Valdívila Delgado. – 2020.

76 f. : il

Dissertação (Mestrado em Ciências) - Programa de Pós-
Graduação em Sistemas de Informação, Escola de Artes,
Ciências e Humanidades, Universidade de São Paulo, em
2019.

Versão corrigida

1. Inteligência artificial. 2. Processos de Markov. 3. Teoria
da decisão. 4. Finanças. 5. Heurística. I. Delgado, Karina
Valdívila, orient. II. Título

CDD 22.ed.– 006.3

Dissertação de autoria de Denis Benevolo Pais, sob o título “**Abordagens eficientes e aproximadas com políticas estacionárias para *CVaR* MDP**”, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em 05 de dezembro de 2019 pela comissão julgadora constituída pelos doutores:

Profa. Dra. Karina Valdivia Delgado

Instituição: Universidade de São Paulo

Presidente

Prof. Dr. Carlos Roberto Lopes

Instituição: Universidade Federal de Uberlândia

Prof. Dr. Esteban Fernandez Tuesta

Instituição: Universidade de São Paulo

Prof. Dr. Masayuki Oka Hase

Instituição: Universidade de São Paulo

Dedico este Trabalho

A Deus, por me conceder saúde e sabedoria para seguir sempre em frente.

À minha mãe Isis e ao meu pai Aldo.

Aos meus queridos e Amados Sobrinhos, e família.

*A todos que acreditam na educação como uma poderosa ferramenta de transformação e
libertação individual e social.*

Agradecimentos

Agradeço muito a minha orientadora Dra. *Karina Valdivia Delgado*, pela oportunidade de realizar este trabalho. Agradeço pela confiança e por me atender e ensinar com paciência e principalmente pela parceria criada no desenvolvimento deste trabalho. Agradeço por todos os ensinamentos compartilhados, e por me guiar nos primeiros passos da pós-graduação.

Agradeço muito o professor Dr. *Valdinei Freire da Silva*, por todo apoio e dedicação que teve comigo e com minha pesquisa durante o mestrado e pelas suas contribuições vitais para o desenvolvimento deste trabalho.

Agradeço ao amigo, Felipe Fernandes, pela parceria e pelo interesse na minha pesquisa, por escutar, ler, dar sugestões e contribuir na revisão do texto final.

Agradeço ainda a todos meus familiares, pelo suporte e amor recebido durante esse período.

Resumo

PAIS, Denis Benevolo. **Abordagens eficientes e aproximadas com políticas estacionárias para $CVaR$ MDP** 2020. 76 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2019.

Processos de decisão Markovianos (*Markov Decision Processes* – MDPs) são amplamente utilizados para resolver problemas de tomada de decisão sequencial. O critério de desempenho mais utilizado em MDPs é a minimização do custo total esperado. Porém, esta abordagem não leva em consideração flutuações em torno da média, o que pode afetar significativamente o desempenho geral do processo. MDPs que lidam com esse tipo de problema são chamados de MDPs sensíveis a risco. Um tipo especial de MDP sensível a risco é o $CVaR$ MDP, que inclui a métrica $CVaR$ (*Conditional-Value-at-Risk*) comumente utilizada na área financeira. Um algoritmo que encontra a política ótima para $CVaR$ MDPs é o algoritmo de Iteração de Valor com Interpolação Linear chamado $CVaRVILI$. O algoritmo $CVaRVILI$ precisa resolver problemas de programação linear várias vezes, o que faz com que o algoritmo tenha um alto custo computacional. O objetivo principal deste trabalho é projetar abordagens eficientes e aproximadas com políticas estacionárias para $CVaR$ MDPs. Para tal, é proposto um algoritmo que avalia uma política estacionária para $CVaR$ MDPs de custo constante e que não precisa resolver problemas de programação linear, esse algoritmo é chamado de $PECVaR$. $PECVaR$ é utilizado para inicializar o algoritmo $CVaRVILI$ e também é utilizado para se obter um novo algoritmo heurístico para $CVaR$ MDPs chamado $MPCVaR$ (*Multi Policy CVaR*).

Palavras-chaves: Processo de Decisão Markoviano, Processo de Decisão Markoviano Sensível ao Risco, Política Estacionária, $CVaR$

Abstract

PAIS, Denis Benevolo. **Efficient and Approximate Approaches with Stationary Policies for *CVaR* MDP**. 2020. 76 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2019.

Markov Decision Process - MDPs are widely used to solve sequential decision-making process problems. The objective function or criteria of assessment mostly used in this problem's case is the minimization of the expectation of the total cost. However, this approach does not consider the variability of the cost (in other words , fluctuations related to the mean), that can affect significantly your general performance. MDPs which deal with this kind of problems are called Risk Sensitive MDPs. A special kind of Risk Sensitive MDP is the *CVaR* MDP, which includes the *CVaR* (*Conditional-Value-at-Risk*) metric, a robust way in order to measure risks and commonly used to measure financial risk. One algorithm that finds the optimal policy for *CVaR* MDPs is the *CVaR* Value Iteration with linear Interpolation algorithm (*CVaRVILI*). The *CVaRVILI* algorithm needs to solve linear programming problems several times, which makes the algorithm costly to compute. The main objective of this paper is to design efficient and approximate approaches with stationary policies for *CVaR* MDPs. For this purpose, an algorithm that evaluates a stationary policy for *CVaR* constant cost MDPs and that does not need to solve linear programming problems is proposed, this algorithm is called *PECVaR*. *PECVaR* is used to initialize the *CVaRVILI* algorithm and is also used to obtain a new heuristic algorithm for *CVaR* MDPs called *MPCVaR* (*Multi Policy CVaR*).

Keywords: Markovian Decision Process, Risk-Sensitive Markovian Decision Process, Stationary Policy, CVaR.

Lista de figuras

Figura 1 – Políticas com dois níveis de confiança diferentes para um exemplo de navegação autônoma	18
Figura 2 – Distribuição histórica do retorno	31
Figura 3 – Política π_1 para um <i>CVaR</i> MDP com 7 estados	35
Figura 4 – Solução ótima para <i>CVaR</i> MDP	38
Figura 5 – Intuição do Algoritmo PECVaR	43
Figura 6 – Política π_1 para um <i>CVaR</i> MDP com 7 estados	47
Figura 7 – Algoritmo PECVaRMIN	49
Figura 8 – Algoritmo MPCVaR e simulação	53
Figura 9 – Instância do <i>Fast Slow</i> com 7 estados. O estado meta é o cinza claro, o estado inicial é preto.	55
Figura 10 – Instância do <i>Grid World</i> de tamanho 14×16 . O estado meta é o cinza claro, o estado inicial é preto e os obstáculos são cinza escuro.	56
Figura 11 – Instância da <i>Travessia do Rio</i> com 25 estados. O estado meta é o cinza claro, o estado inicial é preto.	57
Figura 12 – Qualidade da aproximação do <i>CVaRVILI</i> para 11, 21 e 31 pontos de interpolação na instância do domínio <i>Grid World</i> com 224 estados . . .	58
Figura 13 – Qualidade da aproximação do <i>CVaRVILI</i> para 11, 21 e 31 pontos de interpolação na instância do domínio <i>Fast Slow</i> com 7 estados	59
Figura 14 – Qualidade da aproximação do <i>CVaRVILI</i> para 11, 21 e 31 pontos de interpolação na instância da <i>Travessia do Rio</i> com 35 estados	60
Figura 15 – Tempo total de execução do algoritmo <i>CVaRVILI</i> com diferentes abordagens de inicialização para duas instâncias do domínio <i>Fast Slow</i> com 11, 21 e 31 pontos de interpolação.	63
Figura 16 – Tempo total de execução do algoritmo <i>CVaRVILI</i> com diferentes abordagens de inicialização para a instância com 224 estados do domínio <i>Grid World</i> com 11, 21 e 31 pontos de interpolação e para a instância com 3392 estados com 11 e 21 pontos de interpolação.	64

Figura 17 – Tempo total de execução do algoritmo <i>CVaRVILI</i> com diferentes abordagens de inicialização para a instância com 35 estados do domínio <i>Rio</i> com 11, 21 e 31 pontos de interpolação e para a instância com 364 estados com 11, 21 e 31 pontos de interpolação.	65
Figura 18 – Políticas utilizadas na simulação do <i>MPCVaR</i> no Fast-Slow com 7 estados.	66
Figura 19 – Simulação <i>CVaRVILI</i> , <i>MPCVaR</i> e <i>PECVaR-MIN</i> no Fast-Slow com 7 estados.	67
Figura 20 – Políticas utilizadas na simulação do <i>MPCVaR</i> no Grid World com 224 estados	67
Figura 21 – Simulação <i>CVaRVILI</i> , <i>MPCVaR</i> e <i>PECVaR-MIN</i> no Grid World com 224 estados	68
Figura 22 – Políticas utilizadas na simulação do <i>MPCVaR</i> no Rio com 35 estados .	69
Figura 23 – Simulação <i>CVaRVILI</i> , <i>MPCVaR</i> e <i>PECVaR-MIN</i> no Rio com 35 estados	69
Figura 24 – Interpolação Linear	76

Lista de algoritmos

Algoritmo 1 – Iteração de Valor	23
Algoritmo 2 – Iteração de Política	25
Algoritmo 3 – $CVaR$ Value Iteration with Linear Interpolation ($CVaRVILI$)	41
Algoritmo 4 – Policy Evaluation $CVaR$ (PECVaR)	46
Algoritmo 5 – Simulação de uma política não-markoviana usando o resultado do algoritmo $MPCVaR$	52

Lista de tabelas

Tabela 1 – Resultados de VaR e $CVaR$	30
Tabela 2 – Comparativo entre VaR e $CVaR$	32
Tabela 3 – Valores obtidos na iteração 1 do algoritmo $CVaRVILI$ para o exemplo da figura 3.	39
Tabela 4 – Valores obtidos na iteração 2 do algoritmo $CVaRVILI$ para o exemplo da figura 3.	40
Tabela 5 – Valores obtidos na iteração 3 do algoritmo $CVaRVILI$ para o exemplo da figura 3.	40
Tabela 6 – Valores obtidos nas iterações do algoritmo $PECVaR$ para o estado s_0 do exemplo da figura 6.	48
Tabela 7 – Valores de P_{-G}^T e ξ obtidos pelo algoritmo $PECVaR$ modificado para o estado s_0 do exemplo da figura 6	52
Tabela 8 – Erro e tempo do algoritmo $CVaRVILI$ - Grid World 224	58
Tabela 9 – Erro e tempo do algoritmo $CVaRVILI$ - Fast Slow 7	58
Tabela 10 – Erro e tempo do algoritmo $CVaRVILI$ - Travessia do Rio 35	59
Tabela 11 – Fast Slow 7 - $CVaRVILI$ e $MPCVaR$	66
Tabela 12 – Simulação Grid World 224 - $CVaRVILI$ e $MPCVaR$	68
Tabela 13 – Simulação Travessia Rio 35 - $CVaRVILI$ e $MPCVaR$	69

Lista de símbolos

$\mathcal{S}$	Espaço de probabilidade
Ω	Espaço amostral
$\mathcal{F}$	Eventos
$\mathbb{P}$	Medida de probabilidade
$\mathcal{Z}$	Espaço de variáveis aleatórias
$\mathbb{E}$	Esperança ou valor esperado
γ	Fator de desconto
s_0	Estado inicial
π	Uma política
π^*	Uma política ótima
μ	Sequência de políticas
V_π	Função valor para a política π
V^*	Função valor ótima π
V^0	Função valor inicial
Q	Função valor para a ação
Z	Uma variável aleatória
$\mathbb{E}_\xi$	Esperança ξ -ponderada de Z
λ	Grau de aversão ao risco
δ	Nível de confiança
β	Nível de confiança
α	Nível de confiança para o CVaR
T	Horizonte do MDP

s'	Estado sucessor
S	Conjunto finito de estados
A	Conjunto finito de ações
P	Função de transição
C	Função de custo
M	MDP
targetS	Estado meta
maxIter	Número máximo de iterações

Sumário

1	Introdução	16
1.1	<i>Objetivo Principal</i>	18
1.2	<i>Objetivo secundário</i>	18
1.3	<i>Organização</i>	19
2	Processos de Decisão Markovianos Neutros ao Risco	20
2.1	<i>Política ótima</i>	22
2.2	<i>Algoritmos para MDPs</i>	22
2.2.1	<i>Iteração de Valor</i>	23
2.2.2	<i>Iteração de Política</i>	24
2.2.3	<i>Formulação com Programação Linear</i>	25
3	Processos de Decisão Markovianos e Risco	26
3.1	<i>Risco e Incerteza</i>	26
3.2	<i>Risco em Processos de Decisão Markovianos</i>	26
3.3	<i>Medidas de Risco de Engenharia Financeira</i>	28
3.4	<i>Métricas VaR e CVaR</i>	29
4	CVaR MDP	33
4.1	<i>Solução ótima para CVaR MDP</i>	37
4.2	<i>Algoritmo CVaRVILI</i>	38
5	Algoritmo Exato Para Avaliar uma Política	43
6	Algoritmos baseados em Múltiplas Políticas para CVaR MDPs: PECVaRMIN e MPCVaR	49
6.1	<i>Algoritmo PECVaRMIN</i>	49
6.2	<i>Algoritmo MPCVaR</i>	50
7	Experimentos	54
7.1	<i>Domínios e Políticas Encontradas pelo algoritmo CVaRVILI</i>	54
7.1.1	<i>Domínio Fast Slow.</i>	54
7.1.2	<i>Domínio Grid World.</i>	55

7.1.3	Domínio Travessia do Rio	56
7.2	<i>Avaliação da Qualidade da Aproximação do CVaRVILI</i>	57
7.3	<i>Diferentes Formas de Inicialização do CVaRVILI</i>	60
7.4	<i>Comparando a Qualidade da Política CVaRVILI, MPCVaR e PECVaR-MIN</i>	62
8	Conclusão	70
8.1	<i>Resumo das contribuições</i>	70
8.2	<i>Publicações geradas</i>	71
8.3	<i>Trabalhos futuros</i>	71
	Referências¹	73
	Anexo A – Interpolação	75
A.1	<i>Interpolação</i>	75
A.2	<i>Interpolação Linear</i>	75

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

A cada dia milhares de decisões são tomadas por agentes humanos e não humanos, decisões com consequências imediatas e a longo prazo. Geralmente decisões não podem ser feitas de forma isolada, pois as decisões tomadas agora podem afetar as decisões futuras. Quando essa relação entre o resultado presente e futuro não é considerada, temos uma grande chance de não atingirmos um bom desempenho. Tais problemas motivam o desenvolvimento de diferentes técnicas para melhorar a modelagem de processos de tomada de decisão.

Um processo de decisão de Markov (*Markov Decision Process*—MDP) é um modelo matemático usado para modelar problemas de tomada de decisões sequenciais nos quais as transições entre estados são probabilísticas. Por meio desta abordagem deve ser possível observar em que estado o processo se encontra e é possível interferir no processo periodicamente (em épocas de decisão) executando ações (PUTERMAN., 1994). Cada ação tem um custo, que depende do estado em que o processo se encontra.

A resolução de MDPs envolve a minimização de uma determinada função de desempenho. A função mais utilizada é o custo total descontado esperado, que é neutra em termos de risco. Esta abordagem, embora muito popular, natural e atraente de um ponto de vista computacional, não leva em consideração a variabilidade do custo, nem sua sensibilidade aos erros de modelagem, o que pode afetar significativamente o desempenho geral do processo (CHOW et al., 2015).

