



Universidade Federal de Pernambuco
Centro de Ciências Exatas e da Natureza
Departamento de Informática

Pós-graduação em Ciência da Computação

**Geração de Trajetórias de Estados por
Mapas Auto-organizáveis com Topologia
Dinâmica**

Ruben Carlo Benante

TESE DE DOUTORADO

Recife

Quarta-feira, 05 de Março de 2008

Universidade Federal de Pernambuco
Centro de Ciências Exatas e da Natureza
Departamento de Informática

Ruben Carlo Benante

**Geração de Trajetórias de Estados por Mapas
Auto-organizáveis com Topologia Dinâmica**

*Trabalho apresentado ao Programa de Pós-graduação em
Ciência da Computação do Departamento de Informática
da Universidade Federal de Pernambuco como requisito
parcial para obtenção do grau de Doutor em Ciência da
Computação.*

Orientador: *Prof. Dr. Aluizio Fausto Ribeiro Araújo*

Recife
Quarta-feira, 05 de Março de 2008

Benante, Ruben Carlo

Geração de trajetórias de estados por mapas auto-organizáveis com topologia dinâmica. / Ruben Carlo Benante. - Recife : O Autor, 2008.

xxxii, 182 folhas : il., fig., tab.

Tese (doutorado) - Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2008.

Inclui bibliografia e glossário.

1. Ciência da Computação - Redes neurais artificiais. 2. Otimização de sistemas. 3. Teoria de controle. I. Título.

006.32

CDD(22.ed.)

MEI2008-049

*A meu pai, companheiro em todos os momentos da minha
vida.*

Agradecimentos

O trabalho de doutorado é muitas vezes tido como um esforço individual, e numa certa visão limitada o é. Entretanto está nas relações interpessoais o essencial apoio que torna a solidão do trabalho suportável, e o incentivo e reconhecimento motivo de alegria para continuar. A muitos devo agradecer, e em especial minha gratidão:

ao meu orientador, Prof. Dr. Aluizio Fausto Ribeiro Araújo, que tornou este trabalho possível, de muitas maneiras;

ao Prof. Willem Ferdinand Gerardus Haselager, por seu incentivo e exemplo, e por mostrar-me novas formas de ver o problema em um momento difícil;

ao Prof. Manoel Eusébio de Lima, por sua parcimónia e paciência como presidente da banca de defesa;

ao Prof. Fernando Buarque de Lima Neto, por suas importantes e sempre construtivas críticas;

ao Prof. Guilherme de Alencar Barreto, por sua dedicação nas revisões, e importantes sugestões;

ao Prof. Glauco Augusto de Paula Caurin, por sua importante participação com uma aplicação real do modelo aqui desenvolvido, e sua clareza e pontualidade nas sugestões;

ao Prof. Tiago Alessandro Espínola Ferreira, por sua colaboração imensa em revisões, sugestões e por nossas frutíferas discussões;

à Prof. Maria Eunice Quilici Gonzalez, por seu exemplo como ser-humano, que hoje e sempre guiará meus passos na vida.

aos amigos e à família, sem os quais nada disso teria sentido.

Só tem capacidade para a filosofia aquele para quem todas as pessoas e coisas, o tempo todo, tenham parecido meras ilusões.

—SCHOPENHAUER (O mundo como vontade e idéia)

Resumo

Este trabalho apresenta um novo modelo de Rede Neural Artificial de Topologia Dinâmica chamado *State Trajectory Generator* (STRAGEN) capaz de gerar trajetórias de estados a partir do mapeamento do espaço de estados de um sistema. O modelo permite a utilização de diferentes critérios para a composição de uma trajetória ótima de acordo com o domínio de interesse, ou diferentes características do mesmo domínio. A representação do mapa de estados feita pelo STRAGEN preserva informações heterogêneas que descrevem a configuração do sistema, evitando assim a deformação de suas propriedades. O modelo é apresentado nas abordagens *off-line* e *on-line*. O STRAGEN *off-line* trabalha em três fases: treinamento, poda e geração de trajetórias; enquanto que a abordagem *on-line* trabalha em duas fases: treinamento e geração de trajetórias. Simulações e resultados para os domínios de robótica (com um manipulador bidimensional e tridimensional, uma mão antropomórfica operando no espaço) e a análise de progressões harmônicas musicais são consideradas ao final como exemplos de domínios distintos.

Palavras-chave: Redes Neurais Artificiais, Redes com Topologia Dinâmica, Geração de Trajetórias de Estados, Mapas Auto-organizáveis, Planejamento, Controle.

Abstract

This document presents a new Artificial Neural Network with Dinamic Topology model called *State Trajectory Generator* (STRAGEN), capable of generating state trajectories by mapping the state space of a system. The model permits the utilisation of different criteria for the composition of an optimum trajectory in accordance with the domain of interest, or different features of the same domain. The state map representation undertaken by STRAGEN preserves heterogeneous information which describes the system's configuration, thus avoiding the deformation of its properties. The model is presented in on-line and off-line approaches. The off-line STRAGEN is executed in three phases: training, pruning and trajectory generaion; while the on-line approach is executed in two phases: training and trajectory generation. Simulations and results for the robotic domain (with a two and three-dimensional robot manipulator and anthropomorphic robot hand operating in space) and an analysis of musical harmonical progression are considered at the end as examples of distinct domains.

Keywords: Artificial Neural Networks, Artificial Neural Networks with Dynamic Topology, State Trajectory Generation, Self-organizing Maps, Planning, Control.

Sumário

1	Introdução	1
2	Descrição do Problema	6
2.1	Abordagens para Planejamento	6
2.1.1	Modelos de Programação Procedimental	7
2.1.2	Modelos de Planejamento	8
2.1.3	Modelos de Aprendizado	11
2.2	Descrição do Problema	16
2.2.1	Identificação e Controle	18
2.2.2	Espaço de Estados	19
2.2.3	Trajетórias de Estados	20
2.2.4	Abordagem Proposta	21
2.3	Domínios Exemplos para Geração de Trajetórias de Estados	28
2.3.1	Planejamento e Controle de Trajetórias de Manipuladores Robóticos	29
2.3.1.1	Cinemática Direta e Inversa	31
2.3.1.2	Dinâmica Direta e Inversa	34
2.3.2	Planejamento e Controle de Trajetórias de Mão Antropomórfica	36
2.3.3	Modelagem de Progressões Harmônicas	37
2.4	Conclusões	39
3	Modelos Anteriores Relevantes	41
3.1	Vetor de Integração para o Ponto Final (VITE)	41
3.1.1	O Modelo VITE	45
3.1.2	Balbulciamento Motor	49
3.2	Mapas Auto-organizáveis (SOM)	51
3.2.1	Algoritmo	53
3.2.2	Exemplo	54
3.2.3	Limitações do SOM	55
3.3	Redes de Representação de Topologia (TRN)	57

3.3.1	Gás Neural (NG)	57
3.3.2	Aprendizado Competitivo Hebbiano (CHL)	60
3.3.3	Combinando as Técnicas	61
3.3.4	Avanços e Limitações do TRN	63
3.4	Crescimento de Estruturas de Células (GCS)	64
3.4.1	Algoritmo	65
3.4.2	Exemplo	69
3.4.3	Aplicações Típicas e Variações	69
3.4.4	Limitações do GCS	70
3.5	Gás Neural Crescente (GNG)	71
3.5.1	Algoritmo	72
3.5.2	Exemplo	73
3.5.3	Avanços e Limitações do GNG	74
3.6	Crescer Quando Necessário (GWR)	76
3.6.1	Algoritmo	77
3.6.2	Discussão	79
3.6.3	Limitações do GWR	82
3.7	Conclusões	84
4	Proposta de Solução	85
4.1	Requisitos do Modelo Proposto	85
4.2	Descrição do Modelo	94
4.2.1	Procedimento do Balbuciamiento Motor	96
4.2.2	Fase de Validação	97
4.2.3	Inicialização do Algoritmo STRAGEN	97
4.2.4	Modelo STRAGEN Off-line	98
4.2.4.1	Fase de Treinamento do STRAGEN-OFF	98
4.2.4.2	Fase de Poda do STRAGEN-OFF	100
4.2.4.3	Fase de Geração da Trajetória por Difusão de Energia	102
4.2.5	Diferenças entre o STRAGEN-OFF e o GWR	103
4.2.6	Modelo STRAGEN On-line	108
4.2.6.1	Fase de Treinamento do STRAGEN-ON	109
4.2.6.2	Fase de Geração da Trajetória por Menor Caminho	110
4.2.7	Diferenças entre o STRAGEN-ON e o STRAGEN-OFF	111
4.3	Conclusões	112

5	Resultados	114
5.1	Simulações para Manipulador Bidimensional	114
5.1.1	Configuração do STRAGEN	114
5.1.2	Resultados	120
5.1.2.1	Treinamento da rede STRAGEN (Fase 1)	120
5.1.2.2	Geração da Trajetória (Fase 2)	125
5.1.2.3	Desviando de Obstáculos	132
5.1.3	Comparações entre STRAGEN, GNG e GWR	133
5.2	Simulações para Manipulador PUMA-560	135
5.2.1	Configuração do STRAGEN	136
5.2.2	Resultados do STRAGEN	137
5.2.3	Trajetória Via-pontos	140
5.2.4	Comparações entre STRAGEN, GNG e GWR	140
5.3	Simulações para Mão Antropomórfica Kanguera	144
5.3.1	Configuração do STRAGEN	145
5.3.2	Resultados do STRAGEN	145
5.4	Simulações para Modelagem de Progressões Harmônicas	148
5.4.1	Configuração do STRAGEN	151
5.4.2	Resultados do STRAGEN	152
5.5	Conclusões	156
6	Conclusões, Discussões e Trabalhos Futuros	157
6.1	Conclusões	157
6.2	Discussões e Trabalhos Futuros	162
	Referências Bibliográficas	167

Lista de Figuras

2.1	Sistema de controle de robôs em módulos funcionais.	8
2.2	Sistemas de Malha Aberta e Malha Fechada.	15
2.3	Identificação e Controle	18
2.4	Esquema de um manipulador composto de duas juntas.	30
2.5	Esquema do Robô PUMA-560, com 6 juntas rotativas.	31
2.6	Problema da cinemática inversa.	33
2.7	Superfície de variação do torque.	35
2.8	Mão Antropomórfica Kanguera.	37
3.1	Modelo do circuito cortical sugerido pelo EVITE.	42
3.2	Movimento Sincronizado.	44
3.3	Modelo VITE.	45
3.4	Modelo SOM.	53
3.5	Exemplos de simulações do modelo SOM.	55
3.6	Modelo NG.	58
3.7	Modelo NG mapeia espaço de estados de manipulador.	59
3.8	Modelo CHL.	61
3.9	Modelo TRN.	61
3.10	Trajetoória no modelo TRN.	62
3.11	Inserção no modelo GCS.	67
3.12	Remoção no modelo GCS.	68
3.13	Exemplo de simulação do modelo GCS.	69
3.14	Exemplo de simulação do modelo GNG.	74
3.15	Diferentes dimensões mapeadas pelo modelo GNG.	75
3.16	Exemplos de simulação do modelo GWR.	80
3.17	Medidas de erro do modelo GWR.	81
3.18	Erro GWR para dados dinâmicos.	81
3.19	Comparativo de variação dos modelos GWR e GNGU.	82

4.1	Fluxograma: STRAGEN-OFF	95
4.2	Remoção no modelo STRAGEN-OFF.	101
4.3	Inserção no modelo STRAGEN.	105
4.4	Difusão de energia.	107
4.5	Trajetórias possíveis na difusão de energia.	108
4.6	Fluxograma: STRAGEN-ON	109
4.7	Remoção no modelo STRAGEN-ON.	111
5.1	Nuvem de pontos de treinamento para robô 2D.	117
5.2	Interface gráfica KSTRAGEN.	119
5.3	Mapas do STRAGEN após treinamento.	120
5.4	Simulação: evolução do treinamento.	122
5.5	Fase de treinamento do STRAGEN-ON.	124
5.6	Gráficos do mapa com diferentes abscissas e ordenadas.	126
5.7	Trajetória gerada pelo modelo STRAGEN com algoritmo de difusão.	128
5.8	Trajetória gerada pelo modelo STRAGEN.	129
5.9	Posicionamento dos braços do robô 2D.	130
5.10	Trajetórias do robô 2D.	131
5.11	Desviando de obstáculos.	132
5.12	Trajetória do robô 2D.	135
5.13	Nuvem de pontos de treinamento para PUMA-560.	136
5.14	STRAGEN-OFF: Trajetória para PUMA-560 (posição).	138
5.15	STRAGEN-ON: Trajetória para PUMA-560 (posição).	139
5.16	STRAGEN-ON: Trajetória para PUMA-560 (ângulo).	140
5.17	STRAGEN-ON: Posição dos braços do PUMA-560 (ângulo).	141
5.18	STRAGEN-ON seguindo via-pontos.	142
5.19	GNG, GWR e STRAGEN: erro de validação e erro E'_1 .	143
5.20	GNG, GWR e STRAGEN: número de nodos e conexões.	143
5.21	Articulações da mão robótica.	144
5.22	Trajetória de mão fechada até mão aberta.	146
5.23	Trajetória de mão aberta até mão fechada.	147
5.24	Ciclo de Quintas da música tonal.	149
5.25	Melodia típica (Schubert, Op.9a).	150
5.26	Transições de estado para 36 acordes da base $\mathbf{B}_7$.	152
5.27	Partitura do experimento 1, usando $\mathbf{B}_7$.	153

5.28	Partitura do experimento 2, usando $\mathbf{B}_8$.	154
5.29	Partitura do experimento 3, usando $\mathbf{B}_9$.	155
5.30	Transições de estado para 21 acordes.	156
6.1	Robô bípede MNRIM.	164

Lista de Tabelas

3.1	Condições para inserção de novo nodo no GWR.	83
4.1	Condições para inserção de novo nodo no STRAGEN.	105
5.1	Constantes cinemáticas de um robô de 2 graus de liberdade.	115
5.2	Base de dados para robô bidimensional.	116
5.3	Evolução do número de nodos e conexões das simulações do robô 2D.	123
5.4	Comparação entre STRAGEN-OFF e STRAGEN-ON (nodos/conexões).	125
5.5	Legenda: tons de cinza para Difusão de Energia.	126
5.6	Comparação entre GNG, GWR e STRAGEN.	134
5.7	Tamanho dos Membros.	134
5.8	Distorção das Posições e Torques.	135
5.9	Comparação entre STRAGEN-OFF e STRAGEN-ON (posição).	138
5.10	Comparação entre STRAGEN-OFF e STRAGEN-ON (ângulos).	139
5.11	Erro E'_1 para GNG, GWR e STRAGEN-ON.	142
5.12	Exemplo de vetor de peso da base de acordes.	151

Glossário

Termo	Descrição
ACM	Métodos Adaptativos com Críticos (também usa-se ACD, <i>Adaptive Critics Design</i>) (p. 14) (Werbos, 1991; Venayagamoorthy et al., 2003).
ALLEGRO	Biblioteca gráfica para programação de jogos para linguagem C/C++, de Shawn Hargreaves, gratuita, compatível com as plataformas: DOS, Unix, Linux, FreeBSD, Irix, Solaris, Darwin, Windows, QNX, BeOS e MacOS X. Provê funções gráficas para 2D e 3D, funções matemáticas, sons, temporizadores, controle de dispositivos de entrada (mouse, joysticks, teclados), gerenciamento de arquivos, compressão de dados e GUI.
ANSI C/C++	Padrão de programação proposto pelo <i>American National Standards Institute</i> para portabilidade do código em diversas plataformas.
ART	<i>Adaptive Resonance Theory</i> , teoria da ressonância adaptativa (Grossberg, 1976).
AUV	<i>Automatic Underwater Vehicle</i> , veículo automático subaquático.
AVITE	<i>Adaptive VITE</i> , baseado no VITE, inclui fase de balbuciamiento motor (Gaudiano & Grossberg, 1991, 1992).
BRAHMA	<i>Brazilian Anthropomorphic Hand</i> , mão antropomórfica brasileira (Caurin et al., 2004).
BUCM	Métodos Baseados na Retropropagação da Utilidade (p. 14) (Werbos, 1991).
CBM	Entrada cerebelo-cortical assumida para o estágio IFV no modelo EVITE (Figura 3.1) (Bullock et al., 1998).

Termo	Descrição
CCLA	<i>Cascade-Correlation Learning Architecture</i> , arquitetura de aprendizado por cascata de correlações (Fahlman & Lebiere, 1990).
CFC	<i>Conventional Feedback Controller</i> , controlador por realimentação convencional.
CHL	<i>Competitive Hebbian Learning</i> , aprendizado competitivo Hebbiano, utilizado para criar conexões entre pares de nós que se ativam simultaneamente para dada entrada (Martinetz et al., 1993).
c.s.	Sulco central no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
DA	Critério de vizinhança para robô bidimensional: Distância Euclidiana entre ângulos das juntas: $\mathbf{V}_\eta = [\mathbf{V}_2]$ (Seção 5.1).
DAT	Critério de vizinhança para robô bidimensional: Distância Euclidiana entre ângulos E distância Euclidiana entre torques: $\mathbf{V}_\eta = [\mathbf{V}_2 \mathbf{V}_3]$ (Seção 5.1).
DCP	<i>Deterministic Control Problem</i> , problema de controle determinístico.
DE	Critério de vizinhança para robô bidimensional: Distância Euclidiana entre as posições do efetuador: $\mathbf{V}_\eta = [\mathbf{V}_1]$ (Seção 5.1).
DEA	Critério de vizinhança para robô bidimensional: Distância Euclidiana entre posições do efetuador E distância Euclidiana entre ângulos: $\mathbf{V}_\eta = [\mathbf{V}_1 \mathbf{V}_2]$ (Seção 5.1).
DEAT	Critério de vizinhança para robô bidimensional: Distância Euclidiana entre posições do efetuador E distância Euclidiana entre ângulos E distância Euclidiana entre torques: $\mathbf{V}_\eta = [\mathbf{V}_1 \mathbf{V}_2 \mathbf{V}_3]$ (Seção 5.1).
DET	Critério de vizinhança para robô bidimensional: Distância Euclidiana entre posições do efetuador E distância Euclidiana entre torques: $\mathbf{V}_\eta = [\mathbf{V}_1 \mathbf{V}_3]$ (Seção 5.1).
DICM	Métodos de Controle Inverso Direto (p. 13) (Werbos, 1991).

Termo	Descrição
Dijkstra	Algoritmo de Dijkstra, também referenciado como algoritmo de menor caminho ou algoritmo de menor custo, gera o caminho de menor custo entre dois pontos, ou o menor caminho caso os custos sejam desconsiderados (Dijkstra, 1959).
DIRECT	Modelo baseado no VITE para problemas de cinemática inversa. (Bullock et al., 1998; Fiala, 1994; Izquierdo et al., 1996).
DOF	<i>Degrees of Freedom</i> , graus de liberdade.
DT	Critério de vizinhança para robô bidimensional: Distância Euclidiana entre torques nas juntas: $\mathbf{V}_\eta = [\mathbf{V}_3]$ (Seção 5.1).
DV	<i>Difference Vector</i> , vetor de diferenças no modelo VITE (Figura 3.1) (Bullock et al., 1998).
DVV	<i>Desired Velocity Vector</i> , vetor de velocidade desejada no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
EESC/USP	Escola de Engenharia de São Carlos / Universidade de São Paulo.
ERG	<i>Endogenous Random Generator</i> , módulo gerador de movimentos aleatórios endogênicos.
ERP	<i>Event-Related Potential</i> , também referido como <i>Event-Related Brain Potential</i> , consiste na gravação multi-canal de campos elétricos gerados pelo cérebro humano captados por uma grande quantidade de eletrodos ou sensores distribuídos na superfície da cabeça.
ET	Equifinalidade Temporal (Ulloa & Bullock, 2003; Sanguinetti & Morasso, 1992).
EVITE	<i>extended VITE</i> , VITE estendido, acrescenta ao VITE mais detalhes de novas áreas do córtex (Bullock et al., 1998, 1999).
FAM	<i>Fuzzy ART Map</i> , combina módulos ART nebulosos (Carpenter et al., 1992).
fase-RT	Células fase reativas-ao-tempo no modelo EVITE.
FDP	Função Densidade de Probabilidade.

Termo	Descrição
FNN	<i>False Nearest Neighbors</i> , falsos vizinhos próximos, conceito proposto por Kennel et al. (1992) originalmente desenvolvido para analisar séries temporais caóticas. Zhao & Small (2006) utiliza o FNN para modelar atratores caóticos em redes MLP.
FOSART	<i>Fully Self-Organizing Simplified Adaptive Resonance Theory</i> , teoria da ressonância adaptativa simplificada completamente auto-organizável (Baraldi & Alpaydin, 2002).
<i>Fuzzy</i>	Redes <i>fuzzy</i> ou redes nebulosas.
<i>Gamma-ON</i>	Ou γ -ON, <i>gamma-observable Neighbours</i> , vizinhança gama-observável (Aupetit et al., 2002).
GC	<i>Growing Chain</i> , cadeia crescente.
GCS	<i>Growing Cells Structures</i> , crescimento de estruturas de células (Fritzke, 1994a).
GG	<i>Growing Grid</i> , grade crescente.
GH-SOM	<i>Growing Hierarchical Self-organizing Maps</i> , mapas auto-organizáveis hierárquicos crescentes.
GNG	<i>Growing Neural Gas</i> , gás neural crescente, derivado do GCS com estrutura mais flexível (Fritzke, 1997a).
GNGU	<i>Growing Neural Gas with Utility</i> , gás neural crescente com utilidade (Fritzke, 1997b).
GO	Sinal escalável voluntário no modelo VITE (Figura 3.1) (Bullock et al., 1998).
<i>Grasp</i>	Pegar (com os dedos). <i>Regrasp</i> , literalmente “re-pegar”, re-posicionar os dedos em ajuste de uma pegada ou movimento do objeto na mão.
GSOM	<i>Growing SOM</i> , mapas auto-organizáveis crescentes.
GWR	<i>Grow when Required Network</i> , crescer quando necessário (Marsland et al., 2002).
IA	Inteligência Artificial.
Ia	Fibra aferente tipo Ia no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
II	Fibra aferente tipo II (realimenta erro de posição) no modelo EVITE (Figura 3.1) (Bullock et al., 1998).

Termo	Descrição
IFV	<i>Inertial Force Vector</i> , vetor de força inercial no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
IGG	<i>Incremental Growing Grid</i> , grade incremental crescente (Fritzke, 1995).
INFANT	modelo que aprende o relacionamento de informações visuais com comandos motores (Kuperstein, 1991).
i.p.s.	Sulco intraparietal no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
ITM	<i>Instantaneous Topological Map</i> , mapa topológico instantâneo (Jockusch & Ritter, 1999).
KANGUERA	Mão robótica antropomórfica biologicamente inspirada, com 22 DOF, sucessora do modelo BRAHMA (Figura 2.8) (Caurin et al., 2005).
KSTRAGEN	Interface gráfica para o programa STRAGEN (Figura 5.2).
Kohonen	Mapas de Kohonen, veja SOM.
LT	<i>Logic Theorist</i> , teórico lógico, programa de raciocínio lógico de Newell e Simon, 1956.
LWINGG	<i>Locally Weighted Interpolating Growing Neural Gas</i> , gás neural crescente com interpolação local de pesos (Flentge, 2006).
MATLAB	Ambiente para cálculos matemáticos da empresa The Mathworks, Inc., 24 Prime Park Way, Natick, MA 01760.
MB	<i>Motor Babbling</i> , balbuciamiento motor, movimentos aleatórios endogenicamente gerados normalmente por crianças em fase de desenvolvimento (Sanguineti & Morasso, 1992). Por hipótese são utilizados para que tais crianças aprendam o controle do próprio braço gradualmente, em um processo chamado de <i>reação circular</i> (Piaget, 1963) que permite o aprendizado de transformações no espaço de trabalho.
MDP	<i>Markov Decision Process</i> , processo de decisão de Markov.
MLP	<i>Multi-Layer Perceptron</i> , perceptron de multicamadas.
MNRIM	Robô bípede de 6 juntas (Foresti, 2006).
NACM	Métodos de Controle Neural Adaptativo (p. 13) (Werbos, 1991).

Termo	Descrição
NARMA	<i>Nonlinear Autoregressive Moving Average</i> , é um modelo matemático para representação de sistemas dinâmicos (Narendra & Mukhopadhyay, 1997).
NG	<i>Neural Gas</i> , gás neural, método de quantização vetorial que distribui nodos no espaço dos dados de entrada (Martinetz et al., 1993).
NICS/UNICAMP	Núcleo Interdisciplinar de Comunicação Sonora / Universidade Estadual de Campinas.
OFPV	<i>Outflow Force and Position Vector</i> , fluxo de força e vetor de posição somados no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
openSUSE GNU/LINUX	Linux é um Sistema Operacional semelhante ao UNIX, originalmente criado por Linus Torvalds com o ajuda da comunidade de desenvolvedores pelo mundo, gratuito, de código aberto, sob a licença pública GNU. openSUSE é uma distribuição LINUX, originalmente uma tradução para o Alemão da distribuição SLACKWARE, produzida atualmente pela comunidade mundial com apoio da empresa NOVELL. S.u.S.E. é um acrônimo para <i>Software und System Entwicklung</i> , em Alemão significando “Desenvolvimento de Programas e Sistemas”, nome da empresa fundadora da marca.
OPV	<i>Outflow Position Vector</i> , vetor de fluxo de saída de posição no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
PCM	<i>Perceptual Control Manifold</i> , espaço de controle perceptual, é o produto do espaço de trabalho pelo espaço sensorial. (Zeller et al., 1997).
PID	<i>Proportion Integral Derivative</i> , controlador derivativo integral proporcional, é um controlador convencional guiado apenas pelo erro. Assume que os parâmetros do sistema controlado são fixos (Barreto et al., 2003b).
PIONEER	Classe de robôs móveis de uso geral.
<i>Pitch-class</i>	Gradientes de frequências nos quais os acordes musicais podem ser decompostos para análise de propriedades acústicas.

Termo	Descrição
PGCS	<i>Probabilistic Growing Cells Structures</i> , estruturas de crescimento de células probabilísticas (Vlassis et al., 1997).
POMDP	<i>Partially Observable Markov Decision Process</i> , processo de decisão de Markov parcialmente observável.
PPC	<i>Present Position Command</i> , comando de posição presente. Representa a posição atual do braço no modelo VITE (Figura 3.3) (Gaudiano & Grossberg, 1991).
PPV	<i>Perceived Position Vector</i> , vetor de posição percebida no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
PTEROA	Veículo aquático utilizado para testes de uma rede de controle supervisionado por Fuji & Ura (1991).
PUMA-560	Robô manipulador industrial de 6 juntas rotativas (Figura 2.5).
QT3	É uma biblioteca multi-plataforma para desenvolvimento rápido de aplicações com GUI, com suporte a C++ e Java, produzido pela Trolltech. A versão de código aberto está disponível gratuitamente sob licença GNU/GPL 2.0.
RBF	<i>Radial Basis Function</i> , funções de base radial (Broomhead & Lowe, 1988). Também utiliza-se na literatura RBFN para <i>Radial Basis Function Networks</i> .
RNA(s)	Rede(s) Neural(is) Artificial(is).
RNA-AO	Redes Neurais Artificiais Auto-Organizáveis.
RNE	<i>Recursive Newton-Euler</i> , função Newton-Euler recursiva, para cálculo de torque, definida no <i>Robotics Toolbox</i> de Corke (1996).
RTT	<i>Rapidly Exploring Random Trees</i> , exploração rápida de árvores aleatórias (Jarvis, 2006).
SCM	Métodos de Controle Supervisionados (p. 12) (Werbos, 1991).
SFV	<i>Static Force Vector</i> , vetor de força estática no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
SGCS	<i>Supervised Growing Cell Structures</i> , crescimento supervisionado de estruturas de células, combina GCS com RBF (Fritzke, 1994b).

Termo	Descrição
SHC	<i>Structural Hill Climbing</i> , subida de encosta estrutural (Angelini et al., 1994).
SOBoS	<i>Self-Organizing Body Schema</i> , esquema corporal auto-organizável (Sanguineti & Morasso, 1992).
SOM	<i>Self-Organizing Map</i> , mapas auto-organizáveis (Kohonen, 1987).
SOM-DT	<i>Self-Organizing Maps with Dynamic Topology</i> , mapas auto-organizáveis com topologia dinâmica.
<i>Spindles</i>	Mecanismo de propriocepção interno aos músculos.
STRAGEN	<i>State Trajectory Generator</i> , gerador de trajetórias de estados, modelo proposto neste trabalho. STRAGEN-OFF: abordagem que utiliza uma fase de poda <i>off-line</i> , e mantém a rede criada totalmente conectada. STRAGEN-ON: abordagem que realiza a poda durante a fase de treinamento (<i>on-line</i>) e permite a criação de áreas desconexas.
TAO	Teoria de Auto-Organização.
TPC	<i>Target Position Command</i> , comando de posição alvo. Representa a posição final do braço desejada no modelo VITE (Figura 3.3) (Gaudio & Grossberg, 1991).
TPV	<i>Target Position Vector</i> , vetor de posição do alvo no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
TreeCGS	<i>Tree Growing Cells Structures</i> , crescimento de estruturas de células em árvores, variante hierárquica do GCS utilizada para gerenciamento de conteúdo e descobrimento de conhecimento por Freeman & Yin (2005).
TRN	<i>Topology Representing Networks</i> , redes de representação de topologia (Martinetz et al., 1991).
TV	Televisão.
UB-Hand 3	Mão robótica antropomórfica que utiliza elásticos nas juntas (Lotti et al., 2005).
VITE	<i>Vector Integration to Endpoint</i> vetor de integração para o ponto final (Bullock & Grossberg, 1988b,a).

Lista de Símbolos

Símbolo	Descrição
$ \cdot $	Cardinalidade do conjunto.
$\ \cdot\ $	Norma vetorial Euclidiana.
$[i]^+$	Função “pisso” $[i]^+ = \max(i, 0)$.
α	Parâmetro ou taxa auxiliar para equações diversas. No GWR usa-se $\alpha_b = \alpha_n = 1,05$ como parâmetros para a função de frequência de não-disparos.
α_f	Taxa final de aprendizado do STRAGEN. Quando não citado usa-se $\alpha_f = 0, 1$.
α^m	Motoneurônio alfa no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
$a_{\max}$	Idade máxima permitida a uma conexão antes de ser eliminada.
$\bar{\mathbf{a}}$	Vetor de limites máximos de atividade que se ultrapassado pelo nodo vencedor, para todos os subgrupos l , impede o STRAGEN de criar um novo nodo (Equação 4.3 e Tabela 4.1).
$\mathbf{a}$	Vetor de atividades do nodo vencedor. a_k é a atividade do subgrupo $1 \leq k \leq l$ do nodo vencedor.
$a(i)$	Atividade de um nodo i no GWR (Eq. 3.27).
a_T	Limite máximo da atividade do nodo vencedor em relação a um padrão de entrada para não se inserir um novo nodo.
$\mathcal{A}$	Conjunto de nodos da rede (ou mapa de saída no $\mathbb{R}^{D_{\mathcal{A}}}$), sendo $ \mathcal{A} $ o número total de nodos do conjunto (cardinalidade).
$\mathbf{A}_i$	Matriz Jacobiana.
A	Conjunto finito de ações que mapeia cada estado.
$A(q)$	Ação aplicada ao estado q .

Símbolo	Descrição
β	Taxa de decaimento do erro local no GNG.
b_0	Estado inicial que acredita-se o sistema se encontrar (crença).
b_f	Estado final que deseja-se atingir, no conjunto de estados de crença.
$\mathbf{B}^{(0)}$	Base de dados não normalizada, de dimensões $L \times D$, contendo padrões de entrada para treinamento.
$\mathbf{B}$	Base de dados normalizada, de dimensões $L \times D$, contendo padrões de entrada para treinamento.
$\mathbf{B}_1$	Base de dados com $L_1 = 600$ padrões de entrada igualmente distribuídos no espaço de trabalho de um robô bidimensional de duas juntas (Tabela 5.2).
$\mathbf{B}_2$	Base de dados de teste com $L_2 = 4096$ padrões de entrada igualmente distribuídos no espaço de trabalho do robô PUMA-560.
$\mathbf{B}_3$	Base de dados completa com $L_3 = 15625$ padrões de entrada igualmente distribuídos no espaço de trabalho do robô PUMA-560 (Figura 5.13 (a)).
$\mathbf{B}_4$	Base de dados de precisão com $L_4 = 6859$ padrões de entrada distribuídos em uma faixa específica da parte frontal do robô PUMA-560 (Figura 5.13 (b)).
$\mathbf{B}_5$	Base de dados com $L_5 = 1000$ padrões de entrada distribuídos no espaço de trabalho do robô Kanguera, apenas com posições de dedos fechando individualmente (Simulação da Figura 5.22).
$\mathbf{B}_6$	Base de dados com $L_6 = 2000$ padrões de entrada igualmente distribuídos no espaço de trabalho do robô Kanguera (Simulação da Figura 5.23).
$\mathbf{B}_7$	Base de dados com $L_7 = 36$ padrões de entrada (acordes) distribuídos em 12 acordes maiores na posição fundamental, 12 acordes maiores em segunda inversão e mais 12 acordes em primeira inversão (todos os 36 acordes podem ser visualizados na Figura 5.26).

Símbolo	Descrição
$\mathbf{B}_8$	Base de dados com $L_8 = 108$ padrões de entrada (acordes) contendo além dos acordes da base $\mathbf{B}_7$ também os 12 acordes menores e os 12 acordes diminuídos em cada inversão (Simulação da Figura 5.28).
$\mathbf{B}_9$	Base de dados com $L_9 = 21$ padrões de entrada, sendo estes acordes da tonalidade do C maior na posição fundamental e nas inversões (Simulação da Figura 5.29).
C	Valor no qual as células de GO saturam (Eq. 3.3) (Bullock et al., 1998).
$\mathcal{C}$	Conjunto de conexões da rede, sendo $ \mathcal{C} $ o número total de conexões do conjunto (cardinalidade).
$C_{i,j}$	Função de conexão. 1 se os nodos i e j estiverem conectados, e 0 caso contrário.
c_{n_1,n_2}	Conexão c entre os nodos n_1 e n_2 .
$c(a, q)$	Função de custo de aplicar a ação $a \in A$ no estado $q \in \mathcal{M}$.
χ	Limite de corte para conexões no STRAGEN. $\chi = \bar{m} + \bar{\omega} \cdot \bar{d}$. Se $ \mathcal{N}(s_1) > 2$, remova todas as conexões c_{s_1,n_i} do conjunto $\mathcal{C}$ para as quais se tem que $\text{Dist}(s_1, n) > \chi$, $n \in \mathcal{N}(s_1)$.
$D_{\mathcal{A}}$	Dimensão do mapa de saída ou conjunto de nodos $\mathcal{A}$.
D ou $D_{\mathcal{M}}$	Dimensão dos padrões de entrada ξ e vetor pesos $\mathbf{w} \in \mathcal{M}$.
D_j	Dimensão de um grupo homogêneo j que compõe o padrão de entrada, com $\sum_{j=1}^m D_j = D$.
$D_{j,i}$	Dimensão de um subgrupo i que compõe o grupo homogêneo j do padrão de entrada, com $\sum_{i=1}^k D_{j,i} = D_j$.
$\mathcal{D}$	Conjunto de distâncias entre os nodos s_1 , s_2 e s_3 , de acordo com o critério η .
$\text{Dist}(i, j)$	Distância Euclidiana entre i e j .
$\text{Dist}_1(s_i, s_j)$	Segmento de menor comprimento de um triângulo formado pelos nodos s_1 , s_2 e s_3 , com s_i e s_j representando os dois nodos pertencentes ao segmento.
$\text{Dist}_2(s_i, s_k)$	Segundo menor segmento de um triângulo formado pelos nodos s_1 , s_2 e s_3 , com s_i e s_k representando os dois nodos pertencentes ao segmento.

Símbolo	Descrição
$\bar{d}$	Desvio padrão.
ε	Taxa de aprendizagem. ε_b é a taxa de aprendizado do nodo vencedor. Quando não citado usa-se $\varepsilon_b = 0,2$. ε_n é a taxa de aprendizado para os nodos vizinhos do nodo vencedor. $\varepsilon_{\mathcal{N}(i)}$ é a taxa de aprendizagem aplicada aos vizinhos do nodo i .
$\varepsilon(t)$	Função de decaimento exponencial no tempo $\varepsilon(t) = \varepsilon_i(\varepsilon_f/\varepsilon_i)^{t/t_{\max}}$, iniciando em ε_i e terminando em ε_f .
E	Energia de um nodo dada pela função de difusão de energia (Eq. 4.14). p.122.
E_1	Erro para avaliar o tamanho médio das conexões, proposta por Marsland et al. (2002) (Eq. 3.37).
E'_1	Erro para avaliar o tamanho médio das conexões criadas (Eq. 5.3).
E_2	Medida de custo que avalia a capacidade da rede de minimizar as distâncias entre os padrões de entrada e os nodos que os respondem (Eq. 3.39).
E_v	Erro de validação da rede (Eq. 4.5).
E_n	Erro local do nodo n .
e_1	Erro do identificador, dado pela diferença entre a saída da planta $\mathbf{y}$ e a saída aprendida pelo modelo $\hat{\mathbf{y}}$ (Figura 2.3).
e_2	Erro do controlador, dado pela diferença entre a saída real da planta $\mathbf{y}$ e a saída desejada $\mathbf{y}^*$ (Figura 2.3).
η	Critério de vizinhança do STRAGEN, define o grupo de informações homogêneas $\mathbf{V}_\eta$ tal que $1 \leq \eta \leq m$, utilizado para avaliar a proximidade entre nodos.
$\mathcal{E}$	Conjunto de conexões $\mathcal{E} \subset \mathcal{C}$ inativas, i.e., que nunca dispararam ligando um nodo vencedor s_1 e um vice s_2 , na fase de poda do STRAGEN (Subseção 4.2.4.2).
f	Função que descreve o comportamento de um sistema dinâmico $f : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$ (Eq. 2.1). Normamente assume-se que $\mathbf{h} \mathbf{f}(0,0) = 0$, i.e., o estado de equilíbrio localiza-se na origem.
$f_{\text{tr}}(q, a)$	Função de transição de estados $q_f = f_{\text{tr}}(q_i, a)$.

Símbolo	Descrição
$f_{de}(t, n)$	Função de difusão de energia (Eq. 4.14) (Zeller et al., 1997).
F_i	Campo de Voronoi do nodo i .
$F[\cdot]$	Função que descreve o comportamento da saída $\mathbf{y}(k+1)$ de um sistema dinâmico (Eq. 2.2).
$\bar{F}[\cdot]$	O problema de <i>neuroidentificação</i> consiste em encontrar uma RNA $\bar{F}[\cdot]$ que aproxime $F[\cdot]$.
$G[\cdot]$	Função que descreve o comportamento da entrada $\mathbf{u}(k+1)$ de um sistema dinâmico (Eq. 2.3).
$\bar{G}[\cdot]$	O problema de <i>neurocontrole</i> consiste em encontrar uma RNA $\bar{G}[\cdot]$ que aproxime $G[\cdot]$.
$G(t)$	No modelo VITE: sinal volitivo GO (Eq. 3.2). g é o sinal GO que multiplica DV dado pela Eq. (3.3). $G_p(t)$ é o sinal involuntário que modela o movimento passivo (Eq. 3.4).
$g^{(0)}$	Entrada provinda do centro de decisão pré-frontal. Ainda, $g^{(1)}$ é a entrada calculada em cascata a partir de $g^{(0)}$ e $g^{(2)}$ análogamente a partir de $g^{(1)}$.
γ^D	Motoneurônio dinâmico gama no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
γ^S	Motoneurônio estático gama no modelo EVITE (Figura 3.1) (Bullock et al., 1998).
$h_y(k)$	$h_y(k) = \exp(-k/y)$ é a função de proximidade do <i>ranking</i> no NG.
h_i	Frequência (ou contador) de disparos do nodo i .
$\tilde{h}_i$	Frequência relativa de disparos do nodo i .
$\bar{h}_i$	Frequência de não-disparos (ou descontador) do nodo i .
h_T	Limiar máximo da frequência de não-disparos $\bar{h}_i$.
$h^{(0)}$	Força inicial, normalmente $h^{(0)} = 1$, utilizada na função $\bar{h}_i(t)$ do GWR.
I	Número de iterações para a fase de validação. Quando não citado, usa-se $I = t_{\max}$.
$I(i)$	Função que retorna um valor aleatório no intervalo $[-i, i]$. Nos testes foi utilizado $i = 1$.
K	Constante $K < 1$ tal que $K = \mathcal{N}(n) /(\mathcal{N}(n) + 1)$, usada na f_{de} .

Símbolo	Descrição
k	Número de conexões emanando do nodo vencedor s_1 tal que $k = \mathcal{N}(s_1) $.
K_{in}	Total de energia cinética do braço do robô (Eq. 2.14).
k -simplex	Dimensão da estrutura da rede GCS.
κ	Limiar de corte característico para se remover um nodo no GCS.
L	Função lagrangiana tal que $L = K_{in} - Pot$ (Eq. 2.14).
L	Número de padrões de entrada contidos na base $\mathbf{B}$. As bases de treinamento utilizadas continham o número de padrões dados respectivamente por: base $\mathbf{B}_1$ para robô bidimensional com $L_1 = 600$; bases $\mathbf{B}_2$, $\mathbf{B}_3$ e $\mathbf{B}_4$ para PUMA-560 com $L_2 = 4096$, $L_3 = 15625$ e $L_4 = 6859$; bases $\mathbf{B}_5$ e $\mathbf{B}_6$ para mão antropomórfica Kanguera com $L_5 = 1000$ e $L_6 = 2000$; e bases $\mathbf{B}_7$, $\mathbf{B}_8$ e $\mathbf{B}_9$ de acordes para progressões harmônicas com $L_7 = 36$, $L_8 = 108$ e $L_9 = 21$.
λ	Número de iterações executadas para que o GCS e o GNG crie um novo nodo.
l	Número de subgrupos em um grupo homogêneo m do vetor de pesos.
m	Número de grupos homogêneos distintos em um vetor de pesos $\mathbf{w}_i$.
$\mathcal{M}$	Mapa de entrada no $\mathbb{R}^{D \cdot \mathcal{M}}$, Espaço de Estados ou Espaço de Trabalho a ser aprendido por uma RNA.
$\mathcal{M}_{free}$	Espaço de trabalho livre de obstáculos (Subseção 2.2.2) (Glasius et al., 1995).
$\mathcal{M}_f$	Conjunto de estados alvo válidos.
μ	Constante da dispersão de energia na função f_{de} , tal que $\mu = K/ \mathcal{N}(n) $.
$\bar{m}$	Média do tamanho das conexões entre nodos.
$\mathcal{N}(n)$	Conjunto de todos nodos que são vizinhos diretos de n e $ \mathcal{N}(n) $ é sua cardinalidade.
$\mathcal{O}$	Conjunto de <i>Obstáculos</i> .
O	Conjunto de observações (realimentação de sensores, por exemplo).

Símbolo	Descrição
$\overline{OX_0Y_0Z_0}$	Sistema de coordenadas com origem em O .
P	Porcentagem usada para o cálculo do limiar de atividade $\bar{a}_k$, para cada subgrupo k , $1 \leq k \leq 1$ do grupo homogêneo ζ (Eq. 4.3). Se não citado, usa-se $P = 1\%$.
$P_a(q' q)$	Probabilidade da ação a levar o sistema para q' dado que este se encontra em q .
P_i	Vetor de posições atuais PPC do VITE (Eq. 3.1).
Pot	Total de energia potencial do braço do robô (Eq. 2.14).
$\frac{d}{dt}P_i$	Variação no tempo do vetor de posições PPC do VITE (Eq. 3.2).
π	Função $a = \pi(q)$ que mapeia estados em ações, chamada de <i>política</i> , que determina as probabilidades das possíveis trajetórias solução para o MDP, e π^* é a política de menor custo esperado.
ϖ	Constante ótima determinada empiricamente $\varpi = 1,5$ para cálculo do limiar χ de remoção de conexões do STRAGEN.
$p(\xi)$	Procedimento MB que retorna um padrão ξ de entrada de uma lista de candidatos Q próximos ao padrão da sua iteração anterior, segundo o critério de proximidade η , ou um padrão aleatório caso seja a sua primeira chamada.
$P(\xi)$	Distribuição de probabilidade dos sinais de entrada.
Φ_i	Padrões de entrada candidatos no procedimento de MB, com $i = 1, \dots, Q$.
$\Phi_{\mathcal{A}}$	Função de medição de topologia (Villmann et al., 1997).
$\tilde{p}_i$	Estimativa da densidade de probabilidade próximo a w_i .
$\Psi_{\mathcal{M} \rightarrow \mathcal{A}}$	Mapeamento de $\mathcal{M}$ em $\mathcal{A}$.
$\Psi_{\mathcal{A} \rightarrow \mathcal{M}}$	Mapeamento de $\mathcal{A}$ em $\mathcal{M}$.
Q	Número de candidatos no procedimento MB. Quando não citado, usa-se $Q = 10$.
$\mathbf{q}_0$	Configuração, estado ou nodo inicial de uma seqüência (também usa-se q_0 para escalar).
$\mathbf{q}_f$	Configuração, estado ou nodo final de uma seqüência (também usa-se q_f para escalar).
$\mathbf{q}_i$	Vetor de coordenadas generalizadas do robô (Eq. 2.14).

Símbolo	Descrição
$\dot{q}_i$	Primeira derivada do vetor de coordenadas generalizadas do robô (Eq. 2.14).
r	Ruído aplicado a todos os componentes $\xi'_k = r \cdot \xi_k$, $1 \leq k \leq D$ do padrão de entrada $\xi \in \mathbb{R}^D$, calculado como $r = R/100 \cdot I(i)$.
$R\%$	Porcentagem de ruído a se aplicar nas amostras de entrada. Nos testes foi utilizado o valor de $R = 0,1\%$.
ρ	Taxa de aprendizado com decaimento do STRAGEN.
σ_f	Máximo de disparos estimados por nodo. Quando não citado, usa-se $\sigma_f = 2 \cdot t_{\max}/L$.
σ_{s_1}	Número de disparos (vitórias) do nodo s_1 .
s_1	Nodo mais próximo (vencedor) do padrão de entrada por algum critério de proximidade.
s_2	Segundo nodo mais próximo do padrão de entrada por algum critério de proximidade.
$S(t)$	Força do estímulo, normalmente $S(t) = 1$, para função de decaimento do GWR $\overline{h_i}$.
Σ	Sistema dinâmico (Seção 2.2).
T_i	Vetor de posições alvo TPC do VITE (Eq. 3.1).
Θ	Conjunto dos possíveis ângulos das juntas de um robô.
θ	Vetor de ângulos das juntas de um robô. Para o robô bidimensional, θ_1 é o ângulo da junta J_1 (ombro) e θ_2 o ângulo da junta J_2 (cotovelo).
τ	Vetor de torques nas juntas de um robô para dada velocidade e aceleração. Para o robô bidimensional, τ_1 é o torque no extremo da junta J_1 e τ_2 o torque no extremo da junta J_2 .
t	Iteração ou tempo atual.
$t_{\max}$	Número máximo de iterações de treinamento. Quando não citado, usa-se $t_{\max} = 3 \cdot L$.
t_f	Tempo final de convergência de $f_{\text{de}}(\cdot)$.
τ_b e τ_n	Parâmetros de decaimento para a função de frequência de não-disparos no GWR. Valor sugerido: $\tau_b = 3,33$ e $\tau_n = 14,3$.

Símbolo	Descrição
$\mathcal{T}$	A trajetória $\mathcal{T} = \{n_0, n_1, \dots, n_{S-1}, n_S\}$ dada pelos nodos (ou configurações do sistema) Eq. (2.4).
$\mathcal{T}^{-1}$	Trajectoria inversa, a seqüência de nodos iniciando pelo ponto final $n_S = n_{q_f}$ em direção ao ponto inicial $n_0 = n_{q_0}$, onde S é o tamanho (desconhecido) da trajetória encontrada.
T	<i>Matriz de Transformação Homogênea, Matriz do Braço</i> ou simplesmente matriz T é uma matriz 4×4 que descreve as translações e rotações entre os membros do braço de um robô. (Eq. 2.7) (Fu et al., 1987).
$\mathbf{u}(k)$	Entradas de um sistema dinâmico, com $\mathbf{u}(k) \in \mathbb{R}$.
$\mathbf{V}_k$	Grupo k de informações homogêneas que compõe o padrão de entrada, com $k = 1, \dots, m$ (Eq. 4.2).
$\mathbf{V}_\eta$	Subvetor do vetor de pesos $\mathbf{w}$ composto pelo(s) grupo(s) de informações homogêneas utilizado como critério de vizinhança. Idem para $\mathbf{V}_{\eta,i}$, específico para o vetor de pesos $\mathbf{w}_i$ do nodo i .
$\mathbf{V}_\zeta$	Subvetor do vetor de pesos $\mathbf{w}$ composto pelo(s) grupo(s) de informações homogêneas e utilizado como critério de atividade. Idem para $\mathbf{V}_{\zeta_k,i}$, específico para o subgrupo k de ζ com $1 \leq k \leq 1$ e vetor de pesos $\mathbf{w}_i$ do nodo i .
$\mathcal{V}$	Conjunto de nodos $\mathcal{V} \subset \mathcal{A}$ que nunca venceram uma competição, na fase de poda do STRAGEN (Subseção 4.2.4.2).
$\frac{d}{dt}V_i$	Variação no tempo do vetor de diferenças DVV.
V_i	Vetor de diferenças DVV do VITE (Eq. 3.1).
$\mathbf{w}_i$	O vetor de pesos do nodo i de uma RNA, $\mathbf{w}_i \in \mathbb{R}^D$.
ξ	Padrão de entrada de dimensão D .
$\xi(t)$	Padrão de entrada apresentado no tempo t .
ξ_η	Subvetor do padrão de entrada ξ que representa o grupo η definido como critério de vizinhança.
ξ_ζ	Subvetor do padrão de entrada ξ que representa o grupo ζ definido como critério de atividade. Idem para ξ_{ζ_k} , específico para o subgrupo k de ζ com $1 \leq k \leq 1$.
ξ_{η,q_i}	Subvetor η de um vetor de entrada ξ no ponto q_i .

Símbolo	Descrição
$\mathbf{x}(k)$	Estado (ou configuração) de um sistema dinâmico, com $\mathbf{x}(k) \in \mathbb{R}^n$.
$y(t)$	função com decaimento exponencial no tempo $y(t) = y_i(y_f/y_i)^{t/t_{\max}}$, iniciando em y_i e terminando em y_f .
$\mathbf{y}(k)$	Saída de um sistema dinâmico, com $\mathbf{y}(k) \in \mathbb{R}$.
$\hat{\mathbf{y}}$	Saída aprendida pelo modelo identificador (Figura 2.3).
$\mathbf{y}^*$	Saída desejada como resposta a ser aprendida pelo modelo. (Figura 2.3).
ζ	Critério de atividades do STRAGEN, define o grupo de informações homogêneas $\mathbf{V}_\zeta$, tal que $1 \leq \zeta \leq m$.

CAPÍTULO 1

Introdução

Muitos se recusam a aceitar a realidade simplesmente porque entrariam em colapso se o fizessem.

—GOETHE (Carta a Schiller)

No que se refere à auto-organização, muito se tem buscado por propriedades que caracterizariam sistemas como sendo desta categoria (Debrun, 1996). A Teoria de Auto-Organização (TAO) parece pretender fazer uma ponte entre o biológico e o mecânico, assim como fez Turing (1936) com a sua máquina, a ponte entre a intuição do mecanicismo e o mecanismo propriamente dito (Benante, 2001a,b).

Esta ponte entre a vida e a máquina não é trivial (Gonzalez et al., 2005), e muitas questões que sempre estiveram abertas na filosofia estão agora, com o advento da informática, recebendo respostas com um novo enfoque. Este enfoque tem em seu vocabulário termos como *sistemas, sensibilidade a condições iniciais, padrões, aprendizagem, predição, redes neurais artificiais, propriedades emergentes, estados, representação do conhecimento, planejamento, controle*, entre outros.

Cada um desses novos termos, considerados ambíguos e polissêmicos por estudiosos das áreas de humanidades, agora recebem ainda outro significado: eles estão sendo estudados sob a ótica das máquinas, e estão sendo redefinidos formalmente com base em algoritmos. Tais algoritmos surgem como hipóteses para serem corroboradas por meio de testes empíricos, e assim tentar explicar, numa linguagem menos ambígua, como a vida (ou alguma propriedade específica dela) se desenvolve, e atinge o grau de complexidade que se observa e que mesmo assim surpreende em suas imprevisibilidades e em suas auto-adaptações, em outras palavras, em sua auto-organização. Kelso (1999) sugere, por exemplo, que o cérebro é um sistema auto-organizado, que trabalha de acordo com leis sinérgicas, e por essa razão é capaz de apresentar fenômenos macroscópicos tais como a visão, a locomoção e a organização muscular com objetivo de pegar um objeto.

Um objetivo mais específico se planeja e se delineia em torno das idéias organizadas nos

capítulos que se seguem, menos ambicioso que explicar a vida, mas que de algum modo tentará colaborar com a elucidação da complexidade de certos sistemas: pretende-se aqui encontrar explicações que possam lançar luzes sobre como é possível utilizar redes neurais artificiais auto-organizáveis que têm a propriedade de solucionar problemas de planejamento e controle de trajetórias de estados.

Trajetórias de estados incluem quaisquer seqüências válidas de transições entre estados de um sistema aprendido ou pré-determinado, passando por estados intermediários possíveis, partindo de um ponto inicial que pode ser dado ou determinado pelo sistema, e atingindo como alvo um estado qualquer (distinto ou não) do sistema. Como se pode perceber pela caracterização do termo, tem-se uma abrangência muito grande de domínios nos quais se pode empregar esta técnica, bastando que para isso o domínio em questão possa ser caracterizado por estados possíveis e haver transições entre eles.

Mais especificamente, este trabalho propõe e implementa um novo modelo de rede neural artificial auto-organizável chamado de *State Trajectory Generator* (STRAGEN), para geração de trajetórias de estados, em que os pontos fornecidos são aprendidos pela rede em um treinamento não-supervisionado guiado por um procedimento chamado de *Motor Babbling* (MB). Os critérios de otimalidade para criar conexões entre estes pontos podem ser caracterizados em diversos tipos para cada domínio em questão, sendo descritos no próprio padrão de entrada, e utilizados pelo STRAGEN para criar os mapas topológicos. O modelo STRAGEN é apresentado em duas abordagens que diferenciam-se no modo como tratam da eliminação de estados inadequados, chamados de STRAGEN *off-line* e *on-line*. Os domínios utilizados nas simulações incluem: um manipulador robótico bi e tridimensional, uma mão robótica antropomórfica e a análise de progressões harmônicas musicais.

Devido aos múltiplos critérios, o modelo deve ser capaz de gerar trajetórias por estados ou por sinais de controle. Para o planejamento e controle, deve ser tarefa da própria RNA adaptar-se e interpretar as especificidades de cada domínio. A geração de trajetórias pode ainda ser ponto-a-ponto ou determinada para passar por algum(ns) ponto(s) específico(s), chamada de trajetória *via-pontos*.

O modelo proposto neste trabalho deve incorporar determinadas propriedades importantes que serão vistas em diversos dos modelos estudados. As principais propriedades que se pretende incluir são:

- A rede terá agregada em suas conexões uma interpretação que representa algo mais que a trivial proximidade topológica, como critérios para modificação de estados e manutenção dos nodos. A rede deve ser capaz de eliminar nodos de modo dinâmico, para manter o mapa topológico fiel à dinâmica do sistema. A rede deve aprender os estados do sistema,

e também as transições, diferindo-se dos modelos clássicos em que se prioriza apenas o aprendizado de nodos, deixando as conexões formarem-se ao acaso.

- A rede será capaz de aprender o mapa topológico através do processo de MB (Balbuciamiento Motor), que por hora pode-se adiantar tratar-se, grosso modo, de uma simulação de movimentos aleatórios feitos por uma criança para aprender a calibrar o sistema visuomotor.
- A rede deve gerar uma trajetória em um custo computacional baixo se comparado aos modelos estudados, e com uma boa qualidade segundo os critérios estabelecidos. Dois algoritmos base para a geração de trajetórias serão investigados, a difusão de energia e o algoritmo de menor caminho entre dois pontos, e um novo modelo será originado destes modelos base, que pretende gerar uma trajetória não-ótima, mas boa, de custo adequado para ser capaz de trabalhar *on-line*.
- O modelo proposto terá que executar a trajetória (controle), e ser capaz de manter velocidades e acelerações coerentes para as trajetórias geradas. As trajetórias geradas poderão ter como entrada pontos intermediários, chamadas de *trajetórias via-pontos*. O modelo será capaz de cumprir a *equifinalidade temporal* (ET), ou seja, o movimento das diversas partes do sistema ocorrem simultaneamente, com velocidades compatíveis com o espaço percorrido para cada parte, de modo que todo o sistema atinja a configuração final ao mesmo tempo. Também deve permitir a troca de alvo durante a execução de uma trajetória, importante propriedade para a utilização real em um ambiente dinâmico.
- Ainda sobre as conexões de vizinhança, para o caso de trajetórias robóticas, pretende-se criar trajetórias com diversos critérios, para que se possa analisar a influência do mapeamento dos dados durante a fase de treinamento na trajetória final gerada. Entre esses critérios cita-se: variação mínima de torques, variação mínima de ângulos (braço preguiçoso ou *lazy arm*) e menor distância espacial.
- A trajetória formada pela rede como propriedade emergente: essa categoria de propriedade refere-se ao fato da rede não representar diretamente a trajetória, mas ter a trajetória como um processo realizado pelos nodos de saída da rede. O modelo proposto mapeia o espaço de estados de forma auto-organizada, que por sua vez representa o conjunto de possíveis trajetórias aprendidas. A trajetória gerada, apesar de representada explicitamente, é criada pelas informações aprendidas no mapa auto-organizado, durante a fase de treinamento, e atualizado durante a fase de execução, ou seja, deve-se buscar a emergência no processo de mapeamento e não na trajetória em si.

- Finalmente, como propriedade fundamental do modelo, sua característica dinâmica. O modelo deve conseguir realizar todas essas tarefas e apresentar tais referidas propriedades partindo de uma rede mínima, chamada de SOM-DT (*Self-Organizing Maps with Dynamic Topology*) que cresce (ou diminui) durante o treinamento. Esta classe de redes se desenvolve mapeando os dados de entrada e criando nodos conforme há necessidade para um mapeamento adequado, diferente dos modelos SOM originais de estrutura definida. Com isto, simplifica-se em grande parte os parâmetros necessários para a criação do modelo, facilitando sua implementação, e consegue-se uma rede de um tamanho mínimo (em número de nodos e conexões) necessário para mapear o espaço de estados com precisão, o que confere a ela um custo computacional variável adaptado a cada simulação.

Muitos modelos de topologia dinâmica possuem características importantes que se pretende alcançar no modelo proposto, e este trabalho irá fazer uma revisão histórica direcionada a estes trabalhos. Cita-se no texto que se segue os principais modelos. Entre estes, os que serão de maior importância são: *Self-Organizing Map* (SOM) (Kohonen, 1987), pela sua inserção histórica, *Topology Representing Networks* (TRN) (Martinetz et al., 1991), por sua característica de crescimento combinando dois algoritmos distintos e bem estabelecidos (*Neural Gas* e *Competitive Hebbian Learning*), *Growing Cells Structures* (GCS) (Fritzke, 1994a), pelas inovações na variação do mapa topológico, *Growing Neural Gas* (GNG) (Fritzke, 1997a), pela evolução que representa em relação ao GCS. O modelo *Grow when Required Network* (GWR) (Marsland et al., 2002) merece destaque por sua capacidade de adaptação, entre diversas outras propriedades importantes que serão estudadas neste algoritmo, que será base do modelo a ser proposto. E os modelos derivados do *Vector Integration to Endpoint* (VITE) (Bullock & Grossberg, 1988b,a), como o DIRECT (Fiala, 1994; Izquierdo et al., 1996), *Adaptive VITE* (AVITE) (Gaudiano & Grossberg, 1991, 1992), *extended VITE* (EVITE) (Bullock et al., 1998, 1999), por apresentarem propriedades neurofisiológicas interessantes no campo de movimentos de braços de robôs.

Será feita a revisão de alguns destes modelos, e outros serão relacionados nos capítulos que se seguem, visando o entendimento de como surgem as propriedades de interesse desta tese, e como tais modelos implementam estas propriedades. Os resultados obtidos indicam que o modelo proposto possui uma característica inovadora, que se concentra principalmente em entender o papel das conexões nos mapas auto-organizáveis e como estas conexões podem influenciar o resultado final. Em trajetórias de estados, além de mapear com a precisão determinada o espaço de entrada, é importante observar como as conexões que ligam os nodos estão relacionadas com a topologia, ou ainda, com os critérios necessários para uma correta transição

entre os estados do sistema.

O modelo proposto, além de ser um modelo completo que faz o mapeamento do espaço de entrada (chamada de fase 1) e a geração da trajetória de estados (chamada de fase 2), ainda conta com a opção de utilizar diferentes critérios de vizinhança (que serão vistos adiante em momento oportuno). Alguns resultados interessantes foram atingidos, derivados do modo que as fases 1 e 2 foram implementadas. Além disso, comparações entre os diferentes critérios de vizinhanças, suas propriedades, a análise de comportamento do modelo para pontos gerados por balbuciamiento motor no espaço, a questão da eliminação e inserção de nodos e os resultados obtidos são feita ao final do documento.

O documento que se segue está assim organizado:

No Capítulo 2 será introduzido formalmente o problema a ser tratado nesta tese, a saber, o problema de geração, planejamento e controle de trajetórias de espaços de estados, utilizando redes neurais artificiais auto-organizáveis de topologia dinâmica (SOM-DT). Além disso, tem-se uma breve relação histórica e um resumo de algumas propriedades importantes de cada um dos modelos que serão tratados neste trabalho, para propiciar ao leitor uma visão geral sobre estes modelos e o estado da arte atual.

No Capítulo 3 serão revisados alguns dos principais modelos de redes neurais artificiais (RNAs) auto-organizáveis (com topologias fixas e dinâmicas), e levantadas algumas características que se mostraram promissoras nestes modelos, em busca de um novo algoritmo para a resolução de problemas de geração de trajetórias de espaços de estados, em especial trajetórias robóticas. Será visto o modelo VITE e seus derivados, que dá uma visão geral sobre importantes propriedades neurofisiológicas que os autores abordaram, em uma rede auto-organizada, mas de estrutura fixa. Com o objetivo de investigar os modelos de Redes Neurais Artificiais (RNA) não-supervisionados, de estruturas topológicas dinâmicas, este trabalho será direcionado para uma revisão dos modelos que atendam estas características, total ou parcialmente.

No Capítulo 4 é feita uma análise dos requisitos até então levantados, e em seguida são descritas duas possíveis abordagens para a solução da classe de problemas proposta neste trabalho, o STRAGEN *off-line* e o *on-line*. As características de cada abordagem, bem como suas diferenças em relação a modelos anteriores e entre si são então discutidas.

No Capítulo 5 apresenta-se os resultados gerados pelo modelo proposto, e uma comparação entre diversos critérios utilizados e entre outros modelos. Além disso há uma descrição mais prática de como os resultados foram conseguidos.

O Capítulo 6 finaliza com uma conclusão sobre os avanços conseguidos e uma discussão sobre as possibilidades que foram levantadas e as que estão em aberto para trabalhos futuros.

CAPÍTULO 2

Descrição do Problema

Grande e pequeno, rápido e lento só existem na mente, pois são inteiramente relativos, e mudam segundo a posição e a ordem dos órgãos dos sentidos.

—BERKELEY (Tratado sobre o conhecimento humano)

Este capítulo visa introduzir e conceitualizar o problema de mapeamento de um espaço de estados para a geração de trajetórias que será objeto de estudo deste trabalho. Na Seção 2.1 é apresentado o problema em suas nuances, em uma série de abordagens distintas. Na Seção 2.2 é detalhado mais especificamente o problema de geração de trajetórias de espaço de estados. Na Subseção 2.2.4 é apresentada uma abordagem para a solução destes problemas utilizando Redes Neurais Artificiais (RNAs). Na Seção 2.3 são dados exemplos de domínios em que a geração de trajetórias de estados pode se aplicar e que foram levantadas nessa tese: trajetórias robóticas, o problema do cálculo da cinemática direta e inversa para a obtenção de pontos da trajetória (Subseção 2.3.1.1) e o problema do cálculo da dinâmica direta e inversa com o mesmo intuito (Subseção 2.3.1.2), controle de uma mão antropomórfica robótica (Subseção 2.3.2) e análise de progressões harmônicas (Subseção 2.3.3).

2.1 Abordagens para Planejamento

Um problema central na Inteligência Artificial (IA), o *Planejamento*, surge da necessidade de se selecionar ações que atuam em um meio (dinâmico ou estático), para alcançar um objetivo (Russel & Norvig, 2002); e está intimamente relacionado com outras duas importantes categorias de problemas. O problema de controle, cuja base da *Teoria de Controle* foi fundamentada por Wiener (1948), mas já se apresentava à humanidade há muito tempo, com exemplos como relógios d'água na antiguidade (250 A.C.), máquinas a vapor auto-reguladas (1700–1800) e o termostato de Cornelis Drebbel (1572–1633). A segunda categoria inclui o problema de *Representação do Conhecimento*, ou seja, o estudo de como expressar informações de modo estruturado para posterior utilização, que na filosofia pode remontar à figuras de linguagens platônicas

(350 A.C.). Na computação, porém, a discussão sobre Representação do Conhecimento (Verschure, 1998) fica mais nítida com a separação que surgiu entre a chamada Inteligência Artificial Clássica, cujo marco inicial é considerado como sendo a criação do programa de raciocínio *Logic Theorist* (LT) de Newell e Simon em 1956, que utilizava estruturas de dados simbólicos, e o campo das Redes Neurais Artificiais (RNAs), inaugurado por McCulloch & Pitts (1943), em que se propunha que o conhecimento podia ser representado de modo distribuído em diversas pequenas unidades de processamento.

Enquanto nos primórdios da IA, as Redes Neurais Artificiais tinham seu alcance limitado, até mesmo questionada sua viabilidade (um exemplo conhecido é a limitação dos perceptrons apontada por Minsky & Papert (1969)), a corrente simbólica ganhava terreno com avanços em diversas áreas, entre elas algoritmos de busca em espaço de estados utilizados para resolver problemas de planejamento e controle.

Na área de sistemas inteligentes (IA e RNA), o problema de planejamento e controle tem sido abordado de três formas principais (Bonet & Geffner, 2001):

- *Modelos de Programação Procedimental*: Abordagem na qual o controlador é programado *na mão* usando uma linguagem procedimental de alto nível, em um sistema ou em camadas.
- *Modelos de Planejamento*: O controlador é automaticamente derivado de uma descrição dos estados, ações e objetivos.
- *Modelos de Aprendizado*: O controlador é derivado de experiências.

2.1.1 Modelos de Programação Procedimental

Como exemplos, para robôs móveis, da primeira abordagem, pode-se citar os trabalhos Brooks (1987) e Agre & Chapman (1990).

A visão de um algoritmo de planejamento como uma sequência de instruções a ser processada para atingir um objetivo (*plan-as-program*, (Agre & Chapman, 1990)) permite se pensar em soluções para o problema de planejamento e controle que são adequadas aos interpretadores/compiladores de linguagens de programação. Um exemplo tradicional de um sistema decomposto em camadas é dado na Figura (2.1).

O maior problema desta abordagem está na especificidade que o algoritmo carrega, tornando a solução pouco ou nada flexível para usos diversos. Brooks (1987) considera que as propriedades mais importantes para um sistema de planejamento e controle são: (a) a capacidade de perseguir múltiplos objetivos, às vezes conflitantes entre si; (b) a capacidade de inte-

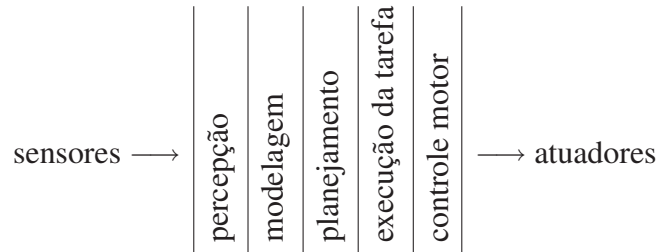


Figura 2.1 Decomposição tradicional de um sistema de controle de robôs em módulos funcionais (Brooks, 1987).

grar a leitura de múltiplos sensores; (c) a robustez, ou seja, o robô deve ser capaz de absorver mudanças bruscas no ambiente, ou a perda de sensores, e continuar com um comportamento minimamente inteligente em busca do seu objetivo; (d) extensibilidade, isto é, a facilidade de inclusão de novos sensores ou novas capacidades ao sistema.

