David Saraiva Farias Fernandes

Análise de Similaridades de Modelagem no Emprego de Técnicas Conexionistas e Evolutivas da Inteligência Computacional Visando à Resolução de Problemas de Otimização Combinatorial: Estudo de Caso – Problema do Caixeiro Viajante

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo (EPUSP) como parte dos requisitos para a obtenção do Título de Mestre em Engenharia Elétrica no Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

Área: Inteligência Artificial

Orientador:

Professor Doutor Marco Túlio Carvalho de Andrade

São Paulo
2009

David Saraiva Farias Fernandes

Análise de Similaridades de Modelagem no Emprego de Técnicas Conexionistas e Evolutivas da Inteligência Computacional Visando à Resolução de Problemas de Otimização Combinatorial: Estudo de Caso – Problema do Caixeiro Viajante

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo (EPUSP) como parte dos requisitos para a obtenção do Título de Mestre em Engenharia Elétrica no Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

Área: Inteligência Artificial

Orientador:

Professor Doutor Marco Túlio Carvalho de Andrade

São Paulo
2009

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 26 de junho de 2009.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Fernandes, David Saraiva Farias

Análise de similaridade de modelagem no emprego de técnicas conexistas e evolutivas da inteligência computacional visando à resolução de problemas de otimização combinatorial : estudo de caso, problema do caixeiro viajante / D.S.F. Fernandes. -- ed.rev. -- São Paulo, 2009.

126 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Inteligência artificial 2.Computação evolutiva (Análise) 3.Redes neurais (Análise) 4. Otimização combinatoria I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

“Alguém sempre espera o melhor de você.” – Autor desconhecido.

“The only place where success comes before work is in the dictionary.” – Albert Einstein.

DEDICATÓRIA

Para: “Todos que me apoiaram neste período de estudos”.

AGRADECIMENTOS

Agradeço a Deus, pela vida que tenho, pela saúde que possuo, pelas oportunidades que me são oferecidas e por toda a experiência e conhecimento que tenho e terei ao longo de minha vida.

Agradeço ao meu orientador, Professor Marco Túlio, pela orientação e suporte ao longo deste período de pesquisas.

Agradeço aos meus pais Anne e Fernandes e ao meu irmão José Alberto pelo auxílio e pelo apoio que sempre me deram.

Agradeço a todos meus colegas e amigos que me auxiliaram de alguma forma na realização deste trabalho.

RESUMO

Este trabalho realiza uma análise dos modelos pertencentes à Computação Neural e à Computação Evolutiva visando identificar semelhanças entre as áreas e sustentar mapeamentos entre as semelhanças identificadas. Neste contexto, a identificação de similaridades visando à resolução de problemas de otimização combinatorial resulta em uma comparação entre a Máquina de Boltzmann e os Algoritmos Evolutivos binários com população composta por um único indivíduo pai e um único indivíduo descendente. Como forma de auxiliar nas análises, o trabalho utiliza o Problema do Caixeiro Viajante como plataforma de ensaios, propondo mapeamentos entre as equações da Máquina de Boltzmann e os operadores evolutivos da Estratégia Evolutiva (1+1)-ES.

ABSTRACT

An analysis between the Evolutionary Computation and the Neural Computation fields was presented in order to identify similarities and mappings between the theories. In the analysis, the identification of similarities between the models designed for combinatorial optimization problems results in a comparison between the Boltzmann Machine and the Two-Membered Evolutionary Algorithms. In order to analyze the class of combinatorial optimization problems, this work used the Traveling Salesman Problem as a study subject, where the Boltzmann Machine equations were used to implement the evolutionary operators of an Evolution Strategy (1+1)-*ES*.

SUMÁRIO

DEDICATÓRIA.....	iv
AGRADECIMENTOS	v
RESUMO	vi
ABSTRACT	vii
SUMÁRIO	viii
LISTA DE FIGURAS	x
LISTA DE TABELAS	xii
1. INTRODUÇÃO	1
1.1 Justificativa e Motivação	1
1.2 Objetivos	4
1.3 Metodologia	4
1.3.1 Estratégia	4
1.3.2 Plano de Desenvolvimento da Pesquisa.....	4
1.3.3 Escopo do Trabalho de Pesquisa	5
1.3.4 Análise de Risco e Custo	6
1.4 Estrutura do Texto	6
2. CONCEITOS E TERMINOLOGIA.....	8
2.1 Inteligência Artificial (IA) e Inteligência Computacional (IC)	8
2.2 Computação Neural	9
2.2.1 Neurônios Artificiais	10
2.2.2 Algoritmos de aprendizado	13
2.2.3 Redes Neurais Artificiais.....	23
2.2.4 Perceptrons de Múltiplas Camadas	27
2.2.5 Rede Neural de Kohonen.....	30
2.2.6 Rede Neural de Hopfield.....	36
2.2.7 Máquina de Boltzmann	41
2.3 Computação Evolutiva	50
2.3.1 Algoritmos Genéticos	53
2.3.2 Programação Evolutiva.....	57

2.3.3	Estratégias Evolutivas.....	58
3.	ANÁLISE COMPARATIVA ENTRE MODELOS DA INTELIGÊNCIA COMPUTACIONAL.....	64
3.1	Análise Geral	64
3.1.1	Visão Funcional.....	64
3.1.2	Visão Estrutural.....	70
3.2	Análise Detalhada.....	72
3.2.1	Aplicação: Problema do Caixeiro Viajante	72
3.2.2	Análise Funcional.....	73
3.2.3	Análise Estrutural	86
3.2.4	Mapeamentos	94
3.2.5	Considerações Finais do Capítulo	99
4.	CONSIDERAÇÕES FINAIS.....	102
4.1	Cumprimento dos Objetivos Propostos	102
4.2	Contribuições do Trabalho.....	103
4.3	Conclusões	103
4.4	Trabalhos Futuros.....	104
5.	REFERÊNCIAS	106
6.	LISTA DE ABREVIATURAS E PALAVRAS RESERVADAS.....	111
7.	LISTA DE SÍMBOLOS	112
8.	GLOSSÁRIO	113

LISTA DE FIGURAS

Figura 1.1 :	Diagrama de Gajski-Kuhn [Andrade-2002, pp. 42].....	1
Figura 1.2 :	Análise do Diagrama de Gajski & Kuhn [Andrade-2002, pp. 43].....	3
Figura 2.1 :	Inteligência Artificial e Inteligência Computacional.	9
Figura 2.2 :	Neurônio Artificial.....	10
Figura 2.3 :	Funções de transferência: (a) Degrau, (b) Sigmóide e (c) <i>Signal</i> e (d) Tangente Hiperbólica.....	12
Figura 2.4 :	Neurônio Artificial em uma Rede Neural.	13
Figura 2.5 :	Neurônio k contido em uma rede neural.	14
Figura 2.6 :	Aprendizado por correção de erro.	15
Figura 2.7 :	Arquitetura neural baseada na regra de aprendizado competitivo.	19
Figura 2.8 :	Máquina de Boltzmann.	21
Figura 2.9 :	Cadeia de Markov para representação dos estados da máquina de Boltzmann com 2 bits.	22
Figura 2.10 :	Estruturas das Redes Neurais.	24
Figura 2.11 :	Perceptron de Múltiplas Camadas (PMC).	27
Figura 2.12 :	Modelo rede neural de Kohonen.....	30
Figura 2.13 :	Distância entre neurônios para mapa (a) retangular e (b) hexagonal.	31
Figura 2.14 :	- Arquitetura de rede de Kohonen utilizada para resolução do problema do caixeiro viajante.....	35
Figura 2.15 :	- Interpretação de um percurso em rede de Kohonen com pesos sinápticos inicializados aleatoriamente [Fausett-1993].	35
Figura 2.16 :	- Percurso resultante da execução do algoritmo da rede de Kohonen (à esquerda) e a posição das cidades no mapa 2D (à direita) [Fausett-1993].....	36
Figura 2.17 :	Rede de Hopfield.	37
Figura 2.18 :	Rede de Hopfield para o problema do Caixeiro Viajante.	40
Figura 2.19 :	Máquina de Boltzmann com neurônios desconectados.....	44
Figura 2.20 :	<i>Crossover</i> de um ponto.	55
Figura 2.21 :	<i>Crossover</i> multiponto.....	56
Figura 2.22 :	<i>Crossover</i> uniforme.	56
Figura 3.1	- Diagrama de contexto para Aproximação de Funções.....	66
Figura 3.2	- Diagrama de contexto de sistemas de Reconhecimento de Padrões e Classificação.	66
Figura 3.3	- Diagrama de contexto de sistemas de Reconhecimento de Padrões e Classificação.	67
Figura 3.4	- Diagrama de contexto de sistemas de Memória Associativa.	68
Figura 3.5	Diagrama de contexto de sistemas de Agrupamento de Dados (<i>Clustering</i>).	68
Figura 3.6	- Diagrama de contexto de sistemas de Otimização Combinatorial.	69
Figura 3.7	- Interseção de aplicações para CN e CE.	70
Figura 3.8	- Estrutura de classes da CN e CE.	71
Figura 3.9	- Estrutura dos sistemas neurais e evolutivos, segundo dois níveis distintos de abstração.....	71
Figura 3.10	- Intersecção estrutural.....	72
Figura 3.11	Arquitetura neural para problema do caixeiro viajante com c cidades.....	74
Figura 3.12	Representação unificada adotada para problema do caixeiro viajante com $c > 2$ cidades.	77
Figura 3.13	- Interface comum para CN e CE visando resolver o problema do caixeiro viajante.....	77
Figura 3.14	Comparação dos algoritmos genéricos da Máquina de Boltzmann para Otimização e Algoritmos Evolutivos binários com retorno único.	82
Figura 3.15	Comparação dos procedimentos de Inicialização da Máquina de Boltzmann para Otimização e dos Algoritmos Evolutivos binários com único pai.....	83
Figura 3.16	Comparação dos procedimentos de controle de Ciclo da Máquina de Boltzmann para Otimização e dos Algoritmos Evolutivos genéricos.	83
Figura 3.17	Comparação dos procedimentos de controle de Ciclo da Máquina de Boltzmann para Otimização e dos Algoritmos Evolutivos genéricos.	84
Figura 3.18	Comparação dos procedimentos de controle de Ciclo da Máquina de Boltzmann para Otimização e dos Algoritmos Evolutivos genéricos.	85
Figura 3.19	- Estrutura expandida da rede Perceptron de Múltiplas Camadas.....	87
Figura 3.20	- Estrutura expandida da rede de Kohonen.	88
Figura 3.21	- Estrutura expandida da rede de Hopfield.	89
Figura 3.22	- Estrutura expandida da rede de Boltzmann.	90
Figura 3.23	- Estrutura expandida da Computação Evolutiva.	91

Figura 3.24	Análise do Diagrama de Gajski & Kuhn [Andrade-2002, pp. 43]	99
Figura 3.25	Equivalências identificadas entre Máquina de Boltzmann para Otimização e Algoritmos Evolutivos binários com um pai e um descendente para o Problema do Caixeiro Viajante.	101

LISTA DE TABELAS

Tabela 2.1 :	Propriedades das redes neurais.	26
Tabela 3.1 :	Classes de utilização de Computação Neural e Computação Evolutiva.....	65
Tabela 3.2 :	Comparativo entre algoritmos da Máquina de Boltzmann e Algoritmos Evolutivos.	81

1. INTRODUÇÃO

Neste capítulo são apresentados os objetivos da pesquisa assim como as motivações iniciais que desencadearam nesta Dissertação.

Nos tópicos seguintes serão descritos os conceitos introdutórios relacionados com este trabalho, os objetivos, as motivações, a metodologia a ser aplicada para obtenção dos resultados desejados e será apresentada a estrutura da dissertação.

1.1 Justificativa e Motivação

Este trabalho dá prosseguimento à pesquisa iniciada por Marco T. Andrade em 2002 [Andrade-2002]. No trabalho realizado as teorias da Inteligência Computacional foram interpretadas com base em conceitos utilizados em projetos de sistemas VLSI (*Very Large Scale Integration*), onde os circuitos digitais podem ser visualizados através de três óticas distintas, uma ótica Funcional, uma Estrutural e uma Geométrica. Este conceito de visualização foi formalizado no trabalho realizado por Daniel D. Gajski, e Robert H. Kuhn em 1983 [Gajski&Kuhn-1983], onde a representação das diferentes visões é realizada através do gráfico denominado Diagrama de Gajski-Kuhn (Figura 1.1).

Figura 1.1 : Diagrama de Gajski-Kuhn [Andrade-2002, pp. 42]

Neste diagrama é exibida uma representação composta por três eixos distintos centrados em uma origem, onde cada um dos eixos do gráfico representa uma diferente forma de visualização de um sistema microeletrônico. Ao longo dos eixos tem-se a representação do circuito em diferentes níveis de granularidade, ou seja, uma representação que varia de baixo nível de abstração, representado pela origem dos eixos, para alto nível, na medida em que os

eixos distanciam-se da origem. Neste diagrama a origem dos eixos representa o sistema físico enquanto que os círculos concêntricos identificam pontos correspondentes de mesmo nível de abstração [Andrade-2002, pp. 41]. Segundo Gajski e Kuhn [Gajski&Kuhn-1983], cada uma das óticas possui características específicas e evidenciam aspectos distintos do circuito em análise:

- Representação Estrutural: tem como foco a visualização dos Componentes lógicos que constituem o sistema. Neste modo de visão tem-se a representação de componentes do circuito como transistores, etc.
- Representação Funcional: tem como foco a visualização do Comportamento, não sendo do interesse desta ótica nem a Física geométrica e nem os Componentes lógicos que o implementam. Este modo é comumente visualizado através de uma representação do sistema em forma de tabela verdade ou expressão booleana.
- Representação Geométrica: tem como objetivo representar a Física geométrica do sistema não sendo de interesse desta representação o Comportamento e nem os Componentes lógicos. Neste modo de visão tem-se comumente a representação do circuito em função de seus módulos que podem ser mapeados geometricamente no espaço, como células, clusters, partições físicas, etc.

Com base nas características destas representações, Andrade-2002 realizou uma análise de três diferentes modelos da Inteligência Computacional de forma análoga aos conceitos propostos pelo diagrama de Gajski-Kuhn. As análises foram realizadas envolvendo a Computação Fuzzy/Nebulosa (CF), a Computação Evolutiva (CE) e a Computação Neural (CN), que são descritas como sub-áreas pertencentes à denominada Inteligência Computacional (IC). Nas análises realizadas cada uma das áreas foi associada com uma das três representações descritas no diagrama de Gajski-Kuhn: i.e. a representação Comportamental, a Geométrica e a Estrutural. Esta associação foi realizada com base em propriedades de cada modelo:

- Computação Neural: área de IC baseada na utilização de redes de neurônios artificiais. Nesta visão os sistemas são representados como “caixa branca” onde “a estrutura de plataforma computacional que o implementa está exposta, mas seu comportamento está escondido, ou implícito” [Andrade-2002, pp. 43]. Como exemplo, para uma visão Estrutural do diagrama de Gajski-Kuhn, uma visualização de transistores interconectados (ou outros componentes) pode ser considerada semelhante a uma estrutura de neurônios artificiais.
- Computação Nebulosa: área de IC baseada no uso de sentenças lingüísticas. Nesta visão os sistemas são representados como “caixa preta”, onde “o comportamento está exposto mas a estrutura da plataforma computacional que implementa está escondida” [Andrade-2002, pp. 42]. Como exemplo, para uma visão Comportamental do diagrama de Gajski-Kuhn, a descrição de um sistema em forma de tabelas verdade (ou outras expressões funcionais como algoritmos, etc.) pode ser considerada semelhante a uma representação através de regras nebulosas que descrevem o comportamento de um sistema.

- **Computação Evolutiva:** área de IC que utiliza algoritmos baseados em conceitos evolutivos. Nesta visão os sistemas são representados como “caixa cinza”, onde as “propriedades estão expostas, e a estrutura e o comportamento estão ocultos” [Andrade-2002, pp. 43]. Como exemplo, para uma visão Geométrica do diagrama de Gajski-Kuhn, a localização de células, clusters, etc. em um sistema microeletrônico pode ser considerada semelhante à localização de genes dentro de um genótipo, etc.

Com base nas considerações realizadas, o autor gerou uma correspondência entre o diagrama de Gajski-Kuhn e as três áreas da IC, conforme apresentada na figura abaixo (Figura 1.2).

Figura 1.2 : Análise do Diagrama de Gajski & Kuhn [Andrade-2002, pp. 43]

Com base neste comparativo, foi definida uma linha de pesquisa com o objetivo de estudar as semelhanças entre as diferentes áreas da IC. Neste contexto, outros trabalhos foram realizados, dentre os quais pode-se citar os trabalhos [Caversan&Andrade-2005] e [Caversan&Fialho&Andrade-2007], onde foram analisadas as relações entre CF e CN, e o trabalho [Fialho-2007], que abordou a CF e a CE. Outras abordagens possíveis envolveriam análises entre CN e a CE ou as três áreas em conjunto. Como forma de dar prosseguimento à linha de pesquisa, este trabalho irá focar na análise da CN e da CE.

1.2 Objetivos

Este trabalho tem como objetivo apresentar um procedimento comparativo entre a Computação Evolutiva (CE) e a Computação Neural (CN), de forma a identificar semelhanças entre as teorias e sustentar mapeamentos entre as semelhanças identificadas. Baseado em semelhanças existentes para a resolução de problemas de otimização, este trabalho irá propor um mapeamento entre elementos de uma Máquina de Boltzmann modelado para o Problema do Caixeiro Viajante e um Algoritmo Evolutivo binário com população composta por um único indivíduo pai e um único indivíduo descendente. Para descrição dos operadores evolutivos, será utilizada a notação padrão para a Estratégia Evolutiva (1+1)-ES.

1.3 Metodologia

Será descrita através dos seguintes tópicos:

- Estratégia.
- Análise.
- Conclusões e considerações finais.

1.3.1 Estratégia

O procedimento de análise será baseado em uma abordagem “*top-down*”, avaliando as teorias inicialmente como um único módulo, e seguindo com a decomposição em módulos de menor complexidade. As abordagens serão realizadas sob duas óticas distintas, as óticas funcionais descrevendo os algoritmos, e estruturais descrevendo as principais classes e entidades manipuladas pelos algoritmos. Também será abordado um estudo de caso aplicado, que visa auxiliar no processo de análise.

