4
Construção do grafo de falhas utilizando Growing Neural Gas

4.1.
Aprendizado Competitivo

Esta seção apresenta os conceitos envolvidos em Aprendizado Competitivo de forma a definir a nomenclatura e estabelecer as relações entre esses conceitos e o problema de extração e visualização de estruturas de falhas sísmicas. Uma discussão mais detalhada desses conceitos pode ser encontrada em Haykin (2001) e nas referências por ele citadas. Por sua vez, Fritzke (1997) apresenta de forma estruturada uma lista com vários algoritmos de Aprendizado Competitivo.

Rede Neural Artificial é um termo genérico que descreve algum tipo de simulação de uma máquina composta por um conjunto de nós ou unidades interconectadas[21]. Os nós se comunicam com seus nós vizinhos, recebendo e transmitindo informações através de conexões chamadas sinapses. Estas podem também armazenar informações, sob a forma de pesos escalares (pesos sinápticos).

Uma rede neural pode ser supervisionada ou não-supervisionada. No primeiro caso, a rede recebe um conjunto de dados de entrada e seus correspondentes dados de saída (par de treinamento), os quais ensinam a rede o comportamento desejado. A rede aprende com o conjunto de treinamento e então é capaz de extrapolar valores de saída para novas entradas. Um exemplo típico de rede neural supervisionada é a rede Perceptron (Haykin, 2001).

No caso de uma rede neural não-supervisionada, não é preciso que sejam fornecidas as saídas desejadas para os dados de treinamento; a rede é capaz de aprender os padrões significativos ou as características existentes nos dados

[21] O termo “unidade” é mais utilizado no contexto de redes neurais, enquanto “nó” é utilizado no contexto de teoria de grafos ou estruturas de dados. Neste trabalho os dois termos são utilizados indistintamente. Além desses dois, o termo “vértice“, utilizado em geometria computacional, ocorre também neste documento, com o mesmo significado que os demais.

de entrada, sem a intervenção explícita de algo que funcione como um professor. Este tipo de aprendizagem também é conhecido como aprendizagem auto-organizada[22].

Entende-se por comportamento auto-organizável a formação espontânea de estruturas organizadas, tais como padrões ou comportamentos, a partir de condições iniciais aleatórias, frutos da interação local entre os componentes do sistema como um todo. Muitas interações locais originalmente aleatórias entre neurônios vizinhos de uma rede podem se fundir em estados de ordem global e finalmente levar a um comportamento coerente na forma de padrões espaciais ou ritmos temporais. Como observado por Turing (1952): "ordem global pode surgir de interações locais".

A organização da rede acontece em dois níveis diferentes:

- Atividade: padrões de atividade da rede são produzidos como respostas a sinais de entrada[23].
- Conectividade: forças de conexão entre unidades da rede (pesos sinápticos) são alterados (plasticidade sináptica) em resposta a padrões de atividade da própria rede.

A interação entre esses dois níveis forma o que é chamado de laço de realimentação. A seguir, são apresentados três princípios inspirados no comportamento do cérebro humano que descrevem a dinâmica de redes auto-organizáveis (Haykin, 2001):

1. Modificações dos pesos sinápticos tendem a se auto-amplificar. Para se obter auto-organização, a realimentação entre as modificações nos pesos sinápticos e as modificações nos padrões de atividade deve ser positiva. Este princípio também é conhecido como postulado de aprendizagem de Hebb[24].

[22] O conceito de auto-organização surgiu, no final da década de 40 e início da de 50 do século XX, no contexto do que era chamado, na época, de cibernética (algo que envolvia neurociências, teoria da informação e teoria de controle). Posteriormente, o conceito foi aplicado, com alterações, em outras áreas da ciência, como na Física, Química e Biologia (Capra, 2006).

[23] Sinal de entrada é um padrão de ativação fornecido à rede pelo ambiente.

[24] O postulado de aprendizagem de Hebb é a mais antiga (data de 1949) e famosa de todas as regras de aprendizagem. Pode ser enunciado como uma regra em duas partes:

2. A limitação dos recursos leva à competição entre sinapses e, com isso, à seleção das sinapses que crescem mais vigorosamente (as mais ajustadas), em detrimento das demais. Isto permite que o sistema se estabilize em algum momento.
3. As modificações dos pesos sinápticos tendem a cooperar. Uma única sinapse por si só não pode produzir eficientemente eventos favoráveis.

De acordo com a segunda lei da termodinâmica, quanto maior a desordem (entropia) de um sistema isolado, mais estável é o estado do sistema. Sistemas auto-organizáveis, entretanto, evoluem para estados organizados, não estando sujeitos a essa lei. Esse comportamento se deve ao fato de que sistemas auto-organizáveis não são sistemas isolados. Assim, para que determinados tipos de transições (que levem à formação de padrões) ocorram, é necessário que o sistema seja mantido longe da situação de equilíbrio total através do fluxo de entrada de energia por meio de estímulos do meio ambiente (Barreto, 2003). Além disso, para que a aprendizagem auto-organizada realize uma função de processamento de informações útil, deve haver redundância nos padrões de ativação fornecidos à rede pelo ambiente (Haykin, 2001). Aqui, pode-se traçar um paralelo com sistemas vivos: para que exista a vida é necessário que os seres não estejam isolados e que recebam um fluxo de energia coerente do meio ao seu redor.

Os algoritmos de Aprendizagem Competitiva pertencem à classe das redes neurais não-supervisionadas ou de aprendizagem auto-organizada. O objetivo comum desses algoritmos é distribuir certo número de vetores em um espaço de dimensão *n* qualquer. A distribuição desses vetores deve refletir a distribuição de probabilidades dos dados de entrada. Os sinais de entrada pertencem a um conjunto $M \subseteq \Re^n$. Assim, uma rede consiste de um conjunto *A* de *N* unidades:

$$A = \{c_1, c_2, ..., c_N\}.$$

---

1. Se dois neurônios em ambos os lados de uma sinapse (conexão) são ativados simultaneamente (i.e., sincronamente), então a força daquela sinapse é seletivamente aumentada.
2. Se dois neurônios em ambos os lados de uma sinapse (conexão) são ativados assincronamente, então aquela sinapse é seletivamente enfraquecida ou eliminada.

A regra de Hebb foi verificada, por exemplo, no funcionamento de partes do hipocampo (Kelso et al., 1986).

Cada unidade tem um vetor de referência (também chamado de vetor de peso sináptico) associado

$$\mathbf{w}_c \in \Re^n,$$

indicando sua posição no espaço de entrada.

Uma unidade pode estar conectada com outras, definindo um conjunto *C* de conexões[25] de vizinhança

$$C \subset A \times A.$$

Normalmente, não são atribuídos pesos às conexões.

Os sinais de entrada são gerados aleatoriamente, de acordo com uma função densidade de probabilidade contínua

$$p(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \Re^n$$

ou por um conjunto de treinamento finito.

Para cada sinal de entrada **ξ**, a unidade vencedora *s*(**ξ**), definida como a unidade cujo vetor de referência está mais próximo da entrada, sofre uma adaptação: seu vetor de referência se move na direção de **ξ**. O Aprendizado Competitivo deriva seu nome desse processo de competição entre os neurônios para serem ativados. Os vizinhos da unidade vencedora podem ou não ser também ativados. Um algoritmo no qual só o vencedor sofre adaptação é dito ser do tipo **o vencedor leva tudo** (*winner takes all*). Um exemplo de algoritmo desta classe é o k-médias (*k-means*). Quando unidades vizinhas também sofrem alguma adaptação, o tipo do algoritmo é dito **o vencedor leva a maior parte** (*winner takes most*). Em geral, os algoritmos de aprendizado competitivo são deste segundo tipo.

Um exemplo de algoritmo de Aprendizado Competitivo largamente utilizado em aplicações práticas é o Mapa de Características Auto-organizável de Kohonen (SOFM, *self organizing feature map*) (Kohonen, 1982). No algoritmo SOFM, as unidades são distribuídas sobre uma grade bidimensional (normalmente, com vizinhança quadrada ou hexagonal), fixando o conjunto de conexões. O número de unidades (tamanho da grade) é um parâmetro do algoritmo. Os nós da rede se ajustam aos sinais de entrada, aprendendo gradualmente o padrão existente. O ajuste da rede se dá de maneira que padrões semelhantes dos dados de entrada sejam mapeados em nós adjacentes e vice-versa: nós adjacentes codificam padrões semelhantes. Assim, a rede cria um mapeamento Φ de um espaço de entrada *M* de dimensão *n* arbitrariamente

[25] Neste trabalho, as conexões são também denominadas **arestas**.

alta em um grafo *G*. O fato das unidades de *G* estarem dispostas em uma grade de estrutura bidimensional possibilita a visualização da classificação gerada para os dados de entrada. Mapas em uma ou três dimensões também podem ser gerados. A Figura 4-1 ilustra o processo de adaptação de uma rede bidimensional de Kohonen com topologia quadrada.

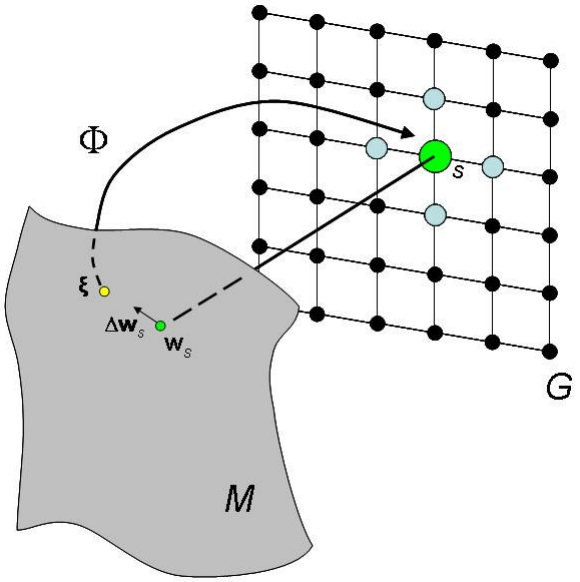

Figura 4-1: Passo de adaptação em uma rede de Kohonen: dado um sinal de entrada **ξ**, é identificada a unidade vencedora *s* e ocorre um deslocamento dos vetores de referência das unidades da vizinhança de *s* em direção a **ξ**. A rede *G* possui uma topologia quadrada.

A forma como se pretende distribuir os vetores de referência das unidades sobre o espaço de entradas pode variar de sistema para sistema. Um objetivo freqüentemente utilizado em sistemas baseados em Aprendizado Competitivo consiste na minimização do valor esperado do erro de quantização, isto é:

$$E(p(\boldsymbol{\xi}), A) = \sum_{c \in A} \int_{V_c} \left\| \boldsymbol{\xi} - \mathbf{w}_c \right\|^2 p(\boldsymbol{\xi}) d\boldsymbol{\xi}$$

onde V_c é a região de Voronoi da unidade *c* (região cujos pontos têm *c* como a unidade mais próxima) e $\|.\|$ é a norma do vetor. Uma aplicação típica em que a minimização do erro é importante é a quantização vetorial. A quantização vetorial permite que dados vetoriais sejam transmitidos por canais de comunicação com largura de banda estreita. Isto é feito transmitindo, para cada dado vetorial, apenas o índice do vetor de referência mais próximo. O conjunto de vetores de referência, chamado de **livro de código** (*codebook*), deve ser conhecido pelo emissor e pelo receptor. Existe uma perda de informação neste processo de

transmissão de dados comprimidos. Mas, nos casos em que os dados são fortemente agrupados, a quantização vetorial permite obter altas taxas de compressão com uma distorção relativamente pequena.

Outra possibilidade é procurar distribuir as unidades de tal forma que, dado um sinal de entrada aleatório **ξ,** cada vetor de referência tenha a mesma chance de ser o vencedor. Nesse caso, a probabilidade de qualquer unidade c_i com vetor $\mathbf{w}_i$ ser a vencedora é:

$$P(s(\boldsymbol{\xi}) = c_i) = \frac{1}{N} \qquad (\forall c_i \in A = \{c_1,...,c_N\})$$

Pode-se interpretar a geração de um sinal de entrada e o seu subseqüente mapeamento na unidade mais próxima em A como um experimento aleatório que atribui um valor $x \in A$ a uma variável aleatória X. No contexto da Teoria da Informação, define-se a entropia da variável aleatória X como:

$$H(X) = -\sum_{x \in A} p(x)\log(p(x)).$$

A entropia é uma medida da "surpresa" ou "informação", quando ocorre um determinado evento $X = x_i$. Numa situação em que $p(x_i) = 1$ (e todos os demais valores têm probabilidade zero), tem-se certeza absoluta do resultado de um evento, e a entropia é nula. No presente caso, tem-se que:

$$H(X) = -\sum_{x \in A} \frac{1}{N}\log(\frac{1}{N}) = -\log(\frac{1}{N}) = \log(N).$$

Pode-se mostrar que esta é a situação de máximo da entropia (Haykin, 2001). Observe que, desta forma, a entropia é máxima na situação em que os possíveis eventos são igualmente prováveis. Tal situação pode ser vista como a de máxima desordem. Assim, o conceito de entropia da Teoria da Informação está em acordo com o conceito da Termodinâmica.