Em muitas situações é preciso garantir com certo grau de certeza que será obtido um determinado resultado. Por exemplo, em um sistema de navegação autônomo, um agente tentará minimizar o comprimento esperado do seu caminho, e para isso ele pode provavelmente movimentar-se próximo de obstáculos na esperança de minimizar a distância. Acrescente a isto o fato deste mesmo agente poder estar num ambiente em que outros agentes, incluindo seres humanos, possam estar presentes. Neste sentindo uma falha ou desvio do caminho planejado pode resultar em uma colisão, ou um grave acidente causando uma perda irreversível (por exemplo, a morte de uma pessoa). É possível tratar desses problemas adicionando uma medida de risco, tornando o MDP sensível a risco.

Diversas medidas para mensuração de risco financeiro têm sido constantemente estudadas e aplicadas em diversos setores. Tais medidas são comumente empregadas em

modelos de otimização estocástica aplicados para problemas do mercado financeiro e também da engenharia no geral. Entre essas medidas estão: variância, *Value-at-Risk* (*VAR*) e *Conditional-Value-at-Risk* (*CVaR*).

CVaR é considerada uma importante métrica de risco. O *CVaR* é definida como a perda esperada condicionada ao fato de se estar no quantil $(1-\alpha)$ da cauda esquerda da distribuição, em que α é o nível de confiança. *CVaR* destaca-se pelas seguintes características: (i) possui propriedades computacionais como eficiência numérica e estabilidade dos cálculos (ROCKAFELLER; URYASEV, 2002); (ii) possui capacidade para proteger um tomador de decisão dos resultados que mais o prejudiquem (CHOW et al., 2015); (iii) *CVaR* é uma medida de risco coerente, isto é, ela atende cinco propriedades matemáticas importantes (Invariância sob translação, Subaditividade, Homogeneidade, Monotonicidade e Convexidade) (ROCKAFELLER; URYASEV, 2000) que serão discutidas posteriormente; e (iv) *CVaR* pode ser representada por uma fórmula de minimização e esta fórmula pode ser facilmente incorporada em problemas de otimização, minimizando o risco do problema ou modelando-a como uma restrição.

Motivados pelas vantagens mencionadas anteriormente, diversos trabalhos sobre MDPs sensíveis a risco que utilizam esse critério foram propostos, entre eles (CHOW et al., 2015), (IYENGAR.; MA, 2009) e (CARPIN.; PAVONE, 2016). Esse MDP que usa *CVaR* como métrica de risco é chamado de *CVaR* MDP. Um algoritmo que encontra a política ótima para *CVaR* MDPs é o algoritmo de Iteração de Valor com Interpolação Linear, chamado *CVaRVILI* (CHOW et al., 2015). A política devolvida pelo algoritmo *CVaRVILI* é uma política estacionária (não depende do tempo) e não markoviana (depende de todo o histórico de ações e estados visitados até o momento). Porém, esse algoritmo é muito custoso pois precisa resolver vários problemas de programação linear.

A Figura 1 mostra uma política com dois níveis de confiança diferentes, $\alpha = 0,01$ e $\alpha = 1$ para um exemplo de navegação autônomo, em que o estado meta é verde, o estado inicial é preto e os obstáculos são azuis. Para cada estado é mostrada a melhor ação a ser tomada (ir para o norte $\uparrow$, sul $\downarrow$, leste $\rightarrow$ e oeste $\leftarrow$). Quando temos α muito baixo ou perto de zero dizemos que estamos sendo muito conservadores ao risco. À medida que aumentamos o α começamos a ser menos conservadores em relação ao risco e no extremo, quando $\alpha = 1$, dizemos que o agente é neutro ao risco. Note que a política com $\alpha = 0,01$ tenta evitar ficar perto dos obstáculos, com intuito de tornar o caminho mais seguro, mesmo que isso signifique um caminho maior. Ao contrário a política com $\alpha = 1$ prefere

caminhos mais curtos até o estado meta entretanto com maior chance de colidir com um obstáculo.

Figura 1 – Políticas com dois níveis de confiança diferentes para um exemplo de navegação autônoma

1.1 Objetivo Principal

O objetivo principal deste trabalho é projetar abordagens eficientes e aproximadas que usam políticas estacionárias para resolver *CVaR* MDPs. Para tal, é formulado um algoritmo que avalia de maneira exata uma política estacionária para *CVaR* MDPs de custo constante que não precisa resolver problemas de programação linear e que tem um custo computacional similar ao de um algoritmo de iteração de valor para MDPs neutros ao risco. Esse algoritmo é chamado de *PECVaR*. *PECVaR* será utilizado para inicializar o algoritmo *CVaRVILI* e também será utilizado para se obter um novo algoritmo heurístico para *CVaR* MDPs chamado *MPCVaR* (*Multi Policy CVaR*).

1.2 Objetivo secundário

O objetivo secundário deste trabalho é melhorar o consumo de tempo do algoritmo *CVaRVILI* utilizando diversas estratégias de inicialização.

1.3 Organização

Esse texto está organizado da seguinte forma. O capítulo 2 apresenta os fundamentos teóricos essenciais para o desenvolvimento deste trabalho são eles: Processos de Decisão Markoviano neutros ao risco e os principais algoritmos para sua resolução. O capítulo 3 apresenta a definição de risco e incerteza e também apresenta algumas formas de lidar com risco em MDPs e também apresenta os fundamentos e as propriedades da métrica VaR e $CVaR$. O capítulo 4 apresenta a fundamentação teórica de um MDP $CVaR$, o operador de Bellman para MDP $CVaR$ e um algoritmo aproximado para MDP $CVaR$ chamado $CVaRVILI$. O capítulo 5 apresenta o algoritmo $PECVaR$ (*Policy Evaluation CVaR*), proposto nesta dissertação, que avalia uma política estacionária de forma exata usando $CVaR$. O capítulo 6 apresenta dois novos algoritmos baseados na abordagem do $PECVaR$ para achar uma solução para um MDP $CVaR$. O capítulo 7 contém os experimentos e resultados obtidos. E por fim, o capítulo 8 apresenta as considerações finais.

2 Processos de Decisão Markovianos Neutros ao Risco

Um processo de decisão Markoviano (*Markov Decision Process* - MDP) (PUTERMAN., 1994) é uma tupla $M = (S, A, P, C, \gamma)$, em que:

- S é um conjunto discreto e finito de estados completamente observáveis que modelam o mundo;
- A é um conjunto finito de ações, a execução de uma ação permite que o sistema mude do estado atual para o próximo estado;
- $P(s'|s, a)$ é a função probabilística de transição que descreve os efeitos da execução de uma ação $a \in A$ em um estado $s \in S$, resultando em um estado $s' \in S$;
- $C(s, a)$ é a função custo de executar uma ação $a \in A$ em um estado $s \in S$; e
- γ é o fator de desconto, sendo $0 \leq \gamma < 1$, assim, depois de t estágios o custo é descontado por γ^t .

O agente executa as ações em passos discretos no tempo. A cada ação executada, o estado do sistema é alterado segundo a função de transição P , sendo que a execução de uma ação em um estado tem um custo. A tomada de decisão é realizada durante um horizonte. O horizonte é o número de passos (ou épocas de decisão) que o agente tem para agir, podendo ser: finito, infinito ou indeterminado. Assim, MDPs podem ser classificados por seu horizonte em:

- MDP com *horizonte finito* que tem um número de passos fixo a se tomar. Podemos imaginar um agente que tem um recurso finito para realizar determinada atividade, por exemplo: combustível, sem chance de reabastecimento, isso limita o número de passos a ser tomado.
- MDP com *horizonte infinito* é quando os passos são tomados repetidamente, sem a possibilidade de parada.
- MDP com *horizonte indeterminado* é parecido ao horizonte infinito, porém com a possibilidade de parada assim que o processo chegar até algum estado final, i.e, até algum estado que termina a execução do processo. Este tipo de MDPs inclui um conjunto de estados meta G .

A solução de um MDP é uma política.

Quanto à sua relação com os passos (ou épocas de decisão) uma política pode ser classificada como:

- *Estacionária*, se a ação recomendada independe do número de passos de decisão. Isso significa que a melhor ação a ser executada no estado s é sempre a mesma, independente do tempo.
- *Não-estacionária*, se a decisão tomada depende do passo de decisão.

É muito comum o uso de políticas estacionárias para horizonte infinito e não-estacionária para horizonte finito. Uma política pode também ser :

- Determinística, quando cada estado $s \in S$ é sempre mapeado em uma única ação.
- Estocástica, quando um estado é mapeado em um conjunto de ações, sendo que a ação tem uma probabilidade de ser escolhida.

Outros tipos de políticas que são de interesse nesta dissertação são as políticas não-markovianas, em que a ação recomendada depende de todo o histórico de ações e estados visitados até o momento.

A solução de um MDP de horizonte infinito ou indeterminado é uma política estacionária determinística $\pi: S \rightarrow A$ que especifica a ação $a = \pi(s)$ a ser escolhida em cada estado s .

Para fazer a comparação entre duas políticas e achar a política ótima π^* , é necessário definir um critério de desempenho. Existem vários critérios que podem ser utilizados, dentre os mais conhecidos podem ser mencionados os seguintes:

- Custo médio por passo de decisão, i.e., $\frac{1}{t} \sum_{k=0}^{t-1} c_k$.
- Custo esperado total, i.e., $E \left[\sum_{k=0}^{t-1} c_k \right]$.
- Custo esperado total descontado, i.e., $E \left[\sum_{k=0}^{t-1} \gamma^k c_k \right]$.

em que t denota o horizonte do MDP. Esses critérios são considerados neutros ao risco, pois não levam em conta o risco de tomar um determinado passo de decisão. O objetivo principal é a minimização de algum desses critérios. Neste capítulo o foco é no custo esperado total descontado.

O valor da política π começando no estado s e executando π , é denotado por $V_\pi(s)$ e é definido como a soma esperada dos custos descontados, em que $s_0 = s$, isto é:

$$V_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k c_k | s_0 = s \right]. \quad (1)$$

Uma política gulosa π_V com respeito a alguma função valor $V : S \rightarrow \mathbb{R}$ é definida como uma política que escolhe uma ação em cada estado s e que minimiza o valor esperado com respeito a V , conforme definida a seguir (PUTERMAN., 1994):

$$\pi_V(s) = \arg \min_{a \in A} \{Q(s, a)\} = \arg \min_{a \in A} \left\{ C(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V(s') \right\}, \quad (2)$$

em que $Q(s, a)$ é o valor do estado s aplicando a ação a .

2.1 Política ótima

Dentre todas as possíveis políticas para um MDP, deseja-se encontrar uma política ótima π^* que minimize o custo total esperado descontado. A função valor ótima, representada por V^* , é a função valor associada com qualquer política ótima. Assim, para um agente que deseja minimizar seu custo total esperado descontado, V^* satisfaz a seguinte igualdade de ponto fixo (BELLMAN, 1957):

$$V^*(s) = \min_{a \in A} \{C(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^*(s')\}. \quad (3)$$

A Equação 3 é conhecida como o Princípio de Otimalidade de Bellman para MDPs ou Equação de Bellman (PUTERMAN., 1994). A política ótima pode ser obtida aplicando *argmin* na equação (3), no lugar de *min*. MDPs que minimizam o custo total esperado descontado são considerados neutros a risco pois não levam em consideração flutuações em torno da média.

2.2 Algoritmos para MDPs

Os três algoritmos clássicos para resolver MDPs são *Iteração de Valor (IV)*, *Iteração de Política (IP)* e a formulação usando programação linear. Os algoritmos IV e IP são considerados métodos de programação dinâmica síncrona e sua principal característica é que realizam atualizações em todo o espaço de estados a cada iteração.

2.2.1 Iteração de Valor

Um algoritmo bem conhecido para determinar o valor $V^*(s)$ de cada estado $s \in S$ do MDP é o algoritmo Iteração de Valor (IV). Começando com um V^0 arbitrário, IV executa atualizações de todos os estados s , calculando V^t baseado no valor de V^{t-1} da seguinte forma:

$$V^t(s) = \min_{a \in A} Q^t(s, a),$$

em que $Q^t(s, a)$ é a função valor do par estado ação e é definida por:

$$Q^t(s, a) = C(s, a) + \gamma \sum_{s' \in S} P(s, a, s') V^{t-1}(s').$$

O pseudocódigo do algoritmo IV é mostrado no algoritmo 1. O algoritmo recebe um MDP e devolve o valor ótimo.

Algoritmo 1 Iteração de Valor

```

1: Entrada: Um MDP( $S, A, P, C, \gamma$ ).
2: Saída: Função valor ótima  $V$ 
3: para cada  $s \in S$  faça
4:    $V^0(s) \leftarrow \min_{a \in A} C(s, a)$ 
5: fim
6:  $t \leftarrow 1$ ;
7: enquanto critério de parada não satisfeito faça
8:   para cada  $s \in S$  faça
9:     para cada  $a \in A$  faça
10:       $Q^t(s, a) = C(s, a) + \gamma \sum_{s' \in S} P(s, a, s') V^{t-1}(s')$ 
11:     fim
12:       $V^t(s) = \min_{a \in A} Q^t(s, a)$ 
13:   fim
14:    $t \leftarrow t + 1$ ;
15: fim
16: devolva  $V$ ;

```

No infinito a função valor converge para V^* , i.e.:

$$\lim_{t \rightarrow \infty} \max_s |V^t(s) - V^{t-1}(s)| = 0. \quad (4)$$

A partir de V^* pode ser obtida a política ótima, determinística e estacionária π^* (PUTERMAN., 1994). Na prática, é possível pensar em ϵ -óptimalidade quando o algoritmo IV alcança a seguinte condição de parada:

$$\max_s |V^t(s) - V^{t-1}(s)| < \frac{\epsilon(1 - \gamma)}{2\gamma}, \quad (5)$$

então é garantido que a política gulosa não perde mais do que ϵ na função valor, quando comparado com V^* (PUTERMAN., 1994). A expressão $\max_s |V^t(s) - V^{t-1}(s)|$ na Equação (5) é chamada de Erro de Bellman (BE - Bellman Error) e a expressão $\frac{\epsilon(1-\gamma)}{2\gamma}$ é chamada de tolerância (tol) (PUTERMAN., 1994). Uma vez que o algoritmo de IV precisa atualizar o espaço de estados inteiro a cada iteração, a complexidade de tempo de execução para cada iteração deste algoritmo de programação dinâmica é $O(|S|^2 * |A|)$.

2.2.2 Iteração de Política

Um outro algoritmo clássico para resolver MDPs é o algoritmo de Iteração de Política (IP) que faz uma busca no espaço de políticas. O algoritmo é iterativo e consiste de dois passos: avaliação de política e melhoria de política.

O algoritmo utiliza uma política inicial π_0 , que pode ser escolhida de forma arbitrária. No passo de avaliação de política podemos calcular a função valor da política na iteração i , $V_{\pi_i}(s)$ para cada estado $s \in S$. No passo de melhoria de política, é calculada a política gulosa π_{i+1} com respeito a V_{π_i} , i.e.:

$$\pi_{i+1}(s) = \arg \min_{a \in A} Q_{\pi_i}(s, a) = \arg \min_{a \in A} \{C(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V_{\pi_i}(s')\}. \quad (6)$$

O algoritmo termina quando $\pi_{i+1}(s) = \pi_i(s)$ para todo $s \in S$. O algoritmo 2 mostra o pseudocódigo do algoritmo de Iteração de Política. Na linha 6 é realizado o passo de avaliação de política e nas linhas 8-12 o passo de melhoria da política.