1.3.2 Plano de Desenvolvimento da Pesquisa

Para obtenção dos resultados desejados, a pesquisa será desenvolvida em três etapas distintas:

- Etapa 1 – Pesquisa Básica
 - Estudo dos conceitos pertencentes à Computação Neural.
 - Estudo dos conceitos pertencentes à Computação Evolutiva.
- Etapa 2 – Pesquisa Aplicada
 - Análise da ótica funcional de modelos pertencentes à Computação Neural e Computação Evolutiva.
 - Análise da ótica estrutural de modelos pertencentes à Computação Neural e Computação Evolutiva.

- Análise comparativa visando à resolução de problemas de otimização.
- Análise da Máquina de Boltzmann para resolução do problema do caixeiro viajante.
- Etapa 3 – Pesquisa Tecnológica
 - Mapeamento de equações de uma Máquina de Boltzmann para resolução do problema do caixeiro viajante em operadores evolutivos de uma Estratégia Evolutiva (1+1)-ES.
 - Comparação dos conceitos estudados com o diagrama de Gajski & Kuhn.
 - Considerações e Conclusões Finais.

1.3.3 Escopo do Trabalho de Pesquisa

Como apresentado anteriormente, este trabalho pertence a uma linha de pesquisa que visa identificar semelhanças entre CN e CE. Dadas as restrições de tempo para desenvolvimento da pesquisa, as análises foram realizadas sobre um grupo restrito de Redes Neurais Artificiais (RNAs) e Algoritmos Evolutivos (AEs), sendo o foco principal das análises o Problema do Caixeiro Viajante para a Máquina de Boltzmann e para a Estratégia Evolutiva (1+1)-ES.

Referente à CN, este trabalho irá abordar os conceitos básicos das seguintes redes neurais:

- Redes Perceptrons de múltiplas camadas.
- Redes Kohonen.
- Redes Hopfield.
- Máquina de Boltzmann.

Apesar de não documentadas neste texto, também foram consideradas como parte do escopo da pesquisa as seguintes redes neurais:

- Redes Radial-Basis Function.
- Rede de Willshaw-von der Malsburg.
- Rede Brain-State-in-a-Box.
- Rede Sigmoid Belief.
- Helmholtz Machine.

Referente à CE, este trabalho irá abordar os conceitos básicos das seguintes áreas:

- Algoritmos Genéticos.
- Programação Evolutiva.
- Estratégia Evolutiva.

1.3.4 Análise de Risco e Custo

Os principais riscos e custos relacionados com o projeto podem ser listados abaixo:

I. Riscos

- **Cronograma:** A pesquisa possui um tempo limitado para conclusão, que poderá implicar em restrições na abrangência dos estudos realizados e conseqüentemente restrições na abrangência das conclusões geradas. Como forma de minimizar este risco, os trabalhos serão orientados por professores qualificados e especialistas das áreas teóricas abordadas neste trabalho.
- **Publicação:** Uma das atividades previstas para a pesquisa é publicação de um artigo em congresso nacional, internacional ou periódico. Esta atividade deve ser realizada dentro do prazo concebido para o trabalho.

II. Custos

- Este trabalho se propõe a avaliar conceitualmente duas áreas distintas. Desta forma, não serão considerados custos materiais específicos de projeto (hardware, etc.).

1.4 Estrutura do Texto

O texto foi estruturado em 8 capítulos distintos:

- **Introdução:** Apresentação do problema e do desenvolvimento do trabalho.
- **Capítulo 2:** Apresentação dos conceitos teóricos relacionados com as redes neurais e algoritmos evolutivos de interesse do trabalho e importantes na unificação de metodologias e na construção do conhecimento sobre o assunto. Como citado anteriormente, as principais Redes Neurais consideradas serão as denominadas Perceptrons de Múltiplas Camadas, redes de Kohonen, redes de Hopfield e Máquinas de Boltzmann. Os principais algoritmos evolutivos avaliados serão os Algoritmos Genéticos, as Estratégias Evolutivas e a Programação Evolutiva.
- **Capítulo 3:** Análise comparativa utilizando uma abordagem “*top-down*”, tanto sob uma ótica funcional quanto estrutural. Também está previsto um estudo de caso específico, o Problema do Caixeiro Viajante, como forma de auxiliar no procedimento de análise. Neste capítulo será descrito um procedimento de análise para identificação de semelhanças entre a CN e a CE assim como será sugerido um mapeamento entre elementos das duas áreas teóricas.

- Capítulo 4: Considerações Finais, onde serão avaliados os cumprimentos dos objetivos, destacadas as contribuições do trabalho e avaliadas as possibilidades de continuidade da pesquisa.
- Capítulo 5 a 8: Referências, Lista de Abreviaturas, Lista de Símbolos e Glossário.

2. CONCEITOS E TERMINOLOGIA

Neste capítulo serão apresentados os conceitos teóricos envolvidos no escopo deste Trabalho. Pretende-se estabelecer uma base teórica para posterior desenvolvimento das análises entre a Máquina de Boltzmann (Computação Neural) e uma Estratégia Evolutiva (1+1)-ES (Computação Evolutiva) para resolução do Problema do Caixeiro Viajante.

Nos tópicos seguintes serão descritos os conceitos relacionados com a Inteligência Artificial, Inteligência Computacional, Computação Neural e Computação Evolutiva.

2.1 Inteligência Artificial (IA) e Inteligência Computacional (IC)

Atualmente não existe uma definição precisa para o termo “Inteligência Artificial”, sendo a origem desta inexistência o caráter subjetivo que o termo “inteligência” possui. Desta forma não entrando na discussão do que seria a “inteligência” propriamente dita, tem-se na Inteligência Artificial uma ciência que busca construir entidades inteligentes, não impondo restrições de como proceder para obtenção de tal característica. Uma discussão mais detalhada sobre as diversas definições existentes para o termo “Inteligência Artificial” pode ser vista em [Russel&Norvig-2003].

Abaixo da Inteligência Artificial (IA) é então definido um subgrupo de pesquisa denominado Inteligência Computacional (IC) . De forma equivalente à IA, a IC também não possui uma definição precisa. Não entrando no mérito da definição do que seria a IC, têm-se abaixo desta área três sub-áreas distintas: a Computação Neural (CN), a Computação Evolutiva (CE) e a Computação Fuzzy/Nebulosa (CF).

A CN, como o próprio nome sugere, possui modelos baseados na utilização de redes de neurônios artificiais, modelos inspirados nas redes neurais biológicas que compõe o sistema nervoso dos seres vivos. Esta teoria representa a abordagem Conexionista da IC.

A CE, por sua vez, é uma teoria baseada nos conceitos evolutivos que regem a evolução dos seres biológicos. Em essência, esta teoria visa obter sistemas capazes de adaptar a ambientes ou evoluir e otimizar soluções de problemas, utilizando para isto operadores evolutivos como a Recombinação, Mutação, Avaliação e Seleção. Esta teoria representa a abordagem Evolutiva da IC.

Por fim, a CF é uma teoria baseada na utilização de instruções lingüísticas imprecisas como forma de programação e controle de sistemas. Basicamente, a teoria tem como objetivo gerar sistemas capazes de simular o modo aproximado do raciocínio humano, ou seja, um modo capaz de raciocinar e agir em ambientes caracterizados por incertezas e imprecisões. A CF representa o paradigma Simbólico da IC.

Na Figura 2.1 encontra-se uma representação gráfica da estrutura composta pela IA, IC e as áreas CN, CE e CF.

Figura 2.1 : Inteligência Artificial e Inteligência Computacional.

Referente à definição do termo “Inteligência Computacional”, uma análise mais detalhada pode ser vista em [Duch-2007].

Dentre as três teorias destacadas, serão de interesse deste trabalho as áreas da CN e a da CE. Nos tópicos seguintes estas duas teorias serão analisadas em maiores detalhes.

2.2 Computação Neural

A CN representa a abordagem Conexionista da IC. Esta área é composta por uma variedade de modelos, que são denominados “Redes de Neurônios Artificiais” ou RNAs. Estes modelos têm como objetivo gerar estruturas artificiais que reproduzem propriedades das redes neurais biológicas, conseqüentemente obtendo sistemas com capacidade de adaptação, processamento paralelo, tolerância a falhas, entre outras características intrínsecas a estas estruturas biológicas.

Como o próprio nome sugere, uma RNA utiliza um modelo artificial do neurônio biológico. Os Neurônios Artificiais são então interconectados em forma de rede, resultando em uma estrutura que pode ser utilizada tanto para aprendizado como para execução de operações.

Dentre as diversas redes neurais existentes, destacam-se como as principais de interesse deste trabalho os Perceptrons de Múltiplas Camadas, redes de Kohonen, redes de Hopfield e a Máquina de Boltzmann. Estas redes foram escolhidas com base nas seguintes propriedades:

- 1) Tipo de neurônio artificial: nesta categoria têm-se duas classes distintas, uma classe de redes que utilizam o neurônio artificial estocástico (utilizada pela máquina de Boltzmann) e outra classe baseada na utilização do neurônio determinístico (utilizada pelas demais redes neurais abordadas neste trabalho).
- 2) Tipo de algoritmo de aprendizado: serão analisadas 5 classes distintas (conforme apresentado em [Haykin-1999]), uma classe de “aprendizado por correção de erro” que engloba o Perceptron de Múltiplas Camadas, uma classe de “aprendizado baseado em memória” e “aprendizado Hebbiano” utilizados pela rede de Hopfield, uma classe

de “aprendizado competitivo” utilizado pela rede de Kohonen e o “aprendizado de Boltzmann”, específico para a máquina de Boltzmann.

- 3) Tipo de realimentação: nesta categoria têm-se duas classes distintas, uma classe de arquitetura sem realimentação (como a utilizada pelos Perceptrons de Múltiplas Camadas e pelas redes de Kohonen) e uma classe com arquitetura realimentada (como a utilizada pelas redes de Hopfield e Máquina de Boltzmann).

Outras redes neurais abordando as propriedades destacadas acima podem ser encontradas na literatura, como pode ser identificado em [Haykin-1999], porém, as redes de PMC, Kohonen, Hopfield e Máquina de Boltzmann se mostram suficientes para um estudo abrangente das principais propriedades da CN. Antes da descrição detalhada de cada uma destas RNAs, serão apresentados os diferentes tipos de neurônios artificiais e algoritmos de aprendizado citados acima.

2.2.1 Neurônios Artificiais

O neurônio artificial pode ser generalizado como um modelo matemático que procura simular a forma, o comportamento e as funções que a célula neural biológica possui. Basicamente, existem dois modelos neurais de interesse para este trabalho, o modelo determinístico e o modelo estocástico.

2.2.1.1 Modelo Determinístico

Figura 2.2 : Neurônio Artificial.

O modelo determinístico é um modelo de neurônio artificial que possui o valor do sinal de saída definido deterministicamente pelos sinais de entrada aplicados ao modelo e os pesos sinápticos no instante do cálculo. Basicamente, o neurônio artificial é composto por três módulos principais, os pesos sinápticos, um módulo somador, e uma função de transferência conforme representado na Figura 2.2.

Além dos três módulos destacados é necessário considerar os sinais de entrada (incluindo o *Bias* – representado pelo sinal b) como forma de equacionar o modelo neural. Desta forma o modelo obtido será composto pelos seguintes elementos:

- 1) Conjunto de sinapses ou *links* de conexão: No modelo apresentado este módulo é representado pelas variáveis x_i .
- 2) *Bias*: Representado pela variável b . Este elemento representa um sinal de entrada externo ao sistema, independente dos demais neurônios da rede.
- 3) Conjunto de pesos sinápticos associados às conexões: Representado pela variável w_i . Esta é uma variável numérica e tem como objetivo simular a intensidade das conexões neurais biológicas.
- 4) Somador de sinais (combinador linear): Matematicamente, este módulo realiza a soma dos sinais entrada x_i multiplicados pelos pesos da conexão w_i , também levando em consideração o sinal do *bias* b . O sinal calculado é representado pela variável v . Este valor é denominado de “sinal interno” do neurônio e pode ser descrito pela equação:

$$v = \sum_{i=1}^m x_i \cdot w_i + b \quad (2.1)$$

- 5) Função de transferência : Permite uma limitação do valor de saída do modelo de forma a simular a saturação dos sinais de saída dos neurônios biológicos. A função de transferência $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ tem como objetivo realizar o cálculo do sinal de saída y a partir do sinal interno v :

$$y = \varphi(v) \quad (2.2)$$

Algumas das funções de transferência comumente utilizadas em redes neurais podem ser visualizadas na figura abaixo [Haykin-1999].

a) $y = \text{Step}(v)$

b) $y = \text{Sigmoid}(v) = 1/(1 + e^{-v})$

Figura 2.3 : Funções de transferência: (a) Degrau, (b) Sigmóide e (c) *Signal* e (d) Tangente Hiperbólica.

2.2.1.2 Modelo Estocástico

O modelo estocástico representa uma variação do modelo determinístico, possuindo desta forma os mesmos elementos em sua composição: sinais de entrada x_i , *bias* b , pesos sinápticos w_i , somador de sinais e função de transferência φ . A diferença entre o modelo estatístico e o determinístico se deve à existência de um fator probabilístico de inversão do sinal de saída, o que exige um sinal de saída binário para o neurônio. Para o modelo estocástico, a função de transferência pode ser definida por:

$$y = \begin{cases} 1 & \text{com probabilidade } P(v) \\ 0 & \text{com probabilidade } (1 - P(v)) \end{cases} \quad (2.3)$$

Nesta equação, a probabilidade $P(v)$ é definida como:

$$P(v) = \frac{1}{1 + \exp(-v/T)} \quad (2.4)$$

onde o parâmetro T representa uma variável de controle da probabilidade de inversão do sinal de saída.

Na medida em que o parâmetro de controle T descrito acima se aproxima de zero, o modelo neural passa a se aproximar de um modelo determinístico. Segundo [Haykin-1999], esta variável é denominada “pseudo-temperatura” e é utilizada para controlar o nível de ruído “térmico” do neurônio e conseqüentemente a incerteza no sinal de saída. Este modelo neural é utilizado na Máquina de Boltzmann.

2.2.2 Algoritmos de aprendizado

Os algoritmos de aprendizado têm como objetivo realizar ajustes nos parâmetros livres das redes neurais, ou seja, os pesos sinápticos w_i e o *bias* b dos neurônios que compõe a rede. O objetivo destes algoritmos é de prover uma capacidade de aprendizado a partir de exemplos ou experiências passadas. Neste trabalho serão abordadas 5 classes distintas de algoritmos de aprendizado:

- Aprendizado por correção de erro
- Aprendizado baseado em memória
- Aprendizado Hebbiano
- Aprendizado competitivo
- Aprendizado de Boltzmann

Para a representação dos neurônios estruturados em forma de rede será necessário utilizar uma versão apropriada do modelo neural descrito na Figura 2.2. Na representação proposta anteriormente para um único neurônio foram identificados os sinais de entrada x_i e seus pesos sinápticos w_i associados, porém, para a representação dos neurônios em uma rede, surge a necessidade de identificação destas entradas e pesos separadamente para cada um dos neurônios. Dada esta necessidade será utilizada uma representação mais detalhada, conforme apresentada na Figura 2.4.

Figura 2.4 : Neurônio Artificial em uma Rede Neural.

Levando em consideração a existência de vários neurônios em rede, pode-se reescrever a equação de cálculo do sinal interno do k -ésimo neurônio da rede através da seguinte equação:

$$v_k = \sum_{i=0}^m x_i \cdot w_{ki} + b_k \quad (2.5)$$

A diferença básica na representação do modelo neural da Figura 2.2 e da Figura 2.4, está na utilização do índice k adicional para a identificação do neurônio em uma rede composta por vários neurônios artificiais. Com este índice, pode-se reescrever a saída do neurônio através da equação:

$$y_k = \varphi(v_k) \quad (2.5b)$$

Uma representação do neurônio k contido em uma rede neural pode ser vista na figura abaixo.

Figura 2.5 : Neurônio k contido em uma rede neural.

Em uma rede neural, comumente é utilizada uma representação dos pesos sinápticos da rede em forma de uma matriz $\vec{W}$ de $m \times m$ elementos:

$$\vec{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,m} \\ w_{2,1} & w_{2,2} & \dots & w_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,m} \end{bmatrix} \quad (2.6)$$

Nesta representação é possível identificar qual o peso sináptico conectando um neurônio <linha> com um neurônio <coluna>. Para as conexões sinápticas $w_{k,k}$, que em teoria conecta o neurônio k com ele mesmo, estes elementos da matriz podem também ser utilizados para descrever o valor do *bias* em redes sem auto realimentação das unidades neurais.

Com base nestas definições, pode-se então prosseguir às análises dos diferentes algoritmos de aprendizado citados anteriormente.

2.2.2.1 Aprendizado por correção de erro

Esta regra de aprendizado tem como objetivo, realizar um ajuste dos pesos sinápticos da rede neural através da apresentação de uma lista de exemplos de treinamento inicialmente conhecidos para uma função ou problema. Cada exemplo de treinamento da lista é definido como uma associação entre sinais de entrada conhecidos e seus respectivos sinais de saída esperados. Pode-se então definir a lista de exemplos de treinamento através do conjunto:

$$\left\{ \left(\vec{x}_i, \vec{d}_i \right) \right\}_{i=1}^N \quad (2.7)$$

Onde, N representa o número de exemplos de treinamento contidos na lista, $\vec{x}_i$ representa o sinal de entrada e $\vec{d}_i$ representa a saída esperada do exemplo de treinamento.

Com base nestes exemplos o algoritmo aplica um procedimento cíclico, onde o sinal de entrada de um exemplo de treinamento escolhido aleatoriamente é apresentado à rede e o sinal de saída obtido é comparado ao sinal de saída esperado do exemplo. Com base na comparação é possível calcular um valor de erro que será utilizado como base para a realização dos ajustes nos parâmetros livres da rede, i.e. pesos sinápticos e *bias*. O mecanismo utilizado pelo algoritmo de aprendizado pode ser representado pelo diagrama realimentado apresentado na Figura 2.6.

Figura 2.6 : Aprendizado por correção de erro.

