Risk Sensitive Probabilistic Planning with ILAO* and Exponential Utility Function

Elthon Manhas de Freitas[1],**Karina Valdivia Delgado**[1], **Valdinei Freire**[1]

[1]Escola de Artes, Ciências e Humanidades – Universidade de São Paulo (USP)
Av. Arlindo Bettio, 1000 – 03828-000 – São Paulo – SP – Brazil

{elthon,kvd,valdinei.freire}@usp.br

Abstract. *Markov Decision Process (MDP) has been used very efficiently to solve sequential decision-making problems. However, there are problems in which dealing with the risks of the environment to obtain a reliable result is more important than minimizing the total expected cost. MDPs that deal with this type of problem are called risk-sensitive Markov decision processes (RSMDP). In this paper we propose an efficient heuristic search algorithm that allows to obtain a solution by evaluating only the relevant states to reach the goal states starting from an initial state.*

Resumo. *Os processos de decisão de Markov (Markov Decision Process – MDP) têm sido usados com muita eficiência para resolução de problemas de tomada de decisão sequencial. Porém, existem problemas em que lidar com os riscos do ambiente para obter um resultado confiável é mais importante do que minimizar o custo total esperado, que é o critério utilizado em MDPs clássicos. MDPs que lidam com esse tipo de problemas são chamados de processos de decisão de Markov sensíveis a risco (Risk-Sensitive Markov Decision Process - RSMDP). Neste artigo é proposto um algoritmo eficiente de busca heurística que permite obter uma solução avaliando apenas os estados relevantes para atingir um conjunto de estados meta partindo de um estado inicial.*

Keywords: *Markov Decision Process, Expected Utility Theory, Risk Sensitive*

1. Introdução

Processo de decisão de Markov (Markov Decision Process – MDP) é um modelo matemático utilizado para tomada de decisão sequencial baseada apenas nas informações do estado atual do ambiente [Puterman 2014]. Este modelo é classificado como estocástico pois não há controle de todas as variáveis presentes no ambiente em que o agente tomador de decisão está inserido. A incerteza nesses processos pode ocorrer por diversos fatores como imprecisão durante a execução de uma ação ou ainda pela existência de outros agentes que podem estar constantemente interferindo no sistema.

A maioria dos trabalhos em MDPs avalia o efeito probabilístico na execução de ações no sistema e têm como objetivo gerar uma política, a ser seguida por um agente executor, de modo a maximizar a recompensa esperada do sistema ou minimizar o custo esperado pois ao longo de sucessivas execuções, o sistema tende gerar um valor médio muito próximo ao valor esperado.

Porém, há problemas da vida real que só podem ser executados apenas uma vez. Por exemplo, um veículo com navegação autônoma tem que considerar que cada trajeto é único e não irá se repetir, logo o processo não poderá simplesmente reiniciar em caso de falha. Realizar um transplante de coração é outro exemplo em que aumentar as chances de sucesso se faz tão importante que aumentos de custo são aceitos quase sem questionamento. Outros problemas tem uma duração tão longa que não podem ser executados várias vezes, como realizar uma viagem a Marte, investir para a aposentadoria de uma vida [Moldovan and Abbeel 2012] ou implantar um grande projeto empresarial. Estes são alguns exemplos em que mitigar, evitar e até eliminar os riscos do ambiente é muito mais importante do que maximizar o retorno esperado. Estes são problemas de máxima aversão ao risco e para este extremo existe uma otimização denominada *minmax*, que minimiza o custo considerando o pior caso.

Exceto pelos casos extremos, o cotidiano é pautado por aversão e propensão ao risco. Um piloto de corrida está disposto a forçar um pouco mais o carro na última volta para conseguir melhorar sua posição, assim como as pessoas estão dispostas a conhecer um lugar novo em busca de novas experiências. Devemos considerar também que pessoas diferentes têm níveis de aceitação diferentes de risco, até mesmo momentos distintos podem afetar e tornar a pessoa mais propensa ou mais aversa ao risco.

Para lidar com esse tipo de problemas, há uma pequena parcela de trabalhos que avaliam a sensibilidade e a tolerância ao risco e de alguma forma consideram estes parâmetros em seus modelos, os chamados Processos de Decisão de Markov Sensíveis ao Risco (Risk Sensitive Markov Decision Process – RSMDP). Existem vários critérios que podem ser usados para lidar com risco, entre eles, o critério que usa a utilidade exponencial esperada [Howard and Matheson 1972, Jaquette 1976, Denardo and Rothblum 1979, Rothblum 1984, Patek 2001, Freire and Delgado 2016], ponderação entre esperança e variância [Sobel 1982, Filar et al. 1989] e estimação de desempenho em um intervalo de confiança [Filar et al. 1995].