Robôs reativos podem ser colocados nesta categoria, uma vez que as camadas de programação para a reação a determinados eventos são normalmente feitas via programação procedimental. Tais robôs não exibem um algoritmo completo para o planejamento de trajetórias, apesar de que conseguem mostrar comportamentos cooperativos que emergem das leis do programa (Willems & Haselager, 2003).

2.1.2 Modelos de Planejamento

A segunda das abordagens citadas, o planejamento, combina técnicas da IA, programação dinâmica e lógica, e é capaz de lidar com sistemas em ambientes determinísticos ou probabilísticos. O planejamento tradicional da IA consiste no uso de linguagens de alto nível para a descrição da solução, modelos matemáticos para a descrição do ambiente, suas ações e objetivos, e algoritmos de busca heurísticas para a obtenção destas soluções (Bonet & Geffner, 2000; Fikes & Nilsson, 1971). Um exemplo é o sistema para geração de trajetórias de aviões de Waydo et al. (2007), que usa programação lógica.

Três dos mais importantes modelos matemáticos para descrever o tipo de planejamento, de acordo com a informação disponível no ambiente são: o modelo de estados determinístico (para a resolução de problemas de controle determinísticos (DCP, *Deterministic Control Problem*); o modelo conhecido como MDP (*Markov Decision Process*), e o processo de Markov parcialmente observável, ou POMDP (*Partially Observable Markov Decision Process*).

- *Problema de Controle Determinístico* (DCP):

O planejamento de uma trajetória de estados em um espaço é classicamente entendido como

(Bertsekas, 2000) um modelo de estados definido sobre um espaço de estados $\mathcal{M}$ discreto e finito, com um espaço inicial q_0 , um alvo específico q_f ou um conjunto de alvos $\mathcal{M}_f \subseteq \mathcal{M}$ válidos, e um número finito de ações A que mapeia deterministicamente cada estado, chamado de *Modelo de Controle Determinístico*. Este modelo é caracterizado por:

- P1.** Um espaço de estados finito $\mathcal{M}$.
- P2.** Um estado inicial $q_0 \in \mathcal{M}$.
- P3.** Ações $A(q) \subseteq A$ aplicáveis a cada estado $q \in \mathcal{M}$.
- P4.** Uma função determinística de transição $f_{tr}(q, a)$ para cada $q \in \mathcal{M}$ e $a \in A(q)$.
- P5.** Um custo associado a cada ação $c(a, q) > 0$.
- P6.** Um ou mais alvos dado por um conjunto não vazio $\mathcal{M}_f \subseteq \mathcal{M}$.

Este caso, chamado de *planejamento clássico*, pode ser formulado como um caso de busca heurística em um espaço de estados. Não há qualquer algoritmo para a criação de um mapeamento, uma vez que o espaço de estados e seu mapeamento é dado de antemão. Mas nem sempre se tem tanta informação sobre o sistema a ponto de se construir um mapeamento como este.

Uma *solução* para um problema de controle determinístico é uma sequência de ações $a_0, a_1, \dots, a_n$ que gera uma trajetória de estados $\mathcal{T} = \{q_0, q_1 = f_{tr}(q_0, a_0), \dots, q_{n+1} = f_{tr}(q_n, a_n)\}$ tal que cada ação a_i é aplicável ao estado q_i , e o estado final desejado (alvo) é q_{n+1} . O custo de uma trajetória é a soma dos custos de cada ação $\sum_{i=0}^n c(a_i, q_i)$, e a trajetória é ótima se o custo é mínimo.

- *Processo de Decisão de Markov* (MDP):

O modelo de controle determinístico assume que uma ação aplicada em um estado leva com 100% de certeza a outro estado ou configuração do sistema. No Processo de Decisão de Markov há uma probabilidade associada a cada ação acerca de sua configuração resultante. Enquanto que no modelo DCP não há necessidade de se observar o resultado de uma ação, o modelo MDP assume que os efeitos de uma ação são completamente observáveis. O MDP pode ser descrito como:

- P1.** Um espaço de estados finito $\mathcal{M}$.
- P2.** Um estado inicial $q_0 \in \mathcal{M}$.
- P3.** Ações $A(q) \subseteq A$ aplicáveis a cada estado $q \in \mathcal{M}$.

- P4.** Uma função de transição que define as probabilidades $P_a(q'|q)$ para q e $q' \in \mathcal{M}$ e $a \in A(q)$.
- P5.** Um custo associado a cada ação $c(a, q) > 0$.
- P6.** Um ou mais alvos dado por um conjunto não vazio $\mathcal{M}_f \subseteq \mathcal{M}$.

Os estados resultantes de uma ação não são *previsíveis*, porém é possível observar *a posteriori* o estado final, o que permite uma reavaliação para se escolher a ação a ser aplicada em seguida. Desta forma, uma solução para o MDP não é uma trajetória de estados (ou sequência de ações), mas uma função π , chamada de *política*, que mapeia estados q em ações $a \in A(q)$. Uma política π determina as probabilidades de cada trajetória de estados $q_0, q_1, q_2, \dots$, iniciando em q_0 , dada pelo produto das probabilidades de todas transições $P_a(q_{i+1}|q_i)$ e $a_i = \pi(q_i)$. Uma *solução ótima* para o MDP é a política de controle π^* que tem o menor *custo esperado*.

- *Processo de Decisão de Markov Parcialmente Observável (POMDP):*

Uma generalização do MDP que permite modelar um ambiente em que os resultados das ações sejam *parcialmente observáveis* é chamado de POMDP (*Partially Observable Markov Decision Process*). A descrição de um modelo POMDP é similar ao MDP, com a adição de um novo conjunto de crenças em estados, que é utilizado para guiar o processo de solução enquanto não se tem informação sobre o real estado do sistema. A descrição do POMDP pode ser feita conforme abaixo:

- P1.** Um espaço de estados finito $\mathcal{M}$.
- P2.** Um estado inicial $q_0 \in \mathcal{M}$.
- P3.** Ações $A(q) \subseteq A$ aplicáveis a cada estado $q \in \mathcal{M}$.
- P4.** Uma função de transição que define as probabilidades $P_a(q'|q)$ para q e $q' \in \mathcal{M}$ e $a \in A(q)$.
- P5.** Um custo associado a cada ação $c(a, q) > 0$.
- P6.** Um ou mais alvos dado por um conjunto não vazio $\mathcal{M}_f \subseteq \mathcal{M}$.
- P7.** A crença num estado inicial b_0 .
- P8.** Um conjunto O de observações o com probabilidades $P_a(o|q)$.

A *solução* para um modelo POMDP é uma função que mapeia os estados de crença em ações, uma vez que mapear os estados reais não é possível pois que nem sempre são conhecidos. O efeito das ações nos estados de crença são totalmente previsíveis, de modo que se não houver nenhuma observação o modelo reduz-se a um problema de controle determinístico (DCP) nos

estados de crença. Neste caso, a solução é encontrar uma trajetória do estado de crença inicial b_0 para o estado de crença final b_f (ou conjunto de estados finais que satisfaçam o problema).

Na presença de observações (realimentação de sensores, por exemplo), um POMDP deixa de ser determinístico (no espaço de crenças) e passa a ser um MDP no espaço de crenças, isto é, a *solução ótima* é uma política de controle que mapeia estados de crença em ações tais que minimizam o custo *esperado* de ir de um estado de crença inicial b_0 a um estado de crença final b_f .

O problema de planejamento na presença de sensores, no *Modelo de Planejamento*, pode ser descrito como um POMDP cujas soluções são políticas que mapeiam estados de crença em ações. Na literatura de IA clássica, ou planejamento com *Modelos de Programação Procedimental*, tais políticas costumam ser representadas como planos de contingência, ou seja, planos sequenciais complementados por testes e ramificações (Bonet & Geffner, 2001).

2.1.3 Modelos de Aprendizado

A terceira abordagem citada, o *aprendizado*, inicia-se historicamente com três publicações que delineiam o campo das Redes Neurais Artificiais (RNAs): o trabalho de McCulloch & Pitts (1943) sobre a computabilidade de redes neurais baseadas em perceptrons; o famoso livro *Cibernética* de Wiener (1948) que descreve conceitos importantes da teoria de controle, comunicação e processamento de sinais; e principalmente o livro *The Organization of Behaviour* de Donald Olding Hebb, publicado em 1949, que explicita pela primeira vez uma regra fisiológica de aprendizado para modificações sinápticas, mais tarde conhecida como regra de Hebb, que indica que uma sinapse tem sua efetividade aumentada pela repetida ativação de um neurônio por outro através desta sinapse.

Durante as décadas de 60 e 70, poucos avanços foram realizados, e problemas importantes para o desenvolvimento e uso das RNAs ainda estavam sem solução, como o problema de “atribuição de créditos” para perceptrons multicamadas, por exemplo. Porém nesse período pode-se destacar um grande avanço de paradigma: Winograd & Cowan (1963) interessados nos trabalhos anteriores de von Neumann sobre a confiabilidade das unidades de processamento (ou neurônios artificiais), sugeriram pela primeira vez um novo paradigma de *Representação do Conhecimento* através da redundância de informações distribuídas em diversas unidades paralelas que podiam coletivamente representar um conceito individual, o que acarretou um aumento na robustez e no paralelismo que as RNAs passariam a oferecer (Haykin, 1994). Hoje há sistemas que tentam fazer o caminho inverso: extrair regras do conhecimento distribuído representado nos nodos das RNAs (Campos & Ludermir, 2005).

Mas foi na década de 80 que as RNAs voltaram a ter seu *status* de campo frutífero renovado. Entre os principais avanços desta época, tem-se: A criação das Redes Neurais Auto-organizadas, por Kohonen (1987), que abriu todo um novo campo para o uso de RNAs – apesar de Kohonen ser historicamente considerado o divulgador do conceito de SOM, já haviam trabalhos preliminares propostos por Willshaw & von der Malsburg (1976). O desenvolvimento da teoria da ressonância adaptativa (ART, *Adaptive Resonance Theory*), por Grossberg (1976), que estabeleceu um novo princípio de auto-organização baseado em uma troca de informações dinamicamente entre duas camadas em ressonância até a estabilização da rede. A criação das *redes de Hopfield* cujo princípio é armazenar informações em uma rede dinâmica estável, e permitiu uma visão inovadora do modo de funcionamento das redes recorrentes (Hopfield, 1982). O desenvolvimento do famoso algoritmo de *back-propagation* (retropropagação), por Rumelhart et al. (1986), que influenciou sobremaneira todo o desenvolvimento posterior do campo de Redes Neurais Artificiais. Rumelhart et al. (1986) divulgaram o modo de como utilizar o algoritmo *back-propagation* para aprendizado de máquina, porém anos depois descobriu-se que foi Werbos que em 1974 descreveu primeiramente este algoritmo em sua tese de doutorado na universidade de Harvard. A criação de um novo modelo alternativo aos perceptrons multicamadas (e alternativo ao algoritmo *back-propagation*), chamado de *Radial Basis Functions* (RBF), por Broomhead & Lowe (1988), de especial interesse para análise numérica. E de especial interesse para a área de controle, um artigo de Barto et al. (1983) sobre *Aprendizado por Reforço* mostrou como era possível criar um sistema que aprendesse a equilibrar um pêndulo invertido sem necessidade de um supervisor externo. Apesar de que o Aprendizado por Reforço já tivesse sido estudado na tese de doutorado de Minsky em 1954, foi o artigo de Barto, Sutton e Anderson que conseguiu chamar a atenção deste tipo de aplicações para RNAs. Em 1996, Bertsekas e Tsitsiklis formalizaram matematicamente o Aprendizado por Reforço, no livro *Neurodynamic Programming*.

Na literatura, pode-se dividir os modelos usados para planejamento e controle por RNAs em cinco grandes categorias: Controle Supervisionado, Controle Inverso Direto, Controle Neural Adaptativo, Retropropagação com Utilidade e Métodos Adaptativos com Críticos (Werbos, 1991).

- *Métodos de Controle Supervisionados* (SCM):

Os *métodos de controle supervisionados* são aqueles em que uma RNA aprende o mapeamento dos sensores de entrada já previamente relacionadas com as ações desejadas, adaptando seus pesos pelo treinamento de um banco de dados criado com informações dadas por um *expert* humano. Guo et al. (1995) usou o algoritmo de *back-propagation* para controlar um AUV

(*Automatic Underwater Vehicle*) e reproduzir trajetórias treinadas. Fuji & Ura (1991) criaram um controlador para o veículo aquático PTEROA, utilizando uma rede supervisionada que usa como supervisor uma outra rede nebulosa (*fuzzy*), chamada de pré-controlador.

É de se notar que a maioria dos esforços para a solução de problemas geração de trajetórias de estados caem nesta categoria, que inclui os modelos baseados em retropropagação empregadas em perceptrons de multicamadas (MLP) e os modelos baseados em Funções de Base Radial (RBF) (Araújo & Barreto, 2002), já famosos por sua grande aplicabilidade e que também já têm críticas bem fundamentadas sobre o processo de atuação, limitando a flexibilidade das suas estruturas internas, entre outras.

- *Métodos de Controle Inverso Direto (DICM):*

Os *Métodos de Controle Inverso Direto*, segundo Werbos (1991) são aqueles em que uma RNA aprende a dinâmica inversa de um sistema, de modo que possa fazer com que o sistema siga uma trajetória desejada. O treinamento da rede é realizado em uma fase inicial, em que o sistema é permitido navegar à deriva em seu espaço de estados, aleatoriamente, e para cada estado atingido a rede aprende os valores obtidos pelas informações sensoriais daquele estado (Barreto & Araújo, 2001; Martinetz et al., 1990; Morasso & Sanguineti, 1995; Walter & Schulten, 1993). O modelo VITE (*Vector Integration to Endpoint*) (Gaudiano & Grossberg, 1991) é considerado o pioneiro em utilizar essa técnica com relativo sucesso, através de uma rede auto-organizada de arquitetura rígida (explanada adiante). Também o modelo SOBoS (*Self-Organizing Body Schema*) utiliza uma rede auto-organizada para o treinamento do mapeamento não-linear sensório-motor (Sanguineti & Morasso, 1992), através de uma fase de geração aleatória de movimentos, chamada de fase de balbucimento ou *babbling phase*. Conforme Barreto et al. (2003b), os modelos baseados em Mapas auto-organizáveis (*Self-organizing Maps* – SOM), superam os problemas de pré-definição da estrutura rígida dos modelos que compõem a categoria anterior (dos *Métodos de Controle Supervisionados*) por serem mais flexíveis, mais simples, e não necessitarem de supervisor externo.

- *Métodos de Controle Neural Adaptativo (NACM):*

Na abordagem chamada de *Métodos de Controle Neural Adaptativo*, modelos para representação de sistemas dinâmicos não-lineares, mais especificamente o modelo matemático NARMA (*Nonlinear Autoregressive Moving Average*) que dá uma exata representação da entrada-saída de um sistema dinâmico (sob certas condições), são substituídos por modelos neurais (neuroidentificadores e neurocontroladores), normalmente RNAs recorrentes ou baseadas em redes multicamadas, resultando em uma grande robustez e na capacidade de lidar com a

não-linearidade (Werbos, 1991; Narendra & Mukhopadhyay, 1997; Narendra & Parthasarathy, 1990).

- *Métodos Baseados na Retropropagação da Utilidade (ACM):*

Os *Métodos Baseados na Retropropagação da Utilidade*, que normalmente são implementados com redes de topologias similares às redes de retropropagação através do tempo (Werbos, 1990, 1991), consistem em adaptar uma RNA para aproximar um controlador *ideal* através do ajuste de pesos que otimizam as ações da rede (i.é., a *utilidade* da ação), ao invés do método original de ajustes de pesos dos parâmetros do controlador. Trabalhos nesta linha foram feitos Jordan em 1986 e também por Zipser e Elman (veja Elman (1990) e Werbos (1991)). Kawato (1990) propôs um esquema de aprendizado de dinâmica inversa que supostamente funciona para quaisquer topologias de RNAs capazes de aproximar funções não-lineares objetivas através de uma minimização do sinal de erro (o que inclui a MLP). Porém o sinal de erro deve provir de um Controlador por Realimentação Convencional (CFC), i.é., o sistema dinâmico deve ter um modelo prévio que descreve seu comportamento – que pode ser uma outra RNA já treinada (Gomi & Kawato, 1990; Kawato, 1990).

- *Métodos Adaptativos com Críticos (ACM)*

Os *Métodos Adaptativos com Críticos* são uma família complexa de técnicas de otimização que combinam os conceitos de aprendizado por reforço (Bishop, 1996a,b; Sutton & Barto, 1998) e programação dinâmica (Werbos, 1991). Para uma dada série de ações de controle que precisam ser executadas em seqüência, e das quais não se sabe o seu resultado conjunto até que todas as ações forem executadas, torna-se impossível desenvolver um controlador ótimo utilizando técnicas tradicionais de RNAs supervisionadas (Venayagamoorthy et al., 2003). O método adaptativo crítico determina as leis de controle ótimas para um sistema por sucessivos ajustes de duas RNAs em trabalho conjunto. A primeira, chamada de RNA de ação (ou “ator”), atua no sistema enviando os sinais de controle, e a segunda, a RNA “crítico”, aprende a performance desejada. Em conjunto, essas redes aproximam a equação de Hamilton-Jacobi-Bellman da teoria de controle ótimo. A idéia de usar um “crítico” permite adiantar o processo de aprendizado da rede controladora (ou “ator”), por exemplo uma rede de retropropagação. Enquanto que uma rede de retropropagação sozinha busca ajustar seus parâmetros através das derivadas dos erros em respeito às suas entradas, uma vez acoplada a uma outra rede que faz o papel de “crítico”, normalmente uma rede de aprendizado por reforço, a rede “ator” passa a ajustar seus parâmetros em cima de respostas já refinadas pelo “crítico” que podem apresentar

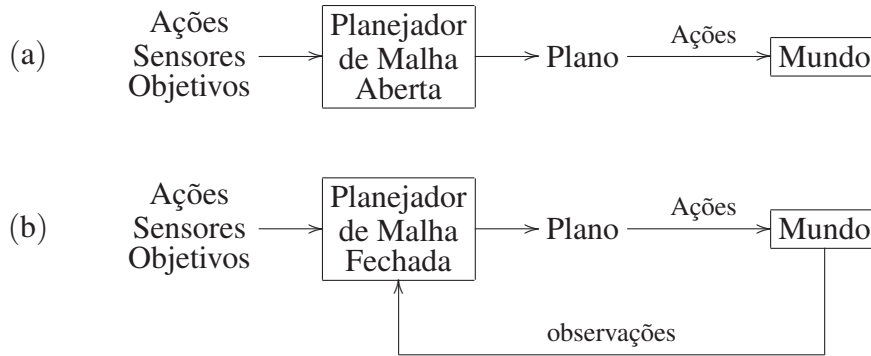


Figura 2.2 Sistemas conforme a realimentação: (a) Malha Aberta: sem realimentação e (b) Malha Fechada: com realimentação.

certos atributos desejados do sistema (Venayagamoorthy et al., 2003). Um exemplo deste método é empregado por Miyamoto et al. (2004) para o aprendizado de trajetórias via-pontos para controle motor de um pêndulo invertido (*cart-pole problem*).

Estas foram as cinco categorias de modelos de aprendizado para sistemas dinâmicos. Além destas categorias, os sistemas de controle ainda podem ser classificados como sistemas de *Malha Aberta* e sistemas de *Malha Fechada* (Bonet & Geffner, 2001). A diferença entre eles é que um sistema de malha fechada utiliza a realimentação do sistema para se corrigir automaticamente (Gomi & Kawato, 1990), mantendo-se o objetivo. O sistema de malha aberta recebe as informações completas da tarefa a ser planejada, gera um plano e encaminha esse plano ao sistema, sem tratar de modo especial a realimentação (Figura (2.2)). Entretanto, um sistema de malha aberta não está obrigado a deixar de atuar após receber uma tarefa. Um sistema de malha aberta rápido e dinâmico pode usar sua entrada para replanejar e refazer as trajetórias sempre que lhe for requisitado, gerando um novo plano que cumpre as exigências dadas.

Como se pode notar nestas cinco categorias de modelos de aprendizado para planejamento, além das possibilidades de se usar malha aberta ou fechada, bem como nos métodos de IA clássica levantados no início do capítulo, a geração de trajetórias de estados, o planejamento e controle de sistemas dinâmicos são áreas em franco desenvolvimento e de grande interesse na pesquisa corrente em áreas como Ciências da Computação, Teoria de Controle, Estatística, Programação Dinâmica. É interessante observar que no período de 1990–1995, uma rápida pesquisa com os termos “neural networks” revelou que foram publicados 9955 artigos sobre Redes Neurais Artificiais, dentre estes, as áreas de *Aproximação de Funções* e *Reconhecimento de Padrões* se destacaram com aproximadamente 8000 artigos e a área de *Neurocontroladores* teve publicados 1960 artigos diversos (Narendra, 1996).

O breve levantamento feito nesta seção mostra a abrangência da área de neurocontroladores. Entre as possíveis áreas para trajetórias de estados que foram exploradas por soluções baseadas em redes neurais artificiais pode-se citar por exemplo: máquinas de estados finitos (Fahlman & Lebiere, 1990; Fahlman, 1991), reprodução de trajetórias (Araújo & Barreto, 2002), identificação e controle de sistemas dinâmicos (Barreto & Araújo, 2004; Narendra & Parthasarathy, 1990; Werbos, 1990; Chen & Narendra, 2002; Mazak et al., 1996; Gomi & Kawato, 1990), controle de reator de polimerização (Mazak et al., 1996), manipuladores robóticos (Barreto et al., 2003b; Gaudiano & Grossberg, 1991; Bullock et al., 1998), análise léxica de textos (Elman, 1990), planejamento em domínios estocásticos (Kaelbling et al., 1998; Bonet & Geffner, 2000), controle motor (Miyamoto et al., 2004), locomoção ou planejamento de trajetórias para robôs móveis (Glasius et al., 1995; Chauvin et al., 2003; Nehmzow, 2001; Bobrow, 1988; Keymeulen & Decuyper, 1996; Provost et al., 2006), robôs móveis aquáticos (Fuji & Ura, 1991) e robôs móveis bípedes (Miller, 1994), para robôs móveis usando redes nebulosas (Kubota et al., 2006), modelos de previsão na área econômica (Werbos, 1990), controle visuomotor (Walter & Schulten, 1993), percepção visual de profundidade da retina humana (Nishina & Kawato, 2004), percepção de seqüências musicais (Page, 1999; Rauber et al., 2002a), geração de trajetórias via-pontos (Wada & Kawato, 2004; Bullock et al., 1999), regulador de voltagem e controle para turbinas geradoras de energia (Venayagamoorthy et al., 2003), controle de vazão de gás natural (Werbos, 1991), reconhecimento de voz (Illina & Gong, 1996), predição de trajetória de objetos móveis (Park et al., 2007), entre outras quaisquer em que se pode representar um sistema dinâmico através de um conjunto de estados de configuração.

Tendo a visão geral desta seção servido para localizar as possíveis abordagens para gama tão variada de problemas, este trabalho visa contribuir com a geração de trajetórias de estados, em especial com modelos de aprendizagem através de redes neurais artificiais com topologia dinâmica e métodos de controle inverso direto, como uma solução original na literatura.

2.2 Descrição do Problema

A seção anterior apresentou diversas abordagens em que o problema de geração de trajetórias num espaço de estados pode ser entendido. Este trabalho tem por objetivo abordar o problema do ponto de vista do mapeamento auto-organizado de um espaço de estados, para posterior geração de trajetórias.

Um *sistema dinâmico* pode ser definido como uma descrição matemática determinística da evolução dos estados de um sistema no tempo (Ott, 2002, p.6). Tal definição de sistema é muito

geral e pode adequar-se a diversos problemas, como já visto.

Seja um sistema dinâmico qualquer Σ desconhecido, que pretende-se mapear:

$$\begin{aligned}\Sigma : \quad \mathbf{x}(k+1) &= \mathbf{f}[\mathbf{x}(k), \mathbf{u}(k)] \\ \mathbf{y}(k) &= \mathbf{h}[\mathbf{x}(k)]\end{aligned}\tag{2.1}$$

onde $\mathbf{u}(k)$ representa as entradas do sistema, $\mathbf{x}(k)$ representa o estado (ou configuração) do sistema, e $\mathbf{y}(k)$ representa a saída do sistema; com $\mathbf{x}(k) \in \mathbb{R}^n$, $\mathbf{u}(k) \in \mathbb{R}$, $\mathbf{y}(k) \in \mathbb{R}$, $f : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$, $\mathbf{h} : \mathbb{R}^n \rightarrow \mathbb{R}$. Assume-se que $\mathbf{f}(0,0) = 0$, de modo que se considera o estado de equilíbrio do sistema (2.1) na origem (Khalil, 2001).

O problema de controle consiste em manter as variáveis do sistema em equilíbrio, ou influenciá-las na direção de uma trajetória (Narendra, 1996). Normalmente o sistema a ser modelado se enquadra em um destes três casos:

1. $\mathbf{f}$ e $\mathbf{h}$ são conhecidos, e os estados são acessíveis;
2. $\mathbf{f}$ e $\mathbf{h}$ são desconhecidos, mas os estados são acessíveis;
3. $\mathbf{f}$ e $\mathbf{h}$ são desconhecidos, os estados não são acessíveis; somente entradas e saídas são acessíveis.

Como a saída do sistema depende apenas de sua entrada e do estado anterior, Equação (2.1) pode ser reescrita da seguinte forma (Narendra & Mukhopadhyay, 1997):

$$\begin{aligned}\mathbf{y}(k+1) &= F[\mathbf{y}(k), \dots, \mathbf{y}(k+n-1); \\ &\quad \mathbf{u}(k), \dots, \mathbf{u}(k-n+1)]\end{aligned}\tag{2.2}$$

E sua função inversa, que exprime as entradas da rede em função das saídas e entradas anteriores, pode ser representada como:

$$\begin{aligned}\mathbf{u}(k+1) &= G[\mathbf{y}(k), \dots, \mathbf{y}(k-n+1); \\ &\quad \mathbf{u}(k-1), \dots, \mathbf{u}(k-n+1)]\end{aligned}\tag{2.3}$$

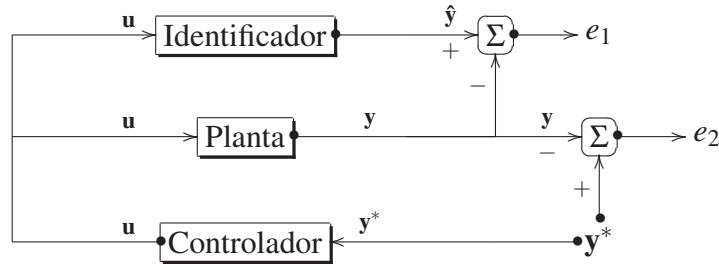


Figura 2.3 Identificação e Controle

2.2.1 Identificação e Controle

Do ponto de vista da teoria de sistemas dinâmicos, as RNAs são consideradas mapas não-lineares parametrizados. Desde o início dos anos 90, quando se apresentou provas conclusivas (Narendra, 1996) de que as redes de retropropagação eram capazes de aproximar quaisquer funções contínuas, que seu uso se espalha por diversos campos de aplicação, entre eles a identificação e o controle de sistemas não-lineares dinâmicos.

- *Identificação:*

Quando $\mathbf{f}$ e $\mathbf{h}$ são desconhecidos, é necessário primeiro fazer a *identificação* do sistema. Um identificador (ou *neuroidentificador*) é um modelo capaz de aprender a prever o comportamento de um sistema dinâmico através da análise das entradas $\mathbf{u}(k)$ e dos estados $\mathbf{x}(k)$ deste (Werbos, 1990).

O problema de identificação pode ser formalizado da seguinte forma (Narendra & Parthasarathy, 1990):

Dadas a entrada $\mathbf{u}(k)$ e a saída $\mathbf{y}(k)$ de um sistema dinâmico que assume-se estável com algum conjunto de parâmetros cujos valores são desconhecidos, o objetivo é construir um modelo de identificação que ao receber a mesma entrada $\mathbf{u}(k)$ produza a saída $\hat{\mathbf{y}}(k) \approx \mathbf{y}(k)$. Desta forma, o problema de neuroidentificação consiste em encontrar uma RNA $\bar{F}[\cdot]$ que aproxime $F[\cdot]$.

Na Figura (2.3), e_1 é o erro do identificador, dado pela diferença entre a saída real da planta e a saída aprendida pelo modelo. O processo de treinamento de uma rede neuroidentificadora consiste em minimizar este erro. Isto pode ser feito através de uma realimentação do erro, usando algoritmos de retropropagação, ou através do mapeamento auto-organizado dos pares $(\mathbf{u}, \mathbf{y})$ para futura utilização.

- *Controle:*

O identificador descreve o sistema dinâmico. Uma vez de posse de um modelo identificador, ou tendo à disposição o sistema dinâmico original, é o papel do controlador *agir* no sistema.

Quando $\mathbf{f}$ e $\mathbf{h}$ são conhecidos (ou estimado por $\tilde{F}[\cdot]$), o problema de controle consiste em desenvolver um controlador que gere as entradas $\mathbf{u}(t)$ tais que o sistema mantenha-se em equilíbrio dinâmico (Narendra & Parthasarathy, 1990). Assim, o problema de neurocontrole consiste em encontrar uma RNA $\tilde{G}[\cdot]$ que aproxime $G[\cdot]$.

A abordagem para o problema de planejamento e controle de sistemas dinâmicos na qual se utiliza uma RNA $\tilde{F}[\cdot]$ para identificar o sistema e outra $\tilde{G}[\cdot]$ para o controlar, como visto na seção anterior, é chamado de *Métodos de Controle Neural Adaptativo*. Na Figura (2.3), e_2 é o erro do controlador, calculado pela diferença entre a saída real da planta $\mathbf{y}$ e a saída desejada $\mathbf{y}^*$. O processo de treinamento de uma rede neurocontroladora consiste em minimizar o erro e_2 . Redes retroalimentadas minimizam o erro pelo ajuste de pesos baseados no cálculo de gradientes. No *Método de Controle Inverso Direto*, em especial usando redes auto-organizáveis, uma única rede é capaz de fazer a identificação e o controle do sistema (Barreto et al., 2003b). Mapas auto-organizáveis são treinados como neurocontroladores criando nodos representativos dos estados do sistema, ajustando possíveis desvios ou ruídos. Os mapas auto-organizáveis são redes ideais para responder na ausência de padrões completos, e uma vez mapeados os pares $(\mathbf{u}, \mathbf{y})$, elas são capazes de atuar como neuroidentificadores ou neurocontroladores, a depender de qual parcela do padrão está incompleta e é a ela apresentado para ser completado.

2.2.2 Espaço de Estados

Os estados de um sistema contém a configuração (ou situações) daquele sistema em um dado momento (Bresciani Filho & D'Ottaviano, 2000). Esta configuração determina unicamente o estado do sistema. Se a configuração do sistema atual depende de sua configuração anterior o estado é considerado dinâmico, caso contrário, estático (Barreto & Araújo, 2004). Qualquer que seja o tipo de estado, para que um espaço de estados seja bem caracterizado, facilitando com isso o uso de técnicas para sua solução, é importante que toda a informação disponível esteja codificada no estado (Russel & Norvig, 2002).

Em sistemas contínuos o estado pode ser discretizado por observações feitas em intervalos de tempo regulares. Um sistema discretizado apresenta um erro de *quantização* que é minimizado quando o número de estados na descrição do sistema cresce (Kohonen, 1987).

Um *Espaço de Estados* de um sistema, também chamado de espaço de trabalho, é o conjunto $\mathcal{M}$ de estados que caracterizam este sistema. Cada ponto $\mathbf{q} \in \mathcal{M}$ é uma configuração

única e não-redundante do espaço de trabalho.

Chama-se de *Obstáculos* o conjunto $\mathcal{O}$ de estados inválidos do sistema, que podem ser estáticos (i.e., os estados inválidos não mudam com o tempo) ou dinâmicos (i.e., os estados inválidos e válidos variam e se intercambiam). Os obstáculos definem áreas alcançáveis pelo sistema, porém temporária ou permanentemente inválidas. Áreas em que o sistema não é definido não são consideradas obstáculos, mas simplesmente como sendo fora do espaço de trabalho (Glasius et al., 1995). A área do sistema disponível para atuação é dada pela diferença do espaço de estados pelos obstáculos $\mathcal{M}_{\text{free}} = \mathcal{M} - \mathcal{O}$.

Para o planejamento de trajetórias no espaço de estados de um sistema é necessário o mapeamento deste sistema. Nas mais diversas áreas da ciência pode-se perceber a importância de se mapear estados (ou configurações de sistemas) e suas transições, com um método de aprendizado automático, com o objetivo de se conseguir extrair deste sistema respostas factíveis e/ou ótimas.

O mapeamento de um espaço de estados através de exemplos por redes neurais não-supervisionadas, isto é, nos quais é a própria massa de dados que dirige o aprendizado, pode ser atribuído pioneiramente a Kohonen (1987).

2.2.3 Trajetórias de Estados

A definição de trajetórias de estados é mais abrangente do que o nome sugere, uma vez que o estado atual de um dispositivo ou sistema pode representar um conjunto qualquer das possibilidades de configurações deste sistema, não sendo necessariamente um sistema físico que se moverá em um ambiente. A trajetória então passa a ser o conjunto de transições deste sistema de sua configuração inicial para uma outra configuração pré-definida, assumindo no intervalo desta tarefa pontos distintos intermediários no espaço de estados possíveis do sistema.

Da seção anterior, tem-se que trajetórias plausíveis entre uma configuração inicial $\mathbf{q}_0$ e uma configuração final $\mathbf{q}_f$ são mapas de um intervalo real fechado em uma configuração $\mathcal{M}_{\text{free}}$ do sistema (Glasius et al., 1995). Este problema se torna mais complexo se se conhece apenas uma parcela do espaço de estados, que é gradualmente apresentada, e o conjunto de transições é dinâmico, ou seja, é criado a partir de propriedades dos estados, como sua proximidade. Neste caso uma análise do sistema para a criação de agrupamentos (*cluster*) usa a medida subjetiva de similaridade dos estados para agrupá-los (Baraldi & Alpaydin, 2002). A qualidade do mapeamento e do conjunto de transições criado influenciará diretamente a geração de trajetórias no espaço aprendido pelo neurocontrolador.

Uma trajetória $\mathcal{T}$ é definida como uma seqüência finita de estados:

$$\mathcal{T} = \{\mathbf{q}_0, \dots, \mathbf{q}_i, \dots, \mathbf{q}_f\} \quad (2.4)$$

onde $\mathbf{q} \in \mathbb{R}^D$ é um estado do sistema, que pode ser configurado como sendo um vetor de dimensão D .

2.2.4 Abordagem Proposta

Redes Neurais Artificiais (RNAs) empregadas na solução de problemas de trajetórias de estados realizam tipicamente quatro tarefas: reconhecimento, reprodução, associação e geração (Araújo & Barreto, 2002). Uma RNA pode rotular parte ou o todo de uma dada seqüência que lhe for apresentada, realizando reconhecimento. Na reprodução, uma RNA pode produzir o restante ou toda uma seqüência quando parte desta lhe é mostrada. Se a seqüência produzida for diferente daquela mostrada, então a RNA realiza associação. Por último, uma RNA pode gerar toda uma trajetória de estados ou seu próximo padrão se alguma informação sobre a trajetória desejada for provida. A reprodução de trajetórias de estados difere da geração pois esta última produz caminhos não aprendidas anteriormente.

Este trabalho visa propor e implementar um modelo de RNA auto-organizável para geração de trajetórias de estados aprendidos de um sistema dinâmico, com base em pontos fornecidos e critérios de otimalidade para criar conexões entre estes estados. Devido aos múltiplos critérios, o modelo deve ser capaz de gerar trajetórias por estados ou por sinais de controle. Este planejamento e controle será testado para diferentes domínios, nos quais deve ser tarefa da própria RNA adaptar-se e interpretar as especificidades de cada domínio. A geração de trajetórias pode ser ponto-a-ponto ou determinada para passar por algum(ns) ponto(s) específico(s), chamada de trajetória *via-pontos*.

A abordagem utilizada proposta é a dos *Métodos de Controle Inverso Direto*, com malha aberta. A principal característica deste método é a suposição de independência entre estados sucessivos (Barreto et al., 2002a), o que permite que a *aleatoriedade* dos estados sejam mapeadas em redes auto-organizáveis estáticas (Walter & Schulten, 1993; Martinetz et al., 1990; Gaudiano & Grossberg, 1991).

O centro do problema a ser abordado pode ser brevemente colocado como: dado um conjunto muito grande de estados (com comportamentos mensuráveis próximos, ruidosos, de natureza e escalas múltiplas¹) e um conjunto de critérios para a medição de proximidade entre

¹Estados podem conter informações heterogêneas, de natureza e escalas múltiplas, i.e., variáveis com distintos intervalos de atuação, qualidade de informação e importância para a identificação do sistema.

estados, deseja-se construir um mapa topológico dinâmico representando todos estes estados com precisão adequada, e que estabeleça nas conexões entre seus nodos uma maneira plausível para a transição suave entre eles. As trajetórias serão calculadas dinamicamente, dependendo do critério de vizinhança.

Para atingir o objetivo de criar trajetórias baseadas em diferentes critérios de otimização, o mapa topológico criado deve ser capaz de fornecer informações compartimentadas, que possam ser utilizadas para cálculos de proximidades distintos segundo propriedades específicas de cada domínio em questão.

No caso de RNAs não-supervisionadas, ou seja, a classe de redes auto-organizáveis de Kohonen (SOM), o aprendizado do mapeamento não-linear se dá pela exposição da rede a exemplos de estados do espaço de entrada e seu respectivo comando de controle, formando um padrão de entrada ξ a ser treinado (Kohonen, 1987; Barreto et al., 2003b). O próprio mapeamento criado, uma função ψ tal que $\psi : \mathcal{M} \rightarrow \mathcal{A}$, $(\xi \in \mathcal{M}) \mapsto (i \in \mathcal{A})$, é entendido como o mapeamento não-linear de um espaço de dimensão $\mathcal{M} \in \mathbb{R}^{D_{\mathcal{M}}}$ em um mapa de saída de dimensão $\mathcal{A} \in \mathbb{R}^{D_{\mathcal{A}}}$.

Deste modo, um vetor de entrada $\xi = [\xi_1, \dots, \xi_n]^T$, $\xi \in \mathcal{M}$, tendo configurada uma rede cujo vetor de pesos é $\mathbf{w} = [w_1, \dots, w_n]^T$, $\mathbf{w} \in \mathcal{M}$, é mapeado em um nodo $s_1 \in \mathcal{A}$, normalmente escolhido pela menor distância Euclidiana:

$$\|\mathbf{w}_{s_1} - \xi\| = \min_{i \in \mathcal{A}} \|\mathbf{w}_i - \xi\| \quad (2.5)$$

Para resolver, por exemplo, o problema de controle robótico, a RNA deve mapear entradas provindas de sensores e gerar saídas que controlam o robô. Uma RNA auto-organizável pode ser entendida como uma tabela dinâmica, com as colunas de entrada e saída, de modo que seus nodos (neurônios) seriam linhas dessa tabela, ou seja, cada nodo contém em si tanto os valores de entrada como os seus respectivos valores de saída. Mas as semelhanças param aqui, uma vez que esta suposta tabela com suas linhas separadas no espaço, e conectadas não por uma ordem de seqüência, mas por critérios de semelhança entre cada espaço de estados. Cada nodo tem a capacidade de representar não somente o ponto que lhe é semelhante, que por si só já inclui informações do estado (posição) e de como atingí-lo (sinal de controle), mas também a estrutura topológica em torno deste ponto, através de conexões com outros nodos semelhantes. Além disso, uma das principais diferenças entre uma rede SOM e uma tabela é que as informações que compõe a rede SOM são auto-programáveis, i.e., as entradas da tabela são encontradas por aprendizado não-supervisionado e evoluem para uma representação estatística do espaço de estados (Barreto et al., 2003b).

Alguns exemplos de RNA ilustram a facilidade de se utilizar este método para o aprendizado de associações entre mapas. Em robótica, para citar um exemplo, é possível criar o mapeamento entre coordenadas cartesianas e os respectivos ângulos das juntas de um robô (Barreto et al., 2003b), como realizado pela rede proposta por Coiton et al. (1991). Cada nodo da rede armazena informações das coordenadas cartesianas (x, y, z) e dos correspondentes ângulos das juntas ($\theta_1, \theta_2, \theta_3$). A rede é treinada com uma função de tipo Chapéu Mexicano, com o decrescimento da vizinhança, de modo que os nodos imediatamente vizinhos do nodo vencedor são ajustados de modo excitatório, enquanto os nodos mais distantes são inibidos durante seu ajuste. Este simples modelo resolve o problema da cinemática inversa sem conhecimento *a priori* das características do braço e seus sensores.

Este trabalho considerará em especial o domínio das trajetórias robóticas (entre eles um manipulador robótico bi e tridimensional, e uma mão robótica), tendo como espaço de estados as configurações possíveis de todas as suas juntas. É interessante notar pelo exemplo que o sistema deve se resguardar do problema de gerar pontos intermediários inválidos, não alcançáveis, perigosos, ou que devam ser evitados por outros motivos quaisquer. Também entre os exemplos de domínios levantados encontra-se o de análise de progressões harmônicas musicais, em que se discute o conceito de reducionismo harmônico a propriedades acústicas, incluído neste trabalho para mostrar a generalidade do modelo proposto.

Para o mapeamento e controle destes sistemas, a utilização de mapas auto-organizáveis de topologia dinâmica, e por isso variável, com a intenção de aproveitar-se desta dinâmica da rede para melhor aprender as relações topológicas do sistema, é um método inovador.

Dentre a classe de RNAs auto-organizáveis consideradas importante para a resolução deste tipo de problema, em especial os modelos com topologia dinâmica, segue-se uma breve revisão dos modelos mais importantes, alguns dos quais serão melhor aprofundados no capítulo seguinte.

- *Self-Organizing Map* (SOM) (Kohonen, 1987), um dos primeiros modelos, com estrutura topológica rígida, cuja principal característica foi apresentar a idéia de auto-organização, e os problemas que surgiram foram sendo minimizados pelos modelos sucessores. *Cascade-correlation Learning Architecture* (CCLA) (Fahlman & Lebiere, 1990), apresenta uma topologia crescente, como solução para o problema de convergência do algoritmo *back-propagation*.
- *Growing Cells Structures* (GCS) (Fritzke, 1994a), baseado no SOM, permite o crescimento da estrutura topológica em formato de um grafo *k-simplex*, levando em conta o

tamanho da *Região de Voronoi*² de cada nodo. *Probabilistic Growing Cells Structures* (PGCS) (Vlassis et al., 1997), assume uma função de densidade de probabilidade (fdp) gaussiana dos dados de entrada para tirar proveito de propriedades dessa fdp e com isso criar os agrupamentos (representados pelos nodos) com a mesma quantidade de pontos, trocando as Regiões de Voronoi do GCS (hipertetraedros) por hiperesferas, e estimando o seu posicionamento e raio.

- Os modelos *Growing SOM* (GSOM), *Growing Grid* (GG), *Incremental Growing Grid* (IGG), (Fritzke, 1995) são modelos incrementais voltados para a preservação de topologia, visualização de dados e organização da informação, baseados no SOM, em que uma rede retangular de nodos cresce adicionando-se colunas ou linhas inteiras de nodos, permitindo-a adaptar-se à distribuição de dados com mais precisão que o SOM original. Os modelos *TreeCGS*, *Growing Chain* (GC), *Growing Hierarchical Self-organizing Maps* (GH-SOM) são variantes hierárquicas, que crescem criando ramos em estruturas de árvores, usadas para organização de informação (Rauber et al., 2002b; Freeman & Yin, 2005).
- *Neural Gas* (NG) (Martinetz et al., 1993), (Fritzke, 1997a), é um método de quantização vetorial que não cria conexões entre os nodos, mas apenas os distribui no espaço dos dados de entrada, seguindo uma taxa de movimentação dos nodos que varia inversamente à distância do nodo ao ponto de treinamento apresentado. *Competitive Hebbian Learning* (CHL) (Martinetz et al., 1993), (Fritzke, 1997a), não move os nodos no espaço, mas é utilizado apenas para criar conexões entre cada par de nodos mais próximos de um dado ponto de treinamento apresentado. *Topology Representing Networks* (TRN) (Martinetz et al., 1991), é a combinação do NG com o CHL, de modo que os nodos primeiro se espalham pelo espaço dos dados de entrada, e em uma segunda fase cria-se as conexões entre eles. Uma das desvantagens é não ser incremental.
- *Growing Neural Gas* (GNG) (Fritzke, 1997a), é derivado do GCS, flexibilizando a sua estrutura topológica. O modelo GNG apresenta um comportamento semelhante ao TRN, com a vantagem de ser incremental. *Growing Neural Gas with Utility* (GNGU) (Fritzke, 1997b), ataca o problema do crescimento indiscriminado da rede GNG inserindo um termo como critério de *utilidade* de cada nodo, que permite eliminá-lo caso este critério atinja seu limiar estabelecido. O *Locally Weighted Interpolating Growing Neural Gas*

²A região circunvizinha do nodo, para a qual este responde ativando-se, sendo limitada nas divisas entre outras regiões de nodos vizinhos, e ilimitada caso o nodo não possua outro vizinho que o limite (por exemplo, nas bordas do mapa).

(LWINGNG), baseado no GNG, é um método de aproximação e interpolação de funções, de espaços de alta dimensão para baixa dimensão (Flentge, 2006).

- *Instantaneous Topological Map* (ITM) (Jockusch & Ritter, 1999), é inspirado no SOM e no GNG, o modelo ITM tem o objetivo de criar mapeamentos topológicos assumindo que os padrões de entrada são correlacionados. O modelo mostra-se interessante para a criação de trajetórias de estados, uma vez que aprende rapidamente seqüências de estímulos fortemente correlacionadas.
- *Adaptive Resonance Theory* (ART) (Grossberg, 1976), é um sistema que cria uma ressonância entre uma camada de reconhecimento de padrões não-supervisionada e uma camada de orientação supervisionada (que por sua vez modela uma avaliação externa do casamento de padrões possível), de modo que a camada não-supervisionada possa classificar padrões com uma certa precisão dada, ρ , que quando alcançada define a convergência do algoritmo. Caso a precisão não seja alcançada em um primeiro momento, o ART1 cria novas categorias para enquadrá-lo. *Fuzzy ART Map* (FAM) (Carpenter et al., 1992) é composto de dois módulos Fuzzy ART (que, grosso modo, é uma generalização do ART1 para tratar de padrões reais). Estes dois módulos formam uma rede de três camadas, sendo o primeiro módulo responsável por particionar os padrões de entrada em hiper-retângulos que são passados para o segundo módulo para serem conectados à sua classificação correta. Consegue com isto classificar corretamente mesmo regiões desconexas. *Fully Self-Organizing Simplified Adaptive Resonance Theory* (FOSART) (Baraldi & Alpaydin, 2002), consegue evitar a sensibilidade à ordem de apresentação dos padrões que ocorre no FAM, já que o objetivo dos modelos ART e derivados é o agrupamento (e por isso não deve sofrer com processos indutivos). FOSART é um algoritmo para agrupamento de aprendizado *on-line*, com característica de preservação de topologia, que aplica um critério de mínima distância para a média para classificar os dados de entrada.
- *Gamma-observable Neighbours* (γ -ON) (Aupetit et al., 2002), utiliza o parâmetro γ (um escalar no intervalo $[0, 1]$), que é a distância normalizada de um dado de entrada qualquer e todos os nodos da rede, utilizado para definir a vizinhança do padrão entrado. Quando $\gamma = 0$, somente o nodo coincidente com o padrão está no seu campo de observação, e quando $\gamma = 1$ todos os nodos da rede são observáveis a partir do ponto do padrão. Com essa técnica para o cálculo da vizinhança, se consegue uma convergência mais rápida que a dos modelos que utilizam Regiões de Voronoi.
- *Grow when Required Network* (GWR) (Marsland et al., 2002), derivada do GNG, utiliza

CHL para criar e destruir nodos (sendo a destruição baseada em um fator de *idade* da conexão, que envelhece quando não utilizada). Sua vantagem é que pode criar nodos sempre que necessário a cada passo do algoritmo (o que difere do GNG, que só cria nodo a cada λ passos). Além disso, o critério para criação de nodos envolve a atividade do nodo vencedor, para evitar que se crie nodos desnecessários quando o nodo vencedor poderia ser treinado para responder à respectiva entrada.

- Uma outra classe de RNA, baseada em redes auto-organizáveis é a classe dos modelos derivados do *Vector Integration to Endpoint* (VITE) (Bullock & Grossberg, 1988b,a). Uma característica marcante do modelo VITE é o seu empenho em manter a plausibilidade biológica, baseando-se em pesquisas sobre o funcionamento do córtex primata. Este modelo traz consigo a hipótese de que as trajetórias de estados para movimentos de um manipulador visualmente guiado são processos que emergem da contínua integração entre diferentes áreas do cérebro, pela percepção da contínua diferença entre a posição atual do braço e a posição alvo desejada.

O modelo VITE foi inspiração para diversos outros modelos, como o DIRECT (Bullock et al., 1998; Fiala, 1994; Izquierdo et al., 1996), que aprende de modo autônomo o mapeamento entre o espaço das coordenadas angulares das juntas para o espaço da tarefa, de modo a resolver o problema da cinemática inversa. Além disso, este modelo não necessita do conhecimento da geometria do manipulador, o que lhe permite adaptar-se à mudanças neste, como o crescimento, por exemplo. Fiala (1994) propôs uma variação do modelo DIRECT para incluir movimentos com uma curva de velocidade em forma de sino, semelhante à desenvolvida por seres-humanos (Bullock et al., 1998), e inclui uma fase de aprendizado chamada de *Motor Babbling* (MB), ou 'balbuciamiento motor'. O modelo *Adaptive VITE* (AVITE) (Gaudiano & Grossberg, 1991, 1992), alguns anos antes, também propôs acoplar ao modelo VITE esta fase de aprendizado de MB. Mais detalhes do MB serão vistos na Seção 3.1.2, no momento basta saber que MB são movimentos aleatórios, endogenicamente gerados, que segundo Piaget (1963) permitem explicar como uma criança aprende a controlar o seu braço com crescente precisão ao praticar tais movimentos. Com esses movimentos aleatórios, consegue-se aprender o mapeamento entre a posição do braço nas coordenadas do sistema de visão com a sua posição nas coordenadas dos sensores angulares das juntas. Isso é feito utilizando-se o conceito de *reação circular* apresentado por Piaget, em que o sistema que normalmente determina o movimento do braço passa antes por uma fase inversa em que o próprio braço calibra o sistema. O modelo *extended VITE* (EVITE) (Bullock et al., 1998, 1999) acres-

centa ao modelo VITE outras áreas do córtex importantes quando se consideram sinais aferentes que tratam das forças inerciais, forças de carga, e cálculo da posição percebida, permitindo que este modelo consiga exercer força contra obstáculos ou compensar pesos externos com suavidade.

É sabido que as RNAs podem ser entendidas como uma classe de modelos computacionais capazes de aprender mapeamentos não-lineares, que unida com a capacidade de se auto-organizar e auto-adaptar a diferentes conjuntos de dados, tornam esses modelos adequados para o trabalho em questão. Em geral, as RNAs auto-organizáveis incrementais são utilizadas em duas grandes áreas: classificação de padrões e visualização de dados, apesar de serem utilizadas também em uma gama diversa de áreas desde otimização combinatorial até aproximação de funções (Fritzke, 1996).

Entre as propriedades comuns aos modelos incrementais pode-se citar:

- A estrutura da rede consiste em um número de nodos (também chamados de unidades) e um conjunto de conexões entre estes nodos.
- Cada nodo i tem associado a ele uma *posição* w_i (também chamado de vetor referência ou vetor de pesos) no espaço de entrada.
- A adaptação dos vetores de pesos é feita pela geração de um sinal de entrada ξ , a seleção de um vetor de peso mais próximo (chamado de *vencedor*) e o consecutivo movimento deste vencedor em direção ao sinal de entrada. Além disso, é comum mover-se também os vetores de peso dos nodos que estão conectados diretamente ao vencedor (chamados de nodos *vizinhos*), porém em menor intensidade.
- A cada passo desta adaptação do mapa topológico, se calcula um *erro acumulado*.
- Em determinado momento deste processo de adaptação, seguindo um critério a ser definido pelo modelo, cria-se um novo nodo no mapa topológico. O erro acumulado pode ser usado para determinar a posição deste novo nodo, por exemplo.

As diferenças entre os diversos modelos incrementais normalmente são notadas nas restrições impostas à topologia da rede. Tais diferenças serão melhor detalhadas no Capítulo 3.

Um modelo de RNA deve ser capaz de aprender o mapeamento não-linear entre a entrada e saída e gerar uma topologia capaz de ser utilizada para dar respostas com precisão adequada para o domínio em questão. Num exemplo de domínio, o planejamento e controle de manipuladores robóticos, uma RNA deve ser capaz de suprir a necessidade de cálculos de cinemática

inversa e de dinâmica. Quando considera-se que o manipulador pode apresentar variações em sua geometria, como por exemplo crescimento e fadiga, os modelos que necessitam da informação explícita desta geometria ficam comprometidos, pois os mapeamentos aprendidos podem variar no tempo. Um sistema auto-organizável que colha estes dados em tempo de execução e se adapte às informações colhidas pode contornar este problema. As redes SOM reconfiguráveis podem adaptar-se a estas mudanças (Bullock & Grossberg, 1988a; Barreto et al., 2003b).

As redes SOM são mais simples, geralmente mais rápidas, e portanto são uma alternativa viável às redes MLP e RBF para aproximação de funções e para o desenvolvimento de neuro-controladores (Barreto et al., 2003b). Em comparação com técnicas tradicionais de controle, computacionalmente caras ao realizar cálculos extensos além de necessitar de precisão da descrição dos parâmetros e de serem sensíveis a ruídos, a utilização de RNAs se torna ainda mais atrativa.

Além disto, o aprendizado em RNAs não-supervisionadas tem como característica não necessitar de um “professor” externo, o que confere ainda maior autonomia ao sistema (Araújo & Barreto, 2002). Este trabalho se torna relevante como exemplo de aplicação principalmente no domínio robótico, devido ao fato de se ter pouca literatura que trata da utilização de RNA não-supervisionada (Barreto & Araújo, 2001; Bullock & Grossberg, 1988a; Fiala, 1994) para a modelagem e o controle robótico. Entretanto, o controle robótico é apenas um exemplo de domínio para geração de trajetória de estados, de modo que este trabalho possa ser avaliado com exemplos específicos. Tendo em vista o caráter abrangente das trajetórias de espaço de estados, este trabalho contribui para uma tentativa de unificação de propriedades das redes neurais artificiais auto-organizadas e incrementais em busca de uma solução simples e única para diversos domínios.

Será visto no Capítulo 4 como se pode aproveitar esse esparsa conhecimento, em especial considerando três dos modelos mais importantes, a saber, o GNG, o GWR e o EVITE, que por si só já reúnem qualidades importantes a serem implementadas. Na seção seguinte será apresentado o problema de geração de trajetórias para alguns domínios específicos usados como exemplos neste trabalho.

2.3 Domínios Exemplos para Geração de Trajetórias de Estados

Nesta seção serão apresentados quatro diferentes domínios para os quais a geração de trajetórias de estados pode ser empregada, e que serão utilizados como exemplos para os testes feitos com o modelo proposto neste trabalho. Na Subseção 2.3.1 apresenta-se o domínio dos manipulados

res robóticos, em especial um manipulador ideal de 2 juntas, bidimensional, e um manipulador industrial de 6 juntas, o PUMA-560. Ainda na área de robótica, a Subseção 2.3.2 apresenta algumas dificuldades extras ao tratar do planejamento e controle de uma mão antropomórfica. A Subseção 2.3.3 apresenta um problema sobre a similaridade entre acordes musicais e a teoria da progressão harmônica. No problema da progressão harmônica, não há necessidade de controle, apenas o planejamento de trajetórias para avaliar a sequência dos acordes.

2.3.1 Planejamento e Controle de Trajetórias de Manipuladores Robóticos

Considerando o estudo de caso da geração de trajetórias de estados de manipuladores robóticos, a complexidade de informações que uma RNA tem que processar pode ser averiguada quando se leva em conta as propriedades que um manipulador deve apresentar para simular as propriedades existentes no movimento de um braço humano. Entre estas propriedades, cita-se a autonomia, que permite ao robô alterar a sua trajetória baseando-se em informações provenientes de fontes de sinais como juntas, visão, sensor de toque, força, peso de carga, inércia, e então reformular a trajetória, ou mesmo cancelá-la. A autonomia engloba a capacidade de auto-geração da trajetória (planejamento) e das ações necessárias para executá-la, bem como a reação e o replanejamento decorrente de situações inesperadas. A autonomia possibilita lidar com ambientes inóspitos, dinâmicos, que demandam integração rápida entre percepção, tomada de decisão e execução.

O planejamento e controle de trajetórias robóticas demandam cálculos complexos serem realizados, aumentando sobremaneira esta complexidade quando se substitui um ambiente estático por outro em mudanças dinâmicas. Além disto, é comum haver dificuldade para se conseguir bases de dados para robôs industriais em ambientes de trabalho, de modo a suprir o treinamento de redes supervisionadas.

Não bastasse as dificuldades supracitadas, sistemas cuja percepção é oriunda de sinais de sensores relacionados com visão, juntas, torque, força, precisam atuar em tempo hábil, exigindo respostas rápidas. Um outro fator importante é a não-linearidade do mapeamento necessário entre a entrada (percepção) e a saída (ação). Além de não-linear, este problema pode ser mal posto quando deseja-se tratar da cinemática e dinâmica inversas.

O problema de planejamento e controle robótico simplificadaamente consiste em determinar como o braço se move de um ponto a outro. Dividindo-se este problema geral em três fases distintas (Barreto et al., 2003b), têm-se: a fase de planejamento da tarefa, a fase de planejamento da trajetória e a fase de controle do braço.

A fase de planejamento da tarefa consiste em perceber as posições atuais e finais do braço,

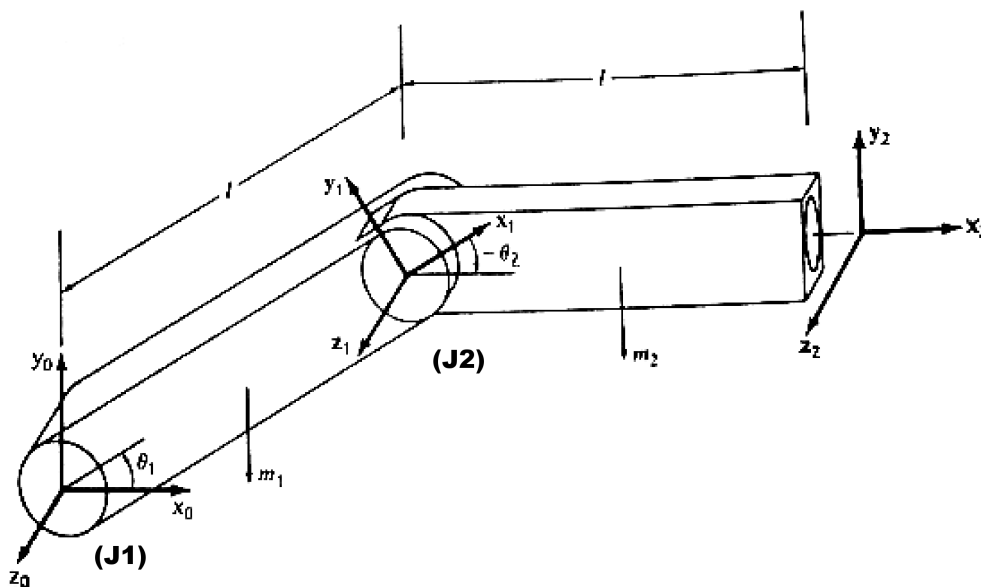


Figura 2.4 Esquema de um manipulador composto de duas juntas J_1 e J_2 , com coordenadas na origem dos sistemas $\overline{OX_0Y_0Z_0}$ e $\overline{OX_1Y_1Z_1}$, e com liberdade de rotação no eixo Z por ângulos θ_1 e θ_2 , respectivamente; e de dois membros m_1 e m_2 de comprimento $l = 1$. O efetuador é localizado na origem do sistema $\overline{OX_2Y_2Z_2}$. É tarefa do controle robótico realizar as rotações e translações entre estes sistemas de coordenadas para o sistema de coordenadas base (Fu et al., 1987, p. 99).

além de, se for o caso, considerar também pontos intermediários que deverão ser seguidos, e/ou pontos que deverão ser evitados (como os pontos que estariam impedidos de alcance imediato devido a existência de obstáculos). Pode-se também nesta fase incluir a velocidade que se passará por tais pontos, a aceleração, ou quaisquer outras informações necessárias para ajustar o dispositivo para a tarefa a ser realizada.

A fase de planejamento da trajetória consiste em achar os próximos pontos para os quais o braço deverá se mover, considerando a história do movimento até então realizado, os custos do movimento, aceleração de cada junta, torque, velocidade, solavanco (*jerk*) e presença de obstáculos, entre outros que podem vir a ser implementados, de modo a criar um movimento harmônico e coordenado, além de suave, em direção ao alvo, e satisfazendo as restrições da fase anterior.

E finalmente, a fase de controle consiste em achar os torques necessários, considerando-os como sinal de atuação, para realizar o movimento planejado, satisfazendo as restrições dadas pelas fases anteriores. Esta fase, quando se utiliza RNA, pode ser integrada à fase anterior, tornando-se indistinguível.

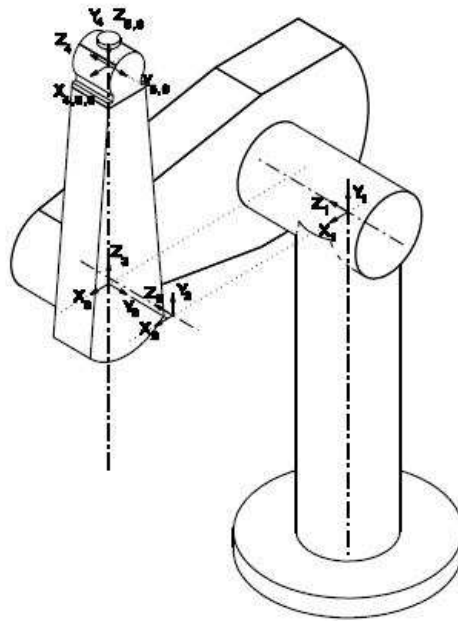


Figura 2.5 Esquema do Robô PUMA-560, com 6 juntas rotativas.

2.3.1.1 Cinemática Direta e Inversa

O controle motor normalmente é realizado pela tradução de um mapa de coordenadas de entradas cartesianas, a posição do efetuador em um braço mecânico, em um mapa de variáveis de saída, tais como os ângulos das juntas ou os torques. Para realizar este tipo de controle é necessário o cálculo da cinemática e da dinâmica de um robô (Fu et al., 1987).

A cinemática é a ciência que trata do movimento independentemente das forças de atuação. No caso de um manipulador robótico, a cinemática direta se refere ao cálculo da posição e da orientação do efetuador, dadas as coordenadas angulares de cada junta.

No planejamento e controle robótico entretanto, é a cinemática inversa que normalmente é requerida. A cinemática inversa trata do cálculo do conjunto de ângulos das juntas de um robô, dada uma posição do efetuador no mapa de coordenadas cartesianas. O problema de gerar trajetórias usualmente recai no cálculo da cinemática inversa, pois, em geral, tem-se o ponto final desejado e procura-se por uma seqüência de ângulos que levem o braço para o alvo.

Em robôs com graus de liberdade redundantes há diversas soluções de posicionamento do braço para uma dada coordenada do efetuador. Mais especificamente, a redundância ocorre se o número de equações dadas pelos elementos da *Matriz de Transformação Homogênea* (também conhecida como *Matriz do Braço* ou simplesmente matriz T) do manipulador for maior que

o número de variáveis desconhecidas (ângulos das juntas). A Matriz do Braço é uma matriz 4×4 proposta Euler e utilizadas por Denavit e Hartenberg (Fu et al., 1987) para descrever as translações e rotações entre os membros do braço de um robô e assim sistematicamente estabelecer um sistema de coordenadas de referência base em que todos os movimentos são descritos. O robô industrial PUMA-560 (Figura 2.5) com seis graus de liberdade, tem em sua matriz T um conjunto de 12 equações em função dos *senos* e *cosenos* dos ângulos $\theta_1, \dots, \theta_6$, ou seja, 6 variáveis desconhecidas, caracterizando um sistema de múltiplas soluções. Considere o robô bidimensional da Figura (2.6), com duas juntas e braços de tamanho unitário. Sua matriz T pode ser descrita por 6 equações em função de 2 variáveis desconhecidas θ_1, θ_2 :

$${}^0A_1 = \begin{bmatrix} C_1 & -S_1 & 0 & lC_1 \\ S_1 & C_1 & 0 & lS_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & lC_2 \\ S_2 & C_2 & 0 & lS_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

$$T = {}^0A_2 = {}^0A_1 {}^1A_2 = \begin{bmatrix} C_{12} & -S_{12} & 0 & l(C_{12} + C_1) \\ S_{12} & C_{12} & 0 & l(S_{12} + S_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.7)$$

onde C_i é o $\cos(\theta_i)$; S_i é o $\sin(\theta_i)$; C_{ij} é o $\cos(\theta_i + \theta_j)$; S_{ij} é o $\sin(\theta_i + \theta_j)$.

A Equação (2.6) descreve as matrizes de rotação que transformam as coordenadas de um dado membro para a coordenada do membro anterior. Assim, por exemplo, a matriz ${}^1A_2$ transforma as coordenadas da posição do efetuador do sistema $\overline{OX_2Y_2Z_2}$ para o sistema $\overline{OX_1Y_1Z_1}$. A Equação (2.7) é chamada de *Matriz do Braço*, e descreve as equações necessárias para transformar as coordenadas da posição do efetuador diretamente para as coordenadas do sistema base $\overline{OX_0Y_0Z_0}$.

Em coordenadas cartesianas, a posição espacial (x_1, y_1) do cotovelo (J2), e a posição espacial (x_2, y_2) do efetuador (J3), dados os ângulos das juntas, sendo a junta (J1) fixa na origem $(x_0 = 0, y_0 = 0)$, e considerando o tamanho dos membros $l_1 = l_2 = l$, descreve-se a cinemática

direta de um robô bidimensional por:

$${}^0x_1 = lC_1 = l_1 \cos(\theta_1) \quad (2.8)$$

$${}^0y_1 = lS_1 = l_1 \sin(\theta_1) \quad (2.9)$$

$${}^0x_2 = l(C_{12} + C_1) = l_2 \cos(\theta_1 + \theta_2) + l_1 \cos(\theta_1) \quad (2.10)$$

$${}^0y_2 = l(S_{12} + S_1) = l_2 \sin(\theta_1 + \theta_2) + l_1 \sin(\theta_1) \quad (2.11)$$

$${}^1x_2 = lC_2 = l_2 \cos(\theta_2) \quad (2.12)$$

$${}^1y_2 = lS_2 = l_2 \sin(\theta_2) \quad (2.13)$$

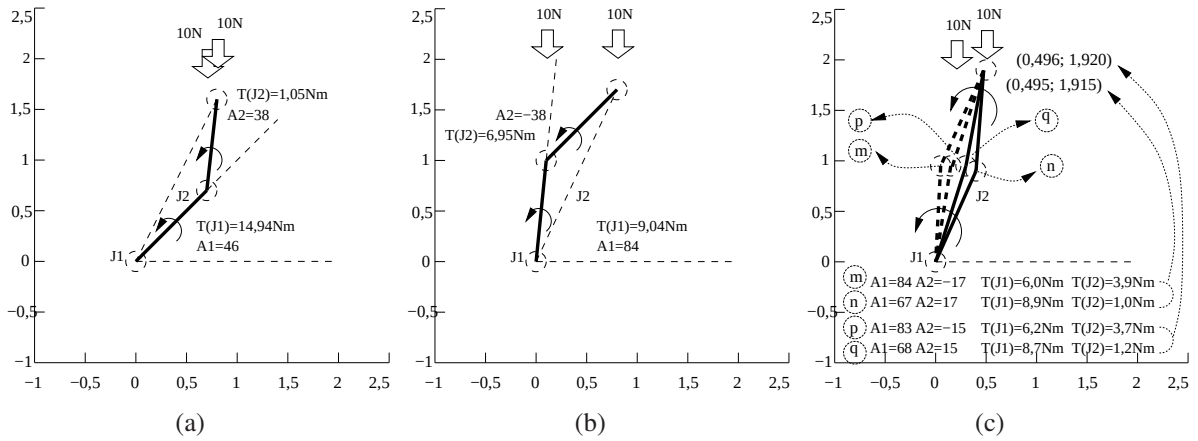


Figura 2.6 Problema da cinemática inversa: ilustração de como diversas configurações de ângulos podem levar ao mesmo ponto final no efetuador ou em pontos suficientemente próximos, considerando uma dada precisão. (a) Efetuador posicionado em $(0, 8; 1, 71)$, com ângulos das juntas $A_1 = 46^\circ$ e $A_2 = 38^\circ$; (b) Efetuador posicionado em $(0, 8; 1, 71)$, com ângulos das juntas $A_1 = 84^\circ$ e $A_2 = -38^\circ$; (c) Quatro configurações distintas m, n, p, q que posicionam o efetuador em $(x, y) = (0, 4955 \pm 0, 15\%; 1, 9175 \pm 0, 15\%)$: (m) $(x, y) = (0, 495; 1, 915)$ e $(A_1, A_2) = (84, -17)$, (n) $(x, y) = (0, 495; 1, 915)$ e $(A_1, A_2) = (67, 17)$, (p) $(x, y) = (0, 496; 1, 920)$ e $(A_1, A_2) = (83, -15)$ e (q) $(x, y) = (0, 496; 1, 920)$ e $(A_1, A_2) = (68, 15)$.

