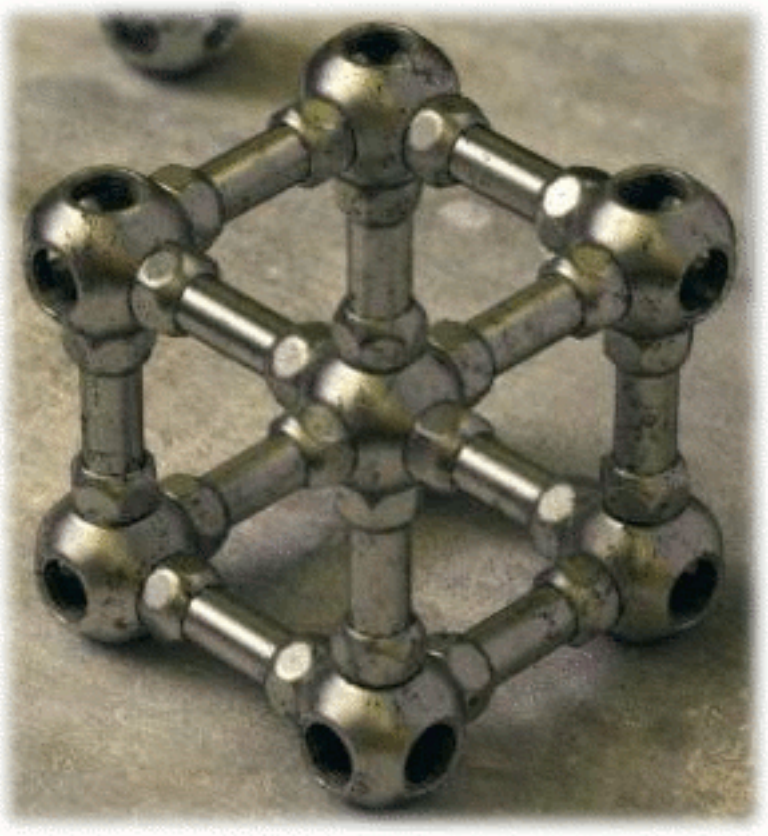

PROCESSOS DE DECISÃO MARKOVIANOS COM PROBABILIDADES IMPRECISAS: UMA SOLUÇÃO COM PROGRAMAÇÃO EM DOIS NÍVEIS [1]

Author(s):
Karina Valdivia Delgado
Leliane Nunes de Barros

[1]This work was supported by Fapesp Project LogProb, grant 2008/03995-5, São Paulo, Brazil.

PROCESSOS DE DECISÃO MARKOVIANOS COM PROBABILIDADES IMPRECISAS: UMA SOLUÇÃO COM PROGRAMAÇÃO EM DOIS NÍVEIS

Karina Valdivia Delgado*, Leliane Nunes de Barros*

*Universidade de São Paulo
Rua de Matão 1010, Butantã
São Paulo, SP, Brasil

Emails: kvd@ime.usp.br, leliane@ime.usp.br

Resumo— Os algoritmos exatos de Programação Dinâmica (iteração de valor e iteração de política) propostos nas décadas de 70 e 90 para resolver um MDPIP (*Markov Decision Process with Imprecise Probabilities*) apresentam desempenho limitado principalmente por adotar uma representação enumerativa de estados. Recentemente, foram propostas soluções aproximadas para MDPIPs com representação fatorada de estados e uso de estruturas de dados que permitem cálculos eficientes usando Programação Dinâmica, isto é, soluções capazes de resolver problemas com milhares de estados. Além disso, foi proposto um método de programação multilinear aproximado para resolver MDPIPs fatorados capaz de, em certas condições, superar o desempenho das soluções baseadas em Programação Dinâmica. Nesse trabalho, mostramos uma nova solução para MDPIPs baseada em programação matemática: a programação fatorada e aproximada em dois níveis.

Palavras-chave— Processos Markovianos de Decisão (MDPs), tomada de decisão seqüencial, probabilidades imprecisas

1 Introdução

A tomada de decisão sequencial é uma atividade importante e tem sido estudada nas áreas de pesquisa operacional, planejamento e robótica. Processos Markovianos de Decisão (MDPs) (Puterman, 1994) (Bellman, 1957) (Russell and Norvig, 2003) fornecem um arcabouço matemático para modelar a tomada de decisões sequencial em domínios estocásticos e tempo discreto. Um MDP modela a interação entre um agente e seu ambiente. A cada instante o agente (robótico ou de software) faz uma escolha de ações (com efeitos probabilísticos) e decide executar uma ação que produzirá um estado futuro e uma recompensa. O objetivo do agente é maximizar a recompensa ganha ao longo de uma sequência de escolhas de ações. Por exemplo, no dominio do robô entregador de café (Boutilier et al., 1999)(Figura 1), suponha que o robô está num estado em que sua locação é o quarto 2, o Carlos quer as impressões, o Marcio quer café e a bateria do robô está baixa. Nesse instante o robô tem que escolher uma ação (com efeitos probabilísticos) entre: *ir para a estação de recarga*, *ir para a cozinha* ou *ir para a impressora*. Por exemplo, a ação *ir para a impressora* têm uma probabilidade de 0.95 de sucesso e 0.05 de falhar. Suponha que o robô escolha executar a ação *ir para a impressora*, e como resultado da aplicação da ação ele falha ao pegar as impressões obtendo uma recompensa de -10 unidades. Esse problema pode ser modelado com um MDP ajudando o robô a decidir qual é a melhor ação a ser escolhida em cada estado. Extensões conhecidas de um MDP são mais adequadas para representar problemas práticos de maior interesse para aplicações reais. Entre elas: (1) um MDP em que estados são representados por variáveis de estado, chamado de MDP fatorado, que permite representar o espaço de estados de maneira mais compacta; (2) um MDP cujas probabilidades não são completamente conhecidas, chamado de MDPIP (*Markov Decision process with Imprecise Probabilities*). Nesse artigo, estamos interessados em mostrar como resolver um **MDPIP fatorado** formulado como um problema de programação matemática de dois níveis.