Neste diagrama, os seguintes módulos podem ser identificados:

- Sinal de Entrada: Representado pelo vetor $\vec{x}(n) = \{x_1(n), \dots, x_E(n)\}$ onde E denota o número de neurônios pertencentes à camada de entrada da rede neural.
- Rede Neural: Representada pela função $f(\vec{W}(n))$ que calcula um sinal de saída $\vec{y}(n) = \{y_1(n), \dots, y_k(n), \dots, y_S(n)\}$ baseado na matriz de pesos sinápticos $\vec{W}(n)$ da rede e nos sinais de entrada $\vec{x}(n)$ aplicados à rede.

- Saída Obtida: . Neste exemplo a rede neural gera uma saída $\vec{y}(n) = \{y_1(n), \dots, y_k(n), \dots, y_S(n)\}$ onde S denota o número de neurônios pertencentes à camada de saída. A saída de um neurônio k é dado pelo elemento $y_k(n)$ do vetor de sinais de saída.
- Saída Esperada: Saída representada em forma de vetor $\vec{d}(n) = \{d_1(n), \dots, d_k(n), \dots, d_S(n)\}$ onde S denota o número de neurônios pertencentes à camada de saída. Neste vetor de sinais esperados, a saída esperada de um neurônio k é dado pelo elemento $d_k(n)$ e representa o valor esperado para o k -ésimo neurônio da camada de saída da rede neural, para a apresentação do sinal de entrada $\vec{x}(n)$.
- Erro: Representado pelo vetor $\vec{e}(n) = \{e_1(n), \dots, e_k(n), \dots, e_S(n)\}$ onde S denota o número de neurônios pertencentes à camada de saída. Neste vetor o elemento $e_k(n)$ representa o erro do k -ésimo neurônio da camada de saída da rede neural em relação ao valor esperado $d_k(n)$. Este sinal de erro pode ser calculado através da seguinte equação:

$$e_k(n) = d_k(n) - y_k(n) \quad (2.8)$$

Onde o termo n representa o instante de tempo no ciclo de aprendizado.

Com base nas definições apresentadas, tem-se um mecanismo capaz de realizar ajustes iterativos dos pesos sinápticos da rede neural através de sucessivas apresentações de exemplos de treinamento. Os ajustes realizados visam aproximar o sinal de saída obtido do sinal de saída esperado para o exemplo de treinamento escolhido no instante n . Neste algoritmo os ajustes podem ser realizados a cada exemplo de treinamento apresentado ou após a apresentação da lista completa de exemplos. O objetivo do algoritmo é de minimizar a função de custo $\xi(n)$, definida como o erro quadrático dos sinais de erro $\vec{e}_k(n)$:

$$\xi(n) = \frac{1}{2} e_k^2(n) \quad (2.9)$$

Com base na função de custo calculada, é então definido o ajuste a ser aplicado a cada um dos parâmetros da rede neural [Haykin-1999, p.53]:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \quad (2.10)$$

Onde η é uma constante positiva que determina a taxa de aprendizado do neurônio ou da rede neural como um todo.

Uma vez definido o ajuste Δw_{kj} a ser aplicado, o peso sináptico é então atualizado para o instante $n+1$ através da atribuição:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \quad (2.11)$$

Como o algoritmo aplica uma realimentação sobre o sistema, surge a necessidade de uma escolha adequada para o parâmetro de aprendizado η , a fim de garantir estabilidade e convergência no processo.

Este método de aprendizado é utilizado pelos Perceptron de Múltiplas Camadas, rede neural que será analisada posteriormente neste trabalho. Cabe ressaltar que outras redes neurais podem fazer uso deste mecanismo de aprendizado, porém apenas o PMC será abordado neste trabalho.

2.2.2.2 Aprendizado baseado em memória

Neste método de aprendizado, toda (ou quase toda) a experiência conhecida e correta é explícita e previamente armazenada em uma larga memória. De forma equivalente ao aprendizado por correção de erro, os exemplos armazenados são compostos por um par de sinais, identificando entradas conhecidas com respectivas saídas esperadas. Algebricamente, estes exemplos podem ser definidos de forma equivalente ao conjunto de exemplos de treinamento visto na equação (2.7):

Em modo de operação, quando o sistema necessitar classificar um vetor de entrada $\vec{x}_{teste}$ desconhecido, o algoritmo utiliza algum mecanismo de medida de similaridade de forma a obter um exemplo de treinamento semelhante ou próximo do sinal de entrada desconhecido e define a saída do sistema com base no vizinho (ou vizinhos) mais próximo do vetor de teste. Uma função de similaridade comumente adotada é a distância euclidiana entre os vetores da entrada desconhecida e entrada dos exemplos conhecidos. Cabe ressaltar que outros métodos de similaridade podem ser aplicados, de acordo com as necessidades de um problema sendo resolvido.

Considerando o conjunto de sinais de entrada dos exemplos conhecidos $\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N\}$, pode-se definir um vetor $\vec{x}_n \in \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N\}$ como sendo o vizinho mais próximo de um vetor $\vec{x}_{teste}$ se este for identificado como o vetor de maior semelhança em relação ao vetor de teste, segundo a função de similaridade adotada para o algoritmo.

Para uma função baseada na distância euclidiana, pode-se definir que:

$$\vec{x}_n \in \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N\} \quad (2.11)$$

É o mais próximo do vetor de entrada $\vec{x}_{teste}$ se:

$$\min_i d(\vec{x}_i, \vec{x}_{teste}) = d(\vec{x}_n, \vec{x}_{teste}) \quad (2.12)$$

Onde $d(\vec{x}_n, \vec{x}_{teste})$ é a distância Euclidiana entre os vetores $\vec{x}_n$ e $\vec{x}_{teste}$.

Independente da função de similaridade adotada, lembrando que a similaridade é utilizada para classificação de um sinal de entrada (onde as classes são os exemplos conhecidos), pode-se então definir o sinal $\vec{x}_N$ como sendo a “classe” do vetor de entrada $\vec{x}_{teste}$. Consequentemente a saída esperada do vetor $\vec{x}_{teste}$ será a saída esperada do vetor $\vec{x}_N$. Algumas variações deste algoritmo visam obter conjuntos de vetores vizinhos ao invés de identificar um único vetor [Haykin-1999, p.54], como por exemplo a regra dos “ k -vizinhos”.

Este método de aprendizado é utilizado para a rede de Hopfield, cabendo ressaltar que outras redes neurais também podem fazer uso deste algoritmo de aprendizado, porém não sendo abordados neste trabalho.

2.2.2.3 Aprendizado Hebbiano

A regra de aprendizado de Hebb pode ser definida como uma regra de dois passos [Haykin-1999, p.55]:

1. Se dois neurônios definindo uma sinapse forem ativados simultaneamente ou sincronizadamente, então o peso sináptico é seletivamente fortalecido.
2. Se dois neurônios definindo uma sinapse forem ativados assincronamente, então a sinapse é seletivamente enfraquecida.

Por não existir uma implementação fixa desta regra, no geral ela é descrita através de uma função genérica definida como $F : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$:

$$\Delta w_{k,j}(n) = F(y_k(n), x_j(n)) \quad (2.13)$$

Onde $w_{k,j}$ é o peso sináptico do neurônio k com sinais pré-sinápticos e pós-sinápticos denotados por x_j e y_k .

Como exemplo, uma função simplificada que atende aos dois passos do aprendizado de Hebb pode ser descrita através da equação:

$$\Delta w_{k,j}(n) = F(y_k(n), x_j(n)) = \eta y_k(n) x_j(n) \quad (2.14)$$

Onde η representa a taxa de aprendizado do neurônio ou da rede neural.

Com base nesta equação, se um modelo trabalhar com os sinais x_j e y_k na faixa de valores positivos, o valor $\Delta w_{k,j}$ sempre será positivo, fazendo com que o peso sináptico cresça indefinidamente e induzindo os sinais de saída da rede neural para valores de saturação. Para

solucionar este problema, formas mais elaboradas da regra podem ser definidas, como por exemplo a função:

$$\Delta w_{k,j}(n) = \eta (y_k(n) - \bar{y}) (x_j(n) - \bar{x}) \quad (2.15)$$

Onde $\bar{x}$ e $\bar{y}$ representam o valor médio temporal para os sinais pré-sinápticos e pós-sinápticos.

Este método de aprendizado é utilizado para a rede de Hopfield, cabendo ressaltar que outras redes neurais podem fazer uso deste aprendizado, porém não sendo objeto de estudo deste trabalho.

2.2.2.4 Aprendizado Competitivo

No aprendizado competitivo, os neurônios da camada de saída de uma rede neural competem entre si para serem ativados. Para esta regra de aprendizado ser aplicada, existem basicamente três requisitos necessários [Haykin-1999, p.58]:

1. A rede neural deve possuir um conjunto de neurônios idênticos, segundo suas modelagens matemáticas e equacionamentos, onde os pesos sinápticos dos neurônios artificiais são gerados aleatoriamente.
2. Deve existir um limite para a intensidade dos pesos sinápticos de cada um dos neurônios.
3. Deve existir um mecanismo de competição, que define um único neurônio de saída como neurônio vencedor.

Na forma mais simples, a regra de aprendizado competitiva utiliza uma rede neural com uma única camada de neurônios da saída, como apresentado na figura abaixo.

Figura 2.7 : Arquitetura neural baseada na regra de aprendizado competitivo.

Nesta arquitetura todos os neurônios da camada de entrada possuem conexão com todos os neurônios da camada de saída, podendo existir também conexões entre os neurônios de saída.

Comumente as conexões entre os módulos de saída são conexões inibitórias, desta forma auxiliando no procedimento de competição e definição de um único neurônio vencedor.

Segundo a regra de competição, para que um neurônio k seja o vencedor este deve possuir o maior sinal local v_k dentre os neurônios da camada de saída. Neste caso a saída y_k do vetor de saída é então definida com valor 1. Para os demais neurônios da rede (não vencedores), as saídas serão então definidas com valor 0:

$$y_k(n) = \begin{cases} 1 & \text{se } v_k(n) > v_j(n) \text{ para todo } j, j \neq k \\ 0 & \text{caso contrário} \end{cases} \quad (2.16)$$

Onde $v_k(n)$ é o sinal interno do neurônio vencedor k e $v_j(n)$ é o sinal interno de um dos neurônios perdedores.

Como forma de limitar os valores de saída das unidades em competição, é realizada uma normalização dos pesos sinápticos dos neurônios que compõe a rede:

$$\sum_{j=1}^m w_{k,j}(n) = 1 \quad (2.17)$$

Dada esta normalização, o aprendizado dos neurônios passa a ser um deslocamento de valores entre os pesos sinápticos, ou seja, se um peso sináptico for fortalecido, então uma ou mais outras ligações são enfraquecidas para manter a soma dos pesos com valor 1. Neste procedimento, caso um neurônio seja vencedor, serão fortalecidos os pesos sinápticos associados com as entradas ativas, enquanto que serão enfraquecidas as demais conexões. Os ajustes nos pesos sinápticos podem então ser definidos através da equação:

$$\Delta w_{k,j}(n) = \begin{cases} \eta (x_j - w_{k,j}) & \text{se neurônio } k \text{ vencer} \\ 0 & \text{se neurônio } k \text{ perder} \end{cases} \quad (2.18)$$

Onde η representa a taxa de aprendizado do neurônio ou da rede neural.

Em resumo, com base neste mecanismo, sempre que um neurônio for vencedor, então seus pesos sinápticos irão se aproximar do vetor de entrada utilizado.

Quanto à aplicação, esta regra de aprendizado é utilizada pela rede de Kohonen, cabendo ressaltar que outras redes podem fazer uso deste procedimento, porém não sendo abordadas neste trabalho.

2.2.2.5 Aprendizado de Boltzmann

Como o próprio nome sugere, esta regra de aprendizado é específica para a rede neural denominada Máquina de Boltzmann. Uma máquina de Boltzmann é uma rede neural

composta por neurônios estocásticos, cujas saídas são binárias e calculadas com base em uma probabilidade de serem “ativados” ou “desativados”. Neste modelo os neurônios são estruturados utilizando uma arquitetura realimentada, porém sem auto-realimentação (a saída de um neurônio não realimenta sua própria entrada). Para determinadas aplicações com a Máquina de Boltzmann (memória associativa, etc.) os neurônios que compõe a rede são divididos em dois grupos distintos, um grupo de neurônios visíveis, que possuem interface com o ambiente, e um grupo de neurônios escondidos, que operam livre de interferências de estímulos externos.

Figura 2.8 : Máquina de Boltzmann.

Uma vez que os neurônios visíveis permitem uma interface da rede com o ambiente externo, é então possível configurar a rede para operar em dois modos distintos:

1. Operação associada: modo de operação onde os estados dos neurônios visíveis são definidos pelo ambiente externo.
2. Execução livre: todos os neurônios da rede operam livremente sem influência do ambiente externo.

Dada a utilização de neurônios estocásticos na composição da máquina de Boltzmann, a saída desta rede neural não é uma saída fixa (também denominado estado da rede) e sim é uma saída oscilatória cujos valores possuem uma distribuição de probabilidade. Desta forma, o estado da rede neural pode então ser representado por uma variável aleatória X . Dada a característica probabilística das transições entre estados distintos da rede neural, pode-se então representar as saídas da rede através de uma cadeia de Markov, sendo a probabilidade de transição dependente do parâmetro de controle T do neurônio estocástico que compõe a rede (vide equação 2.4).

Como exemplo, supondo uma rede neural com saída $\overline{y(n)} = \{y_1(n), y_2(n)\}$ tem-se que os estados da rede podem possuir os seguintes valores possíveis: $\{y_1(n)=0, y_2(n)=0\}$, $\{y_1(n)=1, y_2(n)=0\}$, $\{y_1(n)=0, y_2(n)=1\}$ e $\{y_1(n)=1, y_2(n)=1\}$. Com base nestes estados possíveis, os estados da rede poderão ser representados através de uma cadeia de Markov como exibida na figura abaixo.

Figura 2.9 : Cadeia de Markov para representação dos estados da máquina de Boltzmann com 2 bits.

A idéia base por trás do algoritmo de aprendizado de Boltzmann é de identificar qual a configuração de pesos sinápticos que implicará em uma saída desejada como a saída mais provável da rede neural, um conceito denominado verossimilhança. Neste procedimento é então utilizada uma “função de verossimilhança”, que deve então ser maximizada dado um estado desejado.

Com a existência de duas classes distintas de neurônios, visíveis e escondidos, os estados da rede podem ser classificados através de duas variáveis aleatórias distintas, uma variável aleatória X_α para representar os estados dos neurônios visíveis e uma variável aleatória X_β para representação dos estados dos neurônios escondidos. Desta forma, segundo [Haykin-1999, p.567] os ajustes nos pesos sinápticos são dadas pela equação:

$$\Delta w_{ji} = \eta (\rho_{ji}^+ - \rho_{ji}^-) \quad (2.18)$$

Onde η é a taxa de aprendizado, ρ_{ji}^+ é um termo identificando a correlação entre os estados i e j da rede neural quando esta estiver operando em modo associado e ρ_{ji}^- é um termo identificando a correlação entre os estados i e j da rede neural quando esta estiver operando em modo livre.

Os termos ρ_{ji}^+ e ρ_{ji}^- podem ser detalhados como:

$$\rho_{ji}^+ = \sum_{\vec{x}_\alpha \in \tau} \left(\sum_{\vec{x}_\beta} P(\vec{X}_\beta = \vec{x}_\beta \mid \vec{X}_\alpha = \vec{x}_\alpha) x_j x_i \right) \quad (2.19)$$

Com $P(\vec{X}_\beta = \vec{x}_\beta \mid \vec{X}_\alpha = \vec{x}_\alpha)$ representando a probabilidade dos neurônios escondidos estarem em um estado $\vec{x}_\beta$ dado que foram observados os neurônios visíveis no estado $\vec{x}_\alpha$, e

$$\rho_{ji}^- = \sum_{x_a \in \tau} \left(\sum_{\vec{x}} P(\vec{X} = \vec{x}) x_j x_i \right) \quad (2.20)$$

Com $P(\vec{X} = \vec{x})$ representando a probabilidade de termos os neurônios da rede neural em um estado $\vec{x}$ dada nenhuma constatação dos estados visíveis da rede.

Nestas equações, a taxa de aprendizado η é dada por:

$$\eta = \frac{\epsilon}{T} \quad (2.21)$$

Onde ϵ é uma constante (sem valor pré-definido) e T representa a variável de controle utilizada pelos neurônios estocásticos da rede. A definição desta taxa de aprendizado tem como objetivo fazer com que o valor de η se aproxime de zero com o passar do tempo. O valor utilizado para a variável ϵ e para a taxa de variação da variável T depende da modelagem da rede para cada problema a ser resolvido.

O algoritmo de aprendizado se resume a um procedimento cíclico, onde a rede neural opera inicialmente em modo associado para cálculo do termo ρ_{ji}^+ , seguido pela operação em modo livre para cálculo do termo ρ_{ji}^- e posteriormente são realizados os cálculos dos ajustes sinápticos da rede neural.

Esta regra de aprendizado é comumente utilizada pela Máquina de Boltzmann para problemas de classificação. Para algumas classes distintas de problemas, como por exemplo os problemas de otimização, outros mecanismos de identificação dos pesos sinápticos devem ser utilizados. Para o caso específico da otimização, não existe um procedimento genérico de mapeamento dos problemas nos parâmetros da rede, como se pode ver em [Aarts&Korst-1989], onde diversos problemas de otimização são descritos e analisados de forma específica.

2.2.3 Redes Neurais Artificiais

As redes neurais representam a abordagem conexionista da IC. Nesta área têm-se modelos baseados na interligação de neurônios artificiais em forma de rede, de forma a simular as estruturas dos neurônios biológicos no sistema nervoso dos seres vivos. A utilização de redes neurais tem como objetivo obter estruturas com poder de processamento paralelo, tolerância a falhas, escalonabilidade, entre outras características intrínsecas das redes biológicas. Com base nestas estruturas, tem-se na atualidade uma grande diversidade de classes de utilização que podem ser resolvidas com as RNAs. Dentre as utilizações principais, pode-se citar:

- Classificação
- Reconhecimento de padrões

- Memórias Associativas
- Otimização
- Aproximação de Funções
- Agrupamento de Dados
- Recuperação de Padrões

Além da diversidade de aplicações, as redes neurais também possuem grande diversidade de propriedades e classificações. Como exemplo, serão analisadas as classificações das redes neurais quanto aos seguintes aspectos:

- Arquitetura (Tipo de interligação neural)
- Tipo de apoio no processo de aprendizado
- Tipo de apresentação de exemplos de aprendizado
- Dinâmica da arquitetura
- Tipo de memória associativa
- Tipo de modelo neural

Arquitetura (Tipo de interligação neural)

Quanto à arquitetura de interligação de neurônios na rede, estas podem ser subdivididas em duas classes distintas:

- Redes com realimentação de sinal: a propagação de sinal possui um caminho fechado na rede.
- Redes com propagação direta (forward propagation) de sinal: a propagação de sinais da rede não possui realimentação.

