Projeto de Controlador para Obtenção de Caminho Mínimo em Grafo Estocástico

Caio A. C. Costa Gabriel T. M. Pereira Samuel E. Vendramini

16 de julho de 2021

Resumo

Projeto de algoritmo controlador para grafos estocásticos, baseado no algoritmo de Dijkstra. Particularizou-se para uma planta-exemplo, que satisfaz condições encontradas em contextos reais, como o trânsito de uma cidade. Verificou-se a sua eficiência comparando-o com um controlador teórico presciente, por meio de simulações sucessivas.

Palavras-chaves: Controle Estocástico, Controle Ótimo, Caminho Mínimo, Dijkstra

1 Introdução

Para entendimento deste projeto, é necessário conhecimento dos paradigmas de informação em Controle Estocástico, do Algoritmo de Dijkstra e um resultado de Estatística, a ser demonstrado.

1.1 Controle Estocástico e Paradigmas de Informação

Controle é o processo de tomada de decisão em várias etapas em um dado sistema dinâmico, chamado de **planta**. O objetivo de um projeto de Controle é encontrar um **controlador**, isto é, uma função de transferência (ou mais geralmente, uma política) que retorne uma ação dadas as entradas e os estado atual do sistema.

O Controle (Ótimo) Estocástico estende o Controle Clássico pela introdução de variáveis aleatórias na planta ou na perturbação dos sensores; neste projeto, a perturbação está na planta. Em um projeto de Controle Estocástico, existem pelo menos três paradigmas de informação (LALL, 2014), que levam a diferentes controladores:

1. **Malha fechada (informação atual)**: tem-se conhecimento apenas do passado, sendo a decisão tomada levando em conta a última realização das variáveis aleatórias, mas apenas a distribuição das futuras realizações.
2. **Malha aberta (informação nula)**: a política é decidida antes de qualquer realização das variáveis aleatórias, com conhecimento apenas de suas distribuições de probabilidade.
3. **Presciente (informação total)**: em um dado momento, é conhece-se as realizações passadas, e é possível prever todas as realizações futuras da variáveis aleatórias.

Embora todos os paradigmas possuam aplicações, adotaremos o de malha fechada, por ser mais realista. Contudo, faremos uma comparação de performance entre o presciente e o de malha fechada, visando a determinar qualitativamente a eficiência do controlador a ser projetado.

1.2 Algoritmo de Dijkstra para Caminho Mínimo em Grafos

É dado o problema de, em um grafo com pesos direcionado ou não, achar o menor caminho até todos os vértices a partir de uma raiz S. Em 1959, Edsger W. Dijkstra propôs um algoritmo com complexidade estado da arte para resolver a variante desse problema onde as arestas não tem peso negativo (DIJKSTRA, 1959). Usaremos desse algoritmo para o projeto do nosso controlador nesse relatório.

O algoritmo consiste em uma busca em largura adaptada para o contexto de arestas com pesos, uma vez que a busca em largura tradicional é capaz de achar menores caminhos no caso degenerado o qual as arestas tem todas o mesmo peso. A ideia de Dijkstra era de que guardássemos também uma fila de próximos elementos a serem visitados pelo algoritmo, mas que nela os elementos (vértices) fossem ordenados de acordo com a distância atual deles à raiz, isso é, a menor distância possível de chegar a um vértice V usando apenas vértices já processados no algoritmo. Assim, a cada passo, iríamos processar o elemento de menor distância possível para a raiz S que ainda não foi processado.

A observação que garante a otimalidade do algoritmo é a de que sempre que processamos um vértice novo nós achamos a menor distância dele para a raiz. Isso é visto pelo fato de que, dado que pegamos a cada momento o vértice V de menor distância, se nesse processamento o menor caminho da raiz S para V não for atingido então o menor caminho de V pra S teria que passar por um outro vértice U ainda não processado, o que significa que deve existir um caminho de S a U menor do que um caminho de S a V. Isso é um absurdo pois, se pegarmos o primeiro vértice do caminho de S a U que ainda não foi processado, ele teria que ter sido adicionado à lista pelo seu vizinho processado e teria de ser processado, portanto, antes de V, dada a maneira que guardamos a fila.

