20 e 21 de outubro
Instituto Nacional de Pesquisas Espaciais - INPE
São José dos Campos - SP

Avaliação do algoritmo de colisão de múltiplas partículas na solução de problemas inversos

Eduardo F. P. Luz[1], José C. Becceneri[2], Haroldo F. Campos Velho[2]

[1]Programa de Doutorado em Computação Aplicada – CAP
Instituto Nacional de Pesquisas Espaciais – INPE

[2]Laboratório Associado de Computação e Matemática Aplicada – LAC
Instituto Nacional de Pesquisas Espaciais – INPE

{eduardo.luz, becce, haroldo}@lac.inpe.br

Abstract. *The Multiple Particle Collision Algorithm (MPCA) is an stochastic optimization method developed specially for high performance computational environments. Its advantages resides in the intense use of computational power provided by multiple processors in the task of search the solution space for a near optimum solution. This work presents the application of MPCA in solving inverse problems written as optimization problems, its advanges and disadvantages are also described, so are the obtained results.*

Resumo. *O Algoritmo de Colisão de Múltiplas Partículas (MPCA) é um método de otimização estocástico desenvolvido especialmente para ambientes computacionais de alto desempenho. Sua vantagem reside no aproveitamento do poder computacional provido por múltiplos processadores na tarefa de explorar o espaço de soluções em busca de uma solução próxima do ótimo. Este trabalho apresenta a aplicação do MPCA na solução de problemas inversos escritos como problemas de otimização, suas vantagens e desvantagens também são descritas, assim como os resultados obtidos.*

Palavras-chave: *Metaheurística, Otimização, Problemas inversos.*

1. Introdução

A teoria da otimização é o ramo da matemática que engloba o estudo quantitativo da solução ótima e dos métodos usados para encontrar esta solução. Esta teoria é vista em prática quando analisamos a coleção de técnicas, métodos, procedimentos e algoritmos que podem ser usados para localizar esta solução ótima [Antoniou and Lu 2007].

Problemas de otimização tem por objetivo encontrar a melhor combinação dentre um conjunto de variáveis para maximizar ou minimizar uma função, definida como sendo

uma função objetivo ou função custo. Os problemas de otimização podem ser classificados como [Becceneri 2008]:

- Problemas de otimização contínua: cujas variáveis assumem valores reais ou contínuos;
- Problemas de otimização combinatória (ou discreta): cujas variáveis assumem valores discretos ou inteiros;
- Problemas de otimização mista: com variáveis inteiras e contínuas ao mesmo tempo.

Os procedimentos algorítmicos utilizados para resolver os problemas listados acima geram uma grande lista de métodos, que podem ser classificados inicialmente como:

- Métodos determinísticos (ou exatos)
  - de ordem zero: onde somente o valor da função objetivo é utilizado, e.g., direções conjugadas de Powell;
  - de primeira ordem: usam o valor da função objetivo e de sua primeira derivada, e.g., máxima descida;
  - de segunda ordem: usam o valor da função objetivo, de sua primeiro e segunda derivada, e.g., método de Newton.
- Métodos estocásticos (ou aleatórios), e.g., recozimento simulado.

Uma das principais vantagens dos métodos estocásticos reside no fato destes poderem balancear o comportamento de busca global (*exploration*) e o comportamento de busca local (*exploitation*). O ajuste deste balanço possibilita que o algoritmo não fique preso em ótimos locais e continue sua busca pelo ótimo global.

Já os problemas inversos são formulações matemáticas que envolvem a determinação de causas desconhecidas a partir de efeitos observados, ou desejados. Na maioria das vezes, problemas inversos são de difícil solução, e a teoria proposta por [Tikhonov and Arsenin 1977] apresenta uma formulação de problemas inversos como um problema de otimização.

Este trabalho apresenta um método de otimização contínua, baseado no comportamento estocástico de partículas dentro de um reator nuclear para a solução de problemas inversos descritos como problemas de otimização.

2. Algoritmo de colisão de múltiplas partículas

Sistemas computacionais de alto desempenho são uma realidade constante em diversos centros de pesquisa. Estes sistemas são capazes de fornecer Teraflops (10^{12} operações de ponto flutuante por segundo) ao usuário, possibilitando a solução de problemas de altas dimensões e/ou grande porte com uso de novos algoritmos que tiram vantagem destes ambientes.