Figura 2.10 : Estruturas das Redes Neurais.

Tipo de apoio no processo de aprendizado

Referente à existência de apoio externo no processo de aprendizado, as redes neurais podem receber as seguintes classificações:

- Aprendizado supervisionado: o aprendizado da rede é realizado através da apresentação de exemplos de treinamento fornecidos por um agente externo.

- Aprendizado não supervisionado: aprendizado independente de realimentação externa.
- Aprendizado por reforço: agente externo assinala acertos e erros para a rede. A rede recebe uma medida de recompensa ou punição de acordo com sua saída. As alterações internas na rede são realizadas de forma a maximizar a recompensa para sua saída.

Tipo de apresentação de exemplos de aprendizado

Para os algoritmos de aprendizado, no geral estes são baseados em métodos numéricos (gradiente descendente, etc.), que são executados de forma cíclica, ajustando gradualmente as configurações da rede neural. Cada ciclo do algoritmo tem como objetivo calcular os novos valores dos parâmetros livres (pesos sinápticos e *bias*), sendo que os cálculos podem ser realizados para cada novo exemplo de treinamento apresentado à rede ou levando em consideração uma média baseada em um conjunto inteiro de exemplos. Com base nesta característica as redes podem receber novas classificações:

- Modo padrão de aprendizado: Correção dos pesos da rede é realizada a cada novo exemplo.
- Modo batch de aprendizado: Correção dos pesos da rede é realizada após análise de todos os exemplos de um conjunto de treinamento.

Dinâmica da arquitetura

Além dos pesos sinápticos e *bias*, os algoritmos de aprendizado também podem alterar as estruturas das redes, desta forma permitindo classificação quanto à dinâmica da arquitetura da rede. As redes desta forma podem receber as seguintes classificações:

- Estáticas: topologia com número de neurônios e ligações bem definidas e estáticas.
- Dinâmicas: redes neurais com topologia dinâmica, onde o número de neurônios e as conexões sinápticas podem ser alteradas de forma a obter comportamentos diferenciados.

Tipo de memória associativa

Levando em consideração as classes de utilização, para as memórias associativas as redes neurais podem ser classificadas como:

- Redes hetero-associativas: redes que são capazes de mapear elementos pertencentes a um conjunto de entradas possíveis em um elemento pertencente a um conjunto de saídas possíveis.
- Redes auto-associativas: as redes desta classe são um caso especial de redes hetero-associativas, onde a rede mapeia na saída o próprio exemplo de entrada,

mesmo este sendo uma versão ruidosa ou incompleta do sinal a ser recuperado. Em essência, este tipo de memória é utilizado para a recuperação de sinais ruidosos, ou seja, é utilizado para recuperar uma memória $m1$ a partir de um sinal ruidoso $m1'$.

Tipo de modelo artificial

Esta configuração define se a rede utiliza neurônios artificiais determinísticos ou estocásticos, conforme vistos anteriormente:

- Rede determinística: utiliza o modelo de neurônio artificial determinístico.
- Rede estocástica: utiliza o modelo de neurônio artificial estocástico.

Com base nas propriedades descritas acima e levando em consideração o tipo de aprendizado que cada rede possui, um grupo restrito de redes neurais pode ser selecionado para análise. As redes selecionadas encontram-se apresentadas na Tabela 2.1.

	Arquitetura		Algoritmo aprendizado	Tipo de aprendizado			Modo de aprendizado		Dinâmica da arquitetura		Modelo neural	
	Propagação	Realimentação		Supervisionado	Não supervisionado	Reforço	Padrão	Batch	Estática	Dinâmica	Determinístico	Estocástico
PMC	•		Correção de Erro	•	•	•	•	•	•	•	•	
Kohonen	•		Competitivo	•	•	•	•		•	•	•	
Hopfield		•	Memória Hebbiano	•	•	•	•	•	•	•	•	
Boltzmann		•	Boltzmann	•	•	•	•	•	•	•		•

Tabela 2.1 : Propriedades das redes neurais.

Cabe ressaltar que outras redes neurais foram avaliadas previamente a este texto, tendo sido desconsideradas devido ao alto grau de semelhança que tinham com uma das quatro redes descritas na Tabela 2.1. As redes desconsideradas para a análise detalhada neste texto foram:

- Redes Radial-Basis Function: Rede neural com propagação de sinal e arquitetura semelhantes ao PMC [Haykin-1999, p.256].
- Rede de Willshaw-von der Malsburg: Rede neural com propagação de sinal e arquitetura semelhantes à rede de Kohonen [Haykin-1999, p.445].
- Rede Brain-State-in-a-Box: Rede neural realimentada com arquitetura semelhante à rede de Hopfield, porém com limitações nas amplitudes dos sinais [Haykin-1999, p.703].
- Rede Sigmoid Belief: Rede neural estocástica, baseado nos mesmos princípios da Máquina de Boltzmann, porém com arquitetura desprovida de realimentação. O objetivo desta rede era de realizar um aprendizado sem a necessidade da fase negativa

de operação (operação desacoplada do ambiente) da máquina de Boltzmann [Haykin-1999, p.569].

- Helmholtz Machine: Rede neural estocástica, baseado nos mesmos princípios da Rede Sigmoid Belief, porém acrescentando simplificações adicionais [Haykin-1999, p.574].

Nos tópicos seguintes serão analisados em maiores detalhes as redes neurais descritas na Tabela 2.1.

2.2.4 Perceptrons de Múltiplas Camadas

Segundo esta arquitetura, tem-se uma rede neural com propagação de sinal sem realimentação. A escolha da função de transferência depende da aplicação a ser realizada, porém comumente é utilizada a função sigmóide, tanto por esta possuir derivada, cálculo necessário para aplicação do algoritmo de aprendizado Backpropagation, como também pela possibilidade de aproximação de funções de maior complexidade como somatória de funções sigmóide. Na figura abaixo é apresentada a arquitetura de conexão neural para uma rede do tipo Perceptron de Múltiplas Camadas (PMC).

Figura 2.11 : Perceptron de Múltiplas Camadas (PMC).

Como evidenciado na figura acima, a arquitetura do PMC é subdividida em três classes de camadas: uma camada com os sinais de entrada da rede, uma camada de neurônios de saída, e uma ou mais camadas de neurônios escondidos. Nesta arquitetura, a camada de entrada não apresenta nenhum processamento neural sendo meramente uma representação dos sinais de entrada da rede neural.

Outra característica desta rede neural está no mecanismo de aprendizado, que utiliza um algoritmo de aprendizado por correção de erro. Neste procedimento o aprendizado é realizado de forma supervisionada. O algoritmo de aprendizado utilizado por esta rede neural é o denominado algoritmo de Backpropagation.

2.2.4.1 Backpropagation

Em uma rede neural do tipo Perceptron de Múltiplas Camadas (PMC), os sinais de saída podem ser considerados como uma função dos pesos sinápticos e *bias* dos neurônios. Desta forma, o erro calculado para um dado exemplo de treinamento também será função destes parâmetros internos da rede. Com base neste princípio, a idéia principal do algoritmo Backpropagation consiste na aplicação de um método numérico para minimização da função de erro da rede neural. O método de minimização utilizado neste algoritmo é o denominado método do gradiente descendente da função de erro.

Em resumo, o algoritmo de aprendizado da rede PMC pode ser descrito através de duas etapas de processamento distintas. A primeira fase é referente à propagação do sinal do começo da rede para seu fim, enquanto que a segunda etapa do algoritmo visa gerar um cálculo do erro obtido na saída da rede neural e propagar o erro da camada de saída até a camada de neurônios de entrada ao mesmo tempo em que são realizados os ajustes nos pesos sinápticos.

Na primeira etapa, a propagação dos sinais é realizada respeitando a ordem das camadas de neurônios que compõe a rede. As saídas dos neurônios de uma determinada camada são calculadas em função das saídas da camada anterior, além dos pesos sinápticos e *bias* de cada um dos neurônios individuais da rede. Como não existem conexões entre neurônios de uma mesma camada, então não existe uma ordem específica para a escolha dos neurônios a serem processados pertencentes à mesma camada neural. O algoritmo de propagação processa cada uma das camadas neurais de forma diferenciada, desta forma existindo tratamentos específicos para as camadas de entrada, escondidas e de saída. Nas descrições a serem apresentadas, uma vez escolhida uma camada de processamento, os índices i representarão os elementos pertencentes a uma camada anterior à camada vigente, os índices j representarão os elementos pertencentes à camada posterior e o índice k representará índice do neurônio na camada vigente.

Na primeira etapa, de propagação de sinal, se o cálculo da saída está sendo realizado para um neurônio da camada de entrada, então as saídas são apenas atualizadas como:

$$y_j(n) = x_i(n) \quad (2.22)$$

Caso um neurônio em análise seja um neurônio pertencente às demais camadas neurais, então o sinal cálculo do sinal local do neurônio deve ser realizado através da equação (2.5), referente ao modelo determinístico de neurônio artificial. Por simplicidade, pode-se reescrever a equação levando em consideração a representação do *bias* através do índice $i = 0$:

$$v_k(n) = \sum_{i=0}^m w_{ki}(n) \cdot y_i(n) \quad (2.23)$$

Lembrando que o índice k representa o k -ésimo neurônio de uma camada cujas saídas estão sendo calculadas, e o índice i representa uma varredura de todos os neurônios da camada anterior. O parâmetro m representa o número total de entradas do neurônio k , não incluindo

a *bias*, que é contabilizado através do índice $i=0$, $w_{k,i}$ representa o peso sináptico conectando o neurônio i da camada anterior ao neurônio k da camada em processamento.

O sinal de saída do neurônio será dado pela equação (2.2) sendo a função de transferência mais comumente utilizada a função sigmóide, porém podendo ser adotada qualquer outra função que possua derivada primeira. Desta forma pode-se reescrever a função de forma genérica como sendo:

$$y_k(n) = \varphi_k(v_k(n)) \quad (2.24)$$

Onde o parâmetro φ_k representa a função de transferência do neurônio em processamento.

Caso os neurônios em análise sejam os neurônios da camada de saída, então os valores de saída destes neurônios serão armazenados como sendo as saídas da rede neural como um todo:

$$o_j(n) = y_k(n) \quad (2.25)$$

Uma vez contabilizados os sinais de saída $o_j(n)$, são calculados os erros de saída em relação aos sinais esperados $d_j(n)$.

$$e_j(n) = d_j(n) - o_j(n) \quad (2.26)$$

Com os cálculos dos erros realizados, concluímos a primeira fase de processamento e inicia-se a varredura da rede neural, iniciando da última camada neural de saída, para a primeira camada de entrada. Nesta varredura são calculadas as alterações sinápticas dos neurônios através da equação:

$$\Delta w_{k,i}(n) = \eta \delta_k(n) y_i(n) \quad (2.27)$$

Para os neurônios da camada de saída o gradiente local $\delta_k(n)$ será dada por:

$$\delta_k(n) = e_j(n) \varphi'_k(v_k(n)) \quad (2.28)$$

Para os neurônios pertencentes às camadas escondidas o gradiente local $\delta_k(n)$ será dada por:

$$\delta_k(n) = \varphi'_k(v_k(n)) \sum_j \delta_j(n) w_{j,k}(n) \quad (2.29)$$

Considerando uma função de transferência do tipo sigmóide tem-se que:

$$\varphi_k(v_k(n)) = \frac{1}{1 + \exp(-a v_k(n))} \quad (2.30)$$

$$\phi'_k(v_k(n)) = \frac{a \exp(-a v_k(n))}{[1 + \exp(-a v_k(n))]^2} \quad (2.31)$$

Onde $a > 0$ é uma constante que permite um ajuste na forma da função de transferência (aumento no valor de a faz com que a onda se aproxime de uma função degrau) e $v_k(n)$ é o sinal local do neurônio k da camada em processamento.

Com base nesta função, para neurônios da camada de saída pode-se reescrever o gradiente local como sendo:

$$\delta_k(n) = a [d_j(n) - o_j(n)] o_j(n) [1 - o_j(n)] \quad (2.32)$$

Para os neurônios das demais camadas neurais pode-se reescrever o gradiente local como:

$$\delta_k(n) = a y_k(n) [1 - y_k(n)] \sum_j \delta_j(n) w_{jk}(n) \quad (2.33)$$

2.2.5 Rede Neural de Kohonen

A rede neural de Kohonen pertence a uma classe de redes denominada de mapas auto-organizáveis. Segundo esta arquitetura, a rede neural é estruturada em camadas de neurônios onde a realimentação de sinal ocorre apenas entre os neurônios da camada de saída. Tal realimentação é comumente utilizada devido à regra de aprendizado competitiva, onde os neurônios de saída aplicam sinais inibitórios entre si. Outra característica desta arquitetura neural está na arquitetura física dos neurônios da rede, comumente baseadas em saídas estruturadas em mapas de duas dimensões ou vetores unidimensionais. Saídas com dimensões maiores também podem ser utilizadas.

Figura 2.12 : Modelo rede neural de Kohonen.

O objetivo principal dos mapas auto-organizáveis é de transformar padrões dos sinais de entrada, de dimensões arbitrárias, em um mapa de uma ou duas dimensões, embora possam ser utilizados mapas com dimensões superiores. O mapeamento é realizado topograficamente à medida que as informações vão sendo apresentadas à rede.

Nesta rede neural, o conceito de topografia indica que as distâncias entre neurônios podem ser medidas com base nos índices das colunas, linhas, etc. Neste contexto, uma vizinhança topográfica pode então ser definida com base em uma função de vizinhança, que comumente analisa a distância euclidiana entre os neurônios do mapa de saída, porém, podendo ser utilizados outros mecanismos de identificação de vizinhança.

Figura 2.13 : Distância entre neurônios para mapa (a) retangular e (b) hexagonal.

O algoritmo responsável pela formação dos mapas auto-organizáveis é baseado no algoritmo de aprendizado competitivo descrito anteriormente.

2.2.5.1 Mapa auto-organizável

O algoritmo clássico utilizado nesta rede neural pode ser resumido através da execução dos três procedimentos seguintes:

1. **Competição.** Para cada padrão de entrada, os neurônios da rede computam seus valores de saída. O neurônio com a maior saída é declarado o neurônio vencedor.
2. **Cooperação.** O neurônio vencedor determina a posição topográfica relacionado com o padrão de entrada aplicado à rede. Com base neste padrão é então realizado um ajuste no conjunto de neurônios vizinhos do neurônio vencedor.
3. **Adaptação sináptica.** Permite que a rede aprenda os padrões de entrada a partir dos ajustes nos pesos sinápticos da rede. Os ajustes são aplicados ao neurônio vencedor de forma a fortalecer o padrão que este conseguiu identificar. Os neurônios vizinhos também são influenciados pelo neurônio vencedor, possuindo ajustes em seus pesos sinápticos de forma menos acentuada do que o neurônio vencedor.

Processo competitivo

Considerando um vetor de entrada de m dimensões, podem-se representar cada um dos padrões como um vetor $\vec{x}$ dado por:

$$\vec{x} = [x_1, x_2, \dots, x_m]^T \quad (2.34)$$

Segundo a arquitetura da rede de Kohonen, cada um dos neurônios da camada de saída possui conexão com todos os neurônios da camada de entrada, conseqüentemente tendo para cada neurônio de saída um vetor de pesos sinápticos de mesma dimensão m que os padrões de entrada sendo classificados. Desta forma podem-se representar os vetores de pesos sinápticos como sendo:

$$\vec{w}_k = [w_{k,1}, w_{k,2}, \dots, w_{k,m}]^T, k = 1, 2, \dots, l \quad (2.35)$$

Onde l é o número total de neurônios na camada de saída da rede.

Para o procedimento de competição, é utilizado o critério da menor distância euclidiana entre os vetores $\vec{x}$ e $\vec{w}_k$. O neurônio selecionado segundo esta definição é então nomeado o neurônio vencedor $i(\vec{x})$:

$$i(\vec{x}) = \arg \min_k \|\vec{x} - \vec{w}_k\|, k = 1, 2, \dots, l \quad (2.35)$$

Processo Cooperativo

Uma vez selecionado o neurônio vencedor $i(\vec{x})$, pode-se então definir uma vizinhança topográfica de neurônios cooperativos, onde a cooperação na rede neural se manifesta através de uma influência do neurônio vencedor no ajuste dos pesos sinápticos dos neurônios vizinhos. A vizinhança definida pelo neurônio vencedor é então definida como $h_{k,i(\vec{x})}$, sendo a distância lateral entre o neurônio vencedor $i(\vec{x})$ e o neurônio excitado k representada através do parâmetro $d_{k,i}$. Este parâmetro representa a distância lateral entre neurônios no mapa de saída. Em outras palavras, a vizinhança topológica $h_{k,i(\vec{x})}$ é definida como uma função da distância, e comumente segue dois requisitos distintos:

1. A vizinhança topográfica $h_{k,i(\vec{x})}$ é simétrica em relação ao ponto $d_{k,i} = 0$, ou seja, esta vizinhança possui valor de representação máxima no neurônio vencedor $i(\vec{x})$.

2. A amplitude da vizinhança topológica $h_{k,i}(\vec{x})$ decai monotonicamente com o aumento da distância $d_{k,i}$, tendendo a zero para $d_{k,i} \rightarrow \infty$. Esta é uma condição necessária para convergência.

Uma escolha típica para $h_{k,i}$ que satisfaz estes requerimentos é a função gaussiana [Haykin-1999 p.449]:

$$h_{k,i}(\vec{x}) = \exp\left(-\frac{d_{k,i}^2}{2\sigma^2}\right) \quad (2.36)$$

Onde o parâmetro σ representa a amplitude efetiva da vizinhança topológica ou o grau em que os neurônios excitados na vizinhança do neurônio vencedor participam do processo de aprendizado. Este parâmetro não possui um valor pré-definido, devendo ser escolhido de acordo com o problema sendo resolvido.