Dessa forma, a cada passo do algoritmo processamos um novo vértice cuja menor distância dele ao vértice S é encontrado. No processamento dele, fazemos todos os ajustes para achar a menor distância de S a todos os outros vértices ainda não processados. Como as arestas não tem peso negativo, sabemos que não existe um caminho de S a S menor do que 0 e podemos, portanto, começar com o vértice S e manter durante todo o algoritmo a propriedade descrita. Garantindo, assim, processar todos os vértices do grafo e achar para cada um deles o caminho mínimo até S. Analisando brevemente a complexidade desse algoritmo, é possível ver que devemos ordenar apenas vértices e, usando uma estrutura de dados que faça a inserção ou retirada dos vértices em $\log V$, o resto da complexidade é a mesma de uma busca em largura, chegando em $\mathcal{O}((V+E)\log V)$.

1.3 Outros

Assume-se que o leitor tem familiaridade com números complexos e conceitos básicos de estatística. Além disso, demonstra-se (DEVORE, 2011) o

Lema 1. *Sejam X, Y variáveis aleatórias independentes. Então, $\mathbb{V}[X+Y] = \mathbb{V}[X] + \mathbb{V}[Y]$.*

Demonstração. Para qualquer variável aleatória W, vale

$$\mathbb{V}[W] := \mathbb{E}[(W - \mathbb{E}[W])^2] = \mathbb{E}[W^2] - \mathbb{E}[W]^2 \implies \mathbb{E}[W^2] = \mathbb{V}[W] + \mathbb{E}[W]^2 \tag{1}$$

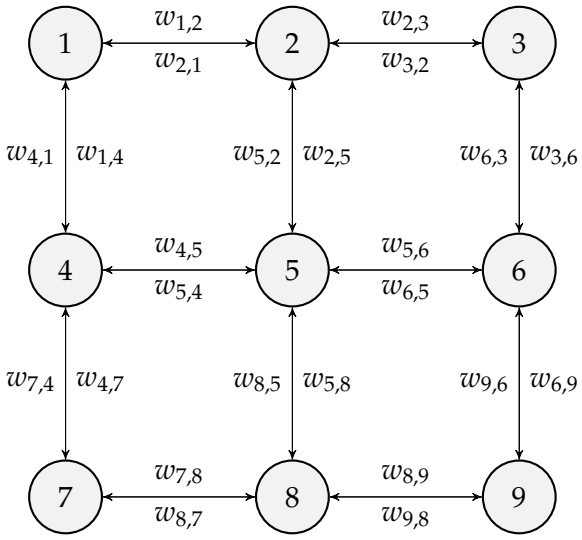

Figura 1 – planta de exemplo 3×3

Logo,

$$\begin{aligned}
\mathbb{E}[(X+Y)^2] &= \mathbb{E}[X^2 + 2XY + Y^2] && (2)\\
&= \mathbb{E}[X^2] + \mathbb{E}[Y^2] + 2\mathbb{E}[XY] && (3)\\
&= (\mathbb{V}[X] + \mathbb{E}[X]^2) + (\mathbb{V}[Y] + \mathbb{E}[Y]^2) + 2\mathbb{E}[X]\mathbb{E}[Y] && (4)\\
&= \mathbb{V}[X] + \mathbb{V}[Y] + (\mathbb{E}[X] + \mathbb{E}[Y])^2 && (5)\\
&= \mathbb{V}[X] + \mathbb{V}[Y] + \mathbb{E}[X+Y]^2 && (6)
\end{aligned}$$