Na Figura (2.6) (a), tem-se o efetuador posicionado em $(x_2, y_2) = (0, 8; 1, 71)$, utilizando como ângulos das juntas os valores $A_1 = 46^\circ$ e $A_2 = 38^\circ$, e em (b) o efetuador está colocado na mesma posição cartesiana, porém as juntas estão postas em $A_1 = 84^\circ$ e $A_2 = -38^\circ$. Em apenas um plano bidimensional e com um robô com duas juntas tem-se caracterizada a redundância de soluções, por ângulos que produzem posições simétricas.

A Figura (2.6) (c) ilustra que mesmo no plano, com um robô de apenas duas juntas, pode-se conseguir devido à precisão do sistema, mais que duas configurações semelhantes com posições similares. Por exemplo, considere posicionar o robô no ponto $(x_2, y_2) = (0, 4955; 1, 9175)$, com

margem de erro³ de $\pm 0,15\%$. A figura em questão apresenta quatro posicionamentos, dois a dois simétricos, em que todos distam (erro) do ponto desejado em exatos $(0,0005; 0,0025)$. A posição (m) é simétrica à posição (n), e ambas posicionam o robô no ponto $(0,495; 1,915)$, em que os braços ficam levemente mais abertos, e o erro faz o efetuador ficar levemente abaixo da posição correta. As duas posições simétricas internas (p) e (q) colocam o efetuador levemente acima da posição correta, em $(0,496; 1,920)$, também com erro $(0,0005; 0,0025)$.

2.3.1.2 Dinâmica Direta e Inversa

A dinâmica envolve o estudo das forças e torques necessários para que um objeto físico possa equilibrar-se em alguma posição desejada, acelerar ou mover-se e parar em alguma outra posição.

A dinâmica direta calcula posições no espaço, dados o torque ou a força. Este problema surge na robótica quando a leitura de sensores acoplados e distribuídos no robô é utilizada para se estimar a posição no espaço de seu efetuador. O mais comum, entretanto, é o cálculo da dinâmica inversa. Esta, por sua vez, calcula o torque, dadas a posição, as velocidades e as acelerações de cada junta. Para se calcular a dinâmica inversa, faz-se necessário levar em consideração efeitos de inércia, forças centrífugas, atrito e viscosidade das juntas e a gravidade, tornando este um problema complexo de ser resolvido (Barreto et al., 2003b; Fu et al., 1987; Bullock & Grossberg, 1988a).

Enquanto o manipulador robótico percorre uma trajetória no espaço, os torques envolvidos são calculados analiticamente por complexas equações recursivas como: Lagrange-Euler, Newton-Euler ou Hamiltoniano, equações de movimento dinâmico. Estas equações podem levar em consideração efeitos de inércia, forças centrífugas e de Coriolis, fricção e viscosidade das juntas, gravidade, carga, ação e reação, descritas no vetor de coordenadas generalizadas e na lagrangiana (Marion, 1971, p.203). Calcular estas equações dinamicamente é tarefa significativamente mais complexa se comparada com a complexidade do cálculo da cinemática inversa.

A equação geral de movimento de um manipulador, expressa na formulação de Lagrange-Euler é dada por (Fu et al., 1987, p.92):

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = \tau_i \quad i = 1, 2, \dots, n \quad (2.14)$$

onde:

³O valor arbitrário escolhido de $\pm 0,15\%$ apenas exemplifica o problema, sem perda de generalidade para qualquer precisão dada.

L é a função lagrangiana: $L = K_{in} - Pot$;

K_{in} é o total de energia cinética do braço do robô;

Pot é o total de energia potencial do braço do robô;

q_i é o vetor de coordenadas generalizadas do robô;

$\dot{q}_i$ é a primeira derivada do vetor de coordenadas generalizadas do robô;

τ_i é a força ou torque aplicada ao sistema na junta i para mover os membros $i, i + 1, \dots$

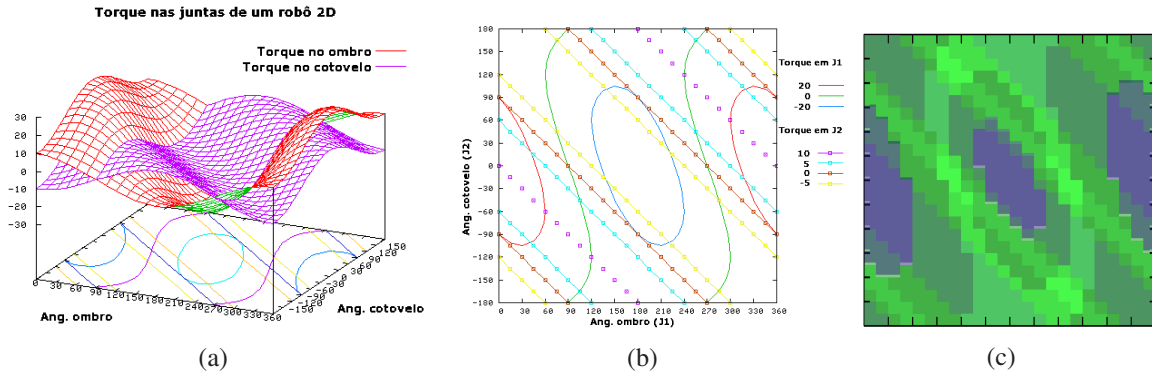


Figura 2.7 (a) Superfície de variação do torque de um robô bidimensional, com variação para as juntas equivalentes ao ombro (J1) entre 0° e 360° e ao cotovelo (J2) entre -180° e 180° . (b) Curvas de nível em 2D, da superfície de variação do torque. (c) Curvas de nível: tons mais claros estão mais próximos de zero.

A Figura (2.7) ilustra os torques a que estão submetidas as duas juntas $J1$ e $J2$ do robô bidimensional representado na Figura (2.4), quando em cada uma delas se aplica uma força de $10N$. Na superfície tridimensional da Figura (2.7) (a) pode-se observar como a variação do torque de uma junta qualquer depende dos ângulos de ambas as juntas, e portanto, da configuração do robô. Estes torques representados nesta superfície consideram o robô estático na posição, ou seja, com velocidade e aceleração iguais a zero. Na Figura (2.7) (b) tem-se as curvas de níveis da superfície 3D representada em (a). Pode-se visualizar claramente nesta figura as áreas em que o robô estaria submetido a menos esforço, conforme mostra a Figura (2.7) (c), na qual as tonalidades mais claras representam ambas as juntas com torques mais próximos de zero.

Os métodos tradicionais de cálculo de torques possuem limitações como a precisão do cálculo, a velocidade de sua realização, ou a dependência do conhecimento da geometria do braço de modo pré-determinado. Entre estes métodos convencionais, destacam-se (Barreto et al., 2003b) os controles *proportional-integral-derivative* (PID), que são completamente dirigidos por erros e assumem que os parâmetros do robô são fixos; e o *controle de torque computado*, que consegue criar trajetórias para as várias configurações de um manipulador, porém requer uma velocidade de cálculo elevada, além de muita precisão nos parâmetros do modelo dinâmico

do robô.

Para contornar estas limitações, em especial a dificuldade de estimar corretamente os parâmetros do manipulador, e de obter cálculos da dinâmica inversa de modo rápido e preciso, a utilização de RNA auto-organizáveis se mostra adequada.

2.3.2 Planejamento e Controle de Trajetórias de Mão Antropomórfica

Uma mão antropomórfica, como um objeto robótico, apresenta as mesmas propriedades vistas na seção anterior: possui um conjunto de juntas que podem ser descritas por posição espacial, ângulos ou torques necessários para mantê-las em sustentação ou movimento. É a potencialidade para a execução de tarefas que confere à mão robótica um problema de planejamento mais complexo.

Esta potencialidade é evidenciada pela capacidade que tais robôs possuem de manusear objetos, sendo este o principal problema de planejamento hoje na literatura de mãos robóticas (Borst et al., 2003b; Hoffmann et al., 2005, 2007). A pegada (*grasp*) pode ser caracterizada de dois principais modos: A pegada de precisão, em que o objeto é sustentado pela ponta dos dedos, permitindo o seu manuseio com bastante liberdade. E a pegada de força, em que a mão deve exercer ou sofrer grande força, e para isto toda a superfície da mão é utilizada no contato com o objeto.

Ainda há que se considerar o tipo de dedos. Normalmente os modelos de mãos robóticas podem ser divididos em: dedos rígidos e sem fricção (ideais), dedos rígidos com fricção e dedos macios. Para que um objeto mantenha-se preso na ponta dos dedos, é preciso que a soma das forças atuantes no objeto, provindas dos dedos e de distúrbios externos esteja em equilíbrio:

$$\sum_{i=1}^N F_i + F_{\text{ext}} = 0$$

A posição que os dedos encostam em um objeto para realizar a pegada pode ser classificada em: (a) Fechada-forte. Esta pegada é capaz de aplicar forças de atrito em qualquer direção do objeto para compensar qualquer vetor de força que venha a causar distúrbios no sistema. (b) Fechada-suave. É uma pegada do tipo fechada-forte, sem nenhum atrito aplicado nos pontos de contato (Borst et al., 2003b). Entretanto, para desenvolver planos mais elaborados de pegada, é necessário, antes de mais nada, como no caso de manipuladores vistos na seção anterior, algum meio para a resolução do problema de cinemática inversa e dinâmica inversa.

Diversos modelos de mãos robóticas têm sido desenvolvidos nos últimos anos. O modelo *UB Hand 3* (Lotti et al., 2005) utiliza elásticos nas juntas para reduzir a complexidade mecânica

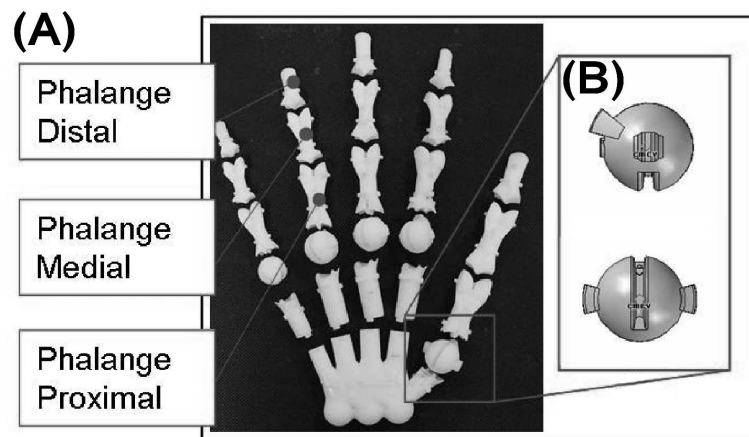


Figura 2.8 Mão Antropomórfica Kanguera: (a) Mão completa. (b) Detalhe da junta. Figura extraída de (Benante et al., 2007b).

da mão e manter um nível de destreza aceitável. Um diferente tipo de junta é proposto por Palli & Melchiorri (2006), composta por um tendão utilizado para fechar o dedo, e uma mola para que o dedo possa voltar à posição original. Para simular modelos para planejamento e controle de mãos robóticas, a mão antropomórfica brasileira BRAHMA (Caurin et al., 2005) e a sua sucessora, a mão Kanguera, protótipos inspirados biologicamente em mãos humanas, compõem um ambiente flexível aberto para a implementação de diferentes controladores e planejadores. Neste modelo, visto na Figura (2.8) (a), as juntas dos dedos (Figura (2.8) (b)) são feitas com cabos de transmissão, e o material da mão é um polímero biocompatível. O modelo possui 22 graus de liberdade (DOF), sendo 4 em cada um dos dedos e 2 adicionais no pulso.

Caurin et al. (2004) propõem uma solução analítica para o problema da cinemática inversa, e dividem as possíveis posições dos dedos em uma série discreta de seqüências de estados estáticos que representam o movimento que se quer realizar. Apesar de ser uma solução elegante, está restrita a realizar o movimento pré-programado. Estes mesmos estados ou configuração de estados distintos podem ser utilizadas para o treinamento de uma rede não-supervisionada capaz de aprender a topologia do espaço de trabalho da mão e gerar trajetórias planejadas dentro dos estados aprendidos.

2.3.3 Modelagem de Progressões Harmônicas

Como um outro exemplo de domínio para o uso de mapeamento e geração de trajetórias num espaço de estados, são considerados os estudos de progressões harmônicas. Na história da música, a produção realizada entre o século XVIII e o final do século XIX é conhecida como

Música Tonal (Lovelock, 1987). No século XX, apesar da música de vanguarda ter seguido outras direções (Griffiths, 1978), a música popular e a música de entretenimento ainda seguem aqueles preceitos. Pode-se chamar de música tonal toda forma de música construída sobre as escalas maiores e menores, que apresentam uma estrutura hierárquica específica. A música tonal, assim como outras formas de música, possui três elementos principais: ritmo, melodia e harmonia. Os dois últimos elementos, ritmo e harmonia, são os que mais a caracterizam, justamente por seguirem a estrutura hierárquica das escalas maior e menor (tal estrutura hierárquica forma as bases do chamado sistema tonal, um conjunto de regras ou normas de composição musical).

A harmonia tonal emprega quatro tipo de acordes, que são chamados de tríades (conjuntos de 3 notas): a tríade perfeita maior, a tríade perfeita menor, a tríade aumentada e a tríade diminuída (Cooper, 1975). Sobre cada grau da escala maior e menor pode-se construir uma tríade, sendo aquelas elaboradas sobre o primeiro grau (tônica) e sobre o quinto (dominante) as mais importantes. Basicamente, o que a harmonia tonal prescreve é como construir uma sucessão de acordes dentro de uma tonalidade, de acordo com certas convenções (Parncutt, 1989). Tais prescrições podem ser descritas como regras de uma sintaxe (Lerdahl & Jackendoff, 1996).

O primeiro tratado de harmonia tonal foi o de Jean-Philippe Rameau (1971), publicado originalmente em 1722. O principal objetivo de Rameau foi a tentativa de reduzir as convenções da harmonia a princípios naturais. Posteriormente, importantes compositores e teóricos também se aventuraram em direções semelhantes a de Rameau, crendo que as novas descobertas da ciência poderiam contribuir para a explicação dos fenômenos musicais. Schoenberg (1999) foi um destes, publicando um dos mais importantes tratados de harmonia. Nesta obra, ele tentou explicar a natureza das inversões triádicas, por exemplo, pela própria constituição das séries harmônicas das notas dos acordes. O que Schoenberg desejava era mostrar que as inversões de acordes eram de fato uma questão acústica, e não uma escolha convencional e artística. Rameau havia sido o primeiro a postular que um acorde podia ser invertido e manter a suas propriedades essenciais (Rameau, 1971, p. 40). Schoenberg, em seu tratado, afirma que em um caso específico, o da segunda inversão, o acorde perde suas características essenciais, aproximando-se dos acordes uma quinta acima, tentando justificar sua posição por fatos acústicos (Schoenberg, 1999, p. 107ss.).

Oliveira et al. (2005) empregaram uma rede de Kohonen para testar a hipótese de Schoenberg de que as inversões de acordes eram primordialmente um fenômeno natural baseado nas suas constituições acústicas, e o resultado não corroborou a hipótese, levando a crer que tais inversões são mesmo características socioculturais ou artísticas. De fato, a harmonia tonal

foi investigada através de Mapas Auto-organizáveis de Kohonen em alguns estudos (Leman, 1991; Page, 1999; Griffith, 1999). Tais estudos, apresentam categorizações de acordes semelhantes aos encontrados por Oliveira et al. (2005), mas tais categorizações são estáticas, e não levam em conta a transição de acordes. A harmonia não é restrita à categorização tipológica; uma importante parte da teoria da harmonia é seu aspecto morfológico, ou seja, as formas de transição entre um acorde e outro. Sendo assim, os estudos mencionados acima são limitados em suas possibilidades investigativas por não permitirem o estudo da transição entre acordes. Neste caso, a geração de trajetórias de estados pode se aplicar para a criação de progressões harmônicas. Tais trajetórias podem ser utilizadas para se verificar se as progressões de acordes típicas dos repertórios das músicas tonais podem decorrer de classificações de estruturas acústicas. Esta questão, de certa forma, retoma o ideal de autores como Rameau ou Schoenberg, que tentaram reduzir a harmonia tonal a princípios acústicos; se esta postura for correta, uma rede neural alimentada apenas com informações acústicas deve ser capaz de gerar encadeamentos harmônicos coerentes assim como parece gerar classificações tipológicas coerentes de acordes.

2.4 Conclusões

Este capítulo mostrou a importância do problema de planejamento na área de IA, e fez um resumo das três principais abordagens na literatura: Modelos de Programação Procedimental, Modelos de Planejamento e Modelos de Aprendizado. Os Modelos de Aprendizado foram discutidos com maior profundidade, subdividindo-os em cinco categorias: (1) Métodos de Controle Supervisionado, (2) Métodos de Controle Inverso Direto, (3) Métodos de Controle Neural Adaptativo, (4) Métodos Baseados na Retropropagação da Utilidade e (5) Métodos Adaptativos com Críticos. Além dessas divisões, também se abordou os conceitos de planejamento em malha aberta e fechada.

Com a visão geral dada pela seção inicial, passou-se a formalização e descrição do problema de geração de trajetórias de estados, com definições de conceitos importantes tais como: sistemas dinâmicos, identificação, controle, espaço de estados e trajetórias de estados. Em seguida dissertou-se sobre a abordagem utilizada para buscar soluções para este problema, categorizada como Métodos de Controle Inverso Direto, dentre os Modelos de Aprendizado, utilizando um planejador de malha aberta. Ainda nessa seção que trata da abordagem foi apresentada uma revisão da literatura que mostra como este problema foi tratado por diversos autores.

Por fim, o capítulo encerra-se com uma discussão sobre a os diferentes domínios que o modelo será aplicado, como exemplo da generalidade desse e de sua ampla aplicabilidade. Os

domínios discutidos incluem a geração de trajetórias para manipuladores robóticos bi e tridimensionais, para uma mão robótica tridimensional, e para a análise de progressões harmônicas musicais.

No próximo capítulo será visto um conjunto de modelos que podem ser utilizados para o mapeamento de espaço de estados e geração de trajetórias, e suas limitações.

Modelos Anteriores Relevantes

A mente é uma folha de papel em branco.

—LOCKE (Col. Pensadores)

Neste capítulo será visto um conjunto seletivo de modelos cujo funcionamento e propriedades possam ser utilizadas como fundamentação e inspiração para a criação de uma solução para o problema de geração de trajetórias de estados com topologia dinâmica. Na Seção 3.1 se apresenta o modelo Vetor de Integração para o Ponto Final (VITE). Na Seção 3.2 vê-se o modelo de Mapas Auto-organizáveis (SOM). Na Seção 3.3 tem-se o modelo de Redes de Representação de Topologia (TRN). Na Seção 3.4 introduz-se o modelo Crescimento de Estruturas de Células (GCS). Na Seção 3.5 uma visão geral sobre o modelo Gás Neural Crescente (GNG) é dada. Na Seção 3.6 tem-se o modelo Crescer Quando Necessário (GWR).

3.1 Vetor de Integração para o Ponto Final - *Vector Integration to Endpoint (VITE)*

O modelo VITE (Bullock & Grossberg, 1988a; Fiala, 1994; Gaudiano & Grossberg, 1992) é uma descrição aproximada de modelos de dinâmica de neurônios para controle adaptativo de trajetórias de movimentos de um braço durante uma busca guiada visualmente. Sua inspiração está diretamente ligada aos avanços do conhecimento do funcionamento do córtex, e a sua estrutura se propõe explicar diversas propriedades percebidas na atividade cerebral de vários animais, incluindo os seres humanos, e em dados comportamentais (Bullock & Grossberg, 1988a; Gaudiano & Grossberg, 1991).

Há importantes características neurais consideradas e tratadas pelos autores (Bullock & Grossberg, 1988a; Bullock et al., 1998), durante o movimento de um membro em direção a um dado alvo. Primeiramente, o movimento de um braço é regido por juntas angulares e percebido pelo córtex motor primário e parietal por células especializadas, muitas das quais atualmente estão identificadas.

A Figura (3.1) apresenta um esquema do córtex, sugerido pelo modelo *extended VITE*

Figura 3.1 Modelo do circuito cortical sugerido pelo EVITE, onde: GO, sinal escalável voluntário; DVV, vetor de velocidade desejada; OPV, vetor de fluxo de saída de posição; OFPV, fluxo de força e vetor de posição somados; SFV, vetor de força estática; IFV, vetor de força inercial; CBM, entrada cerebelo-cortical assumida para o estágio IFV; PPV, vetor de posição percebida; DV, vetor de diferença; TPV, vetor de posição do alvo; γ^D , motoneurônio dinâmico gama; γ^S , motoneurônio estático gama; α^m , motoneurônio alfa; Ia, fibra aferente tipo Ia; II, fibra aferente tipo II (realimenta erro de posição); c.s., sulco central; i.p.s., sulco intraparietal; +, representa excitação; -, representa inibição; $\times$, representa multiplicação; $\int$, representa integração (Bullock et al., 1998).

Uma característica relevante para o uso de Redes Neurais Artificiais Auto-organizáveis (RNA-AO) para explicar o fenômeno de geração de trajetórias é o surgimento de uma trajetória como processo emergente das relações entre os diversos sistemas (ou conjuntos) celulares distintos, sem o seu pré-planejamento completo. Com este mecanismo, torna-se natural a alteração de uma trajetória por outra, por exemplo devido ao movimento do alvo, ou a escolha de um novo alvo, mesmo após o início da sua execução da trajetória. [Bullock et al. \(1999\)](#) argumentam que muitos modelos foram propostos para explicar a formação de trajetória e o alcance de um alvo, diferenciando-se principalmente por três relevantes dimensões : (i) o grau que o modelo acomoda as muitas propriedades fisiológicas de movimentos de busca; (ii) o grau em que a estrutura proposta é consistente com as restrições neurobiológicas e (iii) o grau em que o aprendizado e o desempenho dependem da informação disponível ao sistema, sem necessitar de conhecimento externo; sendo estes os motivos básicos que levaram os autores a estenderem

o modelo VITE para outras tarefas relativas ao movimento e busca. Mais precisamente, a extensão de Bullock et al. (1999) buscou criar trajetórias que devem passar por um ponto dado antes de alcançar um alvo determinado, uma trajetória *via-ponto*.

Além disto, o movimento de qualquer parte do corpo é determinado por contrações e extensões de pares antagônicos de músculos que devem cooperar de modo sinérgico. A força de estímulo aplicada a um grupo muscular não determina de imediato a sua contração, pois este membro pode estar impossibilitado de mover-se por estar, por exemplo, exercendo pressão sobre um obstáculo em seu caminho. Neste caso, a força do estímulo causa a mudança do ponto de equilíbrio do sistema até que o obstáculo ceda ou até que o ponto de saturação da resposta ao estímulo máximo seja alcançado, aquele que ocorrer primeiro. O organismo, através da propriocepção (oriunda dos *spindles* – mecanismos internos aos músculos), realimenta a posição atual do braço baseado na contração efetivamente atingida após um determinado estímulo.

Quando um grupo de diversos músculos é requisitado para realizar uma determinada tarefa, por exemplo, mover ombro-braço-pulso de uma posição inicial B para uma posição final E, os organismos apresentam um movimento harmonioso, em que a amplitude do movimento de cada grupo é compensada com diferentes velocidades para se conseguir a posição final de todas as juntas simultaneamente. A Figura (3.2) mostra três posições iniciais (B1, B2 e B3), uma posição final (E), e seis possíveis trajetórias, sendo as linhas pontilhadas a representação de trajetórias não sincronizadas, e as linhas contínuas, trajetórias sincronizadas (Bullock & Grossberg, 1988a). Para o caso da trajetória B1 → E, por exemplo, a linha pontilhada ao sair de B1 em direção a E, mantém por uma fase inicial as velocidades de alteração das componentes vertical e horizontal idênticas. Uma vez que a componente vertical alcançou a altura adequada (ponto V) para atingir o ponto E, a sua velocidade é reduzida para zero, permanecendo apenas o movimento horizontal. Esse movimento é análogo ao movimento iniciado em B3, com a diferença de que a componente horizontal alcança a posição desejada primeiro, permanecendo apenas a componente vertical. A única trajetória não sincronizada que coincide com a sincronizada se dá quando todas as componentes têm a mesma amplitude de movimento, caso da posição inicial B2. Em um movimento natural de um braço, partindo por exemplo de B1, há uma compensação na velocidade, para que todas as componentes sempre terminem seu movimento ao mesmo tempo. Apesar de termos uma variedade nas velocidades de cada junta, nota-se nos modelos naturais que o gráfico de velocidade tem sempre a mesma forma (quando normalizado no tempo de movimento), que é a forma de um sino, tendo ainda por princípio um pequeno pico, necessário para vencer a força inercial (identificado na população de células fase-tônicas (IFV) da Área 4 – veja Figura (3.1)).

Conseguir tal sincronia de variação de velocidades, também chamada de *equifinalidade*

temporal (Ulloa & Bullock, 2003; Sanguineti & Morasso, 1992), sem pré-planejar uma trajetória é um desafio que o modelo VITE solucionou. Esta sincronia deriva principalmente do sinal volitivo GO crescente no tempo. É provável que este sinal provenha dos gânglios basais, de acordo com recentes observações (Bullock & Grossberg, 1988b).

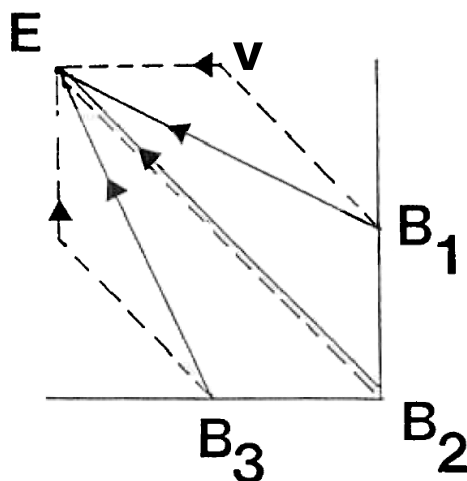


Figura 3.2 Linhas contínuas representam movimento sincronizado, ao contrário do que ocorre nas linhas pontilhadas, não sincronizadas (Bullock & Grossberg, 1988a).

Ainda na Figura (3.1), pode-se interpretar as hipóteses levantadas pelo modelo EVITE sobre as relações entre as diversas áreas do córtex: o vetor de diferenças (DV) entre a posição atual do braço (PPV) e a posição do alvo desejado (TPV), é calculado na Área 5 parietal. O PPV também é calculado na Área 5 parietal, e é originado subtraindo-se do vetor de sinal de posição (OPV) o valor do erro retornado pelos *spindles* musculares Ia provindos da Área 2. O DV, quando liberado por um sinal voluntário de movimento (GO), gera o vetor DVV (*desired velocity vector*), que representa um comando de velocidade sensível à volição, no córtex chamados de células fase. O vetor DVV é em seguida integrado por populações de células tônicas na Área 4, gerando um vetor de sinal de posição (OPV) que controlará diretamente os motoneurônios alfa e gama-estático. A população de células tônicas da Área 4 também envia sinais necessários para se calcular a posição atual do braço (PPV). Há uma conexão recíproca entre as células da Área 5 que formam PPV e as células tônicas (OPV), cuja função é permitir que a Área 4 perceba a movimentação do braço por forças externas. As células fase-tônicas da Área 4 (OPFV - *outflow force-and-position vector*) formam o ponto de concentração dos cálculos que consideram forças externas inerciais e de cargas impostas, recebendo suas entradas do cerebelo e dos centros que integram os *spindles*, possibilitando que se aumente a força exercida pelos músculos caso a atividade motoneural atual não esteja fazendo o movimento progredir na direção desejada.

3.1.1 O Modelo VITE

O modelo VITE pode ser visualizado na Figura (3.3), que apresenta seus componentes principais. No topo da figura, o módulo *Target Position Command* (TPC) representa a posição final do braço desejada. Na base da figura, o módulo *Present Position Command* (PPC) codifica internamente a posição atual do braço. Os sinais para o movimento do braço são gerados pelo PPC. Estes sinais, que são compensados por realimentação e por aprendizado cerebelar, movem a mão para o local codificado por PPC relativamente ao corpo (independentemente do sistema de coordenadas utilizado, mas normalmente se utiliza na literatura um sistema baseado na contração atual dos músculos das juntas), e conseguem lidar com a dinâmica do braço, bem como com condições adversas (como a carga de um objeto, ou a inércia).

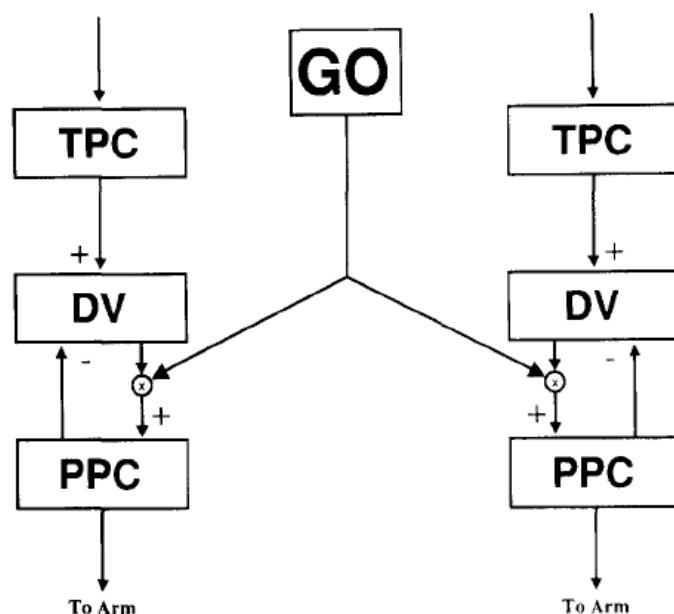


Figura 3.3 Modelo VITE. TPC, *Target Position Command*; DV, *Difference Vector*; PPC, *Present Position Command*; GO, Sinal de ativação que controla a inibição do movimento, ou a sua velocidade. O uso de um único sinal GO permite a sincronização do início do movimento de todos os músculos (Gaudiano & Grossberg, 1991).

O papel do sinal GO é permitir que seja possível, de alguma forma, controlar a velocidade de movimento do braço, através da multiplicação do sinal provindo de DV com um fator multiplicativo. Além disto, o caso limite em que GO é igual a zero permite que se treine posições internamente ao modelo sem alterar a real posição do braço.

Entretanto, o sinal GO valer zero não significa que o braço não possa ter sido movimentado externamente por alguma força desconhecida. Neste caso é necessário considerar outros grupos

de sinais (em especial os sinais de realimentação provindos de Ia) que atualizam PPC através do erro gerado por órgãos *spindles* internos aos músculos, e que corrigem a representação interna de PPC para a real posição do braço no espaço. Portanto, o modelo VITE realiza um processo automático que move PPC continuamente até TPC.

A Figura (3.3) divide o sinal GO para dois subsistemas idênticos, à esquerda e à direita da figura. Essa divisão representa o par de músculos antagonísticos que é utilizado para mover cada junta. Esses músculos são acoplados de modo que quando um grupo (p_i) se contrai, o outro grupo oposto (p_j) se estende. Sem perda de generalização, pode-se dizer que a medida de extensão do músculo antagonístico é exatamente $p_j = 1 - p_i$. Também é possível simplificar o modelo para melhor compreendê-lo, eliminando o movimento do músculo antagonístico, neste caso, sob pena de perder algumas propriedades importantes: (i) O músculo só gera força ao contrair-se, nunca ao estender-se; (ii) a capacidade de enrijecer ambos os músculos ao mesmo tempo permite maior robustez ao se fixar uma determinada posição a ser mantida contra forças externas dinâmicas e desconhecidas.

Na sua forma mais simples (Bullock & Grossberg, 1988b), o modelo VITE obedece as equações que descrevem as variáveis DV e PPC (e com elas, TPC e GO):

- *Vetor de Diferenças (DV)*:

$$\frac{d}{dt}V_i = \alpha(-V_i + T_i - P_i) \quad (3.1)$$

- *Comando de Posição Presente (PPC)*:

$$\frac{d}{dt}P_i = G([V_i]^+) \quad (3.2)$$

onde $[V_i]^+ = \max(V_i, 0)$. As equações (3.1) e (3.2) descrevem as interações de um componente genérico para comandos da posição alvo (TPC) de $(T_1, T_2, \dots, T_n)$, o vetor de diferenças (DV) de $(V_1, V_2, \dots, V_n)$, o comando de posição presente (PPC) de $(P_1, P_2, \dots, P_n)$ e o comando de velocidade (GO), ou sinal $G(t)$.

O vetor de diferenças (DV), continuamente atualiza a discrepância entre a posição atual (PPC) e a posição desejada (TPC). Os sinais oriundos de DV são integrados à PPC até que este se torne igual a TPC, quando DV se torna igual a zero e a movimentação de PPC termina.

O modelo EVITE (Bullock et al., 1998) contém o modelo VITE, e o complexifica, tendo sido concebido com a intenção de incorporar ao seu predecessor algumas novas descobertas sobre o funcionamento do córtex primata, em especial a sensibilidade à cinemática e à cinética, com compensação de cargas e de forças inerciais durante o movimento.

Tanto no modelo VITE como no EVITE, muitas das conexões feitas entre os grupos de células reconhecidos pela fisiologia são hipóteses levantadas que tratam de explicar como estes grupos interagem entre si, preenchendo uma lacuna na literatura desta área.

O circuito gerador de trajetórias VITE permite sincronizar movimentos entre qualquer quantidade estabelecida de juntas, envolvendo musculatura complexa com músculos mono-articulares ou bi-articulares. O modelo EVITE acrescenta ao modelo VITE o controle de velocidades e forças, permitindo analisar como os grupos de células respectivos reagem nas áreas identificadas do córtex.

No modelo EVITE, é incluído o uso de sinais *spindles* provindos dos músculos como a principal fonte de informação para a percepção da posição atual do braço, o que é condizente com dados neurofisiológicos. Entretanto é sabido que indivíduos podem usar outras áreas do cérebro para determinar a posição dos membros, entre elas áreas superiores como a visão, ou o tato, por exemplo.

O modelo VITE é um gerador de trajetórias completo, mas um problema que apresenta é transformar o vetor de diferenças DV em um vetor de direção e velocidade DVV que realmente permite o membro de mover-se em direção ao alvo. No modelo EVITE, o grupo celular OPV intermedia a operação entre DV e PPV, integrando informações sobre inércia e carga.

O sinal GO, que determina a real vontade de mover o membro, é central no entendimento do modelo. Se o sinal GO cresce suavemente do zero até um valor que indica a necessidade de mover todo o conjunto de membros-juntas em direção ao alvo, então o sistema gera uma curva de velocidade em forma de sino, como vista nos movimentos dos primatas (Izquierdo et al., 1996).

$$\begin{aligned}\frac{dg^{(1)}}{dt} &= \varepsilon \left(-g^{(1)} + (C - g^{(1)})g^{(0)} \right) \\ \frac{dg^{(2)}}{dt} &= \varepsilon \left(-g^{(2)} + (C - g^{(2)})g^{(1)} \right) \\ g &= g^{(0)} \frac{g^{(2)}}{C}\end{aligned}\tag{3.3}$$

onde: ε é uma taxa de integração; g é o sinal GO que multiplica DV; $g^{(0)}$ é uma entrada provinda do centro de decisão pré-frontal; $g^{(1)}$ é uma entrada calculada em cascata a partir de

$g^{(0)}$ (análogo para $g^{(2)}$, a partir de $g^{(1)}$); C é o valor no qual as células de GO saturam (veja [Bullock et al. \(1998\)](#) para mais detalhes).

Uma deficiência do modelo VITE é que a característica de equifinalidade temporal não pode ser garantida a menos que o sinal GO seja pelo menos parcialmente reinicializado, para os casos de mudanças repentinas próximas ao final da trajetória. [Schaal & Sternad \(1998\)](#) propuseram um sistema similar para geração de padrões, que não necessita desta reinicialização.

O sistema VITE apresentado consegue gerar uma trajetória e fazer com que o braço atinja a posição alvo definida, porém ele não consegue manter ou recuperar a posição quando uma carga externa ou inercial modifica a posição final do braço após o mesmo ter atingido o alvo e o sinal GO ter caído para zero. Para solucionar parcialmente este problema, para o caso de movimentos involuntários passivos, a Equação (3.4) substitui a Equação (3.2), incluindo um termo que reflete o movimento passivo ([Bullock & Grossberg, 1988b](#)):

- *Comando de Posição Presente (PPC) com sinal de realimentação para movimento involuntário:*

$$\frac{d}{dt}P_i = G([V_i]^+) \cdot G_p([M_i]^+) \quad (3.4)$$

O problema de quando um movimento externo involuntário é imposto ao braço durante o movimento voluntário, no caso do modelo EVITE ([Bullock et al., 1998](#)) é abordado pela adição dos grupos de células tônicas da área 4 (OPV), ligando-as com a posição real do braço (PPV) por realimentação – veja Figura (3.1). O sistema EVITE reage adequadamente a movimentos impostos externamente, seja durante a performance da trajetória, ou após atingir o alvo na tentativa de manter a posição estável. Ainda, o sistema é capaz de aumentar a força de atuação nos músculos para vencer obstáculos que porventura estiverem no trajeto do braço. Entretanto, ainda ocorrerão pequenas diferenças na inicialização do movimento, devido à inércia própria do braço. O sistema iniciará o trajeto com dificuldade de vencer a inércia da posição de repouso. Para solucionar dificuldades de iniciar um movimento, encontrada no modelo VITE, devido à forças inerciais, o EVITE, incorporou as atividades dos motoneurônios alfa e um vetor que indique a força inercial (IFV), associadas às células fase reativas-ao-tempo (fase-RT). O pulso de realimentação está disponível nos canais aferentes Ia e II. A população de células fase-tônicas (OFPV) localizadas entre os motoneurônios alfa e as células tônicas da Área 4 (OPV) recebem o sinal compensatório oriundo de IFV que facilita romper a inércia no início do movimento, e repassa esse sinal para os motoneurônios.

3.1.2 Balbuciamiento Motor (*Motor Babbling*)

Diversos modelos (Gaudiano & Grossberg, 1991; Martinetz et al., 1990, 1991; Morasso & Sanguineti, 1995; Sanguineti & Morasso, 1992) se inspiraram no conceito Piagetiano de *reação circular* (Piaget, 1963) para o aprendizado de transformações no espaço de trabalho, em especial no espaço de trabalho motor de manipuladores robóticos.

Segundo a teoria do psicólogo suíço Jean Piaget (1963), o termo *reação circular* refere-se ao ciclo eferência-reaferência que permite os humanos, em particular as crianças em estágio prematuro de desenvolvimento, aprender as transformações sensório-motoras através da exploração ativa do ambiente (Sanguineti & Morasso, 1992). Assim, uma criança em fase de aprendizado produz movimentos de seu braço, espontaneamente, ao acaso, e os seus olhos percebem a posição em que o braço se posicionou, relacionando-a espacialmente à força exercida, o que permite o aprendizado de uma transformação entre a representação visual da posição do braço e a representação motora desta mesma posição. O mesmo aprendizado pode se dar entre outros tipos de sinais, como por exemplo o tato e a propriocepção, ou no processo de aprendizado do pronunciamento de fonemas, em que estão envolvidos o sistema de produção de sons, daí o termo *balbuciar*, e o sinal realimentado através do sistema auditivo, gerando pontos de aprendizado no espaço de transformação entre ambos os sistemas. O treinamento repetido eventualmente permite a criança a realizar com precisão os objetivos que ela almeja.

A reação circular é um ciclo autonomamente controlado, entre um sistema com os componentes de produção e percepção. Pelo fato destes componentes serem também sistemas, tal aprendizado é chamado de “intermodal”. O aprendizado “intramodal” está relacionado com o ajuste de cada um dos sistemas envolvidos, de modo independente, para que possam realizar suas tarefas coerentemente, de modo preciso, evitando assim inconsistências entre duas instâncias de um mesmo problema, o que acarretaria uma divergência no aprendizado intermodal. É mais comum o uso de redes auto-organizáveis no ajuste de sistemas intramodais, isto é, evoluir e estabilizar os parâmetros de um sistema. O balbuciamiento motor e a teoria de reação circular apresentam uma nova maneira de se utilizar redes auto-organizáveis para o aprendizado intermodal.

O modelo INFANT (Kuperstein, 1991), por exemplo, é um sistema que aprende as informações visuais vindas de um par de câmeras de TV (informação sensorial) e relaciona-as aos comandos motores que estão atuando no momento da imagem (informação motora). O treinamento é feito em uma fase inicial, chamada de fase de balbuciamiento motor (*motor babbling phase*). O modelo é baseado em uma MLP de duas camadas que treina enquanto um gerador pseudo-aleatório faz movimentos no manipulador.

Outros modelos (Martinetz et al., 1990, 1991, 1993; Morasso & Sanguineti, 1995; Sanguineti & Morasso, 1992; Walter & Schulten, 1993) de aprendizado pelo *Método de Controle Inverso Direto* utilizam redes auto-organizáveis ao invés de redes MLP. Naquelas redes, o sistema não-linear de transformação sensório-motor é aproximado por um *mapa* de nodos que as representa (Morasso & Sanguineti, 1995).

O modelo AVITE (*Adaptive VITE*) busca reproduzir como representações corretas de TPC podem ser aprendidas durante uma fase de balbuciamiento motor (MB, *motor babbling*). O MB define comandos aleatórios de movimentos de PPC, endogenicamente gerados por um novo módulo chamado ERG (*Endogenous Random Generator*), que movem o braço para posições diversas em uma ampla faixa de teste, e após marcar estas posições em TPC, realiza-se um ajuste para que DV tenda a zero, utilizando DV como um sinal de erro (Gaudiano & Grossberg, 1992).

O modelo VITE assume que as coordenadas de TPC e PPC sejam dimensionalmente consistentes e estejam calibrados, entretanto essa não é uma hipótese defensável do ponto de vista neurológico, uma vez que essas transformações são nitidamente aprendidas na fase infantil de uma criança, quando esta aprimora os seus movimentos.

Seguindo as sugestões feitas por Piaget (1963) em seu artigo, foi adicionada uma reação circular ao modelo AVITE, e uma nova fase de MB gera diversas posições do braço, que são repassadas para PPC, e de PPC são copiadas para TPC, fazendo com que o vetor de diferenças obrigatoriamente devesse estar em zero para que reflita a correta calibragem entre PPC e TPC, uma vez que estes últimos são iguais. Note o fluxo inverso de informações, de PPC para TPC, quando na fase normal de funcionamento é TPC que é passado, através de uma contínua integração, para PPC. Uma vez que PPC e TPC são iguais, o ajuste a ser feito nas conexões de DV deverá zerar este vetor. Com este modelo, a fase de MB experimenta o espaço disponível para movimentação e gera pares de treinamento (TPC, PPC) para aprender as transformações de coordenadas do sistema AVITE.

Este resumo apresentou o modelo VITE e algumas de suas variações. Por seu compromisso com características fisiológicas, e mapeamentos de áreas do córtex primata, o modelo se torna complexo, inclui diversos grupos celulares que tentam explicar cada região cerebral e suas iterações. Não basta, deste ponto de vista, simplesmente gerar a trajetória ótima, ou com menor torque, ou que represente algum tipo de critério econômico. O que se pretende é antes de tudo explicar como grupos de neurônios em diferentes áreas do cérebro podem agir em sinergia para conseguir o seu objetivo.

Uma característica importante encontrada neste modelo é utilização de uma fase de balbuciamiento motor para o aprendizado das transformações não-lineares do sistema. Transpor

esta e outras características aqui levantadas, e manter qualidades como a equifinalidade temporal, a geração de trajetórias de estados com diversos critérios (como suavidade da curva, por exemplo) a depender do domínio em questão, para redes de topologia dinâmica, pode talvez não condizer tão especificamente com explicações sobre áreas cerebrais, entretanto, a otimização permitida por este tipo de rede pode melhorar a performance do modelo, desenvolvendo sistemas mais econômicos em termos de nodos e conexões, além de simplificar o conjunto de parâmetros necessários para a sua utilização.

3.2 Mapas Auto-organizáveis - *Self-organizing Maps (SOM)*

A característica dos sistemas sensoriais humanos de apresentar receptores organizados espacialmente, tais como na retina ou na pele, que preservam a topografia destas organizações quando mapeados no sistema nervoso central ao se ligarem a estes via nervos óticos ou sensoriais (Kandel et al., 1995), inspirou a criação dos mapas auto-organizáveis, ou mapas de preservação de topologia. O SOM (Kohonen, 1987) pode ser considerado um dos modelos pioneiros no campo das redes neurais artificiais (RNA) com característica de treinamento não-supervisionado visando agrupamento topológico dos sinais de entrada. Apesar do SOM ser normalmente tido como um dos primeiros trabalhos nesta área, os modelos de mapas topográficos foram originalmente propostos por Willshaw & von der Malsburg (1976) e Grossberg (1976), para a explicação de formações de mapas topológicos no cérebro.

Tais redes são normalmente utilizadas para agrupar massas de dados, deixando que a própria análise dos dados gere as categorias relevantes. Outras aplicações desta rede incluem o descobrimento de regularidades na massa de dados, o levantamento de correlações, a procura por características comuns ou estatisticamente relevantes, problemas de reconhecimento de padrões, análise de textos e/ou imagens, análise de dados financeiros, processos de monitoramento e controle.

Kohonen (1987) aponta que uma das propriedades comuns ao cérebro que foram ignoradas pelas *máquinas de aprendizado* é a *ordem* que surge nas unidades de processamento (neurônios). Essa ordem é necessária para a correta representação topológica dos dados de entrada, e é obtida por um algoritmo relativamente simples.

“Although a part of such ordering in the brain were determined genetically, it will be intriguing to learn that an almost optimal spatial order, in relation to signal statistics, can completely be determined in simple self-organizing process under the control of received information¹.”

—KOHONEN (1987)

Kohonen ainda levanta o tema da *plausibilidade biológica*, com as propriedades de:

- Representação e processamento analógico da informação (que permite qualquer grau de assincronismo dos processos em computação massivamente paralela sem problemas de “interlock”);
- Habilidade de aproximar funções condicionalmente sobre um conjunto de dados (operações estatísticas);
- Tolerância à falhas, degradação suave e recuperação de erros;
- Adaptação a um ambiente em mudança (dinâmico) e a emergência de funções de processamento de informação inteligentes, por auto-organização, em resposta aos dados.

Além de duas suposições básicas biologicamente plausíveis do modelo de von der Malsburg: (1) Uma unidade ativada deve também ativar suas unidades vizinhas fisicamente próximas. (2) O reforço entre as conexões deve ser feito de acordo com regras semelhantes às regras de aprendizado Hebbiano.

Pode-se usar uma rede com estas características quando não se tem informações disponíveis para a saída desejada, caso em que passa a ser tarefa da rede descobrir tal saída.

Para que a rede tenha sucesso, é necessário que os dados de entrada possuam redundância de informação entre as suas características que estão de alguma forma representadas. Na Figura (3.4) (a) vê-se um esquema de SOM e suas respectivas interligações entre os nodos, também chamados de células, neurônios ou unidades de processamento. A Figura (3.4) (b) enfatiza o vetor de pesos que será treinada de modo a representar o espaço dos dados de entrada.

¹ Apesar de que parte desta ordem (dos neurônios) no cérebro serem determinadas geneticamente, será intrigante aprender que uma ordem espacial quase ótima, em relação às estatísticas do sinal, podem ser completamente determinadas em um simples processo auto-organizável sob o controle das informações recebidas.

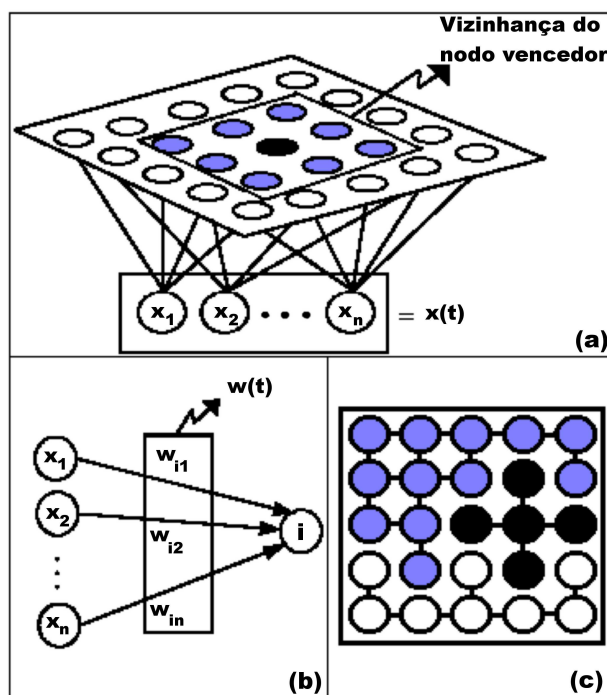


Figura 3.4 SOM: (a) Arquitetura de uma rede 2D. (b) Vetor de pesos. (c) Exemplo de grupos formados após a convergência da rede.

Na sua forma original mais comum, o SOM é um mapa bidimensional e retangular (Kohonen, 1987, p. 119). Esta geometria foi alterada de várias formas nos anos seguintes, com a intenção de diminuir o efeito causado pelo fato de que os nodos das bordas possuem menos vizinhos para compartilhar suas informações. Apesar de algumas melhorias neste quesito, isto definitivamente não é o maior problema destas redes.

3.2.1 Algoritmo

O algoritmo do SOM pode ser assim apresentado:

1. Inicialize a rede com pesos aleatórios, próximos de zero, tais que $\mathbf{w}_i$ é o vetor de pesos das conexões entre a entrada e o nodo i .
2. Inicialize a função de vizinhança, $\mathcal{N}(i)$, para a maior vizinhança.
3. Apresente um padrão $\boldsymbol{\xi} = \begin{bmatrix} \xi_1 & \xi_2 & \dots & \xi_D \end{bmatrix}^T \in \mathbb{R}^D$, onde ξ_j é a entrada j no nodo i .
4. Calcule a distância Euclidiana $\text{Dist}(\boldsymbol{\xi}, \mathbf{w}_i)$ entre a entrada $\boldsymbol{\xi}$ e o vetor de pesos de cada

um dos nodos, dada pela equação:

$$\text{Dist}(\boldsymbol{\xi}, \mathbf{w}_i) = \|\mathbf{w}_i - \boldsymbol{\xi}\|, \quad \forall i = 1, \dots, n \quad (3.5)$$

5. Chame de s_1 o nodo que possui a menor distância $\text{Dist}(\boldsymbol{\xi}, \mathbf{w}_{s_1})$, i.é., o nodo vencedor.
6. Atualize o vetor de pesos para cada nodo i pertencente à vizinhança $\mathcal{N}(s_1)$, segundo a equação:

$$\mathbf{w}_i = \mathbf{w}_i + \varepsilon \cdot (\boldsymbol{\xi} - \mathbf{w}_i) \quad (3.6)$$

7. Diminua a vizinhança de $\mathcal{N}(s_1)$ segundo algum critério.
8. Repita a partir do Passo 2 até que se atinja um critério de parada.

Cabe lembrar que a função vizinhança $\mathcal{N}(s_1)$ contém os nodos vizinhos do nodo vencedor, que também terão seus pesos atualizados (com menor proporção) na direção do padrão atual de entrada. A função vizinhança é a responsável pelo ordenamento dos nodos que tendem a responder para um mesmo padrão de entrada, ou para padrões de entrada similares. É necessário que esta função tenda a zero para assegurar a convergência do método.

3.2.2 Exemplo

No exemplo da Figura (3.5) (a), um mapa auto-organizável de Kohonen, com estrutura inicial bidimensional, retangular, acha uma configuração que distorce sua grade inicial para uma nova disposição que minimiza o erro de resposta de cada unidade.

Pode-se notar nestes exemplos que, devido às características da massa de dados (descontínua e não-convexa), o mapa de Kohonen não consegue distribuir adequadamente suas unidades tais que fiquem dispostas somente nos pontos de probabilidade² não-nula. A Figura 3.5 (a) mostra o problema que ocorre quando existem áreas descontínuas na massa de dados.

Mesmo em áreas contínuas, como o caso da Figura (3.5) (b), há vetores que permanecem fora da função de distribuição de probabilidade devido à existência de regiões não-convexas. Villmann et al. (1997) considera que manter as conexões dentro das áreas de probabilidade não-nula é essencial para a preservação da topologia. A preservação da topologia em um mapa topológico $\mathcal{A}$ depende da propriedade de que dados de entrada $\mathcal{M}$ sejam similares em $\mathcal{A}$ e dados mapeados em $\mathcal{A}$ refiram-se a apenas entradas similares em $\mathcal{M}$. Desta forma, as conexões

²Na Figura (3.5) (a), a área sombreada, e em (b) a área delimitada, representam o espaço em que há probabilidade (não-nula) de ocorrência de um dado de entrada.

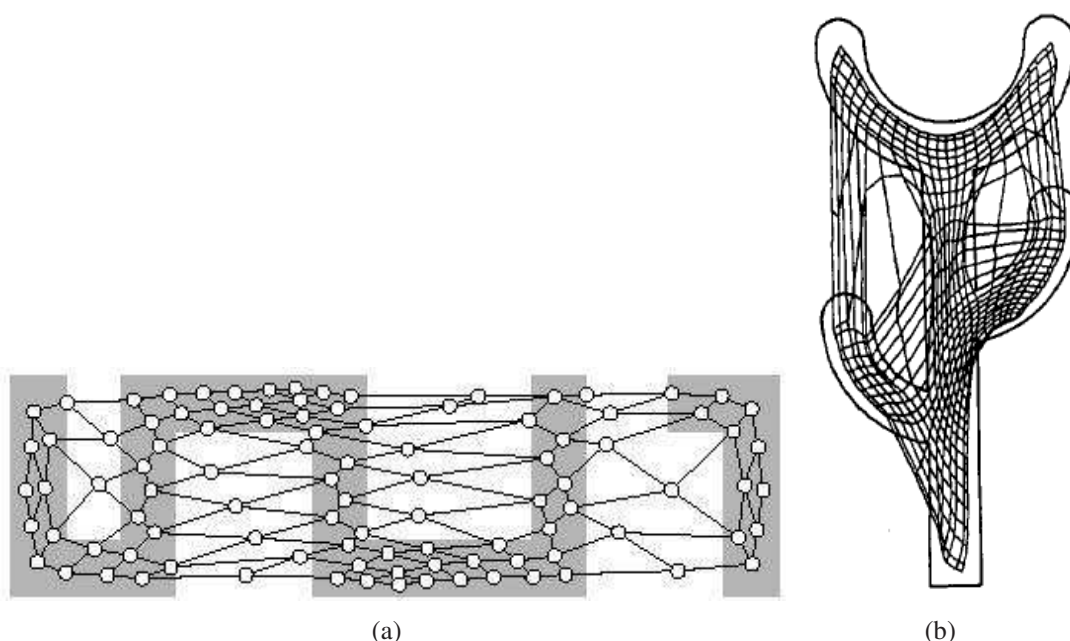


Figura 3.5 (a) Exemplo do treinamento do Mapa de Kohonen após 10000 iterações, com um mapa retangular composto por 100 nodos (Fritzke, 1997a). (b) Exemplo de distribuição de vetores peso para uma função de densidade estruturada em forma de cacto (Kohonen, 1987, p. 137).

que estão externas à área delimitada no exemplo criam vizinhanças inexistentes, diminuindo o grau de preservação de topologia do mapa em questão.

3.2.3 Limitações do SOM

Apesar do SOM ser considerado um avanço nos modelos da época e ser referência para a grande maioria dos modelos auto-organizados que lhe sucederam, o SOM original possui algumas limitações que devem ser consideradas.

Levando-se em conta que o mapa deve ter sua estrutura inicial definida, o primeiro problema que surge é determinar qual o número de nodos necessários para representar a massa de dados que será apresentada à rede. Pode-se definir uma função de erro que mede a distância entre o vetor de pesos e o vetor de entrada para que se tenha uma forma de avaliar se determinado mapa está representando bem os dados de entrada. Existem várias funções de erro que se prestam a esse papel na literatura, entre as quais a mais famosa é a função de erro quadrático.

Outro problema de natureza estrutural surge quando se tem que escolher que tipo de representação geométrica o mapa deverá apresentar. Para que represente bem os padrões de entrada, a topologia do SOM deve ser semelhante ao espaço de entradas (Martinetz et al., 1993). De-

terminar o tipo de mapa sem informações prévias relativas aos padrões de entrada só pode ser feito através de métodos de tentativa e erro, ou como sugere a literatura (Haykin, 1994, p. 449), utilizar uma representação geométrica padrão considerada boa para determinado domínio abordado. A representação geométrica também inclui a dimensão do SOM original que utiliza arranjos uni ou bidimensionais (Kohonen, 1987, p. 119) para reduzir a dimensão do espaço de entradas. Tais arranjos não são sempre adequados pois cada problema de domínios diferentes requer configurações específicas diferentes.

A capacidade da rede de Kohonen de reduzir a dimensão de um conjunto de dados para uma, duas ou três dimensões pode ser considerada uma vantagem, em problemas como análise de componentes principais. Para outros problemas, no entanto, não necessariamente se deseja reduzir a dimensão, mas talvez criar uma representação fidedigna do espaço de padrões de entrada. Ainda pode se desejar reduzir a dimensionalidade para um menor grau e representá-la o melhor possível dentro da capacidade da nova dimensão.

Como se viu na Figura (3.5), o SOM ainda apresenta *defeitos* topológicos quando a geometria do mapa SOM escolhido não é adequada para mapear o espaço de entradas, e normalmente tal geometria não é conhecida *a priori* para se minimizar este problema. Os defeitos topológicos surgem quando nodos ou conexões do mapa criado não cumprem sua função de mapear apenas sinais de entradas similares no espaço de entrada, e vice-versa (Villmann et al., 1997).

Além disso, o SOM original não lida com padrões de entrada dispostos no tempo, ou seja, padrões que se diferenciam apenas quando se considera a ordem em que são apresentados, ou para levantar um problema ainda mais complexo, pode-se considerar que é importante o intervalo de tempo entre as apresentações de cada padrão de entrada, além de sua ordem.

Um outro problema do SOM original compreende o vetor de pesos que é definido para representar o espaço de entradas. O vetor de pesos é considerado um conjunto homogêneo de informação, e por isso pode ser operado como um todo através de cálculos de similaridade como a distância Euclidiana, além de ter sua posição modificada por taxas de aprendizado. Um sistema cuja descrição dos seus estados seja composta por informações de diferentes categorias (heterogêneas) pode ter tais descrições deformadas quando o vetor é operado desta forma.

Outra característica a ser abordada de modo a melhorar o modelo SOM é o papel das suas conexões. As conexões no modelo SOM não são definidas dinamicamente, são pré-determinadas pela geometria do modelo. Estas conexões ligam pretensos vizinhos “topológicos” desde o início do treinamento, quando o mapa ainda não se desdobrou, e portanto o significado das conexões depende da movimentação dos nodos para posições similares. Se há distanciamento de determinados nodos conectados, suas conexões deixam de representar a vizinhança dada por similaridade topológica. Como não são criadas, as conexões não podem

representar qualquer tipo de ordem na apresentação dos padrões de entrada.

Cada um desses problemas foi atacado de diversas formas por muitos algoritmos que surgiram inspirados nos mapas auto-organizados de Kohonen. Esse é sem dúvidas um mérito do SOM, de ter injetado ânimo em um campo de pesquisa inteiro que dele se derivou (Kohonen, 1987; Russel & Norvig, 2002; Debrun, 1996; Fahlman & Lebiere, 1990; Benante, 2001a; Fritzke, 1997a; Marsland et al., 2002; Walter & Schulten, 1993; Sanguineti & Morasso, 1992; Martinetz et al., 1991).

Os métodos que serão vistos nas próximas seções apresentam melhorias em relação ao SOM, e estão dispostos em uma ordem que mostra essa evolução.

3.3 Redes de Representação de Topologia - *Topology-Representing Networks (TRN)*

Este modelo, proposto por Martinetz et al. (1991) baseia-se na combinação de duas técnicas distintas: o método de quantização vetorial *Neural Gas* e a *Aprendizagem Competitiva Hebbiana*. A idéia proposta é distribuir um número específico de nodos em uma região do espaço de dados de entrada, de acordo com alguma função de probabilidade, seguindo o algoritmo *Neural Gas*, e gerar uma topologia, criando-se conexões de acordo com a *Aprendizagem Competitiva Hebbiana*. A topologia poderá ter dimensões diferentes para áreas diferentes do espaço de entrada.

Esta combinação de métodos é conhecida por *Redes de Representação de Topologia* (*Topology-Representing Networks* – TRN) ou simplesmente *Aprendizado Topológico*. Será visto cada algoritmo separadamente, para depois se considerar sua conjugação.

3.3.1 Gás Neural - *Neural Gas (NG)*

O método de quantização vetorial NG não se refere a uma topologia propriamente dita, mas apenas aos nodos isolados, que serão distribuídos num espaço $\mathbb{R}^D$, de acordo com uma probabilidade $P(\xi)$.

O princípio do NG é que para cada sinal de entrada ξ , deve se adaptar todos os nodos segundo uma ordem classificatória, primeiro mais próximo, segundo mais próximo, etc., com relação a entrada ξ . Com o passar do tempo, deve-se diminuir o número de nodos que serão adaptados, até que somente o primeiro vencedor seja adaptado.

• *Algoritmo do NG:*

1. Inicialize o conjunto $\mathcal{A}$ com n unidades i , em posições $\mathbf{w}_i \in \mathbb{R}^D$, $i = 1, 2, \dots, n$ aleatórias, tal que $\mathcal{A} = \{i_1, i_2, \dots, i_n\}$. Inicialize o parâmetro de tempo $t = 0$.
2. Gere um sinal de entrada ξ de acordo com $P(\xi)$.
3. Ordene os elementos de $\mathcal{A}$, e também os seus índices $(i_0, i_1, \dots, i_{n-1})$, de acordo com a distância entre $\mathbf{w}_{i_k}$ e ξ , de tal modo que $\|\mathbf{w}_{i_k} - \xi\| < \|\mathbf{w}_{i_{k+1}} - \xi\|$.
4. Ajuste os vetores de peso segundo a equação:

$$\Delta \mathbf{w}_i = \varepsilon(t) \cdot h_y(k) \cdot (\xi - \mathbf{w}_i) \quad (3.7)$$

com as dependências de tempo: $y(t) = y_i(y_f/y_i)^{t/t_{\max}}$, $\varepsilon(t) = \varepsilon_i(\varepsilon_f/\varepsilon_i)^{t/t_{\max}}$ e $h_y(k) = \exp(-k/y)$ onde $y(t)$ e $\varepsilon(t)$ são funções de decaimento no tempo, para as quais os valores iniciais (y_i, ε_i) e finais (y_f, ε_f) devem ser escolhidos, $t_{\max}$ é o limite de iterações, e k é o índice da posição de um nodo i_k no *ranking* de proximidade dos nodos, e $h_y(k)$ é uma função de proximidade do *ranking*.

5. Incremente o parâmetro de tempo: $t = t + 1$
6. Caso $t < t_{\max}$, continue a partir do passo 2.

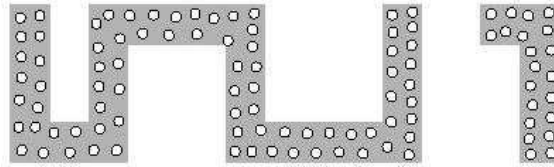


Figura 3.6 Exemplo do algoritmo NG após $t_{\max} = 40000$ iterações. Parâmetros usados na simulação: $\varepsilon_i = 0,4$, $\varepsilon_f = 0,005$, $y_i = 10$, $y_f = 0,01$, e 100 nodos (Fritzke, 1997a).

Na Figura (3.6) pode-se ver o resultado da aplicação deste algoritmo após 40000 sinais de entrada serem avaliados. Nota-se que não há informação topológica no NG, ou seja, não existem conexões entre os nodos. A distribuição dos nodos reflete apenas a função de densidade de probabilidade do espaço de entrada.

O algoritmo NG é normalmente utilizado para quantização vetorial. Com alguma modificação em sua estrutura, Walter & Schulten (1993) o utiliza com sucesso para o aprendizado de um

espaço de estados de um manipulador robótico. A principal modificação é a utilização de três vetores distintos para representar as informações heterogêneas que necessitam ser aprendidas: a posição espacial ($\mathbf{w}_i$), o vetor de saída com os ângulos das juntas $\boldsymbol{\theta}_i$ e a matriz jacobiana $\mathbf{A}_i$. Assim, cada nodo i armazena três diferentes informações, que são tratadas de forma diferente no processo de aprendizado. Para cada iteração do algoritmo, o vetor de entrada ξ apresentado permite extrair de câmeras de vídeos e sensores, todas informações necessárias para o treinamento do NG.

O processo de adaptação de pesos utilizado também é separado para cada parte do vetor, tal como:

$$\mathbf{w}_{i_k} \leftarrow \mathbf{w}_{i_k} + \varepsilon \cdot h_y(k) \cdot (\xi - \mathbf{w}_{i_k}) \quad (3.8)$$

$$\boldsymbol{\theta}_{i_k} \leftarrow \boldsymbol{\theta}_{i_k} + \varepsilon \cdot h_y(k) \cdot \Delta \boldsymbol{\theta}_{i_k} \quad (3.9)$$

$$\mathbf{A}_{i_k} \leftarrow \mathbf{A}_{i_k} + \varepsilon \cdot h_y(k) \cdot \Delta \mathbf{A}_{i_k} \quad (3.10)$$

onde os valores de adaptação $\Delta \boldsymbol{\theta}_{i_k}$ e $\Delta \mathbf{A}_{i_k}$ devem ser calculados. Neste algoritmo, foi utilizado o gradiente descendente de uma função de custo quadrática (Walter & Schulten, 1993). A correção de erro é baseada na regra de aprendizado de Widrow-Hoff.

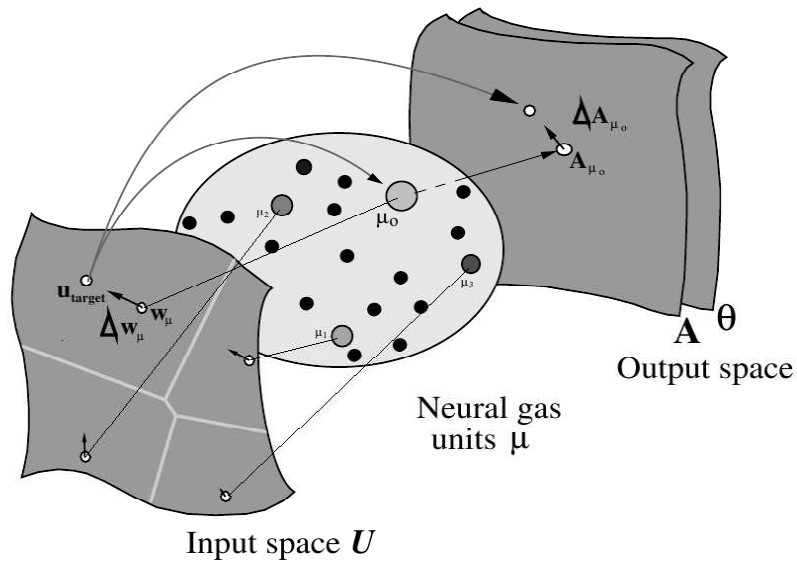


Figura 3.7 Representação esquemática do modelo *Neural Gas* para aproximação linear local aplicada à robótica: *Na esquerda:* o espaço de entradas U . *No centro:* os nodos do NG, que contém informações sobre $\mathbf{w}$, $\boldsymbol{\theta}_i$ e $\mathbf{A}_i$. *Na direita:* os espaços de saída mapeados por cada nodo i , relacionando $\mathbf{w}_i$ com seus correspondentes $\boldsymbol{\theta}_i$ e $\mathbf{A}_i$ (Walter & Schulten, 1993).

Na Figura (3.7) pode-se ver uma representação deste algoritmo. Como é de se esperar, o modelo NG não cria conexões, e portanto não pode representar a topologia do espaço de entradas. Para suprimir essa deficiência, o algoritmo completo TRN combina o NG com o um modelo de aprendizado de conexões, o Aprendizado Competitivo Hebbiano (CHL).