Para o caso de uma saída com uma única dimensão, a distância $d_{k,i}$ referenciada pela equação da vizinhança pode ser dada pelo valor inteiro.

$$d_{k,i} = |k - i| \quad (2.37)$$

Para uma saída de duas dimensões a distância pode ser dada por:

$$d_{k,i}^2 = \|\vec{r}_k - \vec{r}_i\|^2 \quad (2.38)$$

Onde r_k é um vetor discreto que define a posição do neurônio excitado k e r_i define a posição discreta do neurônio vencedor $i(\vec{x})$.

Outra característica que pode ser aplicado ao algoritmo é a redução do tamanho da vizinhança topológica com o tempo, característica obtida através da alteração do parâmetro σ na função de vizinhança topológica $h_{k,i}(x)$. Uma redução comumente adotada para a função de redução do valor de σ é a redução exponencial com o tempo n [Haykin-1999, p.450].

$$\sigma(n) = \sigma_0 \cdot \exp\left(-\frac{n}{\tau_1}\right), n = 0, 1, 2, \dots \quad (2.39)$$

Onde σ_0 é o valor inicial do parâmetro σ no algoritmo, e τ_1 é uma constante de tempo, que não possui valor preciso, devendo ser escolhido adequadamente para o problema ou cenário que faz uso desta rede.

Correspondentemente, a vizinhança topológica assume uma variação no tempo, resultando em uma nova equação de vizinhança:

$$h_{k,i(x)}(n) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(n)}\right), n = 0, 1, 2, \dots \quad (2.40)$$

Processo Adaptativo

O processo adaptativo tem como foco realizar os ajustes sinápticos dos neurônios pertencentes à vizinhança $h_{k,i(x)}$ definida pelo neurônio vencedor $i(\vec{x})$. Segundo [Haykin-1999, p.451] o ajuste nos pesos sinápticos pode ser realizado aplicando a seguinte função para todos os neurônios da camada de saída:

$$w_k(n+1) = w_k(n) + \eta(n) h_{k,i(x)}(n) (\vec{x} - \vec{w}_k(n)) \quad (2.41)$$

Para uma melhor aproximação do algoritmo, a taxa de aprendizado também pode ser definida como função do tempo. Segundo [Haykin-1999, p.453] a variação desta taxa pode ser dada pela equação:

$$\eta(n) = \eta_0 \cdot \exp\left(\frac{-n}{\tau_2}\right), n = 0, 1, \dots \quad (2.42)$$

Onde τ_2 , de forma equivalente à τ_1 , é uma constante de tempo que permite ajustar o algoritmo de acordo com um determinado problema ou cenário em resolução.

2.2.5.2 Otimização

Com base na arquitetura padrão da rede de Kohonen é possível resolver alguns cenários específicos de problemas de otimização. Neste trabalho será analisado o caso específico do Problema do Caixeiro Viajante, uma vez que a arquitetura e os pesos sinápticos da rede devem ser definidos através de um procedimento de inicialização específico para cada tipo de problema desta categoria sendo solucionado.

2.2.5.2.1 Problema do Caixeiro Viajante

Dada a posição de um número de cidades distintas, o Problema do Caixeiro Viajante se resume em achar um menor percurso passando por cada cidade uma única vez e retornar ao ponto de partida inicial. Para a resolução deste problema com c cidades, segundo [Fausett-1993, p.182] a rede de Kohonen utiliza uma arquitetura neural com dois neurônios na camada de entrada e uma quantidade de neurônios $m_{Kohonen}$ superior ao número de cidades c para a camada de saída. Uma vez que todos os neurônios da camada de saída são conectados aos

neurônios da camada de entrada, tem-se que cada saída possui duas sinapses distintas, que são interpretadas como coordenadas (x, y) de pontos de um percurso sobre um mapa 2D. Para o problema do caixeiro viajante, os neurônios de saída são então definidos como um vetor unidimensional, onde cada neurônio representa um ponto do percurso seguindo pelo caixeiro viajante no mapa do problema. As vizinhanças são então definidas com base na posição de cada neurônio no vetor, sendo o último neurônio da camada de saída considerado como vizinho do primeiro (e vice-versa).

Figura 2.14 : - Arquitetura de rede de Kohonen utilizada para resolução do problema do caixeiro viajante.

Com base nesta interpretação, uma inicialização dos pesos sinápticos com valores aleatórios resulta em um percurso de saída ruidoso, como exemplificado na figura abaixo.

Figura 2.15 : - Interpretação de um percurso em rede de Kohonen com pesos sinápticos inicializados aleatoriamente [Fausett-1993].

Para a resolução do problema do caixeiro viajante utilizando esta rede, o algoritmo de aprendizado (descrito anteriormente) é executado utilizando como sinais de entrada as

coordenadas das cidades do mapa. Após sucessivas iterações do algoritmo, os pesos sinápticos das unidades neurais se aproximam de valores que podem ser interpretados como uma aproximação de um percurso do caixeiro viajante. Um exemplo de saída deste algoritmo pode ser visto na figura abaixo.

Figura 2.16 : - Percurso resultante da execução do algoritmo da rede de Kohonen (à esquerda) e a posição das cidades no mapa 2D (à direita) [Fausett-1993].

Dado o uso de um número de unidades neurais superior ao número de cidades do problema, o resultado final gera uma representação dos pontos do percurso, sendo que estes pontos não necessariamente coincidem com as cidades do mapa. Grosseiramente, os pontos representam uma aproximação de um percurso possível, caso não sejam consideradas restrições no problema, como por exemplo cidades não interconectadas (cidades não conectadas não conseguem ser representadas na solução utilizando rede de Kohonen). Dada a natureza das soluções obtidas, que são aproximações de percursos possíveis, as soluções apresentadas podem dar margem a diferentes interpretações dos resultados. Como exemplo, na figura 2.16 é possível interpretar dois percursos distintos a partir dos resultados finais obtidos pela rede neural, um percurso definido pelas visitas ordenadas A-D-E-F-G-H-I-J-B-C, e um percurso distinto definido pelas visitas ordenadas A-D-E-F-G-H-I-J-C-B [Fausett-1993, p.185].

2.2.6 Rede Neural de Hopfield

A rede de Hopfield é uma rede neural com arquitetura realimentada, onde o sinal de saída de cada unidade realimenta a entrada dos demais neurônios da rede. Segundo o modelo padrão, as conexões sinápticas são simétricas, i.e. o peso sináptico do neurônio A para o neurônio B é o mesmo do neurônio B para o neurônio A, assim como não é utilizada a auto-realimentação. Para o processamento da rede e seus algoritmos, é utilizada uma base de tempo discreta, sendo esta característica explicitamente mapeada através de atrasos unitários de tempo, como apresentado na Figura 2.17.

Figura 2.17 : Rede de Hopfield.

Uma das questões envolvendo sistemas com realimentação é referente à possibilidade de oscilação dos sinais de saída, fazendo com que estes não consigam convergir para um ponto de equilíbrio desejado. Como forma de impedir que a rede entre em um estado oscilatório do sinal de saída, algumas propriedades devem ser respeitadas, conforme é possível avaliar em [Bruck-1990] no qual são apresentados casos específicos de redes Hopfield com convergência dos sinais de saída. Para o uso da arquitetura padrão, com pesos sinápticos simétricos e sem auto realimentação, a convergência dos sinais de saída podem comprovados, conforme visto em [Haykin-1999, p.677].

2.2.6.1 Memória associativa

Uma das aplicações comuns para a rede Hopfield é a de memória associativa. Nesta classe de aplicações, a rede é utilizada como uma forma de armazenamento de padrões conhecidos previamente. Uma vez armazenados os padrões desejados para o problema, o objetivo da rede é de recuperação de um item em memória, através da apresentação de um padrão incompleto ou ruidoso do sinal na entrada da rede. Neste contexto, a rede pode ser considerada um sistema de correção de erros.

A essência da memória associativa é de mapear uma memória fundamental $\vec{\xi}_\mu$ em um ponto fixo $\vec{x}_\mu$ do sistema dinâmico. Matematicamente este mapeamento pode ser descrito como:

$$\vec{\xi}_\mu \Leftrightarrow \vec{x}_\mu \quad (2.47)$$

Uma operação de mapeamento de um exemplo no sistema é denominada codificação. Um procedimento de recuperação de um padrão armazenado é denominado decodificação. Os pontos atratores no espaço de estados da rede são as “memórias fundamentais” ou “protótipo de estados” da rede neural. Nesta rede, um dado apresentado com informações parciais sobre

um padrão sendo recuperado representará um ponto de início do processamento no espaço de estados. Dado este ponto inicial, os sinais da rede irão convergir para os pontos atratores, que representam uma memória armazenada.

Nesta arquitetura, os modelos neurais da rede possuem saídas binárias. Uma entrada $\vec{x}$ pode então ser representada pelo vetor :

$$\vec{x} = [x_1, x_2, \dots, x_m]^T \quad (2.48)$$

Onde m representa a dimensão dos vetores de estado.

A saída de um neurônio k da rede é então calculada como:

$$y_k(n) = \begin{cases} +1 & \text{se } v_k > 0 \\ -1 & \text{se } v_k < 0 \\ 0 & \text{se } v_k = 0 \end{cases} \quad (2.49)$$

Utilizando as características citadas, o algoritmo de treinamento pode ser definido através de duas fases distintas de processamento:

Processo de armazenagem

Este procedimento visa executar um algoritmo de aprendizado para armazenamento de um conjunto de M “memórias fundamentais”, denotado por $\{\xi_\mu | \mu = 1, 2, \dots, M\}$. O i -ésimo sinal da memória fundamental ξ_μ pode ser representado através do parâmetro $\xi_{\mu,i}$.

Com base no aprendizado Hebbiano, a armazenagem das memórias fundamentais pode ser realizada através dos pesos sinápticos definidos pela seguinte equação:

$$w_{k,j} = \frac{1}{N} \sum_{\mu=1}^M \xi_{\mu,k} \cdot \xi_{\mu,j} \quad (2.50)$$

Onde $1/N$ é utilizado como constante de proporcionalidade para simplificar a descrição matemática na fase de recuperação.

Em essência, cada um dos pesos sinápticos dos neurônios da rede serão definidos com base nas M memórias fundamentais sendo armazenadas.

Os pesos sinápticos dos neurônios podem também ser descritos em forma matricial:

$$W = \frac{1}{N} \sum_{\mu=1}^M \vec{\xi}_\mu \cdot \vec{\xi}_\mu^T - MI \quad (2.51)$$

Onde $\vec{\xi}_\mu \cdot \vec{\xi}_\mu^T$ denota o produto do vetor $\vec{\xi}_\mu$ com ele mesmo e I representa a matriz identidade, utilizada como forma de representação da ausência de auto-realimentação, atribuindo o valor 0 nos elementos da matriz $w_{k,k}$.

Processo de recuperação

Uma vez inicializada a rede, é possível então executar um procedimento de recuperação das “memórias fundamentais”. Neste procedimento, os sinais de cada um dos neurônios são inicializados com os valores dos padrões ruidosos de interesse. A recuperação então prossegue com cada neurônio k da rede sendo aleatoriamente selecionado para cálculo de seu sinal de saída $y_k(n)$, conforme definido pela equação (2.49). A atualização dos estados de uma iteração para outra é então determinística, mas a seleção do neurônio a ser avaliado é aleatória. O procedimento de atualização é então adotado até que nenhuma alteração dos sinais da rede seja identificada.

2.2.6.2 Otimização

Uma modificação da rede neural discreta de Hopfield pode ser utilizada para problemas de otimização combinatorial com restrições. Uma vez que a arquitetura e os pesos sinápticos da rede devem ser definidos através de um procedimento de inicialização intrínseco ao problema a ser solucionado, as análises da rede para os problemas de otimização serão realizados utilizando um caso específico para estudo: o Problema do Caixeiro Viajante.

2.2.6.2.1 Problema do Caixeiro Viajante

Para a solução do problema do caixeiro viajante com c cidades, a arquitetura da rede de Hopfield utiliza $m = c^2$ neurônios. Basicamente, a estrutura de neurônios é definida em forma de matriz de duas dimensões, onde uma dimensão representa a cidade sendo visitada, e a outra dimensão representa o tempo ou a sequência da visita realizada. Segundo esta arquitetura, um percurso válido é então representado através de uma única unidade ativada em cada linha e em cada coluna. Duas unidades ativadas em uma mesma linha indicam que uma cidade foi visitada duas vezes enquanto que duas unidades ativadas em uma coluna mostram que o viajante esteve em duas cidades ao mesmo tempo. A arquitetura da rede de Hopfield para resolução do problema do caixeiro viajante com 10 cidades é dada pela Figura 2.18.

Posição Cidade	1	2	3	4	5	6	7	8	9	10
A	$y_{A,1}$	$y_{A,2}$	$y_{A,3}$	$y_{A,4}$	$y_{A,5}$	$y_{A,6}$	$y_{A,7}$	$y_{A,8}$	$y_{A,9}$	$y_{A,10}$
B	$y_{B,1}$	$y_{B,2}$	$y_{B,3}$	$y_{B,4}$	$y_{B,5}$	$y_{B,6}$	$y_{B,7}$	$y_{B,8}$	$y_{B,9}$	$y_{B,10}$
C	$y_{C,1}$	$y_{C,2}$	$y_{C,3}$	$y_{C,4}$	$y_{C,5}$	$y_{C,6}$	$y_{C,7}$	$y_{C,8}$	$y_{C,9}$	$y_{C,10}$
D	$y_{D,1}$	$y_{D,2}$	$y_{D,3}$	$y_{D,4}$	$y_{D,5}$	$y_{D,6}$	$y_{D,7}$	$y_{D,8}$	$y_{D,9}$	$y_{D,10}$
E	$y_{E,1}$	$y_{E,2}$	$y_{E,3}$	$y_{E,4}$	$y_{E,5}$	$y_{E,6}$	$y_{E,7}$	$y_{E,8}$	$y_{E,9}$	$y_{E,10}$
F	$y_{F,1}$	$y_{F,2}$	$y_{F,3}$	$y_{F,4}$	$y_{F,5}$	$y_{F,6}$	$y_{F,7}$	$y_{F,8}$	$y_{F,9}$	$y_{F,10}$
G	$y_{G,1}$	$y_{G,2}$	$y_{G,3}$	$y_{G,4}$	$y_{G,5}$	$y_{G,6}$	$y_{G,7}$	$y_{G,8}$	$y_{G,9}$	$y_{G,10}$
H	$y_{H,1}$	$y_{H,2}$	$y_{H,3}$	$y_{H,4}$	$y_{H,5}$	$y_{H,6}$	$y_{H,7}$	$y_{H,8}$	$y_{H,9}$	$y_{H,10}$
I	$y_{I,1}$	$y_{I,2}$	$y_{I,3}$	$y_{I,4}$	$y_{I,5}$	$y_{I,6}$	$y_{I,7}$	$y_{I,8}$	$y_{I,9}$	$y_{I,10}$
J	$y_{J,1}$	$y_{J,2}$	$y_{J,3}$	$y_{J,4}$	$y_{J,5}$	$y_{J,6}$	$y_{J,7}$	$y_{J,8}$	$y_{J,9}$	$y_{J,10}$

Figura 2.18 : Rede de Hopfield para o problema do Caixeiro Viajante.

A idéia base para resolução de problemas de otimização utilizando a rede de Hopfield é de mapear as soluções nos pesos sinápticos da rede neural, de forma que os sinais de saída converjam para uma das soluções ótimas do problema. Para o problema específico do Caixeiro Viajante é utilizada a seguinte lógica de modelagem da rede neural:

- Como forma de restringir a ativação de neurônios pertencentes às mesmas linhas e colunas simultaneamente são utilizadas conexões sinápticas entre todos os neurônios pertencentes às mesmas linhas e às mesmas colunas.
- Os pesos sinápticos são então definidos de forma que as unidades localizadas nas mesmas linhas (ou nas mesmas colunas) tenham tendência de não serem ativadas simultaneamente.
- Estruturalmente, a rede considera a existência de conexões entre as unidades neurais em linhas e colunas adjacentes assim como são consideradas conexões entre a primeira e a última linha e coluna da rede. Estas conexões têm como objetivo representar as distâncias entre as cidades.
- Para as conexões entre unidades neurais não adjacentes, os pesos sinápticos são definidos com valor 0.

Para a identificação do peso sináptico entre as unidades de cada coluna, é então utilizada a seguinte equação [Fausett-1993, p.350]:

$$w(x, i; y, j) = -A\delta_{xy}(1 - \delta_{ij}) - B\delta_{ij}(1 - \delta_{xy}) + C - Dd_{xy}(\delta_{ij+1} + \delta_{ij-1}) \quad (2.52)$$

Onde δ_{ij} é o termo denominado Dirac delta, que possui valor 1 se $i = j$ e valor 0 caso contrário, x e y são índices reservados para indicar a cidade visitada, i e j são índices utilizados para indicar a posição do percurso, A corresponde a uma constante que fortalece a

influência da associação entre neurônios de uma mesma linha, B corresponde a uma constante que fortalece a influência das associações entre as unidades de mesma coluna, C fortalece a influência das conexões *bias* externas e D fortalece a influência das distâncias entre as cidades na equação. Todos estes parâmetros devem então ser escolhidos de forma a gerar soluções consistentes para o problema.

O algoritmo a ser executado para a resolução do problema pode então ser definido a seguir:

Passo 0.

Inicialização de todas as unidades.

Inicialização de Δt com pequeno valor.

Passo 1.

Enquanto a condição de parada não for alcançada, executar 2 a 6.

Passo 2.

Executar 3 a 5 n^2 vezes (n é o número de cidades)

Passo 3.

Escolher uma unidade aleatoriamente.

Passo 4.

Alteração do sinal interno do neurônio:

$$v_{x,i}(\text{novo}) = v_{x,i}(\text{velho}) + \Delta t \left[-v_{x,i}(\text{velho}) - A \sum_{j \neq i} v_{x,j} - B \sum_{y \neq x} v_{y,i} + C \left\{ N - \sum_x \sum_j v_{x,j} \right\} - D \sum_{y \neq x} d_{x,y} (v_{y,i+1} + v_{y,i-1}) \right]$$

Passo 5.