Nos trabalhos baseados em utilidade exponencial esperada é necessário especificar um fator de risco, sendo que os valores factíveis para esse fator de risco dependem do problema de decisão em questão [Patek 2001]. Em [Patek 2001] são provadas as condições para existência de uma política válida e um algoritmo de Iteração de Política para MDPs dirigidos à meta que usam o critério de utilidade exponencial esperada, chamados de RSMDPs dirigidos à meta (Goal Directed Risk Sensitive Markov Decision Process – GDRSMDP). Porém, uma vez que o algoritmo de Iteração de Política Sensível a Risco avalia o conjunto completo de estados em cada iteração, ele pode ser ineficiente para resolver problemas com um número grande de estados.

Este trabalho tem por objetivo propor um algoritmo eficiente que permite avaliar apenas o subconjunto de estados que são relevantes para resolver GDRSMDPs. O algoritmo proposto é uma adaptação do algoritmo ILAO* [Hansen and Zilberstein 2001], um algoritmo de busca heurística para MDPs que foca apenas nos estados que são alcançáveis a partir do estado inicial.

2. Conceitos Fundamentais

Nesta seção são definidos os MDPs dirigidos a meta e os MDPs dirigidos a meta e sensíveis a risco.

2.1. Processo de Decisão de Markov Dirigido à Meta

Um MDP dirigido a meta [Bertsekas and Tsitsiklis 1991, Geffner and Bonet 2013] (Goal-Directed MDP – GDMDP) é uma tupla $GDMDP = (S, A, T, c, S_g)$, em que:

- S é um conjunto finito de estados observáveis;
- A é um conjunto finito de ações;
- $T(s'|s, a)$ 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
- $S_g \subset S$ é um conjunto finito de estados meta. Todo estado meta é absorvente, isto é, $T(s' \in S_g|s \in S_g, a) = 1$ e $c(s \in S_g, a) = 0$ para qualquer $a \in A$.

O problema de um MDP dirigido à meta define um processo dinâmico discreto em que, em qualquer momento t, o agente observa um estado s_t, executa uma ação a_t, transita para um estado s_{t+1} após pagar um custo c_t. Este processo define um horizonte indeterminado, dado que o processo termina após atingir qualquer estado meta em S_g, e o número de passos que o agente tem para agir não é conhecido a priori. O objetivo de um problema de MDP dirigido à meta é alcançar o estado meta com o mínimo custo acumulado esperado, o qual é considerado um critério neutro a risco.

Uma política estacionária π é um mapeamento de estados em ações ($\pi : S \to A$) que representa quais ações devem ser executadas em cada estado. A solução de um MDP dirigido a meta é uma política estacionária π e o valor $V^\pi(s)$ de uma política em um estado $s \in S$ é determinado por:

$$V^\pi(s) = \lim_{M \to \infty} \mathrm{E}\left[\sum_{t=0}^{M} c\left(s_t, \pi(s_t)\right) \middle| \pi, s_0 = s \right].$$

O valor da política π para o estado s pode ser obtido resolvendo o seguinte sistema de equações:

$$V^\pi(s) = \begin{cases} 0 & \text{, se } s \in S_g \\ c(s, \pi(s)) + \sum_{s' \in \mathcal{S}} T(s'|s, \pi(s)) V^\pi(s') & \text{, nos demais casos.} \end{cases}$$

Seja Π o conjunto de políticas estacionárias, a função valor ótima $V^*(s) = \min_{\pi \in \Pi} V^\pi(s)$ é a solução da equação de Bellman:

$$V^*(s) = \begin{cases} 0 & \text{, se } s \in S_g \\ \min_{a \in A} \left[c(s, a) + \sum_{s' \in \mathcal{S}} T(s'|s, a) V^*(s') \right] & \text{, nos demais casos.} \end{cases}$$

A política ótima pode ser obtida com base na função valor ótima por:

$$\pi^*(s) = \arg\min_{a \in A} \left[c(s, a) + \sum_{s' \in S} T(s'|s, a) V^*(s') \right]. \quad (1)$$

2.2. MDP Sensível a Risco e Dirigido a Meta

Com base na Teoria da Utilidade Esperada, em [Howard and Matheson 1972] é proposta a função utilidade exponencial para modelar atitude ao risco do agente. A função utilidade apresenta algumas propriedades interessantes: (*i*) considera-se um parâmetro arbitrário β que modela a atitude ao risco do agente; e (*ii*) é possível construir uma equação de otimalidade, que é o caminho para definição dos algoritmos de Iteração de Valor e Iteração de Política.

Formalmente um MDP sensível a risco e dirigido a meta (Goal-Directed Risk Sensitive Markov Decision Process – GDRSMDP) é definido por uma tupla $GDRSMDP = (GDMDP, \beta)$, em que:

- $GDMDP$ é um MDP dirigido à meta.
- β é o fator de sensibilidade a risco.