3.3.2 Aprendizado Competitivo Hebbiano - *Competitive Hebbian Learning (CHL)*

O CHL considera um número de nodos no espaço $\mathbb{R}^D$ e insere conexões topológicas entre eles segundo o princípio de Hebb: *para cada sinal de entrada ξ , conecte os dois nodos mais próximos deste sinal (s_1, s_2), por uma aresta.*

Isto gera um grafo que é um subgrafo da triangulação de Delaunay. Conforme [Martinetz et al. \(1993\)](#) a triangulação induzida de Delaunay preserva a topologia otimamente nos casos gerais. Esta triangulação induzida de Delaunay é conseguida quando o algoritmo limita-se a conectar nodos apenas se pelo menos parte de suas regiões de Voronoi estiverem em regiões com probabilidade $P(\xi) > 0$.

- *Algoritmo do CHL:*

1. Inicialize o conjunto $\mathcal{A}$ com n unidades i , em posições $\mathbf{w}_i \in \mathbb{R}^D$, $i = 1, 2, \dots, n$ aleatórias, tal que $\mathcal{A} = \{i_1, i_2, \dots, i_n\}$. Inicialize o conjunto de conexões $\mathcal{C}$, tal que $\mathcal{C} \subset \mathcal{A} \times \mathcal{A}$, com o conjunto vazio $\mathcal{C} = \emptyset$.
2. Gere um sinal de entrada ξ de acordo com $P(\xi)$.
3. Determine os nodos $s_1, s_2 \in \mathcal{A}$ tais que:

$$\|\mathbf{w}_{s_1} - \xi\| \leq \|\mathbf{w}_i - \xi\|, \quad \forall i \in \mathcal{A} \text{ e}$$

$$\|\mathbf{w}_{s_2} - \xi\| \leq \|\mathbf{w}_i - \xi\|, \quad \forall i \in \mathcal{A} - \{s_1\}.$$
4. Se ainda não existir, insira a nova conexão entre s_1 e s_2 em $\mathcal{C}$:

$$\mathcal{C} = \mathcal{C} \cup \{c_{s_1, s_2}\}.$$
5. Continue a partir do Passo 2 até que um número arbitrário de sinais de entrada tenha sido atingido.

A Figura (3.8) nos mostra a disposição das conexões criadas pelo algoritmo CHL. Pode-se notar nessa figura uma tendência a manter as áreas sombreadas conectadas, respeitando-se com alguma precisão a descontinuidade na função de densidade de probabilidade.

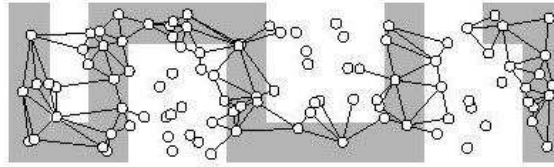


Figura 3.8 Exemplo do algoritmo CHL após 80000 iterações, com $n = 100$ nodos (Fritzke, 1997a).

A inicialização aleatória dos nodos coloca algumas destas em posições que não deveriam ser utilizadas no mapeamento topológico. Como esta técnica não movimenta os nodos, estes nodos que estão fora do campo de probabilidade não-nula estarão fadados à inatividade. Este ponto mostra o interessante motivo para a realização da combinação entre as duas técnicas aqui abordadas.

3.3.3 Combinando as Técnicas

Surgem resultados muito interessantes se, para uma dada massa de dados de entrada, primeiro se executar o algoritmo NG, para posicionar os nodos dentro das áreas de interesse na distribuição de probabilidade, e em seguida aplicar-se o algoritmo CHL, para então criar a topologia da rede neural artificial.

A Figura (3.9) apresenta uma topologia aprendida utilizando-se esta técnica.

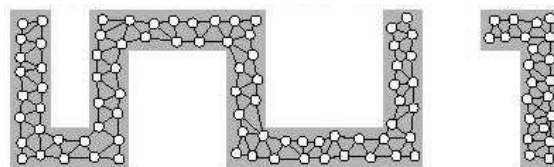


Figura 3.9 Exemplo do algoritmo TRN (união entre NG com CHL) após 40000 iterações. Os parâmetros desta simulação são os mesmos das simulações isoladas do NG (Figura 3.6) e CHL (Figura 3.8) (Fritzke, 1997a).

Nota-se, com relação às técnicas anteriores, a qualidade da representação, que cobre com equiprobabilidade todos os pontos da área sombreada, e mantém a representação da descontinuidade da função de entrada.

Esta técnica também pode ser refinada para que se apliquem ambos os métodos ao mesmo tempo (Martinetz et al., 1991). Neste caso, como os nodos estão em movimento (algoritmo NG) enquanto as conexões estão sendo criadas (algoritmo CHL), pode ocorrer de nodos ficarem fora da área com probabilidade não-nula, o que acarreta a necessidade de algum método de remoção

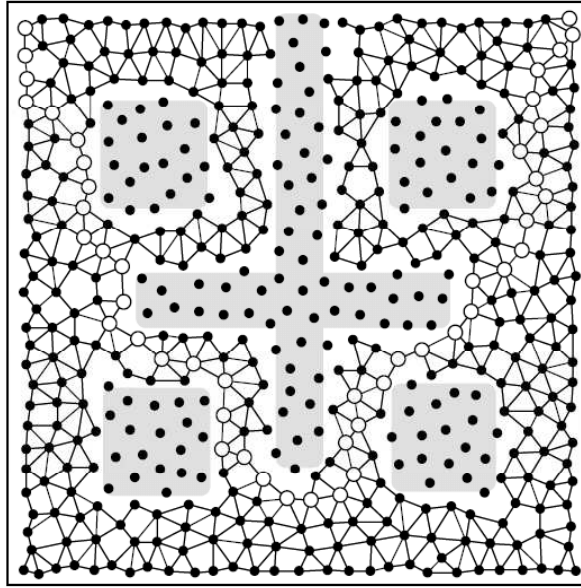


Figura 3.10 Mapeamento da área alcançável por uma rede TRN, e formação de trajetória pelo método de difusão de energia (Zeller et al., 1997). Parâmetros desta simulação: $\varepsilon_i = 0,3$, $\varepsilon_f = 0,05$, $y_i = 100$, $y_f = 0,01$, e $n = 500$ nodos.

de nodos.

O modelo TRN foi utilizado por Zeller et al. (1997) para aprender o Espaço de Controle Perceptual (PCM) de um manipulador robótico. Define-se como PCM o produto do espaço de configuração C de um manipulador de n graus de liberdade $C \equiv \Theta_1 \times \Theta_2 \times \dots \times \Theta_n \subseteq \mathbb{R}^n$, composto pelos ângulos das juntas Θ , pelo espaço sensorial (visual) S captado por sensores diversos (câmeras de vídeo) que dão informações sobre o manipulador, as posições cartesianas da junta, neste modelo. Deste modo, um nodo i do modelo TRN representa ângulos θ_i e posições w_i . Generalizando o espaço PCM para uma dimensão n , a configuração conjunta mão/olho é definida variando todos os ângulos das juntas e considerando a hipersuperfície paramétrica definida pelo espaço $\Theta \times S$.

Uma vez que o modelo TRN provê um mapa topológico ao conectar os nodos pelo CHL, Zeller et al. (1997) propõem a geração de trajetórias através da utilização de uma função de difusão de energia (Capítulo 4, Equação (4.14), p. 102). Outros algoritmos de busca global poderiam ser utilizados, porém o uso da função de difusão de energia, por ser computacionalmente barato, é adequado para no futuro se modificar o modelo para trabalhar em tempo real. A função de difusão de energia é da ordem de complexidade $O(N^2)$, sendo N o número de nodos do mapa, e se sobressai principalmente em mapas onde obstáculos formam figuras complexas, pois é independente do formato destes.

A Figura (3.10) mostra como uma rede TRN (Zeller et al., 1997) foi treinada, mapeando todo o espaço passível de movimento dentro do limite de um manipulador. O modelo TRN não detecta obstáculos, ao invés disso a fase NG move os nodos para fora das áreas de probabilidade nula. Nesta mesma figura, os nodos que pertencem a posições inatingíveis (obstáculos) foram desconectados após o treinamento da rede, por um simples procedimento de remoção que é alheio ao TRN, feito para simular a detecção de obstáculos no espaço visual. Os nodos representados por círculos abertos mostram uma trajetória criada pelo mecanismo de difusão de energia, desde o canto superior esquerdo até o alvo, o canto superior direito.

3.3.4 Avanços e Limitações do TRN

A combinação dessas duas técnicas, ou seja, o uso do algoritmo TRN é um método efetivo de aprendizado. A topologia gerada representa muito bem o espaço de entrada, quando utiliza-se um número suficiente de nodos. Além disto, o CHL evita a formação de defeitos topológicos, como os que se observam no SOM.

Uma limitação deste modelo é que se deve determinar *a priori* um número adequado de nodos, o que não é tarefa trivial. Ainda, como o algoritmo NG necessita de uma taxa de decaimento, é necessário fixar de antemão o número de passos de adaptação. O parâmetro $h_y(k)$ deve ser grande o suficiente para movimentar todos os nodos no início do processo, e deve decair suficientemente rápido para permitir que os nodos se estabilizem em posições equiprováveis não-nulas dentro do espaço de entrada. Caso não seja feita essa sintonia, ter-se-ia em um extremo, o surgimento de nodos “mortos” (fixos em áreas de probabilidade nula, ou muito próximos e mal distribuídos), e no outro extremo, um desperdício de tempo de computação.

Este método tem sido utilizado para quantização vetorial e agrupamento de dados. Há também uma versão supervisionada que associa um mapa linear local a cada unidade e pode ser usado, por exemplo, para predição de séries temporais (Martinetz et al., 1993).

O método de aprendizado do modelo TRN não faz distinção de informações heterogêneas que possam estar configuradas nos seus nodos, o que ocasiona distorções, além de ter uma única medida de similaridade para determinar vizinhanças topológicas, o que limita os tipos de trajetórias que podem ser formadas por um algoritmo de busca aplicado *a posteriori*.

A formação das conexões no TRN leva em conta o CHL, que mede a similaridade entre os nodos após estes estarem distribuídos. Portanto o papel das conexões é nitidamente mapear a topologia do espaço em relação a um vetor completo e uniforme, e não tem qualquer relação com critérios de otimização específicos de domínios para os quais o modelo possa estar sendo aplicado em determinado experimento.

Há também o problema da presença de nodos isolados, que por isso não fazem mais parte do mapa topológico, mostrados na Figura (3.10), quando se tenta mapear obstáculos dinamicamente. Para solucionar este problema, o próprio Zeller et al. (1997) (e também Fritzke (1997a)) sugere uma extensão do TRN que introduz uma topologia crescente (dinâmica), o *Growing Neural Gas* (GNG) que será visto após a apresentação do *Growing Cell Structures* (GCS) na Seção 3.4.

3.4 Crescimento de Estruturas de Células - *Growing Cell Structures (GCS)*

Um outro modelo de mapa auto-organizável, o GCS (Fritzke, 1994a), foi proposto com o intuito de possibilitar que a própria RNA ache uma estrutura neural adequada para a massa de dados com a qual está lidando. O modelo GCS avança no sentido de que não há mais uma estrutura pré-definida, mas apenas alguns nodos³, além de um mecanismo de se criar ou remover nodos. Com isto, a rede pode crescer alterando tanto sua forma, quanto a posição e quantidade de elementos que a compõe. Como será visto adiante, há também um mecanismo para a remoção de elementos redundantes. estabelecido.

O principal objetivo do GCS, é gerar um mapa topológico $\mathcal{A}$ representativo dos dados de entrada $\mathcal{M}$, com menor dimensão $D_{\mathcal{A}} < D_{\mathcal{M}}$, e que atenda a três propriedades:

- Sinais de entradas similares em $\mathcal{M}$ são mapeados em elementos topologicamente próximos em $\mathcal{A}$.
- Elementos topologicamente próximos em $\mathcal{A}$ mapeiam sinais de entrada similares.
- Regiões de $\mathcal{M}$ com alta densidade de probabilidade são representados em $\mathcal{A}$ por uma alta densidade de elementos.

A topologia inicial deste modelo é um conjunto $\mathcal{A}$ de $k + 1$ nodos (vértices) de um grafo *simplex*, onde k é a dimensão do grafo. Durante a execução da rede, os nodos serão adicionados e aqueles considerados desnecessários serão removidos, sempre mantendo a característica *k-simplex* da rede. O modelo preserva a topologia dos dados de entrada se a dimensão do grafo k é a mesma do espaço de entrada (Marsland et al., 2002).

³Serão vistos quantos, com mais detalhes, a seguir.

A área delimitada por um conjunto de nodos é chamada de *hipertetraedro*. A escolha do tetraedro ao invés do cubo se deve ao crescimento dos vértices ser linear e proporcional a k no primeiro, ao passo que este crescimento se torna exponencial (2^k) no segundo caso.

Outro conceito muito importante neste modelo é o de *Regiões de Voronoi*. Informalmente, uma região de Voronoi é a área formada pelo conjunto de pontos F_i que estão *mais* próximos de um nodo i que de outro nodo qualquer. Em outras palavras, todo nodo i tem um vetor de pesos próprio $\mathbf{w}_i$, de dimensão $D_{\mathcal{M}}$, chamado vetor sináptico. O mapeamento $\Psi_{\mathcal{M} \rightarrow \mathcal{A}}$ pode então ser descrito como:

$$\Psi_{\mathcal{M} \rightarrow \mathcal{A}} : \mathcal{M} \rightarrow \mathcal{A}, (\xi \in \mathcal{M}) \mapsto (s_1 \in \mathcal{A}) \quad (3.11)$$

sendo que s_1 é o nodo vencedor, e é encontrado por meio da distância Euclidiana:

$$\|\mathbf{w}_{s_1} - \xi\| = \min_{i \in \mathcal{A}} \|\mathbf{w}_i - \xi\|$$

Feito isto, o conjunto de entrada $\mathcal{M}$ será dividido em i regiões $F_i (i \in \mathcal{A})$, cada qual consistindo em pontos de $\mathcal{M}$ que têm seus vetores sinápticos topologicamente próximos. Uma simplificação é feita pelo autor, com intenção de facilitar os cálculos, para que não exista regiões de Voronoi de tamanho infinito.

3.4.1 Algoritmo

Para o GCS, em comparação ao modelo original de Kohonen, o autor apresenta duas diferenças significativas em seu algoritmo:

- As taxas de aprendizado (ou de adaptação) são constantes no tempo. Tem-se ϵ_b para o nodo vencedor s_1 e ϵ_n para os vizinhos *diretos*⁴ de s_1 .
- Somente o nodo vencedor e seus vizinhos diretos são adaptados.

O processo de adaptação consiste em movimentar o nodo no espaço dos vetores sinápticos visando a minimização da distância média dos sinais de entrada ξ que ele representa. Este processo ocorre a cada passo, sendo que o processo de inserção de um novo nodo ocorre apenas após λ passos de adaptação. Assim, o algoritmo é apresentado a seguir:

⁴Os vizinhos diretos de um nodo são os vizinhos que possuem uma aresta ligando-os ao respectivo nodo.

1. Escolha um sinal de entrada ξ de acordo com a distribuição de probabilidade $P(\xi)$.
2. Localize a unidade vencedora s_1 , a de menor distância Euclidiana, de acordo com:

$$\|\mathbf{w}_{s_1} - \xi\| = \min_{i \in A} \|\mathbf{w}_i - \xi\| \quad (3.12)$$

3. Atualize os pesos de s_1 e de seus vizinhos diretos:

$$\Delta \mathbf{w}_{s_1} = \varepsilon_b (\xi - \mathbf{w}_{s_1}) \quad (3.13)$$

$$\Delta \mathbf{w}_i = \varepsilon_n (\xi - \mathbf{w}_i) \quad \forall i \in \mathcal{N}(s_1) \quad (3.14)$$

4. Acrescente uma unidade ao contador de sinais de s_1 , tal que:

$$h_{s_1} = h_{s_1} + 1 \quad (3.15)$$

5. Decremente todos contadores de sinais por uma fração α , tal que:

$$\Delta h_i = -\alpha h_i \quad (\forall i \in \mathcal{A}) \quad (3.16)$$

- *Inserção de um Nodo e suas Conexões a cada λ passos:*

O objetivo do GCS é obter um conjunto de nodos $i \in \mathcal{A}$ tais que os vetores protótipos $\mathbf{w}_i$ sejam representativos da função de probabilidade $P(\xi)$ com $\xi \in \mathcal{M}$. Isto acontece quando cada nodo i tem a mesma probabilidade de ser o *vencedor* (s_1) para uma dada entrada ξ .

Um indicador da *freqüência relativa* $\tilde{h}_i$ dos sinais de entrada que foram contados por h_i , é dado por:

$$\tilde{h}_i = \frac{h_i}{\sum_{\forall j \in \mathcal{A}} h_j} \quad (3.17)$$

O algoritmo acima consiste em avaliar este indicador (veja Figura 3.11), e criar um novo nodo r exatamente no centro da região de Voronoi que está mais mal representada. Isso é conseguido, escolhendo-se o ponto intermediário entre o nodo com maior freqüência relativa q e seu vizinho mais distante f .

$$\mathbf{w}_r = (\mathbf{w}_q + \mathbf{w}_f)/2 \quad (3.18)$$

A nova formação das ligações dos vértices deve manter a característica de grafo k -simplex, mantendo os hipertetraedros. Isto é feito em dois passos:

- A antiga ligação entre q e f é desfeita.
- O novo nodo r é ligado a q , a f e aos vizinhos que são comuns a ambos (neste caso n_1 e n_2).

Após a inserção do novo nodo, os contadores de sinais de entrada h_i dos nodos i vizinhos de r , devem ser redistribuídos da seguinte forma:

$$\Delta h_i = \frac{|F_i^{(\text{new})}| - |F_i^{(\text{old})}|}{|F_i^{(\text{old})}|} \cdot h_i \quad (\forall i \in \mathcal{N}(r)) \quad (3.19)$$

onde $|F_i|$ é o volume $D_{\mathcal{M}}$ -dimensional da região de Voronoi F_i . E o novo nodo r tem seu contador inicialmente ajustado para:

$$h_r = - \sum_{i \in \mathcal{N}(r)} \Delta h_i \quad (3.20)$$

Esta atribuição inicial ajusta o nodo r com a quantidade de respostas que ele supostamente teria dado, se existisse nessa região desde o começo do processo.

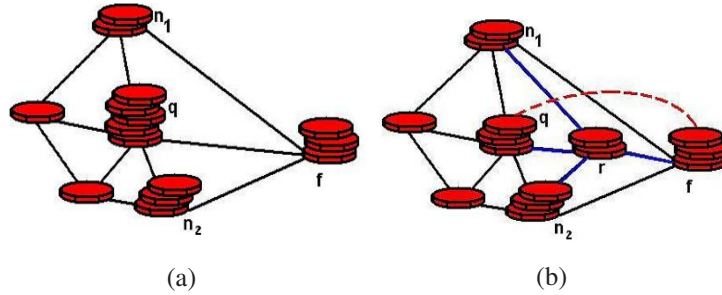


Figura 3.11 Estrutura da rede GCS (a) antes da inserção, e (b) depois da inserção do nodo r .

- *Remoção de um Nodo e de suas Conexões:*

O algoritmo GCS inclui um critério para a remoção de nodos ditos “supérfluos”. Um nodo será tido como supérfluo se estiver localizado em uma região de baixa densidade de probabilidade. Como em geral é muito difícil se obter a densidade de probabilidade, o GCS estima este valor de corte $\tilde{p}$ pela frequência relativa de sinais de um nodo em relação ao tamanho do

seu campo de Voronoi. O limiar de corte κ a ser escolhido dependente do problema, uma vez que as densidades variam para cada caso, tornando muito difícil de se defini-lo como um valor absoluto. Uma aproximação baseada em uma densidade de probabilidade normalizada em relação ao volume total de todos os hipercubos minimiza essa dificuldade, tornando $\kappa = 0,09$ (Fritzke, 1994a) um valor padrão apropriado para a maioria dos casos.

Com estas estimativas em mãos, escolhem-se nodos a serem removidos que não atingiram o limiar $\tilde{p}_i < \kappa$. Mas, além da escolha do nodo, o algoritmo deve tratar do problema de manter a consistência dos hipertetraedros após essas remoções.

O algoritmo: “para remover um nodo, basta removê-lo, e a todas as suas arestas”, não mantém a consistência dos hipertetraedros, como se pode ver na Figura (3.12) (a).

Faz-se necessário manter-se uma base de dados com duas informações: o conjunto de todos os nodos n_i que pertencem a um determinado hipertetraedro h_i , dado por $\bar{C}_{h_i} = \{n_1, \dots, n_n\}$ e o conjunto de todos os hipertetraedros h_i que possuem um determinado nodo n_i , dado por $\bar{H}_{n_i} = \{h_1, \dots, h_n\}$.

No exemplo da Figura (3.12) (a), os triângulos (hipertetraedros) h_1 , h_2 e h_3 são formados pelos nodos $\bar{C}_{h_1} = \{a, b, d\}$, $\bar{C}_{h_2} = \{a, c, d\}$ e $\bar{C}_{h_3} = \{c, e, d\}$, respectivamente. Enquanto que os nodos envolvidos no processo de remoção pertencem é $\bar{H}_{n_a} = \{h_1, h_2, h_4\}$, $\bar{H}_{n_b} = \{h_1, h_5\}$, $\bar{H}_{n_c} = \{h_2, h_3, h_4, h_6\}$, $\bar{H}_{n_d} = \{h_1, h_2, h_3\}$ e $\bar{H}_{n_e} = \{h_3\}$.

A mudança afeta também o algoritmo de inserção, pois é necessário que o banco de dados se mantenha atualizado para representar a estrutura da rede. Com o banco de dados sempre atualizado, para remover um nodo mantendo consistente o grafo k -simplex é necessário e suficiente remover todos os hipertetraedros que possuem o nodo em questão.

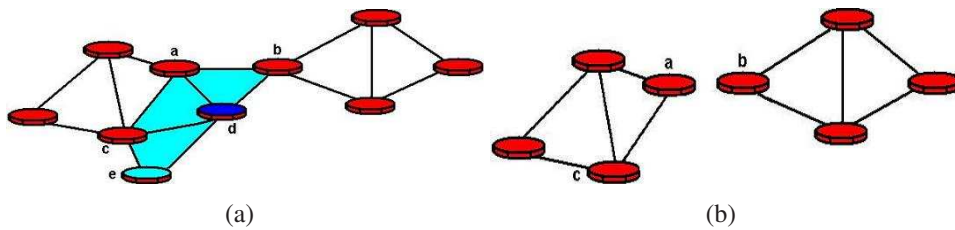


Figura 3.12 Estrutura de uma rede GCS (a) antes de remover o nodo d , e (b) após a remoção do nodo d .

Faz-se isso removendo os hipertetraedros dos conjuntos $\bar{H}_{n_i}$ associados aos nodos envolvidos no processo. Por exemplo, na Figura (3.12) (a), o nodo d pertence aos triângulos h_1 , h_2 , e h_3 , que deixará de existir. Então os nodos envolvidos no processo terão seus conjuntos de hipertetraedros atualizados: $\bar{H}_{n_a} = \{h_4\}$, $\bar{H}_{n_b} = \{h_5\}$, $\bar{H}_{n_c} = \{h_4, h_6\}$, $\bar{H}_{n_d} = \{\}$ e $\bar{H}_{n_e} = \{\}$.

Feita essa atualização no banco de dados, deve-se percorrer todas as arestas envolvidas no processo, eliminando as que não fazem parte de pelo menos um tetraedro. Nova busca, então, deve ser feita, eliminando agora os nodos que não possuem pelo menos uma aresta. No exemplo acima, a primeira busca eliminaria a aresta $\overline{ab}$, e a segunda busca eliminaria o nodo e .

3.4.2 Exemplo

A Figura (3.13) mostra o resultado da aplicação de uma variação do algoritmo GCS proposta por (Fritzke, 1997a), para uma massa de dados convexa e descontínua, com áreas que possuem probabilidade nula.

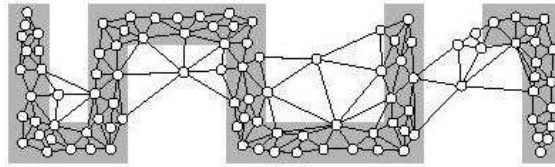


Figura 3.13 Exemplo do algoritmo GCS após 20000 iterações. Parâmetros da simulação: $\alpha = 0,2$, $\beta = 0,0005$, $\varepsilon_b = 0,02$, $\varepsilon_n = 0,006$, e $\lambda = 200$ (Fritzke, 1997a).

Esta mesma distribuição de dados foi usada por Fritzke (1997a) no SOM, e pode-se notar comparando-a com a Figura (3.5) que a quantidade de nodos nas áreas de probabilidade nula diminuiu. Comparando-se com a Figura (3.9), aparentemente a preservação da estrutura do GCS leva a um pior posicionamento que o TRN. Considerando que no exemplo dado o GCS executou por apenas 20000 iterações, enquanto o TRN executou por 40000 iterações, é de se esperar que com mais tempo o GCS melhore sua representação ao inserir mais nodos.

3.4.3 Aplicações Típicas e Variações

Este modelo é utilizado em áreas diversas tais como *otimização combinatorial*, *visualização de dados* (Fritzke, 1994a), *gerenciamento de conteúdo* e *descobrimento de conhecimento* (Freeman & Yin, 2005). Por exemplo, uma rede unidimensional fechada em um anel pode ser usada para gerar aproximações de soluções para o problema do caixeiro viajante, na área de *otimização combinatorial*. Já na área de *visualização de dados*, ao se escolher uma dimensão k -simplex ($k \leq 3$) pequena para representar uma massa de dados de alta dimensão, torna-se possível a visualização gráfica desta massa de dados. Na área de *gerenciamento de conteúdo*, vê-se aplicações para o TreeGCS, uma variante hierárquica do GCS (Freeman & Yin, 2005).

Um primeiro avanço com respeito ao SOM é não ser mais necessário identificar e/ou especificar a estrutura inicial do mapa. Além disto, tem-se parâmetros constantes ao longo do tempo, o que elimina a necessidade de se utilizar algum tipo de função de decaimento.

Porém, outras variações do modelo foram criadas para se adequar melhor a características específicas de determinados problemas, como por exemplo:

- *Redes de Topologias Fechadas*: possuem uma arquitetura na qual os nodos estão relacionados com alguma característica do problema (como por exemplo, as cidades do problema do caixeiro viajante) (Fritzke, 1991).
- *Crescimento Supervisionado de Estruturas de Células (SGCS)*: combina a rede GCS com a abordagem das redes que utilizam funções de base radial. Os novos nodos são inseridos baseando-se no erro acumulado de classificação, o que gera redes pequenas, com grande capacidade de generalização (Fritzke, 1994b).
- *Crescimento Probabilístico de Estruturas de Células (PGCS)*: O algoritmo GCS é um tipo de algoritmo de *k-médias* que realiza agrupamento de dados, de modo *auto-organizado*, através da divisão do espaço de entrada em regiões de Voronoi, enquanto é capaz de, achar uma estrutura e um tamanho de rede que melhor representam este espaço. A proposta do algoritmo PGCS é estender o GCS (Vlassis et al., 1997), assumindo que os dados de entrada de cada agrupamento estão de acordo com a distribuição de probabilidade Gaussiana, e com isto tirar proveito das informações providas por essa distribuição.

3.4.4 Limitações do GCS

Apesar de serem constantes, determinar os diversos parâmetros (ϵ_b , ϵ_n , λ , α e κ) não é tarefa simples, pois é necessário um ajuste empírico para o domínio em questão. O modelo GCS é robusto à maioria de seus parâmetros iniciais, ainda assim é muito sensível às condições de configuração de alguns deles, em especial do parâmetro λ que define o crescimento da rede (Heinke & Hamker, 1998). Determinar o passo adequado ao crescimento da rede, para que adicione nodos a cada λ iterações, coloca nas mãos do projetista uma difícil tarefa de estimar a melhor dinâmica da rede para o problema em questão. Além disto, com o uso de uma taxa de inserção, o GCS não permite controlar a precisão dos agrupamentos determinados pelo modelo, pois um novo nodo é inserido a cada λ passos, independentemente do tamanho dos erros.

Outra limitação é a complexidade do algoritmo, decorrente da manutenção da rede estruturada em um grafo *k-simplex*. Tal representação tem custo computacional alto, sem contrapartida na representação da topologia do espaço de estados (Marsland et al., 2002). Pelo contrário, para

escolher a dimensão k do grafo k -simplex necessita-se conhecimento prévio da dimensão do espaço de entrada, sob risco de se criar uma rede que não representa adequadamente a topologia a ser aprendida.

O PGCS atribui como um problema a rede GCS não distribuir os seus nodos de modo cobrir o espaço de estados com um número homogêneo de nodos, mas criar maior quantidade de nodos onde a distribuição de probabilidade é mais densa. Isto não é necessariamente um problema, uma vez que pode ser interessante representar também a densidade na topologia do espaço de entradas. O GCS, do modo como o processo de inserção de nodos é feito, tem o objetivo principal de equalizar o valor esperado do erro de cada nodo, ao invés de priorizar a representação topológica dos espaço de entradas durante o seu treinamento.

O papel das conexões do GCS tem o objetivo de manter o mapa em uma malha de hiper-tetraedros, e não há qualquer relação com as propriedades dos nodos conectados a não ser a similaridade medida pela distância Euclidiana do vetor completo. Adicionalmente, como se pode notar no exemplo da Figura (3.13), ocorrem os mesmos defeitos topológicos apontados no SOM, com conexões fora da área de probabilidade não-nula, causados pela restrição de manter-se o grafo k -simplex. O número de conexões cortando as áreas de probabilidade nula tende a diminuir com mais treinamento. Entretanto a performance da rede não melhora significativamente se treinada em excesso (Heinke & Hamker, 1998).

3.5 Gás Neural Crescente - *Growing Neural Gas (GNG)*

O modelo GNG (Fritzke, 1997a) é derivado do GCS, sem as suas restrições topológicas, ou, também pode ser visto como uma variação incremental do modelo TRN. O propósito do modelo GNG é gerar um grafo que reflita a topologia dos dados de entrada, inclusive representando diferentes dimensionalidades, caso elas ocorram. Normalmente utiliza-se este modelo para agrupamentos e identificação de grupos, bem como quantização vetorial. As conexões entre os vizinhos podem ser utilizadas como informação sobre valores de interpolação de funções arbitrárias de $\mathbb{R}^D$.

O modelo GNG compartilha com o modelo TRN o uso do aprendizado Hebbiano. Já o procedimento de adição de nodos utilizado neste modelo possui algumas vantagens em relação ao modelo TRN de topologia fixa. Pelo fato do modelo GNG não ter as restrições topológicas de grafos k -simplex do modelo GCS, a sua topologia pode assumir quantas dimensões diferentes forem necessárias para bem representar os dados de entrada. Há ainda outras diferenças, conforme se pode observar no algoritmo apresentado em seguida.

3.5.1 Algoritmo

Considere uma rede consistindo de:

- Um conjunto $\mathcal{A}$ de nodos. Cada nodo $i \in \mathcal{A}$ possui um *vetor de referência* $\mathbf{w}_i \in \mathbb{R}^D$, que pode ser entendido como a sua posição no espaço de entrada.
- Um conjunto $\mathcal{C}$ de conexões (ou arestas) entre pares de nodos. Estas conexões não possuem pesos. Seu único objetivo é a definição da estrutura topológica.

A idéia principal do método é incluir novos nodos sucessivamente, partindo de uma rede mínima, através da avaliação das medidas estatísticas locais durante a apresentação dos dados de entrada e seu histórico nos passos anteriores de adaptação (Fritzke, 1997a). A seguir a descrição formal do algoritmo:

1. Inicialize o conjunto $\mathcal{A}$ com dois nodos n_1 e n_2 em posições $\mathbf{w}_{n_1}$ e $\mathbf{w}_{n_2}$ pertencentes a $\mathbb{R}^D$ aleatórias.

$$A = \{n_1, n_2\}$$

Inicialize o conjunto de conexões $\mathcal{C}$ com uma conexão entre n_1 e n_2 , e coloque a idade desta conexão como zero:

$$\mathcal{C} = \{c_{n_1, n_2}\}$$

$$age_{(n_1, n_2)} = 0$$

2. Gere um sinal ξ de entrada, de acordo com $P(\xi)$.

3. Determine os nodos $s_1, s_2 \in \mathcal{A}$ tais que

$$\|\mathbf{w}_{s_1} - \xi\| \leq \|\mathbf{w}_i - \xi\|, \quad \forall i \in \mathcal{A} \text{ e}$$

$$\|\mathbf{w}_{s_2} - \xi\| \leq \|\mathbf{w}_i - \xi\|, \quad \forall i \in \mathcal{A} - \{s_1\}.$$

4. Se ainda não existir, insira a nova conexão entre s_1 e s_2 em $\mathcal{C}$:

$$\mathcal{C} = \mathcal{C} \cup \{c_{s_1, s_2}\}.$$

Existindo ou não a conexão previa, sua idade passa agora a ser zero:

$$age_{(s_1, s_2)} = 0$$

5. Adicione o quadrado da distância entre o sinal de entrada e a unidade mais próxima no espaço de entrada, a uma variável de erro local:

$$\Delta E_{s_1} = \|\mathbf{w}_{s_1} - \xi\|^2 \tag{3.21}$$

6. Mova s_1 e seus vizinhos topológicos diretos em direção a ξ , por frações de ε_b e ε_n da distância total, respectivamente.

$$\Delta \mathbf{w}_{s_1} = \varepsilon_b (\xi - \mathbf{w}_{s_1})$$

$$\Delta \mathbf{w}_i = \varepsilon_n (\xi - \mathbf{w}_i), \quad \forall i \in \mathcal{N}(s_1)$$

onde $\mathcal{N}(s_1)$ é o conjunto de nodos vizinhos de s_1 .

7. Incremente a idade de todas as conexões que emanam de s_1 :

$$age_{(s_1,i)} = age_{(s_1,i)} + 1, \quad \forall i \in \mathcal{N}(s_1)$$

8. Remova as conexões com idade superior a $a_{\max}$. Se esta operação resultar em unidades que não possuam mais nenhuma conexão, remova-as também.
9. Se o número de sinais de entrada é um inteiro múltiplo do parâmetro λ , então insira uma nova unidade como se segue:

- (a) Determine a unidade q como a possuindo o maior erro acumulado, e determine f , a vizinha direta mais distante de q .
- (b) Interpole r entre q e f : $\mathbf{w}_r = (\mathbf{w}_q + \mathbf{w}_f)/2$.
- (c) Insira conexões entre r e q , e entre r e f , e remova a conexão entre q e f .
- (d) Decremente as variáveis de erro de q e f .
- (e) Interpole a variável de erro de r entre q e f .

10. Decremente o erro de todas as unidades:

$$\Delta E_i = -\beta E_i, \quad \forall i \in A.$$

11. Continue a partir do passo 2 até que algum critério de parada tenha sido atingido (por exemplo, o tamanho da rede, ou alguma medida de performance).

3.5.2 Exemplo

Na Figura (3.14) tem-se a disposição das unidades e suas respectivas conexões após 20000 iterações. A rede utilizou como critério de parada um número máximo de 100 unidades atingidas. Pode-se notar que, como no modelo TRN, as unidades estão bem distribuídas na área dos dados de entrada com probabilidade positiva, e conseguiram representar a descontinuidade que este exemplo apresenta.

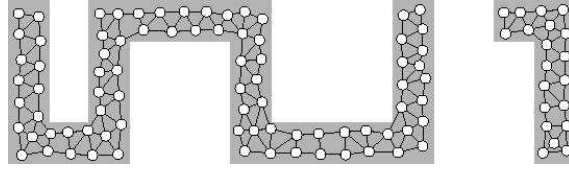


Figura 3.14 Exemplo do algoritmo GNG após 20000 iterações. Parâmetros da simulação: $\alpha = 0,5$, $a_{\max} = 88$, $\beta = 0,0005$, $\lambda = 200$, $\epsilon_b = 0,05$, $\epsilon_n = 0,0006$ (Fritzke, 1997a).

O método descrito acima possui uma fase de adaptação (Passo 6) que movimenta todos os nodos em direção às áreas com sinal de entrada $P(\xi) > 0$. O Passo 4 adiciona uma nova conexão entre o nodo vencedor e o segundo colocado, em relação ao sinal de entrada ξ recebido, que por isso mantém a triangulação de Delaunay. A remoção de nodos feita no Passo 8 é necessária para eliminar conexões que não são mais parte da triangulação de Delaunay, operação esta realizada em conjunto com o Passo 7 que ‘envelhece’ todas as conexões, e com o Passo 4 que ‘rejuvenesce’ apenas as conexões utilizadas, com conseqüente envelhecimento somente das conexões que não estão sendo utilizadas. A remoção e inserção de conexões neste modelo tenta construir e manter a triangulação induzida de Delaunay por todo o espaço que está sendo mapeado. O Passo 5 acumula os quadrados das distâncias entre o sinal ξ e o nodo vencedor, com o intuito identificar nodos que estão dispostos em áreas do espaço de entrada onde o mapeamento está mal representado⁵.

3.5.3 Avanços e Limitações do GNG

Comparando o modelo GNG com o modelo TRN, tem-se como importante avanço não ser necessário pré-definir o tamanho da rede, por ser uma rede de topologia dinâmica, tal qual o GCS. Todos os parâmetros são constantes no tempo, e o número de passos para adaptação não precisa ser definido *a priori*. A variação de performance da rede quanto à dependência da variação dos seus parâmetros (a robustez) foi avaliada e a rede GNG apresenta clara vantagem em uma comparação com as redes MLP e GCS (Heinke & Hamker, 1998), além de convergir muito mais rápido para uma solução que o tempo necessário para uma rede MLP.

Devido à sua característica fractal⁶, o processo de aprendizado pode ser interrompido a qualquer tempo, que tem-se como resposta uma boa representação do espaço de entrada, dado

⁵Por ‘mal representado’ neste caso entenda-se a região do espaço de entrada que está a acumular mais erro local.

⁶O termo *fractal* foi cunhado por Mandelbrot (1975) para explicar objetos com dimensões fracionárias (Ott, 2002). Uma das propriedades dos fractais é a auto-similaridade: um objeto é dito fractal sua estrutura e topologia se repete quando observado em diferentes escalas.

o tempo de computação oferecido.

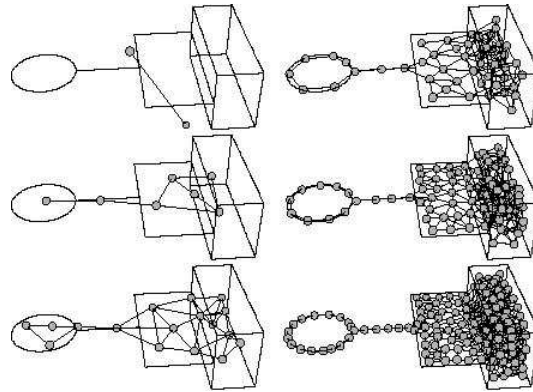


Figura 3.15 Uma figura geométrica composta por formas de diferentes dimensões, sendo mapeada pelo modelo GNG. Parâmetros da simulação: $\alpha = 0,5$, $a_{\max} = 88$, $\beta = 0,0005$, $\lambda = 200$, $\epsilon_b = 0,05$, $\epsilon_n = 0,0006$ (Fritzke, 1997a).

Como limitação, Fritzke (1997a) comenta que este modelo não é interessante para a visualização de dados, uma vez que o modelo não pretende diminuir a dimensionalidade dos dados de entrada (veja Figura 3.15), a não ser que o próprio conjunto de entrada já tenha uma pequena dimensão. Para o mapeamento de espaço de estados, esta característica deve ser considerada positiva, pois pretende-se representar o mapa com maior fidelidade possível e custo computacional adequado.

Este modelo se aplica muito bem para agrupamentos e quantizações vetoriais. É ainda possível combiná-lo com funções de base radial para conseguir um modelo supervisionado incremental (Fritzke, 1997a).

O modelo GNG, entretanto, também depende de um parâmetro que controla o número de passos necessários para a inserção e remoção de nodos, que é de difícil ajuste por ser dependente do espaço de entradas a ser representado. A inserção de nodos no GNG é feita, como no GCS, para minimizar o erro acumulado, a cada λ passos. O novo nodo é inserido na metade da distância entre o nodo de maior erro acumulado e o seu vizinho mais distante. A intenção é utilizar o erro de representação como guia para orientar onde a rede deve crescer com maior prioridade. Neste sentido, não há nenhum apelo sobre plausibilidade biológica, como é o caso do balbuciamiento motor do modelo VITE.

As conexões são inseridas para refletir a inserção de um novo nodo entre os nodos que causaram seu posicionamento, não tendo por isso compromisso com critérios para a geração de trajetórias, mas apenas reflete propriedades topológicas relacionadas a similaridades dos nodos como um todo.

Ainda, as conexões são removidas por um limite de idade máximo que deve ser pré-determinado. Isto permite à rede manter-se atualizada em relação ao espaço de estados dinâmicos, eliminando conexões (e nodos) quando inativos por um dado tempo a ser configurado. O inconveniente é que a rede pode estar se esquecendo de configurações (estados) importantes, que mesmo sem uso no tempo estipulado podem representar uma perda de informação importante e muitas vezes custosa e de difícil aquisição (como o aprendizado de posições de um braço após o balbuciamiento motor).

Por fim, mas não menos importante, a ausência de tratamento diferenciado para vetores com informações heterogêneas impede algumas das características mais importantes para o planejamento de trajetórias, como o uso de múltiplos critérios para vizinhanças e a equifinalidade temporal das trajetórias. Além disto, ao operar o vetor como um todo, sem discriminar seu conteúdo, criam-se distorções entre as diferentes informações nele representadas, que podem estar inclusive em escalas distintas.

3.6 Crescer Quando Necessário - *Grow When Required (GWR)*

Normalmente as redes com topologia dinâmica vistas na literatura adicionam um nodo (ou uma camada de nodos) na posição onde o erro acumulado é alto, ou onde há alguma necessidade de ajuste topológico. Estes nodos são adicionados após um número fixo de iterações, baseados em um parâmetro (λ) pré-definido. Com isto a rede cresce sempre numa mesma taxa, não importando como os dados estão sendo apresentados, e continuam crescendo até o algoritmo atingir um critério de parada. A alteração desta taxa é levemente modificada pelo critério que a rede possa ter estabelecido para a remoção de nodos.

O modelo GWR (Marsland et al., 2002) contorna estas limitações das redes com crescimento ao utilizar um novo critério para inserir e inicializar um nodo. Ao invés de adicionar o novo nodo para dividir a região de um nodo que esteja com a maior taxa de utilização, ou com o maior erro acumulado, adiciona-se o novo nodo sempre que a entrada não for respondida por um nodo pré-existente com uma precisão determinada, além de se considerar a frequência de disparos deste nodo.

3.6.1 Algoritmo

Seja $\mathcal{A}$ o conjunto de nodos do mapa auto-organizável, e $\mathcal{C} \subset \mathcal{A} \times \mathcal{A}$ o conjunto de conexões entre os nodos do mapa. Seja $P(\xi)$ a distribuição de probabilidade dos dados de entrada ξ . E finalmente, seja $\mathbf{w}_i$ o vetor de pesos do nodo i com dimensão D .

1. Inicialize o conjunto $\mathcal{A}$ com duas unidades n_1 e n_2 em posições $\mathbf{w}_{n_1}$ e $\mathbf{w}_{n_2}$ pertencentes a $\mathbb{R}^D$ aleatoriamente selecionadas de $P(\xi)$.

$$\mathcal{A} = \{n_1, n_2\} \quad (3.22)$$

Inicialize o conjunto de conexões $\mathcal{C}$ com o conjunto vazio.

$$\mathcal{C} = \emptyset \quad (3.23)$$

2. Gere um sinal ξ de entrada, de acordo com $P(\xi)$.
3. Para todo nodo i na rede, determine os nodos $s_1, s_2 \in \mathcal{A}$ tais que

$$\|\mathbf{w}_{s_1} - \xi\| \leq \|\mathbf{w}_i - \xi\|, \quad \forall i \in \mathcal{A} \quad (3.24)$$

$$\|\mathbf{w}_{s_2} - \xi\| \leq \|\mathbf{w}_i - \xi\|, \quad \forall i \in \mathcal{A} - \{s_1\} \quad (3.25)$$

onde $\mathbf{w}_i$ é o vetor de pesos do nodo i .

4. Se ainda não existir, insira a nova conexão entre s_1 e s_2 em $\mathcal{C}$:

$$\mathcal{C} = \mathcal{C} \cup \{c_{s_1, s_2}\} \quad (3.26)$$

Existindo ou não a conexão previa, sua idade passa agora a ser zero: $age_{(s_1, s_2)} = 0$

5. Calcule a atividade do melhor nodo s_1 :

$$a(s_1) = \exp(-\|\xi - \mathbf{w}_{s_1}\|) \quad (3.27)$$

6. SE $(a(s_1) < a_T)$ E $(\overline{h_{s_1}} < h_T)$ ENTÃO deve-se adicionar um novo nodo entre os dois nodos melhores (s_1 e s_2), de acordo com os passos abaixo, onde $a(s_1)$ é a atividade do nodo s_1 e $\overline{h_{s_1}}$ é o *descontador* de disparos do nodo s_1 .

(a) Adicione o novo nodo r ao conjunto $\mathcal{A}$:

$$\mathcal{A} = \mathcal{A} \cup \{r\} \quad (3.28)$$

(b) Crie o vetor de pesos para o nodo r como a média entre o sinal de entrada ξ e o melhor nodo s_1 :

$$\mathbf{w}_r = (\mathbf{w}_{s_1} + \xi)/2 \quad (3.29)$$

(c) Insira conexões entre (r, s_1) e entre (r, s_2) , e remova a conexão entre (s_1, s_2) :

$$\mathcal{C} = \mathcal{C} \cup \{c_{r,s_1}, c_{r,s_2}\} \quad (3.30)$$

$$\mathcal{C} = \mathcal{C} - \{c_{s_1,s_2}\} \quad (3.31)$$

SENÃO (não houve uma inserção de um novo nodo), adapte as posições do vencedor s_1 e seus vizinhos topológicos diretos em direção a ξ , por frações de ϵ_b e ϵ_n da distância total e de uma taxa definida sobre a frequência de disparos ($\overline{h_i}$), respectivamente.

$$\Delta \mathbf{w}_{s_1} = \epsilon_b \times \overline{h_{s_1}} \times (\xi - \mathbf{w}_{s_1}) \quad (3.32)$$

$$\Delta \mathbf{w}_i = \epsilon_n \times \overline{h_i} \times (\xi - \mathbf{w}_i), \quad \forall i \in \mathcal{N}(s_1) \quad (3.33)$$

onde $0 < \epsilon_n < \epsilon_b < 1$ e $\overline{h_i}$ é uma taxa com decaimento, definida sobre a frequência de disparos do nodo i .

7. Incremente a idade de todas as conexões que emanam de s_1 :

$$age_{(s_1,i)} = age_{(s_1,i)} + 1, \quad \forall i \in \mathcal{N}(s_1) \quad (3.34)$$

onde $\mathcal{N}(s_1)$ é o conjunto dos vizinhos topológicos diretos de s_1 .

8. Reduza o contador de disparos $\overline{h_{s_1}}$ do nodo vencedor s_1 , e $\overline{h_i}$ de seus vizinhos diretos, segundo:

$$\overline{h_{s_1}}(t) = h^{(0)} - \frac{S(t)}{\alpha_b} (1 - \exp(-\alpha_b t / \tau_b)) \quad (3.35)$$

$$\overline{h_i}(t) = h^{(0)} - \frac{S(t)}{\alpha_n} (1 - \exp(-\alpha_n t / \tau_n)) \quad (3.36)$$

onde $\bar{h}_i(t)$ é uma função que decai exponencialmente quando o nodo i dispara mais freqüentemente, $h^{(0)}$ é a força inicial, normalmente $h^{(0)} = 1$, $S(t)$ é a força do estímulo, normalmente $S(t) = 1$, e as outras constantes na equação controlam o comportamento da curva, com valores sugeridos em: $\alpha_b = 1,05$; $\alpha_n = 1,05$; $\tau_b = 3,33$ e $\tau_n = 14,3$.

9. Remova as conexões com idade superior a $a_{\max}$. Se esta operação resultar em unidades que não possuem mais nenhuma conexão, remova-as também.
10. Continue a partir do passo 2 até que algum critério de parada tenha sido atingido (por exemplo, o tamanho da rede, ou alguma medida de performance).

3.6.2 Discussão

A Figura (3.16) (a) apresenta a estrutura topológica da distribuição de probabilidade dos padrões de entrada. É esta estrutura que a rede GWR deve tentar representar e generalizar. Na mesma figura, na parte (b), tem-se uma primeira fotografia do comportamento da rede GWR após lhe serem apresentados 80 padrões. Na parte (c) a rede GWR está praticamente convergindo para a representação topológica que era seu objetivo, após 320 padrões apresentados.

Este exemplo mostra o comportamento da rede GWR para uma base de dados com distribuição uniforme. A rede GWR tem como principal característica criar nodos muito rapidamente, nas áreas em que há necessidade segundo seus critérios, que se resumem em responder *afirmativamente* a duas questões: (1) o nodo vencedor está longe do padrão atual? (2) o nodo vencedor tem disparado muito? Caso estas duas condições sejam satisfeitas, imediatamente um nodo é criado. Em caso contrário, uma resposta negativa para a condição (1) indica que o vencedor representou bem o padrão atual, portanto não há necessidade de se criar um novo nodo para representá-lo; e uma resposta negativa para a condição (2), indica que o nodo vencedor está sendo subutilizado, o que significa que em vez de se criar um novo nodo, pode-se treinar este nodo para que melhor represente o padrão em questão. Estas duas questões mostram o papel desempenhado pelo descontador de disparos e pela atividade dos nodos.

Para medir a performance da rede GWR, Marsland et al. (2002) utilizaram duas funções de custo distintas, de modo que a combinação destas restrições pudesse indicar uma rede com melhor desempenho. Além destas duas medidas de custo, os autores, através de uma adaptação da função de medição de topologia de Villmann et al. (1997), $\Phi_{\mathcal{A}}^{\mathcal{M}}$, mostraram que a rede GWR atinge o nível de “perfeitamente preservadora de topologia”, mais precisamente, no exemplo dado, a rede GWR atingiu $\Phi_{\mathcal{A}}^{\mathcal{M}}(0) = 0,0023$.

A primeira medida de custo, E_1 , dada na Equação (3.37), penaliza a rede que tem vizinhos

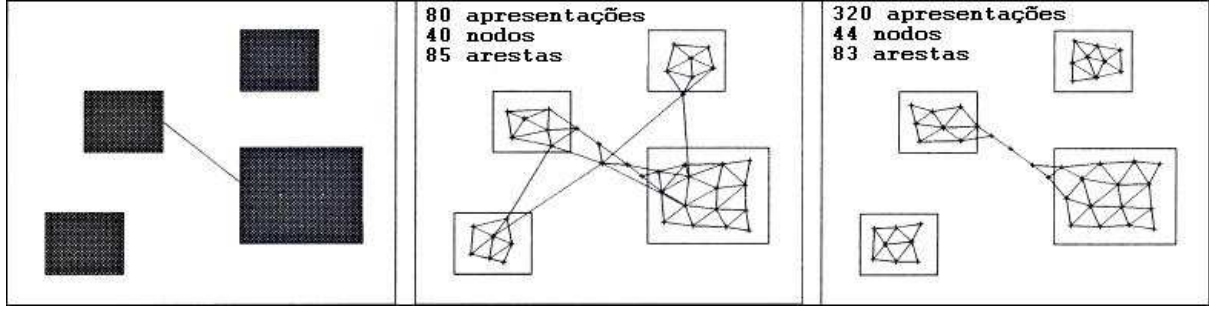


Figura 3.16 (a) Áreas com probabilidade positiva de ocorrência de um padrão de entrada. (b) Estrutura da rede GWR após 80 padrões apresentados à rede. (c) Estrutura da rede GWR após 320 padrões apresentados à rede. Parâmetros da simulação: $\alpha_T = 0,99$, $\epsilon_b = 0,05$ e $\epsilon_n = 0,0006$ (Marsland et al., 2002).

muito distantes uns dos outros.

$$E_1 = \sum_i \sum_{j < i} C_{i,j} \cdot \|\mathbf{w}_i - \mathbf{w}_j\|^2 \quad (3.37)$$

onde:

$$C_{i,j} = \begin{cases} 1 & \text{para } (i,j) \text{ conectados.} \\ 0 & \text{caso contrário.} \end{cases} \quad (3.38)$$

Pode-se observar, para o exemplo dado, a comparação da estimativa E_1 , entre o modelo GWR e o modelo GNG, na Figura (3.17) (a). Como se vê, a rede GWR rapidamente diminui o erro E_1 antes de 80 iterações, e a partir deste ponto permanece abaixo da GNG até 320 iterações. O que significa dizer que esta rede mantém os nodos vizinhos a uma distância pequena já durante o início do processo de execução.

A Equação (3.39) apresenta a segunda medida de custo. Esta função retorna uma avaliação da capacidade da rede de minimizar as distâncias entre os padrões de entrada e os nodos que os respondem. Para o caso de $v \rightarrow \infty$, a medida de custo leva em conta somente o nodo vencedor (caráter *vencedor-leva-tudo*).

$$E_2 = \sum_k \sum_i \|\xi - \mathbf{w}\|^2 \cdot \frac{e^{-v \cdot \|\xi_k - \mathbf{w}_i\|^2}}{\sum_j e^{-v \cdot \|\xi - \mathbf{w}_j\|^2}} \quad (3.39)$$

Pode-se averiguar o desempenho das redes GWR e GNGU (*Growing Neural Gas with Uti-*

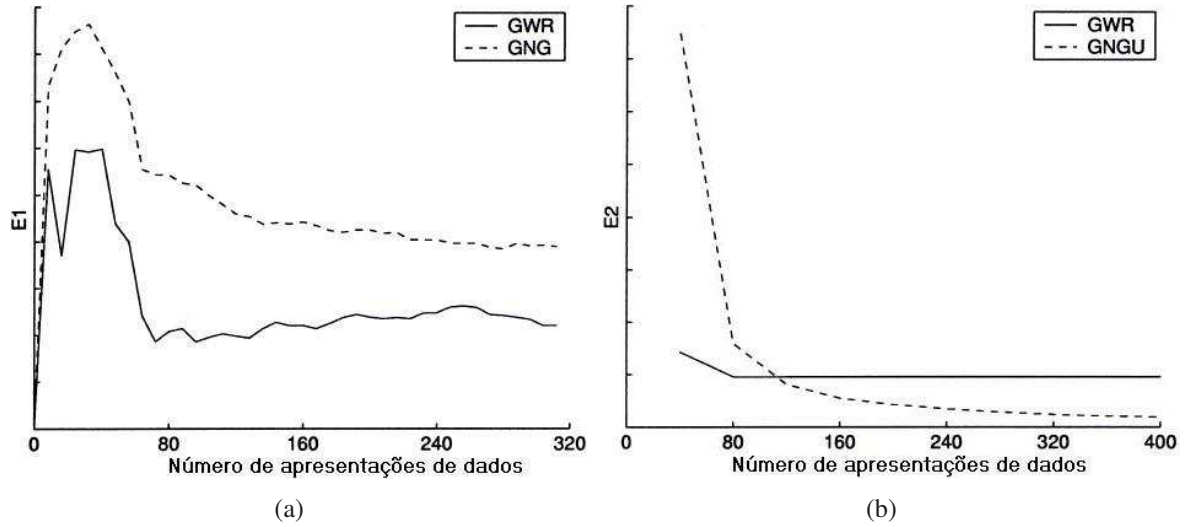


Figura 3.17 (a) Medida de erro E_1 para as redes GWR e GNG. (b) Medida de erro E_2 para as redes GWR e GNGU (Marsland et al., 2002).

lity). Para esta função de custo, as redes mostram resultados semelhantes, sendo que o modelo GWR estabilizou após 80 apresentações de padrões. O nível em que se estabilizou é determinado pelo limiar de atividade a_T que permite a inserção de novos nodos.

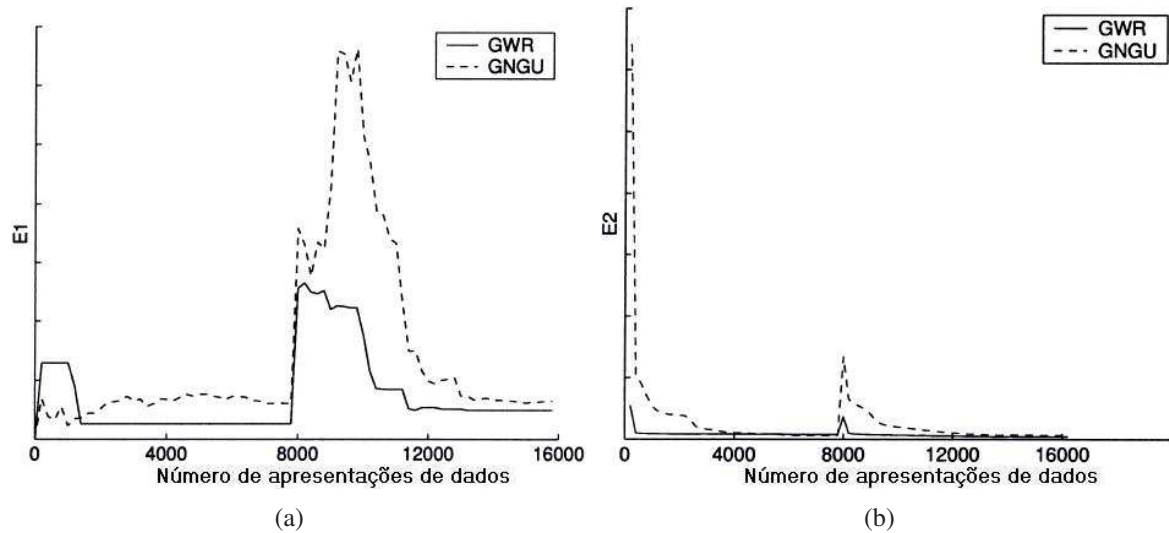


Figura 3.18 (a) Medida de erro E_1 e (b) E_2 para as redes GWR e GNGU (Marsland et al., 2002).

Mais nítida é a qualidade das respostas para as medidas de custo E_1 e E_2 definidas quando há uma mudança brusca na distribuição de probabilidade (ou mapa de entrada) que a rede deve aprender. Na Figura (3.18), por exemplo, há uma mudança no mapa de entrada exatamente

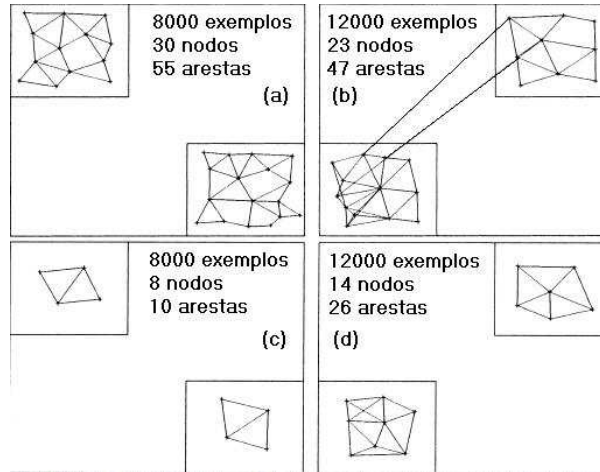


Figura 3.19 GNGU antes (a) e depois (b) da mudança no mapa de entrada. GWR antes (c) e depois (d) da mesma mudança.

a partir da apresentação do padrão de número 8000. Pode-se observar que o modelo GNGU mantém, no caso do E_1 , vizinhos mais distantes e por muito mais tempo que o modelo GWR. Para o E_2 , a medida de custo da distância entre o padrão entrado e o nodo vencedor que o respondeu, observa-se que um mínimo pico de erro para o modelo GWR, que rapidamente se molda para representar o novo mapa de entrada.

A Figura (3.19) mostra o momento da mudança no mapa de entrada. No período anterior aos 8000 padrões apresentados, estes se localizavam nos cantos superior-esquerdo e inferior-direito. Após a apresentação do padrão número 8000, os padrões seguintes se localizavam nos cantos superior-direito e inferior-esquerdo. A rede GNGU manteve por um certo período do treinamento algumas conexões entre áreas distintas, como se pode ver na Figura (3.19) (b).

3.6.3 Limitações do GWR

A rede GWR é robusta em relação a mudanças em seus parâmetros, porém possui mais parâmetros a se ajustar que os modelos GCS e GNG. Não há necessidade de se ajustar um parâmetro λ que indica a taxa de crescimento, como no GNG e GCS, pois a rede cresce sob demanda. Porém ainda é necessário ajustar uma idade limite para o envelhecimento/morte de conexões $a_{\max}$, informação de difícil definição. Também se faz necessário o ajuste da taxa de ativação limite a_T e do limite de disparos de um nodo h_T que juntos definem a taxa de crescimento da rede e acabam por substituir de forma sutil o parâmetro λ .

Ao inserir novos nodos, o algoritmo GWR se baseia nas condições mostradas na Tabela (3.1). A condição 4, única aceitável para a criação de um novo nodo, configura a en-

Tabela 3.1 Condições para inserção de novo nodo no GWR.

Cond.	Atividade	Disparos	Insere
1	Alta	Baixo	Não
2	Alta	Alto	Não
3	Baixa	Baixo	Não
4	Baixa	Alto	Sim

trada de um padrão que está distante do nodo vencedor e este nodo vencedor está com alta quantidade de disparos (ou baixa quantidade de não-disparos, conforme o algoritmo). Como o modelo GWR e também seus antecessores, não faz distinção sobre informações heterogêneas que possam compor a descrição de estados representada por seus nodos, ocorrem distorções nas informações quando os nodos são modificados segundo a Equação (3.29), que considera a média aritmética entre vetores, assumindo uma linearidade nos dados representados. Um outro problema de deformação dos dados surge na condição 3, quando o algoritmo decide que um nodo vencedor que está distante do padrão representado e tem disparado pouco, assim mesmo deve treinar e se mover em direção ao padrão de entrada para representá-lo. Sucessivos passos de adaptação seguindo a Equação (3.33) minimizam a distância Euclidiana entre o padrão de entrada e a informação contida no nodo, e por isso diminuem também o erro quando computado levando em conta esta distância. Entretanto, as informações heterogêneas contidas no nodo são modificadas de forma linear, e o fato de se ter um erro menor não corresponde mais à qualidade da informação armazenada.

Nas condições 1 e 2 da Tabela (3.1), como a atividade do nodo vencedor está alta, o nodo está representando bem o padrão e a adaptação que lhe é feita para se ajustar ao novo dado de entrada é pequena e provavelmente se relaciona a minimizar efeitos de ruídos, portanto não acarretam grandes distorções.

Como nos modelos anteriores mostrados, a inserção dos nodos é feita para minimizar o erro global do mapa topológico, ao utilizar médias e mover os nodos pelo mapa indiscriminadamente ao adaptá-los. A preocupação está na representação do mapa, e não na representação dos estados e conexões do sistema a ser identificado.

O método de remoção das conexões do GWR considera a utilização da conexão durante o treinamento. Uma conexão é considerada *utilizada* quando um dado padrão de entrada ativa o nodos extremos da mesma, um dos nodos como o vencedor do cálculo de proximidade e o outro nodo como segundo colocado. Conexões não utilizadas por um limite definido são removidas. A utilização das conexões tem o termo nomeado como *idade* em sentido figurado,

e sua pouca utilização ocasiona o *envelhecimento e morte*. Como visto no GNG, o uso de idade para a remoção de conexões pode acarretar no esquecimento de estados do sistema que foram treinados, informação valiosa para o mapeamento do espaço de estados.

O modelo SOM tem uma geometria pré-determinada nas suas conexões. O modelo TRN cria conexões, com o algoritmo CHL, baseadas em similaridades dos nodos após a distribuição destes pelo espaço de entrada feita pelo NG. O modelo GCS mantém uma malha de hipertetraedros formada por suas conexões, enquanto que no modelo GNG e GWR as conexões são simplesmente indicadoras de vizinhanças, que podem desaparecer ou reaparecer conforme a utilização dos nodos por elas conectados. Uma vez que as informações representadas nos nodos podem ser heterogêneas e agrupar critérios para a geração de trajetórias, faz-se necessária uma interpretação mais completa do papel das conexões no espaço de estados.

O modelo GWR não reduz a dimensão do espaço de estados, e consegue representá-lo com fidelidade em diferentes dimensões. Não apresenta defeitos topológicos para áreas desconexas ou côncavas, e consegue se adaptar rapidamente, de modo fractal. São características interessantes para um modelo para geração de trajetórias de estados, tendo-se em conta as limitações expostas nesta seção.

3.7 Conclusões

Este capítulo tratou dos modelos mais relevantes da abordagem tencionada para a solução de problemas de geração de trajetórias de estado. Os modelos VITE, SOM, TRN, GCS, GNG e GWR foram apresentados, seus algoritmos descritos e discutidos, suas características e limitações foram analisadas.

Considerando esses diversos modelos apresentados, as suas propriedades, o modo como tratam inserções e remoções de nodos e conexões, a precisão e adequação das representação do espaço de estados, e as suas limitações quando se tem em mente a geração de trajetórias, no Capítulo 4 um novo modelo será proposto. O modelo é chamado STRAGEN (*State Trajectory Generator*) e objetiva conciliar as diversas necessidades e particularidades vistas até este ponto do trabalho.

Proposta de Solução

Algumas coisas despertam, desafiam a inteligência; outras não. As que entram pelos sentidos em companhia de seus contrários são as que desafiam; as outras, não.

—PLATÃO (A república)

Este capítulo visa delinear os requisitos para possíveis soluções para o problema de planejamento e controle de trajetórias em um espaço de estados. Na Seção 4.1 se recapitula brevemente as propriedades encontradas na literatura para a solução de problemas similares, como também é exposto o modo pelo qual estas abordagens podem ser modificadas para se adequarem ao problema proposto nesta tese. Entre estas abordagens, destacam-se a família de modelos VITE e a família de modelos SOM com topologia dinâmica (SOM-DT), e apresenta as principais características que estarão presentes no modelo a ser proposto. A Seção 4.2 apresenta os modelos propostos, e mais especificamente a Subseção 4.2.4 apresenta o STRAGEN-OFF, a primeira versão deste novo modelo, com poda *off-line*. Uma discussão sobre as diferenças do modelo proposto para outros da literatura é feita na Subseção 4.2.5. A Subseção 4.2.6 apresenta uma nova versão do modelo, o STRAGEN-ON, mais rápida, em que a poda se faz de modo *on-line*, além de outras propriedades discutidas na Subseção 4.2.7.

4.1 Requisitos do Modelo Proposto

Foi visto nos capítulos anteriores que existem diversas abordagens para o problema de planejamento e controle de sistemas não-lineares. A abordagem adotada nesta tese, de *Métodos de Controle Inverso Direto*, com sistema de controle de malha aberta, permite criar uma única RNA capaz de fazer a identificação e o controle de sistemas dinâmicos não-lineares, Figura (2.3), p. 18. Os algoritmos baseados em SOM são capazes de fazer a identificação de sistemas não-lineares, através do mapeamento dos estados, e o controle quando seus vetores contém também informações de sinais de controle para cada estado. O planejamento deve ser incorporado ao modelo como uma propriedade adicional à identificação e ao controle, pois

determina a sequência de estados para a solução de uma tarefa, cada qual contendo seus respectivos sinais de controle.

A geração de trajetórias de estados é entendida como a realização do planejamento. As trajetórias permitem o deslocamento do sistema por um conjunto de estados, saindo de um estado inicial e atingindo um estado final, tendo como intermediários os estados que caracterizam um critério de movimento, otimizando uma ou mais medidas do sistema.

Estes requisitos para planejamento e controle demandam uma nova interpretação do conceito de estruturação de informações heterogêneas em um nodo da RNA. O modelo deve não só ser capaz de representar um sistema dinâmico, mas identificá-lo mantendo acesso às diversas informações de distintas variáveis do sistema, cuidando para que durante o seu processo de treinamento e adaptação estas informações heterogêneas não se deterioresem pela aplicação de gradientes ou médias, de modo indiscriminado. A preservação destas informações é essencial para o correto aprendizado de sistemas complexos, além de permitir a qualquer tempo que novos grupos de informação possam ser adicionados à descrição dos estados do sistema sempre que ele revele novos comportamentos.

A compartimentalização de informações heterogêneas nos nodos, além da proteção contra deformações durante o aprendizado da rede, traz duas outras importantes características ao modelo. A primeira delas é a capacidade de utilizar isoladamente os grupos de informações homogêneas para determinar as transições no mapa de estados, durante a geração de trajetórias. A segunda importante característica é dar significado ao papel das conexões na formação topológica do mapa, que passa a ser não só um mapa que representa o espaço de estados, mas também representa as transições entre estados similares de acordo com o critério estabelecido. O mapa topológico gerado a partir de critérios determinados passa a ser um grafo que representa a otimização destes critérios. Desta forma uma trajetória mínima entre estados distintos projetados em um mapa topológico com estas características é também a minimização deste critério para a solução do problema de planejamento entre os estados desejados.