No passo de avaliação de política, o sistema de equações lineares pode ser resolvido em tempo $O(|S|^3)$ ou através de aproximações sucessivas em tempo $O(|S|^2 * K)$ (em que K é o número de iterações para alcançar ϵ -otimalidade). O passo de melhoria de política pode ser executado em $O(|S|^2 * |A|)$ (LITTMAN M. L., 1995).

Algoritmo 2 Iteração de Política

```

1: Entrada: Um MDP( $S, A, P, C, \gamma$ ).
2: Saída: Uma política ótima  $\pi^*$  para o MDP dado como entrada
3: Inicialize  $\pi$  aleatoriamente
4: repita
5:    $\pi' \leftarrow \pi$ ;
6:   Avalie a política atual resolvendo o sistema linear:
7:     
$$V(s) = C(s, \pi'(s)) + \gamma \sum_{s' \in S} P(s, \pi'(s), s') V(s'), \forall s \in S$$

8:   para cada  $s \in S$  faça
9:     para cada  $a \in A$  faça
       
$$Q^{\pi'}(s, a) \leftarrow C(s, a) + \gamma \sum_{s' \in S} P(s, a, s') V(s')$$

10:    fim
11:     $\pi(s) \leftarrow \arg \min_{a \in A} Q^{\pi'}(s, a)$ 
12:  fim
13: até que  $\pi = \pi'$ ;
14: devolva  $\pi$ ;
```

2.2.3 Formulação com Programação Linear

Os MDPs também podem ser resolvidos usando uma formulação de programação linear. Nessa formulação, a equação 3 é reformulada como a seguir:

$$\max : \sum_{s \in S} V(s) \tag{7}$$

sujeito a :

$$V(s) \leq C(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V(s'), \forall s \in S, a \in A,$$

em que $V(s)$ são as variáveis do problema de programação linear.

3 Processos de Decisão Markovianos e Risco

Neste capítulo são apresentadas as definições de risco e incerteza. Além disto serão descritos alguns Processos de Decisão Markovianos que lidam com risco, bem como as métricas VaR (Value at Risk) e $CVaR$ (Conditional-Value-at-Risk) muito utilizadas para a gestão de portfólio de ativos financeiros (chamados também de ações).

3.1 Risco e Incerteza

A incerteza é inerente em muitos casos na tomada de decisão para negócios, para investimentos e para a própria vida particular de cada ser humano. Em muitos casos o futuro incerto toma terreno em muitas atividades que desenvolvemos. O diagnóstico médico e tratamento de enfermidades é um exemplo: um médico deve escolher entre diversas ações (exames, tratamento, alta) sem ter certeza do estado atual do paciente. Outros exemplos são o sistema de navegação de um robô (que deve decidir para qual direção ir, contando apenas com sensores imprecisos). Várias técnicas têm sido utilizadas e aperfeiçoadas para tentar lidar com esta incerteza.

Knight (KNIGHT, 2012) define risco como uma situação na qual uma distribuição de probabilidade (que é estabelecida com base em observações anteriores, ou seja, a priori) pode ser associada aos resultados esperados. Por outro lado, uma situação incerta, é aquela na qual não se pode associar nenhuma distribuição de probabilidades ou somente se pode associar uma distribuição de probabilidade subjetiva, no sentido de probabilidades obtidas com base no julgamento de especialistas, na ausência de elementos anteriores de referência. De modo geral podemos dizer que risco denota uma situação em que as possibilidades do futuro e probabilidades são conhecidas. Já a incerteza se refere a uma situação em que não se conhecem essas possibilidades.

3.2 Risco em Processos de Decisão Markovianos

Quando MDPs são utilizados para modelar algum problema, existem duas fontes principais de incertezas que precisam ser consideradas: incerteza inerente e incerteza do modelo. A incerteza inerente é a incerteza relacionada às transições probabilísticas dentro

do MDP. A incerteza do modelo é dada devido a algum parâmetro do MDP que não é conhecido exatamente afetando a transição de probabilidade e que consequentemente afeta o custo do MDP. Podemos relacionar essa incerteza com algum erro adicionado por um sensor ou medidor em uma aplicação real.

A tomada de decisão em um MDP, como visto anteriormente, envolve o uso de um critério de otimização. No capítulo 2 foram mostrados os critérios mais conhecidos e que são neutros ao risco. Esses critérios, embora muito populares, naturais e atraentes de um ponto de vista computacional, não levam em consideração a variabilidade do custo (ou seja, flutuações em torno da média), nem sua sensibilidade aos erros de modelagem, que podem afetar significativamente o desempenho geral (CHOW et al., 2015).

Ao longo dos anos, muitos trabalhos têm apresentado diversos critérios de otimização para lidar com esse risco. Garcia e Fernández. (2015) classificam esses critérios em quatro grupos:

- Critério do pior caso: a política considerada ótima é aquela que maximiza o retorno associado ao cenário do pior caso, mesmo que o caso seja altamente improvável (GARCIA; FERNÁNDEZ., 2015). Este grupo é conhecido pela máxima aversão às perdas, e o algoritmo mais utilizado é o algoritmo minmax.
- Critério sensível ao risco: tenta balancear risco e retorno e é caracterizado por possuir um parâmetro que permite controlar a sensibilidade do risco. Nestas abordagens o critério de otimização pode ser transformado em:
 - Uma *função utilidade exponencial*, em que a aversão ao risco do agente é modelada usando um parâmetro β , tal que $\beta < 0$ significa aversão ao risco, $\beta > 0$ propensão ao risco e $\beta = 0$ neutro ao risco.
 - Uma função de transformação linear por partes com fator de desconto (MIHATSCH; NEUNEIER, 2002).
 - Uma combinação linear entre esperança e variância (SOBEL, 1982; FILAR; KALLENBERG; LEE, 1989)
- Critério com restrições: o MDP é chamado de MDP com restrições (*Constrained MDP*). Nestes problemas, o custo acumulado esperado é minimizado sujeito a um conjunto de restrições que não devem ser violadas. Essas restrições podem modelar diferentes formas de risco.

- Critérios baseados em métricas de engenharia financeira: entre elas a utilização do *r-squared*, *VaR*(Value at Risk), *CVaR*(Conditional Value at Risk) e a densidade do retorno. Em especial as abordagens utilizando a função *CVaR* têm ganhado destaque graças ao aparecimento de novas abordagens que permitiram a sua aplicação em problemas com maior número de estados.

O foco deste trabalho é nos critérios baseados em métricas de engenharia financeira, em especial no critério *CVaR*.

3.3 Medidas de Risco de Engenharia Financeira

Medidas de risco têm um importante papel em otimização sob incerteza. As métricas para mensuração de risco financeiro têm sido empregadas em larga escala em diversas áreas como ferramenta para gestão de tais riscos. Grande parte dessas medidas têm origem no setor financeiro no qual a incerteza é uma constante que precisa ser gerenciada a todo momento, dado que tal incerteza afeta diretamente as decisões e resultados futuros. Entre essas medidas destacamos a função *VaR*(*Value at Risk*) e a função *CVaR*(*Conditional Value at Risk*).

Uma propriedade desejada em medidas de risco é a coerência (ARTZNER P.; HEATH, 1999), definida a seguir.

Considere um espaço de probabilidade $\mathcal{S} = (\Omega, \mathcal{F}, \mathbb{P})$, em que Ω é o espaço amostral, $\mathcal{F}$ são os eventos e $\mathbb{P}$ é chamada de medida de probabilidade. Seja $\rho : Z \rightarrow \mathbb{R}$, $\rho(Z)$ uma função que mapeia um resultado incerto Z em um número real, em que $Z \in \mathcal{Z}$ onde $\mathcal{Z}$ é o espaço de variáveis aleatórias, definido sobre o espaço de probabilidade $\mathcal{S}$. Neste trabalho Z representa o custo acumulado. Dizemos que, ρ é uma medida de risco coerente, se satisfaz as propriedades (1)-(4) a seguir (ARTZNER P.; HEATH, 1999):

1. Invariância sob translação: Esta propriedade nos diz que quando temos uma quantidade certa $|a|$ (que não muda com o tempo) podemos adicioná-la ou subtraí-la à variável aleatória Z , fazendo com que a nossa medida de risco aumente ou diminua de maneira proporcional a $|a|$.

$$\rho(Z + a) = \rho(Z) + a, \forall Z \in \mathcal{Z} \text{ e } a \in \mathbb{R}. \quad (8)$$

2. Subaditividade: A medida do risco total do sistema (da soma dos custos) é menor ou igual que a medida do risco da soma individual dos custos.

$$\rho(Z_1 + Z_2) \leq \rho(Z_1) + \rho(Z_2), \forall Z_1, Z_2 \in \mathcal{Z}. \quad (9)$$

3. Homogeneidade: O tamanho do risco aumenta proporcionalmente ao tamanho do custo.

$$\rho(\lambda Z) = \lambda \rho(Z), \forall Z \in \mathcal{Z} \quad \text{e} \quad \lambda \geq 0. \quad (10)$$

4. Monotonicidade: Se os custos envolvidos com Z_1 são menores que os de Z_2 para todos os cenários possíveis, então o risco de Z_1 é menor que o de Z_2 .

$$Z_1 \leq Z_2, \text{ então, } \rho(Z_1) \leq \rho(Z_2), \forall Z_1, Z_2 \in \mathcal{Z}. \quad (11)$$

5. Convexidade: Propriedade que é desejada em problemas de otimização, pois consegue capturar a não linearidade do valor dos custos, e é definida como:

$$\rho(\lambda Z_1 + (1 - \lambda)Z_2) \leq \lambda \rho(Z_1) + (1 - \lambda)\rho(Z_2), \forall Z_1, Z_2 \in \mathcal{Z} \text{ e } \forall \lambda \in [0, 1]. \quad (12)$$

3.4 Métricas VaR e CVaR

A função *Value at Risk* (*VaR*) é uma função muito popular utilizada para gestão de portfólio de ativos financeiros. *VaR* mede a pior perda esperada ao longo de determinado intervalo de tempo sob condições normais e dentro de determinado nível de confiança α . Com o cálculo do *VaR*, é possível saber de forma quantitativa os piores e melhores cenários que um ativo ou uma decisão pode alcançar. Logo pode ser usado como um critério para se tomar decisões que envolvem risco. *VaR* com nível de confiança $\alpha \in (0, 1)$ é definido como o quantil $1 - \alpha$ de Z , i.e.:

$$VaR_\alpha(Z) = \min\{z | F(z) \geq \alpha\}, \quad (13)$$

em que Z é uma variável aleatória e neste trabalho é interpretado como custo acumulado.

Apesar da sua ampla utilização, a função *VaR* tem certas limitações, entre as principais estão: (i) não é uma medida coerente de risco, (ii) é instável (existe alta flutuação sobre perturbações), (iii) é inapropriada quando Z não é distribuído normalmente, e (iv) não permite medir as perdas potenciais que excedem o valor do próprio *VaR*.

Uma alternativa que contorna as limitações da função *VaR* é a função *Conditional-Value-at-Risk* (*CVaR*). Essa medida indica de forma mais adequada o potencial de perdas

que ultrapassem o intervalo de confiança, definido ao se calcular a média das perdas que excedem o valor do VaR . Além disso, $CVaR$ é considerada uma medida coerente de risco, não precisa de uma distribuição normalizada para Z e por último apresenta uma maior estabilidade pois a flutuação sobre perturbações é menor. $CVaR$ pode ser definida, com nível de confiança $\alpha \in (0, 1)$ da seguinte forma (ROCKAFELLER; URYASEV, 2000):

$$CVaR_{\alpha}(Z) = \min_{w \in \mathbb{R}} \left\{ w + \frac{1}{\alpha} \mathbb{E}[(Z - w)^+] \right\}, \quad (14)$$

em que $(x)^+ = \max(x, 0)$, representa a parte positiva de x , Z é uma variável aleatória e w representa a variável de decisão que, no ponto ótimo, atinge o valor do VaR .

Com o objetivo de ilustrar e deixar mais evidente as diferenças entre VaR e $CVaR$, é mostrada a seguir uma aplicação dessas métricas para um portfólio contendo uma ação retirada do mercado financeiro Brasileiro (Petrobras PBR-A). É usado um conjunto de 3596 dados históricos dos preços de fechamento ajustado (Adj Close) diários desde 07/10/2003 até 19/01/2018¹. O objetivo é medir o risco deste portfólio. Para isso é utilizado o retorno da ação que é a variação do preço de fechamento ajustado, pois ela reflete a variação real do patrimônio da ação. Foram calculados VaR e $CVaR$ para os níveis de confiança mais comuns a saber: 95%, 99% e 99.9%. Os resultados podem ser vistos na tabela 1. Também foi montada a distribuição histórica dos retornos da ação e sinalizados os valores de VaR e $CVaR$ para o nível de confiança de 99% na figura 2. Nessa figura podemos perceber a diferença nos resultados. Essa diferença deve-se ao fato do modelo VaR focar na minimização das variâncias dos retornos dos ativos, o que resulta em um portfólio com menor variância possível. Já o modelo de $CVaR$ foca na cauda da distribuição dos retornos, ou seja, nos valores mais distantes da média.

Tabela 1 – Resultados de VaR e $CVaR$

$VaR(95\%)$	-4,76%	$CVaR(95\%)$	-7,11%
$VaR(99\%)$	-8,30%	$CVaR(99\%)$	-11,52%
$VaR(99.9\%)$	-14,78%	$CVaR(99.9\%)$	-20,32%

De forma prática podemos interpretar VaR de 2 formas:

- Como possibilidade de perda mínima, para o exemplo, $VaR(99\%)$ pode ser interpretado como: há 1% de chance para a carteira incorrer em uma perda de 8,30% do valor total da carteira em um dia de negociação.

¹ <https://finance.yahoo.com/quote/PBR-A/history>

Figura 2 – Distribuição histórica do retorno

- Como uma medida do nível de confiança, para o exemplo, $VaR(99\%)$ pode ser interpretado como: há 99% de chance de que a carteira não perca mais de 8,30% de seu valor total em um dia de negociação.

Podemos interpretar $CVaR$ como:

- A possibilidade de perda média, para o exemplo, $CVaR(99\%)$ pode ser colocado assim: há 1% de chance para a carteira incorrer em uma perda média nos piores cenários de 11,52% do valor total da carteira em um dia de negociação.
- Uma medida do nível de confiança, para exemplo, $CVaR(99\%)$ pode ser interpretado como: há 99% de chance de que a carteira perca em média até 11,52% de seu valor total em um dia de negociação.

A tabela 2 apresenta as propriedades listadas em (ARTZNER P.; HEATH, 1999) que são desejadas para medidas de risco. Note que o $CVaR$ respeita todas as propriedades listadas, ao contrário do VaR .

O $CVaR$ é uma medida de risco coerente, diferentemente de VaR , que não possui a propriedade de subaditividade. O $CVaR$ possui propriedades que facilitam o processo de

Tabela 2 – Comparativo entre VaR e $CVaR$

Propriedades	VaR	$CVaR$
1 - Invariância sob translação	ok	ok
2 - Subaditividade	x	ok
3 - Homogeneidade	ok	ok
4 - Monotonicidade	ok	ok
5 - Convexidade	x	ok
6 - Coerência	x	ok

otimização e o gerenciamento de risco, sendo assim uma medida mais apropriada que o VaR .