Aplicar a função de saída:

$$y_{x,i} = 0.5 \left(1 + \tanh(\alpha v_{x,i}) \right)$$

Passo 6.

Checagem da condição de parada (um percurso válido foi obtido após um número máximo de iterações, etc.).

Cabe ressaltar que apesar de ser possível resolver problemas de otimização com a rede de Hopfield, para a resolução do problema do caixeiro viajante esta rede neural é criticada por diversos autores, devido a dificuldades na obtenção de percursos ótimos, possibilidade de convergência para percursos inconsistentes e grande sensibilidade dos resultados de saída em relação aos parâmetros de configuração [Fausett-1993, p.352].

2.2.7 Máquina de Boltzmann

Conforme descrito em tópico anterior, a máquina de Boltzmann possui arquitetura de neurônios realimentados que operam de forma binária. Em alguns modelos desta rede neural, os neurônios são divididos em dois grupos distintos, um grupo composto por neurônios visíveis, que provêm uma interface da rede neural com o ambiente em que este opera, e um grupo de neurônios não visíveis, que operam livremente sem interferências externas. Os modelos de neurônios artificiais utilizados são os modelos estocásticos, cujas saídas são calculadas como probabilidades baseadas nas entradas e nos parâmetros livres dos neurônios,

i.e. pesos sinápticos e *bias*. De forma equivalente à rede de Hopfield, esta rede clássica utiliza pesos sinápticos simétricos e ausência de auto-realimentação.

Formalmente, a máquina de Boltzmann pode ser representada através da definição $B = (\mathcal{U}, \mathcal{C})$ onde $\mathcal{U}$ denota um conjunto finito de neurônios e $\mathcal{C}$ é um conjunto conexões, definido como pares de elementos de $\mathcal{U}$. Apesar de não existir uma auto-realimentação na rede, comumente é utilizada a conexão $(\{u, u\} | u \in \mathcal{U})$ para representação da *bias* individual de cada neurônio. Conexões entre neurônios distintos são então representadas através da notação $\{u, v\}$ onde $u, v \in \mathcal{U}$. Caso seja de interesse a não realimentação de dois neurônios é então definida uma conexão sináptica entre as duas unidades com valor 0. Caso dois neurônios da rede sejam conectados (peso sináptico com valor diferente de 0) então os neurônios são denominados “adjacentes”. O peso sináptico entre dois neurônios u e v é representado pela definição $w_{\{u,v\}}$ ou $w_{\{v,u\}}$ uma vez que são utilizadas conexões simétricas ($w_{u,v} = w_{v,u}$). As conexões podem ser tanto excitatórias ($w_{u,v} > 0$) quanto inibitórias ($w_{u,v} < 0$). Em operação, estas redes neurais podem possuir pesos sinápticos constantes ou dinâmicos dependendo da aplicação de interesse. A utilização de algoritmos que mantém os pesos sinápticos constantes pode ser aplicada para problemas de busca como problemas de otimização combinatorial, enquanto que algoritmos que alteram os pesos sinápticos dinamicamente podem resolver problemas relacionados com classificação, memória associativa, etc.

Dado o caráter binário dos neurônios estocásticos, estes são definidos como “ativados” quando a saída estiver com valor 1 ou são definidos como “desativados” quando o sinal de saída possuir valor 0. Com base no estado do neurônio individual, o estado da rede neural $\vec{x}$ (também denominado configuração da rede) pode ser definido como a união de todos os estados de neurônios que compõem a rede, formando um vetor de comprimento $|\mathcal{U}|$ (número de neurônios que compõem a rede). Utilizando estas definições, uma saída do neurônio u na configuração $\vec{x}$ pode ser identificada através da definição $x(u)$. O espaço de configurações ou estados possíveis $\mathfrak{R}$ é dado pelo conjunto de todas as configurações/estados possíveis para o sistema.

Outro conceito utilizado nos procedimentos de otimização é a denominada função de consenso $C : \mathfrak{R} \rightarrow \mathbb{R}$, também denominada “função de energia” por alguns autores como em [Haykin-1999, p.564] e [Smith-1999]. A função de consenso pode ser definida como:

$$C(\vec{x}) = \sum_{\{u,v\} \in \mathcal{C}} w_{u,v} x_u x_v \quad (2.53)$$

Em linhas gerais, o consenso da rede será alto quando muitas conexões excitatórias estiverem ativadas ($x_u x_v = 1$, ou seja, $x_u = 1$ e $x_v = 1$) e será baixo quando muitas conexões inibitórias estiverem ativadas.

Utilizando este conceito de função de consenso, é então possível utilizar a máquina de Boltzmann para resolução de problemas de otimização, onde a rede é utilizada para maximização da função $C(\bar{x})$. Cabe ressaltar que para a utilização desta rede neural nesta classe de problemas, surge a necessidade de mapeamento das soluções nas estruturas da rede, de forma que um estado de maior consenso represente uma “melhor” solução do problema. De forma equivalente à rede de Hopfield, cada problema deve ser mapeado intrinsecamente nas estruturas da rede neural como se pode observar em [Aarts&Korst-1989, p.153] onde diversos problemas de otimização são abordados de forma específica.

Dada a utilização de neurônios estocásticos, que sofrem alterações nos sinais de saída de forma probabilística, a rede neural como um todo apresenta alterações probabilísticas de estado que podem ser aceitas ou rejeitadas. O grau de alteração ou diferença entre dois estados consecutivos depende do parâmetro T de controle do neurônio estocástico (vide equação 2.4). Na medida em que este parâmetro se aproxima de zero, o comportamento da Máquina de Boltzmann passa a ser equivalente à da rede de Hopfield.

Para a aceitação ou rejeição das alterações de estados, os algoritmos da Máquina de Boltzmann devem avaliar se a alteração otimiza ou se reduz o valor da função de consenso. O cálculo do novo valor do consenso da rede deve ser realizado em acordo com o tipo de paralelismo adotado. As Máquinas de Boltzmann, por serem redes neurais, possuem uma arquitetura intrinsecamente paralela (de forma equivalente às redes neurais biológicas). As Máquinas de Boltzmann que utilizam processamento paralelo são conhecidos como máquinas paralelas, enquanto que as que não utilizam paralelismo são denominadas máquinas sequenciais. A diferença básica entre uma máquina sequencial e uma máquina paralela está na quantidade de neurônios que podem inverter seu sinal entre ciclos consecutivos de processamento. Para uma máquina sequencial, apenas um neurônio pode mudar seu estado por iteração do algoritmo, enquanto que para máquinas paralelas mais de um neurônio pode mudar de estado.

Para as máquinas paralelas, estas podem operar com ou sem um sincronismo de estados (todas as instâncias paralelas sincronizam os estados após um período de tempo ou um número de iterações). Para as máquinas com paralelismo síncrono, podem-se ter novamente duas classificações distintas. Nas operações síncronas as máquinas podem operar restringindo alterações de estado de alguns neurônios de acordo com propriedades da rede neural, paralelismo denominado “síncrono limitado”, ou as máquinas podem operar sem restrições para as alterações de estado, paralelismo que é denominado “síncrono ilimitado”. As mesmas restrições podem ser aplicadas para as máquinas assíncronas. Em resumo, 5 tipos distintos de Máquinas de Boltzmann (MB) podem ser descritas:

- MB sequenciais
- MB com sincronismo e paralelismo limitado
- MB com sincronismo e paralelismo ilimitado
- MB sem sincronismo e paralelismo limitado
- MB sem sincronismo e paralelismo ilimitado

As diferenças entre modos de operação serão apresentadas nos tópicos seguintes.

2.2.7.1 Máquina de Boltzmann Sequencial

Basicamente, esta categoria define o modo de operação onde apenas um neurônio é processado por vez. No processo é então analisada a possibilidade de inversão do sinal do neurônio u em análise, resultando em uma alteração do estado da rede $\vec{x}$ para um novo estado $\vec{x}^u$. Supondo que ocorra uma inversão no sinal de saída do neurônio u , a diferença da função de consenso é então dada por:

$$\Delta C_{\vec{x}}(u) = C(\vec{x}^u) - C(\vec{x}) \quad (2.54)$$

$$\Delta C_{\vec{x}}(u) = \left[x_u(u) \sum_{\{u,v\} \in \mathcal{C}_u} w_{\{u,v\}} x_u(v) + x_u^2(u) w_{\{u,u\}} \right] - \left[x(u) \sum_{\{u,v\} \in \mathcal{C}_u} w_{\{u,v\}} x(v) + x^2(u) w_{\{u,u\}} \right] \quad (2.55)$$

Onde $x_u(v)$ é uma função de identificação do valor do neurônio v no vetor de estado da rede $\vec{x}_u$, ou seja:

$$x_u(v) = \begin{cases} x(v) & \text{se } v \neq u \\ 1 - x(v) & \text{se } v = u \end{cases} \quad (2.56)$$

2.2.7.2 Máquina de Boltzmann com paralelismo síncrono limitado

Para o paralelismo síncrono, uma restrição na mudança de estado dos neurônios deve ser aplicada, de forma a permitir que cada instância de processamento paralelo consiga calcular uma variação na função de consenso de forma correta. Em resumo, a rede deve permitir mudanças de estado se e somente se as unidades não forem adjacentes, ou em outras palavras, se possuírem conexão sináptica com valor 0. Esta restrição é necessária já que as instâncias paralelas decidem de forma autônoma (sem informações sobre o processamento das outras instâncias de processamento) uma inversão de estado de uma unidade neural melhor ou não a função de consenso. Para exemplificar a idéia desta restrição, será considerada a máquina de Boltzmann de exemplo da Figura 2.16.

Figura 2.19 : Máquina de Boltzmann com neurônios desconectados.

Para esta Máquina de Boltzmann a função de consenso pode então ser calculada como:

$$C(\vec{x}) = 2 \left(w_{u_1 u_2} x_{u_1} x_{u_2} + w_{u_2 u_3} x_{u_2} x_{u_3} \right) \quad (2.57)$$

Baseado nesta função de consenso, se uma instância de processamento paralelo identificar uma alteração de estado do neurônio u_1 então a diferença na função de consenso será dada por

$$\Delta C_{\vec{x}}(u_1) = 2 w_{u_1 u_2} x_{u_2} (x'_{u_1} - x_{u_1}) \quad (2.58)$$

Para uma instância de processamento independente calcular se a inversão do neurônio u_1 otimiza a função de consenso, então o estado do neurônio u_2 não poderá mudar simultaneamente (uma vez que os neurônios u_1 e u_2 estão associados pelo termo $w_{u_1 u_2} x_{u_1} x_{u_2}$). Desta forma só será possível um processamento paralelo para os neurônios u_1 e u_3 (uma vez que o consenso não possui um termo associando a estes neurônios).

Com base nesta restrição, para este tipo de paralelismo, os neurônios devem ser separados em conjuntos de neurônios independentes, ou seja, neurônios desconectados (não adjacentes). Uma vez que os conjuntos foram identificados, o algoritmo se resume em escolher um subconjunto de elementos $\mathcal{U}_s \subset \mathcal{U}$ aleatoriamente a partir de apenas um conjunto identificado e avaliar se estas unidades invertem ou não seus sinais de saída. Dentre as unidades $\mathcal{U}_s$ avaliadas apenas $\mathcal{U}_{\vec{x}_1 \vec{x}_2}$ possuirão transições aceitas, desta forma gerando uma alteração do estado da rede de $\vec{x}_1$ para $\vec{x}_2$ onde $\vec{x}_1, \vec{x}_2 \in \mathfrak{R}$ ($\mathfrak{R}$ é o conjunto de todos os estados possíveis da rede neural). As unidades não alteradas são representadas através da definição $\mathcal{U}_s \setminus \mathcal{U}_{\vec{x}_1 \vec{x}_2}$. Neste modo de operação a diferença na função de consenso resultante das instâncias paralelas pode então ser calculada por:

$$C(\vec{x}_2) - C(\vec{x}_1) = \sum_{u \in \mathcal{U}_{\vec{x}_1 \vec{x}_2}} \Delta C_{\vec{x}_1}(u) \quad (2.59)$$

Onde

$$\Delta C_{\vec{x}_1}(u) = \begin{cases} -\Delta C_{\vec{x}_2}(u) & \text{se } u \in \mathcal{U}_{\vec{x}_1 \vec{x}_2} \\ \Delta C_{\vec{x}_2}(u) & \text{se } u \in \mathcal{U}_s \setminus \mathcal{U}_{\vec{x}_1 \vec{x}_2} \end{cases} \quad (2.60)$$

2.2.7.2.1 Paralelismo síncrono ilimitado

Em uma máquina de Boltzmann com paralelismo síncrono ilimitado, a alteração dos sinais entre neurônios adjacentes passa a não ser mais restringido, desta forma fazendo com que os

cálculos realizados por cada módulo paralelo possam resultar em valores errôneos da função de consenso. Como consequência deste problema, as equações que definem a diferença de consenso citado para o paralelismo síncrono não valem mais. Cabe ressaltar que a probabilidade de transições errôneas devido ao cálculo de consensos errôneos é reduzida com a redução do parâmetro de controle da pseudo-temperatura do sistema. Na medida em que este parâmetro se aproxima de 0, então as unidades passarão a não aceitar mais transições de estados, e a probabilidade de unidades adjacentes mudarem simultaneamente seus estados também se aproxima de 0.

2.2.7.3 Máquinas de Boltzmann assíncronas

Em uma máquina de Boltzmann com paralelismo assíncrono não existe nenhum mecanismo de sincronização de estados entre as unidades neurais. Desta forma as unidades podem mudar seus estados simultaneamente, mas transições de estado não mais são combinadas em sucessivos ciclos com pausa para sincronização dos estados. Nesta abordagem mais uma vez é possível classificar estas máquinas em duas categorias, com paralelismo assíncrono limitado e ilimitado, onde é permitida a inversão de unidades não adjacentes para o caso ilimitado, e é permitida a inversão de qualquer neurônio da rede no caso de paralelismo ilimitado.

2.2.7.3.1 Paralelismo assíncrono limitado

Esta abordagem pode ser realizada introduzindo o conceito de esquema de blocos [Aarts&Korst-1989, p.148] que impede que unidades adjacentes realizem transições de estado simultaneamente. Neste caso nenhuma transição de estado será aceita erroneamente e a saída da máquina de Boltzmann convergirá assintoticamente para uma distribuição estacionária ótima.

2.2.7.3.2 Paralelismo assíncrono ilimitado

Neste modo de operação, não é utilizado nenhum esquema impeditivo para as transições de estado de neurônios adjacentes, e conseqüentemente transições de estado errôneas poderão ocorrer.

2.2.7.4 Resfriamento

Uma vez que a máquina de Boltzmann utiliza neurônios estocásticos em sua arquitetura, existe então a possibilidade de controle da probabilidade de inversão dos sinais destas unidades através do parâmetro T (pseudo-temperatura), conforme apresentado para o modelo de neurônio artificial estocástico. Este controle tem como objetivo permitir que a rede simule um procedimento denominado “*Simulated Annealing*” na execução de algoritmos para resolução de problemas diversos, dentre os quais se destacam os problemas de otimização combinatorial. O procedimento é um algoritmo que simula o processo térmico denominado

“*Annealing*”, que visa obter estados de energia mínima em sólidos através de um processo de aquecimento e redução gradual de temperatura. Desta forma, para obtenção de estados de menor energia (ou de maior energia para problemas de maximização), a convergência da máquina é realizada através de um aumento da pseudo-temperatura com posterior redução gradual deste parâmetro de controle. Este procedimento é utilizado tanto para sistemas paralelos quanto para os não paralelos. A estratégia ou a função que controla este processo de resfriamento é denominada “cronograma de resfriamento”.

Para a máquina de Boltzmann paralela, uma vez que é vantajoso reduzir a comunicação global entre as unidades neurais em execução, para obtenção de sistemas com alto paralelismo, então é utilizada uma solução onde cada neurônio da rede possui um cronograma de resfriamento próprio, desacoplando as unidades da rede. Um procedimento simples para resfriamento visa gerar uma seqüência finita de valores do parâmetro de controle sempre aplicando uma redução gradual nos valores para cada unidade neural. Basicamente um parâmetro de controle pode ser definido como o anterior multiplicado por uma constante real com valor inferior a 1.

2.2.7.5 Otimização Combinatorial

Esta classe de problemas pode ser vista como um problema de busca de soluções em um espaço de versões. Conforme citado anteriormente, para a resolução desta classe de problemas é então utilizada uma estrutura de máquina de Boltzmann com pesos sinápticos constantes durante a fase de operação.

Quanto ao problema de otimização combinatorial, segundo [Aarts&Korst-1989, p.153] este pode ser definido como uma tupla $(\mathcal{S}, \mathcal{S}', f)$, onde $\mathcal{S}$ denota um conjunto finito de soluções, o parâmetro $\mathcal{S}' \subseteq \mathcal{S}$ representa o conjunto de soluções consistentes para o problema e a função $f : \mathcal{S} \rightarrow \mathbb{R}$ representa a função de custo que atribui um número real para cada solução. Com base nesta definição, o problema visa identificar uma solução consistente cuja função de custo f seja ótima. Algebricamente uma solução de um problema de otimização é representada pela variável $\mathcal{X}$.

Utilizando a estrutura da máquina de Boltzmann, é então possível resolver problemas pertencentes à classe de otimização combinatorial. Em essência, cada tipo de problema deve ser mapeado de uma forma adequada para a estrutura da rede levando em consideração as seguintes propriedades [Aarts&Korst-1989, p.148]:

- O problema de otimização deve ser formulado como um problema binário, ou seja, sendo uma solução $\mathcal{X} = \{x_1, \dots, x_m\}$ composta por saídas onde $x_i = \{0, 1\}, i = 1, \dots, m$.
- Definir uma máquina de Boltzmann $\mathcal{B} = (\mathcal{U}, \mathcal{C})$ ($\mathcal{U}$ denota um conjunto finito de neurônios e $\mathcal{C}$ é um conjunto de conexões, representado por pares de elementos de $\mathcal{U}$), tal que o estado de cada unidade neural da rede $u_i \in \mathcal{U} = \{u_1, \dots, u_n\}$ mapeia seu valor diretamente em um elemento $x_i = \{0, 1\}$ do vetor de estado $\vec{x}$ da rede, ou seja,

se $u_i = 0$ então a rede neural possuirá $x_i = 0$ e se $u_i = 1$ então a rede neural possuirá $x_i = 1$, para $i = 1, \dots, m$.