A maximização da entropia e a minimização do erro em geral levam a configurações diferentes, principalmente se a distribuição de probabilidades é fortemente não-uniforme. Considere a situação da Figura 4-2, onde 50% dos sinais de entrada vêm de uma região muito pequena (azul escuro), enquanto os outros 50% estão distribuídos uniformemente sobre uma região muito maior (azul claro). No caso de maximização da entropia, metade das unidades deve ser disposta em cada região (Figura 4-2, lado esquerdo), enquanto para o caso da minimização do erro de quantização basta alocar uma única unidade na região pequena e distribuir uniformemente todas as demais unidades (Figura 4-2, lado direito).

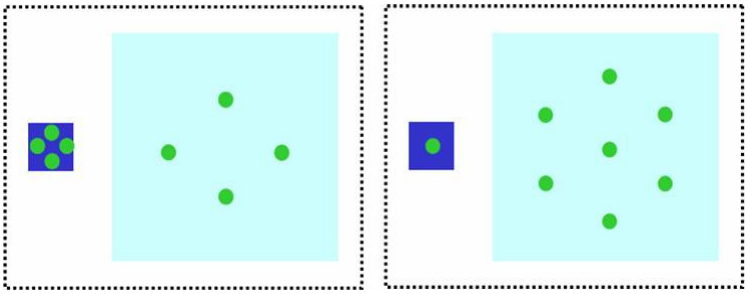

Figura 4-2: Comparação entre a maximização da entropia (à esquerda) e a minimização do erro de quantização (à direita). Nos dois casos, cada uma das duas regiões coloridas de azul escuro e de azul claro tem 50% dos sinais de entrada.

4.2. Mapas que preservam a topologia

Nos mapas de Kohonen a topologia do grafo, isto é, sua estrutura de conectividade, é fixa desde o início do processo de aprendizado da rede neural. A questão que se coloca é se essa forma de organizar o grafo ou rede é sempre aceitável. As arestas do grafo fornecem informações de vizinhança entre os nós; assim, a pergunta pode ser refeita como: se dois nós são adjacentes no grafo, seus vetores de referência são também vizinhos no espaço n-dimensional de entrada? E vice-versa: se dois vetores de referência são vizinhos no espaço de entrada seus nós estarão conectados no grafo? Que condições o grafo deve satisfazer para permitir este comportamento, dada uma determinada distribuição de probabilidades? Estas questões compõem o problema de "preservação de topologia" e foram abordadas e respondidas por Martinetz e Shulten (Martinetz, 1993; Martinetz et al., 1994). Esta seção faz uma apresentação resumida das idéias envolvidas nesses trabalhos, suprimindo demonstrações dos teoremas que estão disponíveis em Martinetz e Shulten (1994).

Considere um grafo *G*, onde cada nó *i* possui um vetor de referência $\mathbf{w}_i$. A distribuição de probabilidade P($\boldsymbol{\xi}$) dos padrões do espaço de entrada não tem suporte sobre todo o $\Re^n$, mas somente sobre uma variedade *M*:

$$M \subseteq \Re^n.$$

O mapeamento Φ de *M* em *G* é determinado pelo conjunto $S = \{\mathbf{w}_1,...,\mathbf{w}_N\}$ dos *N* vetores de referência, de forma que dado um vetor $\boldsymbol{\xi} \in M$ este é mapeado no vértice $i^*(\boldsymbol{\xi})$, cujo vetor $\mathbf{w}_{i^*(\boldsymbol{\xi})}$ é o mais próximo de $\boldsymbol{\xi}$

$$\Phi_S : M \mapsto G,$$
$$\boldsymbol{\xi} \in M \mapsto i^*(\boldsymbol{\xi}) \in G$$

$$\left\| \mathbf{w}_{i^*(\boldsymbol{\xi})} - \boldsymbol{\xi} \right\| \leq \left\| \mathbf{w}_i - \boldsymbol{\xi} \right\| \quad \forall i \in G$$

O mapeamento Φ_s preserva a vizinhança se vetores $\mathbf{w}_i$, $\mathbf{w}_j$ que são vizinhos na variedade *M* são atribuídos a vértices *i*, *j* que são adjacentes no grafo *G*.

O mapeamento inverso é determinado por:

$$\Phi_S^{-1} : G \mapsto M,$$
$$i \in G \mapsto \mathbf{w}_i \in M$$

Analogamente, Φ_s^{-1} preserva a vizinhança se os vértices *i*, *j* adjacentes em *G* são mapeados em vetores $\mathbf{w}_i$, $\mathbf{w}_j$ vizinhos em *M*.

O grafo G com um conjunto de vetores de referência *S* forma um mapeamento de *M* que **preserva perfeitamente a topologia** se tanto o mapeamento Φ_s de *M* em *G*, como Φ_s^{-1} de *G* em *M* preservarem vizinhança.

Observe a Figura 4-3 abaixo, onde são consideradas três diferentes topologias para o grafo da rede. Nos três casos, a variedade *M* é bidimensional e está definida apenas sobre um quadrado. Em cada um dos três casos, o grafo *G* é composto de nove vértices, cujos vetores de referência são distribuídos regularmente sobre a variedade. Na Figura 4-3(a) o grafo tem uma topologia unidimensional. Neste caso, *G* é incapaz de gerar um mapeamento que preserve a topologia de *M*. O máximo que se consegue é preservar a vizinhança no mapeamento inverso Φ_s^{-1} de *G* em *M*. Na Figura 4-3(b) tem-se o caso oposto: a dimensão de *G* é maior do que a de *M*. O resultado é que é possível ter algum ganho na preservação da vizinhança pelo mapeamento Φ_s de *M* em *G*, mas com perda da preservação para a inversa Φ_s^{-1} de *G* em *M*. Na Figura 4-3(c) a topologia do grafo é uma rede quadrada e corresponde à topologia da variedade *M*. Somente neste último caso, *G* forma um mapeamento de *M* que preserva a topologia, isto é, tanto Φ_s como sua inversa Φ_s^{-1} preservam a vizinhança.

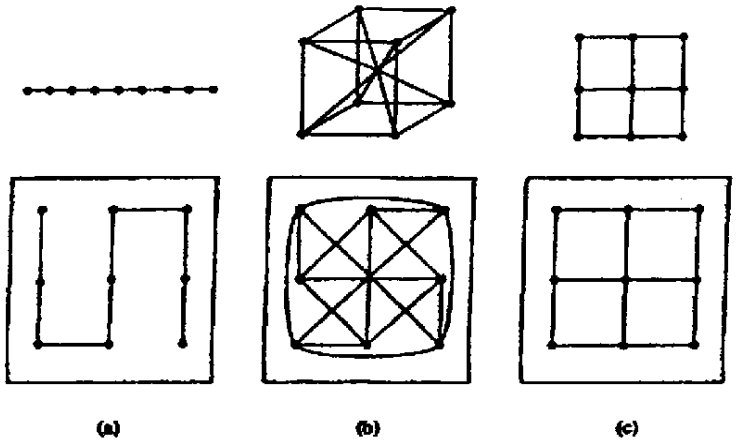

Figura 4-3: Um grafo *G* de nove vértices é mapeado em uma variedade *M* de formato quadrado. No caso (a), o grafo tem topologia unidimensional, em (b) tridimensional e em (c) bidimensional. Adaptada de Martinetz et al. (1994)

O conceito de adjacência em grafos é claramente estabelecido: dois nós são adjacentes se são ligados por uma aresta do grafo. Por outro lado, embora intuitiva, falta ter uma definição formal de vizinhança para a variedade *M*. Para tal, Martinetz e Shulten fazem uso de poliedros de Voronoi[26] do conjunto *S* de vetores de referência. No contexto de Aprendizado Competitivo tipo o vencedor leva tudo, um poliedro de Voronoi V_i é também conhecido como campo receptivo da unidade *i*. Assim, dois vetores $\mathbf{w}_i$, $\mathbf{w}_j$ são vizinhos se seus poliedros de Voronoi, V_i, V_j, são adjacentes, isto é, $V_i \cap V_j \neq \emptyset$. Entretanto, poliedros de Voronoi são definidos sobre todo o $\Re^n$ e como se deseja definir vizinhança de pontos apenas na variedade *M*, é introduzido o conceito de **poliedro de Voronoi restrito** (*masked Voronoi polyhedron*):

$$V_i^{(M)} = V_i \cap M .$$

Assim, define-se que dois vetores $\mathbf{w}_i, \mathbf{w}_j \in M \subseteq \Re^n$ são adjacentes em *M* se seus poliedros de Voronoi restritos também são adjacentes, isto é, se $V_i^{(M)} \cap V_j^{(M)} \neq \phi$ é válido.

[26] O diagrama de Voronoi V_S de um conjunto $S=\{\mathbf{w}_1,...,\mathbf{w}_N\}$ de pontos $\mathbf{w}_i, \mathbf{w}_j \in \Re^n$ é dado por *N* poliedros n-dimensionais, os poliedros de Voronoi. Um poliedro de Voronoi V_i de um ponto $\mathbf{w}_i \in S$ é dado pelo conjunto de pontos $\mathbf{v} \in \Re^n$ que estão mais próximos de $\mathbf{w}_i$ do que qualquer outro $\mathbf{w}_j \in S$, $j \neq i$. (O´Rourke, 1993)

A definição de vizinhança em *M* com o uso dos poliedros de Voronoi restritos formaliza a definição de mapeamento que preserva a topologia apresentada anteriormente, nesta seção.

A triangulação de Delaunay é, por definição, o grafo dual da partição do $\Re^n$ formada pelos poliedros de Voronoi (diagrama de Voronoi) (O´Rourke, 1993). O grafo dual associa um nó para cada poliedro e possui uma aresta ligando cada par de nós cujos poliedros são vizinhos. Analogamente aos poliedros de Voronoi restritos, é definida a **triangulação de Delaunay induzida** (*induced Delaunay triangulation*) $D_S^{(M)}$ de *S*, dado *M*, como sendo o grafo que conecta dois pontos $\mathbf{w}_i$, $\mathbf{w}_j$ de *S* se e somente se seus poliedros de Voronoi restritos forem adjacentes, isto é, a matriz de adjacência **R** do grafo tem a propriedade

$$\mathbf{R}_{ij} = 1 \Leftrightarrow V_i^{(M)} \bigcap V_j^{(M)} \neq \phi \,.$$

Na Figura 4-4, é exibida uma comparação entre a triangulação de Delaunay, à esquerda, e a triangulação de Delaunay induzida, à direita, num grafo bidimensional. Observe que para existir uma aresta entre dois nós na triangulação de Delaunay induzida é necessário que a fronteira entre os polígonos de Voronoi desses nós estejam pelo menos parcialmente encoberta pela variedade (exibida, na Figura 4-4, com sombreado).

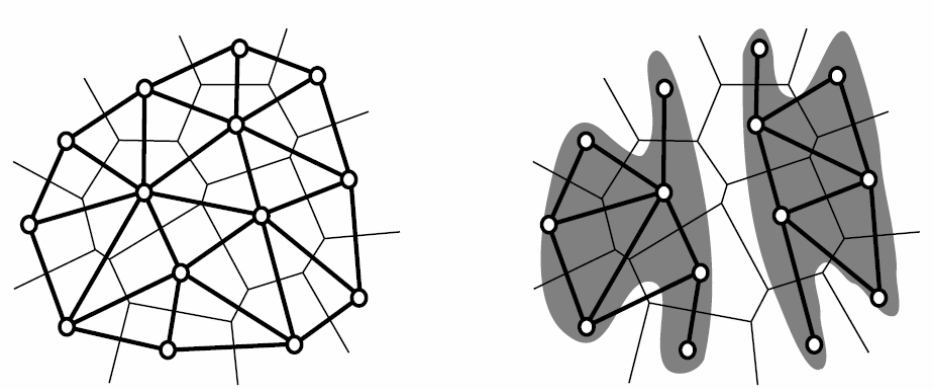

Figura 4-4: À esquerda, Triangulação de Delaunay (linhas grossas) e correspondentes polígonos de Voronoi (linhas finas). À direita, triangulação de Delaunay induzida (linhas grossas) e polígonos de Voronoi restritos (linhas finas) definidos sobre uma variedade *M* (região sombreada). Adaptada de Martinetz e Schulten (1994).

Das definições anteriores, é possível mostrar o seguinte resultado (Martinetz e Shulten, 1994):

Teorema 1. O grafo *G*, cujos vértices formam o conjunto *S* de pontos na variedade *M*, forma um mapeamento de *M* que preserva perfeitamente a topologia se e somente se o grafo *G* for a triangulação de Delaunay induzida $D_S^{(M)}$ de *S*.

