4
Modelos Propostos para Geração de Detectores

A partir da análise do algoritmo do RNSA original descrito na Seção 3.4.1, foram desenvolvidos novos modelos para a geração de detectores com representação real.

4.1
RNSA com aprendizado LVQ

O RNSA com aprendizado LVQ (*Learning Vector Quantization*) foi o primeiro modelo desenvolvido. São propostas duas modificações no algoritmo original.

A primeira modificação proposta consiste em gerar o conjunto inicial de detectores utilizando uma seqüência quasi-aleatória (Morokoff, 1993), que é uma seqüência de amostras representativas de uma distribuição de probabilidade. Essas amostras são determinísticas e não aleatórias, e são retiradas de modo a que os "espaços" deixados entre as amostras sejam preenchidos, reduzindo o desvio-padrão da simulação de Monte Carlo e aumentando a velocidade de convergência (Boyce, Joy & Tan, 1996) (Morokoff, 1993).

Suponha-se que se deseje calcular a integral de uma função $f(x)$, através do método de Monte Carlo, dentro de um intervalo [0,1], usando N pontos ou observações. Ao invés de utilizar pontos de uma seqüência aleatória (ou pseudo-aleatória), pode-se usar pontos de uma seqüência determinística, onde eles são, de certa forma, distribuídos igualmente. Deste modo, a precisão da estimativa da integral será superior àquela obtida com a seqüência aleatória (Morokoff, 1993).

Entre as seqüências quasi-aleatórias mais conhecidas pode-se citar a seqüência de Faure (Faure, 1982), a seqüência de Sobol (Bratley & Fox, 1988) e a de Halton (Halton, 1960).

Para exemplificar o uso de seqüências quasi-aleatórias na geração de um conjunto inicial de detectores, suponha-se que se deseje distribuir estes detectores de raio igual a 0,05 no $[0,1]^2$. A Figura 4.1 mostra um gráfico da área coberta em

função do número de detectores usados. Pode–se observar que a área coberta pelos detectores gerados por seqüências quasi-aleatórias é maior do que aquela coberta pelos detectores gerados por uma seqüência pseudo-aleatória.

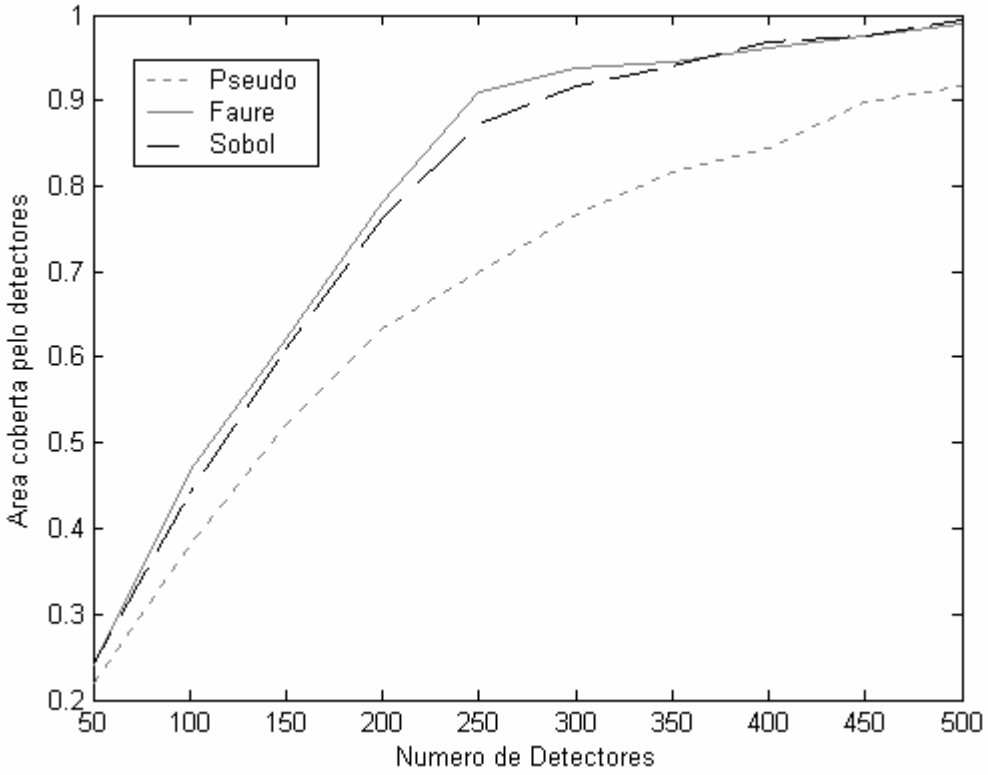

Figura 4.1 Gráfico Área Coberta x Número de Detectores.

A segunda alteração diz respeito à heurística utilizada para movimentar os detectores. De acordo com o algoritmo original, o detector, quando não está no espaço *próprio*, deve ser afastado dos outros detectores para melhorar a cobertura do espaço *não-próprio*. Para que a sobreposição existente entre os detectores diminua mais rapidamente, alterou-se a heurística original de modo a afastar não somente o detector em questão, mas também os detectores mais próximos a ele (no sentido oposto). Esta modificação foi inspirada na regra de aprendizado usada no LVQ (*Learning Vector Quantization*) para a atualização dos pesos.

O algoritmo RNSA-LVQ possui cinco parâmetros:

- *r*: raio de detecção;
- η: taxa de adaptação, isto é, a taxa com a qual os detectores se adaptam a cada passo;
- *t*: idade a partir da qual o detector é considerado incapaz;
- *k*: número de vizinhos do *conjunto próprio* para levar em conta;

- *kd*: número de vizinhos do *conjunto não-próprio*;

O pseudo-código do Algoritmo pode ser visto na Figura 4.2.