Uma das mais marcantes qualidades do modelo VITE é o uso de Redes Neurais Artificiais Auto-organizáveis (RNA-AO) para explicar o fenômeno de geração de trajetórias como um processo emergente das relações entre módulos neurais distintos, sem o seu pré-planejamento. Porém estes módulos neurais que compõem o VITE são redes de estrutura rígida, compostas em blocos inspirados em áreas definidas no córtex dos primatas, Figura (3.1), p. 42. O modelo VITE trabalha apenas com trajetórias de braços robóticos, sem flexibilidade para generalizar a geração de trajetórias em um espaço de estados dado. É um modelo complexo, concebido para levantar hipóteses sobre o funcionamento do cérebro a partir de suas simulações.

As três dimensões do grau de propriedades fisiológicas de um modelo (Bullock et al., 1999)

tentam delinear a preocupação do modelo com explicações biologicamente plausíveis (p. 42). Dos modelos levantados neste trabalho, o VITE está nitidamente em vantagem nos três quesitos apontados por [Bullock et al. \(1999\)](#). Entretanto, o quesito (ii), i.é., a estrutura do modelo baseado na modelagem do córtex para explicar trajetórias de braços robóticos com propriedades semelhantes às trajetórias de braços humanos, por ser fundamental no projeto do VITE, impede este modelo de ser utilizado para realizar tarefas de domínios diferentes do que foi preparado.

O modelo VITE pretende explicar as áreas do cérebro tais quais são definidas na neurologia, para a realização de dada tarefa, e por isso seus módulos são fixos em determinada estrutura. Isto feito, além de tornar o modelo adequado para um domínio específico, a saber, o movimento de um braço, ainda cria uma gama de parâmetros desconhecidos que devem ser ajustados. Um modelo de topologia mais flexível eliminaria tais problemas, primeiro por poder se adequar mais facilmente a outros domínios, segundo por requerer menos parâmetros de ajustes.

Pode-se argumentar que os modelos de topologia dinâmica não são corroborados com dados neurológicos na mesma intensidade que o modelo VITE, uma vez que o agrupamento de nodos deste é baseado no agrupamento de neurônios das áreas do cortex e suas respectivas funções, enquanto que aqueles possuem um único agrupamento de nodos que se desenvolve, e cuja inclusão e eliminação de nodos ocorrem em uma velocidade muito superior àquela apresentada pelo cérebro para criar ou repor os caminhos neurais ou os neurônios porventura perdidos. Este é um argumento oposto ao novo modelo, mas que é contraposto a um resultado esperado que considera modelos mais autônomos, mais simples, necessitando de menos parâmetros pré-definidos externamente, que podem evoluir para dois tipos de trabalhos futuros: a tentativa de ganhos de performance cada vez maior, aproveitando-se das características das máquinas computacionais; ou a evolução do modelo para a inclusão de propriedades importantes na explicação do funcionamento cerebral. Há uma preocupação com a plausibilidade biológica neste trabalho, mas no sentido exposto por [Kohonen \(1987\)](#), porque redes com topologias dinâmicas não se assemelham a grupos neurais encontrados no córtex (veja na p. 52 as propriedades sobre plausibilidade biológica), pois têm a estrutura de um mapa auto-organizável cujo tamanho e topologia variam para representar os dados de entrada. Neste caso, a busca por explicações sobre como sistemas dinâmicos vivos resolvem problemas deve ser procurada no funcionamento e nas propriedades do modelo, sendo a estrutura topológica uma representação criada para metaforizar este processo de solução de problemas, uma abstração das entradas e saídas das conexões.

O modelo proposto por este trabalho deve ser simples e eficiente, tornando a escolha do controle de malha aberta uma opção natural. A escolha do controle de malha aberta pela sua simplicidade, entretanto, não deve limitar a capacidade da rede de se adaptar a diferentes situ-

ações dinâmicas. Porém o processo de realimentação passa a ser mais abrangente, a rede deixa de utilizar medidas de erro para corrigir trajetórias específicas e passa a receber informações gerais, diretamente dos sensores, juntamente com objetivos e ações desejadas, para atuar no mundo, corrigir trajetórias ou modificar objetivos (Figura (2.2), p. 15).

Consegue-se com certa facilidade no modelo VITE, diferentemente de outros propostos, alterar bruscamente o alvo desejado de uma trajetória durante o percurso já iniciado. A trajetória desenvolvida pelo braço assemelha-se às trajetórias desenvolvidas por seres humanos (Bullock et al., 1999) em situações análogas. A velocidade do desenvolvimento da trajetória, entretanto deve ser pelo menos parcialmente reduzida para se conseguir o efeito adequado em casos de mudanças bruscas de direção (Fiala, 1994; Schaal & Sternad, 1998).

Os parâmetros para o ajuste das simulações são considerados outro fator de grande importância na simplicidade de um modelo. Parâmetros com decaimento, dependentes do domínio do problema, tornam a tarefa de configuração da rede em um teste empírico de difícil solução. Existem estudos para resolver o problema de ajuste fino de parâmetros da rede através da automatização do processo de tentativa e erro utilizando algoritmos genéticos (Benante et al., 2004), porém modelos que tenham poucos ou nenhum parâmetro a ser ajustado, e que sejam robustos de modo a não se tornarem sensíveis a estes parâmetros são preferíveis.

No campo da robótica, o modelo VITE consegue com sucesso simular a compensação de velocidades entre as juntas, também chamada de *equifinalidade temporal* (Ulloa & Bullock, 2003; Sanguineti & Morasso, 1992; Bullock & Grossberg, 1988a), para que o conjunto das variáveis de estado do modelo façam sua transição para o estado final de modo sincronizado. Esta característica fica nítida no movimento de um braço humano, quando todas as juntas terminam seu movimento ao mesmo tempo ao atingirem o alvo, não importando a amplitude do movimento (Figura (3.2), p. 44). Esta é uma característica importante, desejada para o modelo proposto. No entanto, como o modelo proposto não se restringe à robótica, nem sempre será de interesse cumprir a equifinalidade temporal, por isto é importante deixar a cargo do processo de otimização a escolha do melhor trajeto, ou permitir algum tipo de direcionamento para a escolha da trajetória. Este direcionamento pode ser implementado com a utilização de trajetórias *via-pontos*. Neste caso, a rede deve ser capaz de receber como entrada não apenas os estados inicial e final desejados para a trajetória, como também pontos intermediários pelos quais a trajetória deve obrigatoriamente passar.

Outra característica importante do modelo VITE é a atuação do sinal GO, que possibilita controlar a velocidade de movimento do braço, através da multiplicação do sinal provindo de DV com um fator multiplicativo, mantendo as características da curva de velocidade para cada junta. Também se pode simular no modelo EVITE o braço ser movimentado externamente por

alguma força desconhecida. Neste caso é necessário considerar outros grupos de sinais aferentes do córtex, que corrigem a representação interna do braço para a real posição no espaço, com compensação de cargas e de forças inerciais durante o movimento. O modelo VITE é realimentado por estes sinais aferentes, em um controle de malha fechada. O modelo proposto, de malha aberta, ganha em simplicidade sob pena de perder a característica de correção automática de trajetórias. No entanto, mesmo em malha aberta, a reapresentação de valores dos sensores ou objetivos para o modelo permite que este faça o replanejamento necessário para corrigir ou modificar a trajetória e se adequar ao novo estado do sistema.

O modelo VITE assume que as coordenadas do sistema de resposta (cartesiano, por exemplo) e do sistema de coordenadas angulares das juntas sejam dimensionalmente consistentes e estejam calibrados. Esta não é uma hipótese coerente com as observações neurológicas, pois acredita-se que tais transformações são aprendidas na fase infantil de uma criança, quando esta aprimora os seus movimentos. O modelo AVITE (Gaudiano & Grossberg, 1992) explica como representações corretas do sistema de respostas podem ser aprendidas durante uma fase de balbuciamiento motor. O MB define comandos aleatórios de movimentos de braço, endogenicamente gerados por um módulo chamado ERG (*Endogenous Random Generator*), que movem o braço para posições diversas em uma ampla faixa de teste, e após marcar estas posições no sistema de resposta, realiza-se um ajuste no mapeamento entre os dois sistemas considerados (Sanguineti & Morasso, 1992). Deste modo, o MB é de especial interesse durante a identificação do sistema, calibrando suas entradas, saídas e sinais de controle.

Os grupos celulares que compõem o VITE, apesar de auto-organizados, possuem estrutura fixa, que não simula a plasticidade cerebral, quando áreas cerebrais se especializam para substituir áreas danificadas ou perdidas. Também não podem com isso apresentar uma rede mínima para a solução do problema proposto, tal como ocorre com os modelos de topologia dinâmica. Uma vez o modelo VITE em funcionamento, os grupos celulares comunicam-se entre si, gerando reativamente as respostas, mas tais grupos jamais se reestruturam para se adequar aos dados, nem têm conexões ou nodos criados ou eliminados, e com isso mantém-se a capacidade de representação da rede, o que pode ocasionar três inconvenientes já sabidos: se a capacidade de representação for menor que a necessária, o modelo atua com erro elevado; se a capacidade for adequada, o modelo serve bem ao problema específico, mas pode não servir para outros problemas semelhantes, e há necessidade de um programador estipular essa capacidade, tarefa não-trivial; se a capacidade for superior à necessária, o modelo atua com precisão, porém realiza mais cálculos que o necessário acarretando maior custo computacional e provável lentidão de resposta. Esses inconvenientes podem ser resolvidos com Mapas Auto-organizáveis de Topologias Dinâmicas (SOM-DT).

Entre os modelos de topologia dinâmica apresentados, tem-se particular interesse por suas características, os modelos GNG (Fritzke, 1997a) e GWR (Marsland et al., 2002). Ambos modelos têm topologia variável, não limitada a um número fixo de nodos, que permite o seu crescimento ou diminuição conforme algum critério.

No GNG, o critério para o crescimento, que ocorre em uma taxa fixa de passos do algoritmo, é o acúmulo de erros na região de um dos nodos, na qual o novo nodo será inserido. Uma diferença entre o GNG e o GWR consiste neste critério de adição de nodos. No GWR, um nodo é acrescentado à rede sempre que for necessário. Esta necessidade é calculada por uma taxa de atividade do nodo vencedor (que responde a um padrão de entrada em questão) e sua frequência de disparos, optando-se por inserir um novo nodo, caso em que o vencedor foi pouco ativo e está com alta taxa de disparos, ou treinar o nodo vencedor já existente, caso contrário.

Uma característica muito positiva e importante desta classe de RNA é o melhor desempenho devido à capacidade de ajuste automático do número de nodos e conexões necessárias para resolver, com precisão adequada, um determinado problema. Pode-se citar também que os SOM-DT passam a ter um item complexo eliminado: a determinação do número de nodos e topologia adequados para a solução de um problema. Além disto, os SOM-DT apresentam outras propriedades que podem ser boas para o problema levantado neste trabalho: a capacidade de mapeamento de espaços de entrada de dimensão variável complexos com baixo custo computacional, e a recuperação da informação mapeada de modo rápido e preciso; e sua grande capacidade de adaptabilidade a espaços em mudança.

Entretanto, o GWR original não é adequado para a geração de trajetórias de estados, apesar de conseguir mapear os dados de entrada, como já visto. O mapeamento dinâmico do GWR é realizado baseando-se apenas na informação dos vetores de entrada e cálculos de nodos vencedores por distância Euclidiana, e portanto não traz consigo nenhum critério que possa melhorar o mapeamento visando a sua utilização posterior para a geração de trajetórias de estados. Além disto, o GWR faz apenas o mapeamento dos dados de entrada, e não é preparado para ter um gerador de trajetórias acoplado. Construir um modelo SOM com topologia dinâmica incorporando algumas importantes propriedades apresentadas pela família VITE para a geração, planejamento e controle de trajetórias de estados é uma proposta inédita para solucionar o problema apresentado (outros modelos como Zeller et al. (1997) e Sanguineti & Morasso (1992) utilizaram redes de topologia estática, e Glasius et al. (1995) baseou-se na rede de Hopfield).

Os SOM-DT caracterizam-se por criar conexões entre os nodos, conforme estes vão sendo inseridos na rede, considerando a proximidade entre os nodos vencedores. A literatura considerada sugere que o papel das conexões está relacionado a dois fatores fundamentais: (i) Vizinhança topológica, determinando quais nodos têm características semelhantes, que são acentu-

adas durante o aprendizado do mapa topológico; **(ii)** Manutenção do nodo pelo seu número de conexões, uma vez que nestas redes em geral, um nodo que perde todas as conexões é excluído da RNA, pois a ausência de vizinhos decorre da perda de capacidade de responder a estímulos.

Ao propor um modelo de SOM-DT para a classe de problemas desta tese uma adição essencial deve ser feita na reinterpretação do papel das conexões. Enquanto que no mapeamento de superfícies, as conexões normalmente representam vizinhanças topológicas espaciais, no mapeamento de uma trajetória de estados isto pode não ser necessariamente verdade. Pode-se pensar nas conexões como representando transições de estados possíveis para rotas que minimizam algum critério, atribuindo-se às conexões papel mais ativo que nas redes auto-organizáveis anteriormente apresentadas. Este papel é justamente representar as alternativas de transições de estados na direção do nodo (estado) vizinho, seguindo critérios estabelecidos para o desempenho desejado. Em movimento de um manipulador robótico, por exemplo, tais critérios podem ser a minimização da variação dos torques, a minimização da variação de ângulos, ou a minimização do solavanco.

Considerando-se este novo papel atribuído às conexões, sua destruição não pode mais ser ditada pela idade, ou pelo uso, como fazem muitas redes (veja GWR). Os nodos treinados, e suas conexões, representam informações do espaço de estados de um sistema dinâmico, muitas vezes valiosas e de difícil captação. Assim, eliminar os nodos e conexões deve ser uma operação feita apenas em último caso, como por exemplo nodos que ficaram isolados devido à modificação da estrutura das conexões durante o aprendizado. O modelo baseado em TRN de [Zeller et al. \(1997\)](#) elimina as conexões dos nodos que se encontram inativos devido à presença de obstáculos, criando nodos isolados que se não forem eliminados apresentam o problema de se ter que reestabelecer as conexões através do CHL. Uma opção para manter estas informações treinadas é desabilitar temporariamente as conexões que cruzam com obstáculos detectados, reintegrando-as ao rol das conexões/nodos ativos quando o obstáculo for removido, possibilidade não aventada na literatura consultada. Com isto, se preserva uma importante informação por um tempo maior, sem limitar a capacidade de adaptação da rede.

Uma vez entendido o papel das conexões no modelo, determinadoras das possibilidades de mudança no espaço de estados, pode-se inferir que o modelo SOM-DT terá agora que aprender não somente os estados válidos do sistema (nodos), mas também as suas transições. Este é um diferencial importante do novo modelo a ser proposto em comparação com o modo de treinamento das redes SOM em geral, onde os pontos podem ser apresentados ao acaso e as conexões são criadas por similaridade espacial. Da forma como são apresentados os pontos para treinamento, as redes SOM-DT sofrem de uma limitação análoga à *Subida de Encosta Estrutural* ([Angeline et al., 1994](#)), que consiste em criar uma topologia em uma direção estabelecida por

regras de aprendizado de modo a otimizar gradualmente o critério estabelecido, normalmente a menor distância Euclidiana entre as posições dos nodos. Faz-se necessário um novo modo de capturar conhecimento a partir dos padrões de treinamento, para que considere não somente os estados válidos, mas também as conexões entre estes estados incorporando a ordem que eles se manifestarão para atingir uma meta, e informações de similaridades refinadas por critérios que definem características do espaço de estados que se pretende otimizar.

Neste ponto o modelo a ser proposto recorre ao sistema de aprendizado de balbuciamiento motor (MB) (Gaudiano & Grossberg, 1992; Sanguineti & Morasso, 1992) apresentado na Seção (3.1). No modelo AVITE, a fase de MB é realizada para calibrar o módulo TPC (*target position command*) com relação ao módulo PPC (*present position command*), em outras palavras, calibrar o sistema de coordenadas em que se representam os objetos e os alvos desejados e o sistema de coordenadas no qual o próprio posicionamento do manipulador é estipulado. Sanguineti & Morasso (1992) colocam o MB de modo mais geral, como uma fase em que o modelo aprende as dependências não-lineares entre seus estados e suas saídas. Portanto, o aprendizado inicial realizado pelo modelo a ser proposto deverá formar trajetórias a partir de estados válidos, guiados aleatoriamente, de modo que não só os pontos sejam aprendidos, mas também as conexões que os ligam. Nesta fase, o manipulador deverá explorar seu espaço de trabalho, para mapear corretamente estados permitidos e obstáculos estáticos presentes.

Uma vez definidas as propriedades do mapeamento topológico, tendo em vista sua posterior utilização para a geração de trajetórias de estados, é necessário um método de planejamento que permita gerar a trajetória aproveitando-se das propriedades do mapa. O mapa topológico gerado pelo modelo proposto deve criar condições para que a navegação entre estados vizinhos otimize o critério de vizinhança estabelecido segundo o domínio em questão.

Zeller et al. (1997) comentam que quaisquer algoritmos de busca ou estratégias de pesquisa globais poderiam ser utilizados em seu modelo baseado no TRN. Porém o autor escolheu o algoritmo de busca baseado em difusão de energia devido a sua ordem de complexidade e garantia de resultado. Neste trabalho duas abordagens serão testadas: a geração de trajetórias através de uma função de difusão de energia (Zeller et al., 1997) e a utilização do algoritmo de menor caminho (Dijkstra, 1959). Ambos algoritmos tem ordem $O(n^2)$, onde n é o número de nodos a se visitar, garantem encontrar o menor caminho no menor custo, se existir, e não têm problemas de se prenderem em mínimos locais. No entanto o uso de árvores binárias permite melhorar a ordem de complexidade do algoritmo Dijkstra em até $O(m + n \log(n))$ (Barbehenn, 1998), onde m é o número de vértices e n o número de nodos, o que pode mostrar um ganho no custo computacional do modelo para seu uso em situações *on-line*.

As trajetórias formadas durante a navegação pelos nodos e conexões do sistema são pro-

priedades emergentes da dinâmica da rede, que representam o espaço topológico identificado segundo critérios estabelecidos. Determinar o nodo vencedor que melhor caracteriza o estado inicial da trajetória deve levar em conta a distância Euclidiana do vetor que configura este estado com os nodos que a rede aprendeu, como é já conhecido nas redes SOM. Entretanto, diferentemente do SOM, não é o vetor todo que deve ser considerado, mas apenas parte deste vetor. O vetor de pesos que caracteriza o nodo da rede passa a ser um conjunto heterogêneo de informações acerca dos estados do sistema, e a determinação de subgrupos homogêneos dentro deste vetor permite ao modelo organizar-se topologicamente em função de características marcantes que se deseja otimizar. Estes subgrupos que organizam as informações do vetor de pesos são divididos em duas funções distintas: os chamados de critérios de vizinhança, são utilizados para determinar conexões na topologia da rede, e os critérios de atividade são utilizados para determinar a atividade do nodo vencedor em relação a um determinado padrão, embora normalmente se deseje que ambos sejam o mesmo.

A criação e remoção de nodos também deve ser reestruturada visando o melhor desempenho da rede na sua utilização em trajetórias de estados. Na criação dos nodos, deve-se compreender que cada nodo representa um estado possível do sistema, e portanto sua posição espacial não pode ser significativamente modificada por uma regra de aprendizado. Nos casos em que os nodos representam grandezas cujos valores podem ser interpolados linearmente, a movimentação de um nodo no espaço pode ser realizada suavemente sem a geração de erros de grande ordem. Esta não é a realidade para nodos que representam estados de sistemas não-lineares que contêm diversos tipos diferentes de informações, organizadas em grupos homogêneos entre si, porém formando um todo heterogêneo. Como exemplo, para um manipulador robótico, suponha um nodo que represente as posições espaciais das juntas, os seus ângulos e torques. Uma pequena modificação na posição (X, Y, Z) de um efetuador pode ocasionar modificações diversas, pequenas para alguns ângulos e torques, e grandes para outros, ou mesmo totalmente diferentes na configuração do sistema como um todo.

Tendo em vista a criação de um modelo de topologia dinâmica para a geração de trajetórias de estado, e levando-se em conta os requisitos apresentados neste trabalho, em especial nesta seção, a seguir está explanada uma primeira solução, o modelo Gerador de Trajetórias de Estados (*State Trajectory Generator*, STRAGEN) *off-line*, seguido de uma discussão das propriedades por ele alcançadas.

4.2 Descrição do Modelo

Esta seção irá descrever dois modelos de redes neurais artificiais com topologia dinâmica para a solução de geração de trajetórias de estados para sistemas não-lineares. O primeiro modelo, STRAGEN *off-line*, possui uma fase de poda de conexões que deve ser realizada após o treinamento, enquanto o STRAGEN *on-line* realiza a poda durante o treinamento. Ambos modelos possuem em comum definições de variáveis, o procedimento de Balbuciamiento Motor, e o cálculo do erro de validação, que portanto serão apresentadas a seguir.

Antes de iniciar a fase de treinamento, uma fase de pré-processamento é necessária para normalizar a base de dados $\mathbf{B}^{(0)}$ de dimensões $L \times D$, onde L é o número de padrões (amostras ou linhas) e D é a dimensão de cada padrão i , tal que $1 \leq i \leq L$, e seu vetor peso é de dimensão $\mathbf{w}_i \in \mathbb{R}^D$. Durante esta fase de pré-processamento, os valores máximos e mínimos da matriz $\mathbf{B}^{(0)}$ são usados para calcular a nova matriz $\mathbf{B}$ normalizada, tal que $w_{ij} \in \mathbf{B} \Rightarrow 0 \leq w_{ij} \leq 1$:

$$w_{ij} = \frac{b_{ij} - \min_k (b_{kj})}{\max_k (b_{kj}) - \min_k (b_{kj})} \quad (4.1)$$

onde $1 \leq k \leq L$ e $b_{i,j} \in \mathbf{B}^{(0)}$. O vetor de pesos é definido como:

$$\mathbf{w}_i = [w_{i1} \ \dots \ w_{iD}]^T$$

O vetor de pesos $\mathbf{w}$ pode ser dividido em grupos de informações de mesma natureza, permitindo o STRAGEN lidar com elementos de domínios distintos, cada um com diferentes ordens de magnitude. Esses grupos podem ser usados como um critério para formar possíveis vizinhanças:

$$\mathbf{w}_i = [\mathbf{V}_1 \ \mathbf{V}_2 \ \dots \ \mathbf{V}_m]^T \quad (4.2)$$

onde cada vetor $\mathbf{V}_k$, $k = 1, \dots, m$ representa um conjunto de variáveis de um domínio que pode ser agrupado e operado como um todo. A dimensão de cada grupo é $\mathbf{V}_k \in \mathbb{R}^{D_k}$ e $\sum_{k=1}^m D_k = D$. Para clarificar o conceito, em um domínio robótico, por exemplo, os grupos podem ser as posições espaciais, juntamente com ângulos e torques das juntas. E um grupo, por exemplo das posições espaciais das juntas, pode ser dividido em *subgrupos* que são vetores que representam a posição espacial de cada junta individualmente.

É preciso também definir dois critérios a serem usados como guia para formar a vizinhança topológica da rede e a detecção de similaridades entre nodos. O *Critério de Atividade*, utilizado

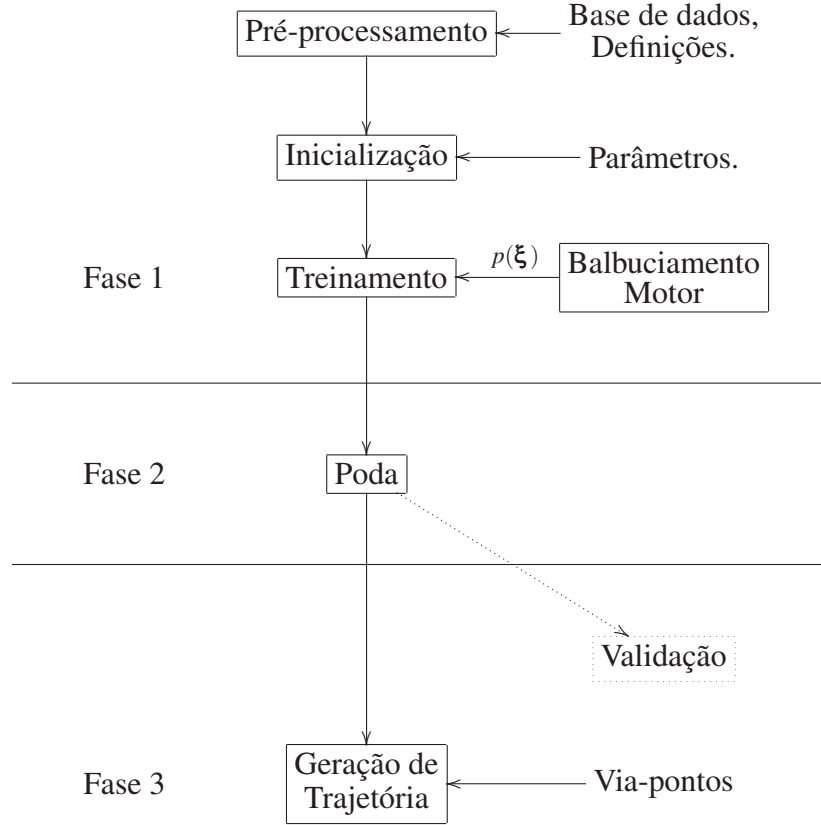


Figura 4.1 Fluxograma: STRAGEN-OFF

para calcular a atividade dos nodos da rede para um dado ponto de entrada, é definido pelo grupo $\mathbf{V}_\zeta$ e seus *subgrupos*. É o critério de atividade que determina a necessidade de inserção de um novo nodo. Destes m diferentes grupos seleciona-se um, chamado ζ , para ser grupo de atividade $1 \leq \zeta \leq m$. O segundo critério é o *Critério de Vizinhaça* η , um grupo escolhido $1 \leq \eta \leq m$, que é usado para avaliar a proximidade de um estímulo e sua representação no mapa topológico, e outras avaliações de proximidade.

O limiar de atividade $\bar{\mathbf{a}} = [a_1 \dots a_1]^T$ para o grupo $\mathbf{V}_\zeta$, é definido por percentual escolhido, $0 < P < 1$, da distância Euclidiana máxima na dimensão D_{ζ_k} , para cada subgrupo k de $\mathbf{V}_\zeta$ que representa uma informação independente, ou seja:

$$\bar{a}_k = \exp \left(P \cdot \sqrt{D_{\zeta_k}} \right), \quad 1 \leq k \leq 1 \quad (4.3)$$

onde 1 é o número de subgrupos homogêneos que compõe $\mathbf{V}_\zeta$.

A Figura (4.1) mostra as diversas fases do STRAGEN-OFF em um diagrama de fluxo. Após o pré-processamento, o procedimento MB é definido para escolher as amostras de dados

a serem usadas durante a fase de treinamento, conforme descrito a seguir.

4.2.1 Procedimento do Balbuciamiento Motor

Originalmente, MB foi usado para criar um mapa de posições cartesianas para unir coordenadas de movimento (Fiala, 1994). O MB trabalha para gerar movimentos endógenos através de trajetórias de estados e aprendizado de pontos isolados no mapa de representação de um sistema de coordenadas para outro. Enquanto outros modelos (Gaudiano & Grossberg, 1991, 1992; Fritzke, 1997a; Marsland et al., 2002) focam nas posições dos nodos, o objetivo aqui é usar MB para ajudar o modelo a criar conexões entre nodos que tenham significado no processo de geração de trajetórias. O STRAGEN usa o MB para construir trajetórias enquanto ele recebe os pontos isolados como estímulos. Sabe-se que redes baseadas em SOM incremental sofrem de um tipo de *Structural Hill-climbing* (Angeline et al., 1994), i.e., elas são sensíveis à ordem na qual os dados são apresentados. O STRAGEN, do modo que define o MB, aproveita-se desta propriedade e apresenta os dados numa ordem conveniente que pode ajudar o algoritmo a criar nodos e conexões mais factíveis em termos de vizinhança, acelerando o treinamento.

Para cada iteração t , quando uma nova amostra é requerida para a fase de treinamento, o procedimento MB $p(\xi)$, para entradas ξ retorna uma nova amostra conforme definido a seguir. Na primeira execução, o MB apenas retorna algum padrão aleatório ξ da base de dados. A partir da segunda execução em diante, para cada nova amostra requerida pela rede, o MB escolhe Q candidatos aleatórios ainda não apresentados, e para estes Q candidatos o MB retorna o mais próximo à última entrada apresentada ξ , de acordo com algum critério η . Desta forma o treinamento é dirigido aleatoriamente dentro de uma vizinhança. O procedimento MB é formalizado como:

1. Se $t = 1$, escolha um padrão aleatório $\xi(t)$ da base de dados $\mathbf{B}$ de dimensões $L \times D$, e remova esta amostra para não permitir repetições.
2. Se $t > 1$, escolha com probabilidade uniforme Q candidatos para ser o próximo padrão Φ_i , $i = 1, \dots, Q$ e escolha

$$\xi(t) = \arg \min_{\forall i} \{\text{Dist}(\xi(t-1), \Phi_i)\} \quad (4.4)$$

Remova $\xi(t)$ da base de dados para não permitir repetições.

Este procedimento atua na fase de treinamento. Uma vez a rede treinada, opcionalmente pode-se calcular o erro de validação da topologia criada, conforme o procedimento a seguir.

4.2.2 Fase de Validação

No STRAGEN-OFF, a fase de validação ocorre após a fase de poda. No STRAGEN-ON, como não há fase de poda, uma vez que o modelo faz remoções de conexões durante o treinamento, a fase de validação pode ser executada logo após o treinamento. A fase de validação permite checar o erro de representação do espaço do problema, gerado pelo modelo. O procedimento é dado por:

1. Repita para $i = 1, \dots, I$ iterações:

- Gere um padrão de entrada ξ da base de dados de acordo com o procedimento MB $p(\xi)$.
- Calcule o erro de validação da iteração i , a distância entre a amostra e o nodo mais próximo, de acordo com o critério adotado:

$$e_i = \|\xi - \mathbf{w}_{s_1}\|$$

2. Retorne o erro de validação como:

$$E_v = \frac{\sum_i e_i}{I} \quad (4.5)$$

4.2.3 Inicialização do Algoritmo STRAGEN

Seja $\mathcal{A}$ um conjunto de nodos e $\mathcal{C}$ um conjunto de conexões entre estes nodos. Seja a distribuição de entrada $p(\xi)$, para entradas ξ como definido no procedimento MB. Seja a taxa de aprendizado ϵ_b . Seja a taxa final de aprendizado $\alpha_f \approx 0$.

Antes de ativar o algoritmo, inicialize o conjunto $\mathcal{A}$ com 2 nodos n_1 e n_2 localizados em $\mathbf{w}_{n_1}$ e $\mathbf{w}_{n_2}$, em $\mathbb{R}^D$, representando 2 padrões aleatórios da base de dados:

$$\mathcal{A} = \{n_1, n_2\},$$

Defina grupos de informações homogêneas no vetor de pesos tal que:

$$\mathbf{w}_i = [w_{i1} \dots w_{iD}]^T = [\mathbf{V}_1 \mathbf{V}_2 \dots \mathbf{V}_m]^T \quad (4.6)$$

E defina o critério de vizinhança η e o critério de atividade ζ tal que $1 \leq \eta, \zeta \leq m$, para algum grupo $\mathbf{V}_\eta$ e $\mathbf{V}_\zeta$.

Inicialize o conjunto de conexões $\mathcal{C}$ com uma conexão entre os dois primeiros nodos inseridos:

$$\mathcal{C} = \{c_{n_1, n_2}\}$$

Com estas definições, parâmetros e inicializações comuns aos modelos STRAGEN-OFF e STRAGEN-ON feitas, o algoritmo está pronto para ser treinado. A seguir, será apresentado o algoritmo do modelo STRAGEN-OFF.

4.2.4 Modelo STRAGEN Off-line

A primeira abordagem do modelo STRAGEN, para treinamento *off-line*, resultado deste trabalho, tem sua estrutura inicial tomada com base no modelo GWR. A descrição do algoritmo será semelhante a de Fritzke (1995) para facilitar comparações. As propriedades que este modelo apresenta são discutidas ao final, comparadas com outros modelos, e com a segunda versão do modelo proposto. As diferenças primordiais em relação aos outros modelos, em especial com relação ao modelo GWR são levantadas. O algoritmo completo é independente do domínio do problema, desde que se possa formalizar o problema em questão como uma configuração de estados com propriedades similares segundo critérios pré-estabelecidos, em um vetor pesos de acordo com a Equação (4.2).

O algoritmo é composto de três fases: treinamento, poda e geração de trajetórias de estados. Na Fase de Treinamento o modelo se adapta e cria uma representação da topologia do espaço de estados enquanto o procedimento de MB lê a base de dados. Na Fase de Poda, o modelo reavalia as conexões criadas durante o treinamento e elimina aquelas consideradas desnecessárias ou indesejadas. Na terceira fase, a de Geração de Trajetórias, o algoritmo encontra o melhor caminho entre dois pontos, de acordo com o algoritmo de difusão de energia (Zeller et al., 1997). Após a fase de treinamento, e antes da fase de geração de trajetórias, pode-se executar a Fase de Validação para medir o erro do mapa topológico criado pelo modelo.

4.2.4.1 Fase de Treinamento do STRAGEN-OFF

O algoritmo de treinamento para o modelo STRAGEN-OFF é mostrado a seguir:

1. Gere um novo ponto de entrada de ξ de acordo com o procedimento MB $p(\xi)$:

$$\xi = [\xi_1 \dots \xi_\zeta \dots \xi_\eta \dots \xi_m]^T \quad (4.7)$$

2. Para cada nodo i na rede, calcule a sua distância com o ponto de entrada $\|\xi_\eta - \mathbf{V}_{\eta,i}\|$, onde ξ_η é o subvetor da entrada ξ que representa o grupo η definido como critério de vizinhança, e $\mathbf{V}_{\eta,i}$ é o subvetor análogo no vetor de pesos $\mathbf{w}_i$. Determine a primeira e segunda unidades mais próximas, $s_1, s_2 \in \mathcal{A}$, usando o critério de vizinhança escolhido η , tal que:

$$\|\mathbf{V}_{\eta,s_1} - \xi_\eta\| \leq \|\mathbf{V}_{\eta,i} - \xi_\eta\|, \quad \forall i \in \mathcal{A} \quad (4.8)$$

$$\|\mathbf{V}_{\eta,s_2} - \xi_\eta\| \leq \|\mathbf{V}_{\eta,i} - \xi_\eta\|, \quad \forall i \in \mathcal{A} - \{s_1\} \quad (4.9)$$

3. Atualize o número de vitórias de s_1 : $\sigma_{s_1} = \sigma_{s_1} + 1$.
4. Insira uma nova conexão entre s_1 e s_2 em $\mathcal{C}$, se ainda não existir.

$$\mathcal{C} = \mathcal{C} \cup \{c_{s_1,s_2}\}$$

5. Calcule a atividade do estímulo ξ com respeito ao nodo *vencedor* s_1 , de acordo com o *Critério de Atividade* ζ . O grupo $\mathbf{V}_\zeta$ representa um conjunto de informações homogêneas que podem ser subdivididas em 1 subgrupos se necessário, assim a atividade $\mathbf{a} = [a_1 \dots a_1]^T$ pode ser calculada independentemente para cada subgrupo:

$$a_k = \exp\left(-\|\xi_{\zeta_k} - \mathbf{V}_{\zeta_k,s_1}\|\right), \quad 1 \leq k \leq 1 \quad (4.10)$$

6. Se *qualquer* atividade for menor que o limiar estabelecido para um subgrupo, ($a_1 < \bar{a}_1$ OU ... OU $a_1 < \bar{a}_1$), então um *novo* nodo s_3 deve ser inserido na localização exata do vetor de entrada:

- (a) Adicione o novo nodo s_3 ao conjunto $\mathcal{A}$:

$$\mathcal{A} = \mathcal{A} \cup \{s_3\}$$

- (b) Crie um novo vetor de pesos associado com o nodo s_3 , i.e., $\mathbf{w}_{s_3} = \xi$.

- (c) Remova a conexão c_{s_1,s_2} do conjunto $\mathcal{C}$.

- (d) Calcule as distâncias: $\mathcal{D} = \{\text{Dist}(s_3, s_1), \text{Dist}(s_3, s_2), \text{Dist}(s_1, s_2)\}$, onde:

$$\text{Dist}(s_3, s_1) = \|\mathbf{V}_{\eta,s_3} - \mathbf{V}_{\eta,s_1}\|,$$

$$\text{Dist}(s_3, s_2) = \|\mathbf{V}_{\eta, s_3} - \mathbf{V}_{\eta, s_2}\| \text{ e}$$

$$\text{Dist}(s_1, s_2) = \|\mathbf{V}_{\eta, s_1} - \mathbf{V}_{\eta, s_2}\|.$$

- (e) Selecione os 2 pares (s_i, s_j) e (s_i, s_k) de nodos com as menores distâncias calculadas, tais que:

$$\text{Dist}_1(s_i, s_j) = \arg \min(\mathcal{D})$$

$$\text{Dist}_2(s_i, s_k) = \arg \min(\mathcal{D} - \{\text{Dist}_1(s_i, s_j)\})$$

- (f) Insira 2 novas conexões entre os pares de nodos considerados para determinar $\text{Dist}_1(s_i, s_j)$ e $\text{Dist}_2(s_i, s_k)$.

$$C = C \cup \{c_{s_i, s_j}, c_{s_i, s_k}\} \quad (4.11)$$

7. Se um novo nodo não foi inserido no passo (6), atualize o vetor de pesos do nodo vencedor s_1 :

$$\Delta \mathbf{w}_{s_1} = \rho \times (\boldsymbol{\xi} - \mathbf{w}_{s_1}) \quad (4.12)$$

onde:

$$\rho = \begin{cases} \epsilon_b \times \alpha_f^{(\sigma/\sigma_f)} & \sigma \leq \sigma_f \\ \epsilon_b \times \alpha_f & \sigma > \sigma_f \end{cases} \quad (4.13)$$

e $0 < \epsilon_b < 1$ é a taxa de aprendizado; $\alpha_f \approx 0$ é a taxa de aprendizado final; σ é o contador do número de vezes que um nodo vencedor foi disparado, e σ_f é o número máximo de vezes que um nodo é suposto a disparar, sugerido como $\sigma_f = 2 \cdot t_{\max}/L$.

8. Repita do passo (1) até o número máximo de iterações $t_{\max}$, ou até que algum critério de parada seja atingido.

4.2.4.2 Fase de Poda do STRAGEN-OFF

Após a fase de treinamento do STRAGEN-OFF, executa-se uma fase de poda por I iterações, responsável por remover conexões mal-posicionadas e possíveis nodos isolados decorrentes desta remoção. A fase de poda do STRAGEN-OFF garante que não remove conexões que possam tornar o grafo desconectado.

1. Crie um conjunto $\mathcal{V} = \mathcal{A}$ de todos os nodos da rede, e um conjunto $\mathcal{E} = \mathcal{C}$ de todas as

conexões.

2. Repita para $i = 1, \dots, I$ iterações:

- Gere um sinal de entrada ξ , de acordo com o procedimento MB $p(\xi)$.
- Exclua o nodo vencedor s_1 do conjunto $\mathcal{V} = \mathcal{V} - \{s_1\}$ e exclua a conexão c_{s_1, s_2} entre o nodo vencedor e o segundo mais próximo do conjunto $\mathcal{C} = \mathcal{C} - \{c_{s_1, s_2}\}$.

3. Repita para todas conexões $c_{i,j}$ remanescentes em $\mathcal{C}$:

- Verifique se ao eliminar a conexão $c_{i,j}$ o grafo fica desconectado, i.e., não existe caminho alternativo do nodo i para o nodo j .
- Se o grafo ficar desconectado, não elimine a conexão $c_{i,j}$. Caso contrário, elimine a conexão $c_{i,j}$ permanentemente de $\mathcal{C} = \mathcal{C} - \{c_{i,j}\}$.

4. Elimine do conjunto $\mathcal{A}$ todos os nodos remanescentes no conjunto $\mathcal{V}$ (nodos que nunca venceram uma competição), juntamente com suas conexões.

5. Elimine nodos isolados do conjunto $\mathcal{A}$, i.e., nodos sem conexões em $\mathcal{C}$.

A Figura (4.2) mostra um exemplo de remoção de conexão. Na Subfigura (4.2) (a), a rede antes da remoção apresenta duas conexões candidatas a remoção. A conexão superior liga dois nodos que nunca serão, para nenhum padrão de entrada apresentado, vencedor e vice em conjunto. Após a remoção, na Subfigura (4.2) (b), o STRAGEN mantém o grafo conectado, i.e., mesmo que por hipótese a conexão central na subfigura (b) não ative simultaneamente seus nodos extremos como vencedor e vice, o STRAGEN-OFF não eliminaria a conexão, para manter o grafo conectado.

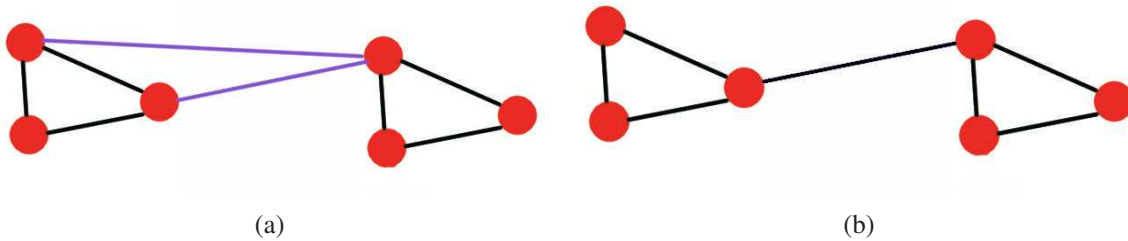


Figura 4.2 Exemplo de remoção de conexão no modelo STRAGEN-OFF (a) antes da remoção, e (b) depois da remoção.

4.2.4.3 Fase de Geração da Trajetória por Difusão de Energia

Uma possível abordagem para a geração de trajetórias de estados é a utilização de difusão de energia (Zeller et al., 1997). Estando a rede treinada, interessa fornecer como entrada duas ou mais posições quaisquer possíveis no espaço de estados, e receber como saída da rede a trajetória que passa por estas posições e os torques necessários para que o manipulador possa atingir cada ponto da trajetória. O algoritmo de difusão de energia é descrito a seguir.

Dados um ponto inicial ξ_{q_0} e um ponto final ξ_{q_f} , encontre o nodo vencedor n_{q_0} para o ponto inicial conforme Equação (4.8), e encontre o nodo vencedor para o ponto final, n_{q_f} , dado pela mesma Equação (4.8).

Determine uma função $f_{de}(t, n)$ de difusão de energia (Zeller et al., 1997) que define um fluxo difundindo-se por todos os nodos n da rede, com objetivo de encontrar uma seqüência de nodos $\mathcal{T}^{-1} = \{n_S, n_{S-1}, \dots, n_1, n_0\}$, iniciando o fluxo pelo ponto final $n_S = n_{q_f}$ em direção ao ponto inicial $n_0 = n_{q_0}$, onde S é o tamanho (desconhecido) da trajetória encontrada.

O procedimento para a difusão da energia é:

1. Inicialize a função de difusão $f_{de}(0, n) = 0, \forall n \neq n_{q_f}$ e $f(0, n_{q_f}) = 1$
2. Repita para todos $n \in \mathcal{A}$ até que $f_{de}(t_f, n_{q_0}) \neq 0$:

$$f_{de}(t+1, n) = \begin{cases} 1 & \forall t, \text{ se } n = n_{q_f} \\ \mu \sum_{j \in \mathcal{N}(n)} f(t, j) & \text{se } n \neq n_{q_f} \end{cases} \quad (4.14)$$

onde $\mathcal{N}(n)$ é o conjunto de todos nodos que são vizinhos de n e $|\mathcal{N}(n)|$ é sua cardinalidade, e $\mu = K/|\mathcal{N}(n)|$, $K < 1$ tal que $K = |\mathcal{N}(n)|/(|\mathcal{N}(n)| + 1)$, e t_f é o tempo final de convergência.

Este procedimento faz a difusão de energia desde o alvo, por todos os nodos, até que porventura atinja o nodo inicial (o que é garantido se houver uma rota possível (Zeller et al., 1997)). Assim que o nodo inicial receber qualquer quantidade de energia, o caminho está formado, e para definí-lo basta seguir a partir do nodo inicial sempre escolhendo-se para próximo nodo o vizinho de maior energia, até que se encontre o nodo alvo (único nodo com energia igual a um). A trajetória final é dada pelos nodos $\mathcal{T} = \{n_0, n_1, \dots, n_{S-1}, n_S\}$. O algoritmo de difusão de energia não leva em conta a distância entre os nodos, mas somente sua vizinhança.

4.2.5 Diferenças entre o STRAGEN-OFF e o GWR

O STRAGEN-OFF implementa uma série de conceitos que o torna viável para a geração de trajetórias, quando comparados com outros modelos discutidos neste trabalho, e em especial ao se comparar com o modelo GWR no qual o STRAGEN inspirou-se inicialmente como base para a sua estrutura geral. O GWR no formato que foi idealizado não resolve o problema de geração de trajetórias. Alguns pontos importantes que foram alterados nessa primeira versão do modelo proposto são discutidos aqui e mostram os avanços e limitações do STRAGEN.

A fase de pré-processamento do STRAGEN apresenta mudanças fundamentais para o tratamento de dados heterogêneos de sistemas não-lineares com vista a gerar trajetórias de estados. O primeiro ponto importante é a normalização da base de dados, feita para eliminar possíveis influências entre grupos de informações distintos para o cálculo da proximidade das vizinhanças. Outro fator importante e inovador é o agrupamento destas informações no nodo, para que sejam tratadas em separado, evitando distorções nas operações de adaptação, e permitindo a sua utilização como critérios de otimização do sistema identificado, implementados através de um mapeamento topológico guiado por tais critérios.

O treinamento do mapa topológico conta com a fase de MB, que permite ao modelo STRAGEN criar a representação do espaço de estados levando em conta estados similares em consonância com a medida de proximidade dada pelo critério de distância (vizinhança) estabelecido. Este procedimento de MB permite a criação de mapas com transições de estados coerentes desde o início do treinamento, pelo fornecimento, ainda que aleatoriamente guiado, de uma ordem na transição de estados.

O STRAGEN é robusto à variação de parâmetros, o que simplifica seu uso nos mais diversos domínios. Conforme resultados dos experimentos descritos no Capítulo 5, é possível usar sempre os valores pré-definidos, não se necessitando ajustar um único parâmetro. O GWR possui diversos parâmetros, entretanto apenas dois são mais importantes, ambos relacionados à inserção de nodos: o limite da atividade e limite de disparos. O STRAGEN não usa limite de disparos para o cálculo da inserção de nodos. Quanto ao limite de atividade, é constante e ajustado por uma porcentagem P , conforme a Equação (4.3), baseado no tamanho máximo da distância Euclidiana do espaço de estados normalizado, o que permite utilizar a mesma porcentagem independente do domínio. Testes foram feitos com $P = 1\%$ e para mapas mais precisos, com $P = 0,3\%$.

No Passo 4 da fase de treinamento, a conexão inserida representa a possível rota entre os nodos s_1 e s_2 , uma vez que são os dois nodos mais próximos do ponto ξ dado, segundo os critérios estabelecidos. O processo é o mesmo no STRAGEN e no GWR. Esta conexão pode,

a princípio criar uma rota impossível, por exemplo cruzando um obstáculo, ou criar conexões muito próximas e redundantes. Enquanto que no GWR é o processo de convergência do algoritmo que se encarrega de eliminar as conexões pela sua idade, o STRAGEN-OFF permite o treinamento da rede e elimina as conexões inadequadas apenas no final, através da fase de poda. A vantagem que se espera obter é que ao final do treinamento o mapa já está bem definido, e as conexões inativas estarão mais nítidas, enquanto que durante o processo de treinamento pode-se eliminar conexões importantes além de ser difícil ajustar o parâmetro de idade que determina esta remoção.

O modo como a inserção de novos nodos é feita no STRAGEN é uma importante característica deste modelo. Como se viu, na Tabela (3.1) p. 83, no algoritmo GWR um novo nodo é inserido se e somente se o novo vencedor estiver distante e atuante. Comparando-se com a Tabela (4.1) pode-se perceber que a política de inserção de nodos no STRAGEN é mais agressiva, inserindo um nodo sempre que a atividade do nodo vencedor não for considerada suficiente para *todos* os subgrupos que compõem o critério de ativação. É esta característica que permite, por exemplo no domínio da robótica, que o STRAGEN diferencie duas posições ambíguas em que o efetuador se encontra no mesmo ponto, porém as outras juntas estão em configurações distintas não sendo consideradas semelhantes. Para clarificar a Tabela (4.1), se for considerado um robô bidimensional, e o critério de ativação V_{ζ} como sendo a posição espacial das juntas, então V_{ζ_1} seria a posição da junta J_2 e V_{ζ_2} a posição do efetuador J_3 considerados individualmente.

O Passo 6 do algoritmo do STRAGEN-OFF mostra o processo de inserção de novos nodos. No GWR, como se viu, além da atividade do nodo, avalia-se a frequência de disparos, de modo que um novo nodo só será adicionado se o vencedor estiver longe do padrão e tiver sido usado com frequência. Quando uma destas duas características falha (atividade e frequência), ao invés de adicionar um novo nodo é feito o treinamento (deslocamento) do nodo vencedor em direção ao padrão de entrada. No modelo proposto, os nodos aprendidos devem permanecer fiéis aos pontos que representam, não devendo mover-se para evitar a distorção das informações sobre o estado aprendido. O processo de adaptação do STRAGEN, em conjunto com as condições de inserção de novo nodo, garantem que um nodo só irá se adaptar se for ativado por um padrão de entrada similar o suficiente para ser considerado um ruído do mesmo padrão.

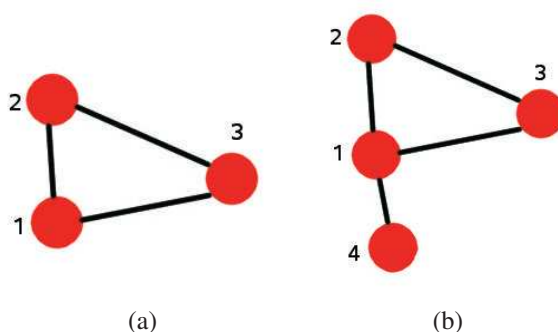
Os subitens do Passo 6 tratam da inserção de um novo nodo no modelo STRAGEN. Os modelos apresentados, incluído o GWR, normalmente inserem o novo nodo entre as posições dadas pelo nodo vencedor e pelo vice (no GWR, veja Equação (3.29), p. 78, no GNG a inserção está caracterizada no Passo 9, p. 73, e no GCS o processo de inserção está descrito na p. 66). Inserindo o novo nodo na exata posição do padrão ξ faz com que o algoritmo represente adequadamente o conjunto de informações dadas pelo padrão de entrada. Além disto, o algoritmo

Tabela 4.1 Condições para inserção de novo nodo no STRAGEN.

Cond.	Atividade				Insere
	V_{ζ_1}	V_{ζ_2}	...	V_{ζ_l}	
1	Alta	Alta	...	Alta	Não
2	Baixa	Alta	...	Alta	Sim
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
3	Baixa	Baixa	...	Baixa	Sim

é simplificado computacionalmente, uma vez que não é tarefa trivial determinar por exemplo torques e ângulos de uma posição qualquer, ou para o caso geral de sistemas não-lineares, determinar a influência de uma variável em relação às outras representadas no estado, que aliás é exatamente a tarefa que se pretende realizar contornando o problema de cálculo analítico de cinemática e dinâmica.

Outra diferença importante em relação a modelos anteriores é mostrada no Passo 6e. Durante a inserção de um novo nodo, o STRAGEN recalcula as conexões que este novo nodo terá com os nodos vencedor e vice, mantendo-se sempre as duas menores. A Figura (4.3) mostra esse processo. A subfigura (a) mostra a rede antes da inserção do nodo 4 apresentado na subfigura (b). Na subfigura (b), o nodo 4 tem como topologicamente próximos os nodos 1 (vencedor) e 3, vice. O STRAGEN calcula as distâncias entre os pares de nodos (4, 1), (4, 3) e (1, 3), e mantém apenas as duas menores conexões, medidas segundo o critério de vizinhança estabelecido, neste caso as conexões (4, 1) e (1, 3), de modo que o nodo 4 inserido fica com apenas uma conexão.

**Figura 4.3** Exemplo de inserção de nodo e conexões no modelo STRAGEN (a) antes da inserção, e (b) depois da inserção.

Também para se evitar o cálculo analítico da mudança nas variáveis de estados de um determinado padrão completo já aprendido é que não se deve mover um nodo já estabelecido e corretamente aprendido. Mover um nodo em apenas um grupo homogêneo, implica em decidir sobre a manutenção das variáveis dos outros grupos intactas, a movimentação linear com a mesma taxa do grupo que foi adaptado, ou o cálculo analítico das variações do sistema não-linear para mover todas as variáveis com precisão. Nenhuma destas soluções são viáveis, as duas primeiras inserindo imprecisão ao sistema e a última eliminando a vantagem de identificar um sistema para se evitar seu cálculo analítico.

Tais diferenças apontadas mostram modificações necessárias para se construir um mapa topológico que prioriza as relações entre os nodos, no papel das conexões, para permitir que as transições de estados sigam critérios estabelecidos. Esta é uma importante modificação em relação às criações de conexões pela distância espacial Euclidiana utilizada no GWR e em diversos outros modelos SOM-DT. Além destas modificações apontadas, é necessário o acréscimo de um outro módulo, o gerador de trajetórias propriamente dito. Na versão STRAGEN-OFF, as fases de treinamento e planejamento ocorrem isoladamente, primeiro treinando-se a rede para representar o espaço de estados e em seguida recuperando informações sobre novas trajetórias, ou replanejando trajetórias com novos objetivos.

O planejamento é feito com a utilização de um algoritmo de busca global, a difusão de energia. Na Figura (4.4) (a) vê-se um mapa treinado em que se aplicou o algoritmo de difusão de energia. O nodo branco representa o início da trajetória e o nodo preto (rotulado com energia igual a 1) representa o nodo final. A difusão concentra uma unidade de energia no nodo final, e distribui esta energia em direção ao nodo inicial, até que este deixe de ser zero, quando o processo termina. A geração de trajetória se dá seguindo a partir do nodo inicial em direção ao nodo final, escolhendo como próximo nodo o que estiver com maior energia. Em casos de múltiplas escolhas com mesma quantidade de energia, o algoritmo se prende à primeira escolhida, ou seja, só troca de opção se melhorar, não se empatar.

Os caminhos e os nodos com suas respectivas energias, apresentados na Figura (4.4) estão descritos na Figura (4.5) para melhor visualização. Pode-se perceber que existem pontos de ambigüidade. A partir do nodo inicial, com energia de $4,8e^{-4}$, as opções são o nodo inferior ($7,2e^{-4}$) e o nodo superior que é o escolhido, com energia de $7,25e^{-4}$. Neste segundo nodo da trajetória, as opções são o nodo superior ($1,3e^{-3}$) e o nodo inferior que é escolhido ($1,6e^{-3}$). A partir desse terceiro nodo, apresenta-se uma ambigüidade. Este nodo tem quatro ligações com energias $7,25e^{-4}$ (voltando para o nodo anterior), $7,2e^{-4}$ (voltando por baixo), e dois nodos de maior energia com valores iguais a $4e^{-3}$. O algoritmo decidiu-se caminhar por baixo, por ter sido o primeiro nodo com maior energia que adquiriu. A partir do quarto nodo há outro

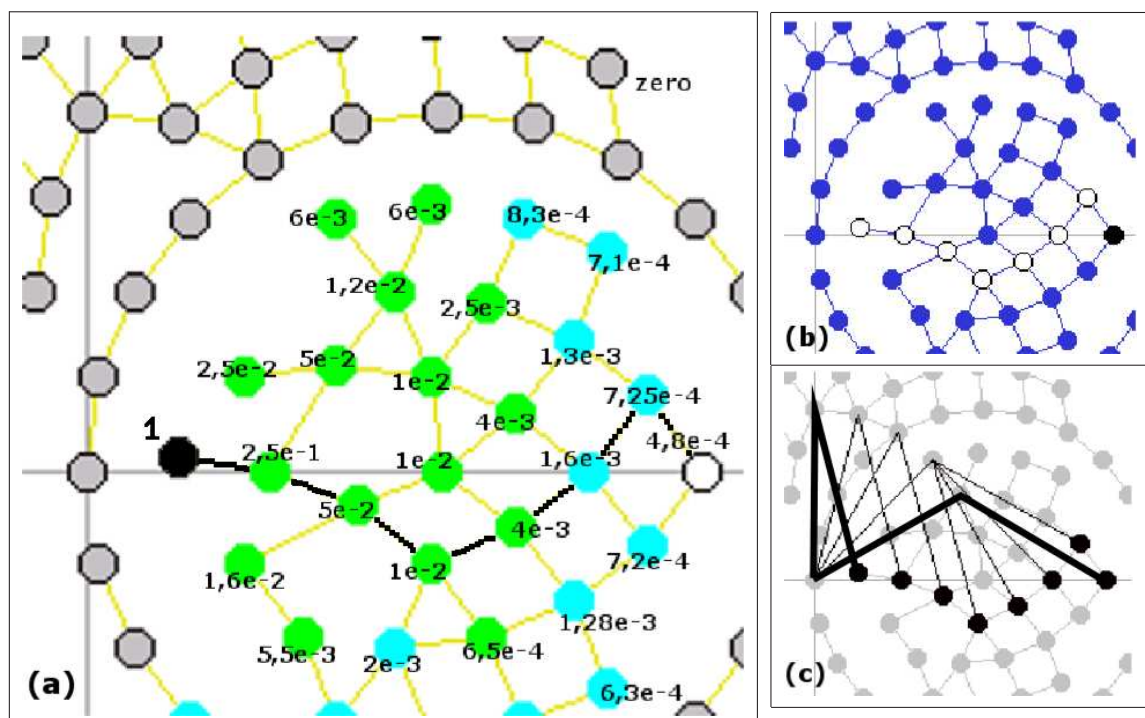


Figura 4.4 Difusão de Energia: Trajetória formada pelos nodos, a partir do nodo inicial até o nodo alvo, com as energias: $4,8e^{-4}$, $7,25e^{-3}$, $1,6e^{-3}$, $4e^{-3}$, $1e^{-2}$, $5e^{-2}$, $2,5e^{-1}$ e 1 . (a) Difusão de energia desde o ponto alvo (preto) até o ponto inicial (branco). (b) Trajetória formada após a difusão, seguindo pelo vizinho de maior energia, desde o ponto inicial (preto) até o ponto final. (c) Posicionamento do braço do robô bidimensional nos pontos da trajetória.

ponto de ambigüidade, os dois de maior energia têm valores $1e^{-2}$, e novamente o algoritmo escolheu continuar por baixo, pelo primeiro nodo que pegou. Do quinto nodo se caminha para o sexto de valor $5e^{-2}$. A partir do sexto nodo, a opção de maior energia é $2,5e^{-1}$. Do sétimo nodo finalmente se atinge o nodo alvo da trajetória (oitavo), com energia igual a 1 . Chamando de opção (a) o nodo inferior (ou o escolhido), (b) a opção ambígua superior e assim sucessivamente, pode-se gerar todas as trajetórias para este trecho, e com isso percebe-se que as escolhas ambíguas não podem acarretar em escolhas ruins, pois as trajetórias sempre terão o mesmo custo total.

Por exemplo, seguindo do nodo inicial com energia $4,8e^{-4}$ até o nodo final com energia 1 , poderia-se passar pelas trajetórias dadas na Figura (4.5).

Na Figura (4.4) (b) vê-se os nodos da trajetória escolhida, e na Figura (4.4) (c) observa-se o posicionamento dos braços nestes pontos. Outros experimentos podem ser vistos com mais detalhes no Capítulo 5. O algoritmo STRAGEN-OFF descrito nesta seção foi modificado para poder trabalhar de forma *on-line* e a seguir será visto o modelo resultante chamado de

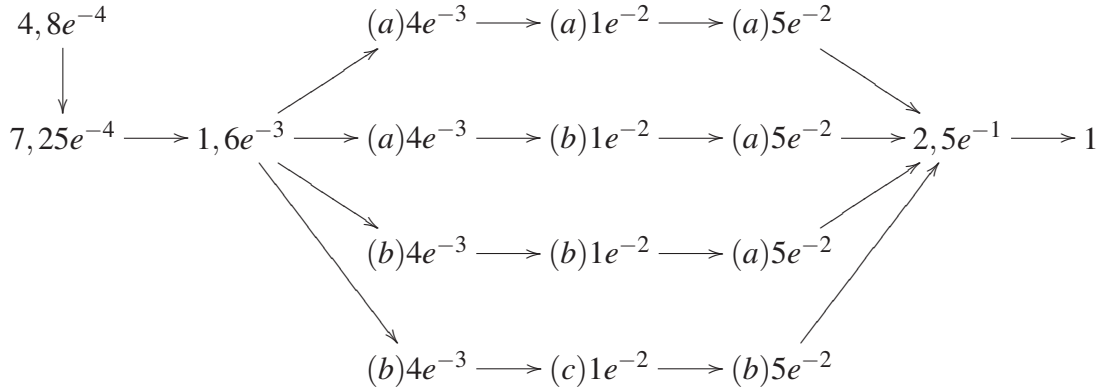


Figura 4.5 Trajetórias possíveis na difusão de energia.

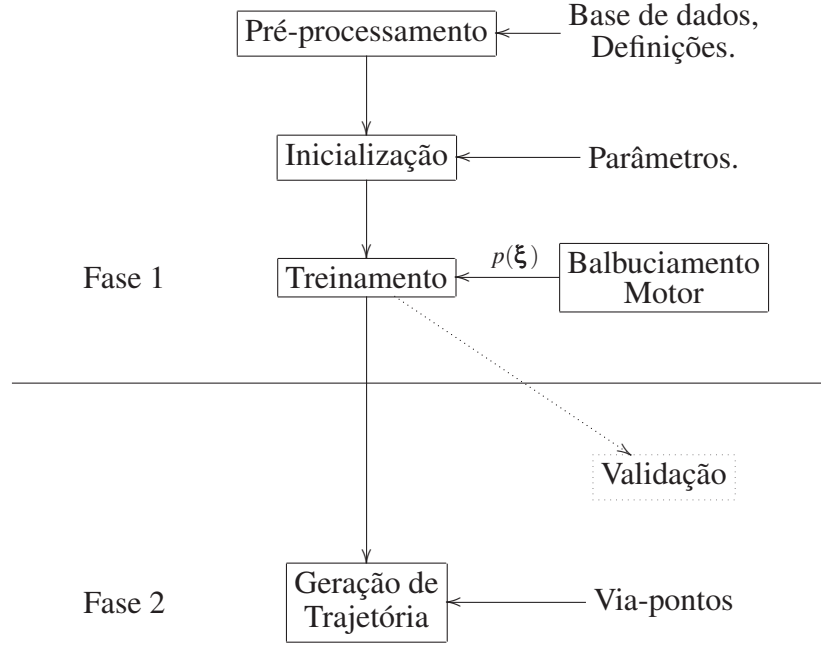
STRAGEN-ON.

4.2.6 Modelo STRAGEN On-line

O algoritmo é composto de duas fases, não havendo uma fase intermediária de poda. Primeiramente, a Fase de Treinamento cria uma representação da topologia do espaço de estados, adaptando-se e podando conexões e nodos desnecessários durante o processo. Na segunda fase, a Fase Geração de Trajetórias, requisita-se ao modelo que encontre a melhor trajetória entre dois pontos diretamente ou passando por pontos intermediários (chamada *via-pontos*), de acordo com o critério estabelecido no treinamento. A trajetória pode ser gerada pelo algoritmo de menor caminho (Dijkstra, 1959) ou pelo algoritmo de difusão (Zeller et al., 1997) explanado anteriormente. Durante a execução da segunda fase o erro de validação é calculado e utilizado para comparar este modelo com os modelos STRAGEN-OFF, GNG e GWR.

Antes de iniciar a fase de treinamento, o STRAGEN-ON também realiza a fase de pré-processamento descrita no início da Seção 4.2, para normalizar a base de dados $\mathbf{B}^{(0)}$, definir as características do vetor de pesos $\mathbf{w}_i = [w_{i1} \dots w_{iD}]^T = [\mathbf{V}_1 \mathbf{V}_2 \dots \mathbf{V}_m]^T$, os critérios de vizinhança η e atividade ζ , e calcular o limite de atividade $\bar{a}_k$ baseado na porcentagem P do espaço de estados que um nodo deve responder, normalmente com $P = 1\%$.

A Figura (4.6) mostra um diagrama com o fluxo das fases do STRAGEN-ON. O STRAGEN-ON também utiliza para seu treinamento o procedimento de Balbuciamiento Motor definido na Subseção 4.2.1. Inicializado o algoritmo de acordo com a descrição da Subseção 4.2.3, seu procedimento segue como abaixo indicado.

**Figura 4.6** Fluxograma: STRAGEN-ON

4.2.6.1 Fase de Treinamento do STRAGEN-ON

O algoritmo de treinamento STRAGEN-ON é uma modificação do algoritmo STRAGEN-OFF, capaz de eliminar nodos e conexões durante o treinamento sem a necessidade de uma fase de poda (Subseção 4.2.4.2). O procedimento do algoritmo STRAGEN-ON é idêntico ao STRAGEN-OFF desde o Passo 1 até o Passo 7. O Passo 8 e subseqüentes são substituídos pelo algoritmo abaixo:

8. Calcule o tamanho médio e o desvio padrão de todas as conexões $k = |\mathcal{N}(s_1)|$ emanando do nodo vencedor s_1 , e o limiar para remoção de conexões χ :

$$\begin{aligned}
 \bar{m} &= \frac{\sum_{i=1}^k \text{Dist}(s_1, n_i)}{k} \\
 \bar{d} &= \sqrt{\frac{\sum_{i=1}^k [\text{Dist}(s_1, n_i) - \bar{m}]^2}{k - 1}} \\
 \chi &= \bar{m} + \varpi \cdot \bar{d}
 \end{aligned} \tag{4.13}$$

onde $|\mathcal{N}(s_1)|$ é o número de vizinhos de s_1 , $n_i \in \mathcal{N}(s_1)$ e $\varpi = 1,5$ é uma constante ótima determinada empiricamente.

9. Se $|\mathcal{N}(s_1)| > 2$, remova todas as conexões c_{s_1, n_i} do conjunto $\mathcal{C}$ para as quais se tem que $n \in \mathcal{N}(s_1)$ e $\text{Dist}(s_1, n) > \chi$.
10. Remova todos os nodos $n \in \mathcal{A}$ isolados, i.e., nodos sem pelo menos um vizinho: $|\mathcal{N}(n)| = 0$.
11. Repita a partir do Passo 1 (p. 98) até o número máximo de iterações $t_{\max}$, ou até que algum critério de parada seja atingido.

4.2.6.2 Fase de Geração da Trajetória por Menor Caminho

Dado qualquer ponto inicial ξ_{q_0} , um ponto alvo ξ_{q_f} , e qualquer número de pontos intermediários ξ_{q_l} , $0 < l < f$, chamado *via-pontos*, encontre o menor caminho de acordo com o algoritmo de caminho mais curto (Dijkstra, 1959; Russel & Norvig, 2002), de q_0 até q_f , passando por todos os via-pontos q_l .

1. Seja a cadeia de pontos alvo $q_0, \dots, q_f$.
2. Seja $i = 0$ e $j = 1$.
3. Repita até $j = f$:

(a) Encontre os nodos vencedores s_i e s_j que representam q_i e q_j , respectivamente:

$$\begin{aligned} \|\mathbf{V}_{\eta, s_i} - \xi_{\eta, q_i}\| &\leq \|\mathbf{V}_{\eta, k} - \xi_{\eta, q_i}\|, \\ \|\mathbf{V}_{\eta, s_j} - \xi_{\eta, q_j}\| &\leq \|\mathbf{V}_{\eta, k} - \xi_{\eta, q_j}\|, \\ &\forall k \in \mathcal{A} \end{aligned}$$

- (b) Encontre o caminho mais curto de s_i até s_j de acordo com o algoritmo de Dijkstra.
- (c) Mova para o próximo trecho, $i = i + 1$ e $j = j + 1$.

Uma vez tendo todos os nodos necessários para seguir algum caminho, cada nodo fornece informação adicional para execução completa da trajetória, (por exemplo, é simples determinar a cinemática e a dinâmica – ou outra informação geral em outros domínios – desde de que os nodos codifiquem esta informação nos vetores de pesos, podendo ser facilmente usada para executar o completo padrão), evitando a necessidade de calcular essas informações.