4 *CVaR* MDP

Em MDPs sensíveis a risco, o custo total esperado, que é neutro ao risco pode ser substituído por alguma medida de risco como *Value at Risk* (*VaR*) ou *Condicional Value at Risk* (*CVaR*). Neste trabalho, examina-se MDPs sensíveis a risco com a função objetivo *CVaR*, referido como *CVaR* MDP (CHOW et al., 2015). Como visto anteriormente, *CVaR* é uma medida coerente de risco que tem ganhado muita popularidade em aplicações financeiras e de engenharia. Esse destaque deve-se especialmente a suas propriedades matemáticas e computacionais. O trabalho de Pflug e Pichler (2015), apresenta um importante arcabouço matemático no qual são estabelecidas propriedades e teoremas que ajudaram a viabilizar a utilização da abordagem *CVaR* em problemas de tomada de decisão sequencial. Neste mesmo trabalho é demonstrado que há uma representação dual para medidas coerentes de risco, e que em especial há uma representação dual de *CVaR*. Essa nova representação dual, que é um fator de suma importância para a sua adoção em problemas de otimização, define como:

$$CVaR_\alpha(Z) = \max_{\xi \in \mathcal{U}_{CVaR}(\alpha, \mathbb{P})} \mathbb{E}_\xi[Z] \quad (15)$$

em que Z , é uma variável aleatória, que no nosso caso representa o custo acumulado, $\mathbb{E}_\xi[Z]$ é a esperança ξ -ponderada de Z , $\mathcal{U}_{CVaR}$ é chamado de envelope de risco (CHOW et al., 2015) e é representado da seguinte forma::

$$\mathcal{U}_{CVaR}(\alpha, \mathbb{P}) = \left\{ \xi : \xi(\omega) \in [0, \frac{1}{\alpha}], \int_{\omega \in \Omega} \xi(\omega) \mathbb{P}(\omega) d\omega = 1 \right\} \quad (16)$$

em que $\mathbb{P}$ é uma medida de probabilidade e $\omega \in \Omega$ (Ω é o espaço amostral).

O envelope de risco pode ser visto como um conjunto de medidas de probabilidade que fornecem alternativas para a medida de probabilidade $\mathbb{P}$. Nesse caso, a medida de desvio correspondente $\mathcal{U}_{CVaR}(\alpha, \mathbb{P})$ estima a diferença do que o agente pode esperar sob $\mathbb{P}$ e sob a pior distribuição de probabilidade. *CVaR* de Z também pode ser interpretada como a *esperança do pior caso de Z* sobe a distribuição perturbada $\xi\mathbb{P}$ (CHOW et al., 2015).

Seja $H_t = H_{t-1} \times A \times S$ o espaço de histórias até o tempo t e $H_0 = S$. Uma história $h_t \in H_t$ é da forma $h_t = (x_0, a_0, \dots, x_{t-1}, a_{t-1}, x_t)$. Seja $\Pi_{H,t}$ o conjunto de políticas dependentes da história até o tempo t , isto é $\Pi_{H,t} = \{\mu_0 : H_0 \rightarrow P(A), \mu_1 : H_1 \rightarrow P(A), \dots, \mu_t : H_t \rightarrow P(A)\}$. Seja Π_H o conjunto de todas as políticas dependentes da história.

O problema que deseja-se resolver é o seguinte (CHOW et al., 2015):

$$\min_{\mu \in \Pi_H} CVaR_y \left(\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) | s_0 = s, \mu \right), \quad (17)$$

em que $\mu = \{\mu_0, \mu_1, \dots\}$ é a sequência de políticas dependentes da histórico com ações $a_t = \mu_t(h_t)$ para $t \in \{0, 1, \dots\}$.

Em (CHOW et al., 2015) é demonstrado que fazendo a minimização do $CVaR$ nesta formulação, o decisor também garante robustez em relação a erros de modelagem.

Baseada na representação dual do $CVaR$, Pflug e Pichler (2015) apresentaram o teorema de decomposição de $CVaR$ descrito a seguir.

Teorema 1 (*Decomposição do CVaR,*) (PFLUG; PICHLER, 2015) Para qualquer $t \geq 0$, seja $Z = (Z_{t+1}, Z_{t+2}, \dots)$ a sequência de custos a partir do tempo $t+1$ em diante. O $CVaR$ considerando a política μ , obedece a seguinte decomposição:

$$CVaR_{\alpha} \left(Z | h_t, \mu \right) = \max_{\xi \in \mathcal{U}_{CVAR}(\alpha, P(\cdot | s_t, a_t))} \mathbb{E} \left[\xi(s_{t+1}) \cdot CVaR_{\alpha \xi(s_{t+1})} \left(Z | h_{t+1}, \mu \right) \middle| h_t, \mu \right], \quad (18)$$

em que h_t é o histórico até o tempo t , a_t é a ação induzida pela política $\mu_t(h_t)$, e a esperança é em relação ao estado s_{t+1} .

Note que a decomposição recursiva descrita no teorema 1, apresenta diferentes termos $CVaR$ do lado esquerdo e direito, com diferentes níveis de confiança (α no lado esquerdo e $\alpha \xi(s_{t+1})$ no lado direito).

Com base nesse teorema de decomposição, em (CHOW et al., 2015) é apresentada uma solução baseada em programação dinâmica (DP) para o problema $CVaR$ MDP. Nessa abordagem é utilizado um MDP com estados estendidos, ou seja o espaço de estados S foi incrementado com $Y = (0, 1]$, que representa o nível de confiança α do $CVaR_{\alpha}$. Assim, em (CHOW et al., 2015) é proposto o operador de Bellman $T : S \times Y \rightarrow S \times Y$ para $CVaR$, definido a seguir:

$$T[V](s, y) = \min_{a \in \mathbb{A}} \left[C(s, a) + \gamma \max_{\xi \in \mathcal{U}_{CVAR}(y, P(\cdot | s, a))} \sum_{s' \in S} \xi(s') V(s', y \xi(s')) P(s' | s, a) \right] \quad (19)$$

em que o envelope de risco $\mathcal{U}_{CVAR}$ é definido pela equação (16). Na equação (19) o agente escolhe a melhor ação ($\min_{a \in \mathbb{A}}$) que minimiza o valor enquanto que a natureza adversária

seleciona os α piores casos escolhendo os ξ s que maximizarão a expressão. Essa equação tem duas propriedades fundamentais para a tratabilidade dos problemas que são a contração e concavidade em y . Porém, esse operador de Bellman apresenta duas dificuldades: (i) nos estados aumentados, Y é contínua; e (ii) aplicar T envolve realizar a maximização sobre ξ .

Exemplo do cálculo da função valor de uma política no *CVaR* MDP

A seguir é calculada a função valor da política π_1 para o exemplo de *CVaR* MDP com 7 estados da figura 3.

Figura 3 – Política π_1 para um *CVaR* MDP com 7 estados

Fonte: (CHOW et al., 2015)

Para o cálculo da política π_1 foi utilizado *CVaR* com nível de confiança $\alpha = 0.6$, ou seja o interesse está no cálculo da média dos 60% piores casos para essa política.

Para realizar esse cálculo identifica-se os caminhos dessa política que podem causar 60% dos maiores custos. O primeiro passo é olhar para a ramificação direita de s_0 no qual obtêm-se os 50% dos piores custos. O valor obtido na ramificação direita é $0.5 * V(s_2, 1)$ que é igual a $0.5 * 12$. Em seguida identifica-se o 10% faltantes dos piores custos da ramificação esquerda. Uma vez que o custo $C(s_5, a_6)$ é o maior da ramificação esquerda e a probabilidade

$P(s_1|s_0, \pi_1(s_0)) * P(s_5|s_1, \pi_1(s_1)) * P(s_g|s_5, \pi_1(s_5)) = 0.1667 > 0.1$, é considerado o estado s_5 apenas. O valor obtido na ramificação esquerda é $0.1 * V(s_5, 1)$ que é igual a $0.1 * 8$.

$$V^{\pi_1}(s_0, \alpha) = \frac{P(s_2|s_0, \pi_1(s_0))V(s_2, 1) + (\alpha - (P(s_2|s_0, \pi_1(s_0))))V(s_5, 1)}{\alpha}$$

$$V^{\pi_1}(s_0, 0.6) = \frac{0.5(12) + 0.1(8)}{0.6} = 11.33$$

Da mesma forma pode ser calculado:

$$V^{\pi_1}(s_1, 0.6) = \frac{\frac{1}{3} * 8 + ((0.6 - \frac{1}{3}) * 6)}{0.6} = 7.11$$

$$V^{\pi_1}(s_2, 0.6) = \frac{0.6 * 12}{0.6} = 12$$

$$V^{\pi_1}(s_3, 0.6) = \frac{0.6 * 4}{0.6} = 4$$

$$V^{\pi_1}(s_4, 0.6) = \frac{0.6 * 6}{0.6} = 6$$

$$V^{\pi_1}(s_5, 0.6) = \frac{0.6 * 8}{0.6} = 8$$

Veja que o resultado é o mesmo utilizando a equação (19). Por exemplo, para s_0 , $\gamma = 1$ e $y = 0.6$ temos:

$$V(s_0, 0.6) = \min_{a \in \mathbb{A}} \left[C(s_0, a) + \gamma \max_{\xi \in \mathcal{U}_{CVAR}(y, P(\cdot | s_0, a))} \sum_{s' \in S} \xi(s') V(s', y\xi(s')) P(s' | s_0, a) \right]$$

$$V(s_0, 0.6) = \left[0 + \max_{\xi \in \mathcal{U}_{CVAR}(y, P(\cdot | s_0, a_1))} \xi(s_1) V(s_1, 0.6\xi(s_1)) 0.5 + \xi(s_2) V(s_2, 0.6\xi(s_2)) 0.5 \right]$$

Assim o problema de maximização a ser resolvido usando um resolvidor para programação linear é:

$$\max : \quad \xi(s_1)V(s_1, 0.6\xi(s_1))0.5 + \xi(s_2)V(s_2, 0.6\xi(s_2))0.5 \quad (20)$$

sujeito a :

$$0 \leq \xi(s_1) \leq \frac{1}{0.6}$$

$$0 \leq \xi(s_2) \leq \frac{1}{0.6}$$

$$\xi(s_1) * P(s_1|s_0, a_1) + \xi(s_2) * P(s_2|s_0, a_1) = 1$$

Os valores de ξ que maximizam a equação 20 são:

$$\begin{aligned} \xi(s_2) &= \frac{1}{0.6} \\ \xi(s_1) &= \frac{1}{3} = 0.33 \end{aligned}$$

Uma vez que $V(s_2, 1) = 12$ e $V(s_1, 0.2) = 0.2 * 8/0.2 = 8$, substituindo esses valores na equação 21, obtem-se:

$$V(s_0, 0.6) = \left[0.33 * V(s_1, 0.2)0.5 + \frac{1}{0.6} * V(s_2, 1)0.5 \right] = 1.33 + 10 = 11.33 \quad (21)$$

4.1 Solução ótima para CVaR MDP

A solução ótima para CVaR MDP é não-markoviana e pode ser obtida com um estado estendido que considera uma variável de estado y que representa um nível de confiança α_t rastreável ao longo do processo. Dessa forma, a solução ótima não só especifica uma função ótima no espaço de estados e nível de confiança, isto é, $V : S \times Y \rightarrow \mathbb{R}$, como também um processo $y_0 = \alpha, y_1, y_2 \dots$, regido por:

$$y_t = \xi(s_{t-1}, s_t)y_{t-1},$$

sendo necessário definir a função $\xi : S \times S \rightarrow \mathbb{R}$. Na figura 4 é ilustrado como uma política ótima para CVaR MDP se relaciona com os estados s e nível de confiança y para definir os próximos estados.

Tabela 3 – Valores obtidos na iteração 1 do algoritmo *CVaRVILI* para o exemplo da figura 3.

t=1	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}
s_0	0	0	0	0	0	0	0	0	0	0	0
s_1	0	0	0	0	0	0	0	0	0	0	0
s_2	12	12	12	12	12	12	12	12	12	12	12
s_3	4	4	4	4	4	4	4	4	4	4	4
s_4	6	6	6	6	6	6	6	6	6	6	6
s_5	8	8	8	8	8	8	8	8	8	8	8
s_g	0	0	0	0	0	0	0	0	0	0	0

Uma vez que $I_{s'}[V](y\xi(s'))$ é a interpolação linear de $yV(s', y\xi(s'))$, na equação 19 podemos substituir $V(s', y\xi(s'))$ por $\frac{I_{s'}[V](y\xi(s'))}{y}$, obtendo o operador de Bellman Interpolado T_I , como mostrado a seguir (CHOW et al., 2015):

$$T_I[V](s, y) = \min_{a \in \mathbb{A}} \left[C(s, a) + \gamma \max_{\xi \in \mathcal{U}_{CVAR}(y, P(\cdot|s, a))} \sum_{s' \in S} \frac{I_{s'}[V](y\xi(s'))}{y} P(s'|s, a) \right]. \quad (23)$$

O algoritmo 3 tem como entrada um MDP (S, A, P, C, γ) e um conjunto Y de pontos de interpolação que são escolhidos pelo especialista que está modelando o problema *CVaR* MDP de modo a balancear o desempenho e o menor erro de aproximação desejado. Note que quanto maior o número de pontos de interpolação houver, menor será o erro de aproximação. Na linha 4, $V^0(s, y)$ é inicializada com o menor custo. Em cada iteração t , o valor de V^t é atualizado até que $\max_{s, y} |V^t(s, y) - V^{t-1}(s, y)| < \frac{\epsilon(1-\gamma)}{2\gamma}$. Caso esse limite seja alcançado o algoritmo interrompe a execução e retorna a função valor ótima V^* . Na linha 11 é calculado $g(s, y, a)$ para cada ação tripla (s, y, a) , para isso é calculada primeiro a interpolação linear de $yV(s', y\xi(s')) = I_{s'}[V](y\xi(s'))$ e depois é chamado um solver de otimização linear (e.g. CPLEX) para maximizar a expressão considerando o envelope de risco. Na linha 12, com a solução encontrada pelo solver calculamos $Q^t(s, y, a)$. Na linha 14 a função valor é atualizada com o menor valor $Q^t(s, y, a)$ para o estado estendido (s, y) . Por fim, uma política estacionária ótima pode ser encontrada a partir de $V^*(s, y)$ calculada por esse algoritmo.

Tabela 4 – Valores obtidos na iteração 2 do algoritmo *CVaRVILI* para o exemplo da figura 3.

t=2	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}
s_0	12	12	12	12	12	12	12	12	12	10.0086	6
s_1	8	8	8	8	8	8	8	8	7.8532	7.1110	6
s_2	12	12	12	12	12	12	12	12	12	12	12
s_3	4	4	4	4	4	4	4	4	4	4	4
s_4	6	6	6	6	6	6	6	6	6	6	6
s_5	8	8	8	8	8	8	8	8	8	8	8
s_g	0	0	0	0	0	0	0	0	0	0	0

Tabela 5 – Valores obtidos na iteração 3 do algoritmo *CVaRVILI* para o exemplo da figura 3.

t=3	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}
s_0	12	12	12	12	12	12	12	12	12	11.3362	9
s_1	8	8	8	8	8	8	8	8	7.8532	7.1110	6
s_2	12	12	12	12	12	12	12	12	12	12	12
s_3	4	4	4	4	4	4	4	4	4	4	4
s_4	6	6	6	6	6	6	6	6	6	6	6
s_5	8	8	8	8	8	8	8	8	8	8	8
s_g	0	0	0	0	0	0	0	0	0	0	0

Por exemplo, considere o CVaR MDP da figura 3, $\gamma = 1$, $\epsilon = 0.001$ e 11 pontos de interpolação $Y = \{y_1=0, y_2=0.01, y_3=0.0167, y_4=0.0278, y_5=0.0464, y_6=0.0774, y_7=0.1292, y_8=0.2154, y_9=0.3594, y_{10}=0.5995, y_{11}=1\}$. O valor $V^t(s, y)$ calculado em cada uma das três primeiras iterações pelo algoritmo *CVaRVILI* são mostradas nas tabelas 3, 4 e 5. Uma vez que os valores da quarta iteração são as mesmas que da terceira iteração, o algoritmo para.