Figura 1: Robô entregador de café

Na Seção 2 introduzimos os principais conceitos relativos a um MDPIP. Na Seção 3 apresentamos a definição de um MDPIP fatorado e a sua formulação usando programação em dois níveis. Além disso, avaliamos os possíveis métodos para resolver esse problema de programação. Finalmente, nas Seções 4 e 5 mostramos um domínio de aplicação e um exemplo completo usando um dos métodos avaliados.

2 MDP com Probabilidades Imprecisas

Em geral, as probabilidades que refletem a escolha da natureza não são precisas. Assim, um maior realismo pode ser dado para os MDPs, permitindo que eles representem imprecisão na determinação das probabilidades de transição. Dessa forma, podemos

representar crenças incompletas, ambíguas ou crenças conflitantes de especialistas sobre um sistema de transições de estados.

2.1 Definição de MDPIPs

Os processos markovianos de decisão com probabilidades imprecisas (MDPIP) (White III and Eldeib, 1994) (Satia and Lave Jr, 1973) representam uma extensão dos MDPs, nos quais existe imprecisão nas probabilidades de transição. As probabilidades associadas a cada transição são definidas por um conjunto de inequações lineares. Esse conjunto de probabilidades é chamado de conjunto credal K, que é definido sobre o espaço de estados (Cozman, 2000). Formalmente, a definição de um MDPIP é descrita pela tupla $\mathcal{M} = (T, S, A, R, K)$, em que:

- T é um conjunto enumerável de estágios (instantes do processo). Uma decisão é tomada a cada estágio. Assim, podemos fazer uma analogia entre os pontos de decisão de um processo sequencial com uma representação de tempo discreto. Assumimos que o estágio inicial é $t = 0$.

- S é um conjunto finito de estados. Se o processo tem conhecimento total do estado atual, então o MDP é chamado de *totalmente observável*, caso contrário, é *parcialmente observável* (POMDP). Neste trabalho, estamos interessados em MDPs totalmente observáveis.

- A é um conjunto finito de ações que permitem que o sistema mude de um estado para outro. A *tomada de decisão* significa a escolha de uma ação, dado que o agente se encontra num determinado estado.

- $R : S\text{x}A \rightarrow \mathbb{R}$ é a função de recompensa, sendo que $R(s, a)$ representa a recompensa obtida pelo agente, dado que ele está no estado $s \in S$ e executa a ação $a \in A$. Neste trabalho, assumimos que a função de recompensa é estacionária (isto é, independe do estágio).

- $K_a(s'|s)$ define as transições markovianas sobre S, que são os conjuntos condicionais credais válidos representados por inequações lineares para expressar todas as possíveis distribuições de probabilidade $P(s'|a, s)$.

Dadas as características de um MDPIP, é possível definir vários critérios para avaliar uma política. Nesse trabalho, estamos interessados em políticas que levem o agente para a maior recompensa esperada, assumindo que as *escolhas da natureza* (para definir as imprecisões) pretendem minimizar a recompensa esperada do agente, isto é, o critério Γ-maxmin. Em (Satia and Lave Jr, 1973) foram mostrados vários resultados importantes para MDPIPs que adotam esse critério, a saber:

- Existe uma política estacionária determinística que é ótima.
- A política ótima produz uma função valor ótima, representada por V^*, que é a única solução da equação de Bellman dada por:

$$V^*(s) = \max_{a \in A} \min_{P \in K_a(s'|s)} \{R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a)V^*(s')\} \quad (1)$$

Na Equação (1) o agente escolhe a ação a que maximiza o valor de $V(s)$ e a natureza escolhe a probabilidade que minimiza $V(s)$. O fator de desconto γ assume valores entre 0 e 1 e é usado para dar menos importância a recompensas futuras.

Definição 1 *(MDPIP - Modelo Conceitual) (Satia and Lave Jr, 1973)*
Um MDPIP modela a interação entre um agente e seu ambiente. A cada instante o agente faz uma escolha de ações (com efeitos probabilísticos) e decide executar uma ação que produzirá um estado futuro e uma recompensa que depende também das escolhas da natureza. O objetivo do agente é maximizar a recompensa ganha ao longo de uma sequência de escolhas de ações assumindo que a escolha da natureza minimiza sua recompensa.

Algoritmos de iteração de valor e iteração de política (White III and Eldeib, 1994) foram propostos para MDPIPs, que usam a representação enumerativa de estados. Dada a ineficiência dessas soluções (a enumeração de todos os estados leva a uma explosão combinatória) foram propostas soluções fatoradas e aproximadas para MDPIPs com representação fatorada e uso de estruturas de dados que permitem cálculos eficientes usando Programação Dinâmica (Delgado, Sanner, de Barros and Cozman, 2009), isto é, resolver problemas com milhares de estados. Além disso, em (Delgado, de Barros, Cozman and Shirota, 2009) foi proposto um método de programação multilinear e aproximado para resolver MDPIPs fatorados, que pode superar o desempenho da solução baseada em Programação Dinâmica. Nesse trabalho, mostramos uma outra solução para MDPIPs baseada em programação matemática: a programação em dois níveis.

2.2 Formulação em dois níveis para MDPIPs

A Equação (1) pode ser reduzida a um problema de programação em dois níveis (Shirota et al., 2007):

$$\begin{aligned} \min_{V^*} \quad & : \quad \sum_s V^*(s) \qquad (2) \\ \textit{Sujeito a} \quad & : \quad V^*(s) \geq R(s, a) + \\ & \gamma \sum_{s' \in S} P(s'|s, a)V^*(s'), \forall s \in S, a \in A. \\ & P \in \textit{argmin} \sum_{s' \in S} P(s'|s, a)V^*(s'). \\ & \textit{Sujeito a} : P(s'|s, a) \in K_a(s'|s) \end{aligned}$$

O Problema 2 é um problema de programação em dois níveis pois define um problema de minimização dentro de outro problema de minimização.

