Estimadores com Plausibilidade Máxima e Otimização Global Estocástica

RESUMO

Um dos métodos mais utilizados para a determinação de funções de distribuição de probabilidades (PDFs) associadas a amostras aleatórias de uma dada população ou processo é o chamado Método da Plausibilidade Máxima (ou Maximum Likelihood Method , em Inglês) , baseado na premissa de que o conjunto dos parâmetros correspondente à PDF real da população sob estudo é aquele que maximiza a função de plausibilidade (likelihood function) , calculada nos pontos amostrais previamente obtidos . Para efetivar a referida otimização, é prática comum a utilização de algoritmos determinísticos relacionados ao gradiente da função acima (ou seu logaritmo , para melhor condicionamento computacional) .

O presente trabalho apresenta uma abordagem alternativa para obtenção do ponto de máximo global , baseada no paradigma conhecido como Fuzzy Adaptive Simulated Annealing , ou Fuzzy ASA . Após a exposição teórica , vários resultados de aplicação serão mostrados e analisados .

Palavras-chave: MLE ; Lógica Fuzzy ; Estimação Estatística ; Otimização Global ; Plausibilidade Máxima ; ASA ;

ABSTRACT

One of the most common methods to determine the probability density function (PDF) associated to independent samples of a given population or process is the so called Maximum Likelihood Method, based upon the principle that the set of parameters correspondeing to the PDF of the actual population under study is one that maximizes the likelihood function, calculated at the sample points. To do the optimization itself , it's common practice to use gradient based , deterministic , algorithms applied to the cited function (or its logarithm , to improve computational conditioning).

This work presents an alternative computational approach to find the global maximum of the function described above, using the method known as Fuzzy Adaptive Sinulated Annealing or Fuzzy ASA . After theoretical explanation, some results are shown and analyzed .

Keywords: MLE ; Maximum Likelihood Estimation ; Fuzzy Logic ; Global Optimization ; Statistical Estimation ; Adaptive Simulated Annealing ; ASA ;

1. INTRODUÇÃO

O método da plausibilidade máxima é uma técnica bastante popular para obtenção de estimadores de parâmetros de distribuições de probabilidades.

O procedimento geral é simples:

Dada uma amostra independente e identicamente distribuída (X1, X2 , ... , Xn) de uma população com suposta função densidade de probabilidade $f(x \mid \theta 1; \theta 2;... \theta \kappa)$, a função de plausibilidade é definida por:

$$L(\theta \mid x) = L(\theta 1,\theta 2,...,\theta k \mid x1,x2,...xn) = \Pi\, f(xi \mid \theta 1; \theta 2;... \theta \kappa) .$$

Quando as premissas de independência e/ou igualdade de distribuições não são satisfeitas para o conjunto amostral , a função de plausibilidade é definida como a densidade conjunta $f(x1,x2,...xn \mid \boldsymbol{\theta})$, considerada como função do vetor $\boldsymbol{\theta}$, uma vez que a amostra é fixa .

O vetor θ^* que maximiza $L(\theta \mid X)$ é denominado estimador de plausibilidade máxima sendo , como o nome indica, uma aproximação do vetor paramétrico correspondente à distribuição real da população ou processo sob estudo .

Embora tenha sido definida por meio da expressão acima, a maximização propriamente dita é realizada normalmente utilizando seu logaritmo , tendo em vista que tal transformação tende a prevenir a ocorrência de *underflows* , além de simplificar a expressão da função de plausibilidade original . Tendo em vista a monotonia da função logaritmo (base 10 , em particular) , o maximizante da função transformada coincidirá com o da função original . Na literatura pertinente encontramos inúmeros casos onde o procedimento de otimização é realizado por meio da anulação do gradiente da função-produto e posterior solução das equações resultantes .

Infelizmente , tal abordagem não pode ser adotada nos casos em que a função não é diferenciável e se mostra pouco eficaz caso a função-objetivo apresente diversos pontos de máximo locais. Outro ponto desfavorável desta abordagem tradicional ocorre quando as equações obtidas pela anulação do gradiente são de difícil tratamento e/ou não possuem expressão de solução em forma fechada . Logo, métodos computacionais alternativos podem vir a contribuir para ampliar o campo de aplicação do método da plausibilidade máxima.

Tais métodos deveriam possuir as seguintes características:

- Prescindir da continuidade ou diferenciabilidade das funções-objetivo .
- Realizar otimização global, ou seja, não ser "aprisionados"em pontos de máximo locais.
- Possuir comprovação teórica de convergência, ao menos em probabilidade , a pontos de máximo global .