Os autores também propõem um algoritmo competitivo para, dado um conjunto de vértices *S*, obter um grafo com preservação de topologia. Para tal, introduzem o conceito de poliedro de Voronoi de segunda ordem. O poliedro de Voronoi de segunda ordem V_{ij} é formado por todos os pontos **v** para os quais $\mathbf{w}_i$, $\mathbf{w}_j$ são os pontos de *S* mais próximos; isto é

$$V_{ij} = \{\mathbf{v} \in \Re^n \mid \|\mathbf{v} - \mathbf{w}_i\| \leq \|\mathbf{v} - \mathbf{w}_k\| \wedge \|\mathbf{v} - \mathbf{w}_j\| \leq \|\mathbf{v} - \mathbf{w}_k\| \forall k \neq i, j\} .$$

Os autores mostram que:

Teorema 2: Seja $S = \{\mathbf{w}_1,...,\mathbf{w}_N\}$ um conjunto de pontos do $\Re^n$, V_i o poliedro de Voronoi do ponto $\mathbf{w}_i$ e V_{ij} o poliedro de Voronoi de segunda ordem dos pontos $\mathbf{w}_i$, $\mathbf{w}_j$. Então a relação abaixo é válida para qualquer par *i*, *j*:

$$V_i \cap V_j \neq \phi \Leftrightarrow V_{ij} \neq \phi .$$

O Teorema 2 permite estabelecer uma regra de criação de arestas muito simples: para cada sinal de entrada, ligar os dois vetores de referência mais próximos com uma aresta. Martinetz (1993) chama esta regra de **Aprendizado Competitivo Hebbiano** (CHL – *Competitive Hebbian Learning*)[27].

Neste ponto cabe verificar em que medida o CHL consegue gerar corretamente a triangulação de Delaunay induzida. Para entender a limitação do CHL, considere a Figura 4-5 que, à semelhança da anterior, exibe um conjunto de nós, o diagrama de Voronoi (linhas finas) e a triangulação de Delaunay induzida (linhas grossas) para uma distribuição de probabilidade (região sombreada). A cruz verde identifica um sinal de entrada obtido. O nó amarelo é o mais próximo e o azul o segundo mais próximo. Seguindo a regra de CHL, os dois nós devem ser ligados. Entretanto, neste caso, os poliedros de Voronoi restritos não são adjacentes. A explicação para isto é que na vizinhança desse

[27] A proposta do CHL foi inspirada no postulado de aprendizagem de Hebb. De uma forma simplificada pode-se dizer que a criação de uma aresta corresponde ao incremento na intensidade da ligação sináptica. Em Martinetz (1993) é apresentada uma derivação da regra CHL a partir de uma formulação quantitativa do postulado de Hebb.

sinal de entrada os nós da rede não são suficientemente densos para evitar a criação de uma aresta ligando nós não vizinhos, segundo a distribuição de probabilidades.

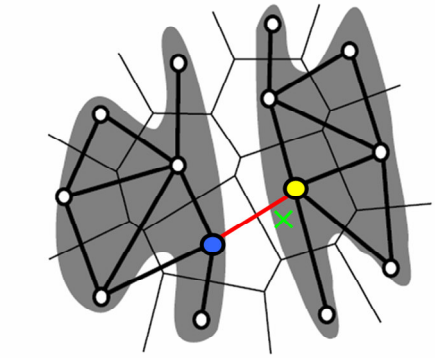

Figura 4-5: CHL pode gerar aresta entre nós cujos poliedros de Voronoi restritos não são vizinhos, se a distribuição de vértices não é densa.

O conceito de conjunto de pontos denso pode ser definido formalmente da seguinte maneira: um conjunto de pontos $S = \{\mathbf{w}_1,...,\mathbf{w}_N\}$ sobre uma variedade $M \subseteq \Re^n$ é **denso** em M se, para cada $\mathbf{v} \in M$, o triângulo $\Delta(\mathbf{w}_{io},\mathbf{w}_{i1},\mathbf{v})$ formado pelo ponto de S mais próximo, pelo segundo mais próximo e pelo próprio ponto se encontra completamente em M, isto é, se $\Delta(\mathbf{w}_{io},\mathbf{w}_{i1},\mathbf{v}) \subseteq M$ é válido.

De posse deste último conceito pode-se finalmente enunciar o resultado final desta discussão sobre construção de grafos que preservam a topologia:

Teorema 3: Seja $S = \{\mathbf{w}_1,...,\mathbf{w}_N\}$ um conjunto de pontos distribuídos sobre uma dada variedade M. Se S é denso em M, então o grafo G formado pela regra CHL é a triangulação de Delaunay induzida $D_S^{(M)}$ de S e, então G forma um mapeamento de M que preserva perfeitamente a topologia.

4.3. Growing Neural Gas

A regra CHL permite construir o conjunto de arestas, dado um conjunto de vértices. Para construir o conjunto de vértices, por sua vez, Martinetz e Shulten (1994) utilizaram o algoritmo de quantização vetorial **Neural Gas** (NG), proposto anteriormente pelos mesmos autores (Martinetz e Shulten, 1991). Entretanto, um dos inconvenientes desse algoritmo é que o número de unidades deve ser informado previamente. Por sua vez, Fritzke (1994) havia proposto o algoritmo

Growing Cell Structures (GCS) como uma alternativa para o SOFM, em que o número de unidades não é fixo, mas varia ao longo da execução do algoritmo (ver seção 6.2). Assim, o Growing Neural Gas (GNG) (Fritzke, 1995, Fritzke, 1997, Holmström, 2002) surgiu da combinação da regra CHL de criação de arestas com o mecanismo de criação de vértices do Growing Cell Structures.

No início da execução do GNG, o grafo (rede) é composto por apenas duas unidades. Para determinar onde inserir novas unidades, medidas de erro local são coletadas durante o processo de adaptação. Periodicamente, uma nova unidade é inserida, próxima à unidade que acumulou, até então, o maior valor de erro. A inserção é feita dividindo a aresta que liga esta unidade à unidade vizinha cujo erro é maior dentre todas as vizinhas. Fritzke (1995) propôs o quadrado da distância entre o sinal de entrada e a unidade vencedora como medida do erro.

A conectividade é definida pela regra CHL. Para cada sinal de entrada gerado aleatoriamente, uma aresta deve ser criada entre a unidade vencedora e a segunda-vencedora, caso esta conexão ainda não exista.

Além disso, os vetores de referência do vencedor e de seus vizinhos são adaptados, deslocando-se na direção do sinal de entrada, por frações dos seus vetores-diferença (em relação à posição do sinal de entrada). Todos os parâmetros do GNG são fixos durante a execução do algoritmo (o que não ocorre com SOFM e NG).

As medidas de erro local coletadas durante o processo de adaptação informam sobre o nível de atividade das unidades. Quanto maior o erro acumulado maior a atividade da unidade e, por conseguinte, mais necessária é a inserção de uma nova unidade em sua vizinhança. Entretanto, erros identificados há muitos ciclos atrás não devem ser considerados integralmente, pois foram gerados quando o número de unidades era menor do que o atual. Assim, de alguma forma os erros gerados nos ciclos recentes devem ter um peso maior do que os mais antigos. Esse efeito é obtido diminuindo de um percentual o valor de erro local de cada unidade. O decremento é executado a cada ciclo.

Por outro lado, tanto as arestas como as unidades podem ser destruídas ao longo da execução do algoritmo GNG. Cada aresta possui um atributo de idade. A cada geração de um novo sinal de entrada, a idade da aresta que liga a unidade vencedora à segunda-vencedora é zerada, enquanto que as demais arestas ligadas à unidade vencedora têm suas idades incrementadas. Arestas envelhecidas (com idade maior do que um valor parametrizado) são removidas. Caso isso resulte em unidades sem aresta, essas unidades também são

removidas. Para compreender o porquê deste comportamento, observe que a existência de uma aresta significa que existiu ao menos um sinal de entrada que teve as unidades ligadas por essa aresta como vencedora e segunda-vencedora. Mas isso pode ter acontecido há muito tempo e, se a configuração em torno dessa aresta sofreu muitas modificações com a inclusão de novas unidades e arestas, pode ser que essa aresta não seja mais útil. Esta regra corrige situações anômalas nas quais uma aresta antiga cruza arestas mais jovens (e de tamanhos menores). Além disso, o aumento do número de vértices em uma certa região pode ter tornado os vértices densos localmente; neste caso, uma aresta ligando vértices cujos poliedros de Voronoi restritos não são vizinhos tende também a ser descartada, pois deixará de ser ativada (ver Figura 4-5). Assim, a regra da idade funciona como um coletor de defeitos derivados da combinação de CHL com a dinâmica de criação e adaptação dos vértices.

A Figura 4-6 reproduz o resultado obtido por Fritzke (1995) com a utilização de GNG a partir de dados de entrada com uma distribuição homogênea sobre três tipos diferentes de regiões: uma região com estrutura tridimensional (paralelepípedo), uma bidimensional (retângulo) e duas unidimensionais (um círculo e uma linha). No início a rede é composta de apenas dois vértices, escolhidos aleatoriamente[28]. De cima para baixo, da esquerda para a direita, são exibidas as redes com 8, 20, 52, 152 e 202 vértices. Observe que a estrutura da rede gerada acompanha corretamente a estrutura dos dados de entrada.

[28] Nesta versão do GNG, os vértices iniciais não foram gerados utilizando a distribuição de probabilidades dos dados de entrada $p(\boldsymbol{\xi})$.

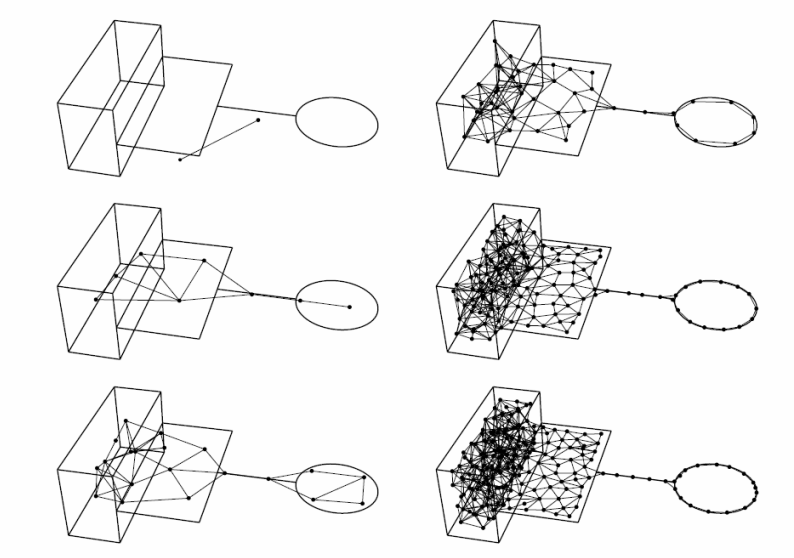

Figura 4-6: GNG cria um grafo que se adapta a uma distribuição de sinais com diferentes topologias. Os dados de entrada são distribuídos em três subconjuntos: com estrutura tridimensional (paralelepípedo), bidimensional (retângulo) e unidimensional (círculo e linha). Adaptada de Fritzke (1995).

O algoritmo GNG é apresentado completo a seguir, conforme descrito por Fritzke (1997):

Algoritmo GNG

A entrada é um volume de probabilidades *p*. A saída é um conjunto de unidades *A*, onde cada unidade tem um peso **w** e um conjunto de conexões *C*.

1. Iniciar conjunto *A* para conter duas unidades c_1 e c_2, com os vetores de referência escolhidos aleatoriamente de acordo com $p(\boldsymbol{\xi})$. Iniciar o conjunto de conexões *C*, $C \subset AxA$, como vazio.
2. Gerar aleatoriamente um sinal de entrada $\boldsymbol{\xi}$ de acordo com $p(\boldsymbol{\xi})$.
3. Determinar a unidade vencedora s_1 e a segunda-vencedora s_2 por

$$s_1 = \arg\min_{c \in A} \left\| \boldsymbol{\xi} - \mathbf{w}_c \right\|$$

$$s_2 = \arg\min_{c \in A \setminus \{s_1\}} \left\| \boldsymbol{\xi} - \mathbf{w}_c \right\|$$

4. Se uma conexão entre s_1 e s_2 ainda não existe, então criar a conexão. Zerar a idade da conexão entre s_1 e s_2.
5. Atualizar a variável de erro local da unidade vencedora, somando o quadrado da distância entre a unidade e o sinal de entrada.

6. Adaptar os vetores de referência do vencedor e dos vizinhos topológicos diretos (conjunto N_{s_1}) por frações ε_b e ε_n, respectivamente, da distância total para o sinal de entrada:

$$\Delta \mathbf{w}_{s_1} = \varepsilon_b(\boldsymbol{\xi} - \mathbf{w}_{s_1})$$
$$\Delta \mathbf{w}_i = \varepsilon_n(\boldsymbol{\xi} - \mathbf{w}_i) \quad (\forall i \in N_{s_1})$$

7. Incrementar de uma unidade as idades de todas as arestas partindo de s1.
8. Remover as arestas com idade maior do que a_{max}. Se isto resultar em nós sem aresta, remover estes nós, também.
9. Se o número de sinais de entrada gerados até o momento for um múltiplo inteiro de um parâmetro λ, inserir uma nova unidade da seguinte maneira:
   - Determinar a unidade *q* com o maior erro acumulado.
   - Determinar dentre os vizinhos de *q*, a unidade *f* de maior erro acumulado.
   - Adicionar um novo vértice *r* e interpolar seu vetor de referência a partir dos vetores de *q* e *f*.

$$A = A \cup \{r\}, \quad \mathbf{w}_r = (\mathbf{w}_q + \mathbf{w}_f)/2.$$

   - Inserir arestas conectando *r* com *q* e *f* e remover a original que conectava *q* com *f*.
   - Diminuir as variáveis de erro de *q* e *f* por uma fração:

$$\Delta \mathrm{E}_q = -\alpha \mathrm{E}_q, \quad \Delta \mathrm{E}_f = -\alpha \mathrm{E}_f.$$

   - Interpolar a variável de erro de *r* a partir de *q* e *f*:

$$\mathrm{E}_r = (\mathrm{E}_q + \mathrm{E}_f)/2$$

10. Diminuir a variável de erro de todas as unidades:

$$\Delta \mathrm{E}_c = -\beta \mathrm{E}_c$$

11. Se um critério de parada (por exemplo: tamanho da rede ou alguma medida de desempenho) não tiver sido alcançado, voltar ao passo 2.