O Algoritmo de Colisão de Múltiplas Partículas (*Multiple Particle Collision Algorithm*, MPCA) é uma variante do Algoritmo de Colisão de Partículas (*Particle Collision Algorithm*, PCA) sendo que este último foi apresentado por [Sacco and de Oliveira 2005] e se baseia na metodologia apresentada por [Metropolis et al. 1953], que define um algoritmo de rejeição usado para gerar uma sequencia de amostras de uma distribuição de probabilidade de difícil amostragem direta.

O PCA se assemelha conceitualmente na física das reações de colisões de partículas, com grande ênfase nos comportamentos de espalhamento (*scattering*) e absorção (*absortion*). Seu uso tem sido comprovadamente eficaz em diversos casos de otimização, de problemas de teste a problemas de aplicações reais.

Os principais parâmetros a serem ajustados no PCA são: o número de iterações do algoritmo; o número de iterações da busca local; o tamanho do raio de perturbação (geração de nova solução candidata); o tamanho do raio de perturbação da busca local (no caso da utilização do esquema canônico); e o a função de probabilidade usada no cálculo do espalhamento.

A execução do PCA se inicia com uma solução aleatória que é avaliada frente a função objetivo e então modificada por uma perturbação estocástica levando a uma nova solução candidata que é então comparada com a solução anterior. Se a nova solução for melhor ela é aceita e se inicia um procedimento de busca local. Esta busca local originalmente se baseia na construção de soluções candidatas adicionadas de uma perturbação estocástica de ordem menor do que a perturbação inicial, porém outras soluções para a busca local podem facilmente ser exploradas, e.g., máxima descida, intensificando a característica de modularidade do algoritmo, possibilitando uma hibridização forte do mesmo.

Se a solução candidata não for aceita, inicia-se o procedimento de espalhamento, caracterizado pelo comportamento de escape de ótimos locais. Este escape é baseado no esquema de Metrópolis e é dado uma probabilidade calculada de acordo com a Eq. 1:

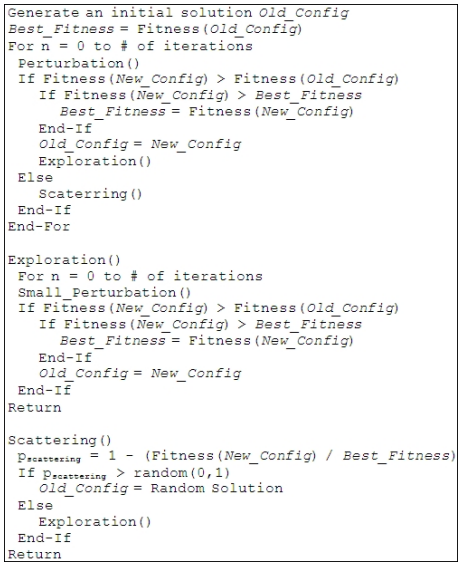
$$p_{scat} = 1 - \frac{F(NewConfig)}{F(BestConfig)} \tag{1}$$

onde $F(NewConfig)$ é o valor da função objetivo para a solução candidata e $F(BestConfig)$ é o valor da função objetivo para a melhor solução até o momento. O valor de p_{scat} é comparado com um número aleatório de distribuição uniforme e se for menor, aplica-se a função de espalhamento.

Analisando o algoritmo do PCA apresentado anteriormente e adotando N como o número de iterações definidas pelo usuário podemos identificar o loop mais interno como sendo responsável por N chamadas da função objetivo. Neste laço a chamada para uma função mais interna gera mais N chamadas à função objetivo. Portando $N \times N$ operações de chamada à função objetivo (operação mais custosa do algoritmo) são executadas, levando a uma complexidade $O(N^2)$.

O MPCA [da Luz et al. 2008] é um algoritmo de busca estocástico baseado no PCA, com a introdução de uma nova característica: o uso de várias partículas ao invés de uma única na exploração de maneira colaborada do espaço de buscas. Esta exploração múltipla é coordenada por uma função de comunicação e sincronização de informação implementada com o uso de MPI, uma biblioteca de troca de mensagens desenvolvida para proporcionar comunicação em uma máquina de arquitetura de memória distribuída.