De (2) para (3), usamos a linearidade da esperança. De (3) para (4), reescrevemos as esperanças de quadrados usando (1) e a esperança do produto usando a independência de X e Y. De (4) para (5), agrupamos o quadrado perfeito e de (5) para (6) usamos novamente a linearidade da esperança. Tomando $W = X + Y$ em (1) e aplicando (6), concluímos que $\mathbb{V}[X+Y] = \mathbb{V}[X] + \mathbb{V}[Y]$. □

2 Metodologia

2.1 Planta-Exemplo

A planta utilizada para fins de demonstração consiste de um grafo em forma de grade bidimensional $n \times n$, em que cada nó na posição (i, j) está conectado por uma aresta direcionada com seus quatro vizinhos nas posições $(i \pm 1, j)$ e $(i, j \pm 1)$, com exceção dos nós na borda do grafo. A Figura 1 mostra o caso $n = 3$. Os pesos w_i, j são aleatórios. O agente demora um tempo igual ao peso da aresta para atravessá-la. A seguir, veremos como esses pesos são inicializados e atualizados.

Para inicializar, são amostrados n números complexos $z_1, \ldots z_n \in A := \{z \in \mathbb{C} \mid 0 \leq \mathrm{Re}(z), \mathrm{Im}(z) < n\}$ uniformemente, com os quais se constrói a função complexa polinomial $f(z) = K(z - z_1) \ldots (z - z_n)$. Seja $g(z) = |f(z - 0.5 - 0.5i)|^2$. O deslocamento de $-0.5 - 0.5i$ é necessário para que os pontos em que se calcula f estejam centralizados em A. Além disso, como a função $|z|^2$ é diferenciável e $f(z)$ é um polinômio, $g(z)$ também é diferenciável. Observe que $g(z)$ é uma função de $\mathbb{C}$ em $\mathbb{R}_+$ cujo valor médio nos pontos do grafo é 1. O ganho K é

normalizado para que a Equação 7 seja válida.

$$\sum_{u=1}^{n}\sum_{v=1}^{n} g(u+iv) = n^2. \tag{7}$$

Finalmente, para a aresta que liga os vértices (u_1, v_1) e (u_2, v_2), atribui-se peso

$$w = 1 + \frac{g(u_1+iv_1) + g(u_2+iv_2)}{2} \tag{8}$$

Passado um tempo Δt, são amostrados números complexos $\Delta z_1, \ldots, \Delta z_n$ de forma que $\mathrm{Re}(z_k)$ e $\mathrm{Im}(z_k)$ tenham distribuição normal com variância T, sendo essas $2n$ variáveis aleatórias independentes e identicamente distribuídas. Então, atualizam-se as raízes de f conforme

$$z_k(t+\Delta t) = \sigma(z_k(t) + \Delta z_k), \tag{9}$$

em que $\sigma(z) = \max(0, \min(n, \mathrm{Re}(z))) + i \cdot \max(0, \min(n, \mathrm{Im}(z)))$ satura z em A. O valor de K também é atualizado mantendo a Equação 7.

Desconsiderando a saturação σ, como a distribuição normal é linear, segue que não é necessário computar os valores de $z_k(t)$ com Δts infinitesimais, bastando atualizar as raízes de f nos tempos de interesse. Além disso, tendo as normais média 0, o valor esperado de cada Δz_k, $\mathbb{E}[\Delta z_k] = 0$, de modo que

$$\mathbb{E}[f(z, t+\Delta t)] = f(z, t). \tag{10}$$

2.2 Controlador

Nós queremos direcionar um objeto saindo do ponto $(1, 1)$ e indo até o (n, n) passando por um caminho cuja soma dos pesos das arestas é a menor possível. Para isso, nós temos em cada instante a informação sobre o valor dos pesos de todas as arestas do grafo e em qual vértice o objeto se encontra. Dessa forma, o nosso controlador deve ser capaz de indicar a melhor direção para o objeto tomar de forma a minimizar a esperança da soma dos valores do caminho até (n, n) partindo da posição atual dele. O controlador poderá direcionar o objeto somente quando ele estiver em um vértice, sendo obrigatório para um objeto percorrer toda uma aresta a partir do momento que o controlador escolhe ela.