O método conhecido como *Adaptive Simulated Annealing* apresenta todas as características acima e se mostra adequado para uso na tarefa em questão. Nas seções a seguir serão apresentados alguns de seus aspectos mais importantes .

2. ESTRUTURA GERAL DOS ALGORITMOS BASEADOS EM *SIMULATED ANNEALING*

Algoritmos baseados em SA utilizam princípios idealizados por Nicholas Metropolis e outros , sendo conhecidos genericamente pelo rótulo de métodos de Monte Carlo.

A abordagem usa três componentes fundamentais que têm grande impacto na implementação final:

- Uma densidade de probabilidade g(.) , usada na geração de novos pontos candidatos.
- Uma densidade de probabilidade a(.) , usada na aceitação/rejeição de novos pontos.
- Um esquema de redução de temperaturas T(.) , que determina como as mesmas variarão durante a execução do algoritmo, ou seja, seu perfil dinâmico.

A estratégia básica é gerar um ponto inicial , escolhido de acordo com critérios convenientes , e ajustar a temperatura inicial de modo que o espaço de fase possa ser suficientemente explorado. A seguir , novos pontos são gerados de acordo com a PDF g(.) e probabilisticamente aceitos ou rejeitados , conforme determinar a PDF a(.).

Se ocorrer a aceitação , o ponto candidato é elevado à condição de ponto básico vigente . Durante a execução do algoritmo as temperaturas são reduzidas , provocando a redução da probabilidade de aceitação de novos pontos posteriormente gerados apresentando valores da função-objetivo superiores àquele do ponto básico corrente (no caso de minimização de funções).Entretanto , existe uma probabilidade não nula de escolha de pontos

situados acima (ou abaixo) deste último , tornando possível uma eventual "fuga" de mínimos (ou máximos) locais .

3. PRINCIPAIS ASPECTOS DA ABORDAGEM ASA

Como citado anteriormente , o método ASA é baseado no conceito de *simulated annealing* , possuindo um grande número de aspectos positivos . Dentre eles , podemos citar :

. *Re-annealing* – trata-se do re-escalonamento dinâmico das temperaturas paramétricas , adaptando PDFs geradoras para cada dimensão de acordo com as sensibilidades exibidas em cada direção de busca . Em poucas palavras , se a função objetivo não apresenta variações significativas quando alteramos um dado parâmetro , pode ser proveitoso estender a amplitude do intervalo de busca naquela dimensão em particular , e vice-versa .

. Facilidades de *Quenching* – como ressaltado anteriormente , a implementação do sistema ASA apresenta a possibilidade de ajuste manual (por parte do usuário) de vários parâmetros estruturais relacionados ao processo de *quenching* , o que poderá resultar em maior velocidade de convergência . Assim , é possível moldar a evolução das temperaturas paramétricas de modo amplo , fácil e limpo .

. Alto nível de parametrização – o sistema ASA foi idealizado de tal modo que possamos alterar virtualmente qualquer subsistema sem esforço significativo .Sendo assim , é possível mudar o comportamento dos processos de geração/aceitação , critérios de conclusão , geração de pontos iniciais , etc..

ASA foi projetado para encontrar pontos extremos globais pertencentes a um dado subconjunto compacto do espaço Euclidiano n-dimensional . Ele gera pontos componente a componente , de acordo com:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}_i \text{ ,}$$

$$\textbf{com } \Delta\mathbf{x}_i = \mathbf{y}_i(\mathbf{B}_i - \mathbf{A}_i) \text{ ,}$$

$$[\mathbf{A}_i, \mathbf{B}_i] = \textbf{faixa de variação da i - ésima dimensão ,}$$

$$\mathbf{y}_i \in [-1,1] \textbf{ é dado por}$$

$$\mathbf{y}_i = \textbf{sgn}(\mathbf{u}_i - 1/2)\mathbf{T}_i[(1 + 1/\mathbf{T}_i)^{|2\mathbf{u}_i - 1|} - 1] \quad \textbf{onde}$$

$$\mathbf{u}_i \in [0,1] \textbf{ é gerado por meio da distribuiç ão uniforme,}$$

$$\mathbf{T}_i = \textbf{temperatur a atual relativa à dimensão i .}$$

A compacidade do espaço de busca não é uma severa limitação na prática , e na falta de prévia informação sobre a possível localização de ótimos globais , basta escolher domínios hiper-retangulares suficientemente abrangentes .