O código para o MPCA é relativamente similar ao código do PCA, portanto a execução de $N \times N$ chamadas à função objetivo (o trecho de código mais custoso) se mantém, mas devido à existência de um novo laço de repetição introduzido pelo controle

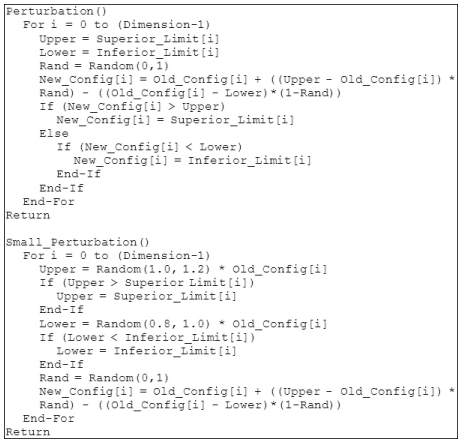


```
Generate an initial solution Old_Config
Best_Fitness = Fitness(Old_Config)
For n = 0 to # of iterations
 Perturbation()
 If Fitness(New_Config) > Fitness(Old_Config)
    If Fitness(New_Config) > Best_Fitness
       Best_Fitness = Fitness(New_Config)
    End-If
    Old_Config = New_Config
    Exploration()
 Else
    Scaterring()
 End-If
End-For

Exploration()
 For n = 0 to # of iterations
 Small_Perturbation()
 If Fitness(New_Config) > Fitness(Old_Config)
    If Fitness(New_Config) > Best_Fitness
       Best_Fitness = Fitness(New_Config)
    End-If
    Old_Config = New_Config
 End-If
Return

Scattering()
 p_scattering = 1 - (Fitness(New_Config) / Best_Fitness)
 If p_scattering > random(0,1)
    Old_Config = Random Solution
 Else
    Exploration()
 End-If
Return
```

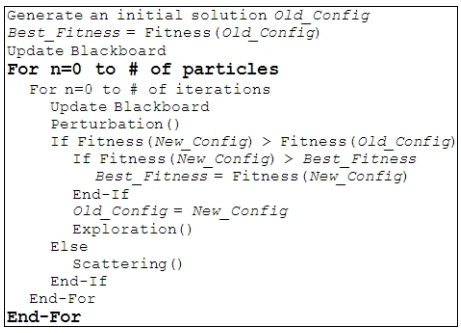
Figura 1. Descrição textual para o laço de repetição principal do PCA, função de exploração (*Exploration*) e de espalhamento (*Scattering*).



```
Perturbation()
  For i = 0 to (Dimension-1)
    Upper = Superior_Limit[i]
    Lower = Inferior_Limit[i]
    Rand = Random(0,1)
    New_Config[i] = Old_Config[i] + ((Upper - Old_Config[i]) *
    Rand) - ((Old_Config[i] - Lower)*(1-Rand))
    If (New_Config[i] > Upper)
       New_Config[i] = Superior_Limit[i]
    Else
       If (New_Config[i] < Lower)
          New_Config[i] = Inferior_Limit[i]
       End-If
    End-If
  End-For
Return

Small_Perturbation()
  For i = 0 to (Dimension-1)
    Upper = Random(1.0, 1.2) * Old_Config[i]
    If (Upper > Superior Limit[i])
       Upper = Superior_Limit[i]
    End-If
    Lower = Random(0.8, 1.0) * Old_Config[i]
    If (Lower < Inferior_Limit[i])
       Lower = Inferior_Limit[i]
    End-If
    Rand = Random(0,1)
    New_Config[i] = Old_Config[i] + ((Upper - Old_Config[i]) *
    Rand) - ((Old_Config[i] - Lower)*(1-Rand))
  End-For
Return
```

Figura 2. Descrição textual para a função de perturbação (*Perturbation*) e de busca local (*Small_Perturbation*). Esta última pode ser facilmente substituída por qualquer outro método de busca local.



```
Generate an initial solution Old_Config
Best_Fitness = Fitness(Old_Config)
Update Blackboard
For n=0 to # of particles
   For n=0 to # of iterations
      Update Blackboard
      Perturbation()
      If Fitness(New_Config) > Fitness(Old_Config)
         If Fitness(New_Config) > Best_Fitness
            Best_Fitness = Fitness(New_Config)
         End-If
         Old_Config = New_Config
         Exploration()
      Else
         Scattering()
      End-If
   End-For
End-For
```

Figura 3. Algoritmo para o MPCA.

das múltiplas partículas, o número de chamadas à função objetivo pode ser elevado ao caso $N \times N \times N$, onde se assume que o número de partículas é igual (ou maior) ao

número de processadores.