Em GDRSMDPs, a função valor da política π é definida por:

$$V^{\pi}(s) = \lim_{M \to \infty} \mathrm{E}\left[-\mathrm{sgn}(\beta) \exp\left(\beta \sum_{t=0}^{M} c\left(s_t, \pi(s_t)\right) \right) \middle| \pi, s_0 = s \right]. \tag{2}$$

em que E representa a esperança quando a política π é executada, $\exp()$ é a função exponencial e o agente é averso ao risco se $\beta > 0$, propenso ao risco se $\beta < 0$ e neutro ao risco se $\beta \to 0$. O valor da política π para o estado s pode ser obtido resolvendo o seguinte sistema de equações:

$$V^{\pi}(s) = \begin{cases} -\mathrm{sgn}(\beta) & \text{, se } s \in S_g \\ \exp(\beta c(s, \pi(s))) \sum_{s' \in \mathcal{S}} T(s'|s, \pi(s)) V^{\pi}(s') & \text{, nos demais casos,} \end{cases} \tag{3}$$

ou em sua forma matricial:

$$\begin{aligned} \mathbf{V}^{\pi} &= (\mathbf{D}^{\pi})^{\beta} (\mathbf{T}^{\pi}_{G^c} \mathbf{V}^{\pi} - \mathrm{sgn}(\beta)(\mathbf{1} - \mathbf{T}^{\pi}_{G^c} \mathbf{1})) \\ &= ((\mathbf{D}^{\pi})^{\beta} \mathbf{T}^{\pi}_{G^c} - \mathbf{I})^{-1} \cdot (\mathbf{D}^{\pi})^{\beta} \cdot \mathrm{sgn}(\beta)(\mathbf{1} - \mathbf{T}^{\pi}_{G^c} \mathbf{1}), \end{aligned} \tag{4}$$

em que $\mathbf{V}^{\pi}$ é um vetor de tamanho $|S| \times 1$, $\mathbf{I}$ é uma matriz identidade de tamanho $|S| \times |S|$, $\mathbf{1}$ é um vetor-coluna apenas com o valor 1 e $\mathbf{D}^{\pi}$ é uma matriz diagonal $|S| \times |S|$ com os elementos de $\exp(\mathbf{c}^{\pi})$. A definição das matrizes $\mathbf{c}^{\pi}$ e $\mathbf{T}^{\pi}_{G^c}$ estão na Definição 1.

Definição 1 ***(Matrizes de Transição de Política e Vetor de Custo de Política)*** *Seja π uma política estacionária e cada estado em S seja enumerado como $1, 2, 3, \ldots, |S|$, $\mathbf{T}^{\pi}$ é uma matriz $|S| \times |S|$, em que cada célula $(\mathbf{T}^{\pi})_{ij}$ representa a probabilidade de transição do estado i para o estado j seguindo a política π, i.e., $(\mathbf{T}^{\pi})_{ij} = T(j|i, \pi(i))$. A matriz $\mathbf{T}^{\pi}_{G^c}$ também é uma matriz $|S| \times |S|$ similar à matriz $\mathbf{T}^{\pi}$, com a diferença que as colunas que representam os estados meta, $s \in S_g$, são alterados para 0, i.e.,*

$$(\mathbf{T}^{\pi}_{G^c})_{ij} = \begin{cases} 0 & \text{, se } j \in S_g \\ T(j|i, \pi(i)) & \text{, nos demais casos} \end{cases}. \tag{5}$$

$\mathbf{c}^{\pi}$ é o vetor $|S| \times 1$ em que cada célula $(\mathbf{c}^{\pi})_i$ representa o custo ao seguir a política π no estado i, i.e., $(\mathbf{c}^{\pi})_i = c(i, \pi(i))$.

Definição 2 ***(política β-factível) [Patek 2001]*** *Uma política π é β-factível se a probabilidade de não estar em um estado absorvente desaparece mais rápido do que o custo acumulado exponencial, i.e.,* $\lim_{t\to\infty}((\mathbf{D}^{\pi})^{\beta}\mathbf{T}^{\pi}_{G^c})^t = \mathbf{0}$*. De forma equivalente, uma política π é β-factível se o raio espectral de $(\mathbf{D}^{\pi})^{\beta}\mathbf{T}^{\pi}_{G^c}$ for menor que 1, i.e., $\rho((\mathbf{D}^{\pi})^{\beta}\mathbf{T}^{\pi}_{G^c}) < 1$.*

Seja Π o conjunto de políticas estacionárias, a função valor ótima $V^*(s) = \min_{\pi\in\Pi} V^{\pi}(s)$ é a solução da seguinte equação:

$$V^*(s) = \begin{cases} -\text{sgn}(\beta) & \text{, se } s \in S_g \\ \min_{a\in A}\left[\exp(\beta c(s,a))\sum_{s'\in S} T(s'|s,a)V^*(s')\right] & \text{, nos demais casos,} \end{cases} \tag{6}$$

e a política ótima pode ser obtida com base na função valor ótima por:

$$\pi^*(s) = \arg\min_{a\in A}\left[\exp(\beta c(s,a))\sum_{s'\in S} T(s'|s,a)V^*(s')\right]. \tag{7}$$

2.2.1. Algoritmo de Iteração de Política Sensível a Risco

Supondo que exista uma política inicial π_0 β-factível, o algoritmo de Iteração de Política Sensível a Risco (Algoritmo 1) permite encontrar uma política ótima π^* [Patek 2001]. Em que a cada iteração i dois passos são executados: *avaliação de política* e *melhora de política*. O passo avaliação de política utiliza a Equação 3 para calcular o valor de $V^{\pi_i}(\cdot)$ e o passo de melhoria de política, melhora π_i obtendo π_{i+1}.

