Algoritmos em matrizes monótonas e Monge convexas

Victor Sena Molero **Orientadora: Cristina Gomes Fernandes**
Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo

Matrizes Monge

Este trabalho discute algoritmos que aproveitam propriedades de matrizes Monge [2]. Para introduzir tais matrizes, olharemos para um problema interessante em geometria computacional.

Problema: *Dado um polígono convexo p com os vértices indexados de 1 até n em sentido horário, encontrar, para cada vértice p_i com $i \in [n]$, um vértice p_j com $j \in [n]$ que maximiza o quadrado da distância euclidiana $d(p_i, p_j)$ entre os dois pontos.*

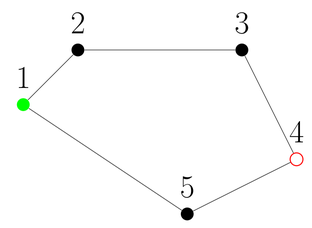

Figura 1: O polígono ilustrado é convexo. O ponto em verde é o mais distante do ponto em vermelho.

É fácil resolver o problema proposto em tempo $\mathcal{O}(n^2)$ calculando a distância de cada vértice do polígono para cada um dos demais. Existe alguma maneira mais rápida de resolver este problema?

Uma matriz A com n linhas e m colunas é **Monge côncava** se, para cada $i, i' \in [n]$ e $j, j' \in [m]$ onde $i < i'$ e $j < j'$, vale que

$$A[i][j] + A[i'][j'] \geq A[i][j'] + A[i'][j].$$

Vamos construir uma matriz Monge côncava A com n linhas e $2n-1$ colunas a partir do polígono p e um valor C suficientemente grande de forma que, para cada linha i de A, uma coluna j que atinge o máximo nessa linha é tal que p_j é um ponto mais distante de p_i. Se $i < j < i+n$, vale que $A[i][j] = d(p_i, p_j)$. Para $j \leq i$, temos que $A[i][j] = -C(i-j)$ e, finalmente, para $i + n \leq j$, vale que $A[i][j] = -C(j-n-i)$.

0	2	17	26	13	0	-13	-26	-39
-13	0	9	20	13	2	0	-13	-26
-26	-13	0	5	10	17	9	0	-13
-39	-26	-13	0	5	26	20	5	0
-52	-39	-26	-13	0	13	13	10	5

Figura 2: Matriz Monge côncava relativa ao polígono da Figura 1. O valor C escolhido foi 13. Os números da forma $d(a, b)$ para algum par de vértices a, b do polígono estão marcados em preto. As entradas restantes estão marcadas em vermelho.

Como encontrar um valor máximo em cada linha de uma matriz Monge sem analisar todas as suas entradas? Mais especificamente, sem gastar tempo quadrático. Para isso, vamos explorar as propriedades das matrizes Monge. Toda matriz Monge côncava é **monótona crescente nos máximos das linhas**: os maiores índices de máximos das suas linhas estão em ordem decrescente.

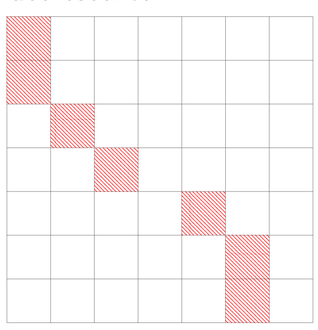

Figura 3: Matriz monótona crescente nos máximos das linhas. Os máximos das linhas estão marcados em vermelho.

Uma matriz Monge côncava é **totalmente monótona côncava nas linhas**: toda submatriz dela é monótona crescente nos máximos das linhas. Estas propriedades são essenciais para os algoritmos discutidos no trabalho.

Existem também as matrizes Monge convexas. As matrizes Monge possuem também outros tipos de monotonicidade e total monotonicidade. Uma matriz Monge côncava é, por exemplo, totalmente monótona côncava nas colunas e monótona decrescente nos mínimos das linhas. Esta diversidade faz com que os algoritmos discutidos aqui resolvam vários problemas distintos.

Referências

[1] Aggarwal, Alok and Klawe, Maria M. and Moran, Shlomo and Shor, Peter and Wilber, Robert, "**Geometric applications of a matrix-searching algorithm**," in *Algorithmica. Springer*, 1987, 2 (1-4), pp. 195–208.