Cálculo de $g(s, y, a)$ no algoritmo *CVaRVILI*

Para calcular $Q(s, y, a)$ é necessário calcular $g(s, y, a)$ (linha 11 do algoritmo 3) que implica em resolver um problema de maximização o qual é equivalente ao seguinte problema de minimização:

$$\begin{aligned}
 \min : & - \sum_{s' \in S} (\xi(s') + x(s')) \\
 \text{sujeito a :} \\
 -\xi(s') & \frac{y_{i+1}V(s', y_{i+1}) - y_iV(s', y_i)}{y_{i+1} - y_i} P(s'|s, a) + x(s') \leq \frac{1}{y} \left[\left(y_iV(s', y_i) - \frac{y_{i+1}V(s', y_{i+1}) - y_iV(s', y_i)}{y_{i+1} - y_i} \right) y_i P(s'|s, a) \right], \forall s' \in S
 \end{aligned}$$

Algoritmo 3 *CVaR* Value Iteration with Linear Interpolation (*CVaRVILI*)

```

1: Entrada: Um MDP( $\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{C}, \gamma$ ) e  $\mathbf{Y}(\mathbf{s}) = \{y_1, y_2, \dots, y_{N(\mathbf{s})}\} \in [0, 1]^{N(\mathbf{s})}$ 
2: Saída: Função valor ótima  $\mathbf{V}$ 
3: para cada  $\mathbf{s} \in \mathbf{S}$  faça
4:    $V^0(\mathbf{s}, y) \leftarrow \min_{a \in A} C(\mathbf{s}, a), \forall y$ 
5: fim
6:  $t \leftarrow 1$ ;
7: enquanto critério de parada não satisfeito faça
8:   para cada  $\mathbf{s} \in \mathbf{S}$  faça
9:     para cada  $y \in \mathbf{Y}$  faça
10:      para cada  $a \in A$  faça
11:
12:          $g(\mathbf{s}, y, a) = \max_{\xi \in \mathcal{U}_{CVAR}(y, P(\cdot|\mathbf{s}, a))} \sum_{s' \in \mathbf{s}} \frac{I_{s'}[V^{t-1]}(y\xi(s'))}{y} P(s'|\mathbf{s}, a)$ 
13:       fim
14:       $Q^t(\mathbf{s}, y, a) = C(\mathbf{s}, a) + \gamma g(\mathbf{s}, y, a)$ 
15:    fim
16:  fim
17:   $t \leftarrow t + 1$ ;
18: fim
19: devolva  $\mathbf{V}$ ;

```

 Fonte: Yinlam Chow, 2015 (CHOW et al., 2015)

Para chegar nas restrições (sem incluir as variáveis novas $x(s')$) desse problema de minimização foram realizadas as seguintes manipulações matemáticas a partir da equação 22:

$$\begin{aligned}
 & I_{s'}[V](y\xi(s'))P(s'|\mathbf{s}, a) \geq 0 \\
 & y_i V(s', y_i) + \left(\frac{y_{i+1}V(s', y_{i+1}) - y_i V(s', y_i)}{y_{i+1} - y_i} \right) (y\xi(s') - y_i) P(s'|\mathbf{s}, a) \geq 0 \\
 & -y_i V(s', y_i) - y\xi(s')P(s'|\mathbf{s}, a) \left(\frac{y_{i+1}V(s', y_{i+1}) - y_i V(s', y_i)}{y_{i+1} - y_i} \right) + y_i P(s'|\mathbf{s}, a) \left(\frac{y_{i+1}V(s', y_{i+1}) - y_i V(s', y_i)}{y_{i+1} - y_i} \right) \leq 0 \\
 & y \left[-\xi(s') \frac{y_{i+1}V(s', y_{i+1}) - y_i V(s', y_i)}{y_{i+1} - y_i} P(s'|\mathbf{s}, a) \right] - \left[\left(y_i V(s', y_i) - \frac{y_{i+1}V(s', y_{i+1}) - y_i V(s', y_i)}{y_{i+1} - y_i} \right) y_i P(s'|\mathbf{s}, a) \right] \leq 0 \\
 & -\xi(s') \frac{y_{i+1}V(s', y_{i+1}) - y_i V(s', y_i)}{y_{i+1} - y_i} P(s'|\mathbf{s}, a) \leq \frac{1}{y} \left[\left(y_i V(s', y_i) - \frac{y_{i+1}V(s', y_{i+1}) - y_i V(s', y_i)}{y_{i+1} - y_i} \right) y_i P(s'|\mathbf{s}, a) \right]
 \end{aligned}$$

Por exemplo, considere o CVaR MDP da figura 3, $\gamma = 1$ e 11 pontos de interpolação $\mathbf{Y} = \{y_1=0, y_2=0.01, y_3=0.0167, y_4=0.0278, y_5=0.0464, y_6=0.0774, y_7=0.1292, y_8=0.2154, y_9=0.3594, y_{10}=0.5995, y_{11}=1\}$. A seguir é mostrado o problema de otimização linear enviado ao CPLEX pelo algoritmo *CVaRVILI* para calcular $g(s_1, 0.6, a_1)$. Note que, uma

vez que os sucessores de s_0 são s_1 e s_2 , temos 2 restrições para cada par de pontos de interpolação consecutivos, além das restrições relacionadas com as variáveis ξ_i e x_i .

Minimizar: $-\xi_1 - \xi_2 - x_1 - x_2$

Sujeito a:

interpolação entre y_1 e y_2

$$\begin{aligned} -\xi_1 \frac{y_2 V(s_1, y_2) - y_1 V(s_1, y_1)}{y_2 - y_1} P(s_1 | s_0, a_1) + x_1 &\leq \frac{1}{0.6} \left[y_1 V(s_1, y_1) - \frac{y_2 V(s_1, y_2) - y_1 V(s_1, y_1)}{y_2 - y_1} y_1 P(s_1 | s_0, a_1) \right] \\ -\xi_2 \frac{y_2 V(s_2, y_2) - y_1 V(s_2, y_1)}{y_2 - y_1} P(s_2 | s_0, a_1) + x_2 &\leq \frac{1}{0.6} \left[y_1 V(s_2, y_1) - \frac{y_2 V(s_2, y_2) - y_1 V(s_2, y_1)}{y_2 - y_1} y_1 P(s_2 | s_0, a_1) \right] \end{aligned}$$

interpolação entre y_2 e y_3

$$\begin{aligned} -\xi_1 \frac{y_3 V(s_1, y_3) - y_2 V(s_1, y_2)}{y_3 - y_2} P(s_1 | s_0, a_1) + x_1 &\leq \frac{1}{0.6} \left[y_2 V(s_1, y_2) - \frac{y_3 V(s_1, y_3) - y_2 V(s_1, y_2)}{y_3 - y_2} y_2 P(s_1 | s_0, a_1) \right] \\ -\xi_2 \frac{y_3 V(s_2, y_3) - y_2 V(s_2, y_2)}{y_3 - y_2} P(s_2 | s_0, a_1) + x_2 &\leq \frac{1}{0.6} \left[y_2 V(s_2, y_2) - \frac{y_3 V(s_2, y_3) - y_2 V(s_2, y_2)}{y_3 - y_2} y_2 P(s_2 | s_0, a_1) \right] \end{aligned}$$

interpolação entre y_3 e y_4

$$\begin{aligned} -\xi_1 \frac{y_4 V(s_1, y_4) - y_3 V(s_1, y_3)}{y_4 - y_3} P(s_1 | s_0, a_1) + x_1 &\leq \frac{1}{0.6} \left[y_3 V(s_1, y_3) - \frac{y_4 V(s_1, y_4) - y_3 V(s_1, y_3)}{y_4 - y_3} y_3 P(s_1 | s_0, a_1) \right] \\ -\xi_2 \frac{y_4 V(s_2, y_4) - y_3 V(s_2, y_3)}{y_4 - y_3} P(s_2 | s_0, a_1) + x_2 &\leq \frac{1}{0.6} \left[y_3 V(s_2, y_3) - \frac{y_4 V(s_2, y_4) - y_3 V(s_2, y_3)}{y_4 - y_3} y_3 P(s_2 | s_0, a_1) \right] \end{aligned}$$

...

interpolação entre y_4 e y_5

...

interpolação entre y_5 e y_6

...

interpolação entre y_6 e y_7

...

interpolação entre y_7 e y_8

...

interpolação entre y_8 e y_9

...

interpolação entre y_9 e y_{10}

...

interpolação entre y_{10} e y_{11}

...

Limites:

$$\begin{aligned} 0 &\leq \xi_1, \xi_2 \leq \frac{1}{0.6} \\ -1000000 &\leq x_1, x_2 \leq 1000000 \end{aligned}$$

5 Algoritmo Exato Para Avaliar uma Política

Embora o algoritmo *CVaRVILI* possa obter uma política ótima aproximada, ele possui um alto custo computacional devido a necessidade de resolver repetidas vezes problemas de programação linear. Por outro lado, o valor de uma política estacionária pode ser encontrado com custo computacional similar ao de um algoritmo de iteração de valor para MDP neutros ao risco. Nesta seção é proposto o algoritmo *PECVaR* (*Policy Evaluation CVaR*), que avalia uma política de forma exata, desde que ela seja estacionária e markoviana. O algoritmo *PECVaR* baseia-se no Teorema 2, em que $P_G^T = \Pr(s_T \in G|\pi)$ é a probabilidade de alcançar a meta em pelo menos T passos.

Intuitivamente, o Teorema 2 indica que o o valor de uma política π para $\alpha = 1 - P_G^T$ considerando T passos pode ser calculada considerando a diferença entre o valor global ($\mathbb{E}[Z]$) e o valor esperado dos melhores casos (os que conseguiram alcançar a meta em pelo menos T passos) dividido pela probabilidade de não alcançar a meta em pelo menos T passos. Esse cálculo pode ser feito de maneira iterativa, identificando em cada passo $T = \{0, 1, 2, \dots\}$ qual é a probabilidade de chegar na meta em pelo menos T passos (ver figura 5).

Figura 5 – Intuição do Algoritmo PECVaR

Teorema 2 *Considere uma política estacionária π e um MDP com custo constante, sem perda de generalidade com custo 1, tem-se que:*

$$CVaR_{\alpha=(1-P_G^T)}(Z|\pi) = \frac{\mathbb{E}[Z] - \mathbb{E}\left[Z|Z \leq \sum_{t=0}^{T-1} \gamma^t\right] P_G^T}{1 - P_G^T}, \quad (24)$$

Demonstração. Note que, se a variável aleatória Z é contínua, a seguinte identidade é verdadeira (CHOW, 2017):

$$CVaR_{\alpha}(Z) = \mathbb{E}[Z|Z > VaR_{\alpha}(Z)] \quad (25)$$

e utilizando a Lei da Probabilidade Total, tem-se que:

$$\begin{aligned} \mathbb{E}[Z] = & \mathbb{E}[Z|Z > VaR_{\alpha}(Z)] \Pr(Z > VaR_{\alpha}(Z)) \\ & + \mathbb{E}[Z|Z \leq VaR_{\alpha}(Z)] \Pr(Z \leq VaR_{\alpha}(Z)) \end{aligned} \quad (26)$$

e implica em:

$$\mathbb{E}[Z|Z > VaR_{\alpha}(Z)] = \frac{\mathbb{E}[Z] - \mathbb{E}[Z|Z \leq VaR_{\alpha}(Z)] \Pr(Z \leq VaR_{\alpha}(Z))}{\Pr(Z > VaR_{\alpha}(Z))}. \quad (27)$$

No caso em que o custo é unitário, tem-se que:

$$\Pr\left(Z \leq \sum_{t=0}^{T-1} \gamma^t\right) = P_G^T \quad (28)$$

e

$$VaR_{\alpha=(1-P_G^T)}(Z) = \sum_{t=0}^{T-1} \gamma^t. \quad (29)$$

Juntando as equações 25, 27, 28 e 29, obtém-se o resultado do teorema. $\diamond$

Na linha 7 o algoritmo *PECVaR* (Algoritmo 4) calcula $\mathbb{E}[Z]$ representado por $V(s)$ e na linha 8 inicializa $CVaR(s, 0)$ com a pior história considerando a política π . O algoritmo utiliza a equação 24 na linha 18 para calcular o valor de todos os estados $s \in S$, considerando cada um deles como estado inicial e para $\alpha = 1 - P_G^T$ em cada iteração. No algoritmo $\mathbb{E}\left[Z|Z \leq \sum_{t=0}^{T-1} \gamma^t\right]$ é representado por $V_{\leq C}^T$.

O algoritmo *PECVaR* itera em valores de $T = \{0, 1, 2, \dots\}$ enquanto $1 - P_G^T$ seja menor que *minAlpha*. Em cada iteração são atualizados:

- A probabilidade de chegar em s' em T passos dado que o estado inicial é s e é usada a política π , isto é $\Pr(s_T = s' | s_0 = s, \pi)$. Para tal, é usada a probabilidade de chegar

nos estados $i \in S$ em $T - 1$ passos. Assim, a seguinte atualização é feita na linha 21 do algoritmo:

$$\Pr(s_T = s' | s_0 = s, \pi) = \sum_{i \in S} P(s' | i, \pi(s)) \Pr(s_{T-1} = i | s_0 = s, \pi) \quad (30)$$

No algoritmo $\Pr(s_T = s' | s_0 = s, \pi)$ é representado por $P_{s'}^T$ e $\Pr(s_{T-1} = i | s_0 = s, \pi)$ é representado por P_i^{T-1} .

- A probabilidade de alcançar algum dos estados meta em pelo menos T passos começando no estado s , isto é $P_G^T(s)$ lembrando que o estado meta é absorvente. Para tal, é utilizada $\Pr(s_T = s' | s_0 = s, \pi)$. Assim, a seguinte atualização é feita na linha 23 do algoritmo:

$$P_G^T(s) = \sum_{s' \in G} \Pr(s_T = s' | s_0 = s, \pi) \quad (31)$$

- O valor $\mathbb{E} \left[Z | Z \leq \sum_{t=0}^{T-1} \gamma^t, s_0 = s, \pi \right]$ (representado no algoritmo por $V_{\leq C}^T$), somando o valor esperado dos melhores casos para $T - 1$ passos (isto é os que alcançaram a meta em pelo menos $T-1$ passos) com o valor esperado dos novos estados que alcançaram a meta em T passos e dividindo esse resultado pela probabilidade de alcançar a meta em pelo menos T passos, isto é:

$$\mathbb{E} \left[Z | Z \leq \sum_{t=0}^{T-1} \gamma^t, s_0 = s \right] = \frac{\mathbb{E} \left[Z | Z \leq \sum_{t=0}^{T-2} \gamma^t, s_0 = s \right] P_G^{T-1}(s) + \sum_{t=0}^{T-1} \gamma^t (P_G^T(s) - P_G^{T-1}(s))}{P_G^T(s)}. \quad (32)$$

Essa atualização é feita na linha 25, em que $\sum_{t=0}^{T-1} \gamma^t$ é representado por C^T .