Algorithm 1 Algoritmo de Iteração de Política Sensível a Risco

Require: ($\mathcal{GDRSMDP}$, β)

Ensure: Uma política ótima π

1: Escolhe uma política inicial β-factível π_0 arbitrariamente
2: $i \leftarrow 0$
3: **while** $\pi_i \neq \pi_{i-1}$ **do**
4: Avaliação de política: obtém o valor da política atual π_i para todo $s \in S$ através da resolução do sistema de equações descrito na Equação 3.
5: **for all** $s \in S$ **do** ▷ Melhoria de política
6: $\pi_{i+1}(s) \leftarrow \arg\min_{a\in\mathcal{A}}\left[\exp(\beta c(s,a))\sum_{s'\in\mathcal{S}} T(s'|s,a)V^{\pi_i}(s')\right]$
7: $i \leftarrow i + 1$

Quando $\beta < 0$ (propenso a risco) e a política π é própria, então π também é β-factível. Entretanto, não há garantia para toda política quando $\beta > 0$ (averso a risco). Dado um GDRSMDP, apenas o trabalho [Freire and Delgado 2016] demonstra como obter o valor de $\beta > 0$ que resulte em uma política β-factível o mais aversa a risco.

2.2.2. Algoritmo de Iteração de Valor Sensível a Risco

Apesar de não ter sido explicitamente definido no artigo de Patek (2001), é possível definir um algoritmo de Iteração de Valor Sensível a Risco em que em cada iteração i é calculado

o valor $V^i(s)$ baseado no valor $V^{i-1}(s)$ para cada estado $s \in S$, isto é:

$$V^i(s) = \min_{a \in A} \left[Q^i(s,a) \right], \tag{8}$$

em que:

$$Q^i(s,a) = \exp(\beta c(s,a)) \sum_{s' \in S} T(s'|s,a) V^{i-1}(s'). \tag{9}$$

Um possível critério de parada é considerar o resíduo $\max_{s \in S} |V_i(s) - V_{i-1}(s)|$ e iterar enquanto o resíduo for maior que um erro mínimo desejado ϵ. Esse algoritmo encontra uma solução se existe pelo menos uma política π_0 β-factível.

3. ILAO* Sensível a Risco

O algoritmo de Iteração de Política e de Iteração de Valor Sensíveis a Risco precisam atualizar o conjunto completo de estados em cada iteração, por esse motivo são muito custosos quando o problema tem muitos estados. Uma outra limitação do algoritmo Iteração de Política Sensível a Risco é a necessidade de ter uma política inicial π^0 β-factível para garantir a convergência do algoritmo.

Por outro lado para resolver MDPs dirigidos a meta clássicos, existe o algoritmo heurístico ILAO* [Hansen and Zilberstein 2001], um algoritmo eficiente que atualiza apenas o subconjunto de estados que são relevantes para o problema considerando que é conhecido o estado inicial s_0 a partir do qual desejamos chegar no estado meta. Esse algoritmo permite reduzir o número de vezes que um estado é avaliado e permite encontrar uma política parcial ótima. Neste trabalho é proposto uma adaptação desse algoritmo que é chamada de ILAO* Sensível a Risco (Risk Sensitive ILAO* – RS-ILAO).

A ideia principal do algoritmo RS-ILAO* é expandir e atualizar os estados alcançáveis, seguindo a política gulosa atual (i.e., a melhor política até o momento) e criando um *grafo explícito* G' o qual contém todos os estados visitados até então e todas as ações aplicáveis nesses estados, bem como os estados sucessores correspondentes. No grafo explícito G', cada nó representa um estado $s \in S$ e cada aresta representa uma ação $a \in A$. Nesse grafo, um estado folha (aquele que não tem sucessores descobertos) é uma folha terminal se é um estado meta, caso contrário é um estado não terminal.

O algoritmo RS-ILAO* consiste em dois passos: (i) **busca em profundidade** em que são feitas a expansão da melhor solução parcial, a atualização dos custos e a identificação das melhores ações criando um *grafo da melhor solução parcial* chamado de G''; e (ii) **teste de convergência** que chama o algoritmo de Iteração de Valor Sensível a Risco para todos os estados de G'' e verifica se o grafo da melhor solução muda.