Da maneira apresentada, a complexidade estimada do MPCA é $O(n^3)$, para um pior caso, porém, quando da distribuição do trabalho entre p processadores, podemos adotar $p = n$, onde n é o número de partículas em uso e desta forma a complexidade retorna a $O(n^2)$, que é a mesma complexidade do PCA canônico.

Os parâmetros do MPCA são os mesmos do PCA sendo necessário incluir o número de partículas a serem utilizadas na busca e o número de processadores que serão utilizados.

3. Problemas inversos

Ao tratarmos da formulação matemática de um problema direto, partimos de uma situação causadora e chegamos, através da aplicação de uma metodologia, cálculo ou equação, aos efeitos gerados pelo fenômeno modelado no problema direto.

Já "resolver um problema inverso é determinar causas desconhecidas a partir de efeitos desejados ou observador" [Engl et al. 1996], sendo esta uma das definições mais recorrentes em uso, levando à condição de que estudar problemas inversos se consiste de usar os resultados de observações para inferir valores de parâmetros que caracterizam o sistema sob investigação.

Desta maneira, se admitirmos que um certo modelo matemático pode ser expresso por $A(u) = f$, então o modelo inverso relativo a este problema é representado por $A^{-1}(f) = u$. Na Fig. 4 a representação gráfica de um problema direto é dado pelo caminho que liga o espaço das causas ao espaço dos efeitos e o problema inverso liga o espaço dos efeitos ao espaço das causas.

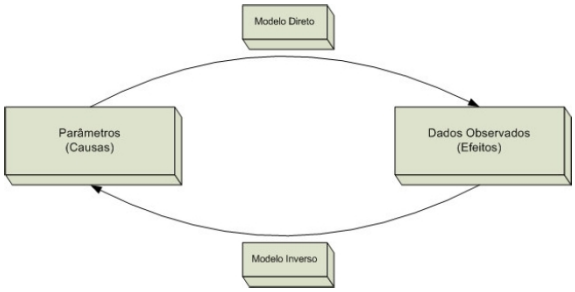


Figura 4. Problemas direto e inverso ligando o espaço de causas e efeitos.

A grande maioria dos problemas inversos são pertencentes à classe dos problemas mal-postos, i.e., aqueles tipos de problemas que violam uma ou mais das condições de Hadamard, que são:

- Existência: o problema deve possuir uma solução;
- Unicidade: esta solução deve ser única;
- Estabilidade: a dependência dos dados deve ser contínua.

A quebra da terceira condição, de estabilidade, leva a uma condição dita como "mal-condicionamento" que pode ser solucionada com o uso de regularização. Maiores informações sobre a teoria de regularização podem ser obtidas em [Tikhonov and Arsenin 1977].

3.1. Transferência radiativa

O primeiro problema a ser apresentado neste trabalho consiste-se de um problema inverso de transferência radiativa em um meio homogêneo plano-paralelo (Fig. 5). Maiores informações sobre o problema podem ser obtidas em [Silva Neto and Becceneri 2009].

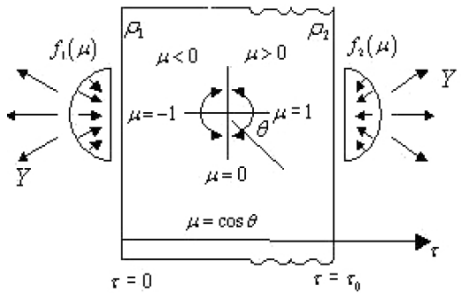


Figura 5. Descrição esquemática do problema de transferência radiativa submetido à incidência de radiação externa de intensidades f_1 e f_2.

As incógnitas a serem estimadas são:

$$\vec{Z} = \tau_0, \omega, \rho_1, \rho_2{}^T \tag{2}$$

em que τ_0 representa a espessura ótica, ω representa o albedo de espalhamento simples e ρ_1 e ρ_2 representam as refletividades difusas.

O problema de otimização neste caso é escrito como uma minimização de erros quadráticos:

$$Q(\vec{Z} = \sum_{i=1}^{N_d} \Big[I_i(\vec{Z}) - Y_i \Big]^2 \tag{3}$$

em que I_i corresponde ao valor calculado pelo modelo frente à solução candidata e Y_i representa o valor medido experimentalmente.

Esta é uma formulação dita implícita pois as incógnitas não aparecem diretamente na formulação da solução, mas estão embutidas na formulação do problema direto correspondente e os seus efeitos são percebidos quando o problema direto é resolvido e então participam da composição da função objetivo [Silva Neto and Becceneri 2009].