- Define um conjunto de conexões $\mathcal{C}$ e os pesos sinápticos de tal forma que a função de consenso seja consistente e preserve a ordem de aptidão das soluções (uma função de consenso com maior valor representa uma “melhor” solução para o problema).

Cabe ressaltar que uma solução é dita consistente se o valor máximo da função de consenso representa uma solução consistente. A preservação da ordem de aptidão indica que para todo estado da rede $\vec{x}_1, \vec{x}_2 \in \mathfrak{R}$, existindo uma função bijetora $g : \mathfrak{R} \rightarrow \mathcal{S}$ ($\mathfrak{R}$ é o conjunto de configurações ou estados possíveis e $\mathcal{S}$ denota um conjunto finito de soluções) que mapeia a máquina de Boltzmann nas soluções do problema de otimização, de tal forma que as soluções $g(\vec{x}_1), g(\vec{x}_2) \in \mathcal{S}'$ satisfazem $f(g(\vec{x}_1)) > f(g(\vec{x}_2)) \Rightarrow \mathcal{C}(\vec{x}_1) > \mathcal{C}(\vec{x}_2)$ para os problemas de minimização e $f(g(\vec{x}_1)) < f(g(\vec{x}_2)) \Rightarrow \mathcal{C}(\vec{x}_1) > \mathcal{C}(\vec{x}_2)$ para os problemas de maximização.

2.2.7.5.1 Problema do Caixeiro Viajante

Para a resolução deste problema utilizando a máquina de Boltzmann, a arquitetura da rede neural utilizada equivale à apresentada para a rede de Hopfield. As propriedades das conexões também são herdadas, onde a arquitetura da rede é baseada na utilização de $m = c^2$ neurônios (c representa o número de cidades do problema) definidos em forma de matriz de duas dimensões, onde uma dimensão representa a cidade sendo visitada, e a outra dimensão representa o tempo ou a seqüência da visita realizada. De forma equivalente à teoria apresentada para a rede de Hopfield, um percurso válido é representado através de uma única unidade ativada em cada linha e em cada coluna.

Para o problema em questão, o conjunto de conexões sinápticas é dividido em três grupos distintos [Aarts&Korst-1989, p.166]:

Conexões *bias*

$$\mathbb{C}_{\text{bias}} = \{\{u_{xi}, u_{xi}\}\} \quad (2.61)$$

Conexões de distância

$$\mathbb{C}_{\text{distancia}} = \{\{u_{xi}, u_{yj}\} \mid x \neq y \wedge i = (j+1) \bmod c\} \sqrt{b^2 - 4ac} \quad (2.62)$$

Conexões inibitórias

$$\mathbb{C}_{\text{inibição}} = \{\{u_{xi}, u_{yj}\} \mid (x = y \wedge i \neq j) \vee (x \neq y \wedge i = j)\} \quad (2.63)$$

Onde os índices x e y são reservados para indicar a cidade visitada, i e j são índices utilizados para indicar a posição do percurso e c representa o número de cidades do problema.

Com base na estrutura da rede e nas definições acima, a escolha dos pesos sinápticos deve ser realizada de forma que as seguintes propriedades sejam satisfeitas [Aarts&Korst-1989, p.166]:

$$\forall \{u_{xi}, u_{xi}\} \in \mathbb{C}_{\text{bias}} : w_{\{u_{xi}, u_{xi}\}} > \max \{d_{xk} + d_{xl} \mid k \neq l\} \quad (2.64)$$

$$\forall \{u_{xi}, u_{yj}\} \in \mathbb{C}_{\text{distancia}} : w_{\{u_{xi}, u_{yj}\}} = -d_{xi} \quad (2.65)$$

$$\forall \{u_{xi}, u_{yj}\} \in \mathbb{C}_{\text{inibição}} : w_{\{u_{xi}, u_{yj}\}} < -\min \{w_{\{u_{xi}, u_{xi}\}}, w_{\{u_{yj}, u_{yj}\}}\} \quad (2.66)$$

Com base nestas propriedades a solução do problema se resume na minimização da função de consenso, que levando em consideração a existência das conexões sinápticas pode ser reescrita como:

$$f(\vec{x}) = C(\vec{x}) = \sum_{\{u_{xi}, u_{xi}\} \in \mathbb{C}_{\text{bias}}} w_{\{u_{xi}, u_{xi}\}} (x_{u_{xi}}^t)^2 + \sum_{\{u_{xi}, u_{yj}\} \in \mathbb{C}_{\text{distance}}} w_{\{u_{xi}, u_{yj}\}} x_{u_{xi}}^t x_{u_{yj}}^t \quad (2.67)$$

2.2.7.6 Classificação

A classificação pode ser definida como a divisão de elementos pertencentes a um espaço de versões em conjuntos distintos denominados classes. Neste procedimento, é adotado um operador de classificação, que analisa as características dos elementos de entrada e define uma saída específica indicando a qual classe o conjunto pertence.

Segundo [Aarts&Korst-1989, p.180] um problema de classificação pode ser formalizado como um par (O, S) onde O denota um conjunto finito de objetos $O = \{o_1, \dots, o_n\}$ e S denota uma coleção de subconjuntos disjuntos $S_1, \dots, S_m$ pertencentes a O . O problema de classificação visa selecionar automaticamente um dado objeto $o_i \in O$ como um membro de um dos subconjuntos $S_j \in S$.

A máquina de Boltzmann pode ser utilizada como um modelo de classificação. Para isso os estados de um subconjunto de unidades da máquina são selecionados como unidades neurais de entrada. O algoritmo de execução então tenta maximizar a função de consenso dados os estados já definidos pelo ambiente. Após a maximização do consenso os estados indicados nas unidades de saída representam o subconjunto S_j que indica a classificação do objeto o_i . No

procedimento de classificação, basicamente existem 2 problemas que necessitam ser resolvidos [Aarts&Korst-1989, p.181]:

1. Achar uma estrutura apropriada para o problema, ou seja, um conjunto de unidades e conexões entre as unidades.
2. Achar um conjunto de pesos sinápticos apropriados.

A identificação da arquitetura da rede neural visando resolver um problema de classificação deve ser realizada de forma aderente para o problema sendo resolvido. Desta forma não há uma regra geral para mapeamento de problemas nas estruturas da rede, devendo cada problema ser analisado de forma pontual.

Uma vez identificada uma arquitetura adequada para representação dos problemas, a classificação passa a ser possível após a aplicação da regra de aprendizado de Boltzmann conforme visto anteriormente.

Segundo [Haykin-1999, p.567] o processo se resume na separação das unidades neurais em dois conjuntos distintos, um conjunto de neurônios escondidos, que representarão as saídas do sistema de classificação e outro de neurônios visíveis, que representarão as entradas a serem classificadas. Com a existência destas duas classes distintas, os estados da rede podem ser classificados através de duas variáveis aleatórias distintas, uma variável aleatória X_α para representar os estados dos neurônios visíveis e uma variável aleatória X_β para representação dos estados dos neurônios escondidos.

Após a separação dos neurônios nas devidas classes, o procedimento visa apresentar à rede neural um conjunto de exemplos de treinamento previamente conhecidos. Para cada exemplo de treinamento apresentado à rede é então realizado um ajuste dos pesos sinápticos de acordo com as equações (2.18), (2.19) e (2.20). Em essência, este mecanismo se assemelha ao aprendizado por correção de erro, onde o sinal de erro é utilizado para corrigir os pesos sinápticos por iteração. Porém, neste modo de aprendizado, em vez da rede neural considerar a diferença entre o sinal obtido e o sinal esperado, o algoritmo considera a diferença entre a probabilidade dos neurônios estarem no estado desejado quando a rede estiver com os neurônios visíveis fixados com os valores de entrada do exemplo, e a probabilidade de encontrar os neurônios no estado desejado quando a rede estiver operando sem influência externa.

2.3 Computação Evolutiva

A Computação Evolutiva (CE) é uma área de conhecimento pertencente à IC. Este modelo é baseado na abordagem Evolucionista, tendo inspiração nos modelos evolutivos propostos por Darwin onde são simuladas características como Seleção, Recombinação, Mutação e Avaliação sobre os indivíduos de uma população, que representam soluções possíveis de um problema. Essas características visam gerar estruturas capazes de simular a capacidade natural de evolução biológica, obtendo sistemas capazes de otimizar uma solução tanto para ambientes estáticos como dinâmicos. De forma geral, o algoritmo pode ser reduzido à um

ciclo composto pelas funções de recombinação, mutação, avaliação e seleção conforme generalizado em [Jong&Lawrence&Schwefel-1997, p. B1.1:2]:

Entradas:

μ número de indivíduos pais

λ número de indivíduos descendentes

Θ_t parâmetros genéricos utilizados pela função de finalização

Θ_r parâmetros genéricos utilizados pelo operador de recombinação

Θ_m parâmetros genéricos utilizados pelo operador de mutação.

Θ_s parâmetros genéricos utilizados pelo operador de seleção.

Saídas:

a^* melhor indivíduo encontrado ao longo da execução, ou

P^* , melhor população encontrada ao longo da execução.

```
1      t ← 0;
2      P(t) ← initialize( $\mu$ );
3      F(t) ← evaluate(P(t),  $\mu$ );
4      while (termination (P(t),  $\Theta_t$ ) ≠ true) do
5          P'(t) ← recombine(P(t),  $\Theta_r$ );
6          P''(t) ← mutate(P'(t),  $\Theta_m$ );
7          F(t) ← evaluate(P''(t),  $\lambda$ );
8          P(t+1) ← select(P''(t), F(t),  $\mu$ ,  $\Theta_s$ );
9      t ← t+1;
10     end while
```

Segundo a literatura, grande parte das implementações dos Algoritmos Evolutivos (AEs) descende de três abordagens distintas [Bäck&Hammel&Schwefel-1997]:

1. Algoritmo Genético
2. Programação Evolutiva
3. Estratégias Evolutivas

As principais diferenças entre as diferentes teorias evolutivas são baseadas nas seguintes propriedades [Bäck&Hammel&Schwefel-1997, IV]:

1. Representação dos indivíduos.
2. Implementação dos operadores de Mutação e/ou Recombinação.
3. Implementação dos operadores de Seleção e/ou Reprodução.

O paradigma denominado Algoritmo Genético, foi originalmente proposto como um modelo adaptativo genérico, mas a aplicação mais comum para este algoritmo está relacionada com otimização [Bäck&Hammel&Schwefel-1997, Introdução] Segundo a abordagem clássica

do algoritmo, existem basicamente três itens que o diferenciam em relação às demais áreas da CE [Jong&Lawrence&Schwefel-1997, p.B1.2:1]:

1. Representação. O algoritmo padroniza a representação dos genes como *string* de bits.
2. Método de seleção proporcional. Segundo o padrão do algoritmo, os indivíduos são selecionados para reprodução com uma probabilidade proporcional ao desempenho observado.
3. Método de variação genética. Para a reprodução, o algoritmo utiliza o mecanismo denominado *crossover*, onde partes do gene dos indivíduos pais são utilizadas para a criação do indivíduo filho.

Segundo [Jong&Lawrence&Schwefel-1997, p.B1.2:1] uma das características marcantes dos Algoritmos Genéticos está na utilização do *crossover* como operador de Recombinação. Para as demais características citadas acima, algumas implementações do Algoritmo Genético utilizam outros operadores de Seleção além de utilizar outros padrões de representação distintas da *string* de bits. Independente destas variações, as diferentes implementações do algoritmo utilizam como base os conceitos de *Schemata* ou blocos de construção de genes para análise do problema.

Diferentemente do Algoritmo Genético, o modelo denominado Programação Evolutiva tem foco na alteração de características de populações, não manipulando os indivíduos e suas características isoladamente. Dada esta característica, esta teoria conseqüentemente não faz uso de mecanismos de reprodução baseados em recombinação genética. Basicamente, as populações são geradas com base em *Schematas* de genes (máscaras identificando padrões semelhantes de genes) e as populações descendentes são definidas com um aumento no número de indivíduos mais aptos, sendo os indivíduos menos aptos abatidos da população [Jong&Lawrence&Schwefel-1997, p.B1.4:1].

O modelo denominado Estratégias Evolutivas aplica os mesmos conceitos do Algoritmo Genético, onde são realizadas operações considerando indivíduos isoladamente. Esta teoria enfatiza os operadores de Mutação e Recombinação como principais mecanismos do algoritmo, além de possuir um operador de Seleção determinístico, sem utilização de mecanismos aleatórios. Nesta teoria, as populações constituídas pelos pais e descendentes normalmente diferem em tamanho [Jong&Lawrence&Schwefel-1997, p. B1.1:2].

Para melhor entendimento das descrições a serem apresentadas nos tópicos seguintes serão apresentadas as principais terminologias a serem utilizadas:

- Genótipo: Nos sistemas naturais, cada indivíduo é composto por um ou mais cromossomos. O conjunto cromossomos que compõe a carga genética de um indivíduo é então denominado genótipo.
- Cromossomo: Nos sistemas naturais, um cromossomo corresponde a um conjunto de genes.
- Genes: Nos sistemas naturais, os genes são definidos como segmentos do cromossomo. Comumente cada gene possui um conjunto de valores possíveis, sendo estes valores predeterminados denominados alelos.
- Alelos: representam os valores possíveis dos genes.

- Fenótipos: representam as características observáveis dos organismos resultantes da interação do genótipo com o ambiente.

2.3.1 Algoritmos Genéticos

Os Algoritmos Genéticos (AGs) representam uma das áreas da CE e são baseados na utilização de operadores de Recombinação, Mutação, Avaliação e Seleção aplicados sobre um conjunto de indivíduos de uma população. Uma das características marcantes dos modelos desta área está na utilização do operador de Recombinação baseado na utilização de *crossover*. Para a representação dos indivíduos, comumente é utilizada a representação baseada em sequência de bits, porém outras representações também são possíveis. Para o operador de Seleção, é utilizado um método de seleção com probabilidade proporcional à aptidão dos indivíduos. Outra característica deste algoritmo está na análise dos problemas baseados nos conceitos de *Schemata*, que representam blocos de construção dos indivíduos da população.

Segundo o procedimento padrão do algoritmo, uma população de indivíduos $P(0)$ é gerada através de algum mecanismo de inicialização (heurística aleatória ou outra adequada para o problema) e posteriormente aptidão de cada indivíduo é avaliada. Com base na população $P(t-1)$ o algoritmo aplica um operador de Seleção para geração de um subconjunto de indivíduos $C(t)$ mais aptos com base em uma probabilidade proporcional à performance observada de cada indivíduo. Após o procedimento de Seleção, ocorre então a aplicação dos operadores de Mutação e Recombinação, desta forma gerando um novo conjunto $C'(t)$. As taxas de mutação e configurações da recombinação devem ser escolhidas de acordo com cada projeto. Outras configurações a serem consideradas referem-se ao número de descendentes que cada pai deve gerar e quantos indivíduos deverão ser selecionados como pais para cada nova iteração do algoritmo. Após a criação dos novos descendentes, uma nova população é então formada a partir dos indivíduos pais e seus descendentes. Como o número de indivíduos M é mantido fixo a cada nova geração, caso o número de descendentes seja menor do que M , o procedimento para completar os indivíduos faltantes pode ser baseado em uma seleção randômica entre os indivíduos pais. Caso o número de descendentes seja exatamente M , então a população da geração anterior será completamente sobrescrita pelos filhos.

A lógica básica executada pelo algoritmo genético pode ser vista em [Jong&Lawrence&Schwefel-1997, B1.2:2]:

```

 $t = 0$ ;
Inicialização de  $P(t)$ ;
Avaliação de  $P(t)$ ;
Iteração até condição de parada:
{
     $t = t + 1$ ;
     $C(t)$  = Seleção para Recombinação/Mutação a partir de  $P(t-1)$ ;
     $C'(t)$  = Mutação com Recombinação de  $C(t)$ ;
    Avaliação de  $C'(t)$ ;
     $P(t)$  = Seleção e sobreposição a partir de  $P(t-1)$  com  $C'(t)$ ;
}

```

Onde P representa a população, C um subgrupo de indivíduos selecionados e C' um subgrupo de indivíduos após operações de recombinação e mutação.

Detalhes de cada um dos operadores evolutivos utilizados pelos AGs serão analisados nos tópicos seguintes.

2.3.1.1 Avaliação

Este mecanismo se resume à aplicação de uma função objetivo a ser otimizada. O valor de saída desta função irá influenciar o processo de Seleção dos indivíduos para as operações de recombinação e mutação.

2.3.1.2 Seleção

Segundo o paradigma dos AGs, podem existir diferentes implementações deste operador conforme visto em [Jong&Lawrence&Schwefel-1997, B1.2:2]:

- Método de seleção direta: Neste método, a probabilidade de seleção de um indivíduo é definida diretamente através da aptidão observada. Este tipo de seleção pode resultar em pouca seletividade quando comparadas a duas funções distintas como $y = ax^2$ e $y = ax^2 + b$. Caso os valores de b sejam demasiadamente grandes em comparação com o termo ax^2 então as probabilidades de seleção de cada indivíduo da população serão próximos, fazendo como que o algoritmo não apresente boa seletividade.
- Método de seleção relativa: Esta estratégia tem como objetivo solucionar o problema da pouca seletividade na avaliação direta. Neste contexto, todas as avaliações geradas são normalizadas em relação ao pior indivíduo da população. Em contraste com o método de seleção direta, apresentado acima, este método de seleção pode implicar em uma excessiva seletividade, fazendo como que as soluções com melhor aptidão tomem conta da população rapidamente (convergência prematura), comprometendo o resultado do algoritmo. Uma alternativa a este problema é a utilização da Seleção

ranqueada onde a probabilidade de seleção é associada à classificação do indivíduo, ou seja, as probabilidades de seleção são então determinadas através de uma função linear, evitando o problema de excessiva seletividade.

- Método de seleção por torneio: Neste mecanismo, um subconjunto de indivíduos é selecionado aleatoriamente, sendo o operador de Seleção utilizado para identificação dos mais aptos do grupo e não de toda a população em conjunto. Esta solução permite um controle no nível de seletividade, que pode ser controlada através do número de indivíduos escolhidos para o subconjunto de competição.