4. CONTROLE FUZZY APLICADO AO MÉTODO ASA

Conforme citação anterior , o uso do mecanismo de *quenching* pode melhorar substancialmente a eficiência do processo de convergência , com a assunção do risco de alcance prematuro de extremos não globais . Em certos casos , contudo , podemos simplesmente não ter alternativa , como é o caso de funções com domínios de elevado número de dimensões . Para resolver o problema , um controlador fuzzy foi projetado . A abordagem é simples : consideramos ASA como um sistema dinâmico MISO (Multiple Input Single Output) e "fechamos a malha" , pela amostragem da saída de ASA (valor corrente da função objetivo) e atuação em suas entradas (um subconjunto de parâmetros ajustáveis em run-time , relacionados ao processo de quenching) de acordo com uma *lei fuzzy* (algoritmo de controle) , que nada mais faz do que emular o raciocínio humano sobre o processo subjacente . Assim , pelo uso de um controlador inteligente podemos acelerar e retardar a evolução das

temperaturas , além de sermos capazes de tomar ações evasivas em caso de convergência prematura .

Há dois principais obstáculos para alcançar tal objetivo :

1 – Como as saídas amostradas (valores da função objetivo) podem informar o estado atual do processo de otimização em andamento ?

2 – Como podemos alterar dinamicamente as entradas do sistema ASA de modo a eliminar situações indesejáveis , como permanência junto a máximos não globais ou progresso insatisfatório ?

A primeira questão foi resolvida graças ao conceito de função de sub-energia , usada no método TRUST [1].

A função de sub-energia é dada por :

$$\mathbf{SE(x,x_0) = log(1/[1 + exp(-(f(x) - f(x_0)) - a)])}$$

onde a é uma constante real , e x_0 é o "ponto básico" atual.

O ponto básico é o melhor ponto de máximo encontrado até então . Logo , a função SE comporta-se qualitativamente como a original f(.) quando o processo de busca "visita" pontos melhores do que o máximo corrente e tende a se "achatar" em pontos piores . Assim , é possível avaliar quando a busca está concentrada acima , nas imediações ou abaixo do ponto máximo atual pela inspeção dos valores assumidos pela função de sub-energia . Tal processo de detecção resulta em conclusões aproximadas como

A busca está PRÓXIMA do máximo vigente.

ou

A busca está MUITO DISTANTE do máximo vigente.

o que leva naturalmente a uma oportunidade de modelagem fuzzy .

A segunda questão acima está relacionada às partes conseqüentes da base de regras fuzzy , na qual temos que inserir ações corretivas para eventuais desvios em relação às diretrizes preestabelecidas para o processo de maximização . Isto foi feito pela variação dos graus de quenching para PDFs de geração e aceitação . A implementação usou fatores de quenching individuais para cada dimensão .

A base de regras do controlador fuzzy contém asserções como

- SE AveSub está PRÓXIMA a zero ENTÃO
  aumente o nível de Quenching.
- SE AveSub está PRÖXIMA ao extremo corrente ENTÃO
  aumente o nível de Quenching
- SE StdDevSub é ZERO ENTÃO
  decresça o nível de Quenching

onde :

- AveSub é uma variável lingüistica correspondente à média abrupta (crisp)

dos 100 últimos valores de sub-energia

- StdDevSub é uma variável lingüistica correspondente ao desvio-padrão abrupto (crisp) dos 100 últimos valores de sub-energia

A incorporação do referido controlador ao método original [2] recebeu o nome de FUZZY ASA [7] .