A seguir é apresentada uma tabela com os parâmetros do GNG, seus respectivos significados e valores típicos:

Parâmetro	Descrição	Valores típicos
λ	Intervalo em ciclos entre 2 inserções de nós	300
ε_b	Fator de adaptação do nó vencedor	0.05
ε_n	Fator de adaptação dos nós vizinhos ao venc.	0.0006
α	Fator de decremento dos erros na inserção	0.5
β	Fator de depreciação dos erros a cada ciclo	0.0005
a_{max}	Idade máxima de uma aresta do nó vencedor	88 - 44

Tabela 1: Parâmetros do Growing Neural Gas.

4.4. Aceleração do GNG

O algoritmo GNG descrito na seção anterior pode ser dividido em duas etapas, como a seguir:

Aprendizado: A cada iteração ou ciclo, um sinal de entrada é obtido, o nó vencedor e o segundo-vencedor sofrem uma adaptação e o erro do vencedor é incrementado. As idades das arestas incidentes ao nó vencedor são incrementadas e as arestas velhas são removidas. Por fim, os erros de todos os nós sofrem uma depreciação. Esta etapa corresponde aos passos 2, 3, 4, 5, 6, 7, 8 e 10 da descrição detalhada.

Crescimento: Após λ ciclos, um novo nó é inserido, dividindo a aresta que liga o nó de maior erro do grafo a um de seus vizinhos. Esta etapa corresponde ao passo 9 da descrição detalhada.

Embora seja possível a remoção de nó (no caso em que todas as arestas do nó tenham sido eliminadas pelo critério de idade), em geral, pode-se assumir que o número de nós n é proporcional ao número de ciclos n_c:

$$n = 1/\lambda \cdot n_c .$$

Assim, tanto a etapa de Aprendizado como a de Crescimento são invocadas $O(n)$ vezes.

Na etapa de Aprendizado, os passos 3 (identificação dos nós vencedor e segundo-vencedor) e 10 (depreciação do erro de todos os nós) têm custo computacional de $O(n)$. Os demais passos desta etapa têm custo $O(1)$[29].

[29] Está sendo assumido que o grafo é sempre esparso, de forma que o número máximo de arestas de um nó pode ser majorado por uma constante.

O passo 9 da etapa de Crescimento também tem custo computacional de $O(n)$, devido à busca seqüencial do nó de maior erro.

Desta forma, o algoritmo GNG como descrito por Fritzke (1995, 1997) (ver seção anterior) tem desempenho computacional $O(n^2)$, onde n é o número de unidades do grafo.

O problema do passo 10 pode ser facilmente remediado mediante o adiamento da diminuição do erro de todas as unidades[30]. O erro de cada unidade só precisa ser atualizado quando da busca da unidade de maior erro (início do passo 9), o que só é feito de λ em λ ciclos do algoritmo. Para tal, é necessário guardar na estrutura de dados de cada unidade, além do erro da unidade, a informação do número do ciclo em que ocorreu a última atualização desse erro. Essa informação é iniciada quando da criação do nó (final do passo 9) e é atualizada a cada ciclo em que a unidade em questão é a vencedora (passo 3):

$$\Delta Ciclos = CicloCor - Ciclo_{s_1}$$
$$\mathrm{E}_{s_1} = (1-\beta)^{\Delta Ciclos}\,\mathrm{E}_{s_1} + \left\|\boldsymbol{\xi} - \mathbf{w}_{s_1}\right\|^2.$$

Na expressão acima *CicloCor* é o número do ciclo corrente, Ciclo_{s1} é o ciclo da última atualização do erro da unidade vencedora s_1 e E_{s1} é o erro então calculado. No início do passo 9, antes de calcular a unidade de maior erro, todas as unidades c têm seus erros atualizados:

$$\Delta Ciclos = CicloCor - Ciclo_c$$
$$\mathrm{E}_c = (1-\beta)^{\Delta Ciclos}\,\mathrm{E}_c$$

Esta modificação torna o passo 10 $O(1)$, sem alterar a complexidade do passo 9.

O problema da pesquisa da unidade mais próxima (início do passo 3) pode ser tratado com a utilização de uma estrutura de dados espacial adaptativa. No caso de dados de entrada definidos no $\Re^3$, pode-se utilizar *octree*. Nessa estrutura de dados em forma de árvore, cada nó da árvore define um cubo que contém as unidades do grafo dessa região do espaço. Um nó da árvore é subdividido em oito partes, recursivamente quando o número de unidades do grafo excede uma dada constante. Já que as unidades mudam de posição (passo 6), os nós da árvore devem ser atualizados a cada ciclo. A inserção e remoção de uma unidade é feita apenas no nível folha da árvore, e a atualização

[30] Essa estratégia de adiamento da depreciação do erro proposta nesta tese para o algoritmo de GNG havia sido utlizada anteriormente no contexto de outro algoritmo de Aprendizado Competitivo, o Neural Meshes (Ivirissimtzis et al., 2003b) (ver seção 6.4).

da árvore é realizada em tempo *O*(log *n*) onde *n* é o número final de unidades do grafo.

Uma complicação adicional na utilização de *octree* no GNG é que, para que ela seja realmente vantajosa, é necessário que, dado um sinal de entrada, seja identificado o nó da árvore correspondente a esse sinal, mas também todos os nós da árvore vizinhos a esse nó. Existem na literatura vários algoritmos de busca de vizinhança em *octrees* (e *quadtrees*) (Bhattacharya, 2001).

Embora a solução por *octree* resolva corretamente o problema descrito acima, nesta tese optou-se pela utilização de uma estrutura de dados alternativa, devido à sua maior simplicidade: a grade regular (*regular grid* ou *bucketing*) (Heckbert, 1997). A idéia básica é dividir o espaço em células iguais no formato de paralelepípedo, cada uma consistindo de uma lista de unidades do grafo. A pesquisa da unidade mais próxima consiste em identificar a célula correspondente ao sinal de entrada, e a partir daí varrer as cascas esféricas em torno dessa célula até achar as duas unidades mais próximas. Já que estas pesquisas são esféricas, as células devem ser cúbicas e não alongadas. Devido à regularidade da grade a pesquisa das células vizinhas é trivial. Além disso, como as unidades são criadas, destruídas e se movem ao longo do tempo de execução do algoritmo, a grade deve ser dinâmica. Para acelerar a indexação o número de índices ao longo de cada eixo foi tomado como potência de dois. Assim, a grade tem $2^k x 2^k x 2^k$ células, para algum valor de *k* pequeno. A potência aumenta assim que alguma célula ultrapasse um número limite de unidades. Nos testes efetuados, foi utilizado o valor de 20 unidades como limite.

Foram realizados alguns testes para avaliar o efeito do uso de uma grade regular dinâmica no GNG. Em todos os testes foi também empregada a estratégia de adiamento da depreciação do erro dos nós. O gráfico da Figura 4-7 a seguir mostra o desempenho em tempo contra o número de unidades para o GNG nos casos sem utilização da grade dinâmica (linha vermelha) e com utilização de grade dinâmica (linha azul). O comportamento quadrático no caso sem grade é verificado, bem como um comportamento que se aproxima do linear no caso com grade. Por outro lado, o gráfico da

Figura 4-8 mostra que para números pequenos de unidades (até aproximadamente 180 unidades) o caso com grade tem um desempenho inferior ao sem grade. Isso se deve ao gasto de tempo associado ao gerenciamento da estrutura de dados que, para poucas unidades, suplanta o eventual ganho nas pesquisas da unidade mais próxima. Os gráficos da Figura 4-9 e da Figura 4-10 mostram o desempenho do GNG com a grade dinâmica, identificando também

os números de unidades com os quais a grade sofre um aumento no número de células (exibido em número de bits por dimensão). Os gráficos mostram que o desempenho do GNG com grade não é exatamente linear. A cada aumento no número de bits ocorre uma “reinicialização” do comportamento quadrático.

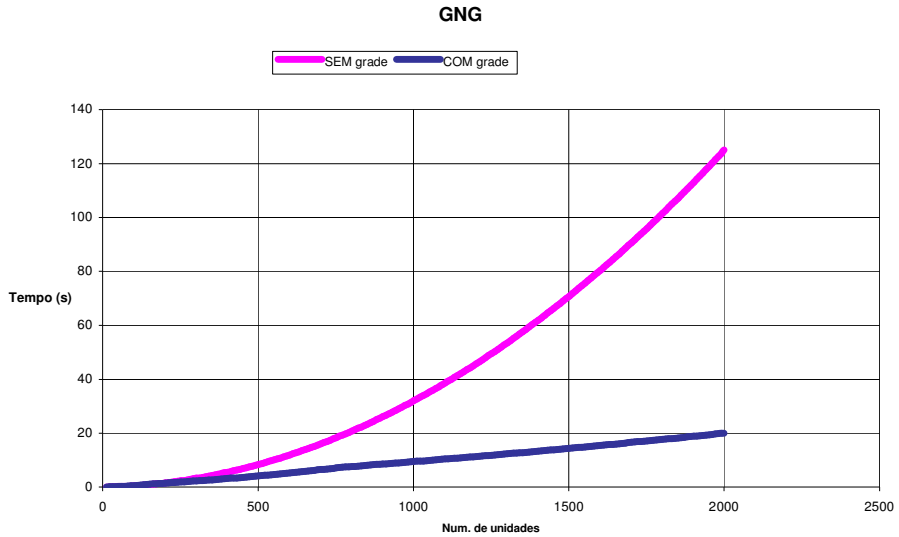

Figura 4-7: Desempenho do GNG com e sem a utilização da grade dinâmica.

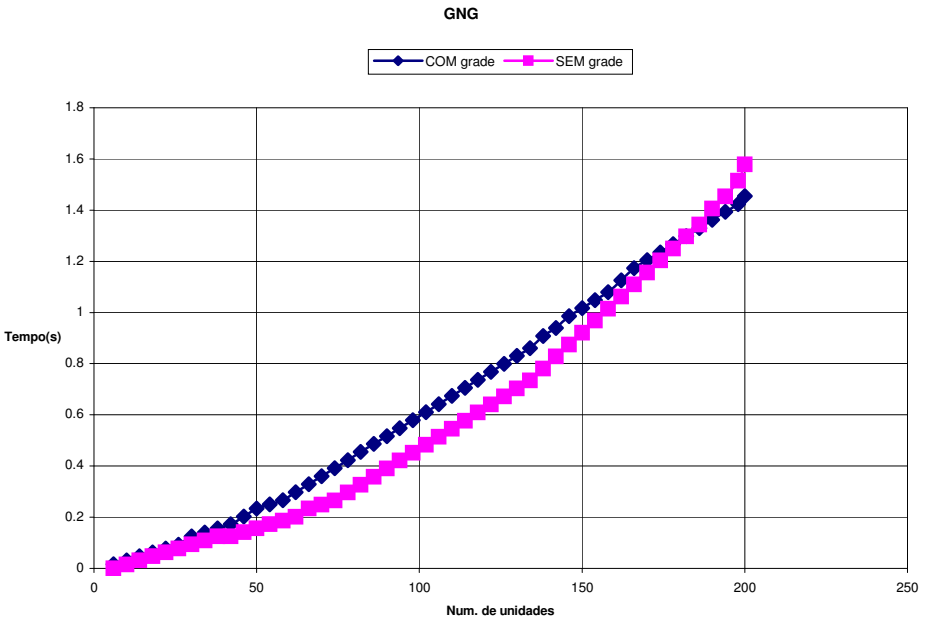

Figura 4-8: Desempenho do GNG com e sem grade dinâmica para poucas unidades.

