Escalonamento Multidimensional: uma Abordagem Discreta

Ana Camila R. Alonso* **Silvia M. S. Carvalho**
Departamento de Matemática Aplicada - IMECC-UNICAMP
13083-859, Campinas, SP
E-mail:acamila@ime.unicamp.br E-mail: silvia@densis.fee.unicamp.br

Carlile Lavor† **Aurelio R. L. Oliveira‡**
Departamento de Matemática Aplicada - IMECC-UNICAMP
13083-859, Campinas, SP
E-mail: clavor@ime.unicamp.br E-mail: aurelio@ime.unicamp.br

Resumo

Neste trabalho, propomos uma abordagem discreta para o problema de escalonamento multidimensional usando um algoritmo *Branch-and-Prune*. Esta abordagem requer um tempo de processamento que aumenta polinomialmente com a quantidade de pontos envolvida no problema, em contraste com um algoritmo *Branch-and-Bound* descrito na literatura, que requer tempo exponencial.

Palavras-chave: *Algoritmo Branch-and-Prune, Escalonamento Muldimensional, Algoritmo Branch-and-Bound*

Área principal (OC - Otimização Combinatória)

Resumo

In this work, we propose a discrete approach to the multidimensional scaling problem using a Branch-and-Prune algorithm. This approach requires a polynomial processing time with the amount of given points of the problem, as opposed as the exponential time required by the Branch-and-Bound algorithm described in the literature.

Keywords: *Branch-and-Prune Algorithm, Multidimensional Scaling, Branch-and-Bound Algorithm*

Main area (CO - Combinatorial Optimization)

1 Introdução

O escalonamento multidimensional é uma técnica de análise de dissimilaridade entre dados largamente aplicada em diversas áreas. O principal objetivo da técnica é representar dissimilaridades como distâncias entre pontos em um espaço de dimensão pequena ($\mathbb{R}^2$ ou $\mathbb{R}^3$) com o objetivo de desenvolver uma melhor representação geométrica, Borg (2010).

A visualização gráfica das correlações entre os dados permite que se explore sua estrutura visual, revelando muitas vezes, regularidades que permanecem ocultas quando se estuda uma matriz numérica associada às dissimilaridades.

*bolsista de Doutorado CNPq - Processo 140239/2009-0
†Bolsista de Produtividade CNPq - Processo 304351/2010-5
‡Bolsista de Produtividade CNPq - Processo 309561/2009-4, Projeto Temático FAPESP 2007/56052-8

Nesta seção, será dada uma breve definição do problema. Um determinado conjunto de dados de tamanho n, apresenta dissimilaridade dada por uma matriz (δ_{ij}), $i, j = 1, ..., n$, onde iremos considerar tais dados como sendo pontos no $\mathbb{R}^m$.

Uma forma de se obter uma "representação" em $\mathbb{R}^3$ para estes dados consiste em determinar $x_i \in \mathbb{R}^3$, $i = 1, ..., n$, minimizando uma função *Stress* de resíduos mínimos:

$$S(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_{ij}(d(x_i, x_j) - \delta_{ij})^2, \tag{1}$$

$$d(x_i, x_j) = \left(\sum_{k=1}^{3} |x_{ik} - x_{jk}|^p \right)^{\frac{1}{p}}, \tag{2}$$

onde $x = (x_1, x_2, ..., x_n)$, $x_i = (x_{i1}, x_{i2}, x_{i3})$ e $d(x_i, x_j)$ denota a distância entre os pontos x_i e x_j. Por sua vez, $\omega_{ij} > 0$, $i, j = 1, ..., n$, denota um parâmetro (peso) fornecido *a priori*.

Para definir um critério específico, é necessário escolher uma norma, a mais utilizada é a Euclidiana ($p = 2$). No entanto, outras normas de *Minkowski* podem ser mais adequadas na análise multidimensional do que a Euclidiana, Arabie (1991). A análise dos resultados para normas diferentes pode destacar propriedades distintas dos dados considerados, de forma similar que diferentes projeções ortogonais de um objeto tridimensional resultam em imagens diferentes.