O algoritmo RS-ILAO* (Algoritmo 2) recebe como entrada um GDRSMDP e o estado inicial s_0; e devolve como saída uma política parcial ótima. Nas Linhas 3 e 4 do Algoritmo 2, o grafo explícito G' é inicializado apenas com o estado inicial. As Linhas 6 a 11 do Algoritmo 2 são executadas enquanto o teste de convergência da Linha 11 não devolva verdadeiro. Esse teste de convergência é feito primeiro chamando o algoritmo de Iteração de Valor Sensível a Risco com o grafo G' (note que as atualizações são feitas no grafo da melhor solução G'') e caso haja mudança em G'' de modo que apareça alguma folha não esperada em G'', o algoritmo continua.

Se o algoritmo ainda não convergiu, é realizada uma busca em profundidade a partir do estado inicial s_0 enquanto o grafo explícito tiver alguma folha não terminal (Algoritmo 2, Linhas 7-10). Durante a busca em profundidade o valor dos estados expandidos e seus ancestrais são atualizados apenas uma vez e a melhor ação é identificada atualizando o grafo G''. Note que os estados marcados como visitados pela busca em profundidade devem ser atualizados para não visitados antes de fazer uma nova busca em profundidade (Algoritmo 2, Linhas 9-10).

Algorithm 2 : RS-ILAO* para GDRSMDP Sensível ao Risco

1: **function** RSILAO($S, s_0, S_g, A, T, c, \beta$)
Require: Um GDRSMDP e um estado inicial s_0
Ensure: Uma política ótima π representada pelo grafo da melhor solução G''
2: **define** variáveis globais: $S, s_0, S_g, A, T, c, \beta, V$
3: $G' \leftarrow$ Grafo Vazio
4: G'.nós$\leftarrow$ G'.nós $\cup\{s_0\}$
5: $convergido \leftarrow$ **False**
6: **while not** $convergido$ **do**
7: **while** G' contém alguma folha não terminal **do**
8: PROFUNDIDADERSILAO (G', s_0) ▷ Expande grafo G'
9: **for all** $s \in G'$ **do**
10: s.visitado = **False**
11: $convergido \leftarrow$ VERIFICARCONVERGENCIARSILAO(G')
return G''

O algoritmo PROFUNDIDADERSILAO (Algoritmo 3), que é chamado na Linha 8 do Algoritmo 2, é responsável tanto pela expansão dos nós do grafo explícito G' quanto pela atualização (revisão) dos custos calculados para os estados do grafo G'. A expansão não ocorre indiscriminadamente, ela ocorre usando a melhor solução parcial. O grafo da melhor solução parcial é um subgrafo do grafo explícito que inclui os nós e arestas que melhor atendem o critério de otimização. Neste algoritmo, se o estado s já foi expandido, todos os sucessores s' do estado s considerando a melhor solução parcial não visitados são expandidos de forma recursiva (Algoritmo 3, Linhas 3-6). Caso contrário, o estado s é expandido (Algoritmo 3, Linhas 7-8). Durante a busca em profundidade, o valor do estado s é atualizado (Algoritmo 3, Linha 9) e a melhor ação para o estado s no grafo explícito é marcada (Algoritmo 3, Linha 10). Note que esses estados e as melhores ações marcadas durante essa busca farão parte do grafo da melhor solução parcial G''.

O algoritmo VERIFICARCONVERGENCIARSILAO (Algoritmo 4), que é chamado na Linha 11 do Algoritmo 2, é responsável por avaliar a convergência e atualizar todos os estados pertencentes ao grafo G''. Na Linha 4, o algoritmo define o grafo da melhor solução parcial G'' como o subgrafo de G que inclui os estados e as melhores ações marcadas durante a busca em profundidade. O algoritmo VERIFICARCONVERGENCIARSILAO executa o algoritmo de Iteração de Valor Sensível a Risco no grafo G'', atualizando todos os estados de G'' enquanto o resíduo seja maior que o máximo erro desejado ϵ (Algoritmo 4, Linha 5-13). Se durante as iterações, algum estado folha em G'' for identificado como ainda não expandido, o algoritmo devolve falso, isto é não convergiu (Algoritmo 4, Linha 10-11).

Algorithm 3 : Busca em Profundidade para RS-ILAO*

1: **function** PROFUNDIDADERSILAO(G', s)
Require: Um grafo explícito G' e um estado $s \in S$
Ensure: Um grafo explícito com os nós expandidos a partir de s
2: s.visitado = **True**
3: **if** s.expandido **then**
4: **for all** $s' \in s$.sucessores(s.melhorAcao) **do**
5: **if not** s'.visitado **then**
6: PROFUNDIDADERSILAO (G', s')
7: **else**
8: EXPANDEESTADO (G', s) ▷ expande o estado s
9: $V(s) \leftarrow \min_{a' \in A} Q(s, a')$
10: s.melhorAcao $\leftarrow \arg\min_{a' \in A} Q(s, a')$