Gere uma população quasi-aleatória de detectores
Enquanto o critério de parada não for satisfeito
Para todo detector *d* **faça**
CélulasPróximas = *k* vizinhos do *ConjuntoPróprio* mais próximos de *d*
MaisProximoaoPróprio = mediana (*CélulasPróximas*)
DetectoresMaisProximos = *kd* vizinhos do *Conjunto não-próprio* de *d*

Se a *distância*(*d*, *MaisProximoaoPróprio*) < *r*
Então
Se idade do detector > t
Então
O detector *d* é incapaz
Substitua *d* por um novo detector aleatório
FimEntão
Senão
Incremente a idade de *d*

$$dir = \frac{\sum_{c \in Celulas\ Proximas} (d - c)}{\left| \sum_{c \in Celulas\ Proximas} (d - c) \right|}$$

$d = d + \eta^* dir$
FimSenão
FimSe
FimEntão
Senão
Faz idade do detector *d* igual a 0

$$dir = \frac{\sum_{d' \in Detectores} \mu_a(d')(d - d')}{\left| \sum_{d' \in Detectores} (d - d') \right|}$$

$d = d + \eta^* dir$
DetectoresMaisProximos = DetectoresMaisProximos - $\eta^* dir$
FimSenão
FimSe
FimParatodo
FimEnquanto

Figura 4.2 – Algoritmo RNSA-LVQ.

4.2 Geração de Detectores de Raios Variáveis por meio de Algoritmos Genéticos

Os algoritmos RNSA e RNSA-LVQ são baseados em heurísticas que tentam distribuir os detectores no espaço *não-próprio* de modo a maximizar a cobertura do mesmo. Estas abordagens têm como grandes desvantagens:

- Não existe uma maneira de determinar se o algoritmo está de fato melhorando o posicionamento dos detectores para que estes fiquem distribuídos da melhor maneira possível;

- Utilizam hiperesferas de raio fixo como detectores, o que pode comprometer a escalabilidade;
- O grande número de parâmetros a serem ajustados para que os algoritmos tenham um bom desempenho.

Para contornar estas dificuldades, é proposto um método de geração de detectores (hiperesferas de raios variáveis) eficientes na cobertura do espaço *não-próprio* por meio de um Algoritmo Evolucionário.

Os Algoritmos Evolucionários são inspirados no princípio darwiniano da evolução das espécies e na genética. Do mesmo modo que a Evolução Natural produz indivíduos mais aptos a sobreviver em um meio-ambiente sujeito a constantes mudanças, os Algoritmos Evolucionários podem ser vistos como procedimentos de otimização que melhoram o desempenho de uma população de soluções em potencial em relação a um problema específico.

Os Algoritmos Evolucionários principais são: os Algoritmos Genéticos, a Programação Genética, as Estratégias Evolutivas e a Programação Evolutiva. Maiores Detalhes sobre cada um destes algoritmos podem ser vistos em (Goldberg, 1989), (Michalewicz, 1992), (Schwefel, 1995), (Fogel, 1994), (Baeck, 1993), (Koza, 1992) e (Koza, 1994).

O Algoritmo Evolucionário utilizado foi o Algoritmo Genético. O funcionamento dos Algoritmos Genéticos pode ser compreendido considerando-se os seguintes aspectos: a representação do problema, o uso dos operadores genéticos (seleção, crossover e mutação) e a aplicação de uma função de avaliação de aptidão. A Figura 4.3 resume o processo de funcionamento de um Algoritmo Genético. Nas seções seguintes será feita uma descrição mais detalhada da representação do problema e da função de avaliação utilizada.

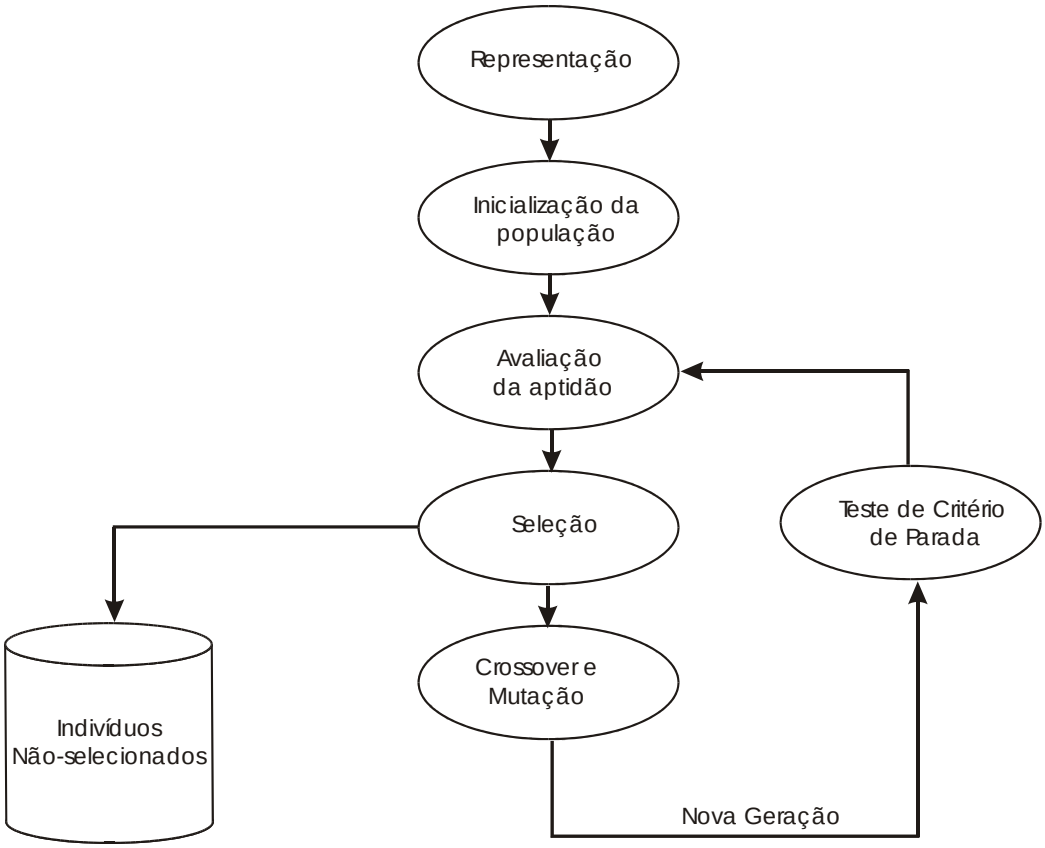

Figura 4.3 Funcionamento de um Algoritmo Genético.

4.2.1 Representação Utilizada

A representação do problema corresponde ao mapeamento das possíveis soluções em uma estrutura de dados que possa ser manipulada computacionalmente. Estas possíveis soluções codificadas recebem o nome de cromossomos.