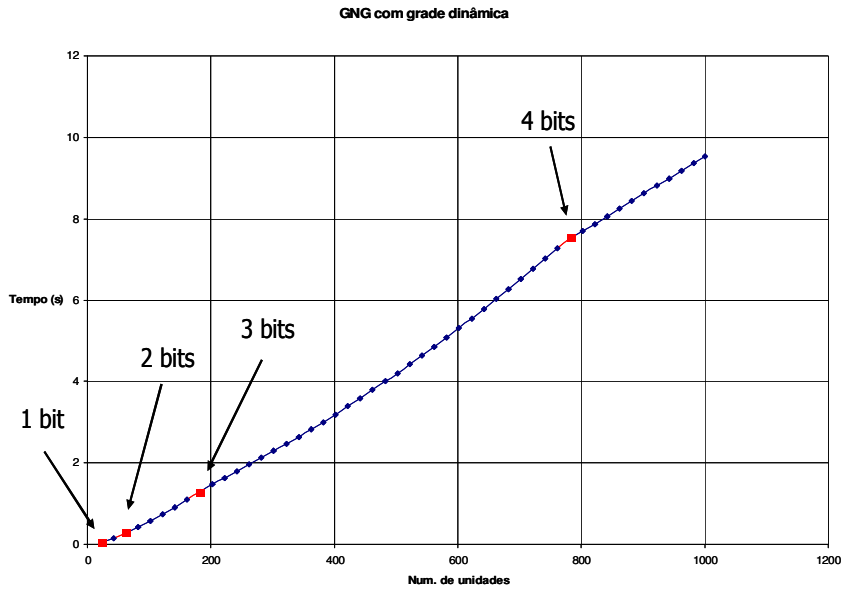

Figura 4-9: Desempenho do GNG com grade dinâmica, exibindo os pontos em que ocorre aumento no número de células de um a quatro bits.

Figura 4-10: Desempenho do GNG com grade dinâmica, exibindo os pontos em que ocorre aumento no número de células de três a seis bits.

A utilização da grade dinâmica diminui o esforço computacional da pesquisa dos nós vencedor e segundo-vencedor. Assim, a aceleração do algoritmo GNG foi obtida com o confinamento do comportamento quadrático à

etapa de Crescimento que é executada λ vezes menos do que a etapa de Aprendizado.

Por outro lado, a dificuldade de se acelerar a pesquisa do nó de maior erro reside no fato de que o erro de todos os nós deve ser depreciado a cada ciclo. O raciocínio a seguir ilustra esta afirmativa: em princípio, se poderia pensar em utilizar uma lista de prioridades como estrutura de dados auxiliar para a pesquisa do nó de maior erro. A Tabela abaixo exibe os custos computacionais das operações de uma lista de prioridades implementada via *heap* (Szwarcfiter et al., 1994):

Operação	**Custo**
Seleção	$O(1)$
Inserção	$O(\log n)$
Remoção	$O(\log n)$
Alteração	$O(\log n)$
Construção	$O(n)$

Tabela 2: Custo computacional das operações de uma lista de prioridades implementada via *heap*.

Com a criação de um nó, um novo elemento é inserido na lista; quando um nó é o vencedor, seu erro é atualizado o que implica na alteração do elemento correspondente da lista. A remoção de um nó implica na remoção do elemento correspondente. A identificação do nó de maior erro passa a ter um custo constante com a operação de seleção no *heap*. Entretanto, a depreciação do erro de todos os nós continua tendo um custo $O(n)$. Nessa depreciação (regra 10 do algoritmo GNG), o erro de cada unidade é multiplicado por uma mesma constante; embora essa operação mantenha inalteradas as posições na fila de prioridades, é necessário varrer toda a fila. Assim, a aceleração, neste caso, só seria possível com uma alteração do algoritmo GNG. Este problema é comum a outros algoritmos competitivos evolutivos, como será visto no Capítulo 6.

4.5. Usos do GNG

Nesta seção são consideradas duas maneiras de se classificar aplicações do Growing Neural Gas: segundo a forma como o grafo é utilizado na solução do problema e segundo a forma como os sinais de entrada são obtidos.

As aplicações de Aprendizado Competitivo, e em particular do Growing Neural Gas, podem ser divididas em duas classes, conforme o tipo de utilização do grafo gerado. Na primeira classe, apenas os vetores de referência dos nós do grafo obtido com o GNG são importantes para a aplicação, enquanto as arestas, após a geração do grafo, são irrelevantes. Pertencem a essa classe de problemas os que utilizam o GNG como um método de quantização vetorial. Outro tipo de problema desta classe é o que utiliza o GNG como método de agrupamento. Nesse caso, ao fim da execução, cada agrupamento é identificado pelo conjunto dos sinais do espaço de entrada que correspondem a uma mesma unidade, isto é, pertencem à região de Voronoi de uma mesma unidade[31]. Como exemplo de aplicação desta classe de problemas, pode-se citar o trabalho apresentado na dissertação de Mestrado de Aurélio Figueiredo (2007), onde o GNG é utilizado na extração automática de horizontes sísmicos. Em um dos exemplos da técnica proposta por Figueiredo, dados definidos no $\Re^{19}$ são quantizados em um livro de códigos com 88 códigos.

A segunda classe de aplicações do Growing Neural Gas utiliza toda a estrutura do grafo gerado, tanto o conjunto de nós (com seus respectivos vetores de referência) como o de arestas. Nessa classe de problemas o GNG é utilizado explicitamente como um extrator de topologia. Como exemplo, pode-se citar o trabalho de Rodriguez et al. (2006), em que o GNG é utilizado na extração de uma seqüência de pontos de referência de imagens bidimensionais de mãos. Dada uma imagem $I(x,y) \in \Re$ de uma mão, é realizada a transformação $\Psi_\nabla(x,y) = \nabla I(x,y)$ que associa a cada um dos pixels sua probabilidade de pertencer ao contorno do objeto (Figura 4-11). Considerando $\boldsymbol{\xi} = (x,y)$ e $p(\boldsymbol{\xi}) = \Psi_\nabla(\boldsymbol{\xi})$, pode-se aplicar o algoritmo GNG à imagem *I*, gerando uma rede que adapta sua topologia ao contorno. A Figura 4-12 ilustra a evolução do grafo gerado por GNG em um teste dessa estratégia. A Figura exibe grafos gerados com 25 (A), 64 (B), 100 (C), 144 (D) e 169 (E) nós. Observe que a conectividade dos pontos é importante para identificar a ordem correta dos pontos de

[31] O grafo gerado pelo Growing Neural Gas pode ser utilizado em problemas de classificação de uma forma diferente através de uma rede RBF: uma rede supervisionada com uma camada escondida dispondo de um conjunto de neurônios que funcionam como centros de funções de base radial (*RBFs, radial basis functions*). Nesse caso, o GNG é utilizado para identificar as posições dos neurônios (Fritzke, 1996) e o

referência. Porém, também é necessário que o grafo gerado pelo GNG sofra uma poda de forma a desconsiderar conexões que estabelecem um "atalho" na seqüência de pontos (isto é, nos casos em que um ponto apresenta conectividade maior do que 2)[32]. Veja a Figura 4-13.

Figura 4-11: Modelagem de mãos 2D. Da esquerda para direita: imagem original em tons de cinza; imagem em preto e branco gerada pela aplicação de um limiar; contorno da mão. Adaptada de Rodriguez et al. (2006).

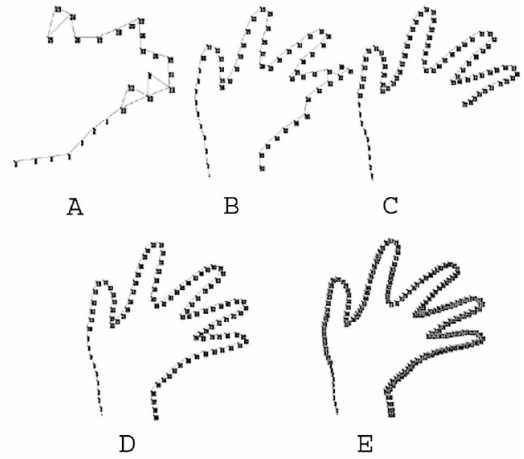

Figura 4-12: Modelagem de mãos 2D. Grafo gerado por GNG com 25 (A), 64 (B), 100 (C), 144 (D) e 169 (E) nós. Adaptada de Rodriguez et al. (2006).

algoritmo completo é conhecido como Growing Neural Gas Supervisionado (Holmström, 2002).

[32] A poda se baseia numa ordenação das unidades do grafo e na comparação das orientações de arestas vizinhas.

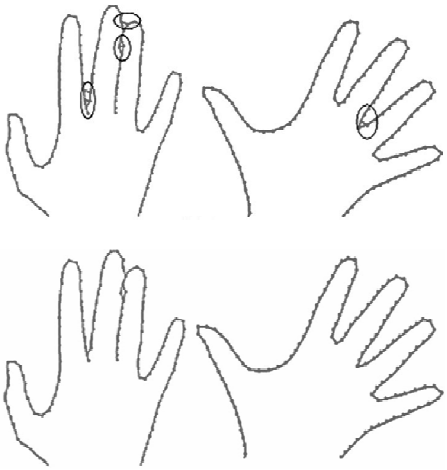

Figura 4-13: Exemplos de “atalhos” nos grafos GNG para modelos de mãos (duas imagens da parte de cima) e suas respectivas correções (duas imagens da parte de baixo). Adaptada de Rodriguez et al. (2006).

Outra forma de classificar as aplicações de Aprendizado Competitivo é em relação ao modo como os sinais de entrada são obtidos. Em muitos problemas, a função de probabilidade $p(\boldsymbol{\xi})$ não é explicitamente conhecida, mas se dispõe de um método direto de obtenção de sinais de entrada. Esse é o caso da aplicação de GNG no trabalho de Figueiredo (2007): os vetores de entrada são gerados varrendo-se seqüencialmente o volume de amplitudes sísmicas e, para cada amostra do volume, é gerado um vetor com as 9 amostras anteriores no traço sísmico, a amostra em questão, e as 9 seguintes no traço. Pode-se dizer que, em situações como essas, o conjunto de treinamento para o algoritmo de Aprendizado Competitivo é previamente conhecido.

Por outro lado, em uma segunda classe de problemas, a função de probabilidade $p(\boldsymbol{\xi})$ é explicitamente conhecida, sendo necessário selecionar sinais de entrada de acordo com $p(\boldsymbol{\xi})$. O caso mais simples corresponde à situação em que se tem uma distribuição uniforme de probabilidades. Esse é o caso, por exemplo, da utilização de algoritmos de aprendizado competitivo para gerar uma malha a partir de uma nuvem de pontos (Yu, 1999; Ivrissimtzis, 2003): todos os pontos da nuvem devem ser considerados igualmente prováveis.

Em outros casos, como o trabalho de Rodriguez et al. (2006), apresentado acima, a função de probabilidade $p(\boldsymbol{\xi})$ é não-uniforme e é calculada explicitamente. Esse também é o caso da extração de regiões de falha por GNG que é apresentada nesta tese. Os sinais de entrada entregues ao GNG devem ser gerados de acordo com uma função de probabilidade obtida a partir do atributo de falha.

Formas ingênuas de selecionar pontos no domínio dos dados podem levar a uma distorção da distribuição de probabilidades ou gerar um tempo de execução indeterminado. Assim, a seleção de pontos deve ser feita de forma a garantir fidelidade com a distribuição de probabilidades e assegurar tempo de execução finito e determinado. O Apêndice B apresenta alguns algoritmos de geração de números aleatórios para probabilidades discretas não-uniformes. Nas implementações desenvolvidas nesta tese foram utilizados o algoritmo de inversão por pesquisa binária e o método de tabelas-guia (ver Apêndice B).

A Figura 4-14 abaixo resume a discussão sobre classificação apresentada nesta seção.

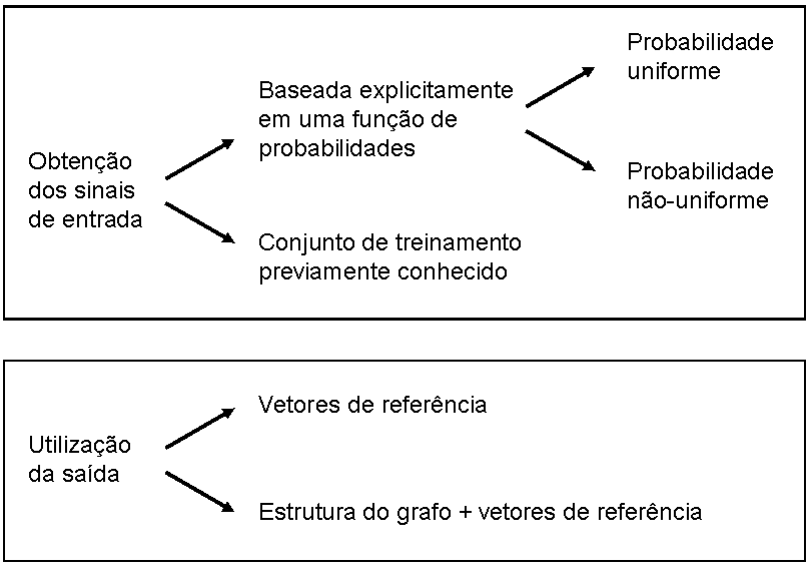

Figura 4-14: Duas maneiras de se classificar aplicações do Growing Neural Gas.

4.6. Geração do grafo por GNG para extração de falhas

Neste trabalho, propõe-se utilizar o Growing Neural Gas como um extrator de estruturas topológicas presentes nos dados de entrada, à semelhança do que foi obtido no exemplo da Figura 4-6 da seção 4.3. No problema de identificação de falhas sísmicas, os dados de entrada correspondem a um atributo de falha que, como visto no Capítulo 2, é um dado escalar regularmente amostrado sobre o espaço tridimensional. Assim, os sinais de entrada do GNG são vetores do $\Re^3$ que correspondem a posições do volume do atributo de falha. Essas posições são obtidas aleatoriamente, de acordo com uma distribuição de probabilidades associada ao atributo de falha.