Neste trabalho, investigou-se uma abordagem alternativa para a obtenção dos pontos $x_i \in \mathbb{R}^3$. Esta abordagem consiste em um algoritmo tipo *Branch-and-Prune* que permite a resolução de instâncias bem maiores em comparação com um algoritmo *Branch-and-Bound* descrito na literatura, Žilinskas (2009).

2 Nova Formulação do Problema

Considere um conjunto ordenado de n pontos com coordenadas cartesianas dadas por $x_1, ..., x_n \in \mathbb{R}^3$. A distância Euclidiana entre os pontos $i-1$ e i é denotada por r_i para todo $i = 2, ..., n$, o ângulo $\theta_i \in [0, \pi]$ representa o ângulo formado entre os segmentos definidos pelos pontos $i-2$, $i-1$ e i, para todo $i = 3, ..., n$, e o ângulo de torção $\omega_i \in [0, 2\pi]$ representa o ângulo entre as retas normais dos planos definidos pelos pontos $i-3, i-2, i-1$ e $i-2, i-1, i$, para todo $i = 4, ..., n$.

O conhecimento de r_i, θ_i e ω_i permite fixar os três primeiros pontos de acordo com a sequência determinada. O quarto ponto é determinado pelo ângulo de torção ω_4, r_2, r_3, e θ_3, o quinto ponto, por sua vez, é determinado pelos ângulos de torção ω_4 e ω_5, bem como pelos valores de r_i e θ_i associadas. As coordenadas Cartesianas $x_i = (x_{i1}, x_{i2}, x_{i3})$, para cada ponto i, podem ser obtidas usando as seguintes relações, Phillips (1996):

$$\begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ 1 \end{bmatrix} = B_1 B_2 ... B_i \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \forall i = 1, ..., n, \tag{3}$$

onde B_1 é a matriz identidade de dimensão 4,

$$B_2 = \begin{bmatrix} -1 & 0 & 0 & -r_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

$$B_3 = \begin{bmatrix} -\cos\theta_3 & \sin\theta_3 & 0 & -r_3\cos\theta_3 \\ \sin\theta_3 & -\cos\theta_3 & 0 & -r_3\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

$$B_i = \begin{bmatrix} -\cos\theta_i & -\sin\theta_i & 0 & -r_i\cos\theta_i \\ \sin\theta_i\cos\omega_i & -\cos\theta_i\cos\omega_i & -\sin\omega_i & r_i\sin\theta_i\cos\omega_i \\ \sin\theta_i\sin\omega_i & -\cos\theta_i\sin\omega_i & -\cos\omega_i & r_i\sin\theta_i\sin\omega_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6)$$

para $i = 4, ..., n$.

Dadas as distâncias r_2, r_3 e o ângulo θ_3, é possível calcular as matrizes de torção B_2 e B_3 para determinar os três primeiros pontos:

$$x_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, x_2 = \begin{pmatrix} -r_2 \\ 0 \\ 0 \end{pmatrix} \text{ e } x_3 = \begin{pmatrix} r_3\cos\theta_3 - r_2 \\ r_3\sin\theta_3 \\ 0 \end{pmatrix}.$$

O *seno* do ângulo de torção ω_4 pode assumir apenas dois valores: $\sin\omega_4 = \pm\sqrt{1-(\cos\omega_4)^2}$, uma vez que a distância entre x_1 e x_4 também é conhecida, Liberti (2008). Consequentemente, existem duas posições possíveis x_4 e $x^{'}{}_4$ para o quarto ponto. Estas posições são fornecidas abaixo:

$$x_4 = \begin{bmatrix} -r_2 + r_3\cos\theta_3 - r_4\cos\theta_3\cos\theta_4 + r_4\sin\theta_3\sin\theta_4\cos\omega_4 \\ r_3\sin\theta_3 - r_4\sin\theta_3\cos\theta_4 - r_4\cos\theta_3\sin\theta_4\cos\omega_4 \\ -r_4\sin\theta_4\sqrt{1-(\cos\omega_4)^2} \end{bmatrix},$$

$$x_4^{'} = \begin{bmatrix} -r_2 + r_3\cos\theta_3 - r_4\cos\theta_3\cos\theta_4 + r_4\sin\theta_3\sin\theta_4\cos\omega_4 \\ r_3\sin\theta_3 - r_4\sin\theta_3\cos\theta_4 - r_4\cos\theta_3\sin\theta_4\cos\omega_4 \\ r_4\sin\theta_4\sqrt{1-(\cos\omega_4)^2} \end{bmatrix}.$$