Neste método, cada cromossomo representa um possível conjunto de detectores que estão codificados da seguinte forma: cada gene do cromossomo representa um ponteiro (índice) para um determinado ponto em uma distribuição de pontos quasi-aleatórios. Portanto, cada cromossomo indica quais pontos da distribuição quasi-aleatória devem ser os centros dos detectores. A razão pela qual foi utilizada uma distribuição quasi-aleatória se deve ao fato de que a área coberta pelos detectores gerados por seqüências quasi-aleatórias é maior do que aquela coberta pelos detectores gerados por uma seqüência pseudo-aleatória (Seção 4.1). A Figura 4.4 mostra um exemplo didático desta codificação. Os círculos representam o conjunto próprio, que possui três elementos, e os pontos numerados

são pontos de uma distribuição quasi-aleatória. Se um cromossomo for (3,6,10,4,8), então os pontos que têm estes índices serão escolhidos como centros.

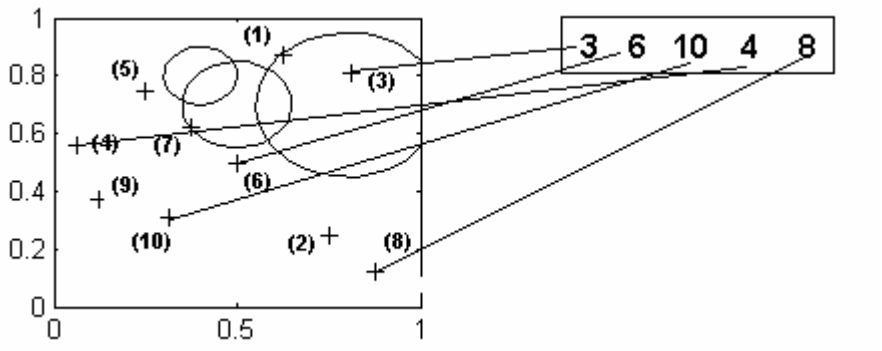

Figura 4.4 - Exemplo da codificação utilizada.

Uma vez determinados os centros dos detectores, uma função decodificadora calcula o maior raio possível para cada um, levando em conta que eles não podem atacar o conjunto *próprio* e que um certo limiar de sobreposição deve ser observado. A função decodificadora utiliza uma estratégia "gulosa" na qual, segundo uma certa ordem, cada detector vai sendo colocado no espaço por vez, sendo seu raio determinado em função do conjunto *próprio* e dos detectores que já foram colocados. Um aspecto que deve ser enfatizado é o fato de que, pelo processo de determinação dos raios dos detectores, nenhum deles ataca o conjunto próprio. Caso um centro escolhido esteja dentro do conjunto *próprio*, ele é descartado.

Este tipo de representação foi escolhido visando a diminuir o problema de escalabilidade, pois o tamanho do cromossomo depende apenas do número máximo de detectores que se quer usar e não da dimensão do espaço do problema.

A Figura 4.5 mostra como a função decodificadora determina os raios dos detectores a partir dos centros. Para cada um dos detectores indicados no cromossomo, a função decodificadora calcula, primeiramente, qual o maior raio possível para o detector, de forma que ele não ataque os pontos próprios; a seguir ajusta este raio para que haja um certo grau de sobreposição entre este detector e outros que já tenham sido gerados.

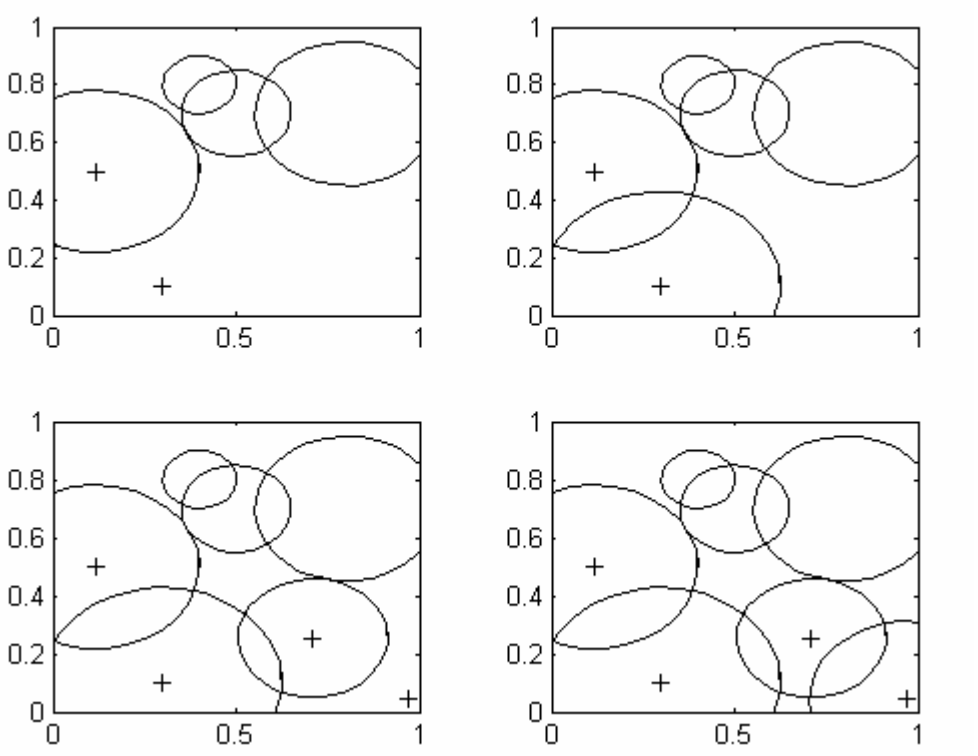

Figura 4.5 Determinação dos raios dos quatro primeiros detectores (+).

4.2.2
Função de Aptidão

A função de aptidão utilizada foi o volume coberto pelos detectores. A principal vantagem de se utilizar esta função é saber de antemão seu valor com uma certa margem de erro. Uma estimativa do volume ocupado pelo conjunto próprio V_p pode ser feita através de uma integração de Monte Carlo. Como o espaço *próprio*/*não-próprio* corresponde a um hipercubo unitário $[0,1]^n$, o valor do volume que deve ser coberto pelo conjunto de detectores é 1- V_p (Gonzalez et al., 2003).

O cálculo do volume coberto pelos detectores é feito através de uma integral de Monte Carlo, pois não se conhece a forma analítica da função que representa o volume. O cálculo desta integral é feito da seguinte forma: gera-se uma distribuição de pontos (pseudo-aleatória ou quasi-aleatória) capaz de cobrir de modo homogêneo o hipercubo unitário. Se N pontos forem gerados e N' destes pontos caírem dentro do volume coberto, pode-se aproximar este volume por N/N'.