3 Definição de MDPIP fatorado

Chamamos de *MDPIP fatorado* um MDP em que os estados são descritos usando variáveis de estado X_i para $i = 1..n$; as transições são representadas por Redes Credais Dinâmicas e as funções valor e recompensa também são fatoradas (Delgado, de Barros, Cozman and Shirota, 2009).

Função transição de estado fatorada. Uma Rede Credal Dinâmica (*Dynamic Credal Network* - DCN) é uma representação mais compacta do modelo de transição de estados para um MDPIP fatorado. Uma rede credal generaliza o conceito de rede bayesiana, permitindo que cada variável, para cada configuração de seus pais, seja associada a um conjunto de densidades de probabilidade, no lugar de uma simples densidade, como acontece com as redes bayesianas (Cozman, 2000).

Num MDPIP fatorado definimos uma rede credal dinâmica para cada ação a, pela tupla $\tau_a =< G_a, \mathbb{K}_a >$, sendo G_a um grafo cíclico dirigido de duas camadas, uma camada representando o estado atual e uma camada representando o próximo estado (exatamente como nas redes bayesianas dinâmicas). Os nós estão associados a variáveis de estado X_i e $\mathbb{K}_a$ é uma coleção de conjuntos credais condicionais $K_a(X_i'|s[parents(X_i')])$. Neste caso, temos um conjunto de distribuições conjuntas de probabilidade, cada uma satisfazendo a seguinte equação:

$$P(s'|s,a) = \prod_i P(x_i'|s[Parents(X_i')], a) \quad (3)$$

Função recompensa fatorada. Em MDPIPs fatorados, para definir a recompensa de aplicar a ação a no estado s é usado um conjunto de funções de recompensas locais R_i que dependem de um subconjunto de variáveis de estado $D_i(a) \subset \{X_1, ..., X_n\}$. Seja k_R a quantidade de subconjuntos de s para os quais são definidas funções de recompensa local. A recompensa para um estado s dado que a ação a foi executada é definida como:

$$R(s,a) = \sum_{i=1}^{k_R} R_i(D_i(a), a) \in \mathbb{R} \quad (4)$$

3.1 Funções base e função valor aproximada

A função valor é aproximada por uma combinação linear de funções base $h_1, ..., h_k$, ou seja:

$$\widehat{V}(s) = \sum_{j=1}^{k} w_j h_j(s)$$

Uma condição necessária para a realização de cálculos eficientes (Koller and Parr, 1999), e que será considerada nesse trabalho, é que o escopo de cada h_i esteja restrito a algum subconjunto de variáveis de estado $C_i \subset \{X_1, ..., X_n\}$. Sendo que C_i não está necessariamente relacionado com D_i. Note que se permitimos que as funções base tenham domínios de tamanho ilimitado, poderíamos cair no extremo de usar todas as variáveis de estado, o que estaria em contradição com o objetivo de ter uma representação compacta (Patruscu, 2004).

Conhecendo o conjunto H de funções base, o objetivo agora passa a ser calcular o conjunto de pesos w de modo a obter uma boa aproximação da função valor ótima. Para permitir que os cálculos sejam feitos mais eficientemente, algumas suposições devem ser estabelecidas (Guestrin, 2003): i) cada função base depende de algumas (poucas) variáveis de estado; ii) no modelo de transição, as variáveis do próximo estado dependem de algumas (poucas) variáveis da camada do estado atual; iii) as recompensas locais dependem de algumas (poucas) variáveis de estado e iv) o número de ações é pequeno.

Apesar dessas suposições parecerem, a princípio muito restritivas, existem muitos problemas práticos que podem ser modelados fazendo-se essas suposições (Guestrin, 2003).

3.2 Programação em dois níveis para MDPIP fatorado

Usando a função valor fatorada como foi proposta na Seção 3.1 e substituindo-a no problema da Equação (2) temos:

$$\begin{aligned}
\min_{w} \quad & : \quad \sum_s \sum_{i=1}^{k} w_i h_i(s) \quad (5)\\
Sujeito\ a \quad & : \quad \sum_{i=1}^{k} w_i h_i(s) \geq R(s,a) + \\
& \quad \gamma \sum_{s' \in S} P(s'|s,a) \sum_{i=1}^{k} w_i h_i(s'), \\
& \quad \forall s \in S, a \in A. \\
& \quad P \in argmin \sum_{s' \in S} P(s'|s,a) \sum_{i=1}^{k} w_i h_i(s'). \\
& \quad Sujeito\ a : P(s'|s,a) \in K_a(s'|s)
\end{aligned}$$

Em que:

$$P(s'|s,a) = \prod_i P(x_i'|s[Parents(X_i')], a)$$

Note que o problema resultante possui as seguintes características:

CA1: A função objetivo do primeiro nível é linear.

CA2: As restrições do primeiro nível são não-lineares.

CA3: A função objetivo do segundo nível é não-linear.

CA4: As restrições do segundo nível são lineares.

CA5: É um problema de programação em dois níveis (*Bi-level Probabilistic Program - BLPP*) geral.

CA6: As restrições do primeiro nível contém variáveis do segundo nível (que correspondem às probabilidades).

BACKPROJECTION_$g_i^a(h_i, P_a)$
1 /* Definição do escopo de Backproj :
2 $\Gamma_a(\mathbf{C}') = \cup_{X'_j \in \mathbf{C}'} Parents_a(X'_j)$
3 **para** *cada atribuição* $\mathbf{y} \in \Gamma_a(\mathbf{C}')$ **faça**
4 $g_i^a(\mathbf{y}) = \sum_{\mathbf{c}' \in \mathbf{C}'} \prod_{j|X'_j \in \mathbf{C}'} P_a(\mathbf{c}'[X'_j]|\mathbf{y}) h_i(\mathbf{c}')$

Figura 2: Algoritmo BACKPROJECTION_g_i^a.