5. IMPLEMENTAÇÃO

Para realizar a estimação de parâmetros por meio da abordagem acima, foi construído um programa extensível cuja estrutura consiste de um núcleo fixo , interface gráfica amigável e que permite a incorporação de *plug-ins* contendo o código das funções a otimizar. Como resultado final , são fornecidos os parâmetros correspondentes aos pontos ótimos pertencentes ao domínio das funções sob tratamento. A opção foi por oferecer uma "máquina de minimização global" geral , pois , em caso de maximização , basta inverter (em termos aditivos) o sinal da função-objetivo no módulo descritor da mesma . A idéia é extremamente simples: para investigar os parâmetros de uma dada distribuição , dadas as amostras aleatórias , basta construir um módulo de software (cuja estrutura será detalhada abaixo) contendo alguns poucos métodos com interfaces bem definidas . Dentre eles , um será responsável por avaliar a função-objetivo (inverso aditivo do logaritmo da função de plausibilidade , no caso de MLE) em pontos arbitrários do domínio e devolver o valor correspondente ao chamador . Tal tipo de arquitetura permite desacoplar o mecanismo de otimização propriamente dito dos cálculos particulares relativos a uma dada função numérica. Logo, após os procedimentos de ajuste iniciais , o núcleo central inicia um processo iterativo , chamando continuamente a função responsável pelo cálculo dos valores numéricos e simulando um processo estocástico que visita estados com "energias" cada vez menores e rumo à configuração que deverá conter algum ótimo global para o problema – é claro que as condições para convergência deverão ser cumpridas para que o resultado desejado seja atingido. Além disso, pode ser necessário ajustar alguns parâmetros para obtenção de melhor desempenho

A seguir , mostramos algumas aspectos da interface gráfica e a *splash window*:

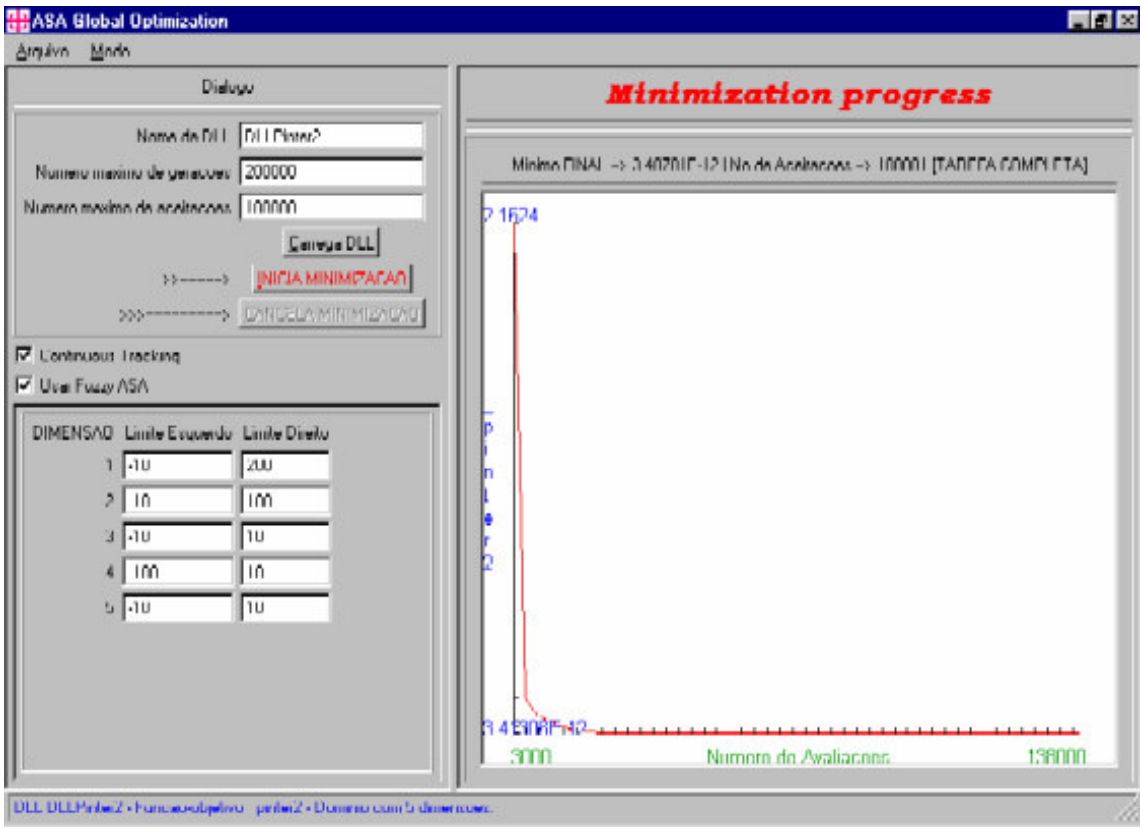

FIGURA 1 – Interface gráfica

FIGURA 2 – SPLASH WINDOW

O programa foi codificado na linguagem C++ , de modo a permitir seu fácil transporte a diversos ambientes, dentre eles vários “sabores” do sistema operacionais Unix (Linux , em particular) . Para utilizá-lo , basta codificar um módulo carregável dinamicamente que contenha , no mínimo , 3 rotinas específicas do problema em questão , a saber:

- **extern "C" char * NomeDaFuncaoObjetivo(void)**

Devolve uma string contendo o nome da função-objetivo a ser investigada.