Uma vantagem relacionada à integral de Monte Carlo é que o erro da integral está relacionado ao número de pontos utilizado e não à dimensão do problema (Mackay, 1999).

4.3 Geração de Detectores com Raios Variáveis por meio do Particionamento Quadtree

A decomposição Quadtree procura decompor um conjunto convexo cartesiano definido sobre um domínio reticulado convexo em subconjuntos cartesianos sucessivamente menores. Quando um conjunto cartesiano $S \in \Re^n$ passa por uma decomposição Quadtree, ele é dividido em 2^n subconjuntos s sem intersecção entre eles de tal modo que $\cup s_i = S \mid i = 1 \ldots 2^n$.

O particionamento Quadtree foi usado originalmente na representação de imagens (Smith & Chang, 1994), (Faloutsos et al., 1996). Neste caso, a imagem é dividida em blocos que são mais homogêneos que a imagem original, revelando informações a respeito da estrutura da imagem. A decomposição Quadtree é normalmente utilizada como uma primeira etapa em algoritmos de compressão.

A Figura 4.6 mostra a função PEAKS com a qual se deseja fazer uma decomposição Quadtree.

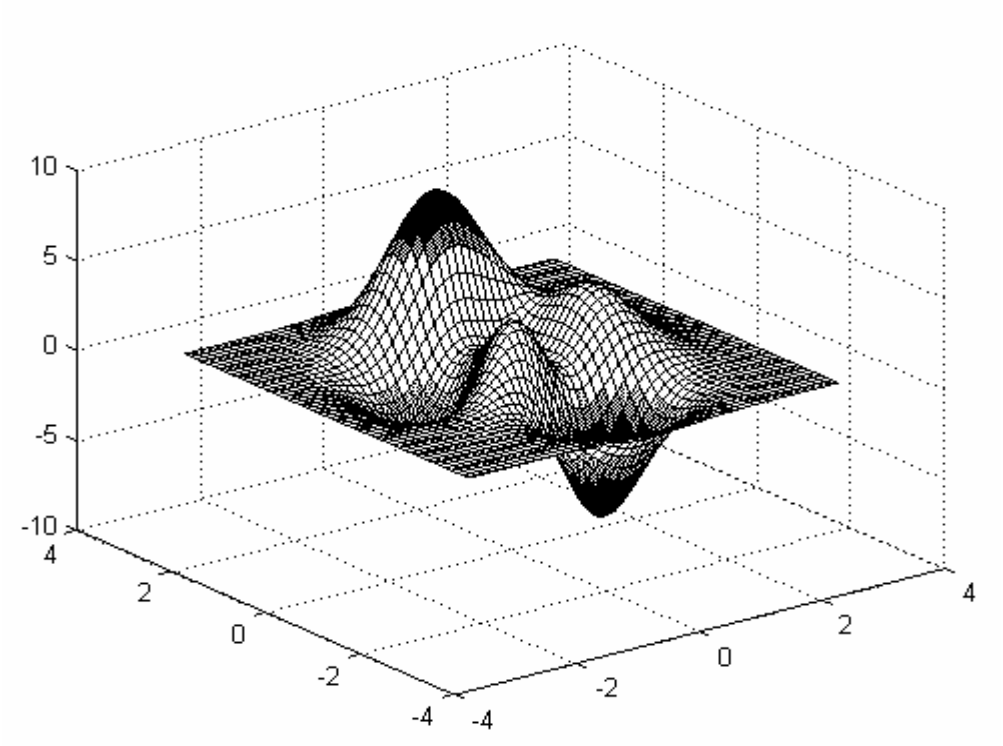

Figura 4.6 – A função Peaks.

Neste caso, uma região do domínio deve ser dividida sempre que a diferença entre o valor máximo e o valor mínimo da função PEAKS nesta região for maior do que um certo valor de limite (por exemplo, 0,8). O resultado pode ser visto na Figura 4.7.

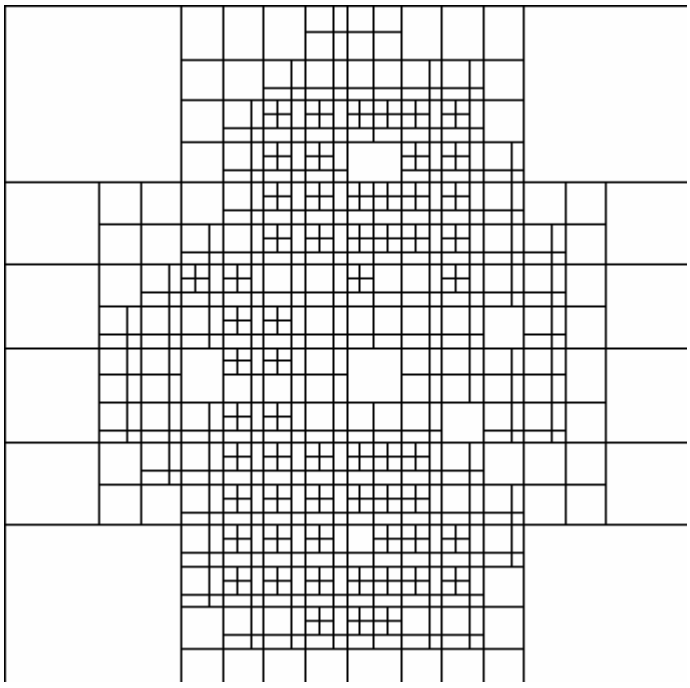

Figura 4.7 – A decomposição Quadtree da função PEAKS.

O procedimento para gerar um conjunto de detectores utilizando a decomposição Quadtree pode ser implementado de duas maneiras diferentes: a primeira realiza o particionamento do domínio da chamada função de detecção e a segunda se baseia no particionamento do próprio detector. As próximas seções vão descrever cada uma destas formas com maior detalhe.

4.3.1
Particionamento Quadtree por meio da Função de Detecção

O procedimento para gerar um conjunto de detectores utilizando a decomposição Quadtree através da função de detecção é iniciado definindo-se um reticulado (*grid*) sobre o qual será feita a decomposição. A seguir a Função de Detecção (*FD*) em relação ao conjunto *próprio* é calculada para cada ponto do *grid*. Esta função de detecção tem a seguinte forma:

$$FD(g_i) = \frac{1}{n}\sum_{j=1}^{n} \frac{1}{(2\pi r_s^{\,2})^{(d/2)}}\, e^{-\left(\frac{\|g_i - s_j\|}{\sqrt{2}r_s}\right)^2}$$

onde: s_j é um elemento do conjunto próprio, g_i é um ponto do grid e r_s é o raio definido para os elementos do conjunto *próprio*.