3.3 Operação Backprojection

Uma das operações que pode ser executada eficientemente explorando a estrutura do MDP fatorado é um pré-cálculo chamado de *backprojection*, realizado em (Guestrin, 2003) para MDPS, e que pode ser aplicado na resolução de MDPIPs fatorados.

Sendo as restrições do problema associado ao MDPIP fatorado da forma:

$$\sum_i w_i h_i(s) \geq R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \sum_i w_i h_i(s'), \quad \forall s \in S, a \in A \tag{6}$$

com $P(s'|s,a) \in K_a(s'|s)$. Essas restrições podem ser reescritas da seguinte maneira:

$$\sum_{i=1}^{k} w_i h_i(s) \geq R(s,a) + \gamma \sum_{i=1}^{k} w_i \sum_{s' \in S} P(s'|s,a) h_i(s')$$

$$\sum_{i=1}^{k} w_i h_i(s) \geq R(s,a) + \gamma \sum_{i=1}^{k} w_i g_i^a(s).$$

$$\sum_{i=1}^{k} w_i (\gamma g_i^a(s) - h_i(s)) \leq -R(s,a)$$

$$\sum_{i=1}^{k} w_i c_i^a(s) \leq -R(s,a)$$

Em que:

$$P(s'|s,a) \in K_a(s'|s)$$
$$P(s'|s,a) = \prod_i P(x'_i|s[Parents(X'_i)], a)$$
$$g_i^a(s) = \sum_{s' \in S} P(s'|s,a) h_i(s')$$
$$c_i^a(s) = \gamma g_i^a(s) - h_i(s)$$

Ainda que as probabilidades sejam variáveis, a operação *backprojection* que explora a estrutura de MDPs fatorados (Guestrin, 2003) pode ser utilizada para MDPIPs fatorados, uma vez que a única diferença é que num MDPIP consideramos $P(s'|s,a)$ como uma variável que pertence a $K_a(s'|s)$. A partir de g_i^a podemos calcular também c_i^a. O algoritmo BACKPROJECTION_g_i^a da Figura (2), permite calcular g_i^a em que a função base h_i tem escopo C e o modelo de transição da ação a é P_a. Pode-se mostrar que se h_i tem escopo restrito a C, então a operação *backprojection* tem escopo restrito aos pais de C' na rede bayesiana dinâmica da ação a (Koller and Parr, 1999) (passo 2 do algoritmo da Figura 2). O cálculo de g_i^a é feito para todas as configurações $\mathbf{y}$ possíveis das variáveis do subconjunto $\Gamma_a(\mathbf{C}')$ (passo 3). No passo 4, j toma os valores do índice das variáveis de estado que pertencem ao escopo de h_i, sendo que cada $\mathbf{c}'$ é uma configuração das variáveis de estado que pertencem ao escopo de h_i.

3.4 Como resolver o programa de dois níveis?

É importante notar que podemos usar a operação *backprojection* para calcular as restrições do primeiro nível e a função objetivo do segundo nível do Problema (5). A maioria dos algoritmos para BLPP tenta resolver programas em dois níveis somente para BLPP linear, BLPP quadrático ou BLPP linear-quadrático (o que não se aplica para nosso problema devido a CA5), por exemplo, o algoritmo *K-th Best* (Bialas and Karwan, 1978). Outros algoritmos resolvem BLPP gerais cujas restrições do primeiro nível não contém variáveis do segundo nível (o que não se aplica devido a CA6), por exemplo: o algoritmo *branch-and-bound* (Bard, 1988), o algoritmo de regiões de confiança (Colson et al., 2005) e o algoritmo de restauração inexata (Andreani et al., 2007) . Outros resolvem BLPP gerais sem restrições no primeiro nível (o que não se aplica devido a CA2), por exemplo, o método de máxima descida (Savard and Gauvin, 1994), em que a direção máxima de descida é a solução de um BLPP quadrático. Em suma, todos esses algoritmos não podem ser usados para resolver o BLPP associado ao MDPIP fatorado, devido às características CA2, CA5 e CA6 de nosso problema.

Um método que pode ser usado para resolver esse problema em dois níveis é a reformulação usando as condições de Karush-Kuhn-Tucher (KKT) (Bard and Falk, 1982). Esse método transforma o problema em dois níveis:

$$\begin{aligned} \min_{x,y} \quad & : \quad F(x,y) \\ s.t. \quad & : \quad G(x,y) \leq 0 \\ & \quad\; y \in argmin_{y'} f(x,y') \\ & \quad\; s.t. : g(x,y') \leq 0 \end{aligned} \tag{7}$$

no seguinte problema não-linear novo, substituindo o problema do segundo nível por suas condições KKT (Bard and Falk, 1982):

$$\begin{aligned} \min_{x,y,\lambda} \quad & : \quad F(x,y) \\ s.t \quad & : \quad G(x,y) \leq 0 \\ & \quad\; g(x,y) \leq 0 \\ & \quad\; \lambda_i \geq 0, \; i = 1, ..., m_2 \\ & \quad\; \lambda_i g_i(x,y) = 0, \; i = 1, ..., m_2 \\ & \quad\; \nabla_y L(x,y,\lambda) = 0 \end{aligned} \tag{8}$$

$$where \quad : \quad L(x,y,\lambda) = f(x,y) + \sum_{i=1}^{m_2} \lambda_i g_i(x,y)$$

Depois disso, o novo problema não-linear (Problema (8)) pode ser resolvido usando um *solver* não-linear. No Problema (8) L é a função de Lagrange associada ao problema do segundo nível, i.e., o método cria um multiplicador de Lagrange λ_i para cada restrição do problema do segundo nível. Uma característica dessa reformulação é que existem mais variáveis no novo problema: $\lambda_i\ i = 1, ..., m_2$ (em que m_2 é o número de restrições no segundo nível). Uma outra caraterística dessa reformulação é que cresce o número de restrições: o número de restrições no novo problema não-linear relacionado ao MDPIP fatorado é $|S|*|A|+3*m_2+VarProb$, em que $VarProb$ é o número de variáveis de probabilidade no segundo nível. Além disso, para fazer a reformulação é necessário calcular as derivadas parciais para cada variável do segundo nível.