- **extern "C" unsigned int NumeroDeParametrosDaFO(void)**

Devolve a dimensão do espaço no qual o domínio da função-objetivo está imerso.

- **extern "C" double FuncaoObjetivoPropriamenteDita**
  (
  - **int NoDeDimensoes,**
    double *Vetor,
    void *Custom
)

Devolve o valor da função, dados o ponto de cálculo e um ponteiro contendo informações de responsabilidade do usuário .

A título de exemplo, mostramos abaixo o código-fonte de um módulo completo na limguagem C :

```
#include <windows.h>
#include <math.h>

char NomeComUnderscore[] = "_pinter1";

BOOL  DllMain(HINSTANCE hInst,DWORD dwReason,LPVOID Reservado)
{

 switch (dwReason)
 {
      case DLL_PROCESS_ATTACH:                  break;
      case DLL_PROCESS_DETACH:                  break;
      case DLL_THREAD_ATTACH:                   break;
      case DLL_THREAD_DETACH:                   break;
 }

 return TRUE;

 }

extern "C" double  __declspec(dllexport)  pinter1(
                    int nodedimensoes, // No de elementos no vetor abaixo
                    double *x, // Vetor
                    void *Custom // Informacao especifica do usuario
                    )
{

int i;
```

```
double f1,f2,f3,aux;

    f1=0;
    for (i=0;i<nodedimensoes;i++)
        f1 += (i+1)*x[i]*x[i];

    f2 = pow(x[nodedimensoes-1]+5*sin(x[0])+x[1]*x[1],2);
    aux = pow(x[nodedimensoes-2]+5*sin(x[nodedimensoes-1])+x[0]*x[0],2) ;

    f2 = f2 + aux;

    for (i=1 ; i < nodedimensoes-1 ; i++)
       f2 += pow(x[i-1]+5*sin(x[i])+x[i+1]*x[i+1],2);

    f3 = log(1+fabs(pow(x[nodedimensoes-1],2)+2*x[0]+3*x[1]));
    f3 = f3*f3 ;
    aux     =     log(1+fabs(nodedimensoes*pow(x[nodedimensoes-2],2)+2*x[nodedimensoes-
1]+3*x[0]));
    aux = aux*aux;

    f3 = f3+aux;

  for (i=1 ; i < nodedimensoes-1 ; i++)
  {
   aux = log(1+fabs((1+i)*pow(x[i-1],2)+2*x[i]+3*x[i+1]));
   f3 += aux*aux ;
  }

  aux=f1+f2+f3;

  return aux ;

}

extern "C" char *  __declspec(dllexport)  NomeDaFuncaoObjetivo(void)
{
    return (char *) NomeComUnderscore;
}

extern "C" unsigned int  __declspec(dllexport)  NumeroDeParametrosDaFO(void)
{
    return 5;
}
```

A partir deste ponto, é obviamente necessário compilar e "link-editar" o programa-fonte para criação do binário (shared object , DLL , etc.) que será carregado em *run-time* pelo programa principal , após as devidas configurações via interface gráfica . Os controles disponíveis na versão atual são relativos ao nome do arquivo que contem o código da biblioteca dinamicamente carregável , ao número de aceitações e gerações do algoritmo de otimização , à definição do domínio de otimização que restringe o espaço de estados do processo a ser simulado e ao tipo de algoritmo a ser utilizado durante a execução . Como

instrumento complementar, uma janela mostra a evolução dos valores numéricos assumidos pelos pontos intermediários, na busca pelo extremo global .

6. OBTENÇÃO DE ESTIMADORES COM PLAUSIBILIDADE MÁXIMA

Mostraremos agora o resultado da aplicação do programa acima à obtenção simultânea dos 2 parâmetros de uma distribuição lognormal , tendo como entrada amostras geradas *a priori* por meio da CDF ideal (distribuição com parâmetros 4 e 2) e compostas por 100 , 500 ,5000 e 8000 elementos – cabe ressaltar que o procedimento não lançará mão de qualquer método relacionado a gradientes e outros mecanismos do Cálculo Diferencial , além de desconsiderar por completo aspectos particulares desta distribuição específica . Primeiramente , apresentamos o código-fonte da DLL utilizada:

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <exceptio.h>
# include <cstdlib>
# include <iostream>
# include <iomanip>
# include <cmath>
# include <ctime>
#include "prob.h"

#define ARQUIVO "AMOSTRAS.DAT"
#define NOMAXIMODEENTRADAS 10000

char NomeComUnderscore[] = "_MLELognormal";

static int NoDeAmostras ;
double SampleArray[NOMAXIMODEENTRADAS];

BOOL  DllMain(HINSTANCE hInst,DWORD dwReason,LPVOID Reservado)
{
FILE *fp;
char amostra[100] ;

 switch (dwReason)
 {
      case DLL_PROCESS_ATTACH:

            NoDeAmostras = 0;
            fp = fopen(ARQUIVO,"r");

            while ( fscanf(fp,"%s",amostra) != EOF  && \
                  NoDeAmostras < NOMAXIMODEENTRADAS
                )
                        {
                         SampleArray[NoDeAmostras] = strtod(amostra,NULL);
                              NoDeAmostras++ ;
```

```
                        }
            fclose(fp);


   if (NoDeAmostras == 0) return FALSE; // Inicializacao falhou
                              // => impede a carga

                     break;

      case DLL_PROCESS_DETACH:                    break;
      case DLL_THREAD_ATTACH:                     break;
      case DLL_THREAD_DETACH:                     break;
 }

 return TRUE;

 }

extern "C" double  __declspec(dllexport)  MLELognormal(
                     int nodedimensoes, // No de elementos no vetor abaixo
                     double *x, // Vetor
                     void *Custom // Informacao especifica do usuario
                     )
{
double aux = 0 ;
double resu ;


 for ( int i=0 ; i < NoDeAmostras ; i++ )
     {
        resu = log_normal_pdf(SampleArray[i],x[0],x[1]) ;

        if (resu < 1e-20)
             aux += -30;
        else  aux += log(resu) ;

     }

  return -aux ; /// Note a inversao aditiva para maximização

}

extern “C” char *  __declspec(dllexport)  NomeDaFuncaoObjetivo(void)
{
    return (char *) NomeComUnderscore;
}

extern “C” unsigned int  __declspec(dllexport)  NumeroDeParametrosDaFO(void)
{
    return 2;
}
```

Foram gerados 4 conjuntos de amostras independentes contendo 100 , 500 , 5000 e 8000 pontos distribuídos de acordo com a lognormal com parâmetros (4,2) , cada um dos quais tendo sido utilizado para estimar os 2 parâmetros da referida distribuição de acordo com o método acima apresentado. Os resultados são mostrados na tabela abaixo:

No de pontos na amostra	**Parâmetro A estimado**	**Parâmetro B estimado**
100	**3.696686**	**2.019308**
500	**3.964747**	**1.92806**
5000	**3.982984**	**1.968152**
8000	**3.994115**	**1.990389**

Pode-se observar que o comportamento assintótico do procedimento proposto é coerente com os resultados teóricos, uma vez que o par de parâmetros estimados tende visivelmente ao par (4,2) quando o número de amostras aumenta .

7. CONCLUSÃO

Tendo em vista as dificuldades relacionadas à aplicação de técnicas convencionais de otimização ao Método da Plausibilidade Máxima quando as distribuições sob estudo apresentam expressões de complexidade elevada, foi demonstrado que , utilizando técnicas estocásticas de otimização global , podemos obter soluções de modo uniforme e independentemente de certas características analíticas das expressões funcionais das respectivas distribuições de probabilidade .

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Barhen,J. / Protopopescu ,V. / Reister.D.
TRUST:A Deterministic Algorithm for Global Optimization
Science Magazine ,Vol. 276.

[2] Ingber , L.
ASA : Lessons Learned
Disponível em www . ingber . com

[3] Pintér , J.
Global Optimization in Action
Kluwer Academic Publishers - 1996

[4] Rosen-, B.
Function Optimization Based on Advanced Simulated Annealing
Disponível em www . ingber . com

[5] Robert , C. ; Casella , G.
Monte Carlo Statistical Methods – 2nd Edition
Springer-Verlag - 2004

[6] Oliveira Junior , H.A.
Lógica Difusa – Aspectos Práticos e Aplicações
Editora Interciência – 1999

[7] Oliveira Junior , H.A.
Fuzzy Control of Stochastic Global Optimization Algorithms and Very Fast Simulated Reannealing
Disponível em www . optimization-online . org