Observa-se que esta função terá valores mais altos para os pontos do *grid* que estiverem mais próximos do *conjunto próprio*. A decomposição Quadtree permite que o espaço seja dividido em regiões nas quais o valor da função é praticamente constante. Analisando-se o valor da função em cada bloco, é possível determinar se esta região de pontos "ataca" o conjunto de pontos *próprios* ou não. Aquelas regiões nas quais os pontos não atacam o conjunto próprio são utilizadas para definir os detectores. A posição do detector e o seu raio de ação são definidas a partir das dimensões do bloco.

A Figura 4.8 mostra um conjunto de pontos próprios para os quais será gerado um conjunto de detectores. A Figura 4.9 mostra a função de detecção calculada para os pontos do *grid* e a Figura 4.10 mostra o conjunto inicial de detectores gerado.

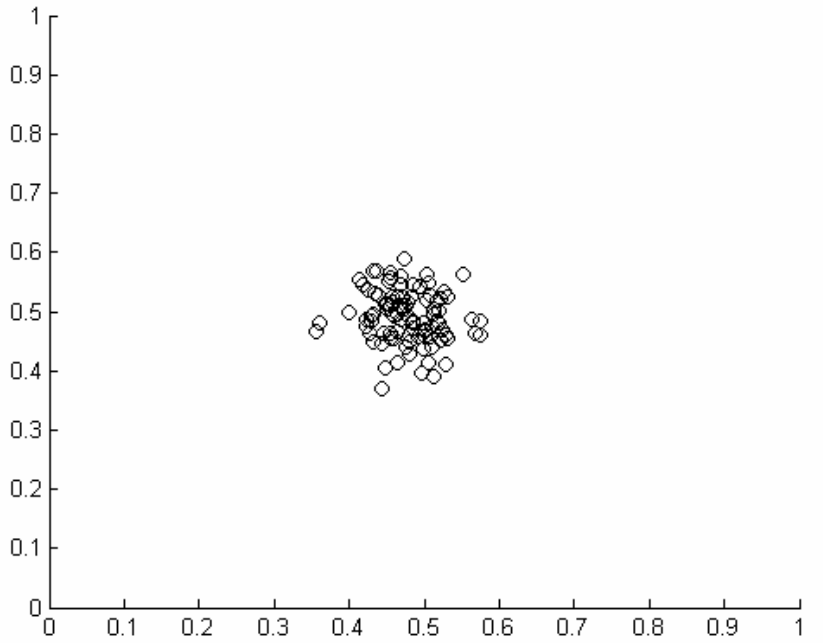

Figura 4.8 Conjunto de pontos próprios.

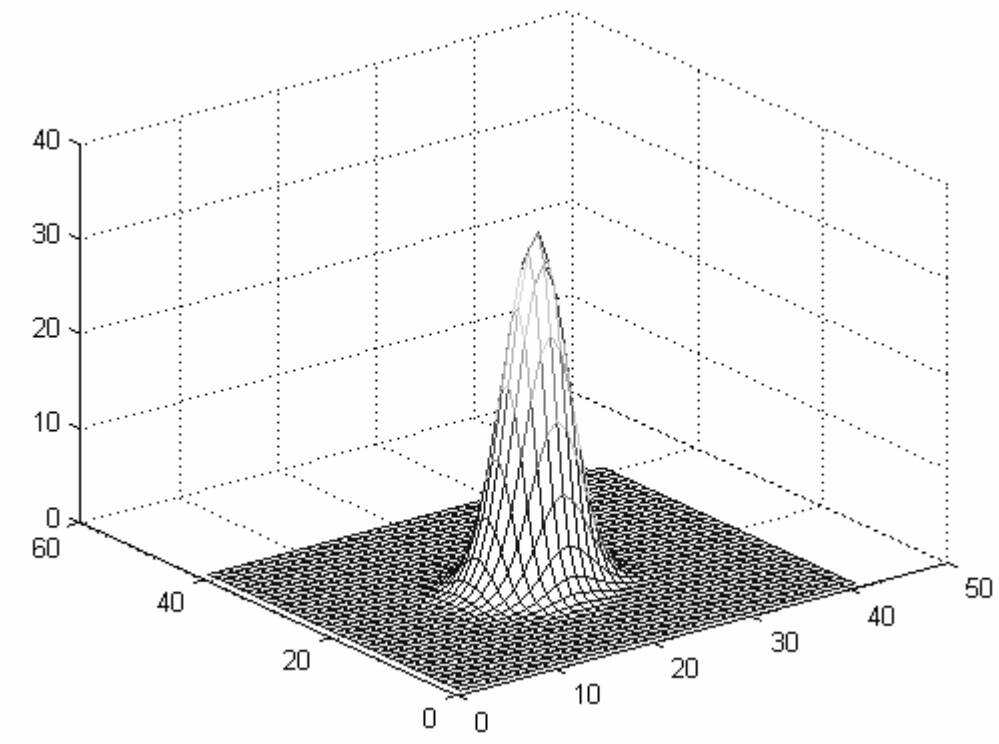

Figura 4.9 – Função de Detecção.

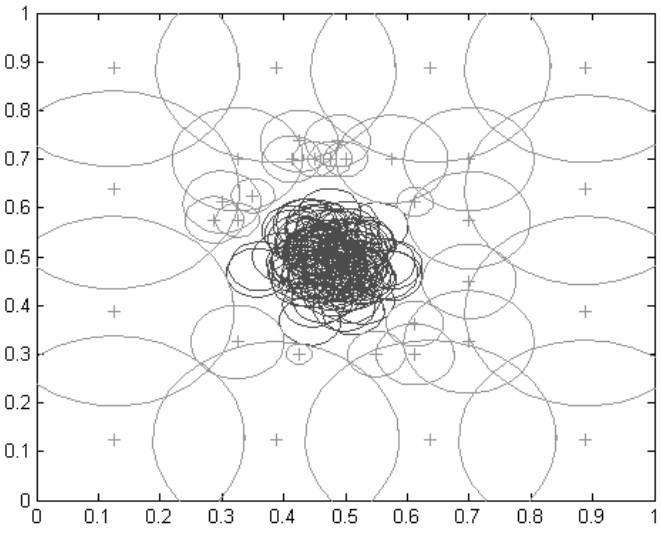

Figura 4.10 – Conjunto de detectores usados.