4.2.7 Diferenças entre o STRAGEN-ON e o STRAGEN-OFF

O treinamento *off-line* do modelo STRAGEN-OFF e a aplicação posterior de uma fase de poda permite a criação de um mapa topológico que representa muito bem o espaço de estados, pois durante a fase de poda se tem total acesso à configuração final do mapa e se pode escolher as conexões e nodos a serem eliminados de acordo com critérios bem definidos. O STRAGEN-OFF elimina nodos e conexões que não vencem nenhuma vez para todos os pontos de entrada apresentados na fase de poda, desde que não se desconecte o mapa. Este procedimento garante que apenas conexões desnecessárias são eliminadas, além de manter o mapa conectado permitindo que todos os estados sejam acessíveis. Por outro lado, o STRAGEN-OFF não representa adequadamente regiões desconexas, uma vez que ele não se permite desconectá-las.

O processo de eliminação de nodos e conexões de modo dinâmico, tendo em vista a utilidade do modelo para a solução de problemas de trajetórias de estados é um problema difícil, que sempre apresenta prós e contras. O algoritmo STRAGEN-ON apresenta um procedimento suave para a redução de conexões de modo dinâmico e automático, que leva em conta a densidade de nodos que está atualmente representando o mapa a ser aprendido, inspirado no processo de eliminação de falsos vizinhos (Zhao & Small, 2006). O procedimento implementado é simples e inovador e consiste em remover conexões que sejam maiores que χ (a média do tamanho das conexões atuais no mapa adicionados a uma taxa do seu desvio padrão) apenas para nodos que tenham três ou mais conexões, conforme descrito no Passo 9.

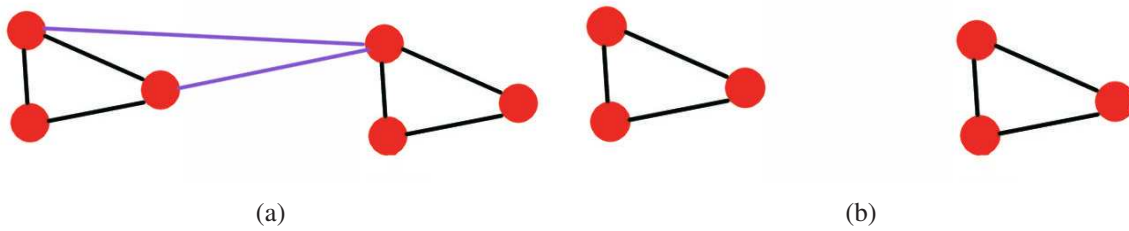


Figura 4.7 Exemplo de remoção de conexão no modelo STRAGEN-ON (a) antes da remoção, e (b) depois da remoção das conexões maiores que χ .

Esta pequena modificação no STRAGEN torna-o capaz de mapear com precisão o espaço de estados e em menos tempo, uma vez que não necessita mais da fase de poda. Além disto, o mapa do STRAGEN-ON pode representar regiões desconexas, geradas de tal forma que não surgem nodos isolados no processo e sem o custo computacional de avaliar cada conexão removida, pois o STRAGEN-ON não elimina conexões de nodos que tenham apenas uma ou duas conexões. Existem dois passos no algoritmo que removem conexões. O Passo 9, que trata ex-

plicitamente da remoção de conexões, e o Passo 6c que trata do processo de inserção de nodos. O Passo 6c, apesar de remover uma das conexões entre os nodos vencedores próximos do nodo inserido, o faz apenas para substituí-la por outras que melhor representem a similaridade entre os três nodos e, portanto, também não ocasiona nodos isolados. A Figura (4.7) (a) mostra uma rede exemplo antes do passo de remoção, e em (b) após a remoção das conexões maiores que χ (Eq. 4.13).

Outras diferenças entre o STRAGEN-OFF e o STRAGEN-ON derivam do uso *on-line* possível neste último. Zeller et al. (1997) propõe o seguinte procedimento para planejar uma trajetória no algoritmo TRN:

1. Leia a posição inicial ξ_{q_0} e a posição alvo ξ_{q_f} no espaço visual.
2. Ache os nodos vencedores para a posição inicial n_{q_0} e alvo n_{q_f} .
3. Detecte os obstáculos no espaço visual e elimine as conexões estão obstruídas.
4. Faça a trajetória segundo o algoritmo de difusão de energia.
5. Siga pelo maior gradiente desde o nodo inicial n_{q_0} até o nodo final n_{q_f} .

O STRAGEN-ON tem a capacidade de desabilitar conexões se necessário, quando utiliza como gerador de trajetórias o algoritmo de menor caminho de Dijkstra. O algoritmo de Dijkstra tem a propriedade de computar não só o menor caminho, mas o caminho com o menor custo. Ao se alterar o custo de uma conexão para ∞ , o algoritmo automaticamente busca trajetórias alternativas. Desta forma, o Passo 3 proposto acima não precisa eliminar a conexão, mas só desabilitá-la temporariamente, durante o tempo que o obstáculo está sendo detectado, e com isso não se perde informações importantes sobre a vizinhança dos nodos obstruídos.

4.3 Conclusões

Este capítulo inicialmente levantou, dentre os modelos analisados, os requisitos necessários para o modelo a ser proposto. Em seguida, foi apresentado o modelo *State Trajectory Generator* (STRAGEN), em uma primeira abordagem chamada de *off-line*. As inovações e limitações do modelo foram então discutidas, em comparação com modelos anteriores, em especial comparando-o com o modelo GWR. Suas limitações indicaram o caminho para a proposta de uma nova abordagem, chamada de *on-line* que foi em seguida descrita. O capítulo termina

com uma discussão sobre as importantes diferenças entre ambas abordagens propostas e suas características.

No próximo capítulo serão apresentados os experimentos feitos com o STRAGEN para quatro domínios distintos: manipulador robótico bidimensional, manipulador robótico com 6 graus de liberdade trabalhando no espaço, mão robótica trabalhando no espaço e progressões harmônicas musicais. Os resultados são finalmente comparados e discutidos para cada domínio em questão.

CAPÍTULO 5

Resultados

Mas creio que um triângulo, se pudesse falar, diria igualmente que Deus é triangular; e um círculo afirmaria ser a natureza divina primordialmente circular.

—SPINOZA (Carta 60)

Neste capítulo, na Seção 5.1, será visto como foi criada a base de dados para testes do robô bidimensional, os ajustes iniciais necessários para as simulações, resultados das simulações e comparações do algoritmo STRAGEN com os algoritmos GNG e GWR. Na Seção 5.2 serão vistas simulações e elucidações sobre o comportamento do modelo proposto quando atuando em um robô tridimensional industrial PUMA-560. A Seção 5.3 apresenta as simulações feitas no modelo de mão antropomórfica Kanguera. A Seção 5.4 mostra como utilizar o STRAGEN para a análise de seqüências harmônicas de acordes.

5.1 Simulações para Manipulador Bidimensional

5.1.1 Configuração do STRAGEN

Robôs podem ser matematicamente definidos como uma cadeia de elos (*links*) conectados serialmente por juntas (*joint*), assumindo-se que cada junta possui um grau de liberdade, seja translacional ou rotacional. Para um robô com n elos, numerados de 1 a n , há $n + 1$ juntas numeradas de 1 a $n + 1$. A origem do sistema de coordenadas é normalmente fixada na junta 1; cada junta i conecta os elos $i - 1$ e i e ao mover-se, atua sobre o respectivo elo i ; a junta $n + 1$ carrega o efetuador (extremo do elo n). O braço do robô é o conjunto de todos os seus elos e juntas. Um membro i é o conjunto do elo i e da junta i .

Utilizando a ferramenta de simulação robótica denominada *Robotics Toolbox*, do software MATLAB (Corke, 1996), foi criada uma nova instância de um robô planar de dois elos, que opera no plano vertical (XY), Figura (2.4).

Para criar cada membro (elo e junta), de um robô bidimensional, tem-se:

```
L1=link([0 1 0 0 0])
L2=link([0 1 0 0 0])
```

tal que `link([alpha A theta D sigma])`, que representa o elo, descrito em termos dos parâmetros de Denavit-Hartenberg (Craig, 1986), tem sua definição como segue:

Um elo pode ser considerado como um corpo rígido que define a relação entre dois eixos de juntas vizinhos, e é representado por dois parâmetros: o seu tamanho A e o ângulo que forma com a junta anterior α .

Uma junta pode ser descrita por outros dois parâmetros: o ângulo da junta θ_i , descreve a rotação de um elo i com respeito ao elo anterior $i - 1$. O ângulo é a variável livre em juntas rotativas (enquanto o deslocamento D é constante). O outro parâmetro é o deslocamento do elo pelo eixo da junta D , que descreve a distância percorrida em relação à origem do eixo dado pela junta anterior. O deslocamento é a variável livre para juntas de translação enquanto θ permanece constante.

Tabela 5.1 Constantes cinemáticas de um robô de 2 graus de liberdade.

alpha	A	theta	D	sigma
0,0	1,0	0,0	0,0	R (std)
0,0	1,0	0,0	0,0	R (std)

Por último, a função *link* possui um parâmetro *sigma* que é apenas uma *flag* que indica se a junta é de revolução (*sigma* igual a zero) ou prismática (*sigma* diferente de zero).

O comando:

```
r2d=robot(L1 L2)
```

retorna um objeto MATLAB com as definições do robô, em que se podem utilizar as funções definidas para cálculo de cinemática e dinâmica.

Uma vez definido o robô, é possível gerar uma base de dados com os torques e posições respectivas do efetuador e das juntas. Para o cálculo dos torques do robô *r2d* definido acima, uma das opções é utilizar a função *Newton-Euler Recursiva* (RNE), embutida na função:

```
torques=rne(r2d, theta, v, a)
```

Tabela 5.2 Base $\mathbf{B}_1$ com $L_1 = 600$ pontos: Posição e ângulo das juntas e torque em equilíbrio estático de um robô bidimensional movendo-se no plano XY.

X_1	Y_1	X_2	Y_2	θ_1	θ_2	τ_1	τ_2
1,0	0,0	0,0	0,0	0,0	-180,0	10,0	-10,0
1,0	0,0	0,034	-0,259	0,0	-165,0	10,341	-9,659
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1,0	0,0	2,0	0,0	0,0	0,0	30,0	10,0
1,0	0,0	1,966	0,259	0,0	15,0	29,659	9,659
1,0	0,0	1,866	0,50	0,0	30,0	28,660	8,660
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1,0	0,0	0,0	0,0	0,0	180,0	10,0	-10,0
0,966	0,259	0,0	0,0	15,0	-180,0	9,659	-9,659
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0,866	0,50	0,866	1,50	30,0	60,0	17,320	0,0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0,500	0,866	0,500	1,866	60,0	30,0	10,0	0,0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0,0	1,0	0,0	2,0	90,0	0,0	0,0	0,0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

onde foi considerado v (velocidade) zero, a (aceleração) zero, e θ uma matriz composta por todas as possíveis combinações dadas pelos respectivos ângulos das juntas $\theta_1 = \{0; 15; 30; \dots; 330; 345\}$ e $\theta_2 = \{-180; -165; -150; \dots; 165; 180\}$, perfazendo um total 24×25 , ou seja, 600 pontos de treinamento em todo o espaço de trabalho do robô. A Figura (5.1) mostra a distribuição destes pontos no espaço de entrada, denominada de base de dados $\mathbf{B}_1$. Um trecho desta base de dados é apresentada na Tabela 5.2. As colunas X_1 e Y_1 representam a posição cartesiana da junta J_2 no espaço; as colunas X_2 e Y_2 representam a posição do efetuador (análogo a junta J_3), sendo a junta J_1 posicionada fixa no ponto (0,0). A coluna θ_1 representa o ângulo que perfaz a junta J_1 , enquanto que a coluna θ_2 representa o ângulo da junta J_2 (Figura (2.4)).

Definida a base de dados $\mathbf{B}_1$ de dimensões $L \times D$, onde $L_1 = 600$ é o número de padrões e $D = 8$ é a dimensão de cada padrão, o vetor peso para treinamento da rede se caracteriza por:

$$\mathbf{w}_i = [w_{i1} \dots w_{iD}]^T, \quad 1 \leq i \leq L$$

$$\mathbf{w}_i = [X_1 \ Y_1 \ X_2 \ Y_2 \ \theta_1 \ \theta_2 \ \tau_1 \ \tau_2]^T \quad (5.1)$$

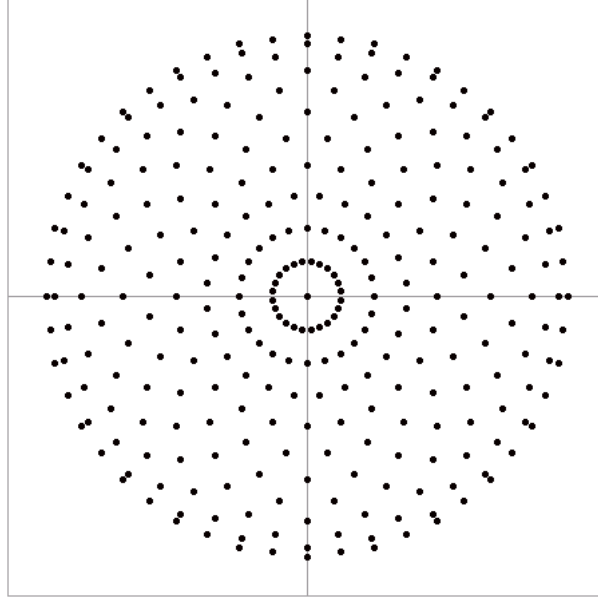


Figura 5.1 Nuvem de 600 pontos para treinamento de um robô 2D, conforme Tabela 5.2. Os pontos representam a posição do efetuador no plano X_2Y_2 .

O vetor de pesos $\mathbf{w}$ pode ser dividido, neste caso, em $m = 3$ grupos homogêneos: As posições espaciais das juntas, os ângulos das juntas e os torques nas juntas.

$$\begin{aligned}\mathbf{V}_1 &= \left[[X_1 \ Y_1]_1^T \ [X_2 \ Y_2]_2^T \right]^T \\ \mathbf{V}_2 &= \left[[\theta_1]_1^T \ [\theta_2]_2^T \right]^T & \mathbf{V}_3 &= \left[[\tau_1]_1^T \ [\tau_2]_2^T \right]^T\end{aligned}$$

O vetor de pesos pode então ser reescrito como:

$$\mathbf{w}_i = [\mathbf{V}_1 \ \mathbf{V}_2 \ \mathbf{V}_3]^T = [X_1 \ Y_1 \ X_2 \ Y_2 \ \theta_1 \ \theta_2 \ \tau_1 \ \tau_2]^T \quad (5.2)$$

e cada vetor $\mathbf{V}_i$, $i = 1, \dots, 3$ representa um conjunto de variáveis homogêneas do domínio que podem ser agrupadas e operadas como um todo. As dimensões de cada grupo são tais que $\mathbf{V}_i \in \mathbb{R}^{D_i}$ e $\sum_{i=1}^3 D_i = D$. As dimensões individuais dos grupos são: $\mathbf{V}_1 \in \mathbb{R}^4$, $\mathbf{V}_2 \in \mathbb{R}^2$ e $\mathbf{V}_3 \in \mathbb{R}^2$.

Tendo os grupos sido formados, deve-se escolher quais serão utilizados no cálculo da proximidade dos nodos com os padrões de entrada, para se definir o nodo vencedor, i.e., o cálculo da *vizinhança*. Chama-se $\mathbf{V}_\eta$ o vetor composto pelos grupos escolhidos para se realizar o cálculo da vizinhança, que será o *critério de vizinhança* da rede. Os critérios utilizados para as

simulações do robô bidimensional foram:

1. (DE) Distância Euclidiana entre as posições do efetuador: $\mathbf{V}_\eta = [\mathbf{V}_1]$.
2. (DA) Distância Euclidiana entre ângulos das juntas: $\mathbf{V}_\eta = [\mathbf{V}_2]$.
3. (DT) Distância Euclidiana entre torques nas juntas: $\mathbf{V}_\eta = [\mathbf{V}_3]$.
4. (DEA) Distância Euclidiana entre posições do efetuador E distância Euclidiana entre ângulos: $\mathbf{V}_\eta = [\mathbf{V}_1 \ \mathbf{V}_2]$.
5. (DET) Distância Euclidiana entre posições do efetuador E distância Euclidiana entre torques: $\mathbf{V}_\eta = [\mathbf{V}_1 \ \mathbf{V}_3]$.
6. (DAT) Distância Euclidiana entre ângulos E distância Euclidiana entre torques: $\mathbf{V}_\eta = [\mathbf{V}_2 \ \mathbf{V}_3]$.
7. (DEAT) Distância Euclidiana entre posições do efetuador E distância Euclidiana entre ângulos E distância Euclidiana entre torques: $\mathbf{V}_\eta = [\mathbf{V}_1 \ \mathbf{V}_2 \ \mathbf{V}_3]$.

O algoritmo STRAGEN explicado no capítulo anterior, nas Subseções 4.2.4 e 4.2.6, foi implementado com sucesso em um sistema openSUSE GNU/LINUX 2.6 x86_64, utilizando linguagem C/C++ padrão ANSI, biblioteca gráfica ALLEGRO, e interface gráfica QT3. A plataforma utilizada para as simulações foi um computador com processador Athlon AMD64 3200+ com 2GB de memória RAM. Os parâmetros são passados para o algoritmo através da linha de comandos e de um arquivo de configuração, e uma interface gráfica chamada KSTRAGEN permite a alteração visual destes parâmetros (Figura 5.2).

O algoritmo foi dividido em duas partes distintas, para sua implementação. A fase 1 compreende o treinamento da rede. A fase 2 compreende a geração da trajetória de estados e controle de malha aberta.

Se nada for dito ao contrário nas próximas seções, os parâmetros utilizados serão sempre: taxa de aprendizado do nodo vencedor $\epsilon_b = 0,2$, taxa final de aprendizado $\alpha_f = 0,1$, máximo de disparos estimados por nodo $\sigma_f = 2 \cdot t_{\max}/L$, e número de candidatos no procedimento MB definido como $Q = 10$. Para o STRAGEN-OFF, serão realizados sete experimentos, um para cada critério definido acima, em que se utiliza como dados de entrada a base de dados com 600 pontos que representam posições do efetuador de um robô bidimensional no plano XY, de dois elos e duas juntas (Tabela 5.2). O número máximo de iterações $t_{\max} = 3 \cdot L = 1800$, ou seja, a base de dados terá seus padrões apresentados à rede 3 vezes.

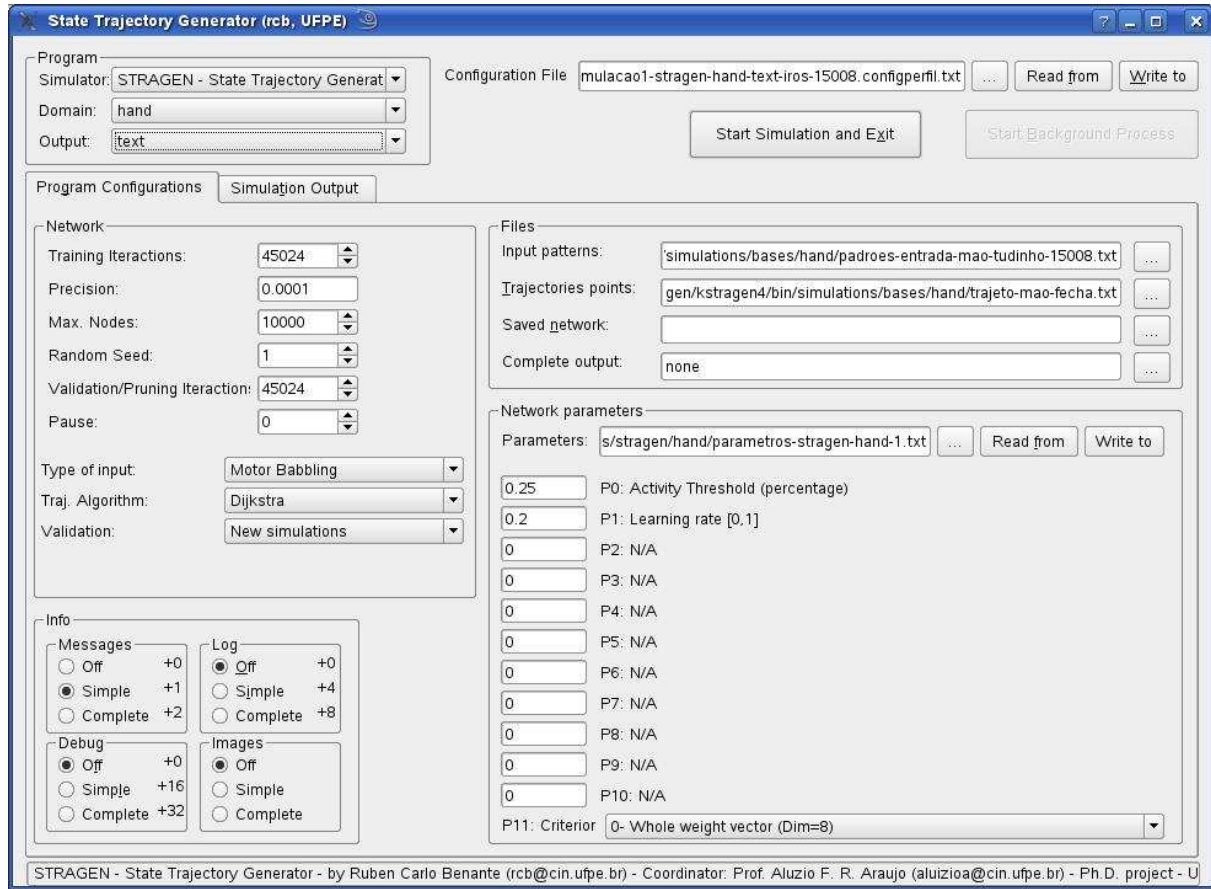


Figura 5.2 Interface gráfica KSTRAGEN.

O objetivo destas simulações é testar o modelo para o domínio em questão, colher dados relativos às trajetórias geradas e comparar os critérios de vizinhança e como estes afetam a trajetória e o mapeamento do espaço de entrada.

Ao final da Seção 5.1.2, os resultados serão comparados com simulações feitas com os modelos GNG e GWR. Para o modelo GNG, os parâmetros adotados são: $\lambda = 10$ e $a_{\max} = 100$. Para o modelo GWR, os parâmetros são: $a_T = 0.97$ e $a_{\max} = 25$. Estes parâmetros foram escolhidos por estarem próximos das sugestões feitas pelos autores nas simulações comparativas entre tais modelos (Fritzke, 1997a; Marsland et al., 2002) levantadas no Capítulo 3, e serão usados nas simulações que se seguem, se nada for dito ao contrário.

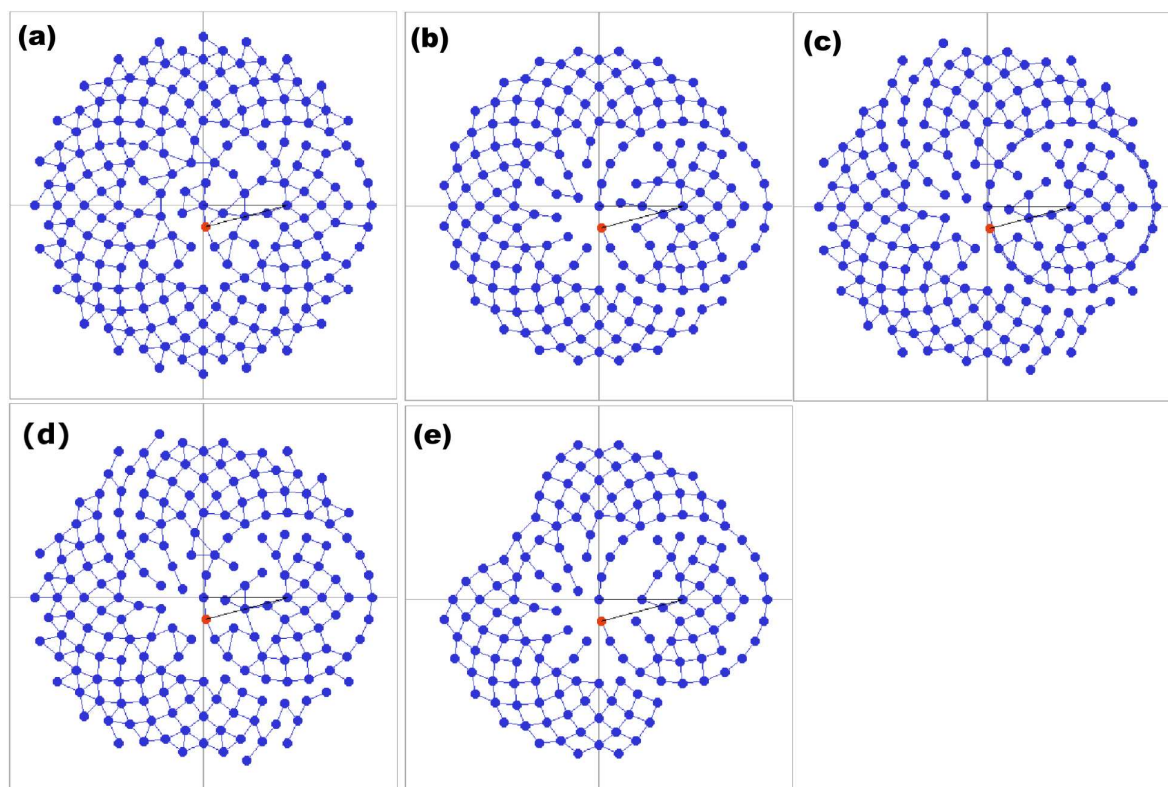


Figura 5.3 Simulações: Mapas finais treinados com os critérios de vizinhança: (a) DE; (b) DA (análogo ao DEA); (c) DT; (d) DET; (e) DAT (análogo ao DEAT).

5.1.2 Resultados

5.1.2.1 Treinamento da rede STRAGEN (Fase 1)

Para enfatizar a importância do Balbucimento Motor, as primeiras simulações abaixo foram feitas sem esta técnica, com uma apresentação sequencial dos pontos, de modo que a Junta 2 (cotovelo) girava todo o ciclo, de -180° até $+180^\circ$ (em intervalos de 15°), para então a Junta 1 (ombro) girar para o próximo intervalo (ângulos definidos de 0° a 360° , em intervalos de 15°), e assim sucessivamente, até que todos os pontos fossem apresentados uma vez.

Apresentar os pontos em uma ordem que permita o aprendizado de conexões possíveis torna viável a simulação da característica de balbucimento motor, na qual o braço movimenta-se em diversas direções enquanto as possíveis rotas e posições são aprendidas e generalizadas. Porém esta ordem não deve ser sequencial, e sim conter alguma aleatoriedade que permita ao modelo criar vizinhanças diversas e com isso apresentar mais robustez e estabilidade.

A Figura (5.3) mostra o mapa gerado por este treinamento quando os critérios de vizinhança

são respectivamente: **(a)** DE - Distância Euclidiana entre posições do efetuador; **(b)** DA - Distância Euclidiana entre ângulos das juntas, mapa gerado idêntico ao gerado pelo critério DEA - Distância Euclidiana entre nodos E distância Euclidiana entre ângulos; **(c)** DT - Distância Euclidiana entre torques nas juntas; **(d)** DET - Distância Euclidiana entre posições do efetuador E distância Euclidiana entre torques; **(e)** DAT - Distância Euclidiana entre ângulos E distância Euclidiana entre torques, mapa gerado idêntico ao gerado pelo critério DEAT - Distância Euclidiana entre nodos E distância Euclidiana entre ângulos E distância Euclidiana entre torques.

Na Figura (5.4) vê-se a evolução da rede quando utilizou-se o critério de vizinhança DE (também na Figura (5.3) (a), o mapa final). Este critério da distância espacial entre nodos é largamente utilizado em redes neurais artificiais. Entretanto, como será visto no momento da geração da trajetória, mais adiante, este critério não é viável para este modelo e domínio específico, por criar caminhos que não podem ser desenvolvidos na prática, como saltos nas configurações do braço. Estas configurações serão melhor vistas nas figuras dos braços a seguir. O critério de vizinhança adotado é fundamental para as etapas seguintes do algoritmo.

A rede treinada com este critério (DE) gerou um mapa com 195 nodos dispostos equidistantes, os quais foram capazes de representar os 600 pontos de treinamento com erro acumulado de 0,006812. Este erro mede a distância Euclidiana quadrática entre os padrões de entrada e os nodos vencedores que os representam. No caso de erro igual a zero, indica que o padrão está exatamente representado por um nodo. O erro acumulado de todos os nodos informa se o mapeamento está representando os dados de entrada adequadamente. Como será visto adiante, os erros entre os mapeamentos do espaço de entrada utilizando os sete critérios adotados variam pouco e portanto não são decisivos na escolha do melhor critério. O melhor critério deve ser escolhido baseado em outras informações que não o erro, por exemplo características sobre o tipo de trajetórias que gera, tais como trajetória mínima, de movimento suave e sem solavancos, além da cobertura do espaço de estados.

Os 600 padrões de entrada distribuídos no espaço podem ser visualizados junto com os nodos para as cinco principais simulações (DE, DA, DT, DET, DEAT) na Figura (5.7) (p. 128). Os padrões de entrada são os pequenos pontos pretos distribuídos. Os nodos são os círculos maiores em tons de cinza. Para cada nodo há um padrão exatamente em seu centro, uma vez que os nodos são criados em cima dos padrões de entrada. Há padrões sem nodos que os representem com essa exatidão, e neste caso tais padrões são representados pelo nodo mais próximo (ou vencedor), o que gera o erro calculado.

Na Figura (5.3) (b) utilizou-se o critério para vizinhança DA. A principal diferença que se pode observar é a ausência de rotas que ligam o terceiro quadrante ao primeiro, passando pelo quarto. Ao se comparar a Figura (5.3) (a) com a Figura (5.3) (b) nota-se que com o critério DE

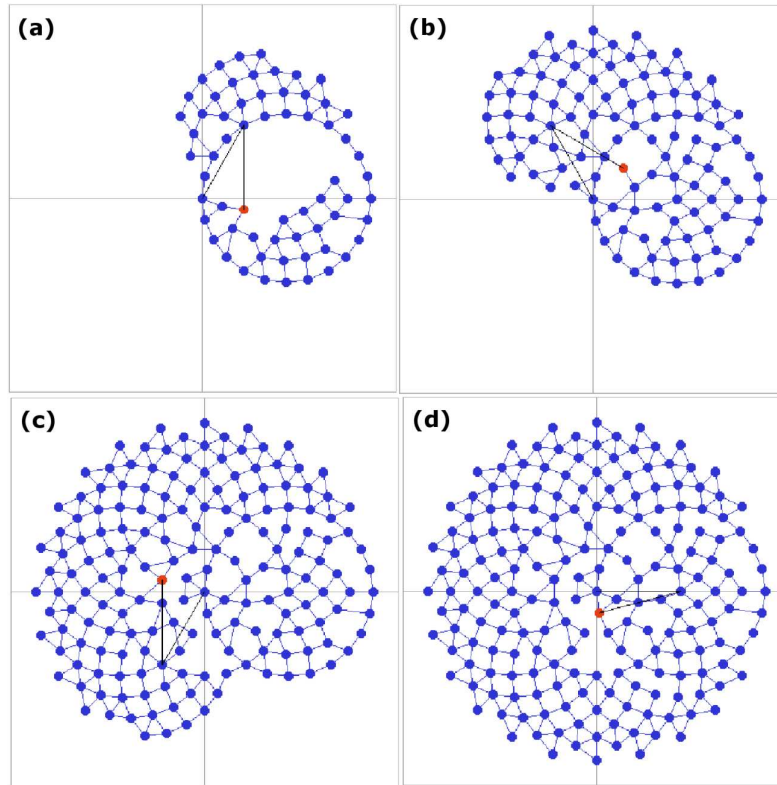


Figura 5.4 Simulação: Evolução do treinamento, critério de vizinhança DE. A rede auto-organizável STRAGEN aprende o espaço de trabalho de um robô bidimensional. (a) 100 iterações, 63 nodos, 102 conexões; (b) 200 iterações, 110 nodos, 196 conexões; (c) 400 iterações, 173 nodos, 322 conexões; (d) 600 iterações, 195 nodos, 366 conexões.

foi criada uma vizinhança ligando o terceiro quadrante com o quarto quadrante, enquanto que com o critério DA, essa ligação não apareceu (veja a ligação entre os nodos 4 e 5, na Fig. 5.9 a). O fato se deve à configuração necessária para o robô se posicionar nestes nodos.

Em relação à simulação DE, na simulação DA a rede ficou menor, com apenas 172 nodos e um erro de aproximadamente 0,008057. O erro reflete a menor quantidade de nodos utilizada. O fato do mapeamento contar com menos nodos ao utilizar este critério de vizinhança é explicado pela própria restrição que este critério estabelece, sendo portanto mais difícil surgirem nodos vizinhos e estes se estabelecerem e sobreviverem ao treinamento. A quantidade de conexões baixou, em relação ao critério anterior, de 366 para 273 conexões, o que gera um ganho de velocidade de cálculo (que é compensado pela pequena perda de precisão).

Na Figura (5.3) (c), utilizou-se o critério de vizinhança DT. Pode-se observar neste treinamento que o número de conexões é muito grande em relação aos critérios anteriores (322 conexões) e que alguns nodos estão conectados diretamente com vizinhos distantes, mesmo

Tabela 5.3 Número de nodos e conexões (N/C) durante evolução das simulações DE, DA, DT, DEA, DET, DAT e DEAT.

Iteração	DE	DA	DT	DEA	DET	DAT	DEAT
100	63/102	59/83	62/112	59/83	62/98	59/83	59/83
200	110/196	103/157	106/176	103/157	106/192	94/136	94/136
400	173/322	157/250	165/290	157/250	165/276	141/217	141/217
600	195/366	172/273	187/322	172/273	187/308	150/233	150/233

tendo vizinhos intermediários em comum, o que cria atalhos na geração de rotas.

Em um primeiro teste quando ainda se estavam levantando as possibilidades da dinâmica de aprendizado do STRAGEN, foi utilizado um limiar máximo para o tamanho das conexões. Curiosamente, utilizar um limiar para limitar a distância entre vizinhos não resolve o problema da multiplicidade de conexões, e ainda cria outros agravantes como o de tornar o algoritmo incapaz de mapear regiões onde as conexões necessitariam de um limiar maior para surgirem. A fase de poda da versão STRAGEN-OFF surgiu a partir da evolução da versão inicial para resolver este problema, e para a versão STRAGEN-ON a poda foi incorporada no processo de treinamento.

O critério DEA gerou um mapa idêntico ao mapa gerado pelo critério DA (Fig. (5.3) b). Isso se deve ao fato de que o critério DA foi mais restritivo que o critério DE, eliminado a influência deste.

Na Figura (5.3) (d) utilizou-se o critério de vizinhança DET que é a combinação entre os critérios DE e DT. Esta combinação permitiu corrigir o problema de excessos de conexões quando se utiliza apenas o torque, sem com isso deixar áreas do espaço de entrada descobertas. Nota-se que a utilização do DT permite rotas mais complexas, principalmente na área próxima a origem, onde o posicionamento é restrito devido às limitações da geometria do próprio braço.

A Figura (5.3) (e), mostra o mapa criado tanto pelo critério DAT quanto pelo DEAT. Devido à alta restrição de vizinhanças imposta pelos critérios DAT (combinação de DA e DT) e DEAT (combinação de DE, DA e DT) suas simulações resultaram em um mapa idêntico, apesar da restrição adicional DE do critério DEAT. Nota-se nesta figura um grande desfalque na representação dos padrões de entrada no segundo quadrante (os padrões de entrada podem ser vistos na Figura (5.7) (e)). A ausência de nodos em grande parte deste quadrante ocorreu devido ao modo como o algoritmo trata nodos sem vizinhos, eliminando-os. Tais nodos não conseguiram cumprir, devido às restrições, um mínimo de vizinhanças para que permanecessem vivos.

Este definitivamente não deveria ser um problema para o mapeamento correto do espaço de

entradas. Com esta simulação, foi encontrado um problema estratégico no próprio algoritmo. Não é possível utilizar a mesma definição de eliminação de nodos do GWR se o critério de vizinhança foi modificado de modo a restringir vizinhanças mesmo de novos nodos. Tais nodos inseridos acabam sendo imediatamente eliminados na seqüência. Faz-se notar a importância do Balbuciamto Motor para solucionar o problema de mapeamento incompleto do espaço de trabalho.

A Tabela (5.3) resume a evolução do treinamento dessas redes mostrando o número de nodos e de conexões para cada simulação, nos intervalos de 100, 200, 400 e 600 iterações.

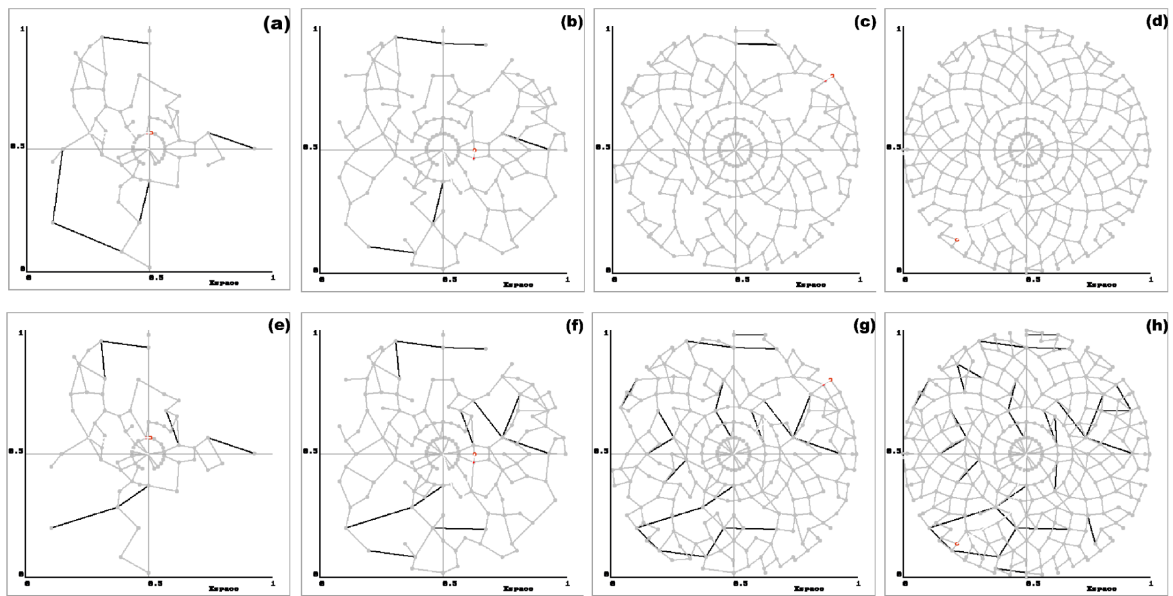


Figura 5.5 Fase de treinamento do STRAGEN-ON: (a) 5.6%. (b) 11.1%. (c) 22.2%. (d) 100.0%. Fase de treinamento do STRAGEN-OFF: (e) 5.6%. (f) 11.1%. (g) 22.2%. (h) 100.0%.

A evolução do treinamento do STRAGEN completo, com Balbuciamto Motor (MB), pode ser vista na Figura (5.5). Em ambos, STRAGEN-ON e STRAGEN-OFF, o ganho de se utilizar o Balbuciamto Motor é nítido por duas principais razões: Primeiro, o treinamento adquire uma característica fractal, ou seja, passa a representar o espaço de trabalho de modo homogêneo desde as primeiras iterações, ganhando em precisão a cada passo, e podendo ser interrompido a qualquer momento com resultado satisfatório para toda a área do espaço (Fritzke, 1997a). Segundo, o mapeamento passa a se estender a todo o espaço de trabalho, resolvendo o problema de posições do espaço não representadas por ter seus respectivos nodos apagados pela falta de vizinhança. Com o MB, a ordem em que os padrões são apresentados facilita a criação de nodos conectados já com seus vizinhos próximos.

Em especial, na Figura (5.5) (d) vê-se o mapa criado pelo STRAGEN-ON já pronto para ser utilizado ou modificado *online*, pois as conexões desnecessárias são apagadas durante o treinamento. Na Figura (5.5) (h) as conexões de cores pretas são as candidatas a serem eliminadas pela fase de poda do STRAGEN-OFF. A Tabela (5.4) mostra um comparativo entre número de nodos e conexões, incluindo informações após a poda.

Tabela 5.4 Comparativo entre número de nodos e conexões entre o STRAGEN-OFF e o STRAGEN-ON.

%		STRAGEN-OFF	STRAGEN-ON
5,6%	nodos	72	72
	conex.	95	90
11,1%	nodos	144	144
	conex.	204	190
22,2%	nodos	237	237
	conex.	365	332
100,0%	nodos	289	289
	conex.	504	444
Fase de Poda	nodos	289	–
	conex.	443	–

O mapa formado pelo STRAGEN pode ser impresso no plano XY de três formas, para facilitar a visualização: (a) O eixo das abscissas *versus* ordenadas formadas pela posição espacial do efetuador (V_1); (b) abscissas e ordenadas representando os ângulos das juntas J_1 e J_2 (V_2); (c) e as abscissas e ordenadas como os torques nas juntas J_1 e J_2 (V_3). Veja a Figura (5.6) o mapa aprendido pelo STRAGEN com o critério DA, plotado das três formas.

5.1.2.2 Geração da Trajetória (Fase 2)

Após a realização dos treinamentos da rede, passa-se para a fase de geração de sua trajetória. Dois algoritmos foram testados para esta fase: o algoritmo de difusão e o algoritmo de menor caminho.

- *Geração de Trajetórias por Difusão:*

Foi escolhido para cada simulação uma trajetória que levasse o braço de uma posição próxima ao centro do espaço (que será chamado de *nodo inicial*, e está representado por um círculo preto) para uma posição na extrema direita-inferior do terceiro quadrante (denominado *nodo final*, representado por um círculo branco).

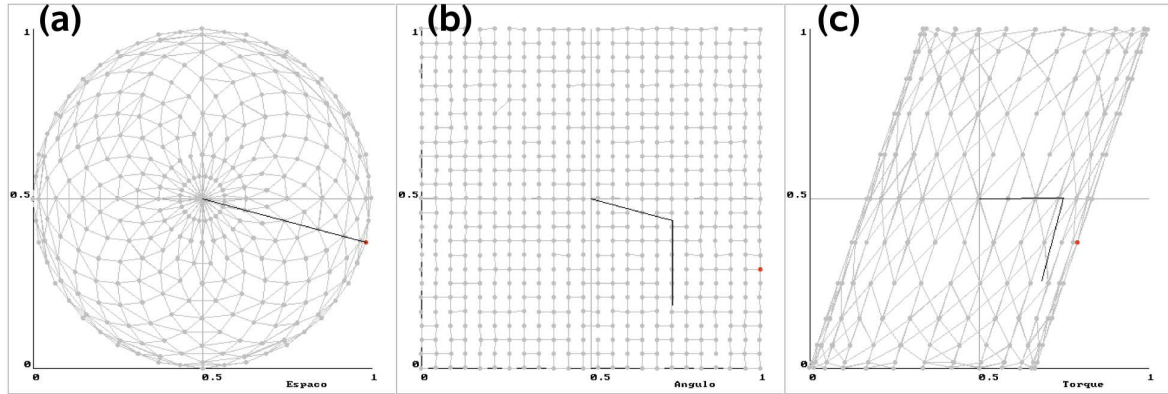


Figura 5.6 Gráficos do mapa com diferentes representações para as abscissas e ordenadas. (a) (X_1, X_2) , (b) (θ_1, θ_2) e (c) (τ_1, τ_2) .

Este algoritmo permite a entrada de dois pontos quaisquer no espaço de entrada, e automaticamente seleciona como nodo inicial o nodo vencedor (mais próximo espacialmente) do primeiro ponto de entrada, e o nodo final o nodo vencedor do segundo ponto de entrada.

Uma vez selecionados os pontos desejados, inicia-se o processo de difusão de energia. Este processo inicia sempre pelo *nodo final* (branco), que terá disponível sempre uma (1) unidade de energia, da qual frações passarão a seus vizinhos e aos vizinhos dos vizinhos, sucessivamente, até que uma pequena fração chegue ao *nodo inicial*.

Na Figura (5.7), as tonalidades de cinza representam parcelas de energia que foram dispendidas, tendo mais energia o nodo mais escuro (com exceção do nodo final, que é branco e é o nodo de maior energia $E = 1$). O processo inicia com o nodo final da trajetória, o círculo branco, difundindo sua energia pela rede, de vizinho para vizinho. O processo termina quando o nodo inicial da trajetória (representado por um círculo preto) recebe algum valor diferente de zero provindo do nodo final, diretamente conectado ou por intermédio de seus vizinhos diretos ou indiretos. A figura em questão foi captada ao

Tabela 5.5 Legenda: tons de cinza para Difusão de Energia.

Tom	Intervalo de Energia
	Nodo final. $E = 1$
	$E = [6 \cdot 10^{-1}; 1[$
	$E = [3 \cdot 10^{-2}; 6 \cdot 10^{-1}[$
	$E = [4 \cdot 10^{-3}; 3 \cdot 10^{-2}[$
	$E = [4 \cdot 10^{-4}; 4 \cdot 10^{-3}[$
	$E = [2 \cdot 10^{-5}; 4 \cdot 10^{-4}[$
	$E = [2 \cdot 10^{-6}; 2 \cdot 10^{-5}[$
	$E = [1 \cdot 10^{-7}; 2 \cdot 10^{-6}[$
	$E = [1 \cdot 10^{-8}; 1 \cdot 10^{-7}[$
	$E =]0; 1 \cdot 10^{-8}[$
	$E = 0$
	Nodo inicial. $E \approx 0$

final do processo de difusão para as simulações com critérios DE, DA (similar ao DEA), DT, DET e DEAT (similar ao DAT) apresentadas anteriormente. Os tons de cinza dos nodos re-

presentam intervalos de valores de energia que o nodo possui no momento, e sua legenda está apresentada na Tabela (5.5).

Ainda na Figura (5.7) (a) referente à simulação com critério DE, pode-se observar que para esta simulação o algoritmo facilmente encontra o *nodo inicial* (círculo preto no centro), partindo do *nodo final* e difundindo a energia de vizinho a vizinho. A maior parte da rede sequer recebe alguma parcela de energia (círculos claros). Uma vez atingido o nodo inicial, o algoritmo de difusão é terminado, pois tem-se uma trajetória ligando ambos os nodos. No caso do critério DE, as vizinhanças denotam os nodos mais próximos espacialmente em relação à posição do efetuador e o algoritmo acha a menor rota possível em número de nodos (mesmo que ocorreram ambigüidades na escolha de trechos da trajetória). A trajetória é mostrada na Figura (5.8) (a), pelos círculos brancos vazados, e a direção é do movimento a ser aplicado é do nodo inicial (círculo preto no centro) para a borda.

Entretanto, encontrar a menor rota em número de nodos pode não ser o objetivo principal de um determinado problema ou domínio. No caso de trajetórias de braços robóticos, nem sempre o melhor caminho é o mais curto. Na Figura (5.9) (a), tem-se as configurações para cada nodo da trajetória, da posição aprendida pela rede para o braço atingir tal posição. A posição inicial do braço é dada pelo posicionamento do efetuador no nodo 1 (nodo inicial, representado nas figuras anteriores por um círculo preto), e seu cotovelo posicionado no ponto A. A posição final desejada para todas as trajetórias é dada pelo posicionamento do efetuador no nodo 9 (representado nas figuras anteriores como um círculo branco), estando seu cotovelo no ponto C.

Continuando na Figura (5.9) (a), pode-se reparar que, partindo da posição inicial do braço, o cotovelo encontra-se na linha da base do primeiro quadrante (ponto A). Durante o percurso desta trajetória, o cotovelo mantém-se fixo no ponto A enquanto a mão sai do nodo 1 e caminha suavemente até o nodo 4, simplesmente abrindo o ângulo do cotovelo (ou junta 2). Neste momento, o próximo nodo (o nodo 5) da trajetória tem uma configuração aprendida para o cotovelo posicionado no ponto B. Este *desaparecimento* do cotovelo do ponto A e seu consecutivo *reaparecimento* no ponto B não é um movimento possível de ser realizado por um robô real, pois houve um salto das juntas devido ao critério de vizinhança adotado. Isto mostra que a vizinhança entre os nodos 4 e 5 da trajetória não deveria existir neste caso, pois apesar de serem nodos espacialmente próximos em relação à posição do efetuador, o modo como o robô aprendeu a atingir estes nodos tem configurações muito distintas. Deve-se buscar alternativas a esse critério de vizinhança do nodo mais próximo, o que acarretou o estudo dos próximos critérios a serem apresentados.

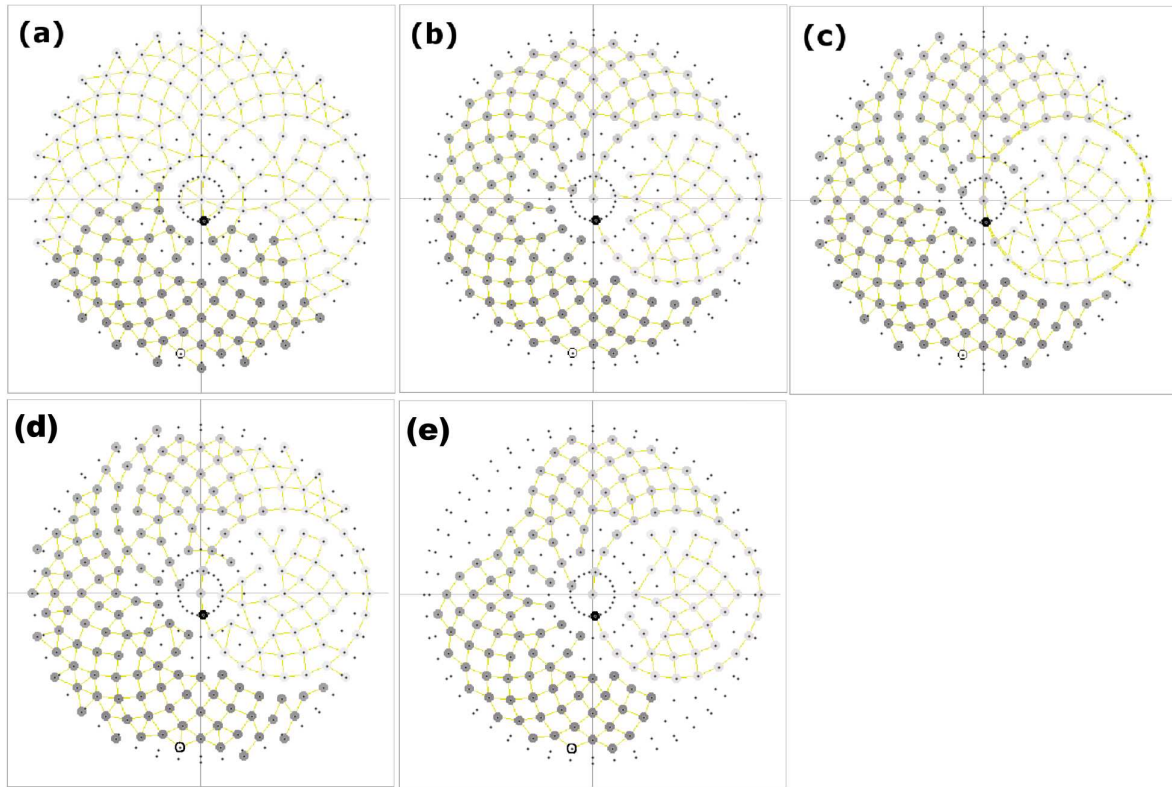


Figura 5.7 A rede STRAGEN gera uma trajetória utilizando-se do algoritmo de difusão. (a) Critério de vizinhança DE; (b) Critério de vizinhança DA; (c) Critério de vizinhança DT; (d) Critério de vizinhança DET (análogo ao DT); (e) Critério de vizinhança DEAT (análogo ao DAT).

Na Figura (5.7) (b), referente à simulação com critério de vizinhança DA, a difusão necessitou preencher praticamente todo o mapa topológico, dando uma volta por cima, e em seguida fazendo um círculo externo no primeiro quadrante. Isso ocorreu pelo fato de não haver um caminho do primeiro quadrante para o terceiro passando pelo quarto quadrante (como houve na mesma figura, parte (a), a ligação entre os nodos 4 e 5). Assim que a energia atinge o ponto inicial da trajetória, o gerador de trajetórias é executado e mostra, na Figura (5.8) (b), a trajetória formada (círculos vazados). Pode-se observar pelo posicionamento dos braços do robô, na Figura (5.9) (b), como o critério DA produz uma trajetória circular de movimentos menos bruscos, muito mais natural. Para o critério DEA (combinação de DE e DA), acrescentar a restrição DE à restrição DA, não influenciou a característica do mapa, uma vez que a restrição DA era forte o suficiente para se sobrepôr em todas as comparações do algoritmo à restrição DE. Isto nem sempre é verdade para outros domínios.

A Figura (5.7) (c) refere-se à difusão aplicada no mapa resultante da simulação com critério de vizinhança DT. Uma característica que pode ser observada durante a difusão foi o fluxo de

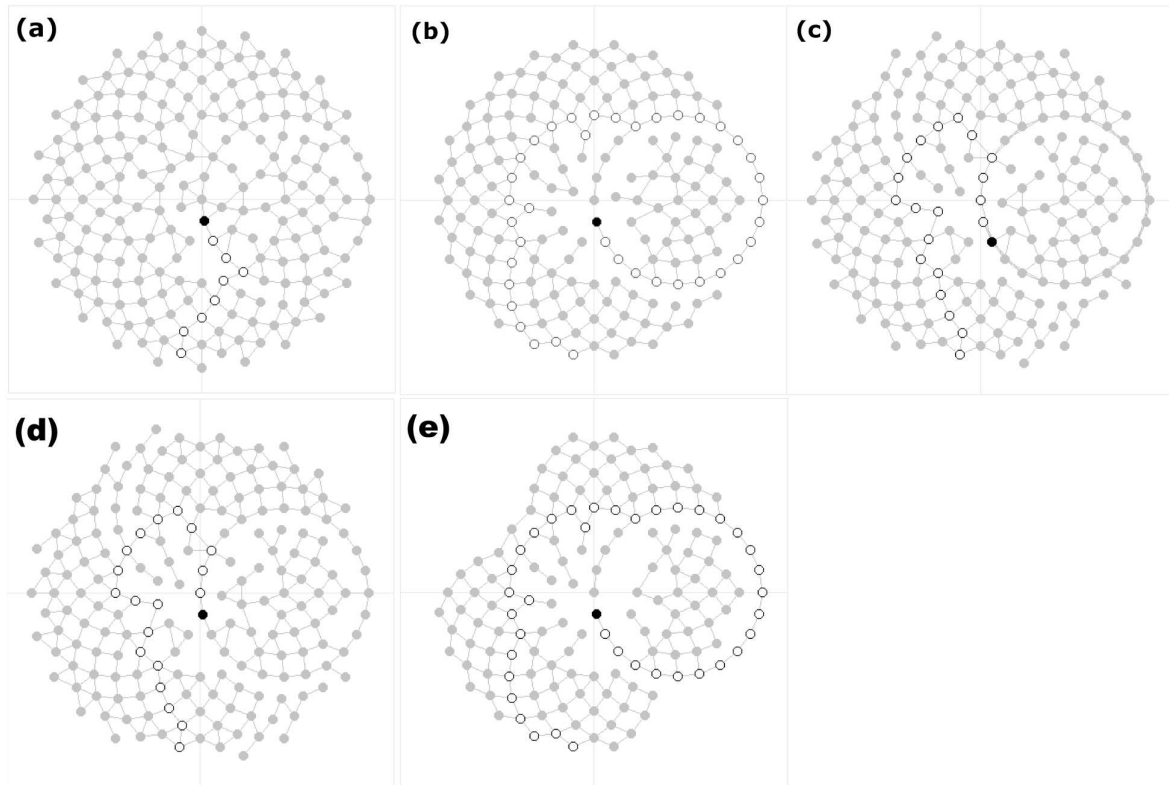


Figura 5.8 A trajetória criada pela rede STRAGEN, caracterizada pelos nodos brancos, tendo como nodo inicial o ponto preto ao centro: (a) Critério de vizinhança DE; (b) Critério de vizinhança DA (análogo ao critério DEA); (c) Critério de vizinhança DT; (d) Critério de vizinhança DET; (e) Critério de vizinhança DEAT (análogo ao critério DAT).

energia encontrar um caminho passando pelo centro (origem), devido às conexões que ali se fizeram e com isso, o trajeto foi encurtado, como se pode observar na Figura (5.8) (c). Os braços na Figura (5.9) (c) mostram ainda que ao minimizar a variação do torque, o trajeto próximo do ponto final tendeu rapidamente ao ponto meta ou alvo, com movimentos mais suaves. Com o critério DT, a trajetória que surge tem a característica de manter o braço sempre se movendo com o mínimo esforço, o que é uma característica da evolução natural.

Nas Figuras (5.7) (d), (5.8) (d) e (5.9) (d) vê-se a difusão, trajetória e a posição dos braços para o critério DET. A trajetória além de minimizar o torque, também minimiza a distância das posições do efetuador. A trajetória gerada para este caso foi semelhante a trajetória do critério DT.

Na Figura (5.7) (e) tem-se o mapa gerado pelo critério de vizinhança DEAT. Este mapa é semelhante ao gerado pela simulação com o critério DAT. Apesar da rota criada (ver Figura (5.8) (e)) ser idêntica às das simulações com critério DA e com critério DEA, isto ocorreu

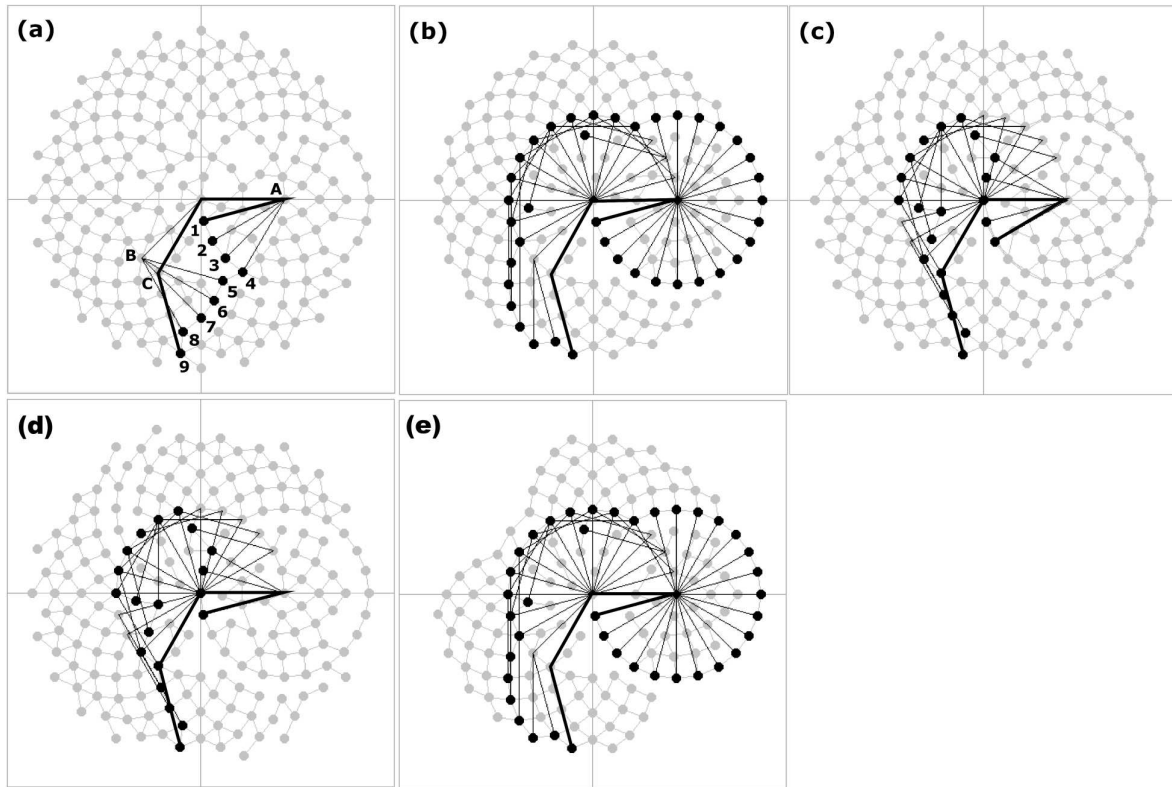


Figura 5.9 Posicionamento dos braços no movimento dado pela trajetória criada pela rede STRAGEN: (a) Critério de vizinhança DE; (b) Critério de vizinhança DA (análogo ao critério DEA); (c) Critério de vizinhança DT; (d) Critério de vizinhança DET; (e) Critério de vizinhança DEAT (análogo ao critério DAT).

devido à escolha dos nodos iniciais e finais. Porém, pode-se perceber que o mapa gerado é diferente, mais especificamente, o mapa desta figura teve muitos pontos que não foram mapeados com precisão adequada, devido a restrições que impedirem a criação de nodos com vizinhanças nestes locais (visíveis principalmente no segundo quadrante). Na Figura (5.9) (e), pode-se comparar os movimentos do braço para esta rota.

O algoritmo de difusão, como implementado, é um algoritmo de busca global. Sua busca consiste em abrir em amplitude diversas camadas de energia, a partir do nodo final, até que uma camada finalmente atinja o alvo criando a trajetória. O ponto fundamental do modelo STRAGEN está no mapeamento dinâmico e com critérios, que permite se aplicar a ele um algoritmo de busca. O algoritmo de Dijkstra do menor caminho (também chamado de algoritmo de menor custo), é utilizado abaixo e substitui o algoritmo de difusão no modelo STRAGEN-ON.

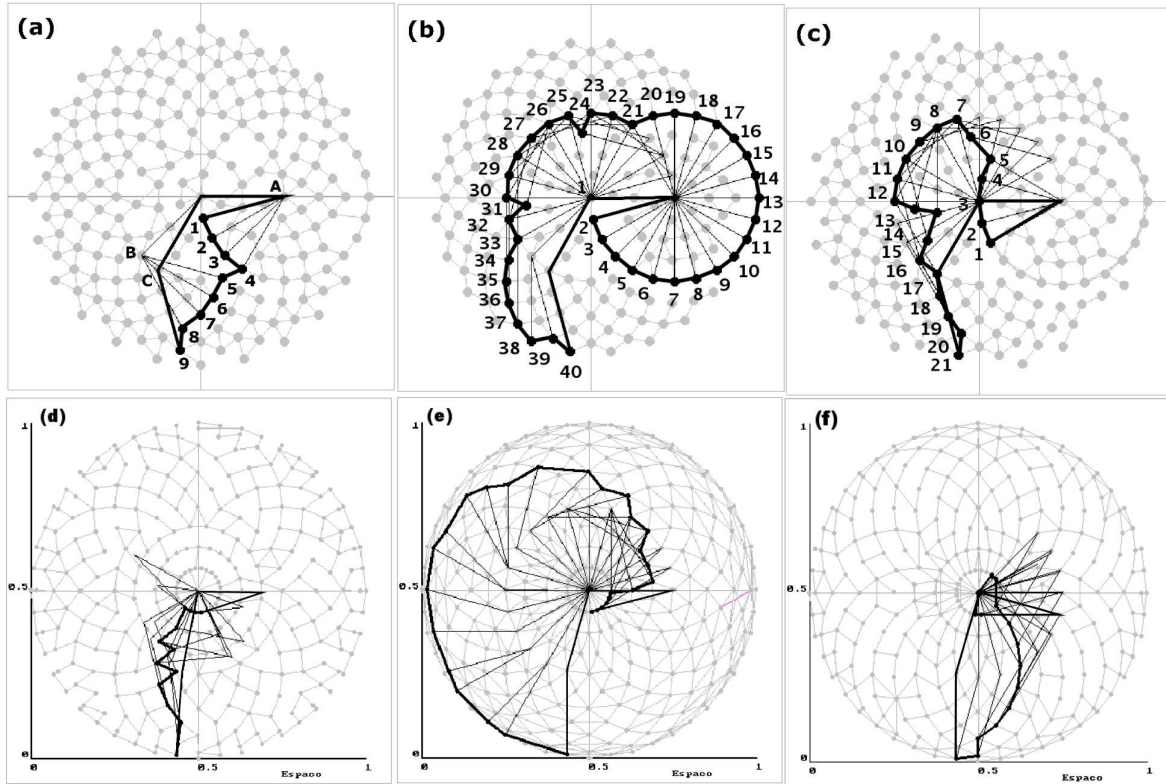


Figura 5.10 Trajetórias do robô 2D: (A) STRAGEN-OFF, critério DE; (B) STRAGEN-OFF, critério DA; (C) STRAGEN-OFF, critério DT. (D) STRAGEN-ON, critério DE; (E) STRAGEN-ON, critério DA (F) STRAGEN-ON, critério DT.

- *Geração de Trajetórias por Menor Caminho:*

O algoritmo Dijkstra foi utilizado na versão *on-line* do modelo STRAGEN, em substituição ao método de difusão. Ambos métodos fazem uma busca *global* no grafo, e por isso não sofrem de mínimos locais. No algoritmo Dijkstra, a busca se faz pela avaliação do custo de movimento de cada nodo. O STRAGEN representa o custo de transição de um nodo por outro pela distância Euclidiana entre os subgrupos que compõe o critério de vizinhança do vetor de pesos destes nodos. Desta forma, um mapa criado com o critério, por exemplo, de mínima distância espacial do efetuador (V_1) terá a trajetória minimizada por esse grupo (análogo aos outros critérios).

A Figura (5.10) (a) apresenta uma trajetória criada para minimizar o critério DE, para o STRAGEN-OFF, comparável à Figura (5.10) (d) de mesmo critério para o STRAGEN-ON. Nesta mesma figura, as subfiguras (b) e (e) mostram uma longa trajetória em que a minimização da variação dos ângulos (critério DA) é desejada, para os mesmos pontos iniciais/finais, respectivamente para o STRAGEN-OFF e STRAGEN-ON. Por fim, as subfiguras (c) e (f) representam

trajetórias criadas para minimizar a variação do torque (critério DT), para o STRAGEN-OFF e STRAGEN-ON.

5.1.2.3 Desviando de Obstáculos

Devido à sua característica dinâmica na representação da topologia do espaço de trabalho, o STRAGEN pode ser treinado por um subconjunto válido do espaço de trabalho e assim aprender sobre áreas factíveis deste espaço, sem necessidade de se conhecer *a priori* a densidade da topologia (ou número de nodos necessários para uma representação de qualidade). Um exemplo de área descontínua é dado por Fritzke (1997a). Esta área é apresentada para diversas redes: os mapas de Kohonen (Figura (3.5)), o modelo TRN (Figura (3.9)), o modelo GCS (Figura (3.13)) e o modelo GNG (Figura (3.14)).

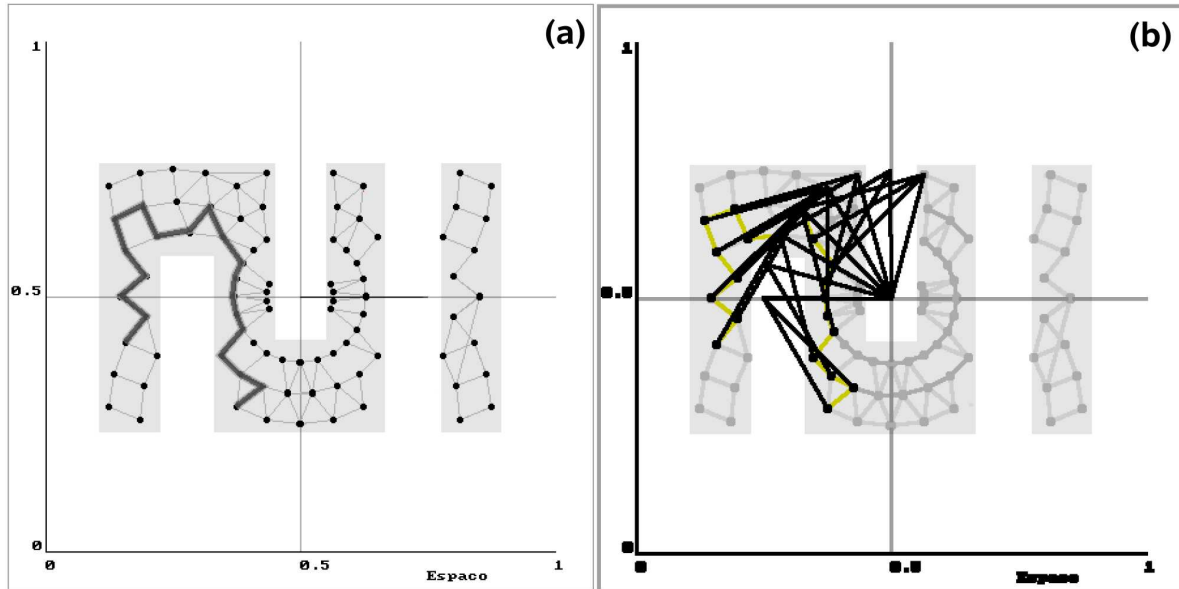


Figura 5.11 (a) Rede STRAGEN treinada com o critério DA, e espaço de trabalho válido somente na área sombreada. A linha escura representa a trajetória gerada. (b) Posicionamento do braço no percurso da trajetória gerada.

