



Recozimento Simulado



www.pads.ufrj.br/~gabriel/cpe723



Otimização Natural – Parte II

Simulated Annealing



UFRJ / COPPE / PEE
José Gabriel R. C. Gomes
CPE723 – Aula 01

Planejamento das Aulas

1. Motivação / 01. Principais Desenvolvimentos / 02. Algoritmo de Metropolis Início L1
2. 03. Otimização por SA / 04. Exemplos / 05. SA Básico
3. 06. Análise de Convergência / 07. FSA / Entrega L1 Início L2
4. 08. Exemplos de Aplicações Práticas
5. 09. Deterministic Annealing / Entrega L2 Início L3
6. Exercícios e Dúvidas
7. Prova Teórica / Entrega L3



1. Principais Desenvolvimentos



Desenvolvimentos (I) – Início

- “Equações de estado para substâncias consistindo de moléculas individuais em interação.” (Metropolis et al., 1953).
- “Mostramos como o algoritmo de Metropolis (...) leva a uma ferramenta natural, baseada em mecânica estatística, para aplicação em otimização.” (Kirkpatrick et al., 1983).

Desenvolvimentos (II) – FSA

- “Se $T(k) \geq c / \log(1+k)$, então com probabilidade convergindo a 1 (à medida em que k tende a infinito), as configurações geradas pelo algoritmo serão aquelas de energia mínima.” (Geman e Geman, 1984).
- FSA (Szu e Hartley, 1987).

Desenvolvimentos (III) – DA

- “Aproveitar o melhor dos dois mundos: **determinístico** (sem vagar aleatoriamente pela superfície de energia fazendo progresso incremental na média) e **resfriamento** (evita a atração imediata por mínimos locais próximos). Reposição das simulações estocásticas pelo uso de ‘esperança’ $E[X]$.” (Rose, 1998).



2. Algoritmo de Metropolis



Metropolis (I) - Definições

- N partículas; temperatura T
- Estado $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \dots \ \mathbf{x}_N]$
- Função Energia $J(\mathbf{X})$ (ou Função 'Custo')
- Propriedade $F(\mathbf{X})$
- Propriedade Esperada:

$$\bar{F} = \frac{\int F e^{-\left(\frac{J}{kT}\right)} dv}{\int e^{-\left(\frac{J}{kT}\right)} dv}$$

Metropolis (II) – Monte Carlo

$$\bar{F} = \frac{\int F e^{-\left(\frac{J}{kT}\right)} dv}{\int e^{-\left(\frac{J}{kT}\right)} dv}$$

- Ex. avaliação pelo método de Monte Carlo:

$$\int_2^3 \frac{1}{x} \cdot \frac{1}{1} dx = \ln(x) \Big|_2^3 = \ln(3) - \ln(2) = 0.4055$$

```
>> X=rand(100000,1); X=X+2;  
>> sum(1./X)/100000 -> ans = 0.4054
```

Metropolis (III) – Pesos para $J(\mathbf{X})$

- Com alta probabilidade: $w(\mathbf{X}) = \exp\left(\frac{-J}{kT}\right) \simeq 0$
- Solução deste problema: “sortear configurações $\mathbf{X}$ com probabilidade $p = w(\mathbf{X})$, e dar a todas elas o mesmo peso”.
- Para isso: $\hat{\mathbf{X}}_{k+1} = \mathbf{X}_k + \epsilon \mathbf{R}$, com $\mathbf{X}_0$ qualquer
$$\Delta J = J(\hat{\mathbf{X}}_{k+1}) - J(\mathbf{X}_k)$$

Metropolis (IV) – Transições

$$1) \quad q = \exp \left(\frac{-\Delta J}{kT} \right)$$

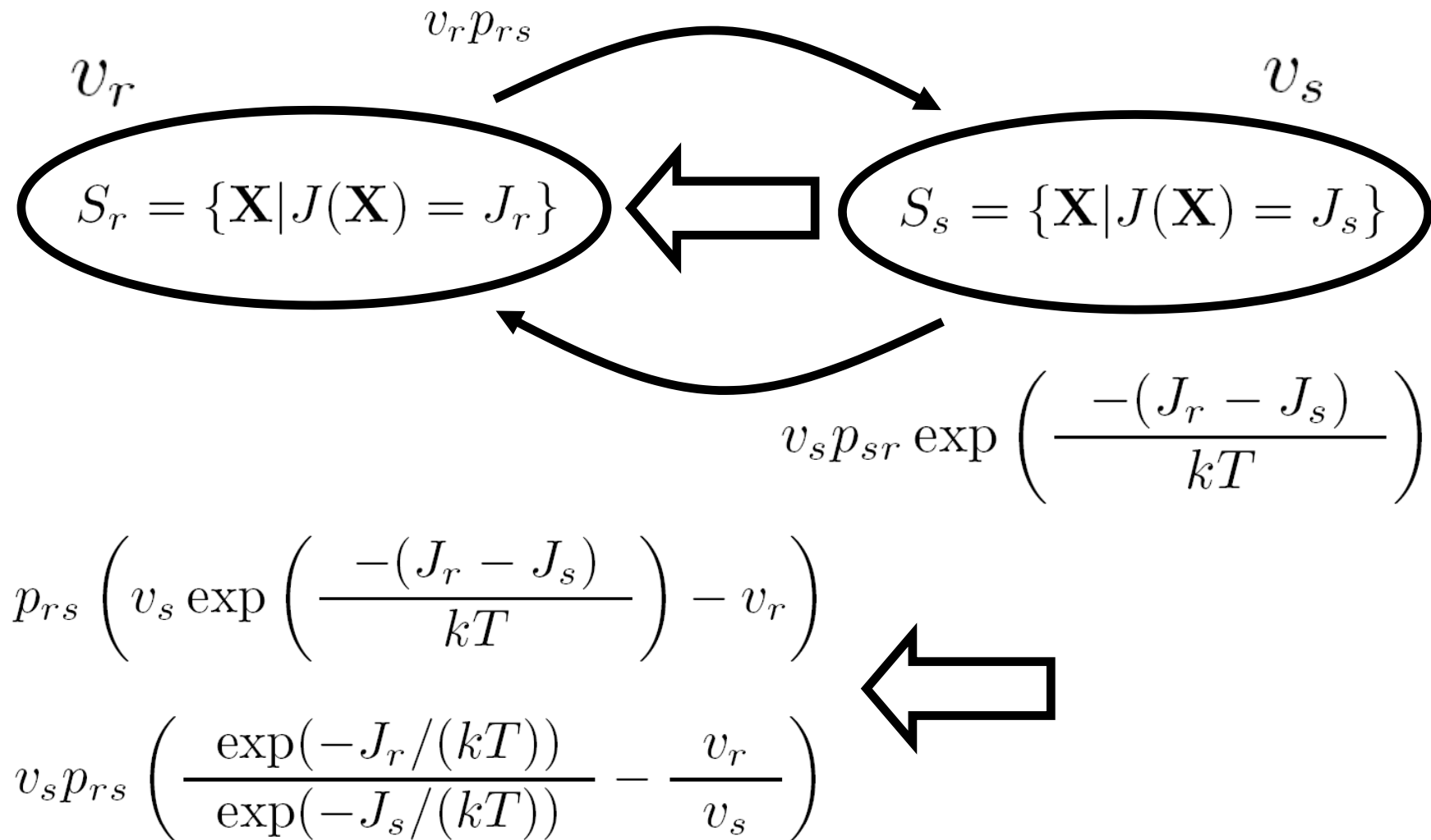
$$2) \quad r \sim U(0, 1) \quad (!)$$

$$3) \quad a = \begin{cases} 0, & \text{se } r > q \\ 1, & \text{se } r < q. \end{cases}$$

$$4) \quad \mathbf{X}_{k+1} = \begin{cases} \hat{\mathbf{X}}_{k+1}, & \text{se } \Delta J < 0 \\ (1 - a)\mathbf{X}_k + a\hat{\mathbf{X}}_{k+1}, & \text{se } \Delta J \geq 0. \end{cases}$$

$$5) \quad \bar{F} \simeq \hat{F} = \frac{1}{M} \sum_{j=1}^M F(\mathbf{X}_k)$$

Metropolis (V) – Convergência





3. Desenvolvimento do SA



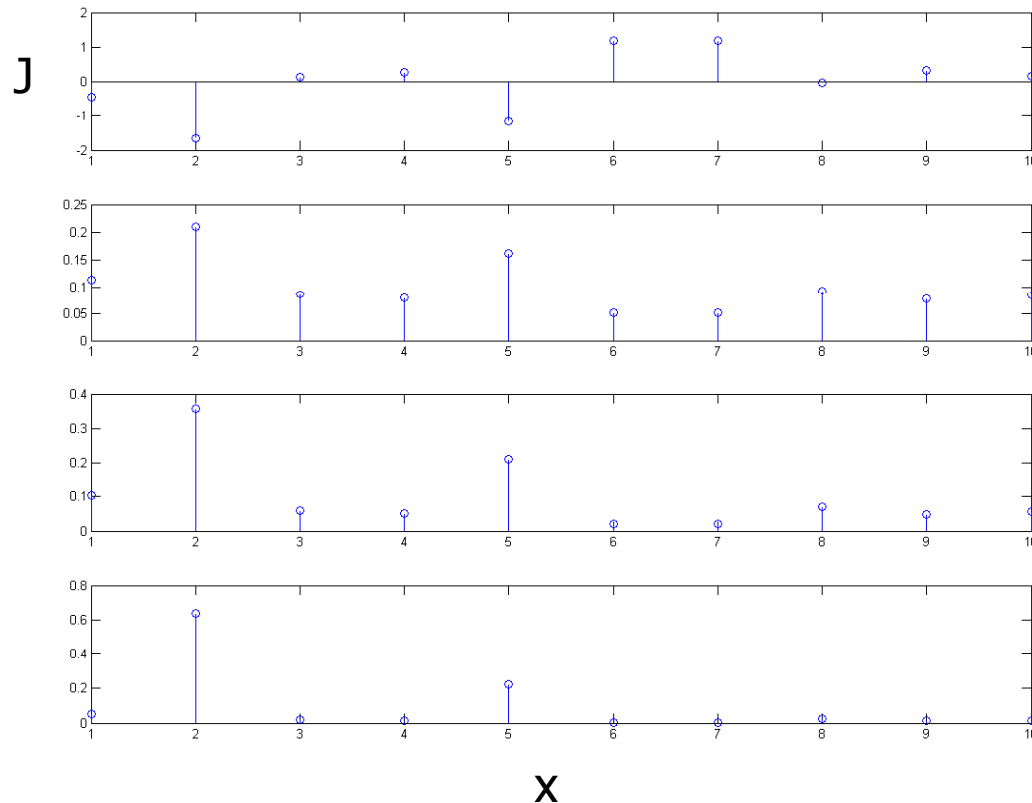
Desenvolvimento do SA

- Mecânica Estatística
- Fator de Boltzmann ($\propto$ Distribuição Boltzmann)

$$e^{-\left(\frac{J}{kT}\right)}$$

- Encontrar 'estados' em temperatura baixa

Fator de Boltzmann – variação com T



$\exp(-J/T)$

T=2

T=1

T=0.5

```
>> randn('state',0)
>> J=randn(10,1);
>> subplot(4,1,1); stem(J);
>> subplot(4,1,2); p = exp(-J/2); stem(p/sum(p));
>> subplot(4,1,3); p = exp(-J); stem(p/sum(p));
>> subplot(4,1,4); p = exp(-2*J); stem(p/sum(p));
```


Baixo T não é suficiente

- Resfriamento deve ser cuidadoso
 - Gastar tempo em temperaturas ≈ 0
 - Do contrário, mínimos locais
 - $T = 0 \rightarrow$ método “gradiente” (mínimos locais)
- Metropolis “heat bath”
 - Função custo no lugar da energia: $J(X) = W(X)$
 - Distribuição estacionária
- Reduzir T ; adaptive “divide-and-conquer”



4. Exemplos

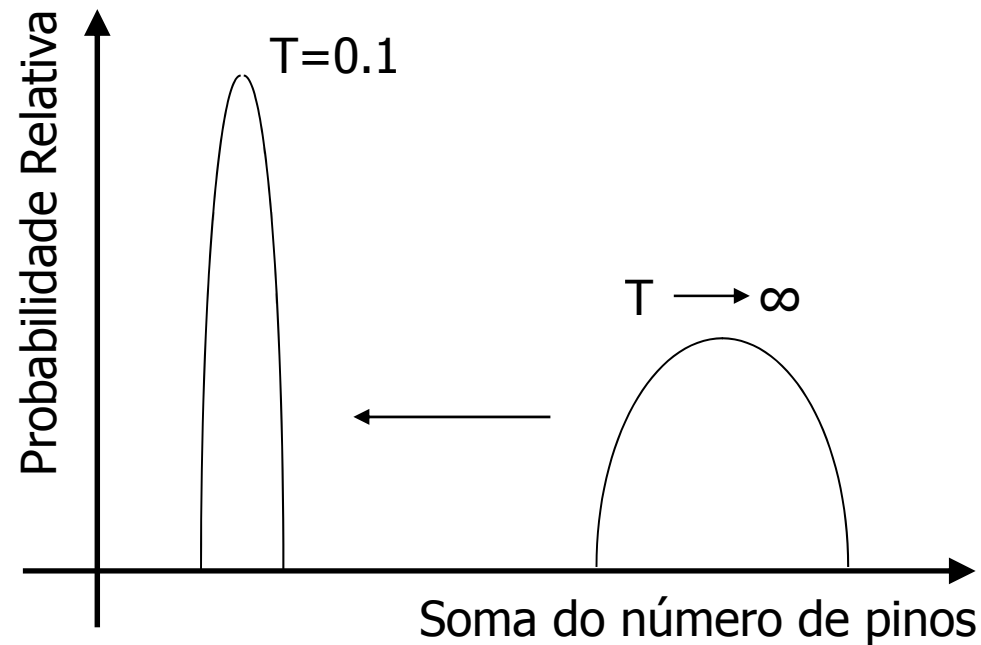


4.1. Projeto de Computadores

- Especificação da lógica do sistema
- Realização física
- Parte 4.1.1: divisão dos circuitos em grupos
- Parte 4.1.2: posicionamento dos circuitos dentro dos grupos
- Parte 4.1.3: roteamento
- Matriz de conectividade