3.2. Localização de fontes de poluição

Para representar a dispersão de partículas na atmosfera, neste segundo problema, adotamos um modelo lagrangiano de dispersão de partículas (*Lagrangean Model for Buoyant Dispersion in Atmosphere*, LAMBDA), que se baseia na forma tridimensional da equação de Langevin para um campo aleatório de velocidade, seguindo a derivação de Thomson [Roberti 2005].

O modelo de dispersão de partículas LAMBDA simula uma certa quantidade de partículas computacionais que imitam o comportamento de uma partícula de contaminante atmosférico real, com o objetivo de simular os movimentos atmosférico sofridos por elementos de fluido ou contaminantes liberados na atmosfera (Fig. 6).

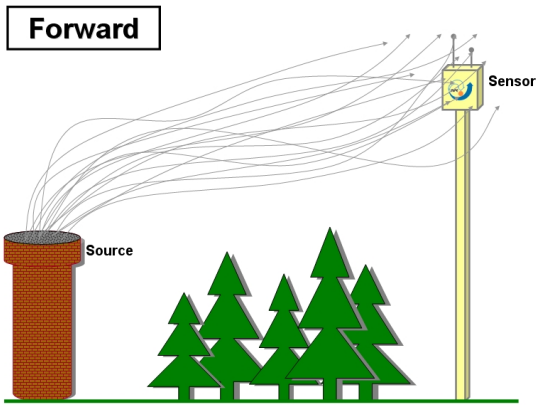


Figura 6. Dispersão de partículas lagrangeana: partículas emitidas de uma fonte de poluição são detectadas por um sensor. As informações do sensor são usadas para localizar a fonte e estimar a sua intensidade de emissão.

A integração temporal para a dispersão lagrangiana calcula a concentração média de um contaminante na posição $\vec{x}$ e no tempo t, dada uma taxa de emissão em uma fonte $S(kgm^{-3}s^{-1})$, sendo definida como:

$$C(x,t) = \int_{-\infty}^{t} \int_{-\infty}^{\infty} S\left(\vec{x_0}, t_0\right) P^a\left(\vec{x}, t | \vec{x_0}, t_0\right) d\vec{x_0} dt_0 \tag{4}$$

em que $P^a(\vec{x}, t|\vec{x_0}, t_0)$ é a densidade probabilidade de transição avançada no tempo, definida para que $P^a(\vec{x}, t|\vec{x_0}, t_0)d\vec{x_0}$ seja a probabilidade de que um elemento de fluido inicialmente em $(\vec{x_0}, t_0)$ seja encontrado no tempo t e volume $d\vec{x}$ centrado em $\vec{x}$.

O modelo LAMBDA implementa o cálculo da concentração em um dado sensor como:

$$C_j = \sum_{i=1}^{N_f} S_i \frac{V_{f,i}}{V_{s,j}} \frac{\Delta t}{N_{PEF,i}} N_{PVS,i,j} \tag{5}$$

com C_j representando a concentração do j-ésimo sensor, N_f o número de fontes, S_i a intensidade da i-ésima fonte, $V_{f,i}$ o volume da i-ésima fonte, $V_{s,j}$ o volume do j-ésimo sensor, Δt a discretização temporal, $N_{PEF,i}$ o número de partículas emitidas da i-ésima fonte e $N_{PVS,i,j}$ o número de partículas emitidas da i-ésima fonte que se encontram no j-ésimo sensor.

Para resolver este problema na formulação de um problema de minimização, utiliza-se um modelo fonte-receptor, que reduz o esforço computacional requerido pela resolução iterativa do modelo direto. Desta forma, calculamos:

$$\vec{C} = M\vec{S}, \tag{6}$$

em que $\vec{C}$ é um vetor de elementos que representam as concentrações médias nos sensores,

M representa a matriz de transição de estados, e $\vec{S}$ representa a intensidade nas fontes de emissão ou absorção. A matriz M é construída com base na Eq. 5, sendo definida como:

$$M_{ij} = \frac{V_{f,i}}{V_{s,j}} \frac{\Delta t}{N_{PEF,i}} N_{PVS,i,j} \tag{7}$$

4. Resultados obtidos

Os resultados apresentados nesta seção levam em consideração a média de 10 experimentos com sementes geradoras de números aleatórios distintas e dados experimentais gerados artificialmente pelo uso dos modelos diretos e posterior adição de ruído aos dados com o intuito de simular a coleta por instrumentos reais. O nível de ruído adotado foi de 2% e visa demonstrar a aplicação do MPCA na solução de problemas inversos.