4.6.1.
Mapeamento do atributo de falha em probabilidades

O mapeamento do atributo em probabilidades pode ser feito de várias maneiras. A mais simples consiste em mapear através de uma função afim o intervalo de valores do atributo no intervalo [0.0, 1.0]. Os diferentes algoritmos de cálculo de atributo de falha podem gerar diferentes conjuntos de valores para um mesmo volume de amplitude sísmica. Assim, se m_F é o valor do atributo correspondente à situação de menor possibilidade de falha sísmica prevista pelo algoritmo utilizado e M_F o de maior possibilidade, então a probabilidade $P(\boldsymbol{\xi})$ é calculada a partir do atributo de falha $A_F(\boldsymbol{\xi})$ como[33]:

$$P(\boldsymbol{\xi}) = \frac{A_F(\boldsymbol{\xi}) - m_F}{M_F - m_F}.$$

A título de exemplo, considere o atributo de falha calculado pelo algoritmo de cubo de coerência (ver seção 2.2.2). Nesse caso o atributo toma valores entre 0 e 1, mas como falhas estão associadas à baixa coerência, tem-se que m_F=1 e M_F=0.

Valores muito baixos de probabilidade de falha normalmente estão associados com regiões onde o dado sísmico se apresenta muito coerente. Devem ser descartados para evitar sobrecarregar o GNG com informações não relacionadas com as estruturas de falhas presentes nos dados. Assim, pode-se definir um limiar de corte *c* abaixo do qual a probabilidade considerada será zero:

$$P(\boldsymbol{\xi}) = L_c\left(\frac{A_{\mathbf{F}}(\boldsymbol{\xi}) - m_{\mathbf{F}}}{M_{\mathbf{F}} - m_{\mathbf{F}}}\right)$$

$$L_c(x) = \begin{cases} x, & x \geq c \\ 0, & x < c \end{cases}$$

O gráfico da Figura 4-15 mostra o mapeamento proposto de um atributo de coerência em um volume de probabilidades. Amostras com valores altos de coerência são ignoradas.

[33] A rigor, essa expressão para $P(\boldsymbol{\xi})$ precisaria ainda ser normalizada (dividindo-se pela soma de todos os $P(\boldsymbol{\xi})$ do volume) e representa um indicador de falha. Apesar disso, como esse indicador possui valores entre 0 e 1, continuará sendo entendida como probabilidade.

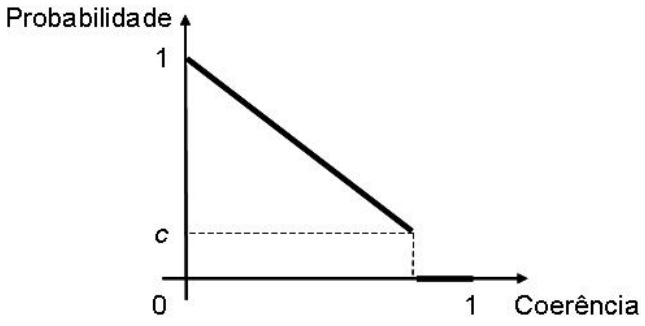

Figura 4-15: Gráfico do mapeamento proposto de atributo de coerência em probabilidades para extração de falhas.

4.6.2. Medida de erro local

Na seção 4.3, foi visto que Fritzke (1995) propôs o quadrado da distância entre o sinal de entrada e a unidade vencedora como medida do erro. Entretanto, como se deseja ter um grafo que descreva a estrutura topológica dos dados, é mais interessante trabalhar com o GNG tendo como objetivo a maximização da entropia ao invés da minimização do erro de quantização (ver Figura 4-2, na seção 4.1). Assim a avaliação do erro local é alterada de forma a funcionar como um contador do número de sinais para os quais a unidade foi vencedora, independente das distâncias entre cada sinal e a unidade. Nesse caso, o passo 5 do algoritmo GNG apenas incrementa de uma unidade a variável de erro do nó vencedor.

Como forma de avaliar o impacto desta modificação, as duas versões do GNG, com minimização do erro e maximização da entropia, foram executadas sobre dados bidimensionais. As imagens da Figura 4-16 exibem uma fatia de tempo de um atributo de falha em tons de cinza, onde a cor preta corresponde a alta probabilidade de falha e a branca a baixa probabilidade. Sobre a fatia é exibido o grafo composto por 150 nós gerado por GNG, do lado esquerdo com atualização do erro local pela distância euclidiana e do lado direito com maximização da entropia. Os valores dos parâmetros são os da Tabela 1 (seção 4.3) com a_{max} = 44 e β = 0.05. Praticamente todo o lado direito da fatia não apresenta estruturas lineares bem definidas tendo apenas valores muito baixos. Observe que, com a minimização do erro, o GNG utiliza muito mais nós para representar essa região dos dados do que no caso de maximização da entropia. Por outro lado, a estrutura vertical do lado esquerdo da fatia é representada com mais nós no segundo caso do que na minimização do erro.

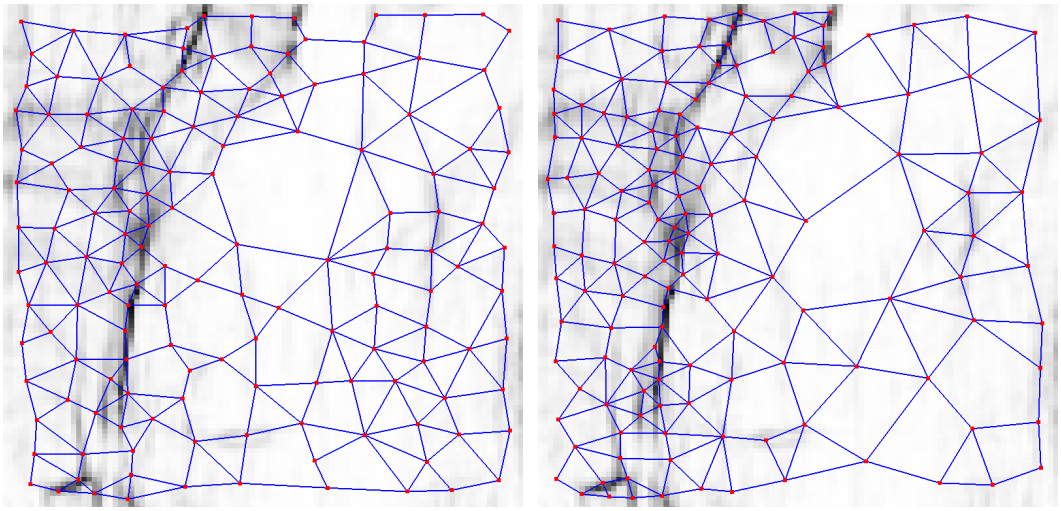

Figura 4-16: Fatia de tempo de atributo de falha em tons de cinza, sobreposto por grafo gerado por GNG. À esquerda, com minimização do erro de quantização e, à direita, com maximização da entropia.

4.6.3. Resultados

Esta seção se encerra com a apresentação dos resultados gerados pelo GNG para um atributo de falha gerado a partir de dados reais tridimensionais. O atributo de falha utilizado é o cubo de coerência por auto-estrutura, calculado a partir de um volume de amplitudes sísmicas (dados reais). Os volumes têm tamanho de 64x106x99 amostras. O tempo de execução foi da ordem de 40s, com autilização de um equipamento com cpu Pentium 4 de 3.4GHz.

O volume de probabilidades foi gerado a partir do atributo de falha utilizando um limiar de corte c = 0.2. Os valores dos parâmetros são os da Tabela 1 (seção 4.3) com a_{max} = 44. O grafo foi gerado com 2000 nós e possui 5503 arestas.

Imagens do atributo de falha e dos resultados são exibidas da Figura 4-17 até a Figura 4-22. O atributo de falha é exibido em fatias ao longo das três direções do levantamento, com a utilização de um mapa de cores que vai do azul, para o valor mais baixo, passando pelo branco até o vermelho, associado ao valor mais alto. Nas Figuras em que o atributo de falha é exibido combinado com o grafo, é utilizado um mapa de cores em tons de cinza, onde o valor mais alto é mapeado em preto, enquanto o mais baixo é mapeado em branco.

As imagens mostram que o grafo acompanha as estruturas presentes no volume de atributo de falha, sendo que a distribuição de nós é mais densa nas

regiões onde a presença da falha é mais nítida. Regiões onde o atributo apresenta valores baixos geram nós mais esparsos com arestas maiores.

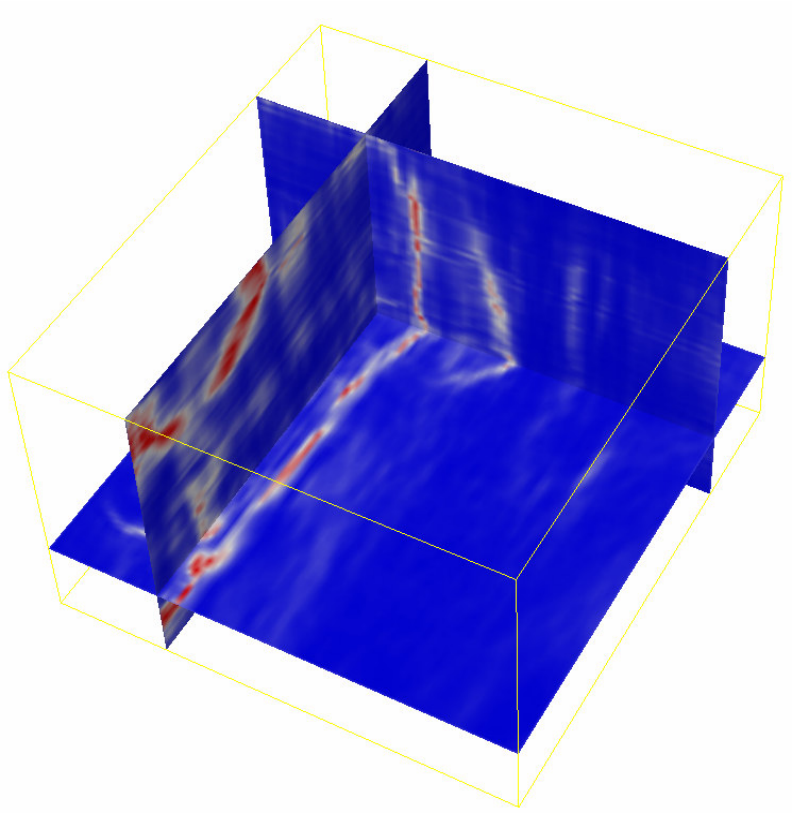

Figura 4-17: Três fatias do cubo de coerência do volume sísmico de teste.

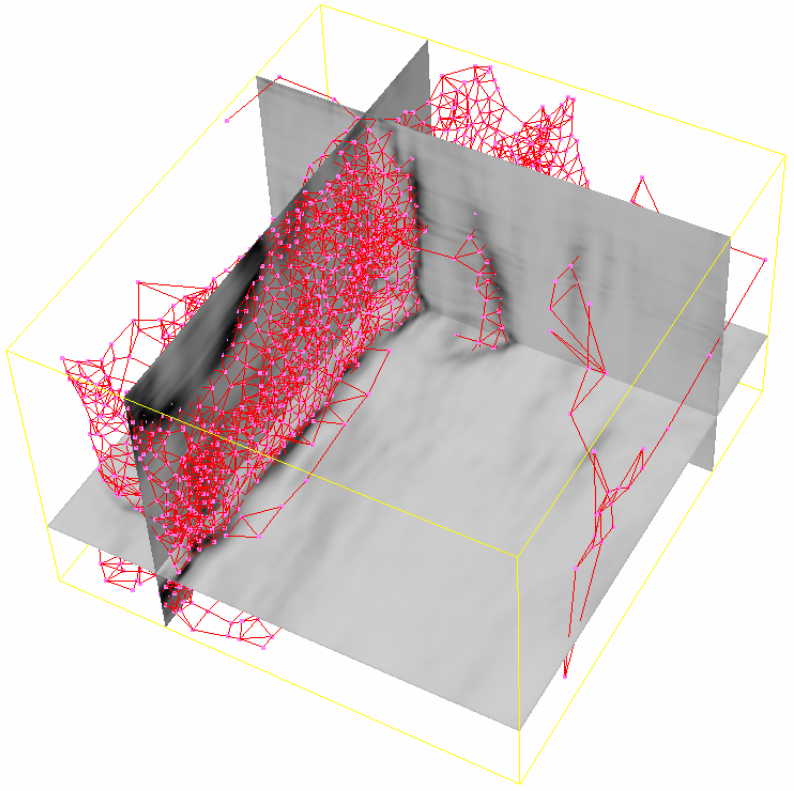

Figura 4-18: Grafo gerado para o cubo de coerência do volume sísmico de teste.