4.1.1. Particionamento

- Ver Figura 1, Kirkpatrick et al., 1983
- 5000 portas lógicas; $T_n = (0.9)^n T_0$



4.1.2. Posicionamento

- Localização física dos circuitos
- Minimização de distâncias
- Grid com 10 x 10 possíveis posições para os chips
- Histograma de cruzamento
- Clusters naturais determinados com adiantamento

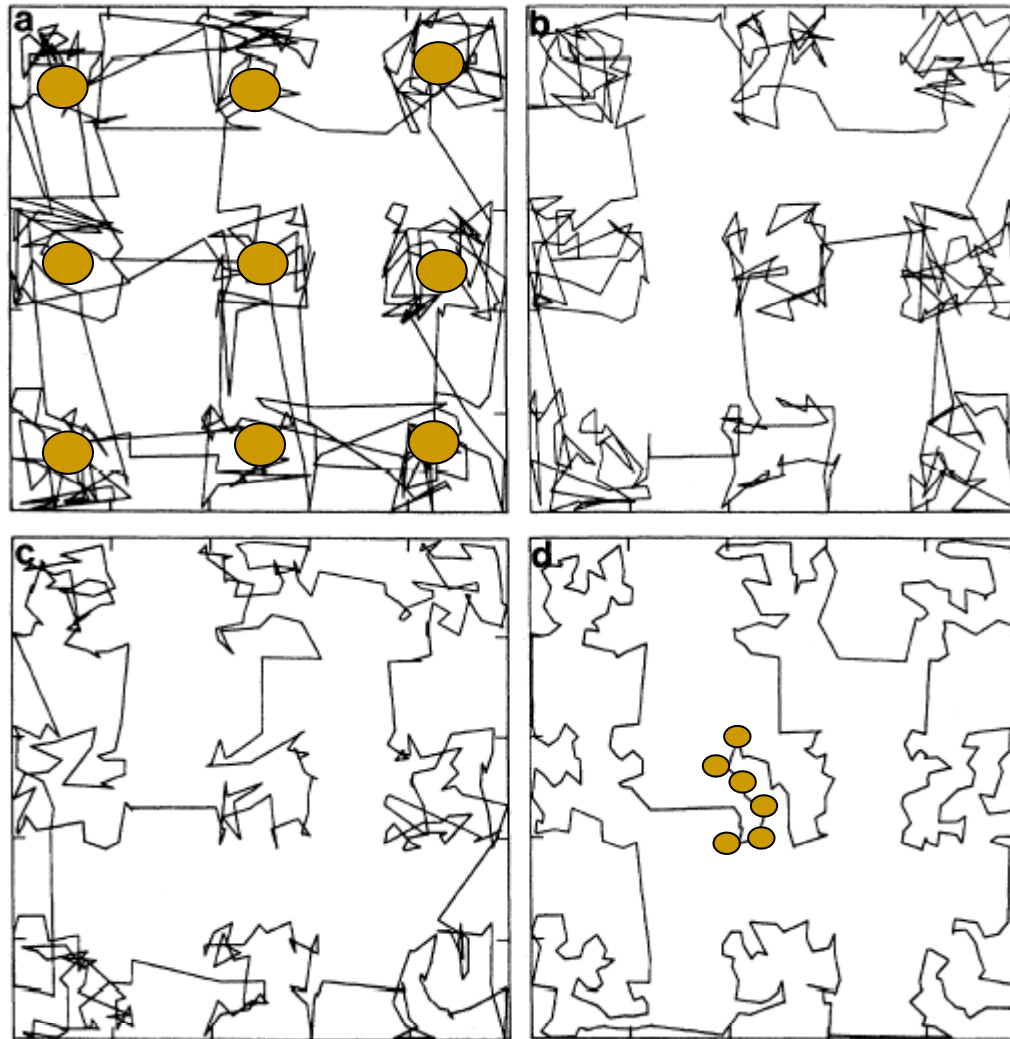
4.1.3. Roteamento

- Roteamento aproximado
- Roteamento fino
- Minimizar comprimento de fio e número de dobras
- Definição das funções custo apropriadas
- Resultados comparativos

4.2. Caixeiro Viajante

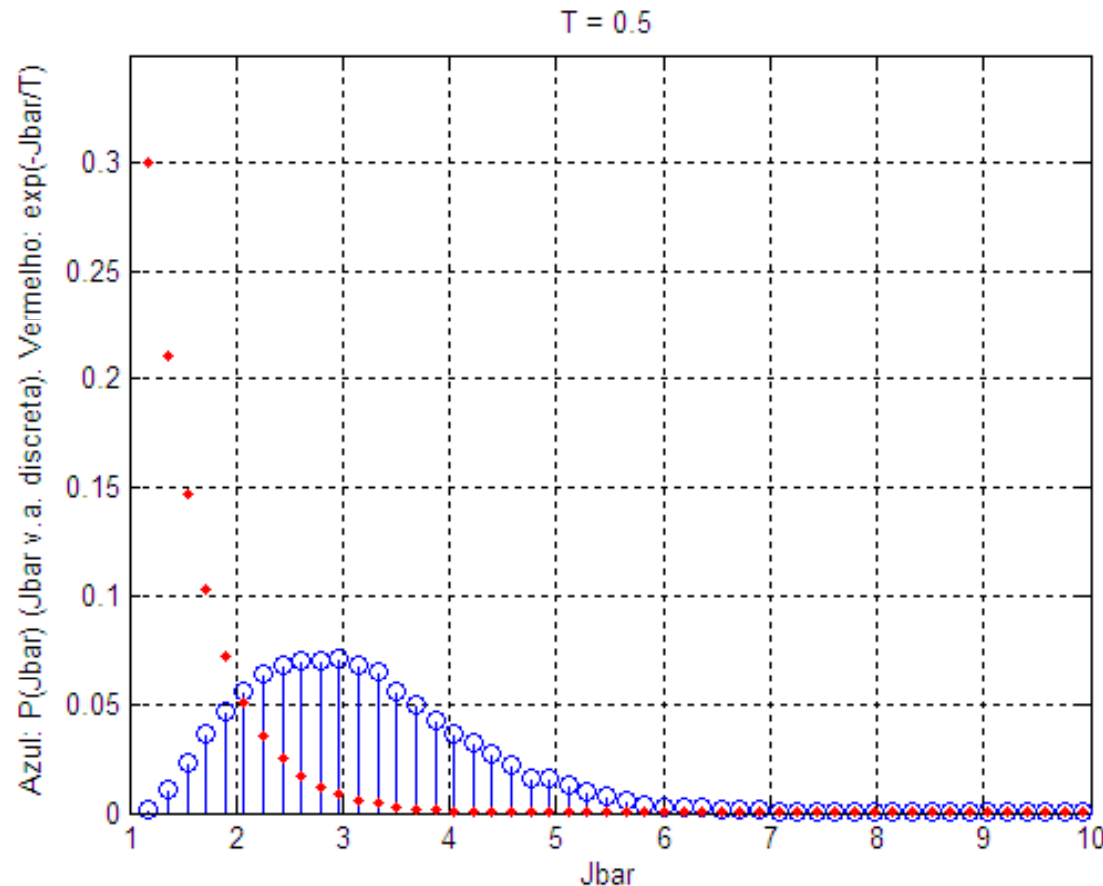
- Cidades: K
- Localização das cidades: c_i ($i = 1, \dots, K$)
- Ordem das visitas: $X = [i_1 \ i_2 \ i_3 \ \dots \ i_K]$
- Otimização Combinatorial (NP-completo)
- K não maior do que 1000
- Métodos: “divide/conquer”, “iterative improvement”
- Performance média dos algoritmos

4.2.1. Caixeiro Viajante



4.3. Cinco 'Partículas' (J Contínuo (!))

- Executar Exemplo #1 ($T = 0.5, 0.1, 0.01, 0.001$)



Resumo

- Descrição simples do estado X
- Função custo
- Gerador de movimentos aleatórios
- “Annealing schedule”: programação das reduções de temperatura (escalas em que o problema é avaliado).

Ver (www.pads.ufrj.br/~gabriel/cpe723):

- Aula #01 (PDF)
- Metropolis et al., 1953 (PDF) (págs 1087 a 1089)
- Kirkpatrick et al., 1983 (PDF)
- [Lista de Exercícios #1](#) (PDF)
- Exemplo #1 (M-File)

Próxima Aula

- Descrição do Simulated Annealing Básico



Simulated Annealing (Algoritmo Básico)



UFRJ / COPPE / PEE
José Gabriel R. C. Gomes
CPE723 – Aula 02



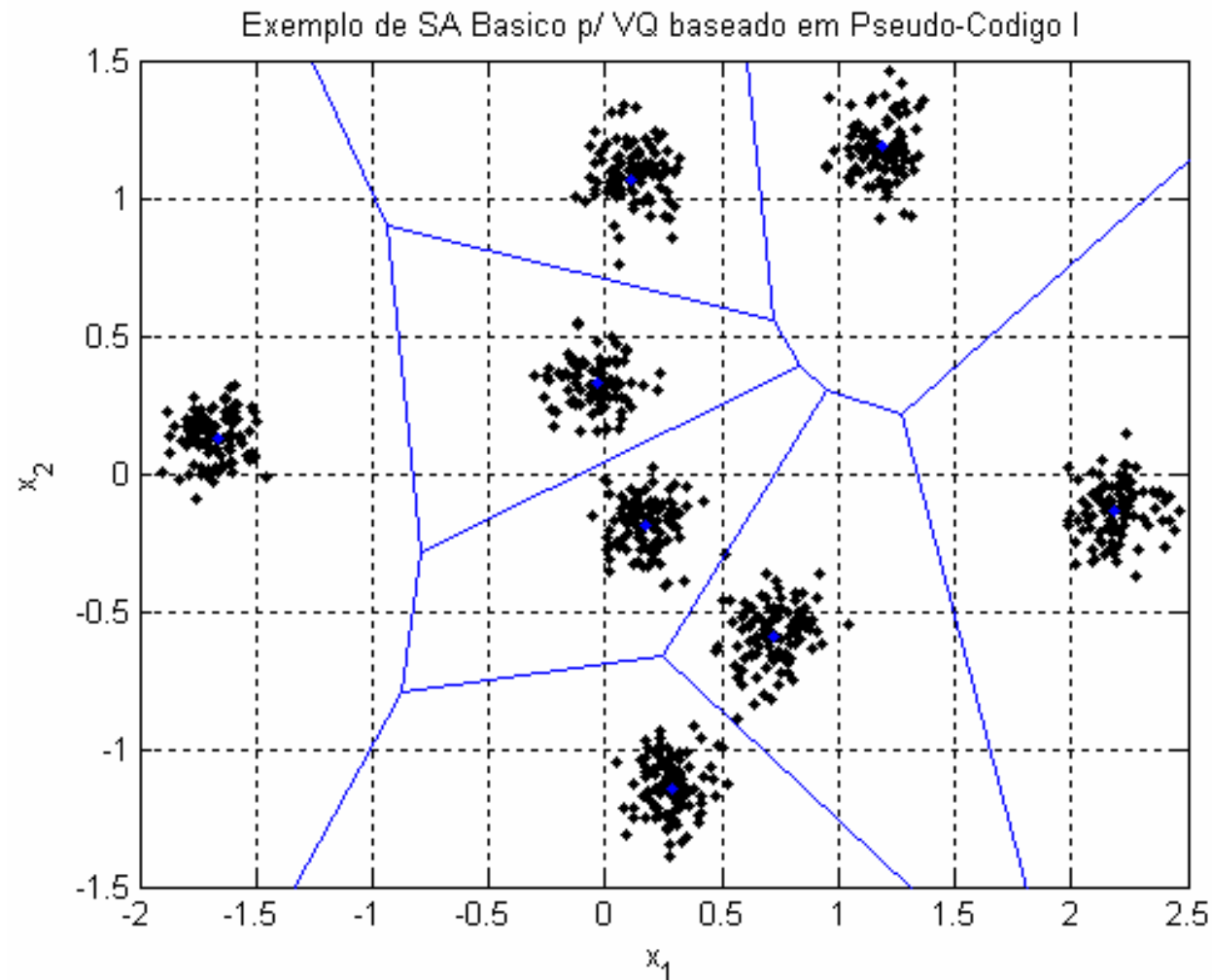
5. SA Básico



Pseudo-Código

- X_0 aleatório; $J_0 = J(X_0)$; $X_{\text{atual}} = X_0$; $J_{\text{atual}} = J_0$;
- Definições: $N = (10000)$; $K = (8)$; $T_0 = (5e-3)$; $\varepsilon = (5e-2)$;
- $\text{fim} = 0$; $n = 0$; $k = 1$; $J_{\text{min}} = J_{\text{atual}}$; $X_{\text{min}} = X_{\text{atual}}$;
- while not(fim)
 - {
 - $n = n + 1$;
 - $X = X_{\text{atual}} + \varepsilon \times \text{randn}(\text{size}(X))$;
 - $J = J(X)$;
 - if $\text{rand}(1) < \exp((J_{\text{atual}} - J)/T)$ { $X_{\text{atual}} = X$; $J_{\text{atual}} = J$;};
 - if $J < J_{\text{min}}$ { $J_{\text{min}} = J$; $X_{\text{min}} = X$;};
 - if $\text{rem}(n, N) = 0$ { $k = k + 1$; $T = T_0 / \log_2(1 + k)$; if $k = K$, { $\text{fim} = 1$;}}}
 - }