2.2.1 Controlador em malha fechada

Uma vez que temos o *feedback*, nosso controlador pode ser definido como de malha fechada. A maneira que usamos a informação de posição e do valor atual das arestas é valendo também da Equação 10. Sabendo que, no próximo instante, a esperança da soma dos pesos das arestas no caminho de qualquer vértice até (n, n) é constante. Podemos calcular atualmente qual é o menor caminho mínimo do vértice final até todos os vértices e decidir ir para o vértice cuja soma do peso da aresta para chegar a ele mais a soma da distancia esperada dele até o destino final é o mínimo. Dessa forma, queremos que, se estamos em um vértice u, o próximo vértice a irmos, $C(u)$, seja tal que

$$C(u) = \mathrm{argmin}_v(w_{uv}^t + \mathbb{E}[D_v]) = \mathrm{argmin}_v(w_{uv}^t + D_v^t), \tag{11}$$

onde D_v^t é a menor soma dos pesos possíveis no instante atual para chegar até (n, n) saindo do vértice v, w_{uv}^t é o valor atual da aresta saindo de u e chegando em v e $E[D_v]$ é a esperança

da menor soma dos pesos de um caminho saindo de v até (n, n) no futuro que, de acordo com Equação 10, é igual a D_v^t.

Desse modo, temos uma equação para o controlador em função somente das informações que temos no atual instante: o vértice atual no qual o objeto se encontra, u, além de w_{ij}^t para todos os outros vértices i, j do grafo. Para isso, devemos ainda calcular de fato o valor de D_v^t em tempo hábil. Isso pode ser calculado com os valores das arestas w_{ij}^t no instante atual usando o algoritmo de Dijkstra. O algoritmo de Dijkstra serve para calcular a menor distância de um vértice qualquer a todos os outros vértices do grafo em tempo $\mathcal{O}(|V| \log |E|)$, onde $|V|$ é o numero de vértices e $|E|$ o número de arestas do grafo. Podemos rodar então esse algoritmo a partir do vértice final, (n, n), invertendo, para isso, o sentido de todas as arestas do nosso grafo e, assim, obter o valor de D_v^t para todos vértices v do grafo. Dessa forma, conseguimos garantir que nosso controlador nos fornece o caminho de menor valor esperado para a soma das arestas tendo somente como base o valor das arestas no instante atual. Ainda, outros controladores podem ser propostos, para efeito de comparação com o usado por nós, para avaliar a eficiência do nosso controlador de malha fechada.

2.2.2 Controlador de Malha aberta

Nesse caso, nós temos ciência apenas da posição do nosso objeto em um determinado instante, tendo somente a informação do valor das arestas no início, imediatamente antes do objeto sair de $(1, 1)$. Dessa forma, o melhor que podemos fazer é seguir o menor caminho indicado pelo algoritmo de Dijkstra no instante inicial. Ainda se valendo da Equação 10, mas sem poder atualizar nossas informações com o decorrer do tempo.

2.2.3 Controlador Presciente

Nesse outro caso, nós temos ciência dos valores das arestas também no futuro, podendo decidir qual caminho tomamos de maneira totalmente ótima. Dado que temos todos os valores necessários para tornar o problema determinístico. Nesse caso, um algoritmo de força bruta é capaz de fornecer o caminho exato que consiste na menor soma possível das arestas em um caminho saindo de $(1, 1)$ e chegando em (n, n).

A ideia do uso desses dois últimos controladores é que, o Presciente não é possível de ser obtido em uma situação real e serviria apenas como um benchmark para o caminho encontrado pelo controlador de malha fechada. E, o controlador de malha aberta, por outro lado, é uma situação com menos informação do que é possível de ser obtida numa situação real e pode ser usada para definir se nosso controlador traz uma evolução ou não quanto ao caminho feito pelo objeto.