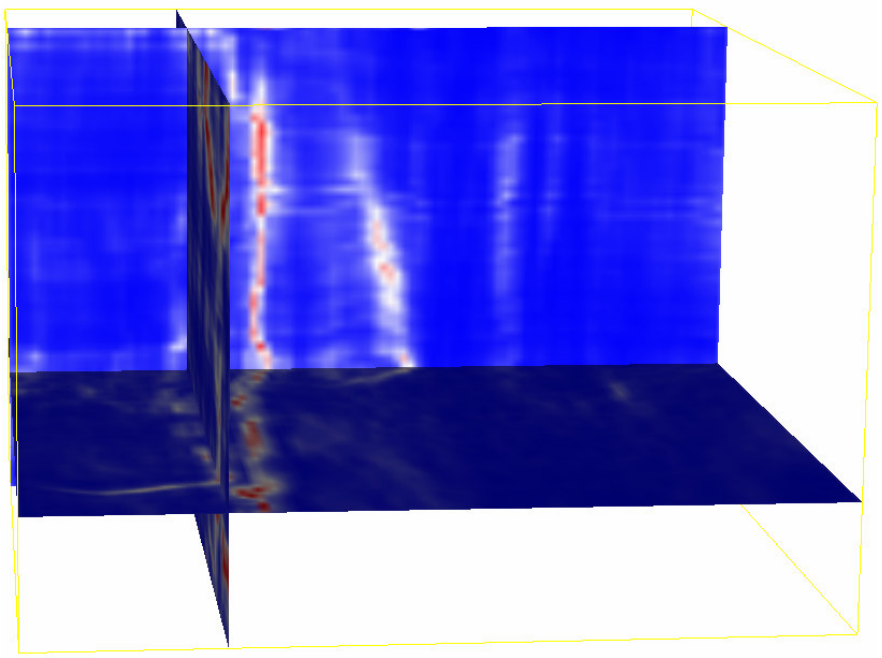

Figura 4-19: Mesmas três fatias da Figura 4-17, sob um segundo ponto de observação.

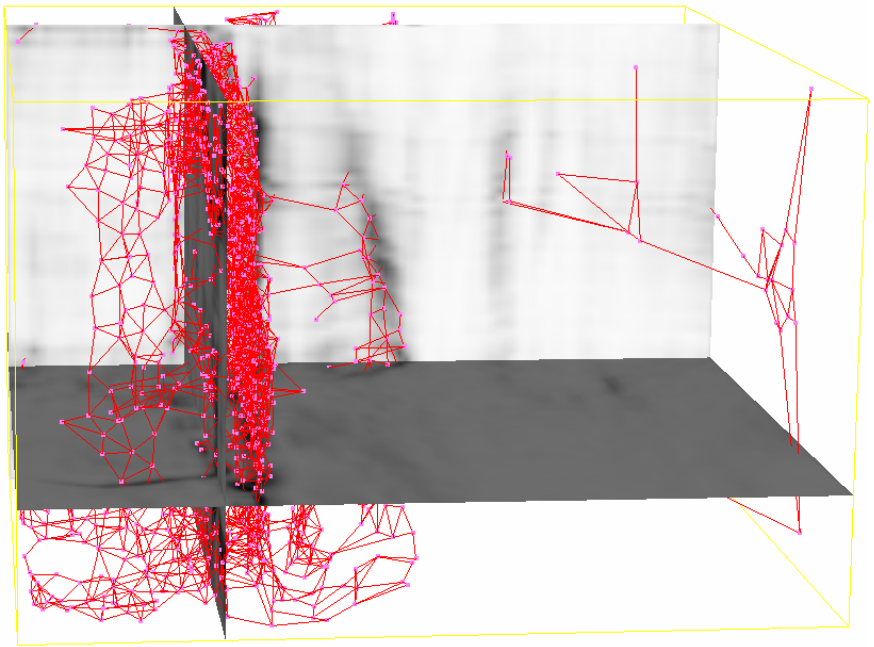

Figura 4-20: Grafo visto sob mesmo ponto de observação da Figura anterior.

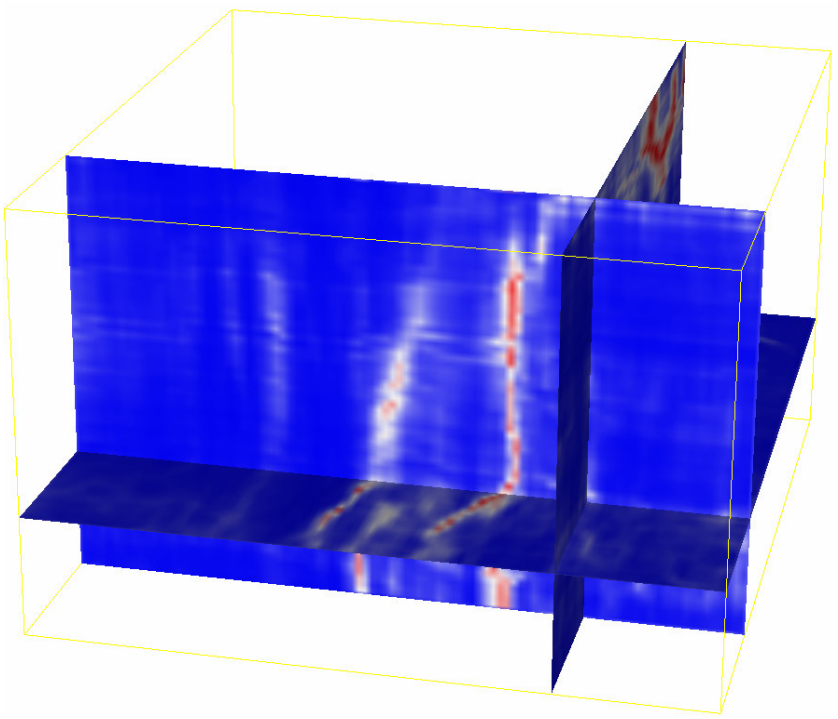

Figura 4-21: Mesmas três fatias da Figura 4-17, sob um terceiro ponto de observação.

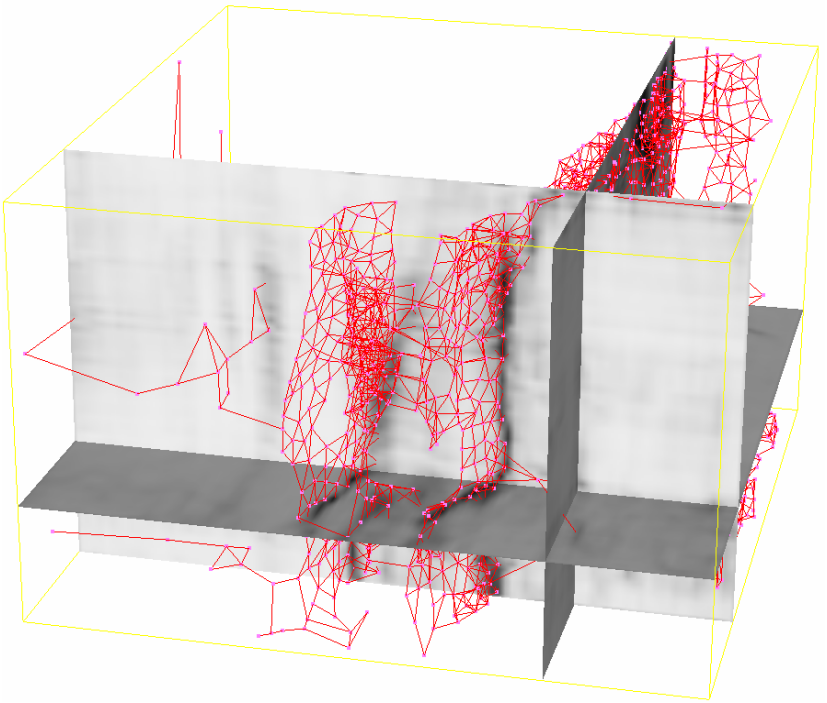

Figura 4-22: Grafo visto sob mesmo ponto de observação da Figura anterior.

4.7. Escala

Qualquer observação por um aparato físico é feita através de algum tipo de abertura cuja largura deve ser maior do que zero (caso contrário nenhum fóton seria captado). O intervalo de escala de qualquer instrumento de observação está limitado pela escala interior (*inner scale*), que é o menor detalhe visto pela menor abertura e a escala exterior (*outer scale*), que é o detalhe mais grosseiro que pode ser descriminado, isto é, a imagem como um todo (campo de visão). No caso de grafos gerados pelo Growing Neural Gas a questão da escala, isto é, de qual o nível de detalhe que o grafo é capaz de reter, também está presente[34].

O grafo cresce com a inserção de novos nós. Em um grafo com poucos nós as estruturas presentes no dado não são capazes de se individualizar. Nós representando regiões de estruturas diferentes aparecem conectados. Conforme o número de nós cresce, conexões desse tipo tendem a ser removidas pela estratégia de eliminação de arestas com idade alta. Mas, como visto na seção anterior, para que isso aconteça é necessário que, dos dois lados da aresta, o grafo seja localmente denso. Assim, a conexão entre estruturas diferentes pode ficar preservada se o grafo como um todo não gerou um número de nós suficiente para se tornar denso dos dois lados. Um motivo para a ocorrência desse fato é que uma das regiões apresenta valores baixos de probabilidade espalhados de forma não organizada, mas se encontra próxima de uma região estruturada com valores de probabilidade maiores.

A Figura 4-23 mostra uma situação em que o grafo gerado por GNG, com aproximadamente 100 nós, foi incapaz de separar as várias estruturas presentes nos dados. Na imagem da esquerda da Figura 4-23 pode-se identificar 5 estruturas principais[35]. Situações desse tipo também podem ser observadas na Figura 4-18, Figura 4-20 e Figura 4-22. Observe que, em geral, as arestas ligando estruturas diferentes são maiores do que as arestas internas a cada estrutura.

[34] A questão da escala aparece também no cálculo do tensor de estrutura do gradiente (ver Apêndice A).

[35] Uma sexta estrutura poderia ser associada ao ponto isolado na parte centro-direita do topo da imagem.

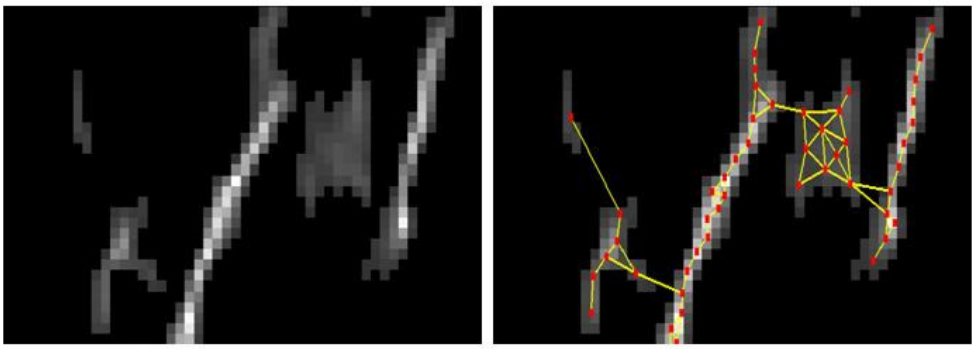

Figura 4-23: À esquerda, função de probabilidade bidimensional. À direita grafo gerado por GNG sem individualização das estruturas.

A Figura 4-24 exibe a continuação da evolução do grafo gerado por GNG até alcançar um número de nós em torno de 300. Observe que, no final, todas as estruturas principais se encontram individualizadas.

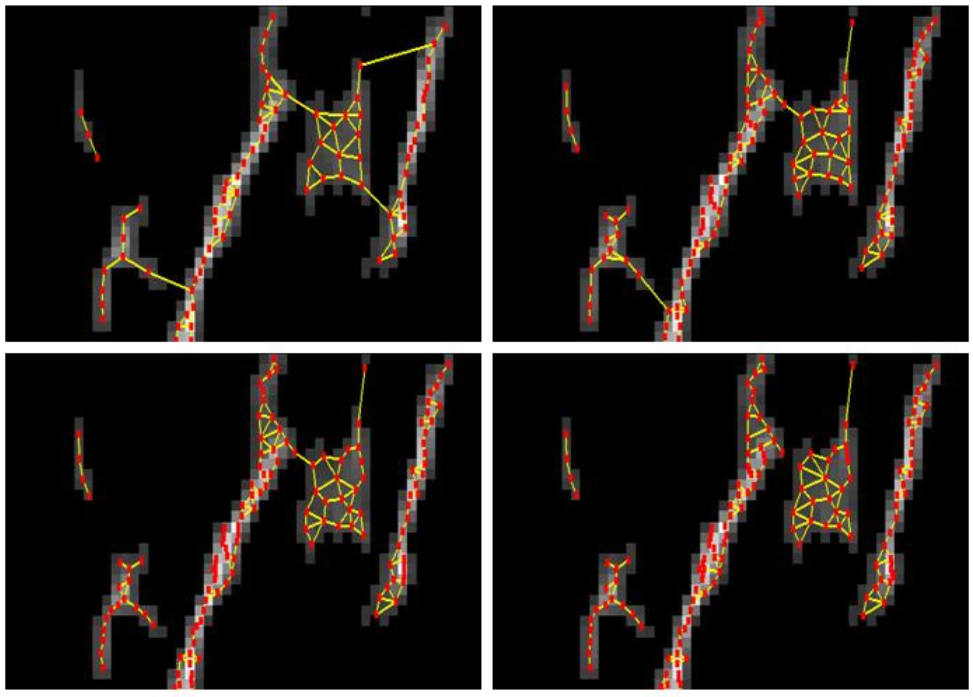

Figura 4-24: Continuação da evolução do grafo GNG da Figura anterior. Conforme o número de nós cresce, mais estruturas são individualizadas.