Exemplo no Matlab – Dados



Exemplo no Matlab – Código

% Gerando Dados Novamente

```
clear all; K = 8; P = 100; randn('state',0); a = randn(1,1); rand('state',0);  
X0 = randn(2,K); X=[]; for k=1:size(X0,2), X = [X repmat(X0(:,k),1,P)+randn(2,P)*0.1]; end;
```

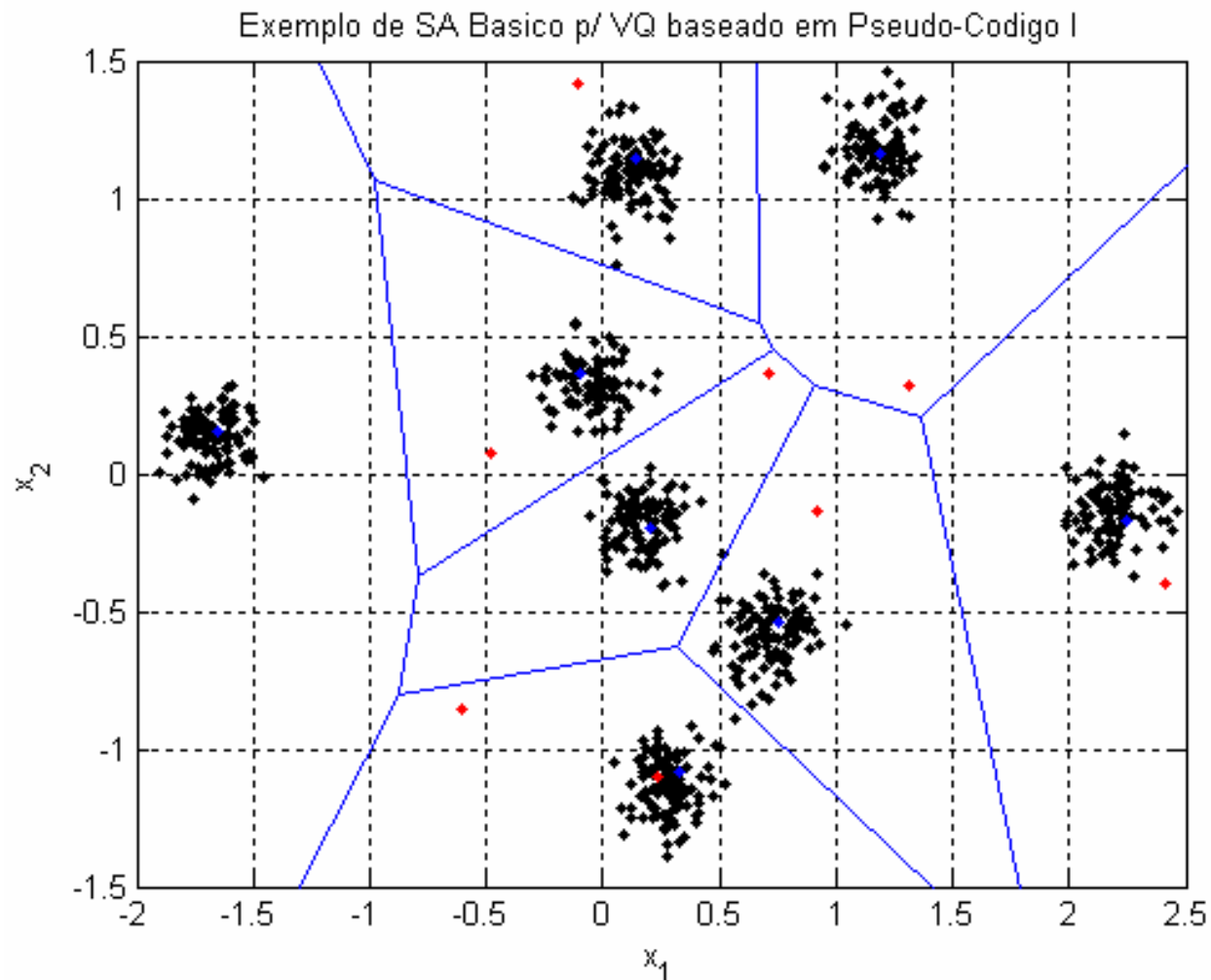
% Metodo 2: Simulated Annealing

```
C0 = randn(2,K); [Cn,p,D0] = ExemploFuncaoJ(X,C0); Ci = C0; Di = D0/size(X,2);  
e = 5e-2; t = 1; T0 = 5e-3; T = T0/log2(1+t); N = 10000; L = 8;  
n = 0; fim = 0; Df = Di; Cf = Ci; BKD = zeros(N*(L-1),2);
```

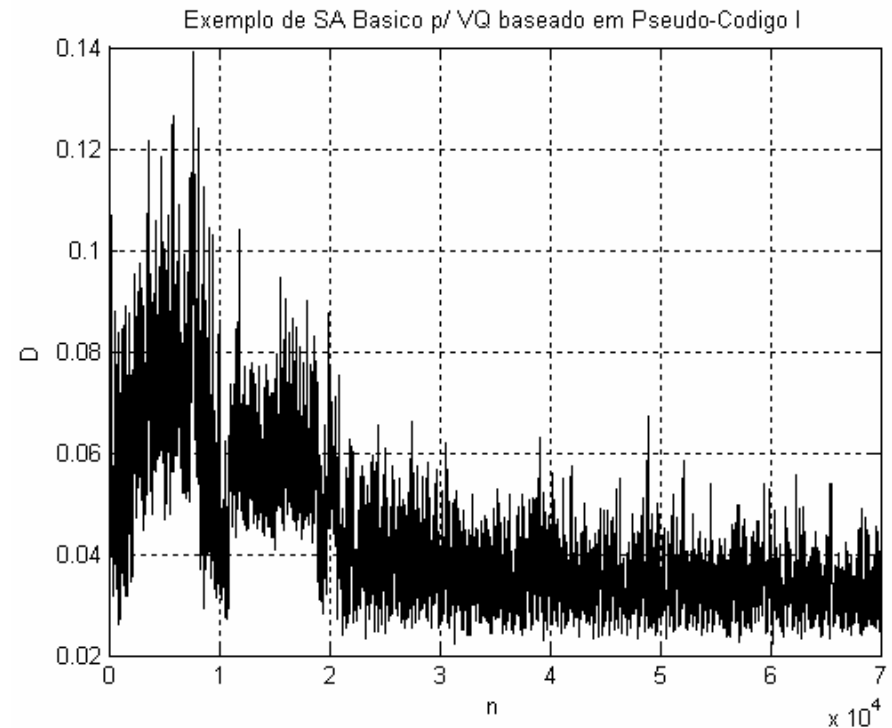
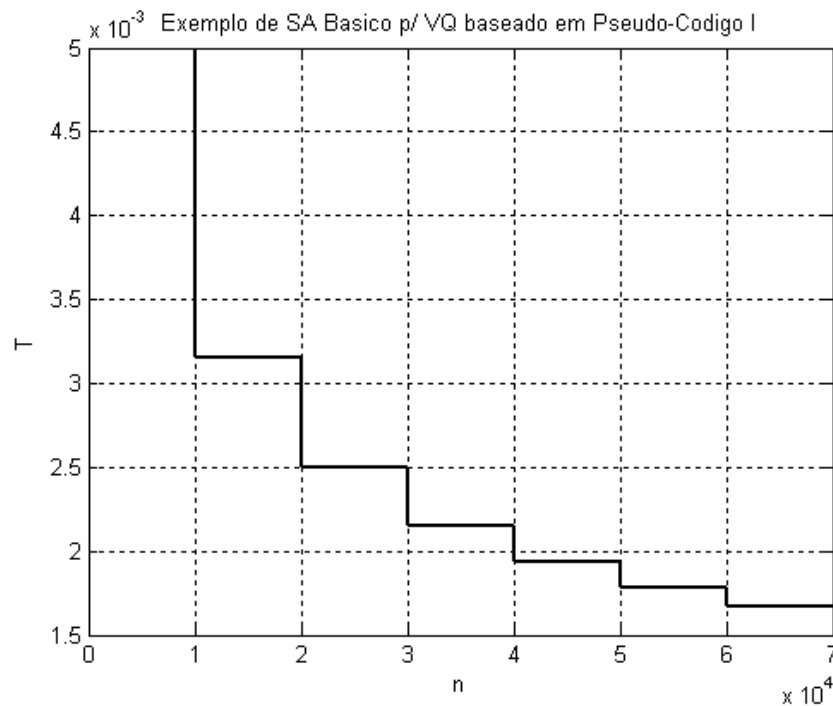
```
while not(fim),  
    n = n + 1; BKD(n,:) = [T Di];  
    C = Ci + e*randn(size(Ci));  
    [Cn,p,D] = ExemploFuncaoJ(X,C); D = D/size(X,2); % Notacao: D = D(C) ao inves de J = J(X)!  
    if D<Df, Df = D; Cf = C; end;  
    if D<Di,  
        Ci = C; Di = D;  
    else  
        a = rand(1,1);  
        if a < exp((Di-D)/T), Ci = C; Di = D; end;  
    end;  
    if (rem(n,N)==0),  
        t = t+1;  
        T = T0/log2(1+t);  
        if t==L, fim = 1; end;  
    end;  
end;  
C = Cf;
```

```
figure; plot(X(1,:),X(2,:), 'k.', C(1,:),C(2,:), 'b. '); grid on; hold on; voronoi(C(1,:),C(2,:));  
plot(C0(1,:),C0(2,:), 'r. '); Df % D = 0.0223
```

Exemplo no Matlab – Resultado



Exemplo no Matlab – $T(n)$ e $J(n)$ (BKD)



Pseudo-Código II (Alternativo)

- Set $T = T_{max}$ and $\mathcal{C} = \mathcal{C}_{init}$
- $t = 0$.
- Repeat until a predetermined value T_{min} for T is reached:
 - $t = t + 1$.
 - Repeat until an equilibrium state is reached at this temperature:
 - * Compute $J(\mathcal{C})$.
 - * Produce a new clustering, $\mathcal{C}'$, by assigning a randomly chosen vector from X to a different cluster.
 - * Compute $J(\mathcal{C}')$.
 - * $\Delta J = J(\mathcal{C}) - J(\mathcal{C}')$.
 - * If $(\Delta J < 0)$ then $(\mathcal{C} = \mathcal{C}')$ else $(\mathcal{C} = \mathcal{C}'$ with probability $P(\Delta J) = \exp(-\Delta J/T)$).
 - $T = f(T_{max}, t)$.

(*) from Theodoridis and Koutroumbas, 2003.

Considerações Finais

- Ajuste de quatro parâmetros: T_0 , ε , N , e K .
- Longo tempo de convergência
- Obtenção de mínimo global

Considerações Finais

- Implementação eficiente da função custo (DLL)
 - Opção 1) Tutorial MexFilesWithVisualC.doc – disponível no site
 - Opção 2) `mex -setup`
`mex ExemploFuncaoJ.cpp`
(obtém-se 'ExemploFuncaoJ.mexw32')

Próxima Aula

- Análise teórica do SA (convergência)
- Ler: Geman and Geman, 1984.
- Executar Exemplo #2 (disponível no site)

Links Interessantes

- http://en.wikipedia.org/wiki/Simulated_annealing
Informação Geral
- <http://www.heatonresearch.com/articles/64/page1.html>
Caixeiro Viajante: Exemplo de Solução
Código Fonte
- <http://www.ingber.com/#ASA>
Adaptive Simulated Annealing



Simulated Annealing – Análise da Convergência



UFRJ / COPPE / PEE
José Gabriel R. C. Gomes
CPE723 – Aula 03



6. Geman, 1984



Motivação dos Autores (até VIII)

- Restauração de Imagens $X = (F, L)$
- Imagem degradada: $G = \Phi(H(F)).N$
- Estimar $x = (f, l)$ (amostras de F e L)
- Grafos, $\pi(\omega)$, $P(X = \omega \mid g)$

Definições de Nosso Interesse

1. Sites: $S = \{s_1, s_2, \dots, s_N\}$
2. Valores possíveis: $\Lambda = \{0, 1, \dots, L\}$
3. Espaço de configurações (estados): $\Omega = \Lambda^N$
4. Ordem das visitas: $\{n_1, n_2, \dots\} \subset S$
5. Dist. de Gibbs: $\pi(\omega) = (1/Z) \exp(-J(\omega)/T)$
6. Faixa de custos: $\Delta = J_{max} - J_{min}$
7. Estado (configuração) inicial: $X(0)$
8. Processo de Markov: $\{X(t), t = 0, 1, 2, \dots\}$

Matriz de Transição do Proc. $X(t)$

$$(M_t)_{\eta, \omega} = \begin{cases} \pi(X_{n_t} = x_{n_t} | X_s = x_s, s \neq n_t), & \text{se } \eta = \omega^x \text{ para algum } x \in \Lambda; \\ 0, & \text{caso contrário.} \end{cases}$$