Uma caraterística importante do método KKT é que o novo problema de programação não-linear não é sempre equivalente ao problema de programação em dois níveis; é equivalente somente quando o problema do segundo nível do programa em dois níveis é convexo e regular (Bard and Falk, 1982).

Em particular, um MDPIP fatorado formulado como um problema de programação em dois níveis é equivalente ao novo problema não-linear, usando esse método, se as funções base tem escopo restrito a uma variável de estado, conforme demostrado a seguir.

Proposição 1 *(Equivalência do problema de dois níveis com o problema não-linear após a transformação KKT para problemas com funções base com escopo restrito a uma variável de estado).*

Para provar essa afirmação, se analisarmos o Problema (5), podemos ver que: i) as restrições no segundo nível são inequações lineares que expressam todas as possíveis distribuições de probabilidade (segundo a definição de $K_a(s'|s)$) e ii) a função objetivo do segundo nível, quando as funções base tem escopo restrito a uma variável de estado, é linear nas variáveis de probabilidade. Assim, neste caso, o segundo nível é convexo e regular e portanto aplicando a reformulação usando as condições KKT obtemos um problema equivalente. Depois desta transformação, podemos usar um solver *não-linear para obter w e $\vec{p}$. Ainda assim, existem varios domínios que podem ser modelados usando funções base com escopo restrito a uma variável de estado.*

4 Modelando um MDPIP fatorado para o exemplo do Administrador de Sistemas.

Nesta seção descrevemos o domínio do Administrador de Sistemas (Guestrin, 2003) modelado como um MDPIP fatorado, isto é, descrevemos um conjunto de Redes Credais Dinâmicas, a função recompensa fatorada e a função valor aproximada.

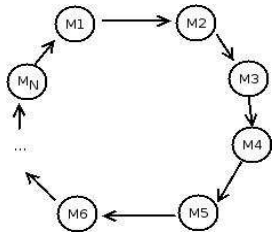

Figura 3: Configuração cíclica do domínio do Administrador de Sistemas

Exemplo 1 *Administrador de Sistemas (Guestrin, 2003). Considere o problema de otimizar o comportamento de um administrador de uma rede contendo n computadores, cada um conectado a um subconjunto de computadores. Várias topologias podem ser definidas desta maneira. Por exemplo, numa rede simples podemos conectar as máquinas de maneira cíclica, em que a máquina i está conectada à máquina $i+1$ e $i-1$. Cada máquina influenciando uma única máquina (Figura 3). Nesta rede, apenas um dos computadores é o servidor. Cada máquina está associada a uma variável binária X_i, representando se ela está funcionando ou não. Em cada passo do tempo o administrador recebe um pagamento (recompensa) por cada máquina que estiver funcionando. Dado que o servidor é o computador mais importante na rede, é dada uma recompensa maior se ele estiver funcionando. O trabalho do administrador do sistema é decidir qual das máquinas reinicializar. Assim, existem n+1 possíveis ações em cada passo do tempo: reinicializar uma das n máquinas ou não reinicializar nenhuma (esta última será chamada de a_0). Após executar a ação reinicializar a máquina i, que chamaremos de a_i, a probabilidade da máquina i estar funcionando no próximo passo no tempo é alta. Em cada passo cada computador tem uma probabilidade baixa de deixar de funcionar, que cresce dramaticamente se seus vizinhos não estão funcionando. As máquinas podem começar a funcionar espontaneamente com uma pequena probabilidade.*

Rede Credal Dinâmica para o exemplo do Administrador. O grafo de transição da rede credal dinâmica para a ação a_0 é mostrado na Figura 4. Nesta configuração os pais de X_i' são X_i e X_{i-1}. As probabilidades de transição para a ação $a_j, j \neq i$ são:

$$
\begin{array}{lcl}
P(X_i' = 1|X_i = 1, X_{i-1} = 1, a_j) & \geq & 0,9 \\
P(X_i' = 0|X_i = 1, X_{i-1} = 1, a_j) & \leq & 0,1 \\
P(X_i' = 1|X_i = 1, X_{i-1} = 0, a_j) & = & 0,667 \\
P(X_i' = 0|X_i = 1, X_{i-1} = 0, a_j) & = & 0,333 \\
P(X_i' = 1|X_i = 0, X_{i-1} = 0, a_j) & = & 0,01 \\
P(X_i' = 0|X_i = 0, X_{i-1} = 0, a_j) & = & 0,99 \\
P(X_i' = 1|X_i = 0, X_{i-1} = 1, a_j) & = & 0,01 \\
P(X_i' = 0|X_i = 0, X_{i-1} = 1, a_j) & = & 0,99
\end{array}
$$

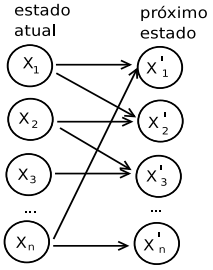

Figura 4: Grafo de transição do rede credal dinâmica da configuração cíclica do problema do Administrador de Sistemas para a ação a_0.