No problema de escalonamento multidimensional, no lugar de pontos no $\mathbf{R}^3$, temos n pontos no $\mathbf{R}^m$. Entretanto, podemos definir uma ordem entre esses pontos de tal forma que, em $\mathbf{R}^3$, obtemos uma representação desses n pontos que mantém as mesmas distâncias dadas em $\mathbf{R}^m$ entre os pontos $i-1$ e i, para $i = 2, ..., n$, e também entre os pontos $i-2$ e i, para $i = 3, ..., n$, Alencar (2012). Com essa representação em $\mathbf{R}^3$, e utilizando as distâncias conhecidas entre os pontos no $\mathbf{R}^m$, aplicamos o algoritmo *Branch-and-Prune* para obter as coordenadas Cartesianas dos pontos em $\mathbf{R}^3$.

3 Algoritmo *Branch-and-Prune*

Uma breve descrição do algoritmo *Branch-and-Prune* será feita aqui, com base na descrição apresentada em Liberti (2008). Em cada passo, encontram-se duas posições possíveis, x_i e $x_i^{'}$, para o ponto i que se deseja representar em $\mathbf{R}^3$, podendo estas posições serem consideradas infactíveis, uma vez não respeitadas as distâncias em $\mathbf{R}^m$. Mais precisamente, cada uma destas possíveis posições deve satisfazer, para todos os pares de distâncias anteriores d_{ij}, a relação $|\|x_i - x_j\| - d_{ij}| \leq \epsilon$, em que ϵ>0 representa uma tolerância predeterminada. Existem quatro resultados possíveis:

1. x_i e $x_i^{'}$ são factíveis: neste caso ambas as posições são armazenadas e exploradas;
2. somente x_i é factível: somente a posição x_i é armazenada e a posição $x_i^{'}$ é descartada, ou seja, a árvore de busca é podada;
3. somente $x_i^{'}$ é factível: somente a posição $x_i^{'}$ é armazenada e a posição x_i é podada;
4. nenhuma posição é factível: podam-se ambas e a busca retrocede.

Seja T uma representação gráfica da árvore de busca. A árvore T é inicializado pelos nós de busca $1 \rightarrow 2 \rightarrow 3$, pois os três primeiros pontos sempre podem ser fixados em posições factíveis x_1, x_2, x_3. Em cada nó da árvore de busca na posição i, são armazenados:

- a posição $x_i \in \mathbb{R}^3$ do i-ésimo ponto;
- o produto cumulativo $C_i = \prod_{j=1}^{i} B_j$ das matrizes de torção;
- um apontador para o nó principal $P(i)$;
- apontadores para os subnós $L(i)$, $R(i)$ inicializado com um valor falso (Podado), se infactível.

O procedimento recursivo a partir da posição $i-1$ é apresentado no Algoritmo 1. Seja $y = (0,0,0)^T$, ϵ>0 uma dada tolerância e v um nó na posição $i-1$ da árvore de busca T.