- O custo acumulado, C^T , é atualizado na linha 24 utilizando:

$$C^T \leftarrow C^{T-1} + \gamma^{T-1} \quad (33)$$

Uma vez que em cada iteração do algoritmo *PECVaR* é calculado o valor de um α específico, o algoritmo percorre apenas alguns valores de α . Porém, valores que não foram percorridos podem também ser calculadas de forma exata, basta ponderar o valor *CVaR* calculado com o melhor custo acumulado considerado (linha 28 e 29).

Considere o CVaR MDP da figura 6, $\gamma = 1$ e 3 pontos $Y = \{ y_1=0.5, y_2=0.6, y_3=1 \}$ para os quais serão calculados os valores e $\min Alpha = 0.1$. Considerando que $V(s_0) = 2.5$, os valores calculados nas linhas 13-30 para o estado s_0 pelo algoritmo

Algoritmo 4 Policy Evaluation $CVaR$ (PECVaR)

```

1: Entrada: Um  $MDP(S, A, P, C, \gamma)$ ,  $Y(s) = \{y_1, y_2, \dots, y_{N(s)}\} \in [0, 1]^{N(s)}$ ,
    $maxIter$ ,  $minAlpha$ ,  $G$ , e  $\pi$ 
2: Saída: função valor  $CVaR$   $V_{CVaR}^\pi$ 
3:  $i \leftarrow 1$ ;
4:  $V(s) \leftarrow 0, CVaR(s, 0) \leftarrow 0, \forall s \in S$ ;
5: enquanto  $i < maxIter$  faça
6:   para cada  $s \in S$  faça
7:      $V(s) \leftarrow C(s, \pi(s)) + \gamma \sum_{s'} V(s') P(s'|s, \pi(s))$ 
8:      $CVaR(s, 0) \leftarrow C(s, \pi(s)) + \gamma \max_{s' \in \{x | P(x|s, \pi(s)) > 0\}} CVaR(s', 0)$ 
9:   fim
10:   $i \leftarrow i + 1$ 
11: fim
12: para cada  $s \in S$  faça
13:    $T \leftarrow 0$ 
14:    $P_G^0, C^0, V_{\leq C}^0 \leftarrow 0$ 
15:    $P_s^0 \leftarrow 1$ 
16:    $P_{s'}^0 \leftarrow 0 \forall s' \neq s \in S$ 
17:   enquanto  $1 - P_G^T \geq minAlpha$  faça
18:      $CVaR(s, 1 - P_G^T) \leftarrow \frac{V(s) - V_{\leq C}^T \times P_G^T}{1 - P_G^T}$ 
19:      $T \leftarrow T + 1$ 
20:     para cada  $s' \in S$  faça
21:        $P_{s'}^T \leftarrow \sum_{i \in S} P(s'|i, \pi(s)) P_i^{T-1}$ 
22:     fim
23:      $P_G^T \leftarrow \sum_{s' \in G} P_{s'}^T$ 
24:      $C^T \leftarrow C^{T-1} + \gamma^{T-1}$ 
25:      $V_{\leq C}^T \leftarrow \frac{V_{\leq C}^{T-1} \times P_G^{T-1} + C^T \times (P_G^T - P_G^{T-1})}{P_G^T}$ 
26:   fim
27:   para cada  $y \in Y$  faça
28:      $T \leftarrow \min\{t | y \geq 1 - P_G^T\}$ 
29:      $V_{CVaR}^\pi(s, y) \leftarrow \frac{(1 - P_G^T) \times CVaR(s, 1 - P_G^T) + (y - (1 - P_G^T)) \times C^T}{y}$ 
30:   fim
31: fim
32: devolva  $V_{CVaR}^\pi$ ;

```

PECVaR são mostradas na tabela 6. Para esse problema, a partir de s_0 , chega-se na meta em: (i) 2 passos com probabilidade 0.5 e custo 2 (ver $P_G^{T=2}$ e $C^{T=2}$, respectivamente); e (ii) em 3 passos com probabilidade 1 e custo 3 (ver $P_G^{T=3}$ e $C^{T=3}$, respectivamente). Assim, o algoritmo *PECVaR* calcula primeiro $CVaR(s_0, 1)$ e depois $CVaR(s_0, 0.5)$, mas não calcula $CVaR(s_0, 0.6)$. Esse último valor é calculado nas linhas 27-30 do algoritmo. Após a execução dessas linhas, obtém-se:

$$\begin{aligned}
 V_{CVaR}^\pi(s_0, 0.5) &= \frac{0.5 * 3}{0.5} = 3 \\
 V_{CVaR}^\pi(s_0, 0.6) &= \frac{(1 - P_G^{T=2}) \times CVaR(s, 1 - P_G^{T=2}) + (0.6 - (1 - P_G^{T=2})) \times C^{T=2}}{0.6} \\
 V_{CVaR}^\pi(s_0, 0.6) &= \frac{0.5 * 3 + 0.1 * 2}{0.6} = 2.83 \\
 V_{CVaR}^\pi(s_0, 1) &= \frac{1 * 2.5}{1} = 2.5
 \end{aligned}$$

Figura 6 – Política π_1 para um *CVaR* MDP com 7 estados

Fonte: (CHOW et al., 2015)

Tabela 6 – Valores obtidos nas iterações do algoritmo *PECVaR* para o estado s_0 do exemplo da figura 6.

	P_G^T	C^T	$V_{\leq C}^T$	$P_{s'}^T$	$CVaR(s, 1 - P_G^T)$
T=0	0	0	0		$\frac{2.5-0}{1}=2.5$
s_0				1	
s_1				0	
s_2				0	
s_3				0	
s_4				0	
s_5				0	
s_g				0	
T=1	0	1	0		$\frac{2.5-0}{1}=2.5$
s_0				0	
s_1				0.5	
s_2				0.5	
s_3				0	
s_4				0	
s_5				0	
s_g				0	
T=2	0.5	2	$\frac{2*0.5}{0.5}=2$		$\frac{2.5-2*0.5}{0.5}=3$
s_0				0	
s_1				0	
s_2				0	
s_3				$0.5*\frac{1}{3}=0.1667$	
s_4				$0.5*\frac{1}{3}=0.1667$	
s_5				$0.5*\frac{1}{3}=0.1667$	
s_g				0.5	
T=3	1	3	$\frac{2*0.5+3*0.5}{1}=2.5$		-
s_0				0	
s_1				0	
s_2				0	
s_3				0	
s_4				0	
s_5				0	
s_g				1	

6 Algoritmos baseados em Múltiplas Políticas para CVaR MDPs: PECVaR-MIN e MPCVaR

Nesta seção são apresentados dois algoritmos: o algoritmo *PECVaRMIN* e o algoritmo *MPCVaR*. Ambos algoritmos utilizam o algoritmo PECVaR para avaliar um conjunto de políticas estacionárias Π e encontrar uma solução para um MDP CVaR. O conjunto Π pode ser visto como uma biblioteca de políticas estacionárias e deve representar o espectro inteiro de risco. Enquanto o algoritmo *PECVaRMIN* devolve uma política estacionária, o algoritmo *MPCVaR* devolve uma política não estacionária.

6.1 Algoritmo PECVaRMIN

O algoritmo *PECVaRMIN* utiliza um conjunto de políticas Π e um nível de confiança arbitrário α e simplesmente calcula o valor mínimo desse conjunto de políticas. Para avaliar cada uma das políticas estacionárias $\pi \in \Pi$ é usado o algoritmo *PECVaR* o qual calcula V^π para cada política π . Para cada estado $s \in S$ é escolhida a melhor ação entre todos os pares estado $s \in S$ e nível de confiança α . Assim, o algoritmo *PECVaRMIN* devolve uma política estacionária $\pi(s)$. A figura 7 ilustra o algoritmo *PECVaRMIN* para k políticas diferentes.

Figura 7 – Algoritmo PECVaRMIN

6.2 Algoritmo MPCVaR

Dado um conjunto de políticas Π e um nível de confiança arbitrário, utilizando o algoritmo PECVaR, pode-se determinar a política estacionária $\pi_{\alpha, \Pi}$ que obtém o melhor valor CVaR simplesmente por:

$$\pi_{\alpha, \Pi} = \arg \min_{\pi \in \Pi} CVaR_{\alpha}(Z|\pi). \quad (34)$$

No entanto, uma vez que a solução ótima é não-markoviana, essa abordagem pode gerar um valor CVaR longe do ótimo.

Por outro lado, se um processo para α_t é definido, pode-se construir uma política não-markoviana. O seguinte teorema pode ser utilizado como uma opção para o processo para α_t .

Teorema 3 *Considere uma política estacionária π , estados arbitrários $s, s' \in S$, um tempo T , a probabilidade de alcançar algum dos estados meta em pelo menos T passos começando no estado s é:*

$$P_G^T(s) = \sum_{s' \in G} \Pr(s_T = s' | s_0 = s, \pi) \quad (35)$$

e a probabilidade de não alcançar a meta em pelo menos T passos e que o estado no primeiro estágio seja s' , começando no estado s é:

$$P_{-G}^T(s, s') = \sum_{s'' \in S \setminus G} \Pr(s_T = s'', s_1 = s' | s_0 = s, \pi) \quad (36)$$

O valor CVaR de uma política π nos pontos $(s, y = 1 - P_G^T(s))$ pode ser decomposto da seguinte forma:

$$V_{CVaR}^{\pi}(s, y = 1 - P_G^T(s)) = 1 + \gamma \sum_{s' \in S} \xi(s, s') V^{\pi}(s', y \xi(s, s')) \Pr(s_1 = s' | s_0 = s), \quad (37)$$

em que $\xi(s, s') = \frac{P_{-G}^T(s, s')}{\Pr(s_1 = s' | s_0 = s)} \frac{1}{1 - P_G^T(s)}$

Demonstração. Considere $C^T = \sum_{t=0}^{T-1} \gamma^t$ para simplificar a notação e a identidade $CVaR_{(\alpha=1-P_G^T)}(Z) = \mathbb{E}[Z|Z > VaR_{(\alpha=1-P_G^T)}(Z)]$, então:

$$\begin{aligned}
V_{CVaR}^\pi(s, y = 1 - P_G^T(s)) &= CVaR_{(\alpha=1-P_G^T)}(Z) \\
&= \mathbb{E}[Z|Z > VaR_{(\alpha=1-P_G^T)}(Z)] \\
&= \mathbb{E}[Z|Z > C^T, s_0 = s] \\
&= \sum_z z \Pr(Z = z|Z > C^T, s_0 = s) \\
&= \sum_{s' \in S} \sum_z z \Pr(Z = z, s_1 = s'|Z > C^T, s_0 = s) \\
&= \sum_{s' \in S} \sum_z z \Pr(Z = z|Z > C^T, s_1 = s', s_0 = s) \Pr(s_1 = s'|Z > C^T, s_0 = s) \\
&= \sum_{s' \in S} [1 + \gamma V_{CVaR}^\pi(s', \frac{P_{-G}^T(s, s')}{Pr(s_1 = s'|s_0 = s)})] \Pr(s_1 = s'|Z > C^T, s_0 = s) \\
&= \sum_{s' \in S} [1 + \gamma V_{CVaR}^\pi(s', \frac{P_{-G}^T(s, s')}{Pr(s_1 = s'|s_0 = s)})] \frac{\Pr(s_1 = s', Z > C^T|s_0 = s)}{\Pr(Z > C^T|s_0 = s)} \\
&= \sum_{s' \in S} [1 + \gamma V_{CVaR}^\pi(s', \frac{P_{-G}^T(s, s')}{Pr(s_1 = s'|s_0 = s)})] \frac{P_{-G}^T(s, s')}{1 - P_G^T(s)}
\end{aligned} \tag{38}$$

e utilizando a definição de ξ e y obtém-se o resultado. $\diamond$

Note que o algoritmo *PECVaR* deve ser modificado para calcular e armazenar os valores de ξ pois essa informação é necessária para calcular os próximos valores de y , isso é feito incluindo os seguintes cálculos antes da linha 19 do algoritmo *PECVaR*:

$$P_{-G}^T(s, s') = \sum_{s'' \in S \setminus G} \Pr(s_T = s'', s_1 = s'|s_0 = s, \pi) \tag{39}$$

$$\xi(s, s') = \frac{P_{-G}^T(s, s')}{Pr(s_1 = s'|s_0 = s)} \frac{1}{1 - P_G^T(s)} \tag{40}$$

$$\xi(s_0, s_1) = 2 \tag{41}$$

Note que para calcular $P_{-G}^T(s, s')$ é necessário ter a probabilidade $\Pr(s_T = s'', s_1 = s'|s_0 = s, \pi)$ que deve ser calculada para todo $s'' \in S \setminus G$ da seguinte forma:

$$\Pr(s_T = s'', s_1 = s'|s_0 = s, \pi) = \sum_{i \in S \setminus G} P(s''|i, \pi) \Pr(s_{T-1} = i, s_1 = s'|s_0 = s, \pi). \tag{42}$$

Na tabela 7 são mostrados os valores de P_{-G}^T e ξ obtidos nas iterações $T=1$, $T=2$ e $T=3$ do algoritmo *PECVaR* modificado para o estado s_0 do exemplo da figura 6.

O algoritmo Multi-Policy *CVaR* (*MPCVaR*) simplesmente chama o algoritmo *PECVaR* modificado para avaliar cada uma das políticas estacionárias $\pi \in \Pi$, dadas como entrada. As saídas são V^π e ξ^π para cada política π . Uma vez que foi feita a avaliação das políticas, para simular a política não-markoviana, pode ser executado o algoritmo 5. O

Tabela 7 – Valores de P_{-G}^T e ξ obtidos pelo algoritmo *PECVaR* modificado para o estado s_0 do exemplo da figura 6

$T = 1$	P_{-G}^T	ξ
(s_0, s_1)	$0.5 * 1 = 0.5$	$\frac{0.5}{0.5} * \frac{1}{1} = 1$
(s_0, s_2)	$0.5 * 1 = 0.5$	$\frac{0.5}{0.5} * \frac{1}{1} = 1$
$T=2$	P_{-G}^T	ξ
(s_0, s_1)	$0.5 * \frac{1}{3} + 0.5 * \frac{1}{3} + 0.5 * \frac{1}{3} = 0.5$	$\frac{0.5}{0.5} * \frac{1}{0.5} = 2$
(s_0, s_2)	$0.5 * 0 = 0$	0
$T=3$	P_{-G}^T	ξ
(s_0, s_1)	$0.5 * 0 = 0$	0
(s_0, s_2)	$0 * 0 = 0$	0

processo de execução do *MPCVaR* e a simulação para k políticas diferentes pode ser visto de forma ilustrativa na figura 8.

Algoritmo 5 Simulação de uma política não-markoviana usando o resultado do algoritmo *MPCVaR*

- 1: **Para todo** $s \in S$ e $y \in Y$
 - 2: $i^* = \operatorname{argmin}_{i \in \{1..k\}} V^{\pi_i}(s, y)$
 - 3: $\pi(s, y) = \pi_{i^*}(s, y)$
 - 4: **Para todo** $s' \in S$
 - 5: $\xi(s, y, s') = \xi^{\pi_{i^*}}(s, y, s')$
-

Figura 8 – Algoritmo MPCVaR e simulação

7 Experimentos

Os experimentos realizados foram divididos em três partes. Na primeira parte (Seção 7.2) foi comparada a qualidade da aproximação do algoritmo *CVaRVILI* proposto em (CHOW et al., 2015), com o novo algoritmo proposto que faz a avaliação exata do valor de uma política, *PECVaR*. Na segunda parte (Seção 7.3) são comparados os tempos de execução do algoritmo *CVaRVILI* utilizando 4 abordagens diferentes de inicialização. Na terceira parte (Seção 7.4) são analisadas as simulações das políticas encontradas pelo algoritmo *CVarVILI*, pelo algoritmo proposto *MPCVaR* e o algoritmo *PECVaR-MIN* (que simplesmente calcula o valor mínimo de um conjunto de políticas escolhidas).