O grafo de transição da rede credal dinâmica para a ação a_i é igual ao grafo da Figura 4 exceto que a variável X_i não depende de nenhuma outra variável e:

$$P(X_i' = 1|X_i, X_{i-1}, a_i) = 0,95 \tag{9}$$

Observe que existe imprecisão na determinação das seguintes probabilidades $P(X_i' = 1|X_i = 1, X_{i-1} = 1, a_j)$ e $P(X_i' = 0|X_i = 1, X_{i-1} = 1, a_j)$ para $j \neq i$.

Função recompensa fatorada. A função recompensa é decomposta em n recompensas locais, uma para cada máquina e igual para todas as ações, R_1, R_2, R_3,...,R_n. Cada máquina contribui com 1 para a recompensa, exceto o servidor que contribui com 2:

- $R_i(X_i = 1) = 1$ *e* $R_i(X_i = 0) = 0$ *para* $i = 1, 2, ..., n$ sendo que a máquina i não é o servidor.
- $R_i(X_i = 1) = 2$ *e* $R_i(X_i = 0) = 0$ se a máquina i é o servidor.

Função valor aproximada. Para esse exemplo, definimos a função base constante $h_0 = 1$ e as demais funções base h_i como: $h_i(X_i = 1) = 1$ e $h_i(X_i = 0) = 0$. Neste exemplo simplificado, h_i tem seu escopo restrito a uma única variável X_i.

5 Resolvendo uma instância do domíno do Administrador com 2 computadores

Considere uma instância do domínio do Administrador de Sistemas em configuração cíclica com 2 computadores M1 e M2 modelada com uma rede credal dinâmica. Sendo M1 o servidor. Existem 3 ações neste modelo a_0, a_1 e a_2. Atribuindo uma variável para cada imprecisão nas probabilidades, temos:

$$\begin{aligned}
&P(X_1' = 1|X_1 = 1, X_2 = 1, a_0) = P_1\\
&P(X_1' = 0|X_1 = 1, X_2 = 1, a_0) = P_2\\
&P(X_2' = 1|X_2 = 1, X_1 = 1, a_0) = P_3\\
&P(X_2' = 0|X_2 = 1, X_1 = 1, a_0) = P_4\\
&P(X_2' = 1|X_2 = 1, X_1 = 1, a_1) = P_5\\
&P(X_2' = 0|X_2 = 1, X_1 = 1, a_1) = P_6\\
&P(X_1' = 1|X_1 = 1, X_2 = 1, a_2) = P_7\\
&P(X_1' = 0|X_1 = 1, X_2 = 1, a_2) = P_8
\end{aligned}$$

Os conjuntos condicionais credais estão definidos pelas seguintes equações lineares:

$$\begin{aligned}
P_1 &\geq 0,9\\
P_2 &\leq 0,1\\
P_1 + P2 &= 1\\
P_3 &\geq 0,9\\
P_4 &\leq 0,1\\
P_3 + P4 &= 1\\
P_5 &\geq 0,9\\
P_6 &\leq 0,1\\
P_5 + P6 &= 1\\
P_7 &\geq 0,9\\
P_8 &\leq 0,1\\
P_7 + P8 &= 1
\end{aligned}$$

5.1 Cálculo de g_i^a e c_i^a

A *backprojection* da base constante h_0 é mostrada na Equação (10).

$$\begin{aligned}
g_0^a &= \sum_{s'} P_a(s'|s)h_0 \qquad (10)\\
&= \sum_{s'} P_a(s'|s)1\\
&= 1
\end{aligned}$$

Vamos calcular a *backprojection* das funções base h_i. Para o nosso exemplo, sabemos que cada função h_i tem escopo restrito a X_i'. Dado que cada função h_i tem escopo restrito a $C' = \{X_i'\}$, a *backprojection* tem escopo restrito a $Parents_a(X_i') : \Gamma_a(C') = \{X_{i-1}, X_i\}$. Aplicando o algoritmo BACKPROJECTION_g_i^a temos:

1. Definimos o escopo de Backproj como $\Gamma_a(\mathbf{C}') = \{X_{i-1}, X_i\}$.

2. Para cada atribuição $\mathbf{y} \in \Gamma_a(\mathbf{C}')$ calculamos:

$$\begin{aligned}
g_i^a(\mathbf{y}) &= \sum_{\mathbf{c}' \in \mathbf{C}'} \prod_{j|X_j' \in \mathbf{C}'} P_a(\mathbf{c}'[X_j']|\mathbf{y})h_i(\mathbf{c}') \qquad (11)\\
&= \sum_{x_i'} \prod_{i|x_i'} P_a(x_i'|x_{i-1}, x_i)h_i(x_i')\\
&= \sum_{x_i'} P_a(x_i'|x_{i-1}, x_i)h_i(x_i')\\
&= P_a(X_i' = 1|x_{i-1}, x_i)1 + P_a(X_i' = 0|x_{i-1}, x_i)0\\
&= P_a(X_i' = 1|x_{i-1}, x_i)
\end{aligned}$$

Temos 4 atribuições possíveis para $\Gamma_a(C') = \{x_{i-1}, x_i\}$. A seguir, mostramos todas as tabelas com os valores g_i^a. Aproveitamos também para calcular c_i^a, para $i = 0$ $c_0^a = -0,05$ (para a ação a_0, veja os dados nas Tabelas 1 e 2).

Para a ação a_1, veja as Tabelas 3 e 4.

Para a ação a_2, veja as Tabelas 5 e 6.