Algorithm 1: Algoritmo *Branch-and-Prune*

Entrada: BranchAndPrune(T, v, i)
1. Se $(i \leq (n-1))$ então
   2. //Calcule as possíveis posições para o i-ésimo ponto:
   3. calcule a matriz de torção B_i; $B_i^{'}$ Eq. (4);
   4. recupere a matriz de torção cumulativa C_{i-1} a partir do nó principal $P(v)$;
   5. Calcule $C_i = C_{i-1}$, B_i, $C_i^{'} = C_{i-1}$, $B_i^{'}$ e x_i,$x_i^{'}$ de $C_i y$, $C_i^{'} y$;
   6. Seja $\lambda = 1$ e $\rho = 1$
   7. //TESTE DE PODA
   8. Se (x_i é factível) então
      9. Criar um nó z, armazenar Q_i e x_i em $z^{'}$. Seja $P(z) = v$ e $L(v) = z$;
      10. Colocar $T \leftarrow T \cup \{z\}$;
      11. BranchAndPrune$(T, z, i+1)$;
   12. Senão
      13. Colocar $L(v) =$ PODADO;
   14. Fim
   15. Se ($x_i^{'}$ é factível) então
      16. Criar um nó $z^{'}$, armazenar Q_i e x_i em $z^{'}$. Seja $P(z) = v$ e $R(v) = z^{'}$;
      17. Colocar $T \leftarrow T \cup \{z^{'}\}$;
      18. BranchAndPrune$(T, z^{'}, i+1)$;
   19. Senão
      20. Colocar $R(v) =$ PODADO;
   21. Fim
22. Senão
   23. //Posição n atingida, foi encontrada uma solução:
   24. solução armazenada em nós principais nas posições de n a 1.
   25. Repita o processo até percorrer toda a árvore.
26. Fim

4 Resultados Computacionais

Os resultados obtidos foram comparados com os resultados apresentados em Žilinskas (2009), onde a função $S(x)$, definida em (1), é minimizada com $p = 1$ usando um método *Branch-and-Bound*. A avaliação dos métodos teve como base a evolução do tempo de processamento medido em segundos (t) e o erro relativo fornecido pela função abaixo:

$$f(x) = \sqrt{\frac{S(x)}{\sum_{i=1}^{n}\sum_{j=1}^{n}\omega_{ij}\delta_{ij}^2}}, \tag{7}$$

onde fixou-se $\omega_{ij} = 1$ para $i, j = 1, ..., n$.

Como instâncias do problema, foi utilizado o conjunto de vértices de um d-simplex, que é um politopo d-dimensional, definido em Žilinskas (2009). Para este conjunto de vértices, as diferenças foram medidas por uma distância *city-block* ($p = 1$) no espaço vetorial original. A dimensão do espaço em que os vértices se encontram inicialmente é d e o objetivo consiste em representar tais vértices em $\mathbf{R}^3$ com o menor valor possível $f^* \geq 0$ para a função $f(x)$ (7). O número de vértices do politopo multidimensional é dado por $n = d + 1$.

O algoritmo foi implementado em *Matlab2010*, em um processador Intel Core 2 Duo 2.66GHz e sistema operacional MAC OSX.

n	tempo(segundos)	f^*
3	0,00	0,00
4	0,01	0,00
5	1,12	0,00
6	25,49	0,00
7	11111	0,0945

Tabela 1: Algoritmo *Branch-and-Bound* da literatura.

A Tabela 1 contém os resultados obtidos pelo método proposto em Žilinskas (2009). A coluna inicial apresenta o número de pontos associados aos vértices do simplex. Na segunda coluna, temos o tempo de processamento, que cresce exponencialmente com o aumento do número de vértices. Por fim, na última coluna, é apresentado o erro relativo, que começa a ser não nulo, a partir do politopo 6-dimensional.

n	tempo(segundos)	f^*
3	0,0263	0,00
4	0,2917	0,00
5	0,3536	0,1057
6	0,3693	0,2952
7	0,3994	0,3407

Tabela 2: Algoritmo *Branch-and-Prune* utilizado neste trabalho.

A Tabela 2 contém os resultados utilizando o algoritmo *Branch-and-Prune*. Esta tabela apresenta valores pequenos para o número de pontos, assim como a anterior. O objetivo consiste em realizar uma comparação entre os dois métodos. Na segunda coluna, pode-se observar um crescimento polinomial do tempo de processamento, bastante inferior comparado com a mesma coluna na Tabela 1.

Na Tabela 3, é possível constatar que o algoritmo *Branch-and-Prune* também mantém um crescimento polinomial no tempo de processamento, à medida que n aumenta, bem como mantém reduzido o erro relativo.