- Transição: $\mu(\omega) \leftrightarrow P(X(t) = \omega): \mu = \left(\prod_{j=1}^{j=t} M_j \right) \mu_0$

- $\pi(\omega)$ é vetor invariante de M_t :

$$M_t \pi = \pi$$

Ilustração Teorema A (Exemplo)

```
% 070331 gabriel@pads.ufrj.br

close all; clear all; rand('state',0);
xmin = ones(4,1);
x = floor(rand(4,1)*4); x0 = x; xc = x;
J = mean((x-xmin).^2); J0 = J; Jc = J;
edges = [0 1 3 5 7 9 11 13 15 17 19 22 25 32]/8;
C = [0 0.25 0.5 0.75 1.00 1.25 1.50 1.75 2.00 2.25 2.50 3.00 3.25 4.00];
N = 200000; BKJ = zeros(N,1); BKa = zeros(N,1); T = 0.5;

for k=1:N,
    x = xc; a = floor(rand(1)*4)+1; b = ceil(rand(1)*3);
    x(a) = rem(x(a)+b,4);
    J = mean((x-xmin).^2);
    if rand(1) < exp((Jc-J)/T),
        xc = x; Jc = J;
    end;
    BKJ(k) = Jc; BKa(k) = a;
end;
S = BKJ(ceil(0.5*N):N); H1 = histc(S,edges)/length(S);
figure; subplot(2,1,1); stem(C,H1); grid on;
```

Ilustração Teorema A

```
k=0; Z=0;
for a=0:3,
    for b=0:3,
        for c=0:3,
            for d=0:3,
                k = k+1; x = [a b c d]'; Jg(k) = mean((x-xmin).^2);
                Z = Z+(exp(-Jg(k)/T)); i(k) = 64*a+16*b+4*c+d+1;
            end;
        end;
    end;
end;
H2 = histc(Jg,edges)/length(Jg); hold on; subplot(2,1,1); stem(C,H2,'r-');
% w = sort(Jg); plot(w,exp(-w/T),'k. ');
subplot(2,1,2); plot(C,H1./(H2'*256),'bo-'); hold on;
subplot(2,1,2); plot(C,(1/Z)*exp(-C/T),'ro:'); grid on;
```

Ilustração Teorema A

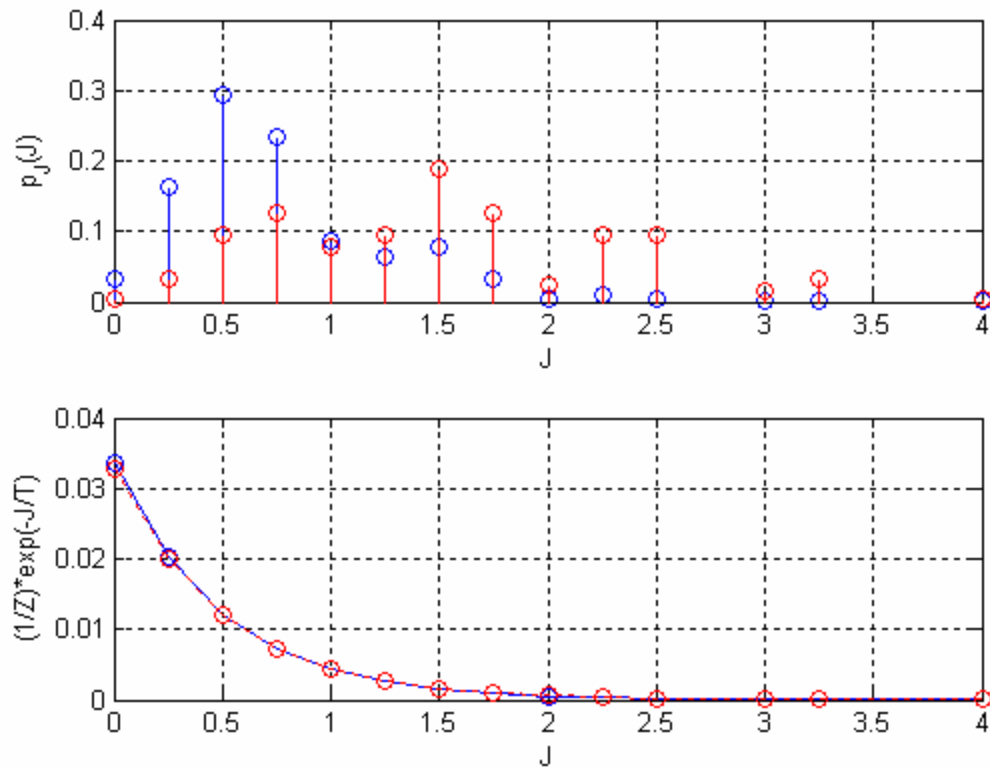


Ilustração Teorema A

```
p=(exp(-Jg/T)/Z); % Invariant probability vector

% Markov Chain Transition Matrices (M1)

M1 = zeros(256,256);
for a=0:3,
    for b=0:3,
        for c=0:3,
            for d=0:3,
                j = 64*a+16*b+4*c+d+1;
                x = [a b c d]'; J = mean((x-xmin).^2);
                for n=1:3,
                    a2 = rem(a+n,4); i = 64*a2+16*b+4*c+d+1;
                    x2 = [a2 b c d]'; J2 = mean((x2-xmin).^2);
                    DeltaJ = J-J2;
                    if DeltaJ>0,
                        M1(i,j) = 1/3;
                    else
                        M1(i,j) = 1/3*exp(DeltaJ/T);
                        M1(j,j) = M1(j,j) + 1/3*(1-exp(DeltaJ/T));
                    end;
                end;
            end;
        end;
    end;
end;
p1 = M1*p'; figure; imshow(M1);
```

Ilustração Teorema A

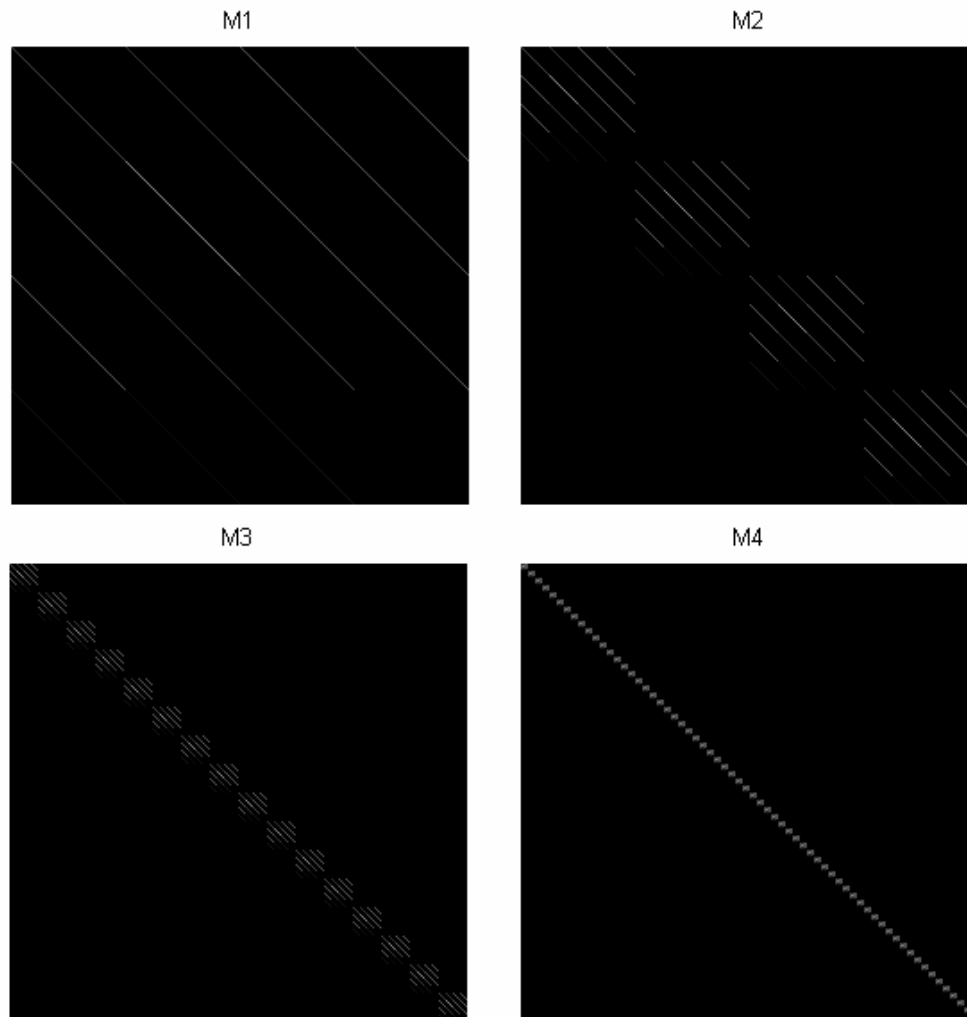


Ilustração Teorema A

```
% M2
```

```
M2 = zeros(256,256);  
for a=0:3,  
    for b=0:3,  
        for c=0:3,  
            for d=0:3,  
                j = 64*a+16*b+4*c+d+1;  
                x = [a b c d]'; J = mean((x-xmin).^2);  
                for n=1:3,  
                    b2 = rem(b+n,4); i = 64*a+16*b2+4*c+d+1;  
                    x2 = [a b2 c d]'; J2 = mean((x2-xmin).^2);  
                    DeltaJ = J-J2;  
                    if DeltaJ>0,  
                        M2(i,j) = 1/3;  
                    else  
                        M2(i,j) = 1/3*exp(DeltaJ/T);  
                        M2(j,j) = M2(j,j) + 1/3*(1-exp(DeltaJ/T));  
                    end;  
                end;  
            end;  
        end;  
    end;  
end;  
p2 = M2*p';  
figure; imshow(M2);
```