4.3.2
Particionamento Quadtree do Detector

A geração de um conjunto de detectores através de uma função de detecção tem como desvantagem a necessidade de cálculo desta função em cada ponto do grid, o que em determinadas situações pode ser bastante custoso. Uma alternativa

para evitar este cálculo é fazer o particionamento a partir do próprio detector. Este procedimento é descrito pelo fluxograma da Figura 4.11.

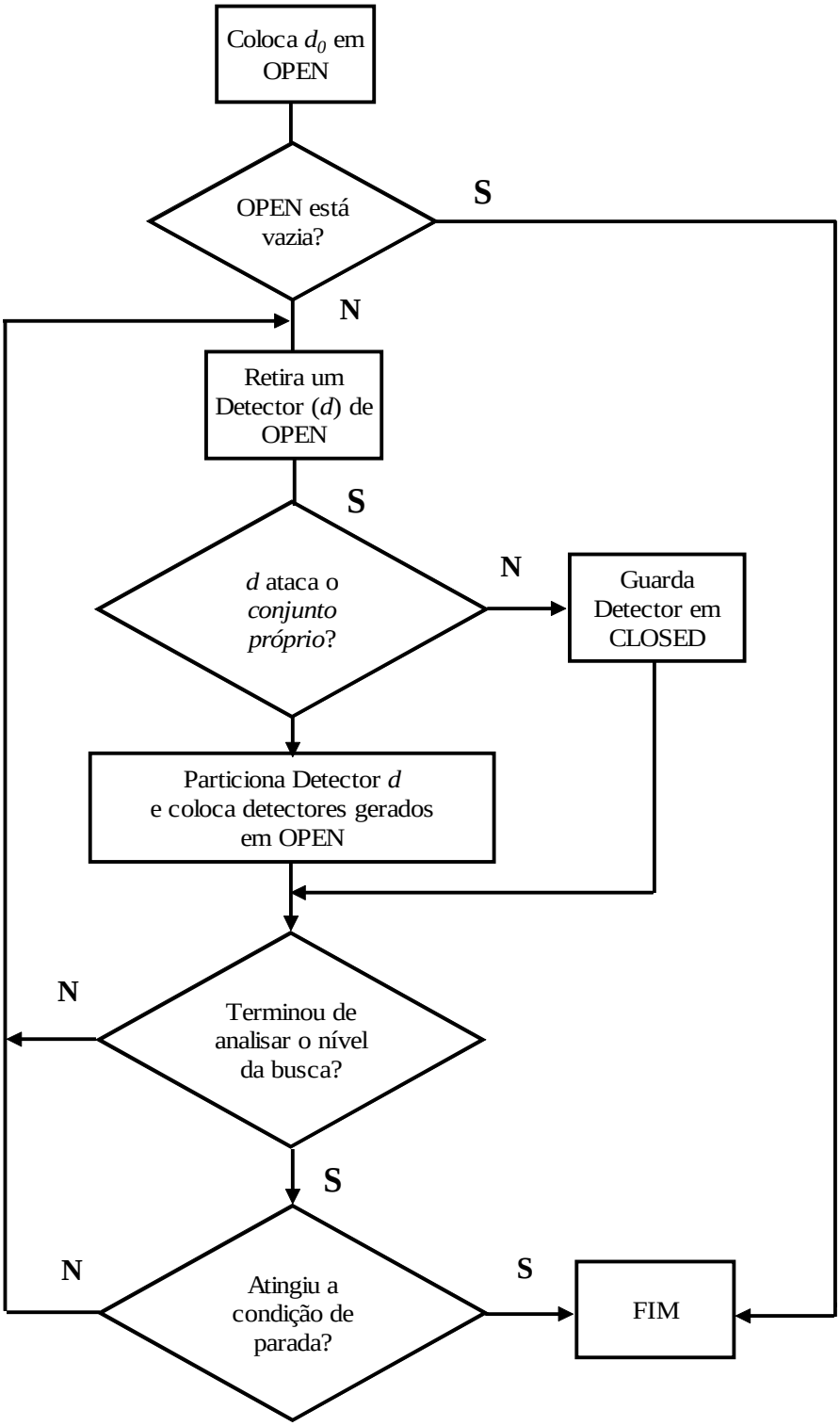

Figura 4.11 – Fluxograma para particionamento do detector.

O algoritmo utiliza duas filas chamadas de OPEN e CLOSED. Na primeira são armazenados os detectores que serão testados e na segunda são armazenados os detectores que devem ser utilizados.

Antes de iniciar o algoritmo, o detector inicial d_0 deve ser colocado na fila OPEN e a fila de CLOSED deve estar vazia. Este detector d_0 tem seu centro localizado no centro do hipercubo e raio igual a $\sqrt{n}/2$, o que significa que ele abrange todo o hipercubo.

O procedimento consiste em retirar um detector *d* da fila de OPEN e testar se este detector ataca o conjunto *próprio*. Caso ele não ataque ele deve ser armazenado na fila de CLOSED. Se ele atacar o conjunto *próprio*, ele deve ser particionado em 2^n detectores, onde *n* é a dimensão do problema. Depois que o detector foi testado, verifica-se se todos os detectores de um mesmo raio já foram analisados, isto é, se já terminou a análise de um nível do algoritmo. Caso a análise do nível tenha terminado, é hora de verificar se a condição de parada foi atingida. Caso contrário, o algoritmo volta para analisar um outro detector.

Podem ser utilizadas diferentes condições de parada: raio mínimo para o detector, número máximo de detectores e volume ocupado pelos detectores. O cálculo do volume é feito através de uma integração de Monte Carlo.

4.4 Geração de Detectores com Raios Variáveis por meio de Redes Imunológicas

Conforme visto na seção (3.4.2), Jerne (Jerne, 1974) propôs a Teoria da Rede Imunológica para descrever a atividade dos linfócitos. Sua hipótese é que anticorpos e linfócitos não agem isoladamente, mas que o sistema imunológico mantém uma rede de células B para o reconhecimento de antígenos. Estas células podem estimular e inibir umas às outras de várias formas, o que levaria à estabilização da rede.