Finalmente, a Figura 1 apresenta um gráfico comparativo do aumento do tempo de processamento em função do tamanho do problema. É possível observar o crescimento exponencial associado à abordagem que adota o algoritmo *Branch-and-Bound*, enquanto que a abordagem adotada neste trabalho, baseada no algoritmo *Branch-and-Prune*, apresenta um crescimento polinomial.

n	tempo(segundos)	f^*
10	0,4059	0,3906
20	0,4770	0,4488
30	0,5657	0,4668
40	0,6555	0,4753
50	0,7554	0,4804
100	1,3403	0,4903

Tabela 3: Algoritmo *Branch-and-Prune* para instâncias maiores que a da literatura

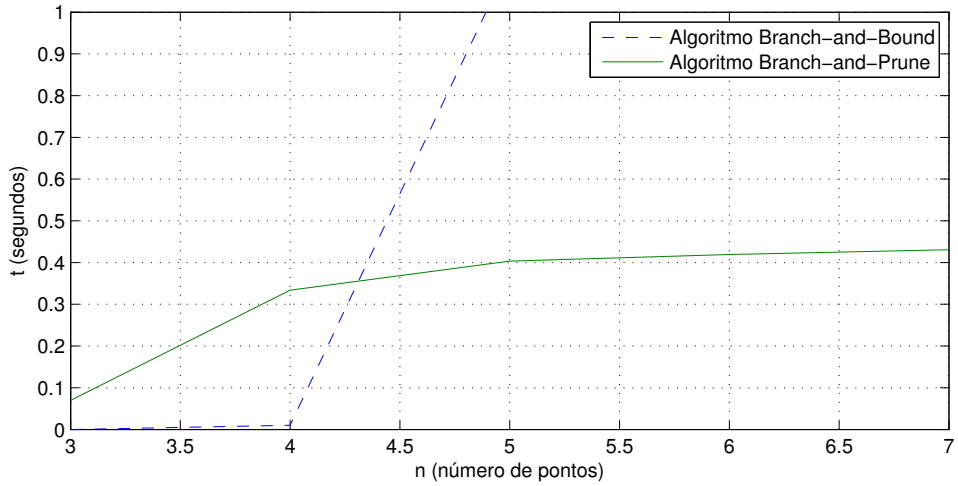

Figura 1: Comparação entre duas abordagens

5 Conclusões

Neste trabalho, propusemos uma nova abordagem para o problema de escalonamento multidimensional, baseada em um algoritmo *Branch-and-Prune*. Para a utilização deste algoritmo em $\mathbf{R}^m$, originalmente proposto em $\mathbf{R}^3$, foi necessário um ordenamento prévio dos pontos em $\mathbf{R}^m$. Os resultados computacionais obtidos apresentam um crescimento polinomial do tempo de processamento, comparado com o crescimento de tempo exponencial do algoritmo proposto em Žilinskas (2009). Este resultado nos motiva a continuar com esta linha de pesquisa estendendo esta abordagem a outras instâncias do problema de escalonamento multidimensional.

Referências

[1] Alencar, J., Alonso, A., Carvalho, S., Lavor, C. e Oliveira, A. (2012), *Different orders for discretization of multidimensional scaling problems*, Em Preparação.

[2] Arabie, P. (1991). *Was Euclid an unnecessarily sophisticated psychologist?* Psychometrika, 56:567-587.

[3] Borg, I. e Groenen, P. (2010), *Modern Multidimensional Scaling: Theory and Applications*, Springer.

[4] Liberti, L., Lavor, C. e Maculan, N. (2008), *A Branch-and-Prune algorithm for the Molecular Distance Geometry Problem*, International Transactions in Operational Research, 15:1-17.

[5] Phillips, A. T., Rosen, J.B., e Walke, V.H. (1996), *Molecular structure determination by convex underestimation of local energy minima*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 23, pp. 181-198, American Mathematical Society, Providence.

[6] Žilinskas, A. e Žilinskas, J. (2009), *Branch and Bound algorithm for multidimensional scaling with city-block metric*, Journal of Global Optimization, 43:357-372.