Ilustração Teorema A

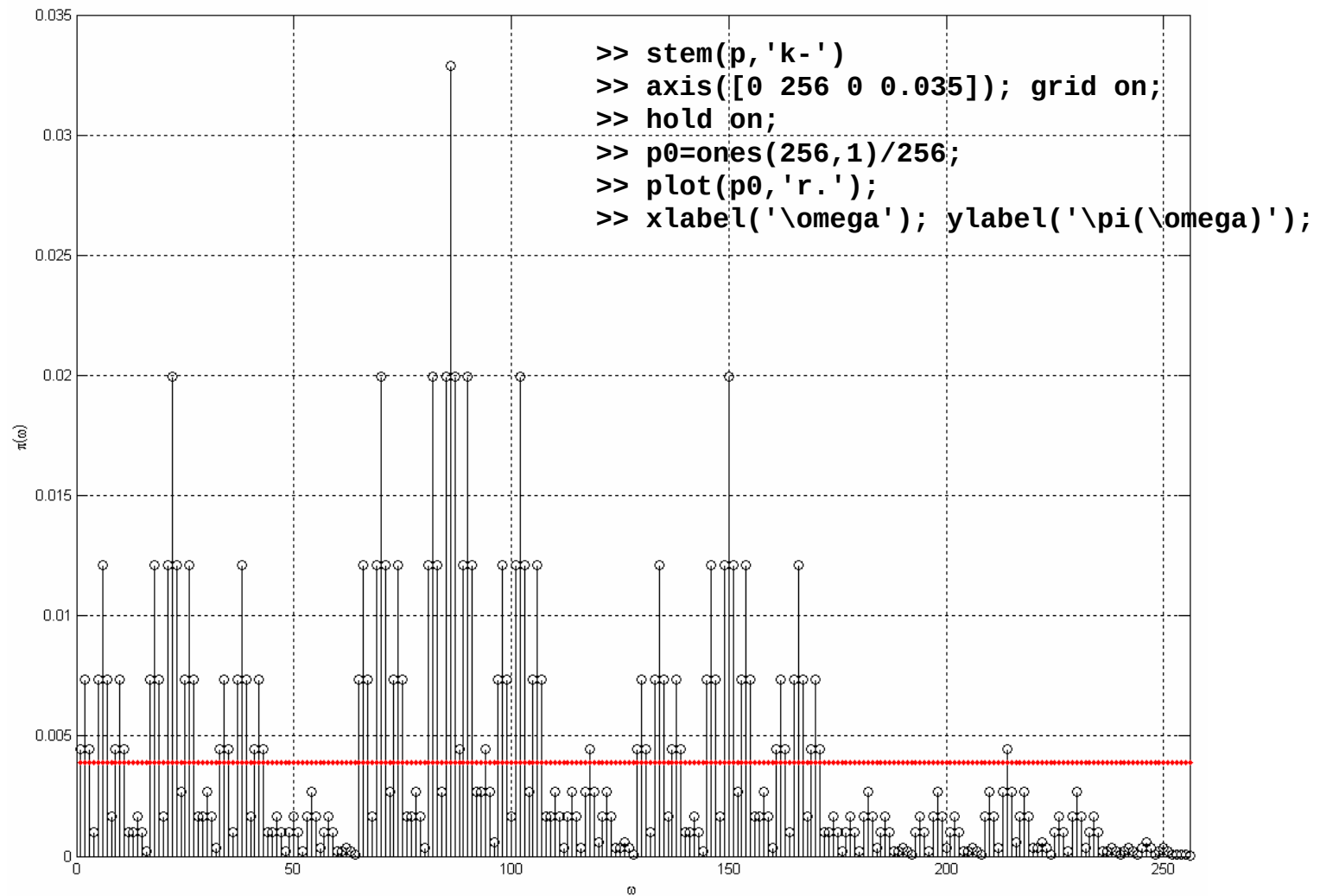
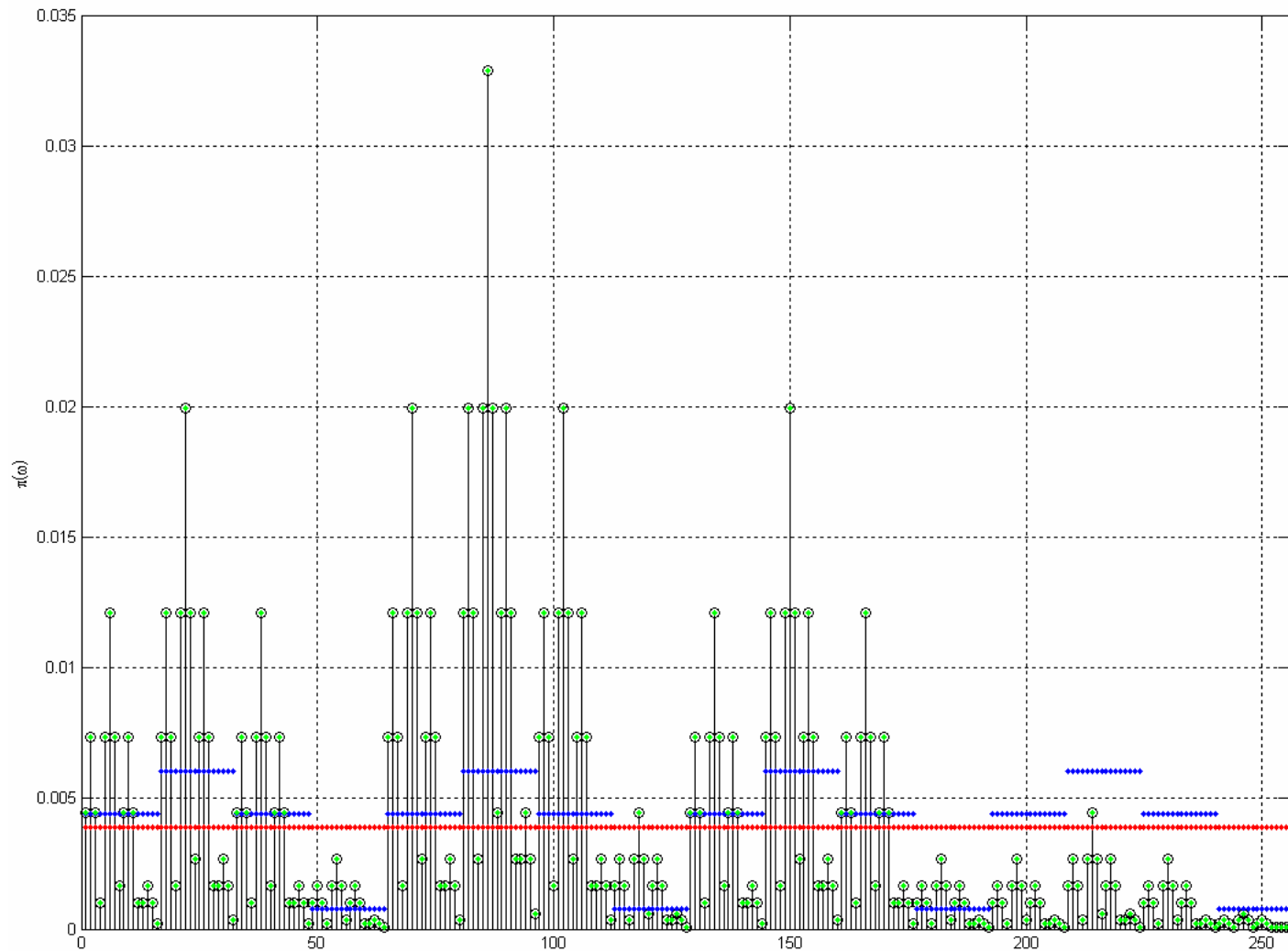


Ilustração Teorema A

```
>> p0 = M2*p0; plot(p0, 'b. ');  
>> pf = M4*p'; plot(pf, 'g. ');
```



Teorema A (“Relaxation”)

$$\pi(\omega) = \frac{1}{Z} e^{-J(\omega)/T}$$

Teorema A: Assuma que, para todo $s \in S$, a seqüência $\{n_t, t \geq 1\}$ contenha s *infinitamente freqüentemente*. Então, para qualquer estado inicial $\eta \in \Omega$ e para qualquer $\omega \in \Omega$, temos:

$$\lim_{t \rightarrow \infty} P(X(t) = \omega | X(0) = \eta) = \pi(\omega).$$

Demonstração Teorema A

- Temperatura fixa (T)
- Visitas completas em $T_0 < T_1 < T_2 < \dots$
- $K(t) = \sup \{ k : T_k < t \}$

Lema 1: Existe uma constante r , $0 \leq r < 1$, tal que para todo t ($t = 1, 2, \dots$) temos:

$$\sup_{\omega, \eta', \eta''} |P(X(t) = \omega | X(0) = \eta') - P(X(t) = \omega | X(0) = \eta'')| \leq r^{K(t)}.$$

- (Lema 1) + ($M_t \pi = \pi$) = Teorema A

Lema 1

- Consiste em calcular o valor: $r = 1 - L^N \delta^N$
- Etapas:
 1. Características locais
 2. Separação “sup” – “inf”
 3. Atribuir probabilidades

Teorema B (“Annealing”)

Teorema B: Assuma que exista um inteiro $\tau \geq N$ tal que para todo t ($t = 0, 1, 2, \dots$) tenhamos:

$$S \subseteq \{n_{t+1}, n_{t+2}, \dots, n_{t+\tau}\}.$$

Seja $T(t)$ qualquer seqüência de temperaturas decrescentes tal que:

- a) $T(t) \rightarrow 0$ se $t \rightarrow \infty$;
- b) $T(t) \geq N\Delta / \log t$ para todo $t \geq t_0$, sendo t_0 um número inteiro maior ou igual a 2.

Então, para qualquer estado inicial $\eta \in \Omega$ e para qualquer $\omega \in \Omega$, temos:

$$\lim_{t \rightarrow \infty} P(X(t) = \omega | X(0) = \eta) = \pi_0(\omega).$$

Demonstração Teorema B

Lema 2: Para todo $t_0 = 0, 1, 2, \dots$

$$\lim_{t \rightarrow \infty} \sup_{\omega, \eta', \eta''} |P(X(t) = \omega | X(0) = \eta') - P(X(t) = \omega | X(0) = \eta'')| = 0.$$

Lema 3: $\lim_{t_0 \rightarrow \infty} \sup_{t \geq t_0} \|P(t, \cdot | t_0, \pi_0) - \pi_0\| = 0.$

- Lema 2: semelhante ao Lema 1, só que com $T(t) \rightarrow 0$ e $T(t) > N\Delta / \log t$.
- (Lema 2) + (Lema 3) = Teorema B

Lema 3

Seja $|\Omega_0|$ igual ao número de elementos no conjunto Ω_0 .
Note que $\|\pi_0 - \pi_{T(t)}\| \rightarrow 0$ à medida em que $t \rightarrow 0$:

$$\begin{aligned}\pi_{T(t)}(\omega) &= \frac{e^{-J(\omega)/T(t)}}{\sum_{\omega' \in \Omega_0} e^{-J(\omega')/T(t)} + \sum_{\omega' \notin \Omega_0} e^{-J(\omega')/T(t)}} \\ &= \frac{e^{-(J(\omega) - J_{\min})/T(t)}}{|\Omega_0| + \sum_{\omega' \notin \Omega_0} e^{-(J(\omega') - J_{\min})/T(t)}} \\ &\xrightarrow{t \rightarrow \infty} \begin{cases} 0, & \text{se } \omega \notin \Omega_0; \\ 1/|\Omega_0|, & \text{se } \omega \in \Omega_0. \end{cases}\end{aligned}$$

Conclusões

- À medida em que o algoritmo converge, a distribuição $P(X(t) = \omega)$ torna-se uniforme dentro de Ω_0 , onde $\omega \in \Omega_0 \rightarrow J(\omega) = J_{\min}$.
- Para que isto ocorra, é necessário que $T(t) > N\Delta / \log t$. (Lema 2)



7. FSA (Szu, 1987)



Idéia Básica

- SA:

Elementos de R: $f_R(r) = \sqrt{1/2\pi} \exp(-r^2/2)$

Resfriamento: $T_0 / \log(1+t)$

- FSA:

Elementos de R: $f_R(r) = c / (r^2 + c^2)^{(D+1)/2}$

Resfriamento: $T_0 / (1+t)$

Próxima Aula

- Aplicações práticas do SA
- Ler:
 1. Geman, 1984 (X, XI, XII, A)
 2. Szu, 1987 (a) e (b)
- Executar Exemplo #3 (disponível no site)



Simulated Annealing – Exemplos de Aplicação



UFRJ / COPPE / PEE
José Gabriel R. C. Gomes
CPE723 – Aula 04

Algumas Aplicações Práticas

- Problemas de complexidade elevada
- Fast Adaptive Simulated Annealing
- Criptografia
- Projeto de Quantizadores Vetoriais



Aplicação de SA ao Projeto de Quantizadores Vetoriais



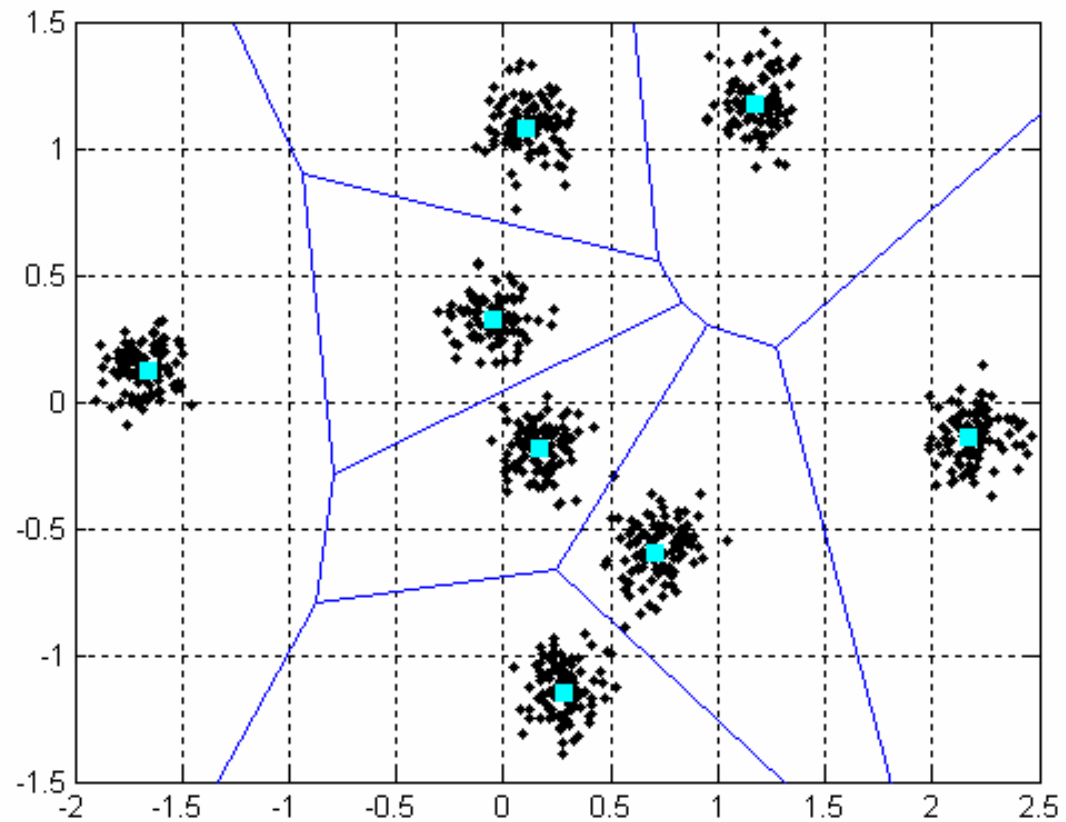
Introdução

- Vetores de Dados
- Reconstrução
- Taxa
- Distorção
- Generalized Lloyd Algorithm (GLA / LBG)
 - Condição da Partição
 - Condição do Centróide
- Implementação / Mínimos Locais