O desligamento dos subgrafos associados a regiões estruturalmente diferentes depende do acréscimo de nós de forma a impedir que a regra CHL preserve as arestas que conectam as estruturas. Esse fato é exemplificado na Figura 4-25, em que a estrutura do canto superior esquerdo se desliga da de baixo, apenas quando o número de nós localizados na estrutura, inicialmente 1,

cresce para 3. Assim, o conceito de escala do grafo gerado por GNG está ligado ao conceito de conjunto de nós denso apresentado na seção 4.2.

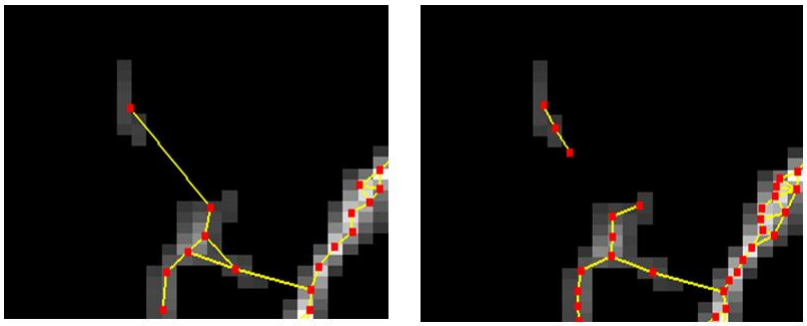

Figura 4-25: Desligamento da estrutura do canto superior esquerdo, como o aumento do número de nós.

Por outro lado, dentro de uma região fortemente estruturada, a inclusão de novos nós irá gerar uma representação mais detalhada da estrutura com a especialização de cada nó, isto é, com a diminuição da região de Voronoi de cada nó. Um limite razoável para esse processo seria impor que a distância entre dois nós não seja menor do que a distância entre amostras do atributo sísmico, que é a escala interior dos dados originais.

Infelizmente, o algoritmo Growing Neural Gas não estabelece nenhum mecanismo de controle sobre o nível de detalhe que o grafo gerado possui em relação às estruturas presentes no dado. A utilização de um número muito grande de nós, além de ser computacionalmente custoso, não garante a segmentação completa do grafo. Assim, é necessário um processamento posterior do grafo, de forma a isolar as estruturas. Essa questão é abordada no Capítulo seguinte.

4.8. Geração de malha geológica por GNG

Cálculos como simulações de fluxo de fluidos em reservatórios são realizados sobre malhas que não são grades regulares. Por exemplo, para reduzir custos computacionais, uma malha de simulação de reservatório é tipicamente amostrada de forma mais grosseira ao longo das dimensões horizontais e sua amostragem pode não ser uniforme (Hale, 2001). Assim, para se adaptar melhor à geologia da subsuperfície, a malha de simulação de reservatório deve ser não-estruturada. Por outro lado, a acurácia da maioria dos cálculos realizados nas malhas depende da regularidade dos elementos da

malha. Simulações realizadas em malhas triangulares fortemente regulares, isto é, com triângulos aproximadamente eqüiláteros, são mais acuradas do que as realizadas sobre malhas irregulares com triângulos alongados (Strang et al., 1973).

Desta forma, a geração de uma **malha geológica** deve compatibilizar dois objetivos que são em alguma medida antagônicos: a malha deve se adaptar às estruturas geológicas, isto é, deve ser mais detalhada ao longo das estruturas e se alinhar com elas, e ao mesmo tempo, deve ser aproximadamente regular.

Nesta seção é apresentada uma proposta de utilização do Growing Neural Gas para a geração de uma malha geológica a partir de um atributo volumétrico que descreva a estrutura geológica em subsuperfície. Sem perda de generalidade, será considerado que o atributo geológico é um atributo de falha; assim, espera-se gerar uma malha geológica que acompanhe as estruturas de falha presentes nos dados. A idéia é, assim como na proposta de extração de falhas, mapear o atributo em um volume de probabilidades para seleção dos sinais de entrada para o GNG. Contudo, para a geração de uma malha geológica não se pode ignorar regiões coerentes, como feito anteriormente. Ao contrário, valores altos de coerência devem ser tratados como um pano de fundo uniforme. Assim, pode-se definir um limiar de corte *c* abaixo do qual a probabilidade considerada será constante:

$$P(\boldsymbol{\xi}) = L_c^g\left(\frac{A_F(\boldsymbol{\xi}) - m_F}{M_F - m_F}\right)$$

$$L_c^g(x) = \begin{cases} x, & x \geq c \\ c, & x < c \end{cases}$$

O gráfico da Figura 4-26 mostra o mapeamento proposto de um atributo de coerência em um volume de probabilidades. Amostras com valores altos de coerência são tratadas como tendo um mesmo valor de probabilidade.

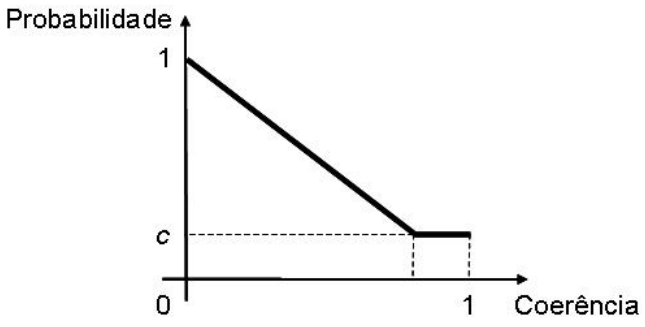

Figura 4-26: Mapeamento de um atributo de coerência em probabilidades para geração de malha geológica.

Além da alteração do mapeamento em probabilidades, o objetivo do GNG também é modificado. De forma a reforçar a geração de triângulos de tamanhos mais semelhantes, o GNG é executado tendo como objetivo a minimização do erro de quantização em vez da maximização da entropia (ver Figura 4-2, na seção 4.1), como no caso de extração de falhas.

Por fim, para gerar uma malha de triângulos é necessário gerar a triangulação de Delaunay para os nós do grafo gerado pelo GNG. A triangulação de Delaunay é utilizada devido a sua propriedade de maximizar o menor ângulo de todas as possíveis triangulações (O'Rourke, 1993)[36].

Essa estratégia para a geração da malha geológica é testada, a seguir, em um dado bidimensional para o qual se dispõe de uma malha gerada pelo processo proposto por Hale (seção 3.4.1). A Figura 4-27 apresenta a imagem de uma fatia de tempo de um atributo de falha extraído de Agüero (2005) e também presente em Pedersen (2002). Agüero (2005) utilizou essa imagem para gerar uma malha aplicando o processo de geração de malha por átomos (Hale, 2002). Com esse processo, foram distribuídos 316 átomos sobre a imagem e em seguida foi aplicada a triangulação de Delaunay, cujo resultado pode ser visto na imagem da esquerda da Figura 4-28.

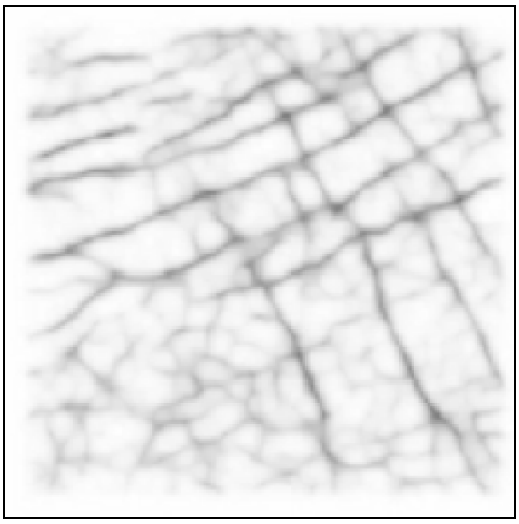

Figura 4-27: Fatia de tempo de um atributo de falha (Agüero, 2005).

[36] Como visto na seção 4.2, devido ao uso da regra CHL, o grafo gerado pelo GNG é uma triangulação de Delaunay induzida dos nós do grafo. Assim, nem todas as arestas da triangulação estão presentes no grafo.

O lado direito da Figura 4-28 apresenta o resultado da triangulação de Delaunay[37] sobre 316 nós do grafo resultante da execução do GNG a partir da mesma imagem da Figura 4-27. No mapeamento em probabilidades foi aplicado um limiar de corte $c = 0.1$. Os valores dos parâmetros do GNG usados são os da Tabela 1 (seção 4.3), com $a_{max} = 88$. Tanto a malha gerada pelo processo de distribuição de átomos de Hale como a gerada pelo GNG apresentam um forte alinhamento com as estruturas presentes na imagem. Entretanto, a malha obtida por GNG se apresenta mais regular do que a obtida pelo método de átomos.

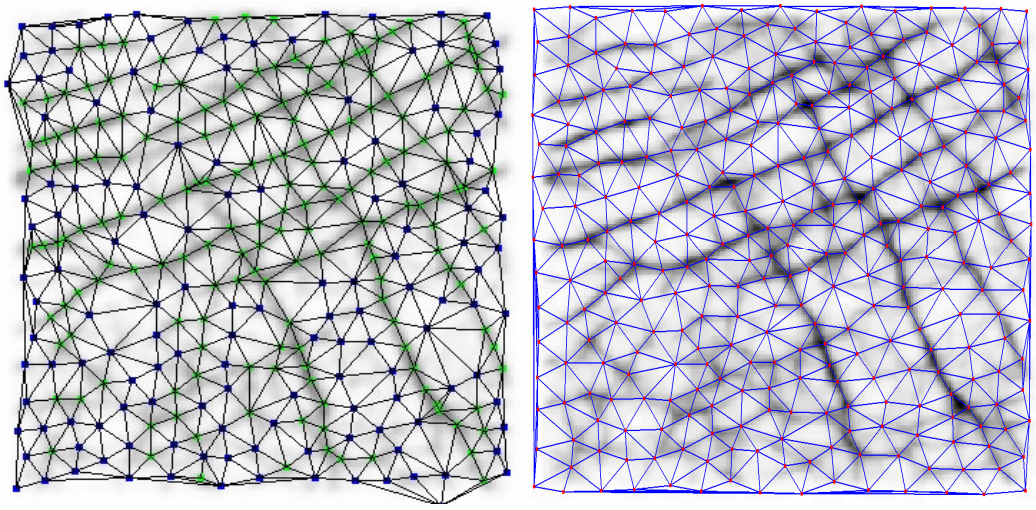

Figura 4-28: À esquerda: triangulação de Delaunay para um conjunto de 316 átomos gerados pela geração de malhas por átomos (Agüero, 2005). À direita: triangulação de Delaunay para um conjunto de 316 nós gerados por GNG.

A Figura 4-29 exibe na imagem à esquerda o grafo gerado por GNG antes da triangulação de Delaunay e à direita após a triangulação. Observe que, como esperado, todas as arestas do grafo são arestas de Delaunay (ver seção 4.2).

[37] Na geração da triangulação de Delaunay para os nós do grafo foi utilizada a biblioteca CGAL (2007).

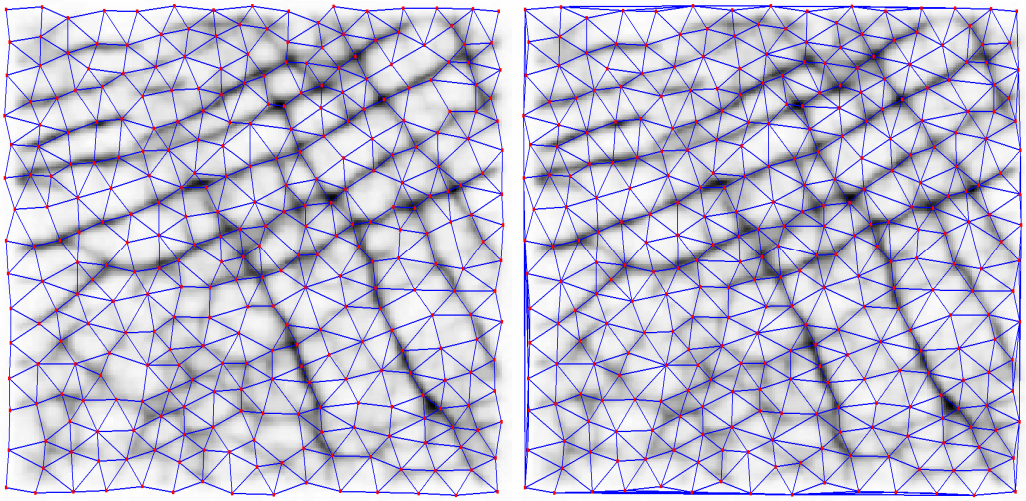

Figura 4-29: À esquerda: grafo gerado por GNG com 316 nós sobre a imagem da Figura 4-27. À direita: triangulação de Delaunay para um conjunto de 316 nós gerados por GNG.