Algorithm 4 : Verificação de convergência do RS-ILAO*

1: **function** VERIFICARCONVERGENCIARSILAO(G', ϵ)
Require: Um grafo explícito G' e um valor máximo de erro
Ensure: Todos os nós do grafo G'' com sua função valor
2: $residuo \leftarrow +\infty$
3: $i \leftarrow 0$
4: $G'' \leftarrow$ melhor solução de G' ▷ G'' é a melhor solução parcial de G'
5: **while** $residuo \geq \epsilon$ **do**
6: **for all** $s \in G''$.nós **do**
7: $V_{i+1}(s) \leftarrow \min_{a \in A} Q(s, a)$
8: s.melhorAcao $\leftarrow \arg\min_{a \in A} Q(s, a)$
9: $G'' \leftarrow$ melhor solução de G'
10: **if** G'' tem alguma folha não expandida **then**
11: **return False**
12: $residuo \leftarrow \max_{s \in G''} |V_{i+1}(s) - V_i(s)|$
13: $i \leftarrow i + 1$
14: **return True**

O algoritmo Q (Algoritmo 5), que é chamado pelos Algoritmos 3 e 4, é responsável pelo cálculo do valor Q para um par estado-ação utilizando a função exponencial. O algoritmo EXPANDEESTADOS (Algoritmo 6), que é chamado pelo Algoritmo 3, é responsável por encontrar os sucessores do estado s e inicializar o valor desses estados com a heurística h.

4. Experimentos

Foram realizados experimentos no Java SE 8 1.8.0_51 rodando sob o Eclipse Oxygen.3a Release (4.7.3a) em um processador Intel Core i5 de 2,6 GHz, 8 GB de memória RAM 1600 MHz, 128 GB de armazenamento SSD. O primeiro grupo de experimentos analisa a capacidade do algoritmo RS-ILAO* encontrar políticas diferentes de acordo com o

Algorithm 5 : Função utilidade exponencial para RS-ILAO*

1: **function** Q(s, a)

Require: Um GDRSMDP, o fator de risco β, um estado $s \in S$ e uma ação $a \in A$.

Ensure: O custo estimado do estado s caso seja executada a ação a

2: **return** $\exp(\beta c(s,a)) \sum_{s' \in S} T(s'|s,a)V(s')$

Algorithm 6 : Função de expansão para RS-ILAO*

1: **function** EXPANDEESTADO(G', s)

Require: Um GDRSMDP, um grafo explícito G' e um estado $s \in S$.

Ensure: O grafo explícito G' expandido no estado s

2: $\mathcal{S}' \leftarrow s$.sucessores() ▷ Subconjunto S' com todos os sucessores de s

3: $V(s') \leftarrow \begin{cases} 1 & \text{se } s' \text{ for estado meta} \\ h(s') & \text{demais casos} \end{cases}, \forall s' \in \mathcal{S}'$

4: G'.nós$\leftarrow G'$.nós$\cup \mathcal{S}'$ ▷ Adiciona os estados $\mathcal{S}'$ como nós do grafo

5: s.expandido $\leftarrow$ **True**

parâmetro de sensibilidade ao risco, incluindo a política de aversão extrema ao risco, considerando o valor de β encontrado pelo algoritmo proposto em [Freire and Delgado 2017]. O segundo grupo de experimentos compara o tempo de execução do algoritmo em função do tamanho do problema. Os experimentos foram realizados no domínio de Travessia do Rio [Freire and Delgado 2017] descrito a seguir.

4.1. Domínio Travessia do Rio

No domínio da Travessia do Rio [Freire and Delgado 2017], um robô anfíbio deve atravessar um rio. O robô deve começar a partir de uma margem esquerda até um certo ponto na margem direita. É possível atravessar o rio pela água ou através de uma ponte localizada no extremo norte. No extremo sul há uma cachoeira que, caso seja alcançada, leva o robô para o ponto de partida do percurso. É possível mover-se para as 4 direções (Norte, Oeste, Leste, Sul) ou ainda ficar parado, as ações são representadas por $\{\uparrow, \leftarrow, \rightarrow, \downarrow, \circ\}$, respectivamente. Cada ação em água tem custo $c_{agua} = 1$ e cada ação em terra ou na ponte tem custo $c_{terra} = 4$.

A Figura 1 apresenta um exemplo deste domínio em coordenadas (x, y) de tamanho 14×5 e seu mapeamento em 70 estados. Os estados $(1, y)$ e $(5, y)$ representam as margens esquerda e direita, respectivamente; os estados (1,1) e (5,1) representam o ponto de recomeço e o final do trajeto, respectivamente; os estados (2,14), (3,14) e (4,14) representam a ponte disponível para travessia; os estados (2,1), (3,1) e (4,1) representam a cachoeira; os demais estados representam o rio.