Exemplo4A.m – K=8 classes

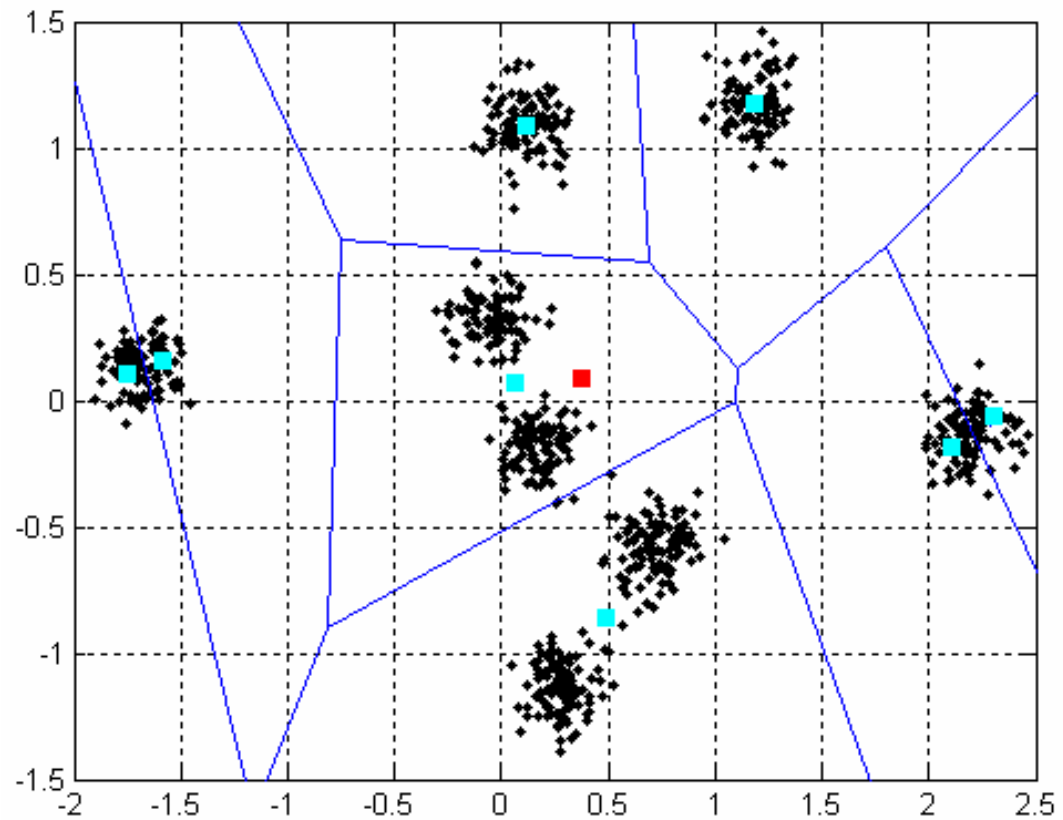
(Igual ao Exemplo2.m)

$$D_{\min} = 0.0195$$



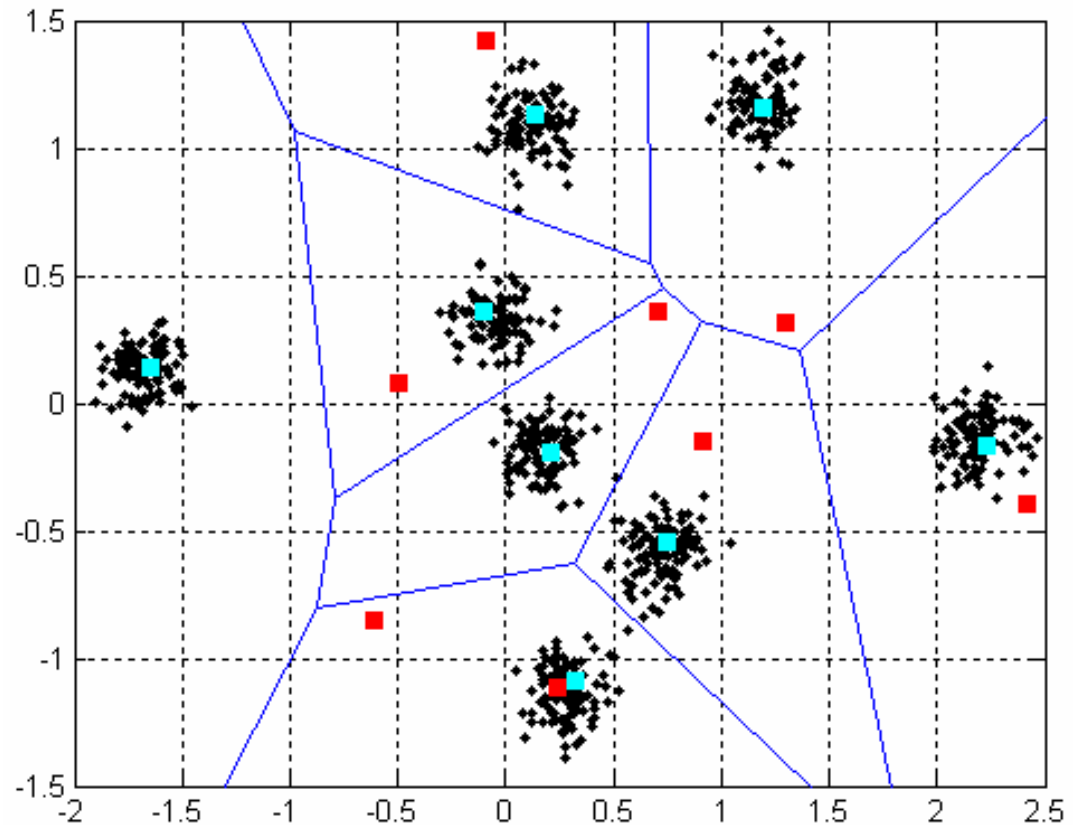
Exemplo4A.m – GLA

$$D_{\text{GLA}} = 0.0672$$

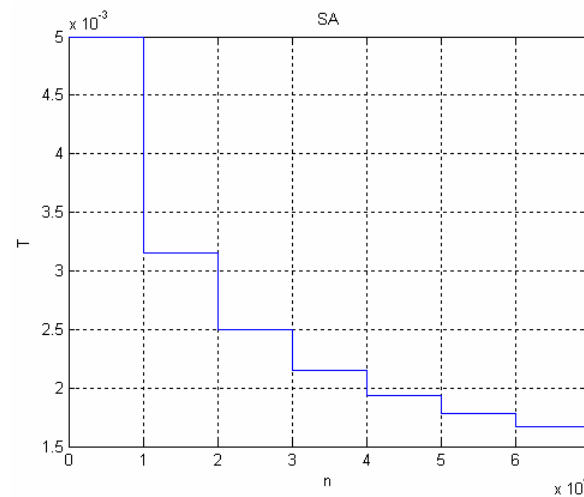
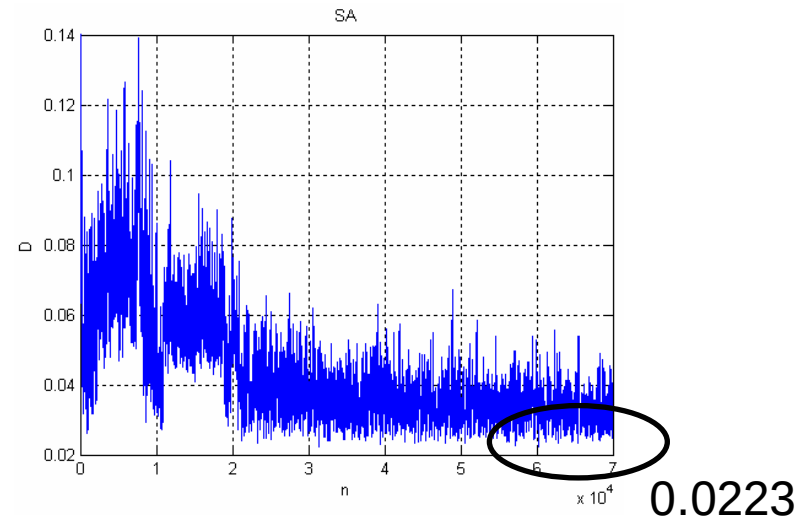
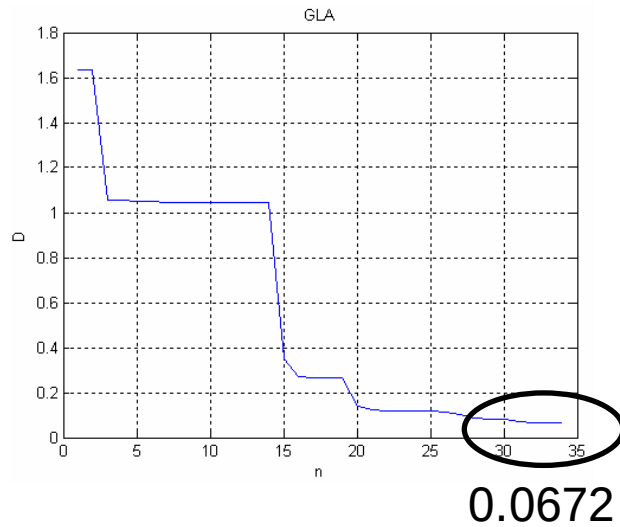