[2] Burkard, Rainer E. and Klinz, Bettina and Rudolf, Rüdiger, "**Perspectives of Monge properties in optimization**," in *Discrete Applied Mathematics. Elsevier*, 1996, 70 (2), pp. 95–161.

[3] Galil, Zvi and Park, Kunsoo, "**Dynamic programming with convexity, concavity and sparsity**," in *Theoretical Computer Science. Elsevier*, 1992, 92 (1), pp. 49–76.

[4] Knuth, Donald E., "**Optimum binary search trees**," in *Acta Informatica. Springer*, 1971, 1 (1), pp. 14–25.

[5] Yao, F. Frances, "**Efficient dynamic programming using quadrangle inequalities**," in *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing. ACM*, 1980, 3 (4), pp. 532–540.

Divisão e conquista

O primeiro método não trivial para encontrar os máximos das linhas de matrizes Monge é chamado de método da divisão e conquista. A ideia é simples e utiliza apenas a monotonicidade da matriz Monge.

Tome uma matriz A monótona crescente nos máximos das linhas com n linhas e m colunas. Tome como ℓ uma linha do meio da matriz (teto ou chão de $\frac{1+n}{2}$). Se r é o maior índice de máximo desta linha, sabemos que os índices de máximos das outras linhas estão nas submatrizes $A[1\,..\,\ell-1][r\,..\,n]$ e $A[\ell+1\,..\,n][1\,..\,r]$, para as quais resolvemos o problema recursivamente.

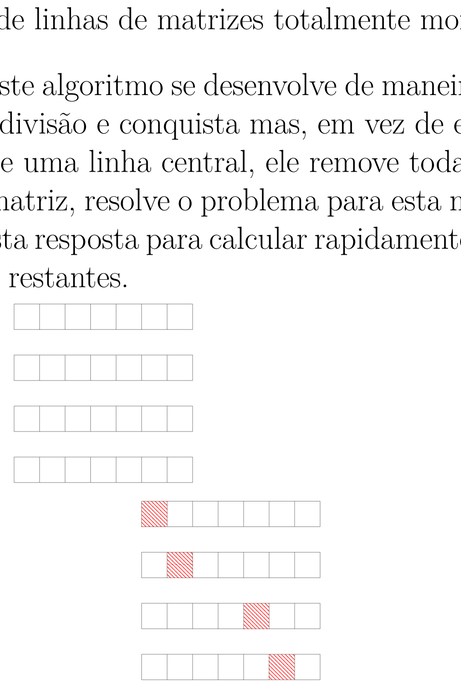

Figura 4: Progressão da divisão e conquista. A cada passo, a linha ℓ está marcada em verde e a célula de máximo desta linha está circulada em vermelho.

Escolhendo ℓ da forma como escolhemos, este algoritmo completa seu trabalho em tempo $\mathcal{O}((n+m)\lg(n))$.

SMAWK

O algoritmo SMAWK [1] é um método para encontrar os máximos de linhas de matrizes totalmente monótonas.

A ideia deste algoritmo se desenvolve de maneira parecida com a da divisão e conquista mas, em vez de encontrar o máximo de uma linha central, ele remove todas as linhas pares da matriz, resolve o problema para esta nova matriz e utiliza esta resposta para calcular rapidamente os valores das linhas restantes.

Figura 5: Progressão do SMAWK. Primeiro, as linhas pares são removidas. Depois os máximos das linhas restantes (em vermelho) são encontrados recursivamente. Finalmente, para encontrar os índices das linhas originais, é necessário visitar apenas algumas posições (circuladas em verde).

Desta forma o algoritmo tomaria tempo $\mathcal{O}((n+m)\lg(n))$, assim como o método da divisão e conquista. Para melhorar esta complexidade, antes de resolver o problema recursivamente, o algoritmo SMAWK remove algumas colunas da matriz formada, tornando-a quadrada.

Graças à total monotonicidade, é possível remover, em tempo $\mathcal{O}(n+m)$, colunas suficientes de forma que a matriz resultante ainda seja totalmente monótona e possua os máximos nas mesmas posições da original. Com isso, o algoritmo todo toma tempo $\mathcal{O}(n+m)$.

Programação Dinâmica

As matrizes Monge são muito úteis para resolver problemas de programação dinâmica [3].