A Figura (5.11) (a) apresenta esta mesma área descontínua no espaço de trabalho do robô bidimensional considerado nesta seção. Os nodos treinados pelo STRAGEN, diferente das redes mencionadas, contém além da posição espacial da amostra do espaço de trabalho outras informações importantes, por exemplo para este domínio, informações sobre ângulos das juntas e torques. Com estas informações é possível gerar trajetórias válidas, desviando de obstáculos estáticos, e ainda assim minimizando o critério utilizado no treinamento do STRAGEN.

A rede TRN, que possui uma topologia estática (Figura (3.10)) foi utilizada em processo de mapeamento semelhante (Zeller et al., 1997).

A Figura (5.11) (b) mostra uma trajetória com a restrição de manter o efetuador na da área válida (cinza). Outros critérios poderiam ser utilizados para treinar a rede e manter além do efetuador também o cotovelo. É possível generalizar esse procedimento para outros domínios, como de robôs móveis, desde que se consiga uma representação adequada do espaço de trabalho destes robôs. O domínio dos robôs móveis não foi abordado neste trabalho, mas é tratado em diversas pesquisas envolvendo redes neurais artificiais (Nehmzow, 2001; Glasius et al., 1995).

5.1.3 Comparações entre STRAGEN, GNG e GWR

Os modelos GNG e GWR foram treinados com a mesma base de dados para serem comparados com o modelo STRAGEN. Para o GNG os parâmetros utilizados foram $\lambda = 10$ e $a_{\max} = 100$ e para o GWR, $a_T = 0,97$ e $a_{\max} = 25$. A base de dados foi apresentada ao GNG e ao GWR utilizando o procedimento original, i.e., as amostras apresentadas aleatoriamente. O STRAGEN utiliza o Balbuciamiento Motor como estratégia para apresentar amostras da base em uma ordem que facilita a criação de conexões adequadas ao critério de vizinhança utilizado.

Os modelos GNG e GWR criam seus vizinhos baseados em informação dada pela distância Euclidiana do vetor de pesos completo. Por isso estes modelos não distinguem o que seria uma vizinhança adequada para diferentes critérios baseados em grupos de informações homogêneas do vetor de entrada.

A base de dados foi apresentada 3 vezes ($t_{\max} = 1800 = 3 \cdot L$) para os três modelos (GNG, GWR e STRAGEN – versões OFF e ON), com ruídos aplicados nas amostras. O ruído r de $R\%$ é calculado como $r = R/100 \cdot I(i)$, sendo $I(i)$ uma função que retorna um valor aleatório no intervalo $[-i, i]$. Este ruído é então aplicado a todos os componentes $\xi'_k = r \cdot \xi_k$, $1 \leq k \leq D$ do padrão de entrada $\xi \in \mathbb{R}^D$. O ruído aplicado foi de $R = 0,1\%$ no intervalo de $[-1, 1]$. O número de iterações para a fase de poda e validação foi também $I = 1800$. A topologia criada por estes modelos para representar o espaço de estados pode ser estimada pela Tabela 5.6.

Em todas as simulações feitas o STRAGEN mostrou melhor erro de validação e distribuição de nodos no espaço de estados.

O STRAGEN cria novos nodos na exata posição do padrão apresentado, enquanto que o GNG e o GWR criam nodos baseados na média dos nodos vencedores e vices. Como o vetor de pesos é formado por informações heterogêneas (posições, ângulos e torques) necessárias para o cálculo da cinemática e dinâmica diretas e inversas, ocasiona-se que o cálculo da média feitos pelo GNG e GWR para o vetor inteiro faz com que as informações ali armazenadas sejam

Tabela 5.6 Comparação entre GNG, GWR e STRAGEN.

	GNG	GWR	S-OFF	S-ON
Erro Valid.	0,00384	0,01464	0,00094	0,00006
Num. nodos	601	344	289	289
Num. conex.	939	386	499	444

distorcidas durante o treinamento.

Na Tabela 5.7, $|\mathbf{V}_i|$ representa o tamanho do membro do manipulador bidimensional, definido como um . O processo de aprendizagem e inserção de novos nodos feitos pelo GNG e GWR nos seus vetores de peso mudam o comprimento do braço do robô. O STRAGEN mantém intacta esta informação porque se restringe a operar os vetores de peso de acordo com os grupos de vizinhança que o divide.

Tabela 5.7 Tamanho dos Membros.

Membro 1	X_1	Y_1	$ \mathbf{V}_1 $
GNG	-0,408000	-0,116000	0,42416978
GWR	0,011800	-0,198200	0,19855095
STRAGEN	-0,707001	0,707001	0,99984999
Membro 2	X_2	Y_2	$ \mathbf{V}_2 $
GNG	-0,759600	0,089200	0,40709901
GWR	-0,251200	-0,113600	0,27627190
STRAGEN	-0,213960	-0,159000	0,99651593

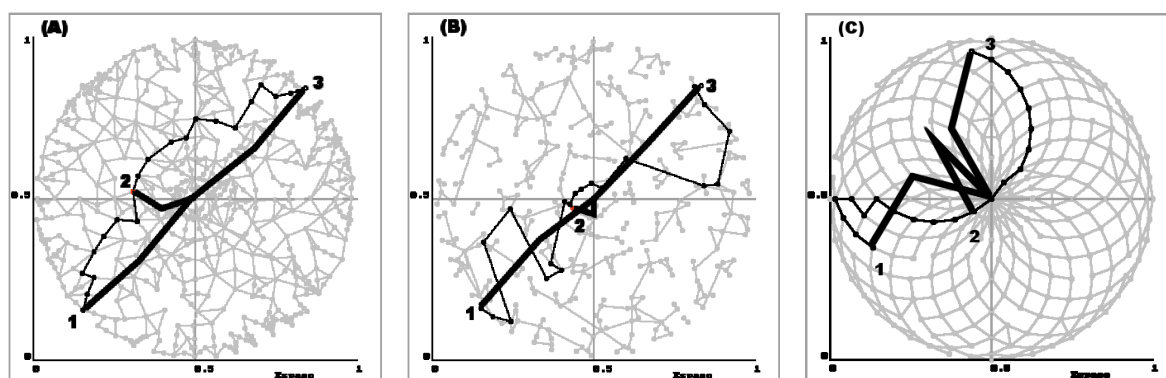
Na Tabela 5.8 pode-se ver uma comparação do vetor peso de cada modelo com o valor real que a informação deveria estar representando após calcular a cinemática direta e a dinâmica dadas pelos ângulos θ_1 e θ_2 . É fácil de perceber as consequências negativas de se misturar as informações heterogêneas do vetor peso do modo como o GNG e o GWR fazem.

A deformação do tamanho dos membros do robô após o treinamento do GNG e do GWR fica clara na Figura (5.12) (a) e (b) respectivamente. Nestes modelos, a metodologia de situar novos nodos, atualizar o vetor de pesos e determinar os vencedores provoca violações das restrições (deformação dos membros), enquanto que o STRAGEN pode lidar com tais restrições. A Figura (5.12) (c) mostra uma trajetória iniciando no ponto 1, visitando alguns pontos intermediários, passando pelo ponto 2 e terminando no ponto 3.

Na seção seguinte o modelo STRAGEN será utilizado para a geração de trajetórias de um manipulador tridimensional industrial PUMA-560.

Tabela 5.8 Distorção das Posições e Torques.

Modelo	θ_1	θ_2	X_2	Y_2	τ_1	τ_2
GNG	185	-24	-0,760	0,089	-11,71	-3,52
Valor real			-1,944	0,232	-29,40	-9,47
GWR	223	-37	-0,251	-0,114	-2,39	-2,62
Valor real			-1,729	-0,779	-24,63	-9,95
STRAGEN	135	165	-0,208	-0,156	-9,12	5,00
Valor real			-0,208	-0,161	-9,18	5,02

**Figura 5.12** Trajetórias do robô 2D: (A) GNG; (B) GWR; (C) STRAGEN-OFF.

5.2 Simulações para Manipulador PUMA-560

O mesmo processo de criação do robô bidimensional no simulador MATLAB foi utilizado para a criação do objeto matemático `p560` que representa o robô PUMA-560. O simulador (Corke, 1996) do robô PUMA560 considera parâmetros como viscosidade das juntas, centro de gravidade dos membros, momento de inércia, inércia da armadura do motor, de modo que os dados colhidos são próximos do real. Uma vez inicializado, utiliza-se as funções do simulador para se obter os dados necessários à criação de uma base de dados consistente com os valores de operação reais do robô, contendo posições das juntas, ângulos e torque necessários para o treinamento do STRAGEN.

Três bases de dados $\mathbf{B}$ de pontos válidos foram criadas, de dimensões $L \times D$ tais que: (a) Uma base de teste com $L_1 = 4096$ pontos, (b) uma base completa com $L_2 = 15625$ pontos e (c) uma base de precisão apenas do espaço de trabalho da frente do robô, com $L_3 = 6859$ pontos. A Figura (5.13) (a) mostra a base L_2 , e na mesma figura, (b) mostra a base L_3 .

A base completa L_2 é feita a partir do cruzamento de todas as possíveis combinações dadas

pela faixa dos respectivos ângulos das juntas tomadas em cinco pontos:

$$\theta_1 = \{-160; -80; 0; 80; 160\}, \theta_2 = \{-225; -157,5; -90; -22,5; 45\},$$

$$\theta_3 = \{-45; 22,5; 90; 157,5; 225\}, \theta_4 = \{-110; -40; 30; 100; 170\},$$

$$\theta_5 = \{-100; -50; 0; 50; 100\} \text{ e } \theta_6 = \{-266; -133; 0; 133; 266\}$$

perfazendo um total 5^6 , ou seja, 15625 pontos de treinamento em todo o espaço de trabalho do robô.

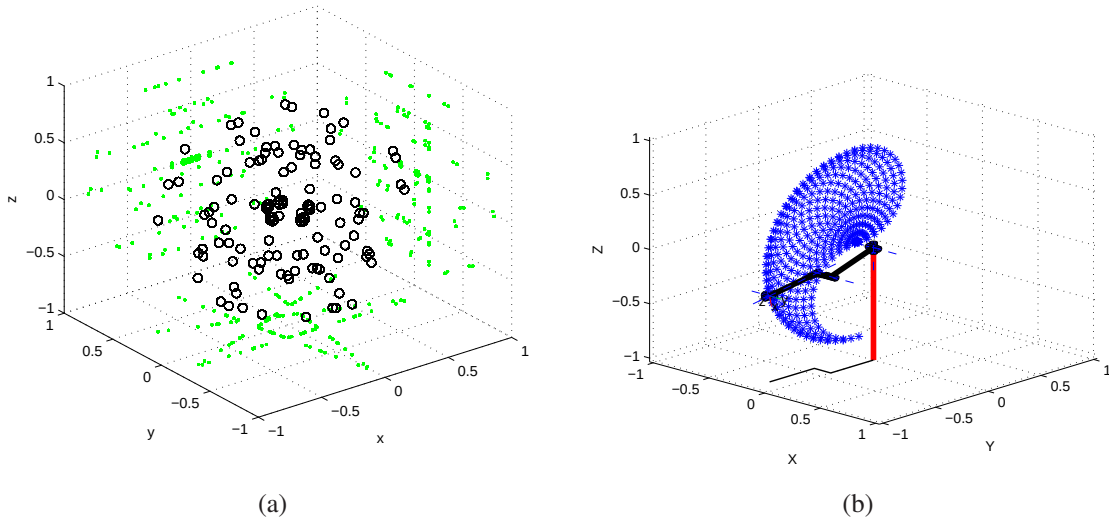


Figura 5.13 Nuvem de pontos de treinamento para PUMA-560. (a) Base completa com 15625 pontos. (b) Base de precisão com 6859 pontos em área restrita na frente do robô.

5.2.1 Configuração do STRAGEN

Tendo-se definida a base de dados, configura-se o STRAGEN dimensionando-se o vetor de pesos e os grupos de informações homogêneas que o forma. Cada posição do manipulador é definida como a posição espacial, os ângulos e os torques de cada junta, perfazendo um vetor de dimensão $D = 30$, i.e., $\mathbf{w} \in \mathbb{R}^{30}$:

$$\mathbf{w}_{n_i} = [X_1 \ Y_1 \ Z_1 \ X_2 \ \dots \ Z_6 \ \theta_1 \ \dots \ \theta_6 \ \tau_1 \ \dots \ \tau_6]^T$$

O vetor de pesos é dividido em $m = 3$ grupos homogêneos de informação, subdivididos cada um em 6 juntas: $\mathbf{V}_1 = \left[[X_1 \ Y_1 \ Z_1]_1^T \ \dots \ [X_6 \ Y_6 \ Z_6]_6^T \right]^T$, $\mathbf{V}_2 = \left[[\theta_1]_1^T \ \dots \ [\theta_6]_6^T \right]^T$, $\mathbf{V}_3 = \left[[\tau_1]_1^T \ \dots \ [\tau_6]_6^T \right]^T$.

As dimensões são $D_1 = 18$, $D_2 = 6$ e $D_3 = 6$, e as dimensões dos subgrupos são $D_{1,i} = 3$ para $1 \leq i \leq 6$, e $D_{j,i} = 1$ para $2 \leq j \leq 3$ e $1 \leq i \leq 6$.

O critério para a primeira simulação foi a minimização da distância espacial da posição das juntas $\mathbf{V}_\eta = [\mathbf{V}_1]$, e o grupo de atividade (utilizado para determinar a inserção de um novo nodo) apesar de poder ser distinto, também foi escolhido como $\mathbf{V}_\xi = [\mathbf{V}_1]$. Note que $[\mathbf{V}_1]$ representa todas as 6 juntas do manipulador. Se apenas a posição espacial do efetuador fosse desejada, o critério deveria ser escolhido como $\mathbf{V}_\eta = [\mathbf{V}_{16}]$.

A porcentagem usada para o cálculo do limiar de atividade $\bar{a}$ para cada junta foi $P = 1\%$, o que dá $\bar{a}_i = \exp(0,01 \cdot \sqrt{D_{1,i}}) = 0,01732$, $\forall i \leq i \leq 6$. Isto significa que os nodos respondem por uma área de 1% do espaço de trabalho, dentro da qual o nodo deve aprender e se ajustar aos padrões que ali incidirem. Isto dá robustez ao modelo, pois permite que ruídos sejam eliminados através do aprendizado.

5.2.2 Resultados do STRAGEN

A Figura (5.14) mostra 12 posições de uma trajetória gerada pelo STRAGEN-OFF com as configurações explanadas acima, e com a base de dados de 4096 pontos. As posições geradas pela rede são todas válidas, cada nodo contendo informações sobre posição, ângulos, e torques das juntas, permitindo à rede mover o braço sem a necessidade de cálculos de cinemática ou dinâmica inversas. Apesar de apresentar um resultado preciso, com erro de validação de 0,002770, o algoritmo aprendeu apenas 64 posições e gastou 4min44s para completar o treinamento. A versão STRAGEN-ON mostrada na Figura (5.15) também foi treinada com o critério de menor variação da posição espacial das juntas, utilizando a base completa de 15625 diferentes posições (ângulos das juntas), o que permitiu a rede aprender todas as 125 posições distintas, com um erro de validação de 0,000091031049 e com o treinamento completado em apenas 1min44s (ou seja, mais de 90% em ganho de velocidade, considerando o aumento do número de nodos da base de dados). A Tabela 5.9 sumariza estas diferenças.

Uma vez que o erro de validação mede a qualidade dos dados aprendidos em relação aos dados apresentados (reais/ruídos), o ganho observado é substancial. A explicação para o ganho apresentado na Tabela 5.9 é simples: devido ao fato do STRAGEN-ON ser mais rápido, é possível usá-lo para treinar o mesmo espaço de problema com uma base de dados mais completa ($\mathbf{B}_3$), o que leva a uma topologia mais acurada. A Figura (5.15) mostra uma trajetória com os mesmos pontos inicial/final da Figura (5.14), agora para o STRAGEN-ON. Apesar do número de nodos do mapa topológico criado pelo STRAGEN-ON ser maior que aquele criado pelo STRAGEN-OFF, não há garantias (nem necessidade) de que a trajetória encontrada con-

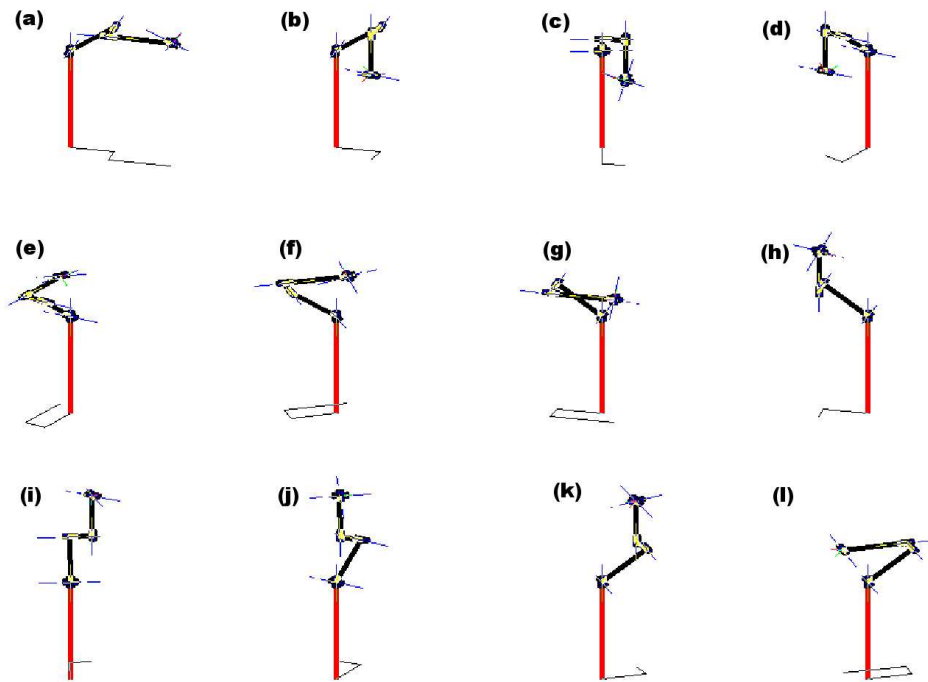


Figura 5.14 STRAGEN-OFF, critério: posição espacial de todas as juntas. Trajetória do PUMA-560 iniciando em (a) e terminando em (l). Base L_1 , erro de validação $2,770 \cdot 10^{-3}$.

Tabela 5.9 STRAGEN-OFF e STRAGEN-ON: critério da menor distância espacial da juntas.

	STRAGEN-OFF	STRAGEN-ON
Erro Valid.	$2,770 \cdot 10^{-3}$	$9,103 \cdot 10^{-5}$
Num. Nodos	64	125
Num. Conex.	102	179
Tempo	4min44s	1min44s
Base de Dados	4096	15625
Iterações	12288	46875

tenha um maior número de nodos. No caso desta simulação, o STRAGEN-ON encontrou uma trajetória de 9 posições, desde o ponto inicial até o ponto final, enquanto que o STRAGEN-OFF encontrou uma trajetória de 12 posições.

A Figura (5.16) mostra a trajetória de 9 posições geradas pelo STRAGEN-ON usando a base de dados B_2 , com critério de vizinhança dado por $V_\eta = [V_2]$, i.e., a minimização da variação dos ângulos das juntas. Todos os padrões foram apresentados à rede 3 vezes cada (46875 iterações). A Tabela 5.10 mostra que a versão *off-line* é ainda mais precisa, quando o treinamento é feito em igualdade de condições, devido ao fato de que os nodos e conexões não

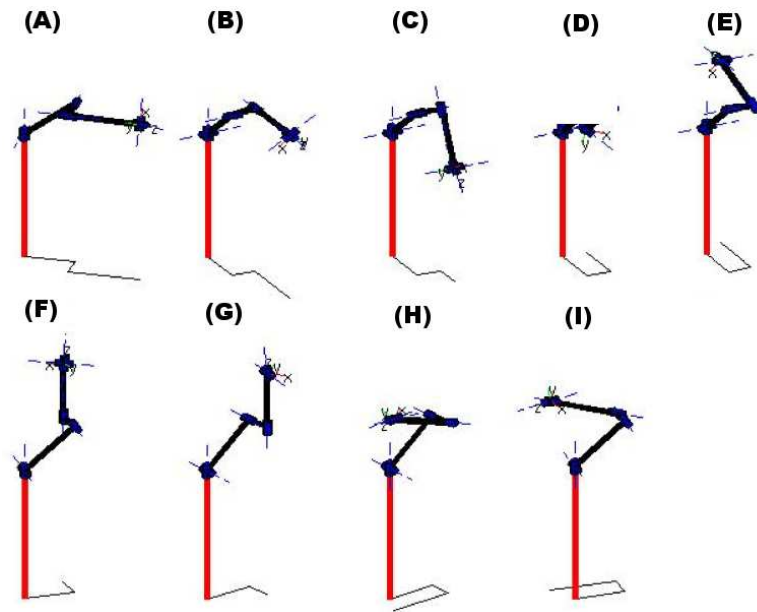


Figura 5.15 STRAGEN-ON, critério: posição espacial de todas as juntas. Trajetória do PUMA-560 iniciando em (a) e terminando em (i). Base L_2 , erro de validação $9,103 \cdot 10^{-5}$.

são removidos durante o treinamento, o que faz com que a topologia seja mais robusta. Em contrapartida, a versão *on-line* apresenta um ganho na velocidade de treinamento de até 91%, sem perda de precisão considerável, além de dispensar a fase de poda (*off-line*) que permite o uso da rede mesmo durante o treinamento.

Tabela 5.10 STRAGEN-OFF e STRAGEN-ON: critério de mínima variação dos ângulos das juntas.

	STRAGEN-OFF	STRAGEN-ON
Erro Valid.	$2,60 \cdot 10^{-5}$	$1,11 \cdot 10^{-4}$
Num. Nodos	15625	15625
Num. Conex.	40139 / 19084 *	31395
Tempo	502min16s	48min2s
Base de Dados	15625	15625
Iterações	46875	46875

A Figura (5.17) (a) mostra as mesmas posições da Figura (5.16) sobrepostas. E a Figura (5.17) (b) mostra a linha da trajetória gerada pelo STRAGEN-ON, percorrida pelo robô nesta simulação, do ponto (a) até o ponto (i).

* Antes/Depois da fase de poda.

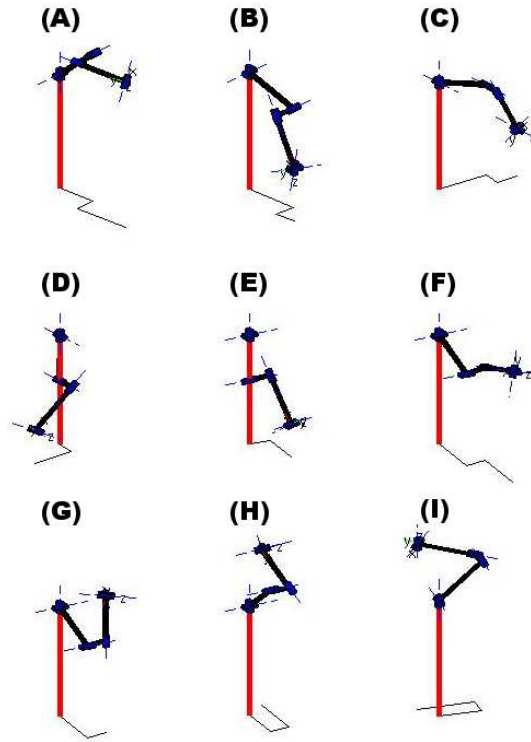


Figura 5.16 STRAGEN-ON, critério: mínima variação do ângulo das juntas. Trajetória do PUMA-560 iniciando em (a) e terminando em (i). Base L_2 , erro de validação $1,11 \cdot 10^{-4}$.

5.2.3 Trajetória Via-pontos

Uma característica importante da geração de trajetória de estados é a capacidade de se criar trajetórias que passam por pontos pré-determinados, chamados *via-pontos*. Uma simulação com a base de dados $\mathbf{B}_3$ e o critério de mínima variação das posições espaciais das juntas foi realizado, para demonstrar esta capacidade. A Figura (5.18) mostra duas trajetórias semelhantes: (a) A primeira *não* utilizou qualquer restrição, e o modelo STRAGEN-ON retornou uma trajetória em que o manipulador se movimenta pelos pontos baixos do espaço de teste. (b) Na segunda trajetória (figura à direita), o robô foi instruído a passar pelo ponto (C). Percebe-se que os pontos (A) e (B) são os mesmos em ambas trajetórias, e a partir do ponto (C), na trajetória *via-pontos*, o caminho de menor custo passou a ser aquele que utiliza os pontos da parte de cima do espaço de teste.

5.2.4 Comparações entre STRAGEN, GNG e GWR

A simulação do PUMA-560 com a base de dados $\mathbf{B}_3$, de $L_3 = 6859$ pontos, da seção anterior foi também utilizada para treinar os modelos GNG e GWR, além do STRAGEN-ON, com o

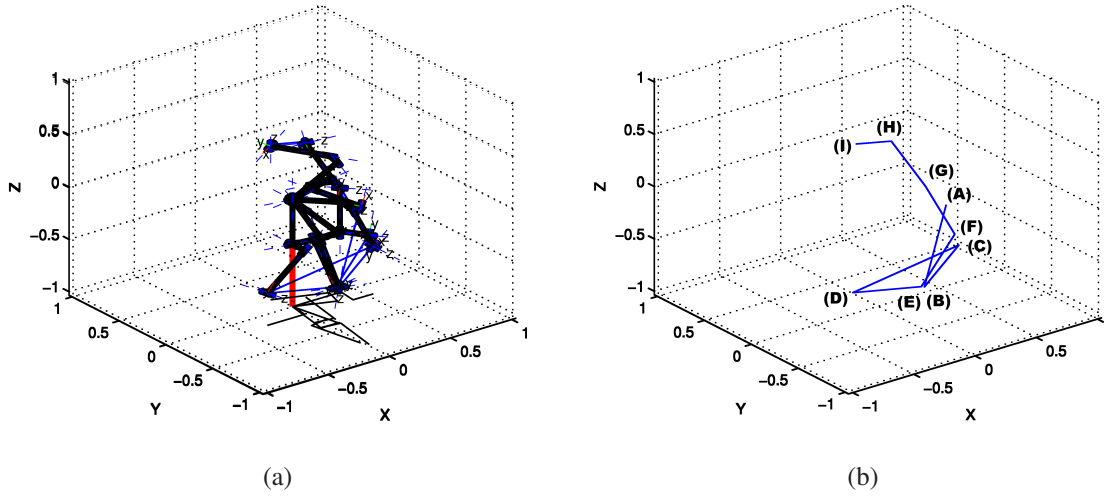


Figura 5.17 STRAGEN-ON: critério: mínima variação angular. (a) Posições do braço do PUMA-560. (b) Trajetória realizada.

critério de mínima variação de ângulos das juntas $\mathbf{V}_1 = \mathbf{V}_2$, com o intuito de se comparar os erros de validação e o erro E'_1 para o tamanho médio das conexões criadas.

A Figura (5.19) (a) mostra a evolução do erro de validação para os três modelos. Analisando essa figura, percebe-se que a vantagem do STRAGEN está na rápida identificação do sistema no início de seu treinamento, o que o torna apto para ser utilizado quase que imediatamente ao início da simulação. Ao decorrer do tempo, o algoritmo GWR se aproxima em precisão do STRAGEN, ficando ambos estabilizados e com melhor resultado que o algoritmo GNG.

Para medir a qualidade da representação topológica destas redes, a Equação (5.3) foi aplicado aos três modelos. Esta equação de medida de erro, chamada de E'_1 , foi inspirada na medida proposta por Marsland et al. (2002) (Eq. 3.37), e avalia o tamanho médio de todas as conexões:

$$E'_1 = \frac{\sum_i \sum_{j < i} C_{i,j} \cdot \|w_i - w_j\|^2}{|\mathcal{C}|} \quad (5.3)$$

onde:

$$C_{i,j} = \begin{cases} 1 & \forall (i,j) \text{ conectado} \\ 0 & \text{c.c.} \end{cases}$$

e $|\mathcal{C}|$ é o número total de conexões do conjunto $\mathcal{C}$.

A Figura (5.19) (b) mostra a medida de erro E'_1 . Nota-se que o STRAGEN mantém o erro E'_1 praticamente constante, iniciando próximo de 0,29 e estabilizando próximo de 0,055. A Tabela (5.11) mostra alguns pontos deste gráfico.

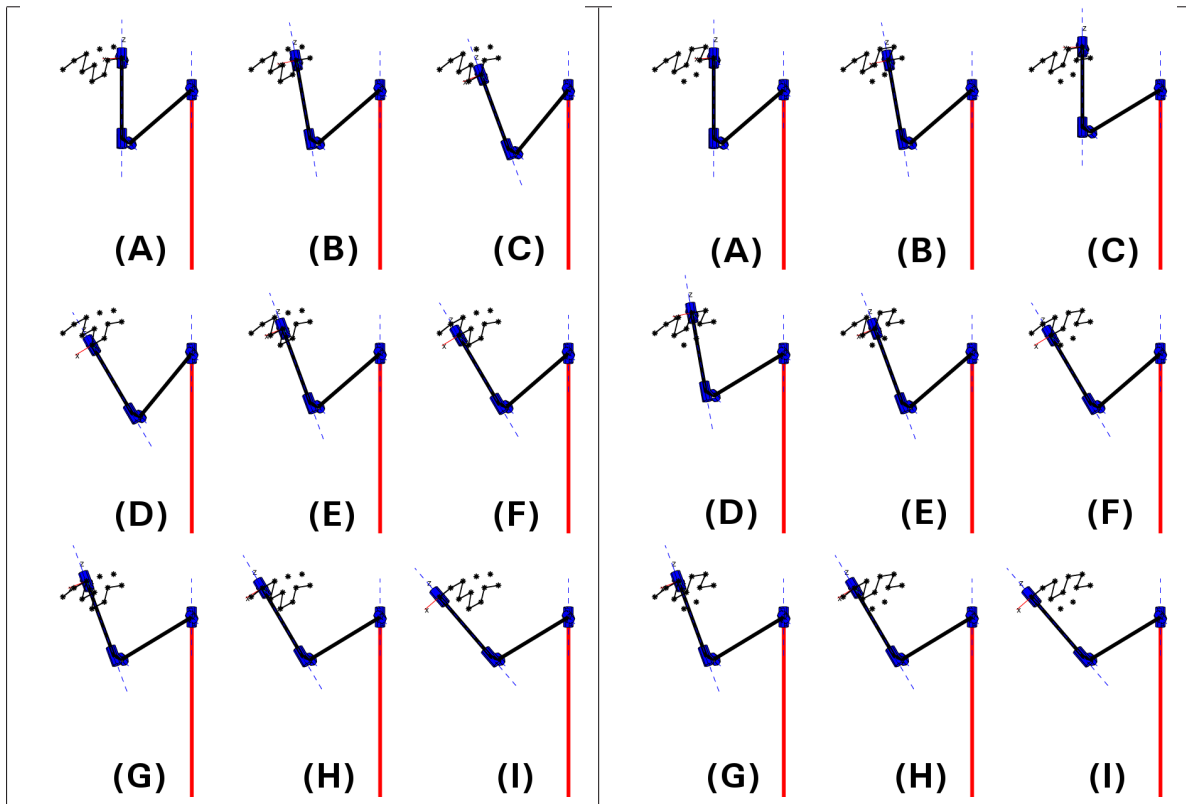


Figura 5.18 STRAGEN-ON via-pontos: *Esquerda*: sem restrições. *Direita*: ponto (C) obrigatório.

Tabela 5.11 Erro E'_1 para GNG, GWR e STRAGEN-ON.

Iteração	GNG	GWR	STRAGEN
0	1,846596	1,845965	0,299287
2000	0,145532	0,250477	0,070423
10000	0,088729	0,105758	0,056240
20577	0,062778	0,055378	0,055575

Deve-se lembrar que apesar do erro apresentado pelo GWR ficar próximo do STRAGEN, o algoritmo GWR consegue essa convergência a custo da distorção das informações contidas no mapa topológico pelo seu processo de aprendizado e inserção de nodos.

A Figura (5.20) (a) mostra o crescimento do mapa topológico para as redes GNG, GWR e STRAGEN. O modelo GNG tem um crescimento linear, baseada na taxa λ . Também o modelo STRAGEN apresenta em princípio uma taxa de crescimento linear, porém esse crescimento é dependente do surgimento de novos padrões ainda não treinados, e é maximizado pelo procedimento de Balbuciamiento Motor. Uma vez que os nodos representaram com precisão os padrões

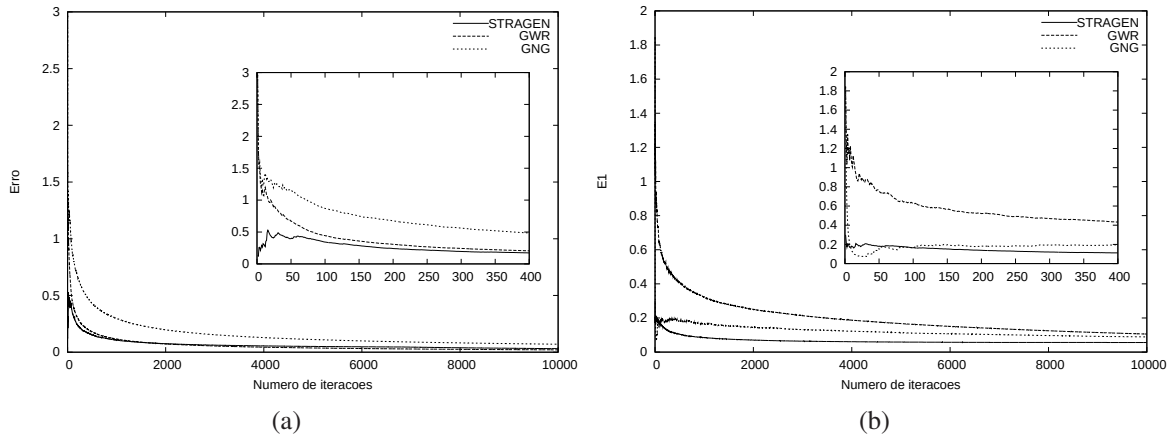


Figura 5.19 (a) Erro de validação do STRAGEN, GWR e GNG. (b) Erro E'_1 .

apresentados, o crescimento do STRAGEN estabiliza imediatamente, só voltando a se alterar no caso do surgimento de novos padrões. O sistema de crescimento do mapa topológico do GWR faz com que haja uma compensação entre criar um novo nodo ou mover um nodo existente, e com isso sua curva de crescimento mostra-se suave. Uma das dificuldades do GWR é configurar seus parâmetros para que a curva tenha máximo aproveitamento para o domínio e base de dados em questão.

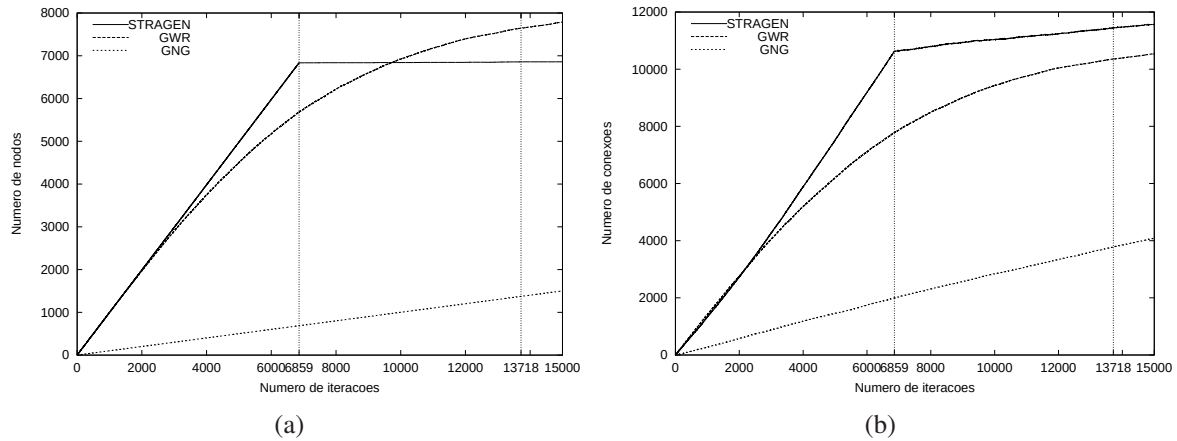


Figura 5.20 (a) Crescimento da rede em número de nodos por iteração. (b) Crescimento da rede em número de conexões por iteração.

Na Figura (5.20) (b) tem-se o crescimento do número de conexões do mapa topológico. Importante notar que o STRAGEN, mesmo após estabilizar o número de nodos necessários para cobrir o espaço de estados com boa precisão, continua a extrair similaridades dos padrões

apresentados e a criar (e também remover), numa taxa menor, conexões entre nodos próximos de acordo com o critério de vizinhança estabelecido.

5.3 Simulações para Mão Antropomórfica Kanguera

O protótipo da mão robótica antropomórfica *Kanguera* desenvolvido pelo Laboratório de Mecatrônica – EESC/USP apresenta desafios diferenciados dos manipuladores robóticos na área de geração de trajetórias de estados e estratégias de controle (Caurin et al., 2005). A Figura (5.21) mostra o protótipo e um esquema genérico de um dedo articulado.

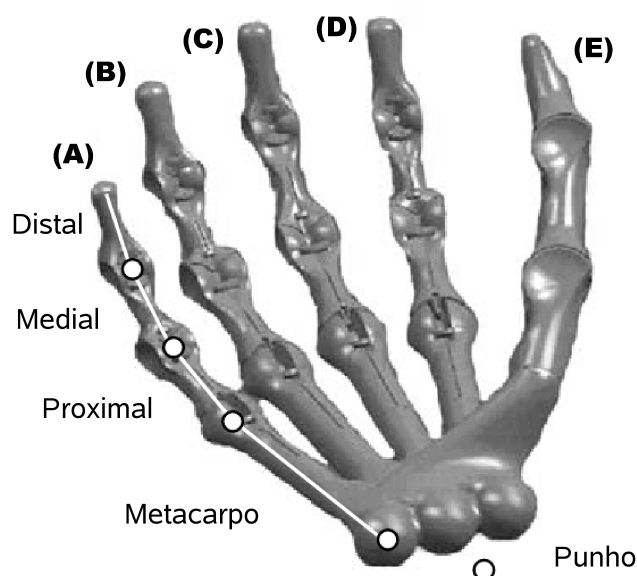


Figura 5.21 Articulações da mão robótica e esquema genérico de um dedo. (Caurin et al., 2004).

Para o domínio das mãos robóticas, o maior desafio consiste em pegar (*grasp*) e manipular objetos (reposicionar os dedos – *regrasp*) de modo ágil, em que a sincronia dos movimentos é essencial para o sucesso da tarefa (Borst et al., 2003a).

Nas seções seguintes serão apresentadas duas simulações que foram feitas utilizando-se o modelo STRAGEN-OFF para mostrar o movimento sincronizado dos dedos. Para treinar o modelo STRAGEN, a equipe da EESC/USP desenvolvedora do protótipo gentilmente gerou e enviou para o autor deste trabalho duas bases de dados com o espaço de estados da mão (a primeira com $L_5 = 1000$ e a segunda com $L_6 = 2000$ pontos para treinamento). Em um trabalho conjunto com esta equipe, após o treinamento e a geração de trajetórias de estados pelo

STRAGEN, tais trajetórias foram reenviadas para serem testadas diretamente no protótipo, e o resultado do movimento é apresentado a seguir.

5.3.1 Configuração do STRAGEN

De posse da base de dados, e conhecendo a dimensão do vetor de pesos bem como a ordem em que as informações estão agrupadas, o primeiro passo é configurar o vetor peso e seus grupos homogêneos. A mão Kanguera contém 20 graus de liberdade nos dedos e 2 no pulso. Para estas simulações o pulso foi mantido imóvel, movendo-se apenas os dedos. As informações do posicionamento dos dedos, dos ângulos das juntas e dos torques foi dividida em um vetor $\mathbf{w}_i \in \mathbb{R}^D$ de dimensão $D = 55$ tal que:

$$\mathbf{w}_i = [X_A \ Y_A \ Z_A \ X_B \ \dots \ Z_E \ \theta_{A_1} \ \theta_{A_2} \ \theta_{A_3} \ \theta_{A_4} \ \theta_{B_1} \ \dots \ \theta_{E_4} \ \tau_{A_1} \ \tau_{A_2} \ \tau_{A_3} \ \tau_{A_4} \ \tau_{B_1} \ \dots \ \tau_{E_4}]^T$$

As informações de diferentes categorias foram agrupadas em $m = 3$ grupos homogêneos, subdivididos por dedos e por grupos de juntas:

$$\begin{aligned} \mathbf{V}_1 &= \left[[X_A \ Y_A \ Z_A]_1^T \ [X_B \ Y_B \ Z_B]_2^T \ \dots \ [X_E \ Y_E \ Z_E]_5^T \right]^T \\ \mathbf{V}_2 &= \left[[\theta_{A_1} \ \theta_{A_2} \ \theta_{A_3} \ \theta_{A_4}]_1^T \ [\theta_{B_1} \ \dots]_2^T \ \dots \ [\theta_{E_1} \ \dots]_5^T \right]^T \\ \mathbf{V}_3 &= \left[[\tau_{A_1} \ \tau_{A_2} \ \tau_{A_3} \ \tau_{A_4}]_1^T \ [\tau_{B_1} \ \dots]_2^T \ [\tau_{E_1} \ \dots]_5^T \right]^T \end{aligned}$$

onde as dimensões são $D_1 = 15$ (e $D_{1,i} = 3, 1 \leq i \leq 5$), $D_2 = 20$ (e $D_{2,i} = 4, 1 \leq i \leq 5$), $D_3 = 20$ (e $D_{3,i} = 4, 1 \leq i \leq 5$), e $\sum_{j=1}^m D_j = D = 55$. Os índices de A a E representam os dedos mínimo, anular, médio, indicador e polegar, nesta ordem. Note-se que nesta base de dados somente a posição da ponta dos dedos está disponível. Os sub-índices dos índices dos dedos, de 1 a 4, representam as juntas entre: (1) metacarpo-falange proximal, (2) falange proximal-falange medial, (3) falange medial-falange distal, (4) ponta da falange distal (ou ponta do dedo).

5.3.2 Resultados do STRAGEN

Para a primeira simulação, a base de dados $\mathbf{B}_5$ contém $L_5 = 1000$ pontos igualmente espaçados no intervalo de possíveis movimentos desde uma configuração de mão fechada até uma configuração de mão aberta, com configurações de dedos fechando individualmente. Como padrão

das outras simulações, a taxa de aprendizado inicial foi de $\epsilon_b = 0,2$, o taxa final de aprendizado de $\alpha_f = 0,1$, o número de candidatos do Balbuciação Motor $Q = 10$, e o número estimado de disparos por nodo $\sigma_f = 6$, apesar de que o algoritmo mostra boa estabilidade quando esses parâmetros são variados. O grupo de atividades e o critério de vizinhança foi $V_\zeta = V_\eta = [V_1]$, que define a posição espacial da ponta dos dedos para minimização na representação topológica do espaço de estados a ser criado. Esta seleção do critério de vizinhança que define como o STRAGEN conectará os nodos e criará caminhos que minimizam o critério escolhido. O limiar $\bar{a}$ de ativação de um nodo foi, como nas simulações anteriores, definido como $P = 1\%$. Este percentual muda o raio de ação de um nodo, de modo que quanto maior, mais representativo (e menos preciso) um nodo é, e quanto menor, mais especializado o nodo (implicando que o STRAGEN precisa criar mais nodos para cobrir a área não representada).

No primeiro experimento a base de dados foi apresentada à rede por 3 vezes ($t_{\max} = 3000 = 3 \cdot L$) com 0,1% de ruído nas amostras. O número de iterações para a fase de poda e para a validação foi $I = 3000$.

Após o treinamento realizado, foi proposto ao STRAGEN como ponto de partida o ponto que representa a configuração de uma mão fechada, e o alvo a configuração da mão aberta, conforme mostra a Figura (5.22).

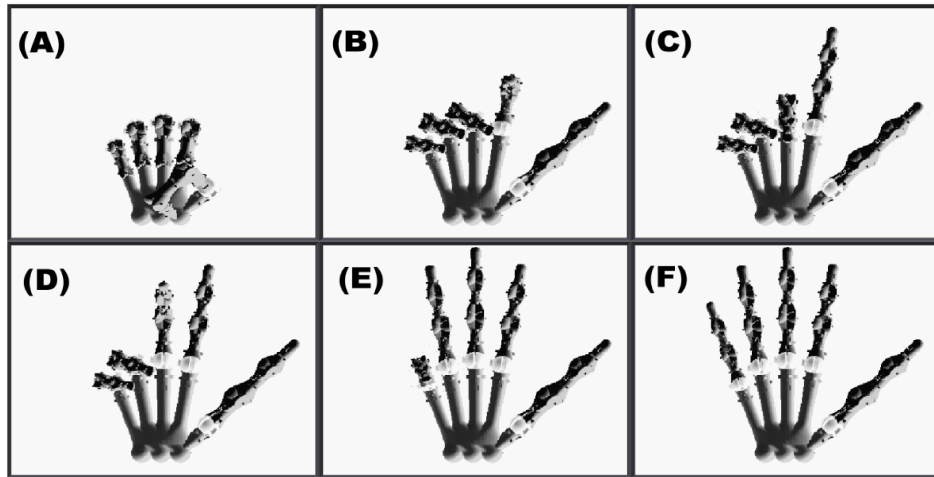


Figura 5.22 STRAGEN-OFF, Simulação 1: Trajetória de mão fechada até mão aberta, movendo dedos individualmente; Mapa com 425 nodos; Trajetória com 360 pontos; Base com $L_5 = 1000$ amostras.

STRAGEN gerou uma trajetória com 360 nodos de comprimento, que foi transferida para a mão robótica para realizar o movimento indicado. O erro de validação do mapa topológico treinado foi de 0,046118.

A segunda simulação utilizou uma base de dados de $L_6 = 2000$ pontos com diferentes

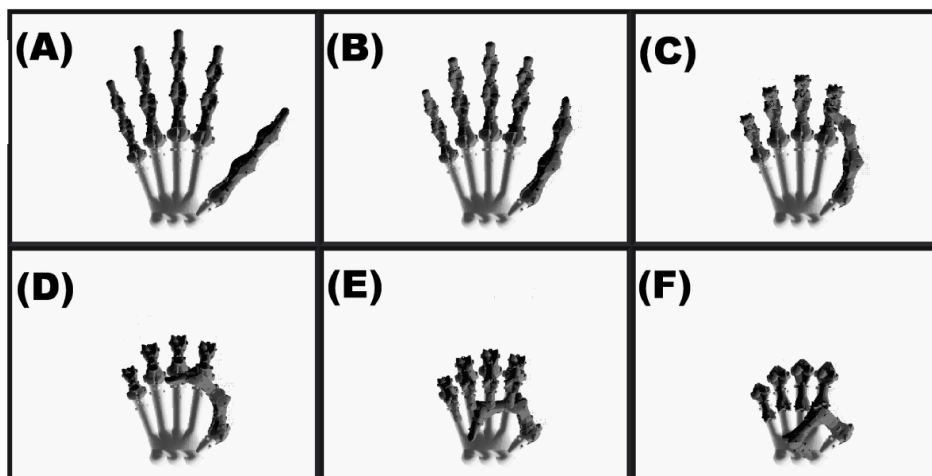


Figura 5.23 STRAGEN-OFF, Simulação 2: Trajetória de mão aberta até mão fechada, movendo dedos simultaneamente; Mapa com 600 nodos; Trajetória com 597 pontos; Base com $L_6 = 2000$ amostras.

configurações igualmente distribuídas no espaço de trabalho do robô Kanguera. A porcentagem P do raio de atividade que um nodo responde foi modificada para $P = 0,3\%$. Com isto, o STRAGEN criou mais nodos no mapa topológico para representar as amostras da base de dados. A mesma trajetória foi testada, invertida a ordem do ponto inicial e final, conforme mostra a Figura (5.23).

Na primeira simulação, a base de dados continha padrões em que a mão se configurava em posições em que os dedos se abriam de modo independente. Com isso, o STRAGEN gerou uma trajetória que abre a mão um dedo por vez. Como visto na Seção 5.2, uma outra maneira de se conseguir delinear os movimentos é utilizando trajetórias via-pontos. Na segunda simulação, a base de dados continha amostras de configurações da mão fechando os dedos em paralelo. Isto permitiu que o STRAGEN escolhesse um caminho mais curto para seu objetivo, fechando toda a mão simultaneamente. A característica de fechar os dedos simultaneamente, independente da distância angular ou distância espacial envolvida no movimento, terminando a trajetória de cada dedo independente simultaneamente aos outros dedos é chamada de *equifinalidade temporal*, conforme visto no Capítulo 2.

Por causa da menor porcentagem de atividade, e maior quantidade de pontos na base de dados, a trajetória gerada pelo STRAGEN continha mais pontos (precisamente 597). Isto refletiu um mapa mais denso criado pelo STRAGEN, com 600 nodos na segunda simulação, e apenas 425 nodos na primeira. Este simples parâmetro P , medido em porcentagem do espaço de estados, pode ser modificado à vontade para se escolher uma melhor precisão para um determinado domínio de problema, caso o padrão de 1% não apresente bom resultado. O erro de validação

para a segunda simulação foi de 0,008248.

Após o treinamento, a rede representou bem os ângulos das juntas aprendidos, com erro por junta menor que 0,0002%. A posição dos dedos para cada configuração aprendida teve uma variação máxima menor que 5,93178%.

5.4 Simulações para Modelagem de Progressões Harmônicas

É uma discussão comum e antiga na musicologia se as regras de harmonia são expressões culturais e artísticas ou se podem ser reduzidas a propriedades acústicas dos acordes. As regras de harmonia não são definições matemáticas, mas permitem que músicos, musicólogos e mesmo não-músicos percebam a tônica de uma música, acompanhem um padrão rítmico e sigam a direcionalidade das frases musicais (Martens et al., 2005). Experimentos com ERP (*Event-Related Potential*) sugerem que músicos e não-músicos apresentam respostas eletrofisiológicas similares a mudanças sutis na função harmônica de um acorde alvo (Tillmann et al., 2000).

Esta estrutura musical é muitas vezes comparada a gramáticas com sintaxe própria, e há trabalhos no sentido de formalizar esta sintaxe através do uso de gramáticas formais (Chemillier, 2004), para se gerar seqüências de acordes musicais. O cérebro humano é capaz de aprender estas regras e regularidades por simples exposição passiva, adquirindo o conhecimento implícito que influencia a percepção e a performance. Tillmann et al. (2000) propôs tentar aprender tais regularidades encontradas nas estruturas das músicas tonais ocidentais através do uso de mapas auto-organizáveis. Para o propósito de treinar a rede, Tillmann et al. (2000) decompôs os acordes em gradientes de frequências (*pitch-class*). Muitos modelos de redes neurais utilizam gradientes para a percepção de música tonal, classificação de acordes, aprendizado de seqüências de melodia, categorização entre outras, veja Tillmann et al. (2000) para mais detalhes.

As relações entre os acordes derivam de um conjunto de regras sintáticas que incluem aspectos tipológicos, como as diferentes estruturas de cada acorde, bem como aspectos morfológicos, como as resoluções e preparações, a força harmônica, etc. Se estas regras são fundamentadas em leis naturais ou meramente convenções é um tópico importante na musicologia desde a publicação do tratado de Rameau em 1722 (Rameau, 1971), em que este defende que as regras da progressão harmônica têm algum tipo de fundamentação física. Seguindo esta linha de raciocínio, Schoenberg (1999) tenta explicar a natureza das inversões triádicas pela própria constituição das séries harmônicas no seu importante tratado de harmonia publicado no início do século XX.

O apelo ao reducionismo da teoria da harmonia se explica pelo fato de que se a harmonia

puder ser reduzida a similaridades físicas entre os acordes, poder-se-ia conclamar que a música tonal é mais natural que quaisquer outros tipos de música, e empregar esse argumento para justificar a ubiquidade da música tonal na história da música (Rameau, 1971). No final da década de 80, início da década de 90, Leman (1991) sugeriu que os Mapas Auto-organizáveis de Kohonen (SOM) simulavam categorizações tonotópicas cocleares (Leman, 1991, 1990). Leman (1991) descobriu que redes SOM pareciam replicar as relações entre acordes previstas em estudos de harmonia, como por exemplo o extremamente conhecido *Ciclo de Quintas* (Figura (5.24)) que exprime as distâncias harmônicas entre os acordes maiores. Estes experimentos dão força à teoria da ontogênese da música tonal, i.e., a música tonal como fenômeno que surge do próprio aparato auditório, e não de expressões artísticas e culturais.

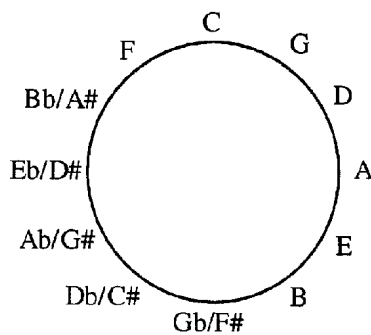


Figura 5.24 Ciclo de Quintas representando as distâncias entre os acordes maiores.

A Figura (5.25) mostra uma típica melodia acompanhada (ou monodia), para piano. Neste exemplo pode-se notar a ocorrência simultânea de duas estruturas: o pentagrama superior, executado pela mão direita apresenta a melodia (contínua seqüência de notas) e o inferior, executado pela mão esquerda, apresenta a harmonia em um ritmo padrão de valsa. É possível derivar a estrutura harmônica que está por trás desta seqüência de acordes:

Ab Db Bb Eb Eb7 Ab/Eb Eb7 Ab

ou, em notação harmônica tradicional

I IV II V V7 I64 V7 I.

Esta estrutura harmônica possui um estilo padronizado, do modo como uma frase cadencial deve ter. Na primeira parte da frase, ou na sua parte antecedente, inicia-se um acorde de tônica, seguido por um subdominante e um supertônico, culminando no acorde de dominante. Na última parte, ou na sua parte conseqüente, há uma cadência perfeita (I64, V7, I). O antecedente estabelece a tonalidade quando apresenta os graus mais importantes: a tônica, o subdominante e o dominante; e o conseqüente confirma a tonalidade com a cadência. O fato importante é que nesta simples e paradigmática frase há uma inequívoca tonalidade estabelecida, produzida por

meio de padrões de seqüências de acordes.



Figura 5.25 Melodia típica (Schubert, Op.9a).

Inclusive, a perfeita progressão harmônica da monodia da Figura (5.25) é decorrência do fato de todos os acordes estarem em posição fundamental, i.e., com a tônica na posição mais grave do acorde. No entanto, cada tríade tonal pode ser apresentada em 3 posições: fundamental, primeira inversão e segunda inversão. Nas inversões, ou a terça do acorde ou a quinta ocupam a posição mais grave, formando a primeira e a segunda inversões, respectivamente. Rameau foi o primeiro teórico a sustentar que as inversões não alteram a qualidade harmônica dos acordes (até o século XVIII duas inversões de um acorde eram consideradas dois acordes diferentes). Desde então, tal fato tem sido aceito generalizadamente, e um conjunto de regras foi sistematizada para o emprego de acordes invertidos em uma composição.

Para Schoenberg (Schoenberg, 1999), as regras para inversões de acordes são uma questão de constituição acústica ao invés de convenções sociais e artísticas (hipótese de Schoenberg). Oliveira et al. (2005) testou a hipótese de Schoenberg através de uma rede SOM que categorizou os acordes por suas propriedades acústicas (*pitchs*) e sugeriu que tal mapeamento não confirma a hipótese proposta. Porém uma limitação da rede SOM para esta análise é a de que a rede pode lidar apenas com os aspectos tipológicos da organização da harmonia. Para melhor investigar a hipótese de Schoenberg e o tratamento dos acordes na harmonia é preciso incluir aspectos morfológicos, e para isso se faz necessário utilizar uma rede capaz de tratar as transições dos acordes em um processo harmônico. O modelo STRAGEN é capaz de fornecer essa informação morfológica que faltava na análise das regras harmônicas.

Em conjunto com Núcleo Interdisciplinar de Comunicação Sonora (NICS/UNICAMP) foram realizados três simulações do STRAGEN-OFF, com diferentes bases de dados de acordes. Os acordes foram representados por um vetor de dimensão $D = 52$ de *pitch-classes* sem redução a oitavas, o que permite descrever os seis primeiros parciais harmônicos de cada nota dos acordes. *Pitch-class* é uma representação comum em teoria e psicologia da música (Tillmann et al., 2000; Bharucha, 1991; Griffith, 1999; Page, 1999), e é sobre ele que o argumento de Schoenberg é construído, portanto para verificar tal argumento é importante ser fiel à forma de representação por ele empregada.

As informações contidas na base são do tipo binárias, com valor 1 para sinalizar que o acorde representado pelo padrão possui a característica do parcial harmônico representado, e 0 se não possui. As 52 posições no padrão de entrada representam a distribuição dos parciais, conforme pode ser observado no exemplo do acorde Dó Maior na Tabela (5.12). Outros tipos de bases para representar o espectro harmônico foram analisadas e os resultados são semelhantes e condizentes com as análises de Schoenberg, razão pela qual se optou pela base binária que é de representação mais simples (Oliveira et al., 2005).

Tabela 5.12 Vetor representando a distribuição das parciais em *pitch-classes* para o acorde Dó maior.

c	C#	D	d#	e	F	f#	g	g#	A	A#	B	c	c#	d	d#	E	f	f#	g	G#	a	a#	B	c	c#
1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	1	0
d	d#	e	f	f#	g	g#	a	A#	B	c	c#	D	d#	e	f	f#	G	g#	a	a#	b	c	c#	d	d#
1	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

A equipe do NICS/UNICAMP gerou três bases de acordes que foram utilizadas para os experimentos. A primeira **B₇**, com $L_7 = 36$ acordes, sendo 12 acordes maiores na posição fundamental, 12 acordes maiores em segunda inversão e mais 12 acordes em primeira inversão. A segunda base **B₈** inclui além dos acordes da primeira, também os 12 acordes menores e os 12 acordes diminuídos em cada inversão, resultando em uma base com $L_8 = 108$ acordes. E a terceira base **B₉** composta de $L_9 = 21$ acordes da tonalidade do C maior na posição fundamental e nas inversões.

5.4.1 Configuração do STRAGEN

Com a definição da base de dados, o vetor de pesos do STRAGEN foi configurado para se adequar à sua dimensão:

$$\mathbf{w}_i = [w_{i1} \ w_{i2} \ \dots \ w_{i52}]^T$$

As informações do vetor peso, neste caso, são homogêneas, não havendo necessidade de configurar grupos para criar diferentes critérios de vizinhança. Neste caso, o número de grupos homogêneos é $m = 1$, de modo que:

$$\mathbf{V}_1 = [w_{i1} \ w_{i2} \ \dots \ w_{i52}]^T$$

e possui dimensão $D_1 = 52$.

5.4.2 Resultados do STRAGEN

Para a primeira simulação, utilizou-se a base $\mathbf{B}_7$ com $L_7 = 36$. Outros parâmetros mantiveram-se constantes, como nas simulações anteriores: taxa de aprendizado inicial foi de $\varepsilon_b = 0,2$, o taxa final de aprendizado de $\alpha_f = 0,1$, o número de candidatos do Balbuciação Motor $Q = 10$, e o número estimado de disparos por nodo $\sigma_f = 6$. Para o critério de vizinhança, como o vetor pesos contém apenas um tipo de informação homogênea, foi utilizado o vetor completo $\mathbf{V}_\eta = [\mathbf{V}_1] = \mathbf{w}_i$, de modo que a distância Euclidiana passa a ser computada para todo o vetor, como fazem as redes SOM tradicionais.

No primeiro experimento a base de dados foi apresentada à rede por 10 vezes ($t_{\max} = 360 = 10 \cdot L$) com 0,1% de ruído nas amostras. O número de iterações para a fase de poda e para a validação foi de $I = 360$.

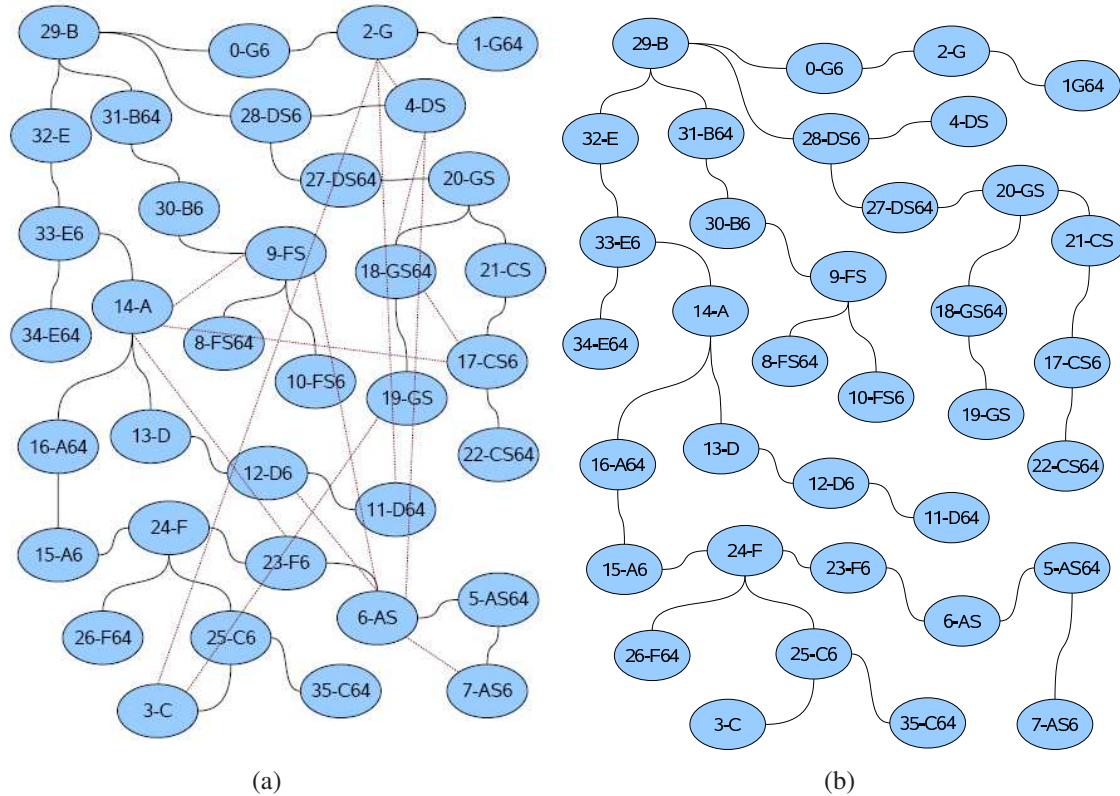


Figura 5.26 Experimento 1: Transições de estado da base $\mathbf{B}_7$. (a) Antes da poda; (b) após a poda.

O mapa aprendido pode ser visto na Figura (5.26) (a), antes da fase de poda do STRAGEN-OFF e (b) após a poda realizada. O resultado foi o esperado de acordo com a perspectiva psicoacústica-computacional, e um Ciclo de Quintas aparentemente emergiu da organização

dos acordes no mapa gerado. O erro de validação do mapa aprendido foi de 0,000215.

Foi solicitado ao STRAGEN que gerasse uma trajetória com acorde inicial **C** e acorde final **E64**. Como o conjunto de dados é relativamente pequeno, a trajetória pode ser acompanhada nas Figuras (5.26) e (5.27), acorde por acorde, no caminho harmônico gerado:

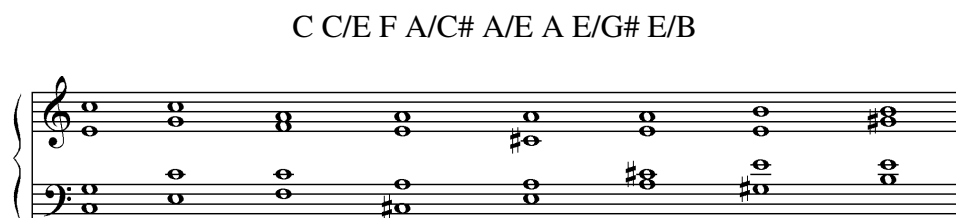


Figura 5.27 Partitura do experimento 1, usando **B**₇.

Apesar do mapa gerado ser coerente com agrupamentos de acordes na teoria acústico-musical, de acordo com a equipe do NICS/UNICAMP, e similar a resultados do modelo SOM (Oliveira et al., 2005), o experimento em si, i.e, a trajetória gerada não apresentou apelo musical, provavelmente devido à limitação da base de entradas, de apenas 36 acordes maiores. Para superar esta limitação, um novo experimento foi solicitado ao STRAGEN, com uma base de dados mais completa.

O experimento 2 foi realizado com a base de dados **B**₈ de 108 acordes, conforme descrito na Seção 5.4. A quantidade de estados aprendidos torna inviável a representação gráfica do mapa topológico. O erro de validação do mapa aprendido no experimento 2 foi de 0,000216. A trajetória requisitada ao STRAGEN conta com o ponto inicial, 5 pontos intermediários, e o ponto final: C → E64 → D#64 → C#6 → A6 → C# → C. A trajetória gerada pelo STRAGEN pode ser conferida em símbolos de acordes, em negrito os pontos inicial, final e intermediários:

[1]

C Eo D# D#6 D#64 D#m64 D#m6 D#m F# F#64
F#m64 F#m D D#o D#o6 D#o64 B G#m6 G#m E E6 **E64**

[2]

E6 E G#m G#m6 B D#o64 D#o6 D3o D F#m F#m64
F#m6 F# D#m D#m6 D#m64 **D#64**

[3]

D#m64 D#m6 D#m F# F#64 F#m64 F#m D D#o
D#o6 D#o64 B G#m6 G#m G#m64 G#64 G# Fm C# **C#6**

[4]

C# Fm G# G#64 G#m G#m6 B D#o64 D#o6

D#o D F#m A A64 A6

[5]

A64 A F#m D D#o D#o6 D#o64 B G#m6 G#m G#m64 G#64 G# Fm C#

[6]

Fm G# G#64 G#m64 G#m G#m6 B D#o64 D#o6 D#o D

F#m F#m64 F#64 F# D#m D#m6 D#m64 D#64 D#6 D# Eo C

Nesta trajetória, que também pode ser visualizada na partitura da Figura (5.28), não surgiu, à primeira vista, estruturas comuns de harmonia em nenhum dos trechos, como por exemplo frases do tipo I II V I que se poderia esperar caso se confirmasse a hipótese de Schoenberg. Apesar disso, não era incomum na idade média músicas com trechos retrógrados como os trechos que as partes [4] e [5] apresentam. Uma possibilidade para a não-corroboração da hipótese de Schoenberg talvez seja que no experimento 2, ao contrário do experimento 1, talvez se tenha dado muita liberdade ao modelo STRAGEN ao treinar uma base de dados tão completa.

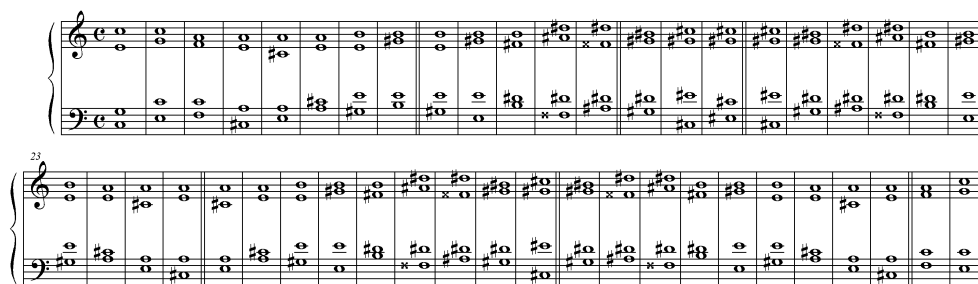


Figura 5.28 Partitura do experimento 2, usando B₈.

O experimento 3 foi concebido para testar trajetórias no espaço de estados caso se limite este espaço a apenas um conjunto de acordes de tonalidade C maior. O mapa topológico treinado pelo STRAGEN apresentou erro de validação de 0,000217. A trajetória via-pontos do experimento 3 requisitada foi: C → Em64 → G6 → Am64 → C. A trajetória gerada pelo STRAGEN é mostrada abaixo tendo em negrito os pontos inicial, final e intermediários:

[1]

C C6 C64 Em6 **Em64**

[2]

Em6 Em G **G6**

[3]

G Bd Bd6 Dm **F**

[4]

Am **Am64**

[5]

Am **F**

[6]

Dm Bd6 Bd G Em Em6 C64 C6 **C**

Esta trajetória também pode ser vista no mapa topológico mostrado na Figura (5.30) e na partitura correspondente, na Figura (5.29).