A abordagem utilizando redes é particularmente interessante para o desenvolvimento de ferramentas computacionais, pois ela naturalmente leva em consideração propriedades como aprendizado, memória, tamanho e diversidade da rede (Castro, 2002d).

De um modo geral, a estrutura da maioria dos modelos de redes imunológicas pode ser descrita conforme mostrado na Figura 4.12 (Perelson, 1989).

Taxa de Variação da População = **Estimulação da Rede** - **Supressão da Rede** + **Entrada de Novos Elementos** - **Morte de Elementos Não Estimulados**

Figura 4.12 – Modelo para descrição de Redes Imunológicas (Perelson, 1989).

Os dois primeiros termos estão relacionados à dinâmica da rede, enquanto que os dois últimos se referem a sua metadinâmica.

A partir destes conceitos sobre redes imunológicas, surgiu a idéia de utilizar um modelo de rede imunológica para gerar os detectores. Neste modelo, as células B representam os detectores e a rede formada por eles dá a posição e o raio de cada detector. De acordo com o modelo de rede imunológica, os detectores podem estimular e inibir uns aos outros, de várias formas, até que a rede se estabilize. Neste caso, até que a distribuição dos detectores e seus raios estejam determinados.

O objetivo de cada detector é envolver o maior volume possível do hipercubo unitário sem que haja sobreposição com os elementos do conjunto próprio e com a menor sobreposição possível com os outros detectores. Portanto, a afinidade (*f*) de um detector será dada pela seguinte equação:

$$f = V_{TD} - V_{OD} - KV_{OS}$$

onde V_{TD} é o volume total do detector, V_{OD} é a parcela do volume do detector que está em sobreposição (*overlap*) com outros detectores e V_{OS} é a parcela do volume do detector que está em sobreposição (*overlap*) com os elementos do conjunto *próprio*. *K* é uma constante que representa uma penalidade para a situação onde há sobreposição entre o conjunto próprio e o detector, o que caracteriza um "ataque" do detector ao conjunto *próprio*. Pode-se dizer que V_{TD} representa a estimulação enquanto que V_{OD} e KV_{OS} representam a supressão da rede.

O algoritmo para obter a estabilização da rede pode ser descrito nos seguintes passos:

1. Inicialize uma população de detectores utilizando uma seqüência quasi-aleatória (para melhor cobertura do espaço).
2. Enquanto (a condição de parada não for atingida) faça
3. Para (cada um dos detectores) faça
    a. Determine sua afinidade
    b. Gere um clone de acordo com sua afinidade
    c. Se o clone tem afinidade maior do que o detector, substitua este pelo clone.
4. Verifique se a condição de parada foi atigida. Em caso positivo, termine o algoritmo.
5. Verifique se volume coberto pela rede de detectores (*V*) está mais próximo do valor esperado (*VE*). Se estiver, retorne ao passo 2.
6. Substitua os detectores com pior desempenho por outros. Retorne ao passo 2.

A condição de parada para o algoritmo pode ser o número de gerações ou se o valor calculado para o volume estiver próximo do esperado.

A afinidade do detector é calculada através de uma "micro-integração de Monte Carlo". Para que esta integração seja feita, utiliza-se um hipercubo envolvendo o detector. Este hipercubo tem aresta igual a 2*r*, onde *r* é o raio do detector e seu centro está localizado no mesmo ponto do centro do detector. Sobre este hipercubo é feita uma distribuição quasi-aleatória para ser empregada na integração de Monte Carlo. O volume total do hipercubo é $(2r)^n$. A multiplicação da porcentagem (*p*) dos pontos da seqüência quasi-aleatória que estão dentro do detector pelo volume do hipercubo fornece uma estimativa do volume ocupado pelo detector. Um raciocínio semelhante pode ser feito para determinar a parcela do volume do detector que se sobrepõe aos outros detectores e também aquela que se sobrepõe aos elementos do conjunto *próprio*. A Figura 4.13 mostra um exemplo usado para ilustrar o cálculo da afinidade.

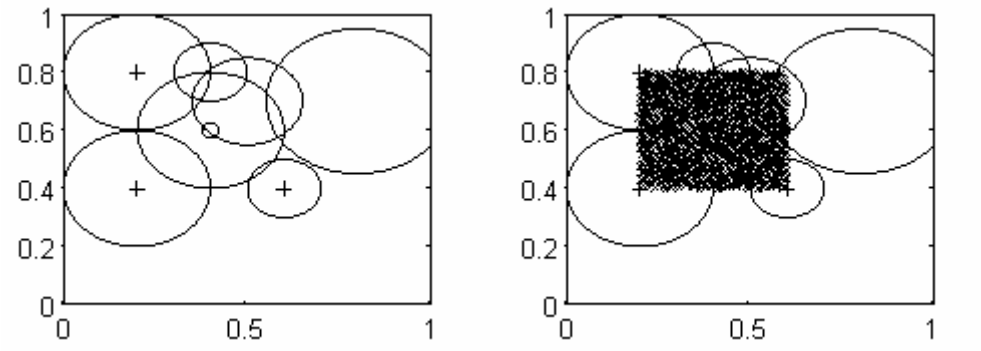

Figura 4.13 – Exemplo para o cálculo da afinidade

O detector para o qual se deseja calcular a afinidade está indicado com um ‘o’ em seu centro. Um hipercubo envolve o detector sobre o qual deseja-se calcular o volume. O número de pontos utilizado para a seqüência quasi-aleatória é 1000. A porcentagem de pontos que se encontram dentro do detector é igual a 0,785, o que dá um volume estimado de 0,1256, que é bastante próximo ao volume total do detector calculado analiticamente (0,1257). O mesmo procedimento é utilizado para determinar os volumes de sobreposição.

Para gerar o clone de acordo com a afinidade, foi utilizado o algoritmo de Seleção Clonal, de forma semelhante ao que é feito na aiNET (Castro, 2001c). Este algoritmo admite que o clone seja obtido através de um método de otimização como o SQP (*Sequential Quadratic Programming*) ou por Estratégia Evolucionária. Uma vez que o Algoritmo de Seleção Clonal já foi descrito na Seção 3.4.3, as duas próximas seções fazem uma breve descrição sobre estes métodos de otimização.

4.4.1 Métodos de Otimização SQP

Os métodos de otimização SQP representam uma importante classe de métodos de programação não-linear. Schittkowski (Schittkowski, 1983) implementou e testou uma versão do método que apresentou desempenho superior em diversos testes realizados. Em (Houck et al., 1995), um método de otimização SQP é utilizado para otimizar a função Corona a partir de um ponto fornecido pelo Algoritmo Genético. Isto fornece ao Algoritmo Genético um operador de busca local que pode melhorar bastante seu desempenho.