O problema "Internet Trouble" da Final Brasileira da Maratona de Programação 2016 pode ser reduzido para uma recorrência E tal que $E[1][j] = A[1][j]$ e $E[\ell][j] = \min\{E[\ell-1][i] + A[i][j] \mid i \in [j]\}$ para todo par de índices $\ell \in [2\,..\,k]$ e $j \in [n]$ onde A é uma matriz e k e n são inteiros. Esta recorrência pode ser resolvida de maneira fácil com programação dinâmica em tempo $\mathcal{O}(kn^2)$. Porém, é possível traduzir esta recorrência para uma matriz monótona crescente nos mínimos das linhas de forma que encontrar os mínimos desta matriz é equivalente a encontrar a solução da recorrência E. Esta redução resolve o problema em tempo $\mathcal{O}(kn\lg(n))$.

Um outro formato comum para programação dinâmica que é relacionado a matrizes Monge é o dado por uma recorrência E tal que $E[i] = \min\{E[j] + C[i][j] \mid j \in [i+1\,..\,n]\}$ para todo $i \in [n-1]$ e $E[n] = 0$ onde n é um natural e C é uma matriz Monge. O trabalho apresenta uma estrutura de dados chamada de envelope que faz com que seja possível resolver esta recorrência em tempo $\mathcal{O}(n\lg(n))$ em vez de $\mathcal{O}(n^2)$, que é o tempo da solução trivial por programação dinâmica.

Otimização de Knuth-Yao

A otimização de Knuth-Yao é uma aplicação das propriedades das matrizes Monge fortemente relacionada a programação dinâmica. Ela se baseia na solução de Knuth [4] para o problema da árvore de busca binária ótima. Yao [5] percebeu que tal solução poderia ser usada para qualquer recorrência tal que $A[i][j] = C[i][j]$ quando $i = j$ e, quando $1 \leq i < j \leq n$, $A[i][j] = C[i][j] + \min\{A[i][k-1] + A[k][j] \mid k \in [i+1\,..\,j]\}$ onde C é uma matriz Monge convexa e **monótona nos intervalos**: sempre que i, i' são índices de linhas e j, j' índices de colunas de C tais que $[i'\,..\,j'] \subseteq [i\,..\,j] \subseteq [1\,..\,n]$, vale que $C[i'][j'] \leq C[i][j]$.

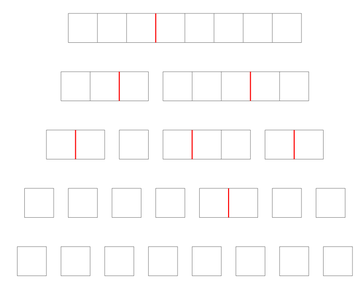

Figura 6: Interpretação da recorrência A. Desejamos dividir um vetor v em um dado ponto sucessivamente até que todas as partes tenham tamanho unitário. O objetivo é encontrar uma ordem de divisões que minimize o custo sabendo que dividir um vetor $v[i\,..\,j]$ em qualquer ponto custa $C[i][j]$.

Tome um subvetor $v[i\,..\,j]$ de v. Se um ponto ótimo para dividir $v[i\,..\,j-1]$ é ℓ e um ponto ótimo para dividir $v[i+1\,..\,j]$ é r, segundo Yao, um ponto ótimo p para dividir o vetor $v[i\,..\,j]$ deve respeitar $\ell \leq p \leq r$.

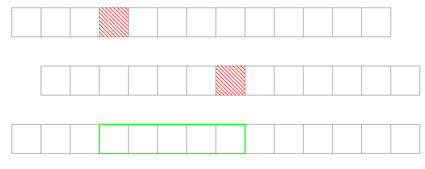

Figura 7: Relação entre os pontos de corte ótimos. Os dois primeiros vetores são subvetores do último. Neles está marcado, em verde, o ponto ótimo de divisão. No último, estão marcados, em vermelho, os candidatos a pontos ótimos de divisão.

Enquanto a recorrência apresentada tem uma solução trivial $\mathcal{O}(n^3)$, a otimização de Knuth-Yao se utiliza desta propriedade para resolver a mesma recorrência em tempo $\mathcal{O}(n^2)$.

Mais informações

http://www.linux.ime.usp.br/~victorsenam/mac0499