Os experimentos foram realizados no software Matlab R2019a, versão acadêmica de 64 bits para macOS, em um processador Intel Core i5 CPU @1.4 GHz, 4 GB de memória RAM @1600 MHz, 128 GB de armazenamento SSD. Além disso, foi desenvolvida uma versão do código¹ em Python.

Nos experimentos são utilizados os parâmetros $\gamma = 0.95$ (fator de desconto) e $error = 1e - 3$ (tolerância ao erro).

7.1 Domínios e Políticas Encontradas pelo algoritmo *CVaRVILI*

Os experimentos foram realizados em três domínios: *Fast Slow*, *Grid World* e *Travessia do Rio* que são descritos a seguir.

7.1.1 Domínio Fast Slow.

No domínio *Fast Slow* existe um caminho reto representado por um grid $N \times 1$. O agente parte do início do caminho e seu objetivo é chegar ao final do mesmo. Em cada estado, o agente pode escolher uma de duas ações possíveis: (i) a ação *fast* que faz com que o agente se movimente para o próximo estado com 75% de probabilidade e para o estado anterior com 25% de probabilidade (exceto no primeiro estado); e (ii) a ação *slow* que permite que o agente se movimente para o próximo estado com 50% de chance ou permaneça no mesmo estado com 50% de probabilidade. O custo de qualquer ação é 1. A Figura 9 mostra uma instância do domínio *Fast Slow* de tamanho 7.

¹ Disponível em: https://dbenevolo@bitbucket.org/dbenevolo/cvar_mdp.git

Figura 9 – Instância do *Fast Slow* com 7 estados. O estado meta é o cinza claro, o estado inicial é preto.

7.1.2 Domínio Grid World.

O domínio *Grid World* (CHOW et al., 2015) tem um mapa de terreno 2D que é representado por um grid $N \times M$. Um agente (por exemplo, um veículo robótico) começa em uma região segura e seu objetivo é viajar para um determinado destino. Em cada passo, o agente pode mover-se para o norte (N), sul (S), leste (L) ou oeste (W). A probabilidade da ação ter o efeito desejado é 95%. Além disso, existe uma probabilidade de $p = 5\%/3 = 1.67\%$ de movimentar-se para um dos outros 3 estados vizinhos.

O custo para mover-se de um estado para o outro, associado ao uso de combustível, é igual a 1. Entre o ponto de partida e o destino, há uma série de obstáculos que o agente deve evitar pois se o agente colidir em algum destes obstáculos, ele ficará danificado, sem a possibilidade de chegar no estado objetivo.

A Figura 10 mostra uma instância do domínio *Grid World* com uma grid de tamanho 14×16 .

Figura 10 – Instância do *Grid World* de tamanho 14×16 . O estado meta é o cinza claro, o estado inicial é preto e os obstáculos são cinza escuro.

7.1.3 Domínio Travessia do Rio

No problema do rio, um agente está em um lado do rio e quer ir para o outro lado do rio. Apenas 4 movimentos são possíveis, ir para o norte (N), sul (S), leste (E) e oeste (W). O agente pode chegar no objetivo de duas formas: (i) nadando a partir de qualquer ponto do rio, ou, (ii) subindo o grid até a ponte que está na posição extrema superior. Na figura 11 pode-se ver um exemplo com 25 estados. O agente partindo do estado com a cor preta tem como objetivo chegar ao canto inferior direito do grid, estado contornado com a linha pontilhada.

As ações executadas na margem do rio têm uma probabilidade de 0.99 de ter o efeito esperado e 0.01 de ficar no mesmo lugar. As ações executadas na ponte são determinísticas. As ações executadas no rio são probabilísticas, sendo que (i) o agente segue as direções cardeais escolhidas com probabilidade $(1 - p)^2$, no qual p é uma determinada probabilidade; (ii) é arrastado pela correnteza com probabilidade $(p)^2$; ou (iii) permanece no mesmo local com probabilidade $2p(1 - p)$. O valor utilizado nos experimentos é $p = 0.28$. A cachoeira é modelada como estados absorventes e o custo de qualquer ação é 1.

Figura 11 – Instância da *Travessia do Rio* com 25 estados. O estado meta é o cinza claro, o estado inicial é preto.

7.2 Avaliação da Qualidade da Aproximação do CVaRVILI

Nesta seção é avaliada a qualidade da aproximação do *CVaRVILI* considerando diferentes quantidades de pontos de interpolação (11, 21 e 31) comparando o resultado com o valor devolvido pelo algoritmo de avaliação de políticas exato *PECVaR*. Para tal comparação foi escolhida apenas uma política, a política neutra ao risco gerada pelo algoritmo de *Iteração de Valor* (Algoritmo 1), chamada de política π_{mean} . Além disso, é calculado o erro de aproximação utilizando a distância euclidiana entre os valores devolvidos por ambos os algoritmos para o estado s_0 .

Na figura 12 é mostrado o valor exato da política π_{mean} para o estado s_0 da instância do domínio *Grid World* com 224 estados calculados pelo algoritmo *PECVaR* e o resultado do cálculo do *CVaRVILI* para 11, 21 e 31 pontos de interpolação.

Na tabela 8, são mostrados o erro de aproximação e o tempo de execução do algoritmo *CVaRVILI* para os três conjuntos de pontos de interpolação 11, 21 e 31 para a mesma instância. Observa-se que enquanto o erro vai diminuindo mais lentamente, quando é aumentada a quantidade de pontos de interpolação, o tempo de execução gasto pelo *CVaRVILI* cresce mais rapidamente. Como esperado, o maior erro encontrado foi para 11 pontos de interpolação. A diferença entre o erro entre 21 e 31 pontos de interpolação foi a menor. Em contrapartida, o tempo para execução para 31 pontos interpolação foi aproximadamente o dobro do tempo do que para 21 pontos. Para essa instância, o tempo de execução do algoritmo *PECVaR* foi de apenas 17.89 segundos.

Figura 12 – Qualidade da aproximação do *CVaRVILI* para 11, 21 e 31 pontos de interpolação na instância do domínio Grid World com 224 estados

Tabela 8 – Erro e tempo do algoritmo *CVaRVILI* - Grid World 224

Pontos de interpolação	Erro	Tempo <i>CVaRVILI</i> (segundos)
11	0.5220	205.17
21	0.4861	473.67
31	0.4699	737.16

Na figura 13 e na tabela 9 são mostrados os mesmos dados, mas agora da política π_{mean} para o estado s_0 para uma instância do domínio *Fast Slow* com 7 estados. Nota-se uma diminuição brusca do erro de aproximação quando compara-se o erro obtido com 11 e o erro obtido com 21 pontos de interpolação, enquanto nos outros casos o erro vai diminuindo mais lentamente quando aumenta-se a quantidade de pontos de interpolação. Para essa instância, o tempo de execução do algoritmo *PECVaR* foi de apenas 0,099 segundos.

Tabela 9 – Erro e tempo do algoritmo *CVaRVILI* - Fast Slow 7

Pontos de interpolação	Erro	Tempo <i>CVaRVILI</i> (segundos)
11	1.2544	8.74
21	0.5420	17.59
31	0.4987	22.23

Figura 13 – Qualidade da aproximação do *CVaRVILI* para 11, 21 e 31 pontos de interpolação na instância do domínio Fast Slow com 7 estados

Na figura 14 e na tabela 10 são mostrados os mesmos dados, mas agora da política π_{mean} para o estado s_0 da instância do domínio *Travessia do Rio* com 35 estados. Nota-se que o erro de aproximação obtido com 11 pontos de interpolação é relativamente mais alto quando comparado com 21 e 31 pontos. Para essa instância, o tempo de execução do algoritmo *PECVaR* foi de apenas 0,41 segundos.

Nas três instâncias testadas, o tempo de execução gasto pelo algoritmo *CVaRVILI* cresce de acordo com o número de pontos de interpolação e o tempo de execução do algoritmo *PECVaR* é muito menor que o *CVaRVILI* mesmo considerando apenas 11 pontos de interpolação.

Tabela 10 – Erro e tempo do algoritmo *CVaRVILI* - Travessia do Rio 35

Pontos de interpolação	Erro	Tempo <i>CVaRVILI</i> (segundos)
11	0.6598	11.54
21	0.2175	24.31
31	0.1720	48.95

Figura 14 – Qualidade da aproximação do $CVaRVILI$ para 11, 21 e 31 pontos de interpolação na instância da Travessia do Rio com 35 estados

7.3 Diferentes Formas de Inicialização do $CVaRVILI$

O algoritmo $CVaRVILI$ pode demorar muito para encontrar a função valor ótima aproximada. Considerando isso, o objetivo dos experimentos nesta seção é investigar a possibilidade de melhoria do algoritmo $CVaRVILI$ utilizando como inicialização da função valor, o resultado da avaliação de uma política feita pelo algoritmo $PECVaR$.

Quatro abordagens diferentes foram utilizadas para inicialização do algoritmo $CVaRVILI$. A primeira utiliza a função valor $V_0 = 0$ (chamada V_0), a segunda utiliza a função valor obtida pelo algoritmo de *Iteração de Valor* para MDPs neutros a risco (chamada V_{mean}), a terceira utiliza a política obtida pelo algoritmo de *Iteração de Valor* para MDPs neutros a risco e o algoritmo $PECVaR$ para calcular a função valor dessa política (chamada V_{pecvar}) e a quarta utiliza 4 políticas diferentes (por exemplo, obtidas pelo algoritmo $CVaRVILI$ ou algum algoritmo de MDPs sensíveis ao risco para diferentes valores de risco), essas 4 políticas são avaliadas pelo algoritmo $PECVaR$ e finalmente o valor mínimo dessas 4 políticas é usado como função valor (chamada V_{pecvar_n}).

Foi calculado o tempo total de execução do *CVaRVILI* considerando as diferentes abordagens de inicialização. No caso da abordagem V_0 , o tempo de execução inclui apenas o custo computacional do algoritmo *CVaRVILI*. No caso da abordagem V_{mean} soma-se (i) o tempo gasto pelo algoritmo de *Iteração de Valor*; e (ii) o tempo de execução até convergência do algoritmo *CVaRVILI*. No caso da abordagem do V_{pecvar} soma-se: (i) o tempo gasto do algoritmo de *Iteração de Valor* para encontrar a política ótima do MDP neutro a risco; (ii) o tempo de execução do algoritmo *PECVaR* para calcular o valor da política encontrada pelo algoritmo de *Iteração de Valor*; e (iii) o tempo de execução até convergência do algoritmo *CVaRVILI*. Por último, no caso da abordagem do V_{pecvar_n} , soma-se: (i) o tempo gasto do algoritmo de *CVaRVILI* para encontrar 4 políticas ótimas; (ii) o tempo de execução do algoritmo *PECVaR* para calcular o valor das políticas encontradas pelo algoritmo de *CVaRVILI*; e (iii) o tempo de execução até convergência do algoritmo *CVaRVILI*.

Para os experimentos foram utilizadas três quantidades diferentes de pontos de interpolação: 11, 21 e 31.

Na figura 15 são apresentados os resultados para duas instâncias do domínio *Fast Slow*, uma de 7 estados e outra de 70 estados. Para a instância *Fast Slow* de 7 estados, o tempo gasto pelo algoritmo de *Iteração de Valor* foi de 0.015 segundos e o tempo de execução do algoritmo *PECVaR* para as diferentes quantidades de pontos de interpolação foi menor ou igual a 0.066 segundos. Para a instância *Fast Slow* de 70 estados, o tempo gasto pelo algoritmo de *Iteração de Valor* foi de 0.21 segundos e o tempo de execução do algoritmo *PECVaR* para as diferentes quantidades de pontos de interpolação foi menor ou igual a 1 segundo. Neste domínio, a abordagem de inicialização com V_{pecvar} teve desempenho melhor em todos os casos do que a inicialização com V_0 , sendo até 5.3 vezes mais rápida que a inicialização com V_0 (no experimento com 7 estados e com 31 pontos de interpolação). Enquanto que a abordagem de inicialização V_{mean} foi pior em 4 dos 6 experimentos do que a abordagem de inicialização V_0 e a abordagem V_{pecvar_n} foi melhor que V_{mean} e V_0 em 4 experimentos dos 6, tendo destaque no experimento com 21 e 31 pontos de interpolação na instância de 70 estados na qual conseguiu ser melhor que todas as abordagens.

Na figura 16 são apresentados os resultados para duas instâncias do domínio *Grid World*, uma de 224 estados e outra de 3392 estados. Para a instância com 224 estados, o tempo gasto do algoritmo de *Iteração de Valor* foi 1.72 segundos e o tempo de execução do

algoritmo *PECVaR* para as diferentes quantidades de pontos de interpolação foi menor ou igual que 11.32 segundos. Para a instância com 3392 estados, o tempo gasto do algoritmo de *Iteração de Valor* foi 165.63 segundos e o tempo de execução do algoritmo *PECVaR* para as diferentes quantidades de pontos de interpolação foi menor ou igual que 262.16 segundos. Neste domínio, a abordagem de inicialização com V_{pecvar} foi a melhor em todos os casos, enquanto que a abordagem de inicialização V_{mean} foi melhor que as outras duas abordagens.

Na figura 17 são apresentados os resultados para duas instâncias do domínio de Travessia do Rio, uma de 35 estados e outra de 364 estados. Para a instância com 35 estados, o tempo gasto do algoritmo de *Iteração de Valor* foi 0.0668 segundos e o tempo de execução do algoritmo *PECVaR* para as diferentes quantidades de pontos de interpolação foi menor ou igual que 0.070 segundos. Para a instância com 364 estados, o tempo gasto do algoritmo de *Iteração de Valor* foi 136 segundos e o tempo de execução do algoritmo *PECVaR* para as diferentes quantidades de pontos de interpolação foi menor ou igual que 271.73 segundos. Neste domínio, a abordagem de inicialização com V_{pecvar} foi a melhor em todos os casos, enquanto que a abordagem de inicialização V_{pecvar_n} foi melhor que as outras duas abordagens.

Note que, o tempo gasto na execução dos algoritmos *Iteração de Valor* e *PECVaR* é muito menor do que o tempo de execução do algoritmo *CVaRVILI*. Nos domínios testados o tempo gasto de execução do algoritmo *PECVaR* foi na maioria dos casos apenas 2 vezes maior do que o algoritmo de *Iteração de Valor* para MDP neutros ao risco.

7.4 Comparando a Qualidade da Política CVaRVILI, MPCVaR e PECVaR-MIN

Nesta seção são feitas simulações usando as políticas encontradas pelo algoritmo *CVaRVILI*, pelo algoritmo *MPCVaR* e *PECVaR-MIN*. Os três algoritmos foram executados para 31 pontos de interpolação. Para fazer a simulação dos algoritmos também foram utilizados 31 valores de $\alpha \in [0.01, 1]$ e realizadas 5000 simulações a partir do estado inicial para cada valor de α . Para cada um dos 31 pontos de α mostrados nas figuras de simulação foi calculado o valor *CVaR*. Por exemplo, para $\alpha = 0.01$ calcula-se a média dos 1% dos piores resultados obtidos nas simulações.