X_2	X_1	$g_1^{a_0}$	$c_1^{a_0}$
1	1	P_1	0,95*P_1-1
1	0	0,01	0,95*0,01-0= 0,0095
0	1	0,667	0,95*0,667-1=-0,36635
0	0	0,01	0,95*0,01-0= 0,0095

Tabela 1: Cálculo de $g_1^{a_0}$ e$c_1^{a_0}$

X_1	X_2	$g_2^{a_0}$	$c_2^{a_0}$
1	1	P_3	0,95*P_3-1
1	0	0,01	0,95*0,01-0= 0,0095
0	1	0,667	0,95*0,667-1=-0,36635
0	0	0,01	0,95*0,01-0= 0,0095

Tabela 2: Cálculo de $g_2^{a_0}$ e $c_2^{a_0}$

X_2	X_1	$g_1^{a_1}$	$c_1^{a_1}$
1	1		0,95 * 0,95 -1 = -0,0975
1	0		0,95 * 0,95 -0 = 0,9025
0	1		0,95 * 0,95 -1 = -0,0975
0	0		0,95 * 0,95 -0 = 0,9025

Tabela 3: Cálculo de $g_1^{a_1}$ e $c_1^{a_1}$

X_1	X_2	$g_2^{a_1}$	$c_2^{a_1}$
1	1	P_5	0,95*P_5-1
1	0	0,01	0,95*0,01-0= 0,0095
0	1	0,667	0,95*0,667-1=-0,36635
0	0	0,01	0,95*0,01-0= 0,0095

Tabela 4: Cálculo de $g_2^{a_1}$ e $c_2^{a_1}$

X_2	X_1	$g_1^{a_2}$	$c_1^{a_2}$
1	1	P_7	0,95*P_7-1
1	0	0,01	0,95*0,01-0= 0,0095
0	1	0,667	0,95*0,667-1=-0,36635
0	0	0,01	0,95*0,01-0= 0,0095

Tabela 5: Cálculo de $g_1^{a_2}$ e $c_1^{a_2}$

X_1	X_2	$g_2^{a_2}$	$c_2^{a_2}$
1	1		0,95 * 0,95 -1 = -0,0975
1	0		0,95 * 0,95 -0 = 0,9025
0	1		0,95 * 0,95 -1 = -0,0975
0	0		0,95 * 0,95 -0 = 0,9025

Tabela 6: Cálculo de $g_2^{a_2}$ e $c_2^{a_2}$

5.2 Resolvendo o MDPIP como um programa de dois níveis

Usando os valores g_i^a e c_i^a pré-calculados podemos explicitar as restrições do Problema (5) como segue:

```
minimize obj: 4*w0 + 2*w1 + 2*w2;

r1: -0.05*w0 + (0.95*P1 -1)* w1 + (0.95*P3-1)*w2<=-3;
r2: -0.05*w0 - 0.36635*w1 + 0.0095*w2 <= -2;
r3: -0.05*w0 + 0.0095*w1 - 0.36635*w2 <= -1;
r4: -0.05*w0 + 0.0095*w1 +  0.0095*w2 <= 0;
r5: -0.05*w0 - 0.0975*w1 + (0.95*P5-1)*w2 <= -3;
r6: -0.05*w0 - 0.0975*w1 + 0.0095*w2 <= -2;
r7: -0.05*w0 + 0.9025*w1 - 0.36635*w2 <= -1;
r8: -0.05*w0 + 0.9025*w1 + 0.0095*w2 <= 0;
r9: -0.05*w0 + (0.95*P7-1)*w1- 0.0975*w2 <= -3;
r10: -0.05*w0 - 0.36635*w1 + 0.9025*w2 <= -2;
r11: -0.05*w0 + 0.0095*w1 - 0.0975*w2 <= -1;
r12: -0.05*w0 + 0.0095*w1 +  0.9025*w2 <= 0;

      P1,P3 argmin w0+P1*w1+P3*w2
      r13: 0.9 - P1 <=0;
      r14: P1 -1  <= 0;
      r15: 0.9 - P3 <=0;
      r16: P3 -1  <= 0;

      P5 argmin w0+0.95*w1+P5*w2
      r17: 0.9 - P5 <=0;
      r18: P5 -1  <= 0;

      P7 argmin w0+P7*w1+0.95*w2
      r19: 0.9 - P7 <=0;
      r20: P7 -1  <= 0;
```

Podemos agora resolver o problema em dois níveis aplicando o método de reformulação utilizando as condições KKT mostrado na Seção 5.2. O programa em dois níveis, neste caso, é equivalente ao problema não-linear descrito a seguir:

```
maximize obj: -4*w0 - 2*w1 - 2*w2;

r1: -0.05*w0 + (0.95*P1 -1)* w1 + (0.95*P3-1)*w2<=-3;
r2: -0.05*w0 - 0.36635*w1 + 0.0095*w2 <= -2;
r3: -0.05*w0 + 0.0095*w1 - 0.36635*w2 <= -1;
r4: -0.05*w0 + 0.0095*w1 +  0.0095*w2 <= 0;
r5: -0.05*w0 - 0.0975*w1 + (0.95*P5-1)*w2 <= -3;
r6: -0.05*w0 - 0.0975*w1 + 0.0095*w2 <= -2;
r7: -0.05*w0 + 0.9025*w1 - 0.36635*w2 <= -1;
r8: -0.05*w0 + 0.9025*w1 + 0.0095*w2 <= 0;
r9: -0.05*w0 + (0.95*P7-1)*w1- 0.0975*w2 <= -3;
r10: -0.05*w0 - 0.36635*w1 + 0.9025*w2 <= -2;
r11: -0.05*w0 + 0.0095*w1 - 0.0975*w2 <= -1;
r12: -0.05*w0 + 0.0095*w1 +  0.9025*w2 <= 0;
r13: 0.9 - P1 <=0;
r14: P1 -1  <= 0;
r15: 0.9 - P3 <=0;
r16: P3 -1  <= 0;
r17: 0.9 - P5 <=0;
r18: P5 -1  <= 0;
r19: 0.9 - P7 <=0;
r20: P7 -1  <= 0;
r21: L1>=0;
r22: L2>=0;
r23: L3>=0;
r24: L4>=0;
r25: L5>=0;
r26: L6>=0;
r27: L7>=0;
r28: L8>=0;
r29: L1*(0.9 - P1) =0;
r30: L2*(P1 -1) = 0;
r31: L3*(0.9 - P3) =0;
r32: L4*(P3 -1) = 0;
r33: L5*(0.9 - P5) =0;
r34: L6*(P5 -1) = 0;
r35: L7*(0.9 - P7) =0;
r36: L8*(P7 -1) = 0;
r37: w1 - L1 + L2 = 0;
r38: w2 - L3 + L4 = 0;
r39: w2 - L5 + L6 = 0;
r40: w1 - L7 + L8 = 0;
```