Cada comando enviado ao robô em terra ou na ponte tem uma probabilidade de 99% de ser obedecido e 1% de não ser realizado por algum problema técnico, o resultado deste problema é que o robô não irá se mover e permanecerá no mesmo lugar. Cada comando enviado ao robô na água é 80% provável de ser executado com precisão e 20% provável de não ser realizado, fazendo com que o robô seja transportado pela correnteza para a posição $(x, y - 1)$. Para os experimentos foi usada a distância de Manhattan como heurística.

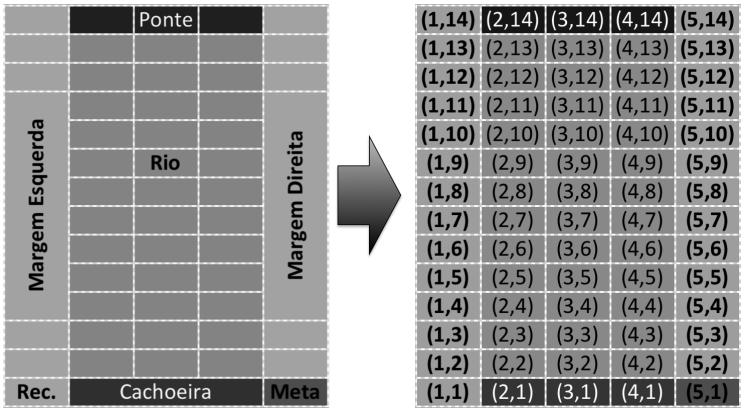

Figura 1. Travessia do rio com grid 14×5 e seu mapeamento em estados

Se a distância para chegar na ponte for muito maior do que a largura do rio, é muito provável que um agente neutro opte por atravessar o rio em busca do menor custo médio. Se o rio for demasiado largo, este mesmo agente pode optar ir pela ponte, se o custo esperado for menor. Um agente propenso a riscos escolheria ir pelo rio independente de sua largura, uma vez que ele não está preocupado com segurança. Por outro lado, um agente averso ao risco sempre escolheria usar a ponte por causa da segurança que esta rota proporciona.

4.2. Obtenção de diferentes políticas de acordo com a sensibilidade ao risco

Como esperado, o algoritmo RS-ILAO* consegue encontrar a política ótima de acordo com o fator de risco β escolhido, isto é, quanto maior o fator de risco mais a busca por um caminho mais seguro se torna importante no momento da definição da política. Diferente do algoritmo de Iteração de Política Sensível a Risco que encontra uma política total, o algoritmo RS-ILAO* encontra uma política parcial. Foi verificado que o valor encontrado pelo RS-ILAO* para os estados pertencentes a essa política parcial são idênticos aos encontrados pelo algoritmo de Iteração de Política Sensível a Risco.

A Tabela 1 apresenta as políticas encontradas pelo algoritmo RS-ILAO* para o domínio de Travessia do Rio em um grid de tamanho 14×5 para quatro fatores de risco diferentes ($\beta = -0,015$, $\beta = 0,0001$, $\beta = 0,030$ e $\beta = 0,040$). Os resultados dos experimentos demonstram que para $\beta = -0,015$ o agente busca o trajeto mais barato, mesmo que seja mais arriscado; enquanto que para $\beta =$ 0,040, o agente busca pela segurança da ponte, mesmo que o trajeto seja quatro vezes mais caro do que o trajeto pela água.

4.3. Comparação de desempenho

A Figura 2 apresenta uma comparação do tempo médio de convergência dos algoritmos de Iteração de Política Sensível a Risco e o algoritmo RS-ILAO* para grids de tamanho 7×3, 7×5, 12×5, 14×5, 20×5, 20×7, 25×7 e 25×8 e $\beta = 0,1$. Para calcular o tempo médio, o algoritmo foi executado 5 vezes para cada tamanho de grid. Os experimentos mostram que o algoritmo RS-ILAO* converge mais rápido que o algoritmo de Iteração de Política Sensível a Risco para todos os tamanhos de grid, sendo até 21 vezes mais rápido.

Tabela 1. Políticas encontradas pelo algoritmo RS-ILAO* para o domínio Travessia do Rio em um grid 14×5 com fatores de risco β distintos

$\beta = -0,015$	$\beta = 0,0001$	$\beta = 0,030$	$\beta = 0,040$
			→ → → → ↓
			↑ ↓
			↑ ↓
		→ → → → ↓	↑ ↓
		↑ → → → ↓	↑ ↓
		↑ → → → ↓	↑ ↓
		↑ → → → ↓	↑ ↓
		↑ ← → → ↓	↑ ↓
	→ → → → ↓	↑ ← → → ↓	↑ ↓
	↑ → → → ↓	↑ ← → → ↓	↑ ↓
	↑ → → → ↓	↑ ← → → ↓	↑ ↓
	↑ ← → → ↓	↑ ← → → ↓	↑ ↓
→ → → → ↓	↑ ← → → ↓	↑ ← → → ↓	↑ ↓
↑ ○ ○ ○ ○	↑ ○ ○ ○ ○	↑ ○ ○ ○ ○	↑ ○