Figura 15 – Tempo total de execução do algoritmo *CVaRVILI* com diferentes abordagens de inicialização para duas instâncias do domínio *Fast Slow* com 11, 21 e 31 pontos de interpolação.

Figura 16 – Tempo total de execução do algoritmo *CVaRVILI* com diferentes abordagens de inicialização para a instância com 224 estados do domínio *Grid World* com 11, 21 e 31 pontos de interpolação e para a instância com 3392 estados com 11 e 21 pontos de interpolação.

Figura 17 – Tempo total de execução do algoritmo *CVaRVILI* com diferentes abordagens de inicialização para a instância com 35 estados do domínio *Rio* com 11, 21 e 31 pontos de interpolação e para a instância com 364 estados com 11, 21 e 31 pontos de interpolação.

Uma instância de 7 estados do *Fast Slow*, uma instância do *Grid World* com 224 estados e uma instância da *Travessia do Rio* com 35 estados são utilizadas. Em todos os casos, foram utilizadas quatro políticas para executar os algoritmos *MPCVaR* e *PECVaR-MIN*.

Na figura 18 é mostrada a avaliação feita pelo algoritmo *PECVaR* para cada uma das quatro políticas utilizadas pelo algoritmo *MPCVaR* para a instância do *Fast Slow* com 7 estados. Note que as políticas π_1 , π_2 e π_3 estão sobrepostas.

Figura 18 – Políticas utilizadas na simulação do *MPCVaR* no Fast-Slow com 7 estados.

Na figura 19 é mostrada a simulação do algoritmo *CVaRVILI*, *MPCVaR* e *PECVaR-MIN* para 31 valores de $\alpha \in [0.01, 1]$.

A tabela 11 mostra o tempo de execução dos algoritmos *CVaRVILI* e *MPCVaR* e a distância euclidiana entre eles em relação a função valor. Observa-se que o algoritmo *MPCVaR* foi 3.1 vezes mais rápido que o algoritmo *CVaRVILI* com uma diferença entre o a função valor do *CVaRVILI* e *MPCVaR* de apenas 0.5604.

Tabela 11 – Fast Slow 7 - *CVaRVILI* e *MPCVaR*

Pontos de interpolação	Distância	<i>CVaRVILI</i> (segundos)	<i>MPCVaR</i> (segundos)
31	0.5604	21.34	6.79

De modo equivalente, foram feitas as simulações usando as políticas encontradas pelos algoritmos *CVaRVILI*, *MPCVaR* e *PECVaR-MIN* para o domínio *Grid World*. A

Figura 19 – Simulação *CVaRVILI*, *MPCVaR* e *PECVaR-MIN* no Fast-Slow com 7 estados.

figura 20 representa o custo exato das quatro políticas escolhidas para a simulação do domínio *Grid World* de 224 estados. Note que há sobreposição das políticas π_2 e π_3 . A figura 21 mostra a simulação dos 3 algoritmos para diferentes valores de α .

Figura 20 – Políticas utilizadas na simulação do *MPCVaR* no *Grid World* com 224 estados

Fonte: Própria

Figura 21 – Simulação *CVaRVILI*, *MPCVaR* e *PECVaR-MIN* no Grid World com 224 estados

Fonte: Própria

A tabela 12 mostra a distância entre os valores calculados pelo algoritmo *CVaRVILI* e o algoritmo *MPCVaR*, bem como o tempo de execução. Observa-se neste domínio um expressivo ganho do algoritmo *MPCVaR* em relação ao *CVaRVILI*, o primeiro foi 20.1 vezes mais rápido que o segundo, sendo que a diferença foi de 2.42.

Tabela 12 – Simulação Grid World 224 - *CVaRVILI* e *MPCVaR*

Pontos de interpolação	Distância	<i>CVaRVILI</i> (segundos)	<i>MPCVaR</i> (segundos)
31	2.42	3137.92	156.94

Também foram feitas as simulações usando as políticas encontradas pelos algoritmos *CVaRVILI*, *MPCVaR* e *PECVaR-MIN* para o domínio *Travessia do Rio*. A figura 22 representa o custo exato das quatro políticas escolhidas para a simulação do domínio *Travessia do Rio* de 35 estados. A figura 23 mostra a simulação dos 3 algoritmos para diferentes valores de α .

A tabela 13 mostra a distância entre os valores calculados pelo algoritmo *CVaRVILI* e o algoritmo *MPCVaR*, bem como o tempo de execução. Observa-se um expressivo ganho do algoritmo *MPCVaR* em relação ao *CVaRVILI*, o primeiro foi 14.6 vezes mais rápido que o segundo, sendo que a diferença foi de 0.9709.

Figura 22 – Políticas utilizadas na simulação do *MPCVaR* no Rio com 35 estados

Figura 23 – Simulação *CVaRVILI*, *MPCVaR* e *PECVaR-MIN* no Rio com 35 estados

Tabela 13 – Simulação Travessia Rio 35 - *CVaRVILI* e *MPCVaR*

Pontos de interpolação	Distância	<i>CVaRVILI</i> (segundos)	<i>MPCVaR</i> (segundos)
31	0.9709	483.52	33.02

8 Conclusão

Existe um significativo aumento do uso da métrica $CVaR$ no contexto de tomada de decisão sequencial como uma forma de garantir robustez em relação à estocasticidade e aos erros de modelagem. Porém, apesar dos avanços e dos resultados obtidos por alguns trabalhos que utilizam a função $CVaR$ em MDPs, existe uma carência de trabalhos que mostrem resultados experimentais abrangentes avaliando esse novo critério.

Neste trabalho, propõe-se o uso do Algoritmo $PECVaR$, um método eficiente para calcular o valor de uma política estacionária de custo constante de MDPs que utilizam a métrica $CVaR$. Diferente do algoritmo $CVaRVILI$, o algoritmo $PECVaR$ não precisa resolver nenhum problema linear e, portanto, não é tão custoso computacionalmente como o algoritmo $CVaRVILI$. Além disso, foram realizados experimentos comparando abordagens diferentes para inicialização do Algoritmo $CVaRVILI$. Os experimentos demonstraram que a abordagem com inicialização com V_{pecvar} foi até 5.3 vezes melhor no tempo total de execução comparado com a abordagem que utiliza a inicialização V_0 .

Problemas com um número significativamente grande de estados ainda podem ser significativamente desafiadores para serem resolvidos em tempo factível usando a abordagem $CVaRVILI$. Acredita-se que com os resultados aqui apresentados, pode-se possibilitar o uso de instâncias com número de estados cada vez maiores. Como visto na Seção 7.4 o novo algoritmo $MPCVaR$, teve um expressivo ganho em termos de tempo de execução e conseguiu ser até 20.1 vezes mais rápido que o algoritmo $CVaRVILI$.

Com o desenvolvimento dessa nova abordagem para avaliação de políticas estacionárias, pretende-se contribuir com a área de planejamento probabilístico em inteligência artificial.

8.1 Resumo das contribuições

Entre as principais contribuições do trabalho estão:

- Formulação do algoritmo $PECVaR$ que realiza a avaliação exata de uma política estacionária e do algoritmo $MPCVaR$, um algoritmo heurístico para encontrar uma política aproximada para $CVaR$ MDPs.

- Análise de abordagens eficientes para inicialização do algoritmo *CVaRVILI*. Foram feitos experimentos comparando o *CVaRVILI* com diferentes inicializações.
- Análise da aproximação feita pelo algoritmo *CVaRVILI*, foi analisado como o número de pontos de interpolação influencia na diminuição do erro de aproximação e o quanto paga-se a mais em termos de tempo de execução para aumentar esses pontos.
- Análise da qualidade das políticas encontradas pelo algoritmo *CVaRVILI* e *MPCVaR*. Foi realizado também uma análise do tempo de execução destes algoritmos para encontrar essas políticas.

8.2 Publicações geradas

1. Em (PAIS; DELGADO, 2018) foi definida uma proposta de pesquisa que tinha como objetivo criar um novo algoritmo aproximado de iteração de valor para resolver um *mean-CVaR* MDP, um MDP sensível a risco que utiliza a média do custo total em conjunto com o critério *CVaR*. Esta proposta de pesquisa foi aceita para publicação no Workshop de Teses e Dissertações em Sistemas de Informação do XI Simpósio Brasileiro de Sistemas de Informação.
2. Em (PAIS; DELGADO; FREIRE, 2019) foi proposto o algoritmo *PECVaR* que avalia uma política estacionária para *CVaR* MDPs de custo constante e não precisa resolver problemas de programação linear. Embora o algoritmo *PECVaR* não possa ser utilizado para encontrar a política ótima, considera-se a hipótese de que o valor *CVaR* de uma política arbitrária é melhor como inicialização do algoritmo *CVaRVILI* do que o valor esperado da mesma política. Esse trabalho foi publicado no XVI Encontro Nacional de Inteligência Artificial (ENIAC), SBC.

8.3 Trabalhos futuros

Para os trabalhos futuros almeja-se generalizar o *PECVaR* para computar políticas estacionárias de custo não constante e testar o algoritmo em outros domínios. Também almeja-se usar outras formas de interpolação no algoritmo *CVaRVILI*, entre elas interpolação polinomial de grau 2 e grau 3. Por outro lado, os resultados com as diferentes abordagens de inicialização corroboram a ideia de que o algoritmo *CVaRVILI* tem como

ser melhorado, podendo utilizar uma boa abordagem de inicialização ou outras técnicas computacionais como, por exemplo, paralelização, que pode trazer significativas melhorias no desempenho do *CVaRVILI*, abrindo caminhos para o processamento de problemas ainda maiores em tempos factíveis. Assim, um outro trabalho futuro possível diz a respeito do uso de paralelização no *CVaRVILI*.

Referências¹

- ARTZNER P., D. F. E. J.; HEATH. Coherent measures of risk. *Mathematical Finance*, 1999. Citado 2 vezes nas páginas 28 e 31.
- BELLMAN, R. E. Dynamic programming. *Princeton University Press*, 1957. Citado na página 22.
- CARPIN., Y.-L. C. S.; PAVONE, M. Risk aversion in finite Markov decision processes using total cost criteria and average value at risk. *Advances in Neural Information Systems*, 2016. Citado na página 17.
- CHOW, Y. *Risk-sensitive and data-driven sequential decision making*. Tese (Doutorado) — Stanford University, 2017. Citado na página 44.
- CHOW, Y. et al. Risk-sensitive and robust decision-making: a CVaR optimization approach. *Advances in Neural Information Systems*, 2015. Citado 12 vezes nas páginas 16, 17, 27, 33, 34, 35, 38, 39, 41, 47, 54 e 55.
- FILAR, J. A.; KALLENBERG, L. C. M.; LEE, H.-M. Variance-penalized Markov decision processes. *Mathematics of Operations Research*, v. 14, n. 1, p. 147–161, 1989. Citado na página 27.
- GARCIA, J.; FERNÁNDEZ., F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 2015. Citado na página 27.
- IYENGAR., G.; MA, A. K. C. Fast gradient descent method for mean-CVaR optimization. *Annals of Operations Research*, 2009. Citado na página 17.
- KNIGHT, F. H. *Risk, uncertainty and profit*. [S.l.]: Courier Corporation, 2012. Citado na página 26.
- LITTMAN M. L., D. T. L. e. K. L. P. On the complexity of solving Markov decision problems. *Eleventh International Conference on Uncertainty in Artificial Intelligence*, p. 394–402, 1995. Citado na página 24.
- MIHATSCH, O.; NEUNEIER, R. Risk-sensitive reinforcement learning. *Machine Learning*, Kluwer Academic Publishers, v. 49, n. 2, p. 267–290, 2002. Citado na página 27.
- PAIS, D. B.; DELGADO, K. V. Processos de decisão markovianos sensíveis a risco: Uma abordagem de otimização mean-cvar. In: *Anais do XI Workshop de Teses e Dissertações em Sistemas de Informação (WTDSI) do Simpósio Brasileiro de Sistemas de Informação*. Caxias do Sul, Rio Grande do Sul, Brasil: Sociedade Brasileira de Computação, 2018. p. 22–25. Citado na página 71.
- PAIS, D. B.; DELGADO, K. V.; FREIRE, V. Exact algorithm to evaluate stationary policies for cvar mdp. In: *Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*. Uberlândia, Minas Gerais, Brasil: Sociedade Brasileira de Computação, 2019. p. 0–0. Citado na página 71.
- PFLUG, G.; PICHLER, A. Time consistent decisions and temporal decomposition of coherent risk functionals. *Optimization online*, 2015. Citado 2 vezes nas páginas 33 e 34.

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

PUTERMAN., M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: Wiley-Interscience, 1994. Citado 5 vezes nas páginas 16, 20, 22, 23 e 24.

ROCKAFELLER, R.; URYASEV, S. Conditional value-at-risk: Optimization approach. *Journal of risk*, 2000. Citado 2 vezes nas páginas 17 e 30.

ROCKAFELLER, R.; URYASEV, S. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, 2002. Citado na página 17.

SOBEL, M. J. The variance of discounted Markov decision processes. *Journal of Applied Probability*, Applied Probability Trust, v. 19, n. 4, p. 794–802, 1982. Citado na página 27.

Anexo A – Interpolação

A.1 Interpolação

Interpolação é um método que permite fazer a reconstituição ou aproximação de uma função ou um conjunto de funções analíticas que possam servir para a determinação de qualquer valor no domínio de definição utilizando um conjunto discreto de pontos já conhecidos.

De forma podemos escrever, dada uma sequência de n reais $x_1 < x_2 < \dots < x_n$ $\{(x_i, y_i) \in I \times \mathbb{R}\}_{i=1}^n$, em que $I = \{x_1, \dots, x_n\}$ e uma família de funções $F = \{\Phi : I \rightarrow \mathbb{R}\}$. O problema de interpolação consiste em encontrar alguma função $f \in F$ tal que:

$$f(x_i) = y_i, i = 1, 2, \dots, n \quad (43)$$

em que f é chamada de função interpoladora dos pontos dados ou simplesmente função que interpola os pontos dados. A interpolação pode ser utilizada em casos quando temos uma função complexa e não conseguimos trabalhar com ela de forma eficiente ou avaliá-la, neste caso utiliza-se interpolação para fazer aproximações desta função através de funções mais simples. Primeiro é necessário escolher um conjunto de pontos da função que queremos aproximar e em seguida utiliza-se esses dados para montar a função mais simples. É preciso lembrar que quando é aplicado esse processo é possível adicionar um erro de truncamento que é o erro cometido em decorrência da representação de uma suposta função por outra. A seguir é descrita a interpolação linear.

A.2 Interpolação Linear

A interpolação linear ou simplesmente interpolação, deduz um valor entre dois valores explicitamente dados encontrando a equação da reta que passa por esses dois pontos. Seja $P = (x, y)$ um ponto, dados x e dois pontos $P_1 = (x_1, y_1)$ e $P_2 = (x_2, y_2)$, se a função for uma reta, o valor de y pode ser calculado da seguinte forma:

$$y = y_1 + (x - x_1) \frac{y_2 - y_1}{x_2 - x_1} \quad (44)$$

Quando a função a estimar não é uma reta, considera-se um trecho de reta entre dois pontos consecutivos. Para se ter uma melhor estimativa, os pontos conhecidos devem

Figura 24 – Interpolação Linear

Fonte: <http://www.hidromundo.com.br/interpolacao-linear/>

estar suficientemente próximos para que o erro seja pequeno. Por exemplo, na Figura 24 o erro é menor ao usar os pontos (x_1, y_1) e (x_2, y_2) , do que usar os pontos (x_3, y_3) e (x_4, y_4) .