De um modo geral, a partir de um problema de minimização apresentado na seguinte forma:

$$\begin{aligned}
&\text{minimize} \quad f(x) \\
&x \in \Re^n \\
&\text{sujeito a} \\
&G_i(x) = 0, \qquad i = 1, \ldots, m_e \\
&G_i(x) \leq 0, \qquad i = m_e + 1, \ldots, m \\
&x_l \leq x \leq x_u
\end{aligned}$$

onde $G_i(x)$ representam as restrições da função $f(x)$, e x_l e x_u representam as restrições de domínio, formula-se um subproblema QP baseado na aproximação quadrática da função Lagrangiana. A cada interação, uma aproximação do Hessiano da função Lagrangiana é feita usando um método de atualização quasi-Newton. Este então é usado para gerar um subproblema QP cuja solução é usada para gerar uma direção de busca.

Uma descrição mais detalhada dos métodos de otimização SQP pode ser vista em (Fletcher, 1980), (Gill et al, 1981), (Powell, 1983) e (Hock & Schittkowski, 1983).

4.4.2 Estratégia Evolucionária

A Estratégia Evolucionária é um Algoritmo Evolucionário que apresenta as seguintes características:

Um indivíduo $a = (x, \sigma, \alpha)$ pode ter até três componentes: $x \in \Re^n$ (uma possível solução), $\sigma \in \Re^{n_\sigma}$ (um conjunto de desvios padrões de uma distribuição normal), e $\alpha \in [-\pi, \pi]^{n_\alpha}$ (um conjunto de ângulos de rotação que representam as covariâncias de uma distribuição normal de dimensão *n*).

A seleção dos indivíduos que farão parte da próxima geração é determinística, e pode ser feita apenas sobre os descendentes ou também levar em consideração os genitores.

O operador de mutação utilizado é implementado através de variações com distribuição normal sobre passos ajustáveis (isto é, a variâncias e covariâncias da distribuição normal). A mutação é o operador principal e a recombinação tem

papel secundário. As variâncias e covariâncias do operador de mutação são os parâmetros estratégicos que fazem parte do indivíduo. Estes mesmos parâmetros estratégicos são otimizados durante a busca através um processo de auto-adaptação. Dependendo do modo que os parâmetros estratégicos são incorporados na representação de um indivíduo, algumas variantes de mutação e auto-adaptação podem ser destacadas:

- $n_\sigma = 1$, $n_\alpha = 0$: o desvio padrão (σ) é o mesmo para todas as variáveis da solução; a mutação é feita da seguinte forma:

$$\sigma' = \sigma.\exp(\tau_0.N(0,1))$$

$$x'_i = x_i + \sigma'.N_i(0,1)$$

onde $\tau_0 \propto (\sqrt{n})^{-1}$.

$N(0,1)$ representa um valor amostrado de uma variável aleatória com distribuição normal de média 0 e variância igual a 1. A notação $N_i(0,1)$ indica qual variável é amostrada para cada valor de i.

- n_σ = n, $n_\alpha = 0$: cada uma das variáveis tem o seu próprio desvio padrão σ_i; agora, a mutação passa a ser realizada da seguinte forma:

$$\sigma_i' = \sigma_i.\exp(\tau'.N(0,1) + \tau.N_i(0,1))$$

$$x'_i = x_i + \sigma_i'.N_i(0,1)$$

onde $\tau' \propto (\sqrt{2n})^{-1}$ e $\tau \propto (\sqrt{2\sqrt{n}})^{-1}$.

- $n_\sigma = n$, n_α = n(n-1)/2: os vetores σ e α representam a matriz de covariância completa de uma distribuição normal de dimensão n, onde as covariâncias são dadas pelos ângulos de rotação α_j que descrevem as rotações de coordenadas necessárias para transformar um vetor de mutação não correlacionado em um vetor correlacionado; agora a mutação passa a ser realizada da seguinte forma:

$$\sigma_i' = \sigma_i.\exp(\tau'.N(0,1) + \tau.N_i(0,1))$$

$$\alpha_j' = \alpha_j + \beta.N_j(0,1)$$

$$x'_i = x_i + N\ (0, C(\sigma', \alpha'))$$

onde $N(0, C(\sigma', \alpha'))$ representa o vetor de mutação correlacionado e $\beta \approx 0.0873$.

A Figura 4.11 ilustra os três diferentes tipos de mutação auto-adaptativa.

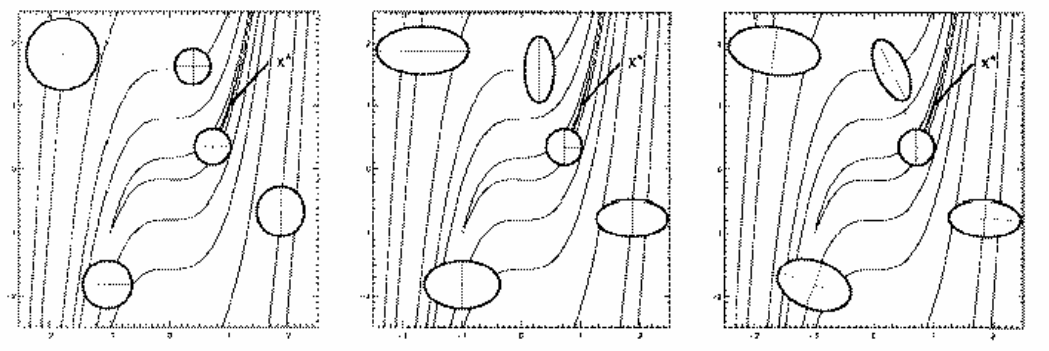

Figura 4.14 – Três tipos de mutação auto-adaptativa (Back et al, 1997).

A Figura 4.11 ilustra os três diferentes tipos de mutação auto-adaptativa. O primeiro (à esquerda) representa o caso no qual todas as variáveis da solução possuem o mesmo desvio padrão, o que é representado por hiperesferas (círculos no caso bidimensional). O segundo (no centro) representa o caso onde cada variável tem seu próprio desvio padrão, o que é representado por hiperelipsóides. No terceiro, as variáveis são correlacionadas, o que é representado por meio de hiperelipsóides rotacionados.