As restrições 37, 38, 39 e 40 são as derivadas parciais referentes a P_1, P_3, P_5 e P_7. Note que o problema reformulado tem 8 variáveis a mais e tem $|S| * |A| + 3 * m_2 + VarProb$=4*3+3*8+4=40 restrições. Usando o *solver* não-linear Minos (Murtagh

and Saunders, 1998), obtemos o seguinte resultado:

```
Valor da função objetivo 211.460492
w0 50.91467320213732
w1 2.3445074109184922
w2 1.5563921884729717
```

Finalmente, dados os valores w_i e o conjunto de funções base, podemos construir a função valor e extrair a política ótima aproximada.

6 Conclusão

O problema de tomada de decisão sequencial, em que as probabilidades de transição não são completamente conhecidas, pode ser modelado por um Processo Markoviano de Decisão com Probabilidades Imprecisas (MDPIP). Por muito tempo, não foram propostas soluções eficientes para esse problema, isto é, soluções capazes de resolver problemas grandes modelados como MDPIPs. Em trabalhos anteriores foram propostas soluções fatoradas aproximadas baseadas em: (i) programação dinâmica (Delgado, Sanner, de Barros and Cozman, 2009) e (ii) programação multilinear (Delgado, de Barros, Cozman and Shirota, 2009). Os resultados mostraram que a solução baseada em programação matemática (multilinear) pode, em certas condições, ser mais eficiente em tempo que as soluções baseadas em programação dinâmica, enquanto essas podem apresentar um menor erro na aproximação da função valor.

Nesse trabalho, propomos uma nova solução baseada em programação matemática: a solução baseada em uma formulação de MDPIPs como programação em dois níveis. Primeiro, definimos um MDPIP geral como um programa de dois níveis. Em seguida, mostramos como transformá-lo num programa fatorado e aproximado por funções base. Finalmente, descrevemos um algoritmo e resolvemos uma instância de problema no domínio de exemplo do Administrador de Sistemas. Em trabalhos futuros pretendemos comparar o desempenho da solução aplicando a transformação KKT com a solução multilinear proposta em trabalhos anteriores (Delgado, de Barros, Cozman and Shirota, 2009).

Referências

Andreani, R., Castro, S. L. C., Chela, J. L., Friedlander, A. and Santos, S. A. (2007). An inexact-restoration method for nonlinear bilevel programming problems, *Computational Optimization and Applications* .

Bard, J. F. (1988). Convex two-level optimization, *Math. Program.* **40**(1): 15–27.

Bard, J. F. and Falk, J. E. (1982). An explicit solution to the multi-level programming problem, *Computers & Operations Research* **9**: 77–100.

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

Bialas, W. and Karwan, M. (1978). Multilevel linear programming, *Technical Report 78-1*.

Boutilier, C., Dean, T. and Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage, *Journal of Artificial Intelligence Research* **11**: 1–94.

Colson, B., Marcotte, P. and Savard, G. (2005). A trust-region method for nonlinear bilevel programming: Algorithm and computational experience, *Comput. Optim. Appl.* **30**(3): 211–227.

Cozman, F. G. (2000). Credal networks, *Artificial Intelligence* **120**: 199–233.

Delgado, K. V., de Barros, L. N., Cozman, F. G. and Shirota, R. (2009). Representing and solving factored Markov decision processes with imprecise probabilities, *6th International Symposium on Imprecise Probability: Theories and Applications (ISIPTA)*, SIPTA, Durham, United Kingdom, pp. 169–178.

Delgado, K. V., Sanner, S., de Barros, L. N. and Cozman, F. G. (2009). Efficient solutions to factored MDPs with imprecise transition probabilities, *19th International Conference on Automated Planning and Scheduling (ICAPS)*, Thessaloniki, Greece, pp. 98–105.

Guestrin, C. E. (2003). *Planning under uncertainty in complex structured environments*, PhD thesis, Stanford University. Adviser-Daphne Koller.

Koller, D. and Parr, R. (1999). Computing factored value functions for policies in structured MDPs, *IJCAI*, pp. 1332–1339.

Murtagh, B. A. and Saunders, M. A. (1998). MINOS 5.5 user's guide, *Technical Report SOL 83-20R*, Systems Optimization Laboratory, Department of Operations Research, Stanford University, California.

Patruscu, R.-E. (2004). *Linear Approximations for Factored Markov Decision Processes*, PhD thesis, University of Waterloo.

Puterman, M. L. (1994). *Markov Decision Processes—Discrete Stochastic Dynamic Programming*, John Wiley & Sons, Inc.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*, Pearson Education.

Satia, J. K. and Lave Jr, R. (1973). Markovian decision processes with uncertain transition probabilities, *Operations Research* **21**(3): 728–740.

Savard, G. and Gauvin, J. (1994). The steepest descent direction for the nonlinear bilevel programming problem, *Operations Research Letters* **15**(5): 265–272.

Shirota, R., Cozman, F. G., Trevizan, F. W., de Campos, C. P. and de Barros, L. N. (2007). Multilinear and integer programming for markov decision processes with imprecise probabilities, *5th International Symposium on Imprecise Probability: Theories and Applications*, Prague,Czech Republic, pp. 395–404.

White III, C. C. and Eldeib, H. K. (1994). Markov decision processes with imprecise transition probabilities, *Operations Research* **42**(4): 739–749.