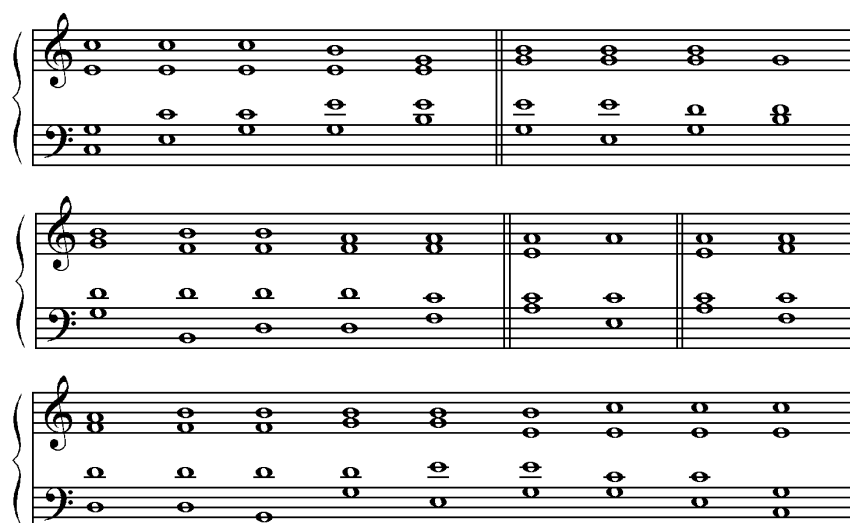


Figura 5.29 Partitura do experimento 3, usando B_9 .

Como era de se esperar, as trajetórias mínimas calculadas pela distância Euclidiana são as mesmas na ida e na volta. A parte [6] sintetiza as partes [3], [2] e [1], ao contrário, eliminando alguns acordes de inversão. Em uma análise comparativa dos três experimentos, pode-se concluir que dois padrões de conexões foram encontrados: (i) os acordes são seguidos por suas inversões; e (ii), os acordes são seguidos por relações de mediantes. Essas conclusões são coerentes com experimentos prévios feitos com redes SOM (Oliveira et al., 2005).

Uma análise musical e da hipótese de Schoenberg como resultado desses experimentos está fora do escopo deste trabalho. É importante, entretanto, notar que a rede STRAGEN pode e foi utilizada para a geração de trajetórias de estados no domínio deste problema, e se mostrou adequada como ferramenta na área de musicologia experimental, nesse caso investigando progressões harmônicas com acordes invertidos.

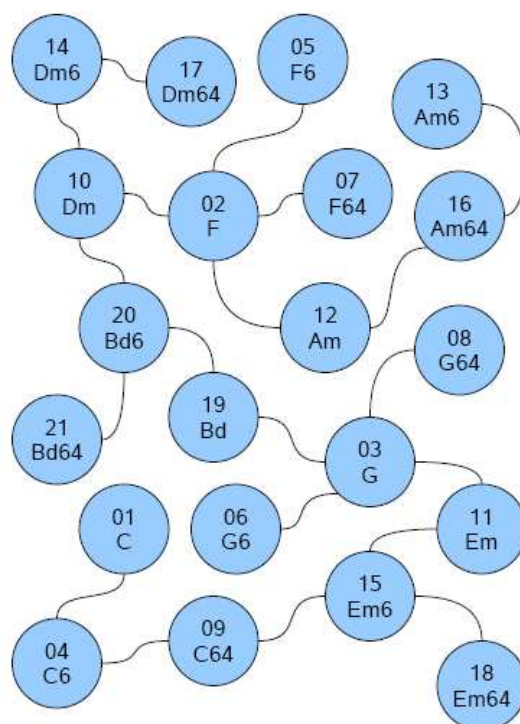


Figura 5.30 Experimento 3: Transições de estado aprendidas pelo STRAGEN-OFF para base de 21 acordes.

5.5 Conclusões

Nesta capítulo foram apresentadas as simulações feitas com o STRAGEN para quatro domínios distintos nas seções: (5.1) manipulador robótico bidimensional, também referenciado como “pêndulo duplo”; (5.2) manipulador robótico PUMA-560, um robô industrial com 6 graus de liberdade trabalhando no espaço; (5.3) mão robótica Kanguera trabalhando no espaço, desenvolvida pela equipe de mecatrônica da EESC/USP; (5.4) análise de progressões harmônicas musicais, projeto e simulações realizados em colaboração com a equipe do NICS/UNICAMP. Para cada um dos domínios discutidos, a configuração do STRAGEN e os resultados obtidos são apresentados. Estes experimentos foram realizados para mostrar a diversidade de uso do STRAGEN e suas características, bem como a maneira pela qual o modelo cumpre os requisitos levantados nos capítulos anteriores.

No próximo capítulo estes requisitos serão considerados para as conclusões deste trabalho, e serão levantadas novas opções para trabalhos futuros.

Conclusões, Discussões e Trabalhos Futuros

*Concordamos que, com nossa fraca inteligência e compreensão, podemos apenas especular, até agora, sobre os grandes mistérios que nos confrontam por todos os lados tentando responder por cada coisa, em existência e experiência. A ciência tem muito sucesso em seu limitado campo de problemas, mas os grandes problemas, o **mysterium tremendus**, na existência de cada coisa que conhecemos, isto não é explicável por qualquer que seja o método científico.*

—POPPER (O cérebro e o pensamento)

Neste capítulo, a Seção 6.1 termina este documento com uma visão dos aspectos relevantes do modelo e suas contribuições.

Na Seção 6.2 discute-se algumas possibilidades levantadas no decorrer deste trabalho que poderiam levar a uma melhoria do modelo apresentado, e algumas questões que tratam de novas formas de abordar o problema são feitas. Também sugere-se caminhos que se abriram para o avanço deste modelo e da área de geração de trajetórias de estados.

6.1 Conclusões

Este trabalho apresentou um novo modelo de redes neurais artificiais com topologia dinâmica para a geração de trajetórias de estados e controle de sistemas não-lineares. Vários requisitos foram levantados para a concepção de um modelo geral para solucionar o problema independente de domínio. Os modelos STRAGEN-OFF e STRAGEN-ON tentam atender estes requisitos.

O primeiro importante requisito para a solução do problema proposto é realizar a identificação (Narendra & Parthasarathy, 1990) do sistema não-linear que se pretende controlar pela construção das transições e representação dos estados determinados no mapa. O modelo STRAGEN, através de um procedimento de aprendizado baseado em SOM-DT faz o mapeamento dos estados do sistema, armazenando-os em uma rede topológica que representa tais

estados.

Os nodos deste mapa topológico armazenam informações heterogêneas que contém as configurações de cada estado, compartimentalizadas em agrupamentos de informações homogêneas (Equação 4.2) e protegendo-as de distorções durante o processo de adaptação da rede.

A adaptação dos nodos durante o treinamento (Equação 4.12) só ocorre mediante a certeza de uma similaridade suficientemente grande, para que não dispare a inserção de um novo nodo. A similaridade é calculada individualmente pelos grupos e subgrupos que compõe o vetor peso que configura os estados (Tabela 4.1), de modo a evitar que uma proximidade pela distância Euclidiana do vetor como um todo faça a rede adaptar um nodo representativo de um padrão em direção a um estado distinto. Por esse motivo, a adição de nodos no modelo STRAGEN é agressiva, o que acarreta uma identificação rápida, e os nodos são inseridos no exato local (Passo 6b) em que ocorrem os estados de entrada, o que diminui o erro de representação.

Uma vez mapeado o sistema a ser tratado, é possível se utilizar da característica de completar padrões do STRAGEN para recuperar informações completas de um estado do sistema baseado em alguma informação parcial. Em especial, esta característica permite ao STRAGEN recuperar informações de controle armazenadas no nodo, a partir de informações do estado atual do sistema (Equação 2.3), o que em determinados sistemas (como o controle robótico) permite evitar o cálculo de cinemática e dinâmica inversa. A cinemática e dinâmica direta, por princípio análogo (e generalizado para outros sistemas), também pode ser recuperada a partir de informações de controle do sistema (Equação 2.2).

Para que o STRAGEN seja capaz de realizar a geração de trajetórias, não basta porém ter as informações de controle representadas e disponíveis. É preciso um processo de geração de trajetórias que atue no mapa topológico aprendido. O STRAGEN implementa este processo através da utilização de dois algoritmos de busca globais clássicos: a difusão de energia (Zeller et al., 1997) ou algoritmo de menor caminho, também referenciado na literatura como caminho de menor custo (Dijkstra, 1959). Uma vez criado um mapa topológico que representa o espaço de estados e contém informações para uma navegação segundo algum critério estabelecido, pode-se utilizar quaisquer algoritmos de busca globais ou técnicas de pesquisa locais para se gerar a trajetória. Ambos algoritmos utilizados têm ordem de complexidade $O(n^2)$, sendo n o número de nodos a visitar. O algoritmo de menor caminho, porém, foi otimizado com árvores binárias o que reduz seu custo para até $O(m + n \log(n))$ (Barbehenn, 1998), sendo m o número de vértices e n o de nodos. As Tabelas (5.9) e (5.10) mostram o ganho de performance do STRAGEN-ON (com poda *on-line* e algoritmo de menor caminho) quando comparado com o modelo STRAGEN-OFF (com fase de poda e algoritmo de difusão).

O STRAGEN implementa um sistema de controle em malha aberta (Figura 2.2). A ca-

racterística deste sistema é não receber correções de erro diretamente da planta. Este método simplifica sobremaneira a RNA (Werbos, 1991; Barreto et al., 2003b), pois não há necessidade de cálculos complexos de gradiente, nem de retropropagação de erros. A simplicidade do sistema é um ponto fundamental no desenvolvimento do STRAGEN. Seus parâmetros são pré-definidos, robustos, e não há necessidade de ajuste a não ser para efeitos comparativos, podendo-se utilizar os valores sugeridos deixando o usuário do sistema livre da tarefa de configuração. Os parâmetros internos são: estimativa do número máximo de disparos (para o decaimento da taxa de aprendizado) $\sigma_f = 2 \cdot t_{\max}/L$, taxa final de aprendizado $\alpha_f = 0,1$, número de candidatos no procedimento de Balbuciamiento Motor $Q = 10$, número máximo de iterações $t_{\max} = 3 \cdot L$ (i.e., três vezes é suficiente para apresentar os padrões da base de dados). O único parâmetro disponível para o usuário é a porcentagem P usada para o cálculo do limiar de atividade (Equação 4.3), que é sugerido como $P = 1\%$.

Não há necessidade de configurar uma topologia, pois o STRAGEN é um modelo de topologia dinâmica que adapta o número de nodos necessários à representação do espaço de estados. Este é um dos principais requisitos do modelo proposto, contribuindo com o avanço na área de planejamento e controle de sistemas não-lineares utilizando redes de topologia dinâmica, até o momento, na literatura consultada, uma abordagem inédita. A inserção de nodos não se faz a taxas fixas, como nos modelos GCS ou GNG, e diferente do modelo GWR, tem-se o cuidado de inserir nodos quando partes do vetor de pesos que agrupam informações distintas indicam um estado com configuração diferente de outros possíveis similares já representados. Em relação ao GWR, o STRAGEN leva em conta a precisão, mas não utiliza a frequência de disparos, pois esta não é importante quando se considera uma determinada precisão.

Tanto no STRAGEN-OFF quanto no STRAGEN-ON, evita-se o surgimento de nodos isolados. O STRAGEN-OFF, em sua fase de poda, elimina nodos isolados, e nunca desconecta o grafo quando tenta eliminar conexões. O STRAGEN-ON permite a representação de regiões desconexas, porém seu sistema de eliminação de conexões garante que nodos com menos que duas conexões não serão importunados, e apenas conexões cujo tamanho ultrapassa σ vezes o desvio padrão adicionado a media são excluídas. Esta proposta de utilizar o tamanho médio das conexões permite um crescimento bem distribuído do mapa topológico, mantendo-se as relações de similaridade entre estados mais próximos, e automaticamente adaptando-se ao crescimento da densidade do mapa conforme ocorram novas inserções de nodos e foi inspirada no conceito de falsos vizinhos (Kennel et al., 1992; Zhao & Small, 2006).

O STRAGEN-ON é capaz de alterar trajetórias de modo dinâmico, a qualquer tempo, bastando para isto que a rede esteja treinada, atuante, interligada ao sistema a ser controlado, e que lhe seja enviado o estado alvo e o estado atual do sistema. As trajetórias requisitadas podem

incluir pontos intermediários para serem alcançados. O modelo pode ser visualmente guiado, ou por qualquer outro sensor ou combinação de sensores que estejam mapeados em seus nodos. O STRAGEN-ON, pode, através de sensores que lhe indiquem a presença de obstáculos, desabilitar temporariamente conexões de nodos que estão obstruídos. Este procedimento é diferente do proposto por Zeller et al. (1997), que remove as conexões. O STRAGEN tenta sempre preservar as conexões que contém informações importantes sobre transições de estados (veja p. 112).

Com a utilização de critérios distintos para a composição das vizinhanças entre estados, é possível ao STRAGEN gerar trajetórias que otimizam diferentes características do sistema representado. As trajetórias podem ser criadas para otimizar critérios de proximidades entre os estados do sistema, ou entre sinais de controle. Esta característica torna o STRAGEN único em comparação aos modelos para planejamento e controle. Além de poder utilizar-se de diferentes critérios, o STRAGEN é um modelo geral, concebido para atuar em domínios diversos, diferente dos modelos apresentados (Subseção 2.1.3, p. 16) na literatura, específicos para cada domínio. A trajetória gerada está intrinsicamente ligada à formação do mapa auto-organizado segundo os critérios de otimização escolhidos, e portanto pode ser considerada como um fenômeno emergente não pré-definido.

A trajetória gerada pelo STRAGEN possui a propriedade de *equifinalidade temporal* (ET) (Ulloa & Bullock, 2003; Sanguineti & Morasso, 1992), que ocorre sem pré-planejamento, como propriedade emergente da otimização de um grupo de variáveis do sistema. No caso da mão Kanguera, por exemplo, pode-se ver os dedos fechando-se em sincronia na Figura (5.23) ao se otimizar a posição espacial da ponta dos dedos. Revendo a Figura (3.2), onde se discute a ET do modelo VITE, o STRAGEN partindo do ponto B1, tendo como alvo o ponto E, fará sempre o caminho mais curto que otimiza pelo critério pelo qual foi treinado. Supondo que o critério seja minimizar a componente horizontal, o STRAGEN fará a trajetória $B1 \rightarrow V \rightarrow E$ (análogo para a componente vertical). Caso o critério seja uma combinação de ambas componentes, a trajetória apresentará a ET para ambas. De modo geral, o STRAGEN sempre apresenta a propriedade de equifinalidade temporal para todas as variáveis envolvidas na composição do critério de vizinhança. Ainda assim, devido ao fato do STRAGEN implementar trajetórias *via-pontos*, é possível instruí-lo a fazer determinada trajetória passando também os pontos intermediários. Na Figura (3.2), pode-se requisitar ao STRAGEN que gere uma trajetória com ponto inicial B1, ponto final E e ponto intermediário V, e com isso forçar o STRAGEN a seguir pelo ponto intermediário mesmo que seu critério de otimização indique uma trajetória mais curta por outros pontos. A Figura (5.22) mostra uma outra trajetória não sincronizada gerada pela ausência de dados no espaço de estados. Neste caso não foi o critério ou uma requisição

explícita de trajetórias com pontos intermediários que criou o movimento não-sincronizado, mas a ausência de estados em que os dedos se fechassem ao mesmo tempo. O STRAGEN mostra-se robusto para gerar uma trajetória suave mesmo com uma base de dados incompleta.

A plausibilidade biológica do modelo está caracterizada pelas relações de proximidade entre nodos que se conectam em vizinhanças topológicas, no sentido explicitado por Kohonen (1987) (p. 52) e Hebb (p. 60). Não há módulos explicativos do cérebro ou do córtex, como nos modelos baseados no VITE (Gaudiano & Grossberg, 1992). O STRAGEN pode contribuir com explicações de caráter abstrato sobre a formação de mapas corticais e suas conexões, mas não se pode inferir que os nodos representam neurônios físicos devido ao fato do STRAGEN se tratar de uma rede dinâmica que cria e apaga nodos e conexões para melhor representar a topologia do espaço de entrada.

Aproveitando-se de um processo inspirado no aprendizado motor das crianças (Piaget, 1963), o STRAGEN utiliza o procedimento de Balbuciamiento Motor (Subseção 4.2.1) para aprender as conexões do espaço de entrada. Vale ressaltar que o BM foi concebido para o aprendizado intermodal entre sistemas motores e visuais (Gaudiano & Grossberg, 1991, 1992; Fritzke, 1997a; Marsland et al., 2002), e que o STRAGEN generaliza seu uso para o aprendizado do mapeamento de sistemas não-lineares diversos. A utilização do BM permite ao STRAGEN tirar proveito da característica Subida de Encosta Estrutural (Angeline et al., 1994) presente nos modelos incrementais, criando com isso conexões mais apropriadas para representar as trajetórias do sistema identificado. O BM ainda permite ao STRAGEN aprender estados válidos do sistema, evitando com isso o mapeamento de áreas inalcançáveis ou obstruídas no espaço de estados.

Ao contrário do GCS (Subseção 3.4.4), que tenta preservar uma estrutura *k-simplex* ao custo da criação de relações de vizinhanças “artificiais”, e do SOM (Subseção 3.2.3) cujas relações de vizinhança são pré-definidas, o STRAGEN cria estas conexões sob demanda e considera a densidade topológica do espaço de estados, se assemelhando neste caso ao GWR. Porém tendo o cuidado, ausente no GWR (Subseção 3.6.3) de manter as conexões que representam as transições de estado, não eliminando-as devido a um sistema de utilização ou *idade*. Não há porque eliminar conexões antigas ou pouco usadas ao representar um espaço de estados de domínio desconhecido, cuja referência para a transição entre os estados é a similaridade entre determinados grupos de características, e não o tempo de uso do sistema.

Com o interesse de evitar a subutilização das características dos SOM-DT, o papel das conexões é discutido neste trabalho. Nos testes feitos mostra-se a importância do critério de vizinhança e como ele influencia na criação das vizinhanças dos mapas topológicos e conseqüentes trajetórias formadas. Enquanto que os modelos discutidos relegam às conexões o papel de me-

ros indicadores de similaridade espacial e atividade dos nodos, o STRAGEN atribui a elas a função de representar transições de estados baseadas em critérios de vizinhança estabelecidos.

Também por entender o mapa topológico como a representação de um sistema não-linear que o processo de adaptação do STRAGEN envolve apenas o nodo vencedor, e não seus vizinhos. Ao receber em uma entrada de dados informações sobre um estado do sistema, não se recebe informações sobre como os outros estados, vizinhos ou não, devam ser ajustados.

Adicionalmente, o STRAGEN insere novos nodos na exata posição do primeiro representante de um estado desconhecido, e não como os outros modelos, em posições médias entre nodos vencedores e vice. Inserir nodos que representam médias de protótipos e não estados que efetivamente ocorrem levam tais modelos a inserí-los onde não há padrões no espaço de estados original. Para evitar a distorção feita por um processo de adaptação que cria protótipos médios inexistentes entre estados do sistema, [Walter & Schulten \(1993\)](#) realizam o cálculo de gradientes $\Delta\theta$, $\Delta\tau$ e da matriz Jacobiana em seu modelo baseado no TRN, o que é custoso além de eliminar a qualidade de recuperação de informação característica dos SOM. É a interpretação do mapa topológico como um sistema dinâmico descrito num espaço de estados que inspira todas estas características do STRAGEN, para que este mantenha as propriedades aprendidas intactas, úteis e recuperáveis.

6.2 Discussões e Trabalhos Futuros

Até que ponto se pode alterar estes mapas topológicos e obter um melhor resultado, baseando-se nos critérios de vizinhança? Percebe-se que nas redes SOM-DT o papel desempenhado pelas conexões está relegado à indicadores de similaridades, quando muitos marcadores de atividade neural para o controle de remoção de nodos inativos. Os testes com o STRAGEN mostraram que o papel das conexões, quando a elas se atribui uma *interpretação*, é essencial para a criação de mapas topológicos, e vai em última instância guiar o algoritmo de geração de trajetórias através de critérios de otimização.

Uma proposta interessante é criar uma rede que inclua simultaneamente e em paralelo diferentes critérios de vizinhança. Essa proposta original pode trazer uma nova forma de lidar com mapas auto-organizáveis. Tais conexões seriam classificadas, por exemplo, por arestas de cores distintas, cada qual ligando seus próprios vizinhos a partir de cada nodo, lembrando o uso de Redes de Petri para a modelagem de sistemas concorrentes ([Jensen, 1994](#)). Com este mecanismo, o algoritmo se torna capaz de alterar o critério de geração de trajetórias sem a necessidade de um novo treinamento da rede, e mais que isso, se torna capaz de achar cami-

nhos alternativos para suprir trechos desconexos em algum critério, alterando-se o critério de vizinhança.

Caracterizar a semelhança em um mapa que representa transições em um espaço de estados de um sistema dinâmico não-linear leva a interpretações muito mais complexas que a simples distância Euclidiana geralmente utilizada em mapas auto-organizáveis. Experimentos psicofísicos indicam que humanos preferem movimentar-se minimizando a energia (torque) usada (Bullock et al., 1999; Bullock & Grossberg, 1988a), o que é coerente com o que se observa na natureza em geral. Seria possível adaptar este princípio para outros sistemas dinâmicos, mesmo que não relacionados a robótica? Quais os limites de um modelo como o STRAGEN, que mapeia espaços de estados de sistemas dinâmicos? Que tipos de critérios podem ser representados para que a geração de trajetórias nestes sistemas possa ser realizada otimizando-o?

Neste ponto se percebe a questão da escolha da métrica, além da escolha do que medir. Nas simulações realizadas até o momento, utilizou-se como métrica a distância Euclidiana. Qual seria a influência de utilizar outras métricas, como a *distância de Minkowski*, que é uma generalização da distância Euclidiana, ou a métrica de *Tanimoto*, que mede a distância entre conjuntos, para um determinado domínio de problemas? Uma das maiores dificuldades na escolha de métricas tem a ver com a invariância a transformações, tais como translações horizontais, verticais, rotações, escalas e normalizações, e assim por diante. Isso se torna ainda mais complexo se for desejado que a medida de distância seja simultaneamente invariante a diversas transformações (Duda et al., 2000).

Neste momento, utilizou-se como algoritmos geradores de trajetória a função de difusão de energia e o algoritmo de menor caminho. Estes algoritmos têm a importante propriedade de sempre achar um trajeto entre dois pontos dados, se por eles existir conexão, e esta trajetória terá o menor número de nodos possível.

Um dos problemas de buscas globais como a difusão de energia e o menor caminho é o custo computacional. Uma possível solução a ser analisada é a utilização de difusão local, que ao invés de gerar a trajetória até o nodo alvo, difundiria a energia por um número limitado de nodos (o limite pode se dar por número de nodos, número de camadas de vizinhos, ou por tempo disponível para a iteração). Uma vez difundida a energia, os nodos da borda desse campo formado são todos candidatos a fazerem parte da trajetória final resultante, e são chamados de *janela de candidatos*. Então é escolhido o nodo vencedor dentre os nodos da janela, aquele que faz com que a posição final da trajetória fique mais próxima. Este modelo alternativo tem a desvantagem de poder cair em mínimos locais, além de que não se garante que a trajetória formada será aquela dada pela otimização do critério de vizinhança de modo global. A trajetória gerada seria, dentro do campo de difusão, seguida pela otimização do critério de vizinhança;

porém nas transições entre cada campo, de cada iteração do processo, o ponto de escolha seria o critério de menor distância entre a posição atual e o nodo alvo, o que compromete a trajetória como um todo e não garante que uma trajetória possa ser encontrada (na presença de obstáculos, por exemplo).

Um outro algoritmo que chama a atenção por sua eficiência é o *Rapidly Exploring Random Trees* (RTT) (Jarvis, 2006), que faz a busca ao acaso em um grafo. Seu funcionamento básico consiste em escolher estados aleatoriamente e gerar trechos de trajetória desde o ponto inicial até este estado, aumentando gradativamente a área de busca. Este algoritmo pode trabalhar simultaneamente, partindo do ponto inicial e final e gerando trechos de trajetórias até que estas se conectem. Entretanto, como não há garantia de que os estados estejam interligados e que uma trajetória possa ser encontrada, é mais interessante utilizar o tempo de processamento apenas a partir do nodo inicial, para que os trechos de trajetórias gerados possam ser utilizados imediatamente, de modo *on-line*. Comparado com a difusão local, o RTT local não necessita expandir todos os nodos da rede a partir do nodo inicial, apenas os trechos escolhidos ao acaso, o que diminui a *janela de candidados* aumentando a velocidade da exploração da árvore.

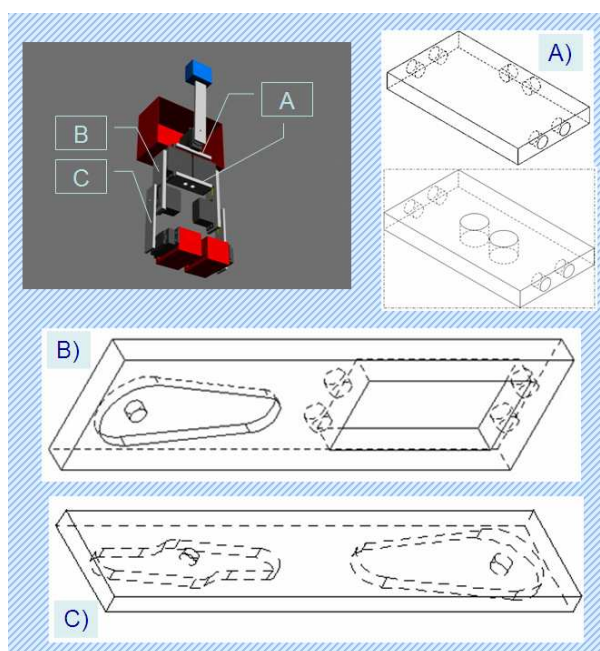


Figura 6.1 Robô bípede MNRIM: (a) Barras de alumínio de 55mm de largura ligando o contra-peso. (b) Barra de alumínio de 90mm de comprimento da perna, ligando a junta do corpo à junta do joelho. (c) Barra de 90mm de comprimento ligando o joelho ao tornozelo. (Foresti, 2006)

O processo de adaptação do mapa topológico do STRAGEN depende da constante iteração com o sistema identificado. A remoção de conexões é um ponto crucial que precisa de

aprimoramentos. O STRAGEN, tendo em vista a representação do sistema, não elimina conexões inativas. As remoções são feitas apenas para melhor ajustar as vizinhanças topológicas em relação ao critério utilizado e à densidade da rede, que indiretamente mede o tamanho das conexões. Este processo revelou-se adequado, de custo baixo, e capaz de preservar a topologia mesmo de áreas desconexas. Entretanto, pela própria atenção dispendida ao papel das conexões no STRAGEN, seria interessante investigar ainda outros métodos para a manutenção das transições de estados. Num olhar superficial, pode parecer não haver outro critério para remoções de conexões que não a avaliação de sua atividade, como fazem os modelos GWR e GCS pelo uso da idade, ou remover nodos em áreas de baixa probabilidade como faz o modelo GCS. Uma nova possibilidade seria aproveitar a característica de representação de informações heterogêneas dos nodos no STRAGEN para coletar do próprio sistema informações ou critérios que possam ser utilizados para determinar a remoção de nodos e conexões. Esta propriedade é dependente do sistema a se representar, porém parece mais plausível do que determinar de antemão um critério que afetará o modo como o sistema é mapeado.

Há que se analisar também a proposta de [De Angulo & Torras \(2005\)](#) para a redução do número necessário de pontos para o treinamento da cinemática inversa de um robô. Ao invés de aprender uma função inversa da cinemática, o método proposto consiste em expressar tal função em uma composição de funções de aprendizado, cada qual com metade da dimensão do mapa original. Seria interessante avaliar como unir este método com o método de Balbuciamiento Motor do STRAGEN, para que se consiga representar com maior precisão uma base de dados com informações limitadas. No entanto, a proposta de [De Angulo & Torras \(2005\)](#) é específica para cinemática inversa, e sua utilização para sistemas dinâmicos não-lineares em geral teria que ser re-avaliada.

A implementação de um sistema de memória que guarda trajetórias realizadas é um ponto interessante para se colocar. Tal sistema poderia aprender as trajetórias durante os primeiros usos, e refazê-las imediatamente quando solicitadas pela segunda vez, ou ainda podia memorizar não só transições e estados durante o balbuciamiento motor, mas trechos de trajetórias que poderiam ser utilizados já na primeira vez que o modelo for requisitado a atuar. Um sistema de memória permitiria também realizar o rastreamento de trajetórias e previsões.

O STRAGEN foi aplicado a diversos domínios neste trabalho: robô bidimensional e tridimensional PUMA-560, mão robótica Kanguera e estudo de progressões harmônicas. Outros domínios interessantes que podem ser desenvolvidos são: trajetórias de robôs móveis ([Glasius et al., 1995](#); [Nehmzow, 2001](#)) em especial robôs da classe PIONEER, estudo de logística e transporte ([Chauhan et al., 2005](#); [Barr et al., 1981](#)), e o interessante problema de equilíbrio em robôs bípedes ([Miller, 1994](#)). [Foresti \(2006\)](#) desenvolveu um robô bípede (Figura 6.1) e

gerou uma base de dados de posições em que o robô encontra-se em equilíbrio estático para ser treinada pelo STRAGEN. Um dos maiores problemas na movimentação de robôs bípedes é a realização de passos em equilíbrio dinâmico (Miller, 1994). Os primeiros robôs a caminhar com sucesso o faziam transitando entre estados de equilíbrio estático. O movimento em equilíbrio dinâmico exige que o robô literalmente comece a “cair” e em tempo movimento seu pé para que sua base se desloque para a região que equilibra a queda em um novo passo. O STRAGEN abre caminhos em distintas áreas de pesquisa e utilizá-lo para contribuir com tais áreas é além de um desafio aprazível um trabalho de pesquisa encantador pela sua diversidade.

Referências Bibliográficas

- Agre, P. E. & Chapman, D. (1990), What are plans for?, in P. Maes, ed., 'Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back', The MIT Press, Cambridge, MA, pp. 17–34.
URL: citeseer.ist.psu.edu/agre89what.html
- Angeline, Saunders & Pollack (1994), 'An evolutionary algorithm that constructs recurrent neural networks', *IEEE Transactions on Neural Networks* **5**(1), 54–65.
- Araújo, A. F. R. & Barreto, G. A. (2002), 'Context in temporal sequence processing: A self-organizing approach and its application to robotics', *IEEE Transactions on Neural Networks* **13**(1), 45–57.
- Ashby, W. R. (1962), Principles of the self-organizing system, in H. Von Foerster & G. W. Zopf Jr., eds, 'Principles of self-organization', Pergamon, Oxford, pp. 255–278.
- Atukorale, A. S. & Suganthan, P. N. (2000), 'Hierarchical overlapped neural-gas network with application to pattern classification', *Neurocomputing* **35**(1-4), 165–176.
- Aupetit, M., Couturier, P. & Massotte, P. (2002), 'Gamma-observable neighbours for vector quantization', *Neural Networks* **15**(8–9), 1017–1027.
- Baraldi, A. & Alpaydin, E. (2002), 'Constructive feedforward ART clustering networks - part I,II', *IEEE Transactions on Neural Networks* **13**(3), 645–677.
- Barbehenn, M. (1998), 'A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices', *IEEE Trans. Comput.* **47**(2), 263.
- Barr, R. S., Glover, R. S. & Klingman, D. (1981), 'A new optimization method for large scale fixed charge transportation problems', *Oper. Res.* **29**(3), 448–463.
- Barreto, G. A. & Araújo, A. F. R. (1998), Competitive and temporal hebbian learning for production of robot trajectories, in '2nd Brazilian Symposium on Neural Networks (SBRN'98)',

pp. 96–101.

URL: <http://csdl.computer.org/comp/proceedings/sbrn/1998/8629/00/86290096abs.htm>

Barreto, G. A. & Araújo, A. F. R. (2001), ‘Time in self-organizing maps: An overview of models’, *International Journal of Computer Research* **10**(2), 139–179.

Barreto, G. A. & Araújo, A. F. R. (2004), ‘Identification and control of dynamical systems using the self-organizing map’, *IEEE Transactions on Neural Networks* **15**(5), 1244–1259.

Barreto, G. A., Araújo, A. F. R., Dücker, C. & Ritter, H. (2002a), ‘Implementation of a distributed control system for puma 560 robot based on a temporal self-organizing network’, *Revista Controle & Automação* **13**(2), 141–155.

Barreto, G. A., Araújo, A. F. R., Dücker, C. & Ritter, H. (2002b), ‘A distributed robotic control system based on a temporal self-organizing neural network’, *IEEE Transactions on Systems Man and Cybernetics, Part C - Applications and Reviews* **32**(4), 347–357.

Barreto, G. A., Araújo, A. F. R. & Kremer, S. C. (2003a), ‘A taxonomy for spatiotemporal connectionist networks revisited: The unsupervised case’, *Neural Computation* **15**(6), 1255–1320.

URL: <http://neco.mitpress.org/cgi/content/abstract/15/6/1255>

Barreto, G. A., Araújo, A. F. R. & Ritter, H. (2003b), ‘Self-organizing features maps for modelling and control of robotic manipulators’, *Journal of Intelligent and Robotic Systems* **36**(4), 407–450.

Barto, A. G., Sutton, R. S. & Anderson, C. W. (1983), ‘Neuronlike adaptive elements that can solve difficult learning control problems’, *IEEE Transactions on Systems, Man and Cybernetics* **5**, 834–846.

Benante, R. C. (2001a), Alcance e limites das redes neurais artificiais na ciência cognitiva, Mestrado em ciência cognitiva e filosofia da mente, Universidade Estadual Paulista - UNESP, Marília, SP, Brazil.

Benante, R. C. (2001b), Auto-organização nas máquinas de Turing., Seminários de auto-organização, Centro de Lógica, Epistemologia e História da Ciência (CLE), UNICAMP, Campinas.

URL: http://www.cle.unicamp.br/arquivoshistoricos/av_seminariosao_1.html

- Benante, R. C. & Araújo, A. F. R. (2007a), 'Self-organizing maps to generate state trajectories of manipulators', *IEEE International Conference on Systems, Man and Cybernetics (ISIC'07)* pp. 1590–1595.
- Benante, R. C., Araújo, A. F. R. & Ludermir, T. B. (2004), Automatização na escolha de parâmetros para o modelo incremental *GNG* usando algoritmos genéticos, in A. K. Barros, A. F. R. Araújo, H. C. Yehia & R. Teixeira, eds, '8th Brazilian Symposium on Neural Networks (SBRN'04)', Federal University of Maranhão, UFMA, IEEE Computer Society, São Luis, MA, Brazil. Digital Media, ISBN: 85-89029-04-2.
- Benante, R. C., Pedro, L. M., Massaro, L. C., Belini, V. L., Araújo, A. F. R. & Caurin, G. A. P. (2007b), 'A self-organizing state trajectory planner applied to an anthropomorphic robot hand', *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 3082–3087.
- Bertsekas, D. P. (2000), *Dynamic Programming and Optimal Control*, Athena Scientific.
- Bharucha, J. J. (1991), 'Pitch, harmony, and neural nets: A psychological perspective', *Music and Connectionism*.
- Bishop, C. M. (1996a), *Neural networks for pattern recognition*, Oxford University Press, Oxford, UK.
- Bishop, C. M. (1996b), Neural networks: A pattern recognition perspective, Technical report, NCRG.
- Bobrow, J. E. (1988), 'Optimal robot path planning using the minimum-time criterion', *IEEE Journal of Robotics and Automation* **4**(4), 443–450.
- Bonet, B. & Geffner, H. (2000), Planning with incomplete information as heuristic search in belief space, in 'Artificial Intelligence Planning Systems', pp. 52–61.
URL: citeseer.ist.psu.edu/bonet00planning.html
- Bonet, B. & Geffner, H. (2001), 'Planning and control in artificial intelligence: A unifying perspective', *Applied Intelligence* **14**(3), 237–252.
- Borst, C., Fischer, M., Haidacher, S., Liu, H. & Hirzinger, G. (2003b), 'DLR hand II: experiments and experience with an anthropomorphic hand', *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'03)* **1**, 702–707.

- Borst, C., Fischer, M. & Hirzinger, G. (2003a), 'Grasping the dice by dicing the grasp', *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'03)* **4**, 3692–3697.
- Bresciani Filho, E. & D'Ottaviano, I. M. L. (2000), Conceitos básicos de sistema dinâmico caótico, Seminários de auto-organização, Centro de Lógica, Epistemologia e História da Ciência (CLE), UNICAMP, Campinas.
URL: http://www.cle.unicamp.br/arquivoshistoricos/av_seminariosao_1.html
- Brooks, R. (1987), 'A robust layered control system for a mobile robot', *IEEE Journal of Robotics and Automation* **2**(1), 14–23.
- Broomhead, D. & Lowe, D. (1988), 'Multivariable function interpolation and adaptive networks', *Complex Systems* **2**, 321–355.
- Bullock, D., Bongers, R., Lankhorst, M. & Beek, P. J. (1999), 'A vector-integration-to-endpoint model for performance of viapoint movements', *Neural Networks* **12**, 1–29.
- Bullock, D., Cisek, P. E. & Grossberg, S. (1998), 'Cortical networks for control of voluntary arm movements under variable force conditions', *Cerebral Cortex* **8**, 48–62.
- Bullock, D. & Grossberg, S. (1988a), 'Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation', *Psychological Review* **95**, 49–90.
- Bullock, D. & Grossberg, S. (1988b), The VITE model: A neural command circuit for generating arm and articulator trajectories, in J. A. S. Kelso, A. J. Mandell & M. F. Shlesinger, eds, 'Dynamic patterns in complex systems', World Scientific Publishers, Singapore.
- Bullock, D. & Grossberg, S. (1989), VITE and FLETE: Neural modules for trajectory formation and postural control, in W. Hershberger, ed., 'Volition Action', North-Holland, Amsterdam, pp. 253–297.
- Bullock, D. & Grossberg, S. (1991), 'Adaptive neural networks for control of movement trajectories invariant under speed and force rescaling.', *Human Movement Science* **10**(1), 3–53.
- Campos, P. G. & Ludermir, T. B. (2005), 'LITERAL and PRORULEXT: Algorithms for rule extraction of ANNs', *5th Int. Conf. on Hybrid Intelligent Systems (HIS'05)* pp. 143–148.
- Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H. & Rosen, D. B. (1992), 'Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analogue multidimensional maps', *IEEE Transactions on Neural Networks* **3**, 698–713.

- Caurin, G. A. P., Aguiar, P. M. & Valente, C. M. O. (2002), 'Planning and control system for an anthropomorphic grippers', *IEEE/RSJ Int. Conf. on Intel. Robots and Systems (IROS'02)* **1**, 340–349.
- Caurin, G. A. P., Albuquerque, A. L., Prado, M. & Carvalho, J. (2005), The implementation of the brazilian anthropomorphic gripper: Design, construction and 3d object manipulation, Technical report, University of São Paulo, Mechatronics Laboratory (EESC), São Carlos - SP, Brazil.
- Caurin, G. A. P., Albuquerque, A. R. L. & Mirandola, A. L. A. (2004), 'Manipulation strategy for an anthropomorphic robotic hand', *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)* .
- Cavanagh, P. & Davidson, M. L. (1977), 'Secondary reaction and response elicitation in operant learning of 6-month-old infants', *Developmental Psychology* **4**, 371–376.
- Chauhan, S. S., Ereemeev, A. V., Kolokolov, A. A. & Servakh, V. V. (2005), Concave cost supply management problem for single manufacturing unit, in A. Dolgui, J. Soldek & O. Zaikin, eds, 'Supply Chain Optimisation, Product/Process Design, Facility Location and Flow Control', Springer-Verlag, pp. 167–174.
- Chauvin, J., Sinagre, L. & Murray, R. M. (2003), 'Nonlinear trajectory generation for the caltech multi-vehicle wireless testbed', *European Control Conference* .
URL: citeseer.ist.psu.edu/chauvin03nonlinear.html
- Chemillier, M. (2004), 'Toward a formal study of jazz chord sequences generated by steedman's grammar', *Soft Computing* **8**(9), 617–622.
- Chen, C. T. (1998), *Linear System Theory and Design*, Oxford University Press, London.
- Chen, L. & Narendra, K. S. (2002), 'Intelligent control using neural networks and multiple models', *Proceedings of the 41st IEEE Conference on Decision and Control* pp. 1357–1362.
URL: citeseer.ist.psu.edu/542344.html
- Clarke, D. W., Mohtadi, C. & Tuffs, P. S. (1987), 'Generalized predictive control. 1. the basic algorithm 2. extensions and interpretations', *Automatica* **23**(2), 137–160.
- Coiton, Y., Gilhodes, J. C., Velay, J. L. & Roll, J. P. (1991), 'A neural network model for the intersensory coordination involved in goal-directed movements', *Biological Cybernetics* **66**(2), 167–176.

- Cooper, P. (1975), *Perspectives in Music Theory: an historical-analytical approach*, Dood, Mead and Company, New York.
- Corke, P. I. (1996), 'A robotics toolbox for MATLAB', *IEEE Robotics and Automation Magazine* **3**(1), 24–32.
- Craig, J. J. (1986), *Introduction to Robotics*, Addison Wesley.
- De Angulo, V. R. & Torras, C. (2005), 'Speeding up the learning of robot kinematics through function decomposition', *Neural Networks, IEEE Transactions on* **16**(6), 1504–1512.
- Debrun, M. (1996), A idéia de auto-organização, in 'Auto-organização: estudos interdisciplinares', Vol. 18 of *Coleção CLE*, CLE/UNICAMP, Campinas, pp. 3–23.
- Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs', *Numerische Mathematik* **1**, 269–271.
- Dittenbach, M., Merkl, D. & Rauber, A. (2000), The growing hierarchical self-organizing map, in S. Amari, C. L. Giles, M. Gori & V. Puri, eds, 'Proc of the International Joint Conference on Neural Networks (IJCNN'00)', Vol. VI, IEEE Computer Society, Como, Italy, pp. 15–19.
URL: citeseer.ist.psu.edu/dittenbach00growing.html
- Duda, R. O., Hart, P. E. & Stork, D. G. (2000), *Pattern classification*, 2nd edn, Wiley-Interscience, New York.
- Elman, J. L. (1990), 'Finding structure in time', *Cognitive Science* **14**(2), 179–211.
URL: <http://www.isrl.uiuc.edu/~amag/langev/paper/elman90findingStructure.html>
- Exner, C., Koschack, J. & Irle, E. (2002), 'The differential role of premotor frontal cortex and basal ganglia in motor sequence learning: Evidence from focal basal ganglia lesions', *Learning & Memory* **9**(6), 376–386.
- Fahlman, S. E. (1991), The recurrent cascade-correlation architecture, in R. P. Lippmann, J. E. Moody & D. S. Touretzky, eds, 'Advances in Neural Information Processing Systems', Vol. 3, Morgan Kaufmann Publishers, Inc., pp. 190–196.
URL: citeseer.ist.psu.edu/fahlman91recurrent.html
- Fahlman, S. E. & Lebiere, C. (1990), The cascade-correlation learning architecture, in D. S. Touretzky, ed., 'Advances in Neural Information Processing Systems', Vol. 2, Morgan Kaufmann, San Mateo, Denver, pp. 524–532.

- Fiala, J. C. (1994), 'A network for learning kinematics with application to human reaching models', *IEEE WCCI, IEEE International Conference on Neural Networks* **5**, 2759–2764.
- Fikes, R. & Nilsson, N. J. (1971), 'Strips: A new approach to the application of theorem proving to problem solving', *Artificial Intelligence* **2**(3–4), 189–208.
- Flentge, F. (2006), 'Locally weighted interpolating growing neural gas', *IEEE Transactions on Neural Networks* **17**(6), 1382–1393.
- Foresti, H. B. (2006), Desenvolvimento de um robô bípede autônomo, Mestrado em engenharia mecânica, Universidade Federal de Pernambuco - UFPE, Recife, PE, Brazil.
URL: http://roboivre.com/download/dissertacao_banca2B.pdf
- Freeman, R. & Yin, H. (2005), 'Web content management by self-organization', *IEEE Transactions on Neural Networks* **16**(5), 1256–1268.
- Fritzke, B. (1991), 'Unsupervised clustering with growing cell structures', *Proc. Int. Joint Conf. on Neural Networks* **II**, 531–536.
- Fritzke, B. (1994a), 'Growing cell structures - a self-organizing network for unsupervised and supervised networks', *Neural Networks* **7**(9), 1441–1460.
- Fritzke, B. (1994b), Supervised learning with growing cell structures, in J. Cowan, G. Tesauro & J. Alspector, eds, 'Advances in Neural Information Processing Systems', Morgan Kaufmann Publishers Inc., San Mateo, CA, pp. 255–262.
- Fritzke, B. (1995), 'Growing grid - a self-organizing network with constant neighborhood range and adaptation strength', *Neural Processing Letters* **2**(5), 9–13.
- Fritzke, B. (1996), Growing self-organizing networks - why?, in 'European Symposium on Artificial Neural Networks (ESANN'96)', D-Facto Publishers, Brussels, pp. 61–72.
- Fritzke, B. (1997a), 'Unsupervised ontogenic networks', *Handbook of neural computation* **1**, 1–16.
- Fritzke, B. (1997b), A self-organizing network that can follow non-stationary distributions, in 'Proceedings of the International Conference on Artificial Neural Networks (ICANN'97)', Springer, Berlin, pp. 613–618.
- Fu, K. S., Gonzalez, R. C. & Lee, C. S. G. (1987), *Robotics: control, sensing, vision, and intelligence*, CAD/CAM robotics and computer vision, McGraw-Hill, USA.

- Fuji, T. & Ura, T. (1991), 'Neural-network-based adaptive control systems for AUVs', *Engineering Application of Artificial Intelligence* **4**(4), 309–318.
- Gaudio, P. & Grossberg, S. (1991), 'Vector associative maps: Unsupervised real-time error-based learning and control of movement trajectories', *Neural Networks* **4**(2), 147–183.
- Gaudio, P. & Grossberg, S. (1992), 'Adaptive vector integration to end-point - self-organizing neural circuits for control of planned movement trajectories', *Human Movement Science* **11**(1–2), 141–155.
- Glasius, R., Komoda, A. & Gielen, S. C. A. M. (1995), 'Neural network dynamics for path planning and obstacle avoidance', *Neural Networks* **8**(1), 125–133.
URL: citeseer.ist.psu.edu/glasius95neural.html
- Gomi, H. & Kawato, M. (1990), 'Learning control for a closed loop system using feedback-error-learning', *Proceedings of the 29th IEEE Conference on Decision and Control* **6**, 3289–3294.
- Gonzalez, M. E. Q., Broens, M. C., Haselager, W. F. G. & Bresciani Filho, E. (2005), 'Self-organization and life: A systemic approach', *Manuscrito* **28**(2), 375–390.
- Griffith, N. (1999), 'Development of tonal centres and abstract pitch as categorizations of pitch use', *Musical Networks: Parallel Distributed Perception and Performance* .
- Griffiths, P. (1978), *Modern Music: A Concise History from Debussy to Boulez*, Thames and Hudson, London.
- Grossberg, S. (1976), 'Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors', *Biological Cybernetics* **23**, 121–134.
- Grossberg, S. & Paine, R. W. (2000), 'A neural model of cortico-cerebellar interactions during attentive imitation and predictive learning of sequential handwriting movements', *Neural Networks* **13**(8–9), 999–1046.
- Guo, J., Chiu, F. C. & Wang, C.-C. (1995), 'Adaptive control of an autonomous underwater vehicle testbed using neural networks', (*OCEANS '95*) *MTS/IEEE. Challenges of Our Changing Global Environment. Conference Proceedings*. **2**, 1033–1039.
- Haaland, K. Y., Harrington, D. L. & Knight, R. T. (2000), 'Neural representations of skilled movement', *Brain* **123**, 2306–2313. Part 11.

- Harrington, D. L., Rao, S. M., Haaland, K. Y., Bobholz, J. A., Mayer, A. R., Binder, J. R. & Cox, R. W. (2000), 'Specialized neural systems underlying representations of sequential movements', *Journal of Cognitive Neurosciences* **12**, 55–7.
- Haslinger, B., Erhard, P., Weilke, F., Ceballos-Baumann, A. O., Bartenstein, P., von Einsiedel, H. G., Schwaiger, M., Conrad, B. & Boecker, H. (2002), 'The role of lateral premotor-cerebellar-parietal circuits in motor sequence control: a parametric fMRI study', *Cognitive Brain Research* **13**(2), 159–168.
- Haykin, S. (1994), *Neural networks: a comprehensive foundation*, Prentice Hall.
- Heinke, D. & Hamker, F. H. (1998), 'Comparing neural networks: A benchmark on growing neural gas, growing cell structures, and fuzzy ARTMAP', *IEEE Transactions on Neural Networks* **9**(6), 1279–1291.
- Hoffmann, H., Petkos, G., Bitzer, S. & Vijayakumar, S. (2007), Sensor-assisted adaptive motor control under continuously varying context, in 'Proc. International Conference on Informatics in Control, Automation, and Robotics'.
- Hoffmann, H., Schenck, W. & M ller, R. (2005), 'Learning visuomotor transformations for gaze-control and grasping', *Biological Cybernetics* **93**, 119–130.
- Hopfield, J. J. (1982), 'Neural networks and physical systems with emergent collective computational abilities', *Proceedings of the National Academy of Sciences* **79**, 2554–2558.
- Ijspeert, A. J., Nakanishi, J. & Schaal, S. (2001), Trajectory formation for imitation with non-linear dynamical systems, in 'Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)', Maui, Hawaii, USA.
- Illina, I. & Gong, Y. (1996), Stochastic trajectory model with state-mixture for continuous speech recognition, in 'Proceedings of the ICSLP'96', Vol. 1, Philadelphia, PA, pp. 342–345.
URL: citeseer.ist.psu.edu/illina96stochastic.html
- Izquierdo, L. F., Bullock, D. & Coronado, J. L. (1996), Alternative spatial coordinate frames and DIRECT control of redundant manipulators, in 'Proceedings of the IARP International Workshop on Medical Robots', Vienna, Austria, pp. 55–62.
- Jarvis, R. (2006), Robot path planning: complexity, flexibility and application scope, in 'Proceedings of the 2006 international symposium on Practical cognitive agents and robots (PCAR '06)', ACM, New York, NY, USA, pp. 3–14.

- Jensen, K. (1994), An introduction to the theoretical aspects of coloured petri nets, in J. W. de Bakker, W. de Roever & G. Rozenberg, eds, 'A Decade of Concurrency', Vol. 803 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 230–272.
- Jeong, W. K., Ivriissimtzis, I. P. & Seidel, H. P. (2003), Neural meshes: Statistical learning-based on normals, in 'Proceedings of the 11th Pacific Conference on Computer Graphics and Applications', pp. 404–408.
- Jockusch, J. & Ritter, H. (1999), 'An instantaneous topological mapping model for correlated stimuli', *Proceedings of the IJCNN'99* p. 445.
- Kaelbling, L. P., Littman, M. L. & Cassandra, A. R. (1998), Planning and acting in partially observable stochastic domains, Technical Report CS-96-08, Microelectronics and Computer Technology Corporation (MCC), Austin, USA.
- Kalish, M. L. (1994), 'Idiosyncratic errors in visually directed reaching', *Journal of Motor Behavior* **26**(3), 296–300.
- Kandel, E. R., Schwartz, J. H. & Jessell, T. M. (1995), *Essentials of Neural Science and Behavior*, Appleton & Lange.
- Kawato, M. (1990), Computational schemes and neural network models for formation and control of multijoint arm trajectory, in W. T. Miller, R. Sutton & P. Werbos, eds, 'Neural Networks for Robotics and Control', The MIT Press, Cambridge, MA, pp. 197–228.
- Kelso, J. A. S. (1999), *Dynamic Patterns: the self-organization of brain and behavior*, 1st edition edn, The MIT Press, London.
- Kennel, M. B., Brown, R. & Abarbanel, H. D. I. (1992), 'Determining embedding dimension for phase-space reconstruction using a geometrical construction', *Phys. Rev. A* **45**, 3403–3411.
- Keymeulen, D. & Decuyper, J. (1996), 'Self-organizing system for the motion planning of mobile robots', *Proceedings of the IEEE International Conference on Robotics and Automation* **4**, 3369–3374.
- Khalil, H. K. (2001), *Nonlinear Systems*, 3rd edition edn, Prentice Hall, London.
- Kohonen, T. (1987), *Self-Organization and Associative Memory*, 6, 2nd edn, Springer-Verlag, Berlin.

- Kohonen, T. (1997), *Self-Organizing Maps*, Springer Series in Information Sciences, 2nd edn, Springer-Verlag, Berlin.
- Kubota, N., Nojima, Y., Kojima, F. & Fukuda, T. (2006), 'Multiple fuzzy state-value functions for human evaluation through interactive trajectory planning of a partner robot', *Soft Computing* **10**(10), 891–901.
- Kuperstein, M. (1991), 'INFANT neural controller for adaptive sensory-motor coordination', *Neural Networks* **4**, 131–145.
- Leman, M. (1990), 'Emergent properties of tonality functions by self-organization', *Interface* **19**(2–3), 85–106.
- Leman, M. (1991), The ontogenesis of tonal semantics: Results of a computer study, in P. Todd & G. Loy, eds, 'Music and Connectionism', The MIT Press, Cambridge, MA, pp. 100–127.
- Lerdahl, F. & Jackendoff, R. (1996), *A Generative Theory of Tonal Music*, The MIT Press, Cambridge, MA.
- Lewis, H. R. & Papadimitriou, C. H. (2000), *Elementos de Teoria da Computação*, 2nd edn, Bookman, Porto Alegre.
- Lotti, F., Tiezzi, P., Vassura, G., Biagiotti, L., Palli, G. & Melchiorri, C. (2005), 'Development of UB hand 3: Early results', *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on* pp. 4488–4493.
- Lovelock, W. (1987), *História concisa da música*, Martins Fontes, São Paulo.
- Mandelbrot, B. B. (1975), *Fractals: Form, Chance and Dimension*, Flammarion, Paris, France.
- Marion, J. B. (1971), *Classical Dynamics of particles and systems*, 2nd edn, Academic Press, Inc. (London) Ltd., London.
- Marsland, S., Shapiro, J. & Nehmzow, U. (2002), 'A self-organizing network that grows when required', *Neural Networks* **15**(8–9), 1041–1058.
- Martens, G., Meyer, H. D., Baets, B. D., Leman, M., Lesaffre, M. & Martens, J. P. (2005), 'Tree-based versus distance-based key recognition in musical audio', *Soft Computing* **9**(8), 565–574.

- Martinetz, T. M., Berkovich, S. G. & Schulten, K. J. (1991), A 'neural-gas' network learns topologies, in T. Kohonen, K. Mäkisara, O. Simula & J. Kangas, eds, 'Artificial Neural Networks', North-Holland, Amsterdam, pp. 397–402.
- Martinetz, T. M., Berkovich, S. G. & Schulten, K. J. (1993), 'Neural-gas network for vector quantization and its application to time-series prediction', *IEEE Transactions on Neural Networks* **4**(4), 558–569.
- Martinetz, T. M., Ritter, H. J. & Schulten, K. J. (1990), 'Three-dimensional neural net for learning visuomotor coordination of a robot arm', *IEEE Transactions on Neural Networks* **1**(1), 131–136.
URL: citeseer.ist.psu.edu/martinetz90threedimensional.html
- Mayne, D. Q., Rawlings, J. B., Rao, C. V. & Scokaert, P. O. M. (2000), 'Constrained model predictive control: Stability and optimality', *Automatica* **36**(6), 789–814.
- Mazak, J., Damen, A. A. H., Backx, A. C. P. M. & Weiland, S. (1996), 'Nonlinear transition controller design using neural networks tested on a polymerization reactor', *Proceedings of the 35th IEEE Decision and Control* **2**, 1720–1721.
- McCulloch, W. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics* **5**, 115–133.
- Milano, M., Koumoutsakos, P. & Schmidhuber, J. (2004), 'Self-organizing nets for optimization', *IEEE Transactions on Neural Networks* **15**(3), 758–765.
- Miller, III, W. T. (1994), 'Real-time neural network control of a biped walking robot', *IEEE Contr. Syst. Mag.* **14**(1), 41–48.
- Minsky, M. L. & Papert, S. A. (1969), *Perceptrons*, The MIT Press, Cambridge, MA.
- Miyamoto, H., Morimoto, J., Doya, K. & Kawato, M. (2004), 'Reinforcement learning with via-point representation', *Neural Networks* **17**(3), 299–305.
- Morasso, P. & Sanguineti, V. (1995), 'Self-organizing body-schema for motor planning.', *Journal of Motor Behavior* **26**, 131–148.
- Narendra, K. S. (1996), 'Neural networks for control: theory and practice', *Proceedings of the IEEE* **84**(10), 1385–1406.

- Narendra, K. S. & Mukhopadhyay, S. (1997), 'Adaptive control using neural networks and approximate models', *IEEE Transactions on Neural Networks* **8**(3), 475–485.
- Narendra, K. S. & Parthasarathy, K. (1990), 'Identification and control of dynamical systems using neural networks', *IEEE Transactions on Neural Networks* **1**(1), 4–27.
- Nehmzow, U. (2001), Mobile robotics: Research, applications and challenges, in 'Proc. Future Trends in Robotics', Institution of Mechanical Engineers, London.
- Nilsson, N. J. (1980), *Principles of artificial intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Nishina, S. & Kawato, M. (2004), 'A computational model of spatio-temporal dynamics in depth filling-in', *Neural Networks* **17**(2), 159–163.
- Oliveira, L. F., Lima, L. G. P., Oliveira, A. L. G. & Toffolo, R. B. G. (2005), Classificação de inversões de acordes por mapas auto-organizados de Kohonen, in 'Proceedings of the 1st International Symposium on Cognition and Musical Arts', Curitiba, PR, Brazil.
- Ott, E. (2002), *Chaos in Dynamical Systems*, 2nd edn, Cambridge University Press.
- Page, M. P. A. (1999), *Musical Networks: Parallel Distributed Perception and Performance*, The MIT Press, Cambridge, MA, USA, chapter Modelling the Perception of Musical Sequences with Self-organizing Neural Networks, pp. 175–198.
- Palli, G. & Melchiorri, C. (2006), 'Model and control of tendon-sheath transmission systems', *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'06)* pp. 988–993.
- Park, J. H., Yun, J. M. & Lee, J. M. (2007), 'Trajectory estimation of a moving object using Kalman filter and Kohonen networks', *Robotica* **25**, 567–574.
- Parncutt, R. (1989), *Harmony: a psychoacoustical approach*, Springer Series in Information Sciences, Springer-Verlag, Berlin.
- Piaget, J. (1963), *The Origins of Intelligence in Children*, Norton, New York.
- Pribe, C., Grossberg, S. & Cohen, M. A. (1997), 'Neural control for interlimb oscillations', *Biological Cybernetics* **77**, 141–152.
- Provost, J., Kuipers, B. J. & Mlikkula, R. (2006), 'Developing navigation behavior through self-organizing distinctive-state abstraction', *Connection Science* **18**(2), 159–172.

- Rameau, J. P. (1971), *Treatise on harmony*, Dover, New York.
- Rao, R. P. N. & Seinowski, T. J. (2003), 'Self-organizing neural systems based on predictive learning', *Philosophical Transactions of the Royal Society of London Series A - Mathematical Physical and Engineering Sciences* **361**(1807), 1149–1174.
- Rauber, A., Merkl, D. & Dittenbach, M. (2002a), 'The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data', *IEEE Transactions on Neural Networks* **13**(6), 1331–1341.
- Rauber, A., Pampalk, E. & Merkl, D. (2002b), Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by musical styles, in 'Proceedings of the 3rd International Symposium on Music Information Retrieval', Paris, France, pp. 71–80.
- Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning representations by back-propagation errors', *Nature (London)* **323**, 533–536.
- Russel, S. & Norvig, P. (2002), *Artificial Intelligence: a Modern Approach*, Prentice Hall Series in Artificial Intelligence, 2nd edn, Prentice Hall, Upper Saddle River.
- Sanguineti, V. & Morasso, P. (1992), 'Cortical maps for motor planning', *Proceedings of RNNS/IEEE Symposium on Neuroinformatics and Neurocomputing* **1**, 457–466.
- Schaal, S. & Sternad, D. (1998), Programmable pattern generators, in 'International Conference on Computational Intelligence in Neuroscience (ICCIN'98)', Research Triangle Park NC.
- Schoenberg, A. (1999), *Harmonia*, Editora Unesp, São Paulo.
- Shadmehr, R., Donchin, O., Hwang, E. J., Hemminger, S. E. & Rao, A. (2005), Learning dynamics of reaching, in A. Riehle & E. Vaadia, eds, 'Motor Cortex in Voluntary Movements: A distributed system for distributed functions', CRC Press.
- Shadmehr, R. & Wise, S. P. (2004), Motor learning and memory for reaching and pointing, in M. S. Gazzaniga, ed., 'The Cognitive Neurosciences', MIT Press.
- Shah-Hosseini, H. & Safabakhsh, R. (2003), 'TASOM: A new time adaptive self-organizing map', *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **33**(2), 271–282.
- Sternad, D., Dean, W. J. & Schaal, S. (2000), 'Interaction of rhythmic and discrete pattern generators in single joint movements', *Human Movement Science* **19**(4), 627–665.

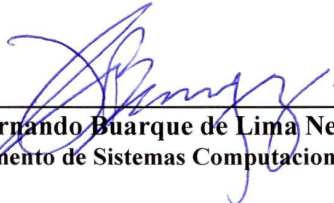
- Sutton, R. S. & Barto, A. G. (1998), *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, MA.
URL: citeseer.ist.psu.edu/sutton98reinforcement.html
- Terhardt, E. (1974), 'Pitch, consonance, and harmony', *The Journal of the Acoustical Society of America* **55**, 1061.
- Terhardt, E. (1979), 'Calculating virtual pitch.', *Hear Res* **1**(2), 155–82.
- Tillmann, B., Bharucha, J. J. & Bigand, E. (2000), 'Implicit learning of music: A self-organizing approach', *Psychological Review* **107**(4), 885–913.
- Turing, A. M. (1936), 'On computable numbers with an application to the entscheidungsproblem', *Proceedings of the London Mathematical Society* **42**(2), 230–265. supl. (1937) Vol.43, p.544-546.
- Turing, A. M. (1950), 'Computing machinery and intelligence', *Mind* **59**, 433–460.
- Ulloa, A. & Bullock, D. (2003), 'A neural network simulating human reach-grasp coordination by continuous updating of vector positioning commands', *Neural Networks* **16**(8), 1141–1160.
- Venayagamoorthy, G. K., Harley, R. G. & Wunsch, D. C. (2003), 'Implementation of adaptive critic-based neurocontrollers for turbogenerators in a multimachine power system', *IEEE Transactions on Neural Networks* **14**(5), 1047–1064.
- Verschure, P. F. M. J. (1998), 'Synthetic epistemology: The acquisition, retention, and expression of knowledge in natural and synthetic systems', *IEEE International Conference on Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence* **1**, 147–152.
URL: citeseer.ist.psu.edu/117799.html
- Villmann, T., Der, R., Herrmann, M. & Martinetz, T. M. (1997), 'Topology preservation in self-organizing feature maps: Exact definition and measurement', *IEEE Transactions on Neural Networks* **8**(2), 256–266.
- Vlassis, N. A., Dimopoulos, A. & Papakonstantinou, G. K. (1997), The probabilistic growing cell structures algorithm, in 'Proceedings of the ICANN'97', pp. 649–654.
- Wada, Y. & Kawato, M. (2004), 'A via-point time optimization algorithm for complex sequential trajectory formation', *Neural Networks* **17**(3), 353–364.

- Walter, J. A. & Schulten, K. J. (1993), 'Implementation of self-organizing neural networks for visuo-motor control of an industrial robot', *IEEE Trans. on Neural Networks* **4**(1), 86–95.
URL: citeseer.ist.psu.edu/walter93implementation.html
- Wang, L. X. & Wan, F. (2001), 'Structured neural networks for constrained model predictive control', *Automatica* **37**(8), 1235–1243.
- Waydo, S., Hauser, J., Bailey, R., Klavins, E. & Murray, R. M. (2007), 'UAV as a reliable wingman: A flight demonstration', *IEEE Transactions on Control Systems Technology* **15**(4), 680–688. California Inst. of Technol., Pasadena.
- Werbos, P. J. (1990), 'Backpropagation through time: what it does and how to do it', *Proceedings of the IEEE* **78**(10), 1550–1560.
- Werbos, P. J. (1991), 'An overview of neural networks for control', *IEEE Control Systems Magazine* **11**(1), 40–41.
- Wiener, N. (1948), *Cybernetics: Or the Control and Communication in the Animal and the Machine.*, The MIT Press, Cambridge, MA.
- Willems, D. J. M. & Haselager, W. F. G. (2003), Cooperative behavior in simulated reactive robots, in T. Heskes, P. Lucas, L. Vuurpijl & W. Wiegerinck, eds, 'Proceedings of the 15th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'03)', University of Nijmegen, Nijmegen, pp. 355–362.
- Willms, A. R. & Yang, S. X. (2006), 'An efficient dynamic system for real-time robot-path planning', *Systems, Man, and Cybernetics, Part B, IEEE Transactions on* **36**(4), 755–766.
- Willshaw, D. J. & von der Malsburg, C. (1976), 'How patterned neural connections can be set up by self-organization', *Proceedings of the Royal Society of London B* **194**, 431–445.
- Winograd, S. & Cowan, J. D. (1963), *Reliable Computation in the Presence of Noise*, The MIT Press, Cambridge, MA.
- Zeller, M., Sharma, R. & Schulten, K. (1997), Vision-based robot motion planning using a topology representing neural network, in J. Kalkkuhl, K. Hunt, R. Zbikowski & A. Dzieliński, eds, 'Applications of Neural Adaptive Control Technology', World Scientific Publishing.
- Zhao, Y. & Small, M. (2006), 'Minimum description length criterion for modeling of chaotic attractors with multilayer perceptron networks', *Transactions on Circuits and Systems I* **53**, 722–732.

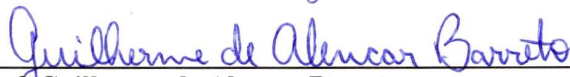
Tese de Doutorado apresentada por **Ruben Carlo Benante** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título “**Geração de Trajetórias de Estados por Mapas Auto-organizáveis com Topologia Dinâmica**”, orientada pelo **Prof. Aluizio Fausto Ribeiro Araújo** e aprovada pela Banca Examinadora formada pelos professores:



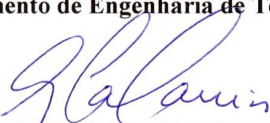
Prof. Manoel Eusébio de Lima
Centro de Informática / UFPE



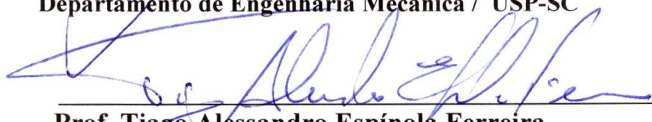
Prof. Fernando Buarque de Lima Neto
Departamento de Sistemas Computacionais / UPE



Prof. Guilherme de Alencar Barreto
Departamento de Engenharia de Teleinformática / UFC

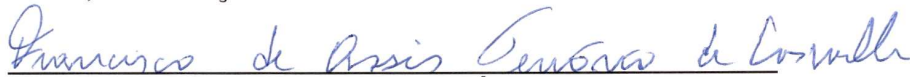


Prof. Glaucio Augusto de Palua Caurin
Departamento de Engenharia Mecânica / USP-SC



Prof. Tiago Alessandro Espínola Ferreira
Departamento de Estatística e Informática / UFRPE

Visto e permitida a impressão.
Recife, 5 de março de 2008.



Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO

Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Para citar esse documento:

BENANTE, R. C. Geração de Trajetórias de Estados por Mapas Auto-organizáveis com Topologia Dinâmica. Recife, 2008. 182f. Tese (Doutorado em Ciências da Computação) – Universidade Federal de Pernambuco.

Entrada BibTeX:

```
@PhdThesis{Benante2008phd,  
  author = {Ruben Carlo Benante},  
  title = {Gera\c{c}\~ao de Trajet\'orias de Estados por  
          Mapas Auto-organiz\'aveis com Topologia Din\^amica},  
  address = {Recife},  
  year = {2008},  
  pages = {182},  
  type = {Doutorado em Ci\^encias da Computa\c{c}\~ao},  
  school = {Universidade Federal de Pernambuco}  
}
```

Tipografado em L^AT_EX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis), versão *rcb*.

A fonte do corpo do texto é a Computer Modern Roman 12pt.