Exemplo4A.m – SA

$$D_{SA} = 0.0223$$

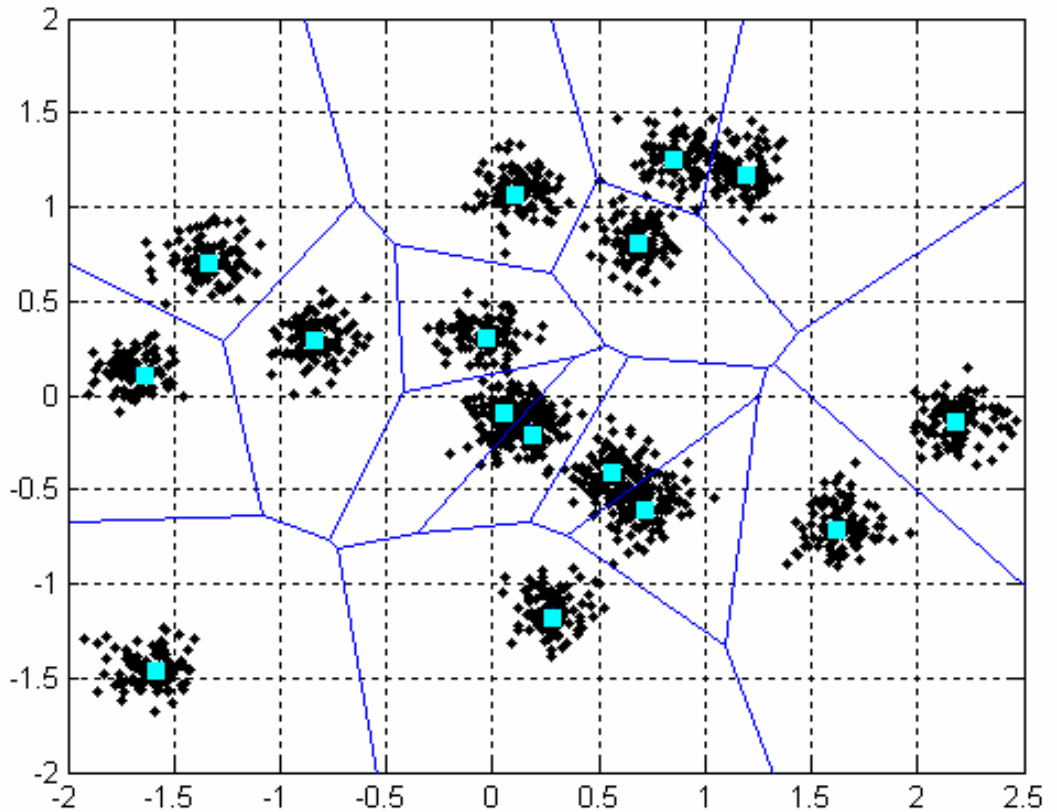


Exemplo4A.m – Comparação



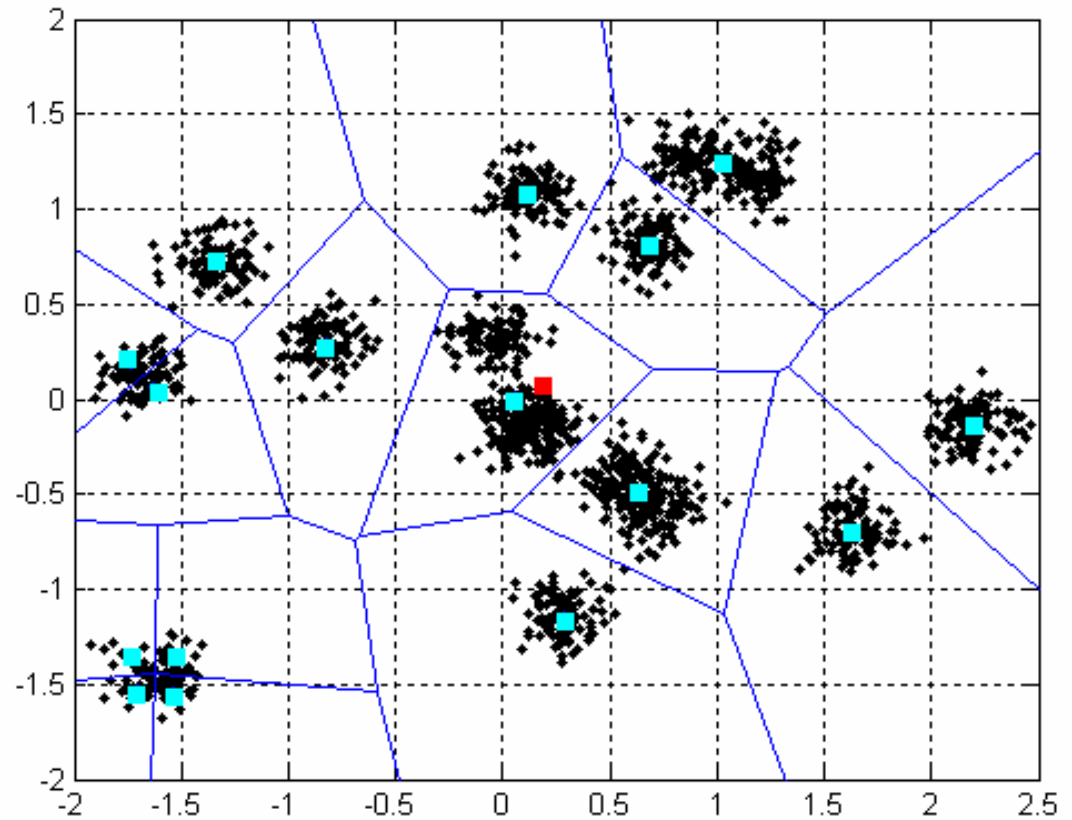
Exemplo4B.m – K=16 classes

$D_{\min} = 0.0188$



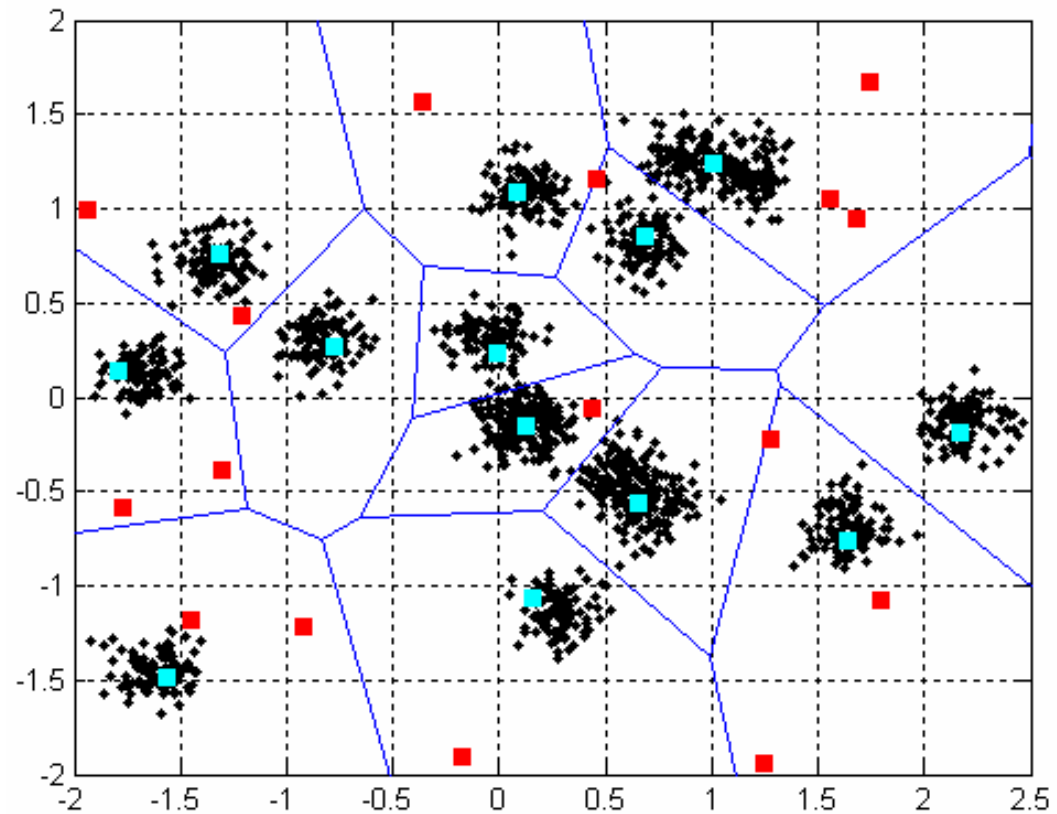
Exemplo4B.m – GLA

$$D_{\text{GLA}} = 0.0338$$



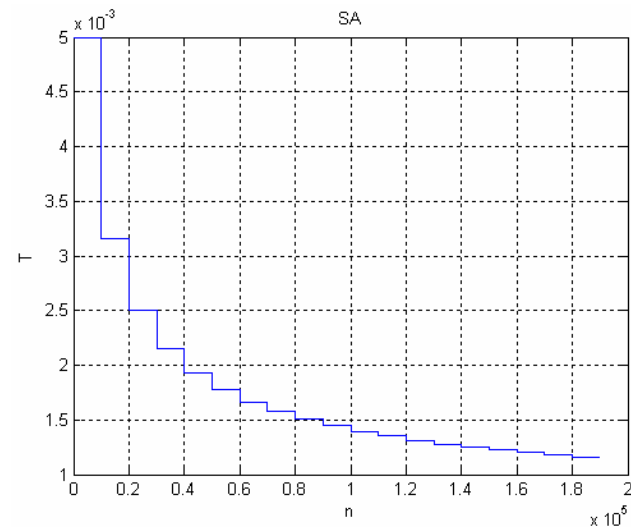
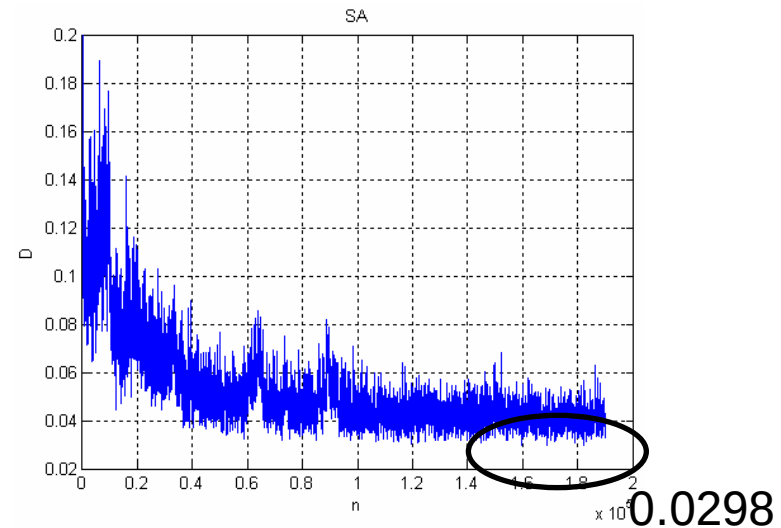
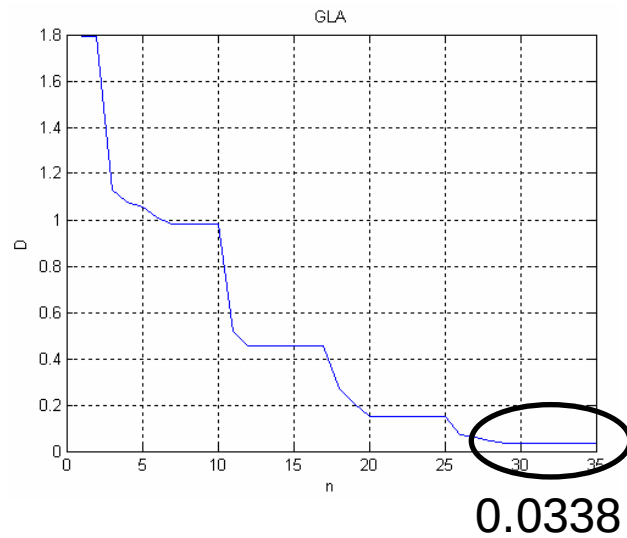
Exemplo4B.m – SA

$$D_{SA} = 0.0298$$



`(C0 = 4*(rand(2,K)-0.5);)` ao inves de `(C0 = randn(2,K);)`

Exemplo4B.m – Comparação



Conclusões

- LGA / LBG:

Mais rapidez e simplicidade
Mínimos locais

- SA: Evita mínimos locais

Mais lento (estocástico)
Mais parâmetros a escolher
Evita mínimos locais

Próxima Aula

- Deterministic Annealing

- Ler:

1. K. Rose. Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems. *Proceedings of the IEEE*, vol. 86, no. 11, November 1998, pp. 2210-2239. (Seção II-A).

2. K. Rose, E. Gurewitz, and G. C. Fox. Constrained Clustering as an Optimization Method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 8, August 1993, pp. 785-794.



Deterministic Annealing – Conceitos Básicos



UFRJ / COPPE / PEE
José Gabriel R. C. Gomes
CPE723 – Aula 05

Motivação (Rose, 1998)

- Habilidade de evitar mínimos locais.
- Aplicabilidade a diferentes tipos de problemas.
- Minimização de funções não-convexas (mesmo quando gradiente desaparece).
- Número mínimo de restrições (max. entropia).
- Analogia com física estatística.

Conceitos Básicos

- “Hard” Clustering

$$D = \sum_x p(x) d(x, y(x)) \approx \frac{1}{N} \sum_x d(x, y(x))$$

$$y(x) = c_i \longleftarrow i = \operatorname{argmin}_k d(x, c_k)$$

- “Soft” Clustering

$$\begin{aligned} D &= \sum_x \sum_y p(x, y) d(x, y) \\ &= \sum_x p(x) \sum_y p(y|x) d(x, y) \end{aligned}$$

Desenvolvimento: Função J

$$H(X, Y) = - \sum_x \sum_y p(x, y) \log p(x, y)$$

$$J = D - TH$$

$$H(X, Y) = H(X) + H(Y|X)$$

$$H(X) = - \sum_x p(x) \log p(x)$$

$$H(Y|X) = - \sum_x p(x) \sum_y p(y|x) \log p(y|x)$$

Desenvolvimento: Minimização de J

$$J = \sum_x p(x) \sum_y p(y|x) d(x, y) + T \sum_x p(x) \sum_y p(y|x) \log p(y|x) \\ + 1 - \sum_x \gamma(x) \sum_y p(y|x)$$

$$\sum_x \gamma(x) = 1$$

$$\frac{\partial J}{\partial p(y|x)} = p(x) d(x, y) + T p(x) (1 + \log p(y|x)) - \gamma(x) = 0$$

$$p(x) (d(x, y) + T \log p(y|x)) + \left(T - \frac{\gamma(x)}{p(x)} \right) = 0$$

$T \log \mu(x)$



Desenvolvimento: Minimização de J

$$\log p(y|x) = -\frac{d(x,y)}{T} - \log \mu(x)$$

$$p(y|x) = \frac{\exp\left(-\frac{d(x,y)}{T}\right)}{\mu(x)} \quad (!)$$

“Partition Condition”

$$\mu(x) = Z_x = \sum_y \exp\left(-\frac{d(x,y)}{T}\right)$$

Valor mínimo de J

Substituindo $p(y|x)$ na expressão $J = D - TH$, obtemos:

$$J_{min} = \sum_x p(x) \sum_y \left[\frac{\exp\left(\frac{-d(x,y)}{T}\right)}{Z_x} d(x,y) \right]$$
$$+ T \sum_x p(x) \sum_y \left[\frac{\exp\left(\frac{-d(x,y)}{T}\right)}{Z_x} \left(\frac{-d(x,y)}{T} - \log Z_x \right) \right]$$

$$J_{min} = -T \sum_x p(x) \log Z_x$$

Minimização em relação a c_i

$$\sum_x p(x, y) \frac{\partial}{\partial y} d(x, y) = 0, \forall y \in Y.$$

$$\frac{1}{N} \sum_x p(y|x) \frac{\partial}{\partial y} d(x, y) = 0, \forall y \in Y.$$

“Centroid Condition”

Algoritmo Básico

- 1. $T = T_0$ bem elevado. Dicionário inicial $C = C_0$.
- 2. Executar até a convergência de J:

Definir partição

Definir centróides

Calcular J

- 3. $T = \alpha.T$; Se $T > T_{\min}$, voltar para (2).
- 4. Fim

Algoritmo Básico - Partição

```
% Partition Condition
```

```
for k=1:size(X,2),  
    for l=1:size(C,2),  
        d = sum((X(:,k)-C(:,l)).^2);  
        p_ygivenx(l,k) = exp(-d/T);  
        d_aux(l,k) = d;  
    end;  
end;  
Zx = sum(p_ygivenx,1);  
p_ygivenx = p_ygivenx./repmat(Zx,size(C,2),1);
```

Algoritmo Básico - Centróide

```
% Centroid Condition
C_new = zeros(size(C));
for k=1:size(C,2),
    y = zeros(size(C,1),1); w = 0;
    for l=1:size(X,2),
        y = y + p_ygivenx(k,l)*X(:,l);
        w = w + p_ygivenx(k,l);
    end;
    C_new(:,k) = y/w;
end;
C = C_new;
```


Algoritmo Básico – Custos e Controle

```
% Cost Functions
```

```
J(i) = (-T/size(X,2))*sum(log(Zx));
```

```
D(i) = mean(sum(p_ygivenx.*d_aux,1));
```

```
LocalT(i) = T;
```

```
if (i>2),
```

```
    if (((abs(J(i)-J(i-1)))/abs(J(i-1)))*100)<Delta),
```

```
        T = alpha*T
```

```
        n=n+1; BKC(n).C = C; BKC(n).T = T;
```