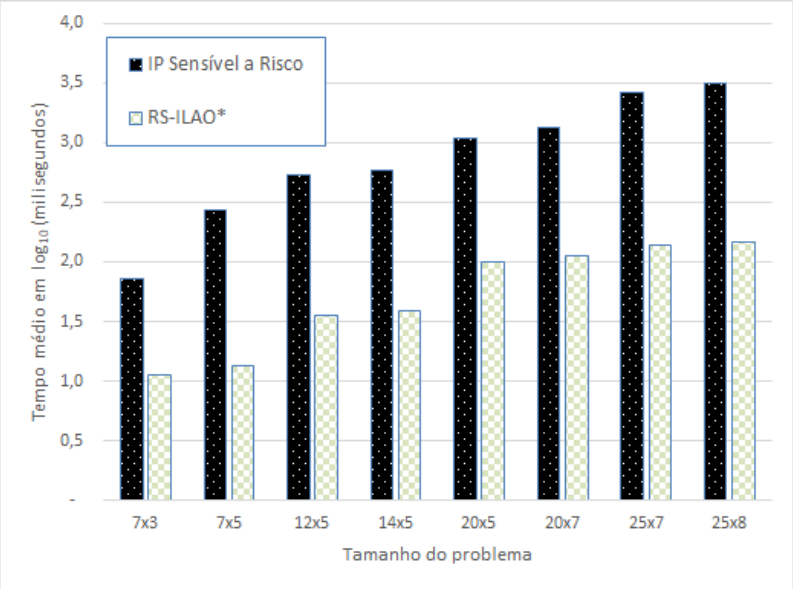

Figura 2. Tempo médio de convergência para o Domínio de Travessia do Rio

5. Conclusão

Neste artigo foi apresentado o algoritmo RS-ILAO* para resolver MDPs sensíveis a risco e dirigidos à meta. Esse algoritmo é uma adaptação do algoritmo ILAO* [Hansen and Zilberstein 2001] que é um algoritmo de busca heurística para MDPs dirigidos à meta que foca apenas nos estados que são relevantes a partir do estado inicial.

Foram realizados experimentos no domínio de Travessia do Rio para analisar a capacidade do algoritmo proposto de encontrar políticas diferentes de acordo com o fator de risco β utilizado. Os experimentos mostram que o algoritmo RS-ILAO*, consegue encontrar políticas ótimas para esse domínio de acordo com o valor β escolhido, além de ser até 21 vezes mais rápido que o algoritmo de Iteração de Política Sensível a Risco, o qual ainda precisa de uma política inicial β-factível para garantir convergência.

Em trabalhos futuros pretende-se demostrar que o algoritmo RS-ILAO* converge após um número finito de iterações se a função heurística usada for admissível.

Agradecimentos

Os autores agradecem à Capes pela concessão da bolsa de mestrado para as atividades de pesquisa e à FAPESP pelo apoio financeiro (processo #2015/01587-0).

Referências

Bertsekas, D. P. and Tsitsiklis, J. N. (1991). An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595.

Denardo, E. V. and Rothblum, U. G. (1979). Optimal stopping, exponential utility, and linear programming. *Mathematical Programming*, 16(1):228–244.

Filar, J. A., Kallenberg, L. C. M., and Lee, H.-M. (1989). Variance-penalized Markov decision processes. *Mathematics of Operations Research*, 14(1):147–161.

Filar, J. A., Krass, D., Ross, K. W., and Ross, K. W. (1995). Percentile performance criteria for limiting average Markov decision processes. *IEEE Transactions on Automatic Control*, 40(1):2–10.

Freire, V. and Delgado, K. V. (2016). Extreme risk averse policy for goal-directed risk-sensitive Markov decision process. In *5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 79–84.

Freire, V. and Delgado, K. V. (2017). GUBS: A utility-based semantic for goal-directed markov decision processes. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 741–749.

Geffner, H. and Bonet, B. (2013). A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141.

Hansen, E. A. and Zilberstein, S. (2001). Lao*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1):35 – 62.

Howard, R. A. and Matheson, J. E. (1972). Risk-sensitive markov decision processes. *Management science*, 18(7):356–369.

Jaquette, S. C. (1976). A utility criterion for Markov decision processes. *Management Science*, 23(1):43–49.

Moldovan, T. M. and Abbeel, P. (2012). Risk aversion in Markov decision processes via near optimal Chernoff bounds. In *Advances in Neural Information Processing Systems, NIPS 2012*, pages 3131–3139.

Patek, S. D. (2001). On terminating markov decision processes with a risk-averse objective function. *Automatica*, 37(9):1379–1386.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rothblum, U. G. (1984). Multiplicative Markov decision chains. *Mathematics of Operations Research*, 9(1):6–24.

Sobel, M. J. (1982). The variance of discounted Markov decision processes. *Journal of Applied Probability*, 19(4):794–802.