3 Resultados e Discussões

A partir das implementações dos controladores de malha fechada e presciente, executou-se o código 1000 vezes gerando números aleatórios a partir da entropia do sistema Linux, por meio do arquivo `/dev/random`. No `Matlab` ®, construiu-se um histograma em forma de densidade de probabilidade para os tempos/custos totais obtidos. Conforme a Figura 2, vemos que o controlador presciente performa, em média, melhor que o controlador de malha fechada, mas a diferença entre os dois não é alta – apontando para a eficiência do controlador projetado.

Como a resolução do controlador presciente é um problema combinatório, e havia a necessidade de executar milhares de vezes para viabilizar o método de Monte Carlo, não foi possível utilizar um grafo de tamanho muito grande, limitando a planta exemplo a $n = 3$.

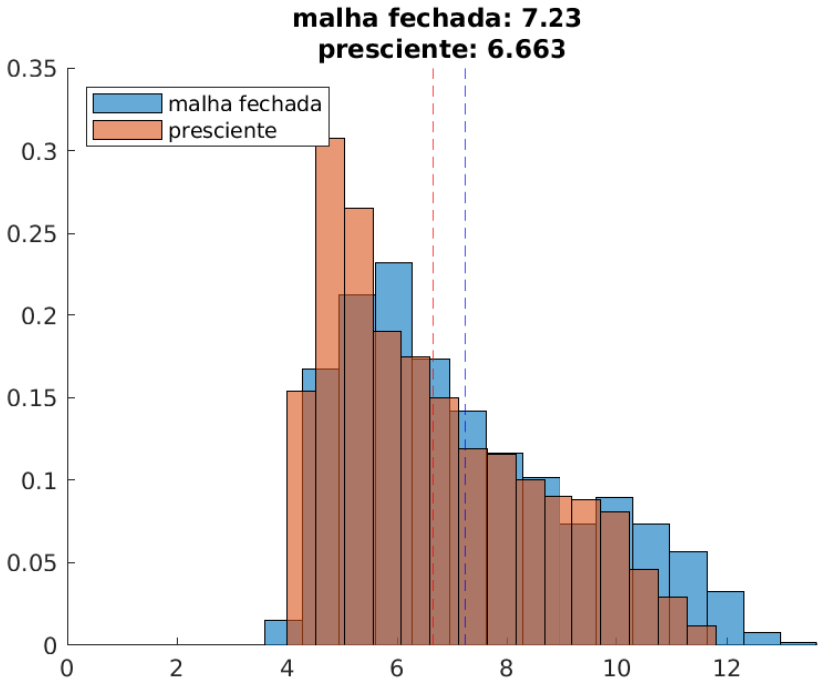

Figura 2 – Histograma de tempo/custo total para simulações de Monte Carlo comparando as performances do controlador de malha fechada e do controlador presciente para a planta-exemplo.

Conclusão

É proposta, portanto, uma forma de caminhar em um grafo estocástico. Foi visto que as ideias de controle podem ser aplicadas também nessa situação, usando de um controle de malha fechada para ter um melhor desempenho, no qual o controlador indica um melhor caminho para seguir e atualiza suas decisões no futuro em virtude das modificações percebidas no grafo.

Assim, evitamos a malha aberta, que não é robusta e, dependendo do prosseguimento do grafo, poderia não convergir para onde queríamos. De acordo com o observado em resultados, é visto ainda que o controle proposto é eficiente e comparável ao caso em que temos um grafo determinístico.

Referências

DEVORE, J. L. *Probability and Statistics for Engineering and the Sciences*. 8th. ed. [S.l.]: Brooks/Cole, 2011. ISBN-13: 978-0-538-73352-6. Citado na página 2.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, v. 1, p. 269–271, 1959. Citado na página 2.

LALL, S. 2014. Disponível em: <https://stanford.edu/class/ee365/lectures/intro.pdf>. Citado na página 1.