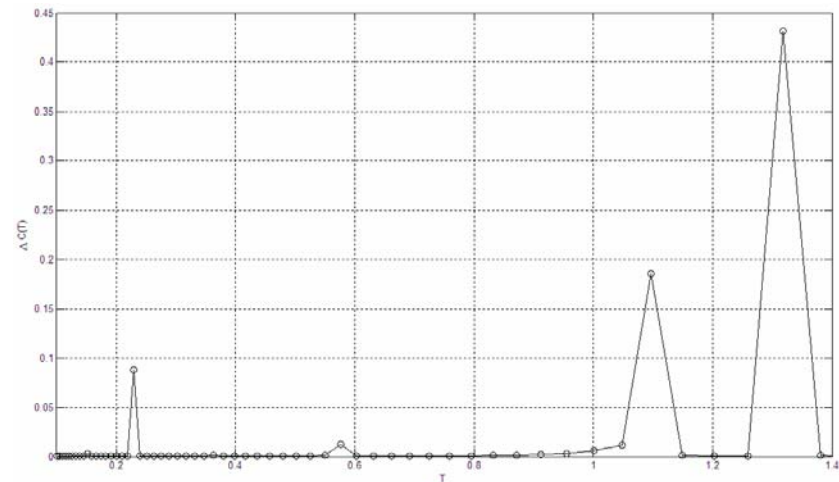
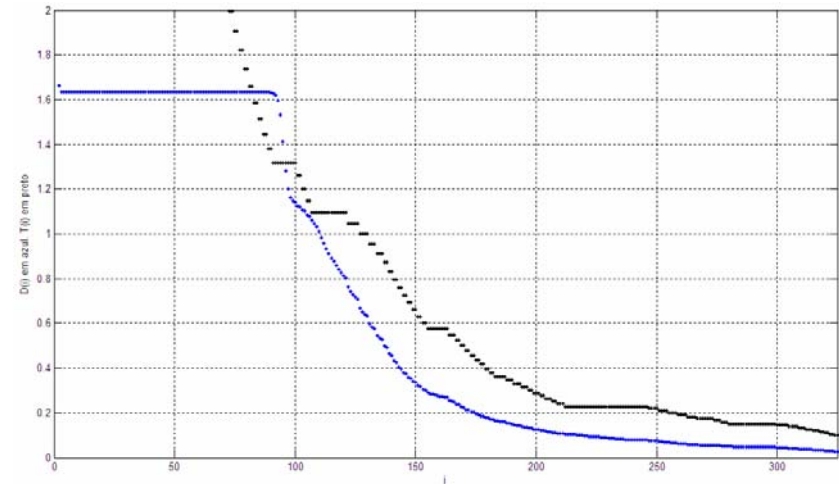
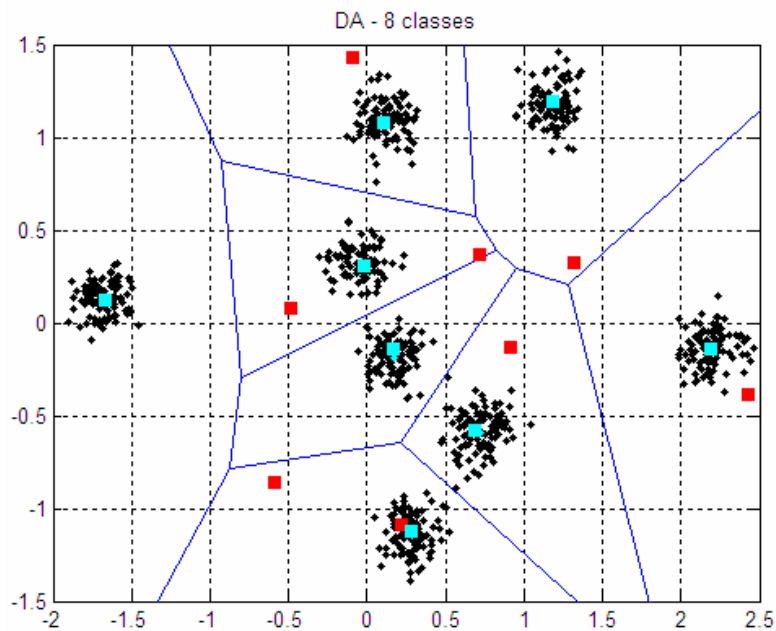
```
        C = C.*(1+epsilon*randn(size(C)));
```

```
    end;
```

```
end;
```

```
if T<0.1, fim = 1; end;
```

Exemplo 1 (Exemplo4A.m) ($K = 8$)

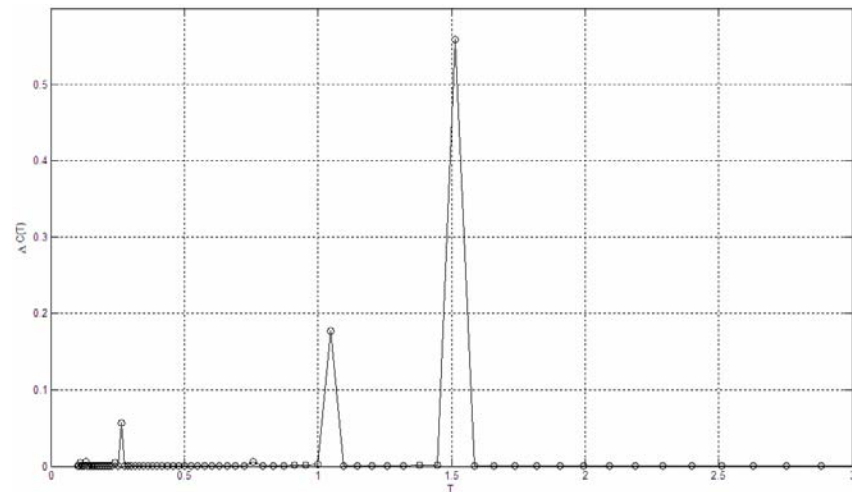
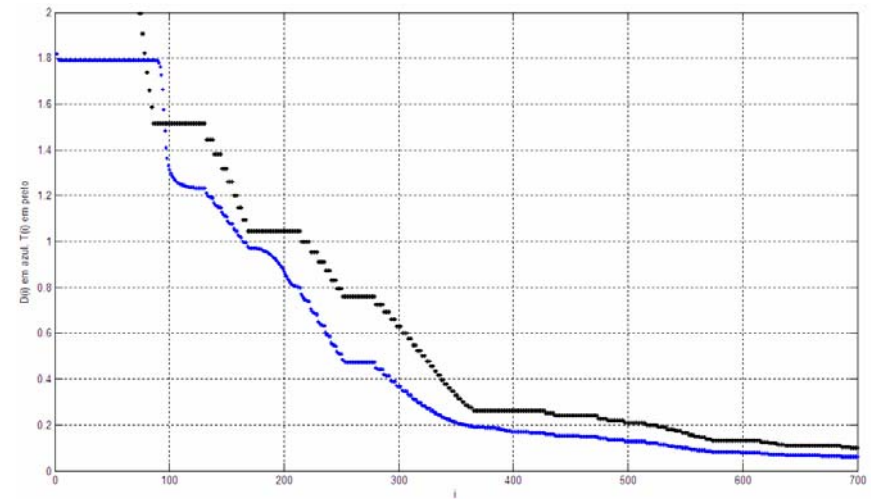
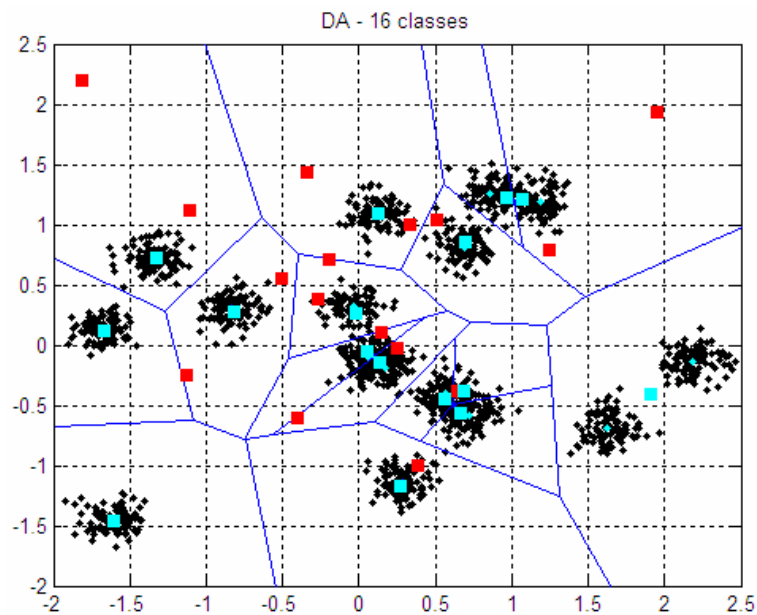


Exemplo 1 (Exemplo4A.m) ($K = 8$)

Algoritmo	Distorção
Mínimo Global	0.0195
GLA	0.0672
SA	0.0223
DA	0.0196

(Ver GLA e SA na Aula #5)

Exemplo 2 (Exemplo4B.m) ($K = 16$)

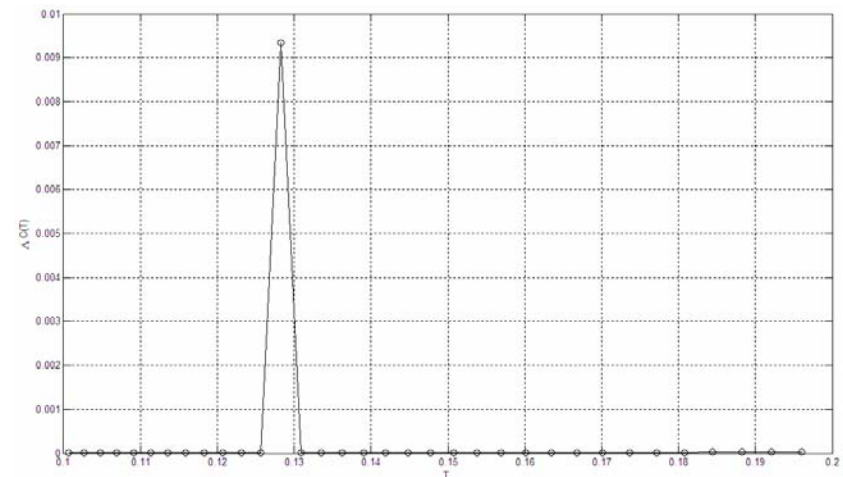
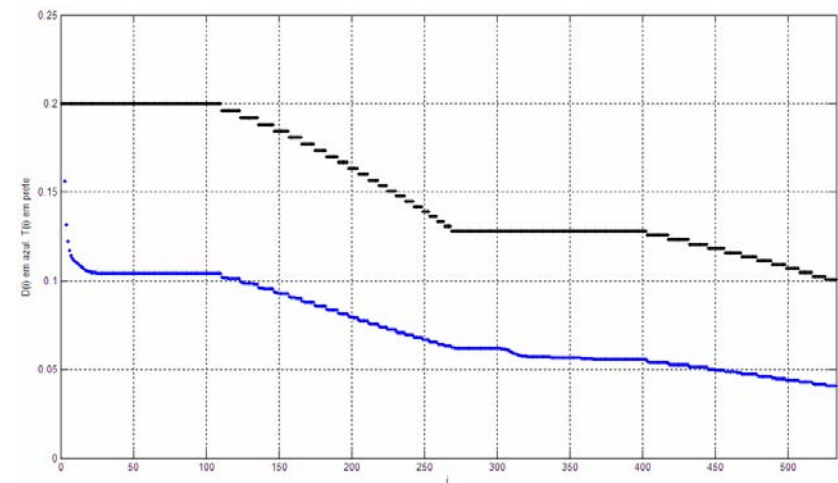
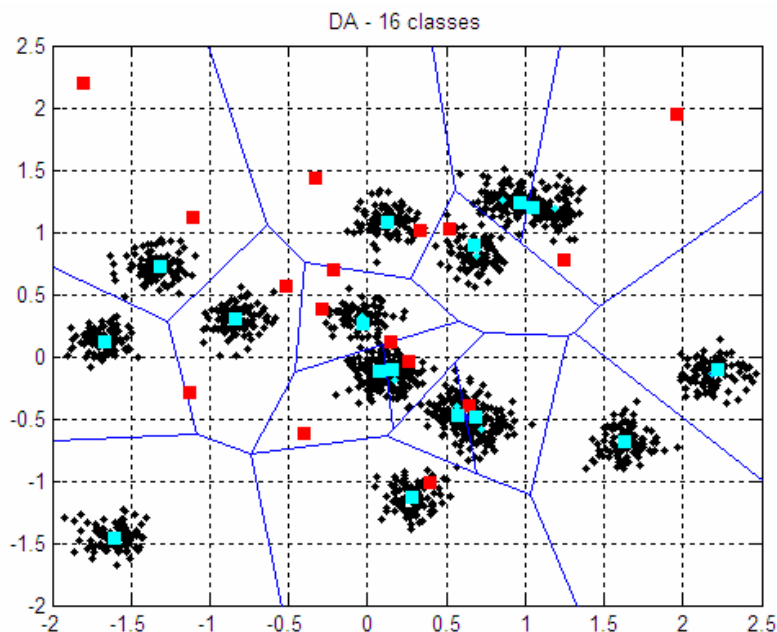


Exemplo 2 (Exemplo4B.m) ($K = 16$)

Algoritmo	Distorção
Mínimo Global	0.0188
GLA	0.0338
SA	0.0298
DA	0.0453 (pior)

(Ver GLA e SA na Aula #5)

Exemplo 3 (Exemplo4C.m) ($K = 16$)



Exemplo 3 (Exemplo4C.m) ($K = 16$)

Algoritmo	Distorção
Mínimo Global	0.0188
GLA	0.0338
SA	0.0298
DA	0.0258 (0.0195 (*))

(*) usando $\alpha = 0.5$ após $T < 0.1$

Conclusões sobre DA

- LGA / LBG:
 - Mais rapidez e simplicidade (DA)
 - Mínimos locais
- SA: Evita mínimos locais (DA)
 - Mais lento (estocástico)
 - Mais parâmetros (DA + PCA)
- “Mudanças de fase” em torno de temperaturas críticas.

Para a Próxima Aula

- Revisão sobre Deterministic Annealing
- Ler: Rose, 1998 (Seções II.A.1 e II.A.3)
- Iniciar Lista de Exercícios #3