Os parâmetros utilizados foram: 8 partículas; 8 processadores; 10000 iterações; 1000 passos de busca local; raio de perturbação Gaussiano $N(0, 1)$; raio de busca local correspondente a 20% do raio de perturbação gaussiano. O algoritmo foi executando em um Cray XT5.

A Tabela 1 apresenta os resultados obtidos para o problema inverso de transferência radiativa. Nota-se que o parâmetro ρ_1 obteve a pior estimativa, porém este é um resultado esperado [Silva Neto and Becceneri 2009].

Tabela 1. Resultados para a aplicação do MPCA no problema inverso de transferência radiativa.

	τ_0	ω	ρ_1	ρ_2
Resultado exato	$1,0000$	$0,5000$	$0,1000$	$0,9500$
Resultado médio	$0,9954$	$0,5144$	$0,1620$	$0,9540$
Desvio padrão	$2,51\text{E}{-2}$	$2,42\text{E}{-2}$	$8,09\text{E}{-2}$	$2,09\text{E}{-3}$

A Tabela 2 apresenta os resultados para a localização de duas fontes de emissão/absorção de poluição. As fontes alternam o comportamento de emissão para absorção e este comportamento é devidamente capturado pelo algoritmo de otimização.

Tabela 2. Resultados para a aplicação do MPCA no problema inverso de localização de fontes de emissão.

	Área 1 - Taxa 1	Área 1 - Taxa 2	Área 2 - Taxa 1	Área 2 - Taxa 2
Resultado exato	$0,90$	$-0,40$	$0,50$	$-0,80$
Resultado médio	$0,8875$	$-0,4114$	$0,5127$	$-0,8820$
Desvio padrão	$0,1163$	$1,18\text{E}{-2}$	$1,96\text{E}{-2}$	$6,59\text{E}{-2}$

5. Considerações finais

O MPCA se mostrou uma alternativa viável à solução de problemas inversos principalmente quando existe a disponibilidade de recursos de computação de alto desempenho.

Atualmente a presença de computadores pessoais com arquiteturas multi-core possibilita a execução de um algoritmo desenvolvido para ambientes de alto desempenho sem

muita perda de desempenho, levando o usuário a um melhor aproveitamento dos recursos disponíveis.

Os resultados demonstram a convergência do MPCA para uma boa solução e abrem a possibilidade de maiores estudos, visando principalmente a hibridização do algoritmo, através da substituição do esquema de busca local, que atualmente é baseado na geração de soluções candidatas vizinhas dentro de um raio Gaussiano por outro esquema mais eficiente.

6. Referências bibliográficas

Referências

Antoniou, A. and Lu, W.-S. (2007). *Practical optimization: algorithms and engineering applications*. Springer, New York.

Becceneri, J. C. (2008). *Meta-Heurísticas e Otimização Combinatória: Aplicações em Problemas Ambientais*. Computação e Matemática Aplicada às Ciências e Tecnologias Espaciais. INPE, São José dos Campos.

da Luz, E. F. P., Becceneri, J. C., and de [Campos Velho], H. F. (2008). A new multi-particle collision algorithm for optimization in a high-performance environment. *Journal of Computational Interdisciplinary Sciences*, 1:1–7.

Engl, H. W., Hanke, M., and Newbauer, A. (1996). *Regularization of inverse problems*. Kluwer, Dordrecht.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092. Disponível em: ¡http://link.aip.org/link/?JCP/21/1087/1¿.

Roberti, D. R. (2005). *Problemas inversos em física da atmosfera*. PhD thesis, Universidade Federal de Santa Maria.

Sacco, W. F. and de Oliveira, C. R. (2005). A new stochastic optimization algorithm based on a particle collision metaheuristic. In *Procedings of 6th World Congress of Structural and Multidisciplinary Optimizaion*, Rio de Janeiro. WCSMO.

Silva Neto, A. J. and Becceneri, J. C. (2009). *Técnicas de inteligência computacional inspiradas na natureza: aplicação em problemas inversos em transferência radiativa*. Notas em Matemática Aplicada. SBMAC, São Carlos.

Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solution of ill-posed problems*. John Wiley & Sons ltd., New York.