Sobre estes mecanismos de seleção, uma estratégia que pode ser aplicada a todos os métodos é denominado Estratégia Elitista. Segundo esta variante, sempre será mantida na população da nova geração o indivíduo da geração anterior com melhor performance.

2.3.1.3 Mutação

Este operador tem como objetivo gerar uma alteração randômica de algum dos alelos ou bits da *string* de identificação dos indivíduos. Em outras palavras, novas soluções são obtidas através de alterações aleatórias de indivíduos da população.

2.3.1.4 Recombinação

Como citado anteriormente, o operador de Recombinação utilizado pelo Algoritmo Genético utiliza como base o método denominado *crossover*. Este mecanismo visa gerar novos indivíduos a partir de partes de duas soluções pais. O procedimento de recombinação pode então utilizar diferentes políticas para a definição das partes a serem trocadas entre os indivíduos:

- *Crossover* de um ponto: Segundo esta política, a recombinação é realizada sobre segmentos de *strings* definidos a partir de um único ponto de corte nos pais. A identificação do locus ou ponto de corte é definida aleatoriamente e os bits localizados após esta posição são trocados entre os dois indivíduos.

Figura 2.20 : *Crossover* de um ponto.

- *Crossover* multiponto: segundo esta política, são definidos dois ou mais pontos de corte dos segmentos de *string* dos pais. A partir destes pontos de corte são então

definidos os seguimentos que serão trocados entre os pais para a criação dos descendentes.

Figura 2.21 : *Crossover* multiponto.

- *Crossover* uniforme: Segundo esta estratégia, as trocas ocorrem não sobre segmentos e sim sobre bits individuais dos pais. Os bits a serem trocados são selecionados aleatoriamente.

Figura 2.22 : *Crossover* uniforme.

2.3.1.5 Schemata

Segundo os conceitos básicos dos AGs, o comportamento das soluções podem ser avaliados com base nos conceitos denominados *Schemata*, que representam blocos de construção dos indivíduos. Dada uma estrutura representando uma *string* de bits de comprimento L , um *Schemata* representa um máscara composta pelos caracteres “0”, “1” e/ou “#”, onde “#” representa um valor coringa, ou seja, um dos dois caracteres binários. Basicamente, os *Schemata* são analisados como partições do espaço L dimensional. Como exemplo um *Schemata* 1# representa tanto o indivíduo 10 quanto o indivíduo com *string* 11.

Sobre o conceito de *Schemata*, são então analisadas duas propriedades distintas: a ordem e o comprimento. A ordem de um *Schemata* é definida como o número de caracteres com valores definidos, ou seja, com valor “0” ou “1”. O comprimento do *Schemata* é então definido como a distância entre o primeiro e o último locus com valores definidos. Como exemplo, se for considerada uma *string* #01##1###, será obtido um *Schemata* de ordem 3 (pois existem 3 valores específicos) e comprimento 4 (uma vez que o primeiro bit específico encontra-se na posição 2 e o último na posição 6).

Com base nos *Schematas*, é possível então avaliar o algoritmo em execução como uma busca baseada na recombinação de elementos pertencentes aos subespaços definidos. Após

sucessivas iterações do algoritmo, a busca passa a ser realizada dentro de um subespaço reduzido de indivíduos mais aptos, uma vez que os indivíduos baseados nos melhores *Schematas* vão sendo selecionados em relação aos indivíduos dos *Schematas* de menor performance. Caso dois indivíduos de mesmo *Schemata* sejam selecionados para o processo de recombinação, então o descendente também pertencerá ao mesmo *Schemata* dos pais, e o algoritmo passará a focar a busca dentro do subespaço definido pelo bloco de construção comum aos indivíduos.

2.3.2 Programação Evolutiva

A programação evolutiva se diferencia das outras técnicas evolutivas por focar na avaliação de populações e não de indivíduos isoladamente. Com base nesta característica, a Programação Evolutiva não atribui crédito individual a nenhuma estrutura dos organismos pertencentes à população vencedora, desta forma sendo descartado pelo algoritmo o operador de Recombinação e sendo utilizados unicamente os operadores evolutivos de Mutação, Avaliação e Seleção.

Segundo o algoritmo, o processamento é iniciado com uma operação de inicialização das populações, onde μ soluções distintas são geradas com base em algum mecanismo (heurística randômica ou outro mecanismo apropriado) e posteriormente avaliadas. Após a definição de uma população $P(t)$, ocorre então a aplicação do operador de Mutação, onde cada membro i da população gera λ_i descendentes formando um subconjunto de indivíduos $P'(t)$. Com base na nova população, é então aplicado o operador de Seleção. Algumas lógicas comumente adotadas para este operador de Seleção são:

1. As melhores μ soluções são mantidas para se tornarem pais da nova geração.
2. As melhores μ soluções são estatisticamente mantidas.
3. É realizada uma seleção proporcional a aptidão que esta apresenta.

Na maioria das aplicações o tamanho da população se mantém constante ao longo das iterações, mas esta regra pode ser alterada de acordo com o problema sendo resolvido. O processo é finalizado quando a solução chega a uma qualidade predeterminada de indivíduos, quando um número máximo de iterações é atingido ou quando algum outro critério de interesse para o problema é satisfeito.

Uma forma genérica do algoritmo de Programação Evolutiva pode ser vista em [Jong&Lawrence&Schwefel-1997, p.B1.4:5]:

Inicialização: $P(0) := \{a'_1(0), a'_2(0), \dots, a'_\mu(0)\}$
 Avaliar População Inicial: $P(0) := \{\Phi(a'_1(0)), \Phi(a'_2(0)), \dots, \Phi(a'_\mu(0))\}$
 Iteração até condição de parada:
 {
 Mutação: $P'(t) := m_{\Theta_m}(P(t))$
 Avaliação: $P(t) := \{\Phi(a'_1(t)), \Phi(a'_2(t)), \dots, \Phi(a'_\lambda(t))\}$
 Seleção: $P(t+1) := s_{\Theta_s}(P'(t) \cup Q)$
 $t := t + 1$
 }

Onde a' representa um indivíduo da população; $\mu \geq 1$ corresponde ao número de pais da população; $\lambda \geq 1$ corresponde ao número de filhos gerados a partir dos pais; $P(t) := \{a'_1(t), a'_2(t), \dots, a'_\mu(t)\}$ corresponde à população no instante de tempo t ; $\Phi: I \rightarrow \mathbb{R}$ representa o operador de mapeamento de aptidão; m_{Θ_m} corresponde ao operador de mutação com parâmetros de controle Θ_m ; $s_{\Theta_s}: (I^\lambda \cup I^{\mu+\lambda}) \rightarrow I^\mu$ corresponde ao operador de seleção; $Q \in \{\emptyset, P(t)\}$ representa o conjunto de indivíduos a serem selecionados pelo operador de seleção.

Neste algoritmo, restrições podem ser adicionadas normalmente sendo consideradas na inicialização das populações e nos operadores de Mutação. Nestes procedimentos, a restrição pode ser considerada segundo duas diferentes políticas:

1. O operador de mutação sempre gera soluções legítimas.
2. Pode ser aplicado um operador de penalidade para soluções situadas fora da região de validade.

2.3.3 Estratégias Evolutivas

Nas Estratégias Evolutivas (EEs), a representação dos genes dos indivíduos não está amarrada a uma representação através de *strings* de bits e não existem restrições para o uso dos operadores de Seleção, como acontece com os AGs. Uma das características marcantes desta área é a notação bem definida para descrição do algoritmo e dos operadores. Segundo [Bäck&Hammel&Schwefel-1991], são descritos quatros diferentes cenários de execução dos algoritmos evolutivos:

- Estratégia Evolutiva $(1+1)$ –ES: Esta estratégia visa descrever o comportamento do algoritmo evolutivo para o caso específico de populações compostas por dois indivíduos, um pai e um descendente;

- Estratégia Evolutiva $(\mu+1)-ES$: Esta estratégia descreve um cenário onde $\mu > 1$ pais participam da geração de um único descendente, resultando em uma população de $(\mu+1)$ indivíduos.
- Estratégia Evolutiva $(\mu+\lambda)-ES$: Estratégia evolutiva que gera λ filhos a partir de μ pais, e escolhe μ mais aptos a partir dos $(\mu+\lambda)$ indivíduos em competição (pais e filhos competem juntos para definir os mais aptos de uma geração). Segundo a notação utilizada, as variáveis citadas devem respeitar a seguinte relação $1 \leq \mu \leq \lambda \leq \infty$.
- Estratégia Evolutiva $(\mu,\lambda)-ES$: Estratégia evolutiva que gera λ filhos a partir de μ pais e seleciona μ indivíduos mais aptos a partir dos λ indivíduos filhos em competição. Segundo a notação utilizada, as variáveis citadas devem respeitar a seguinte relação $1 \leq \mu < \lambda < \infty$.

2.3.3.1 Estratégia Evolutiva com dois indivíduos: $(1+1)-ES$

Neste cenário, tem-se um caso particular do modelo $(\mu+\lambda)-ES$ onde o algoritmo manipula $\mu=1$ pais e $\lambda=1$ descendentes. A análise desta EE separadamente ocorre devido à inexistência do operador de Recombinação, uma vez que este necessita de dois pais para sua execução. Segundo [Bäck&Hammel&Schwefel-1991] o modelo $(1+1)-ES$ pode ser descrito através de um 8-tupla:

$$(1+1)-ES = (P^0, m, s, c_d, c_i, f, g, t) \quad (2.68)$$

Onde P^0 corresponde à população inicial composta por um único indivíduo a_1^0 pai, definido como $(\vec{x}, \vec{\sigma}) \in I$, $I \in \mathbb{R}^n \times \mathbb{R}^n$, $\vec{x}$ sendo o genótipo do indivíduo e $\vec{\sigma} \in \mathbb{R}^n$ sendo o desvio padrão utilizado no operador de mutação; O operador $m: I \rightarrow I$ corresponde ao operador de mutação; O parâmetro $s: I \times I \rightarrow I$ corresponde ao operador de seleção; $c_d, c_i \in \mathbb{R}$ correspondem à escala de controle do operador de mutação; $f: \mathbb{R}^n \rightarrow \mathbb{R}$ corresponde à função objetivo; $g_j: \mathbb{R}^n \rightarrow \mathbb{R}$, $j=1, \dots, q$ correspondem às q funções de restrição do problema; E por fim, o operador $t: I \times I \rightarrow \{0,1\}$ corresponde ao critério de parada do algoritmo [Bäck&Hammel&Schwefel-1991, 2].

Segundo este modelo, P^t representa a população do algoritmo composto por um único indivíduo pai a_1^t , que é utilizada para geração do novo indivíduo filho através do operador de mutação m . A população gerada a partir do pai e do indivíduo filho são então definidos como uma nova população $P^{t'}$ conforme descrito a seguir:

$$P^{t'} = (a_1^{t'}, a_2^{t'}) \in I \times I \quad (2.69)$$

$$a_1^{t'} = P^t = (\vec{x}^t, \vec{\sigma}^t) \quad (2.70)$$

$$a_2^{t'} = m(P^t) = (\vec{x}^{t'}, \vec{\sigma}^t) \quad (2.71)$$

Este procedimento exhibe o uso do operador de mutação, que pode ter implementações diversas. Segundo o exemplo descrito em [Bäck&Hammel&Schwefel-1991] este operador é definido como:

$$\vec{x}^{t'} = \vec{x}^t + N_0(\vec{\sigma}^t) \quad (2.72)$$

Onde N_0 denota o vetor independente de números aleatórios segundo distribuição Gaussiana com média zero e desvio padrão σ_i^t , $i = 1, \dots, n$. Com base nestas definições, o operador de seleção pode ser aplicado para determinar o indivíduo da nova geração:

$$P^{t+1} = s(P^{t'}) = \begin{cases} a_2^{t'} & \text{se } f(x^{t'}) \leq f(x^t) \wedge g_j(x^{t'}) \geq 0 \forall j \in \{1, \dots, q\} \\ a_1^{t'} = P^t & \text{caso contrário} \end{cases} \quad (2.73)$$

Lembrando que $f: \mathbb{R}^n \rightarrow \mathbb{R}$ corresponde à função objetivo; $g_j: \mathbb{R}^n \rightarrow \mathbb{R}$, $j = 1, \dots, q$ correspondem às q funções de restrição do problema.

Para a função de identificação do critério de parada, esta indica que as iterações do algoritmo devem ser interrompidas quando $t(a_1^{t'}, a_2^{t'})$ resultar em 1. A função $t: I \times I \rightarrow \{0, 1\}$ não é restringida pelo algoritmo, podendo possuir diferentes implementação de acordo com as necessidades do problema.

Para o procedimento de controle do algoritmo em [Bäck&Hammel&Schwefel-1991] é sugerido um controle baseado na regra do 1/5 de Rechenberg, que sugere que para um algoritmo utilizar aproximadamente a máxima convergência para as soluções ótimas, a taxa de mutação aplicado aos indivíduos deve ser 1/5. No controle sugerido os valores de desvio padrão $\vec{\sigma}^t \in \mathbb{R}^n$ são considerados globais, ou seja:

$$\vec{\sigma}_i^t = \vec{\sigma}_j^t = \vec{\sigma}^t, \forall i, j \in \{1, \dots, n\} \quad (2.74)$$

Finamente, os parâmetros $c_d, c_i \in \mathbb{R}$ são então utilizados através do seguinte equacionamento [Bäck&Hammel&Schwefel-1991, 2]:

$$\overrightarrow{\sigma}^{t+1} = \begin{cases} c_d \overrightarrow{\sigma}^t & \text{if } p_s^t < 1/5 \\ c_i \overrightarrow{\sigma}^t & \text{if } p_s^t > 1/5 \\ \overrightarrow{\sigma}^t & \text{if } p_s^t = 1/5 \end{cases} \quad (2.75)$$

Onde p_s^t é a frequência de mutações com sucesso medido sobre intervalos de $10n$ ciclos, $c_d = 0.82$ e $c_i = 1/0.82$.

Cabe ressaltar que a função de controle de mutações do algoritmo representa uma função operação genérica de controle, sendo que implementações distintas podem ser utilizadas de acordo com as necessidades de cada problema.

2.3.3.2 Estratégia Evolutiva c/ múltiplos indivíduos $(\mu+1)-ES$

Neste cenário, o modelo $(\mu+1)-ES$ pode ser descrito através de um 10-tupla:

$$(\mu+1)-ES = (P^0, \mu, r, m, s, c_d, c_i, f, g, t) \quad (2.76)$$

Onde P^0 corresponde à população inicial $(a_1^0, \dots, a_\mu^0) \in I^\mu$, com a_μ^0 representando os indivíduos da população através de tuplas $(\vec{x}, \vec{\sigma}) \in I$ com $I = \mathbb{R}^n \times \mathbb{R}^n$; $\mu > 1$ indica o número de pais; $r: I^\mu \rightarrow I$ corresponde ao operador de recombinação genética; $m: I \rightarrow I$ corresponde ao operador de mutação, $s: I^{\mu+1} \rightarrow I$ corresponde ao operador de seleção, responsável pela remoção dos indivíduos menos aptos; $c_d, c_i \in \mathbb{R}$ correspondem à escala de controle do algoritmo; $f: \mathbb{R}^n \rightarrow \mathbb{R}$ corresponde à função objetivo; $g_j: \mathbb{R}^n \rightarrow \mathbb{R}$, $j = 1, \dots, q$ correspondem às q funções de restrição e $t: I^\mu \rightarrow \{0,1\}$ corresponde ao critério de parada do algoritmo [Bäck&Hammel&Schwefel-1991, 3].

Uma das diferenças deste modelo em relação à EE $(1+1)-ES$ está na necessidade de um operador de Recombinação r :

$$r(P^t) = a' = (\vec{x}', \vec{\sigma}') \in I, \vec{x}' \in \mathbb{R}^n, \vec{\sigma}' \in \mathbb{R}^n \quad (2.77)$$

Com as variáveis $\vec{x}'$ e $\vec{\sigma}'$ definidas como:

$$x'_i = \begin{cases} x_{a,i} & , \chi \leq 1/2 \\ x_{b,i} & , \chi > 1/2 \end{cases} \quad \forall i \in \{1, \dots, n\} \quad (2.78)$$

$$\sigma'_i = \begin{cases} \sigma_{a,i} & , \chi \leq 1/2 \\ \sigma_{b,i} & , \chi > 1/2 \end{cases} \quad \forall i \in \{1, \dots, n\} \quad (2.79)$$

Onde $a = (\vec{x}_a, \vec{\sigma}_a), b = (\vec{x}_b, \vec{\sigma}_b) \in I$ são dois pais escolhidos para o procedimento de recombinação genética e a variável χ denota uma variável aleatória no intervalo $[0,1]$. Por convenção, todos os pais de uma população possuem as mesmas probabilidades de serem escolhidos para reprodução e a variável aleatória χ é sorteada para cada componente x'_i e σ'_i .

No algoritmo é então aplicado um procedimento onde um novo indivíduo é gerado a partir do operador de recombinação, seguido do operador de mutação e por fim é realizada uma operação de Seleção conforme descrito em [Bäck&Hammel&Schwefel-1991, 43]:

$$P^{t'} = (a_1^{t'}, \dots, a_{\mu+1}^{t'}) = (a_1^t, \dots, a_{\mu}^t, m(r(P^t))) \quad (2.80)$$

$$P^{t+1} = s(P^{t'}) \text{ tal que } \forall a_i^{t+1} = (\vec{x}, \vec{\sigma}), \nexists a_j^t = (\vec{x}', \vec{\sigma}') : f(\vec{x}') < f(\vec{x}) \quad (2.81)$$

Para problemas em modo geral, é possível utilizar apenas um único valor constante de $\vec{\sigma}^t$ como desvio padrão. Porém, para aumentar a eficiência do modelo $(\mu+1)$ -ES comumente é utilizada a mesma heurística de ajuste do parâmetro σ^t baseada na regra do 1/5 de Rechenberg:

$$\sigma^{t+n} = \begin{cases} c_d \cdot \sigma^t & , se \quad p_s^t < 1/5 \\ c_i \cdot \sigma^t & , se \quad p_s^t > 1/5 \\ \sigma^t & , se \quad p_s^t = 1/5 \end{cases} \quad (2.82)$$

Onde p_s^t é a frequência de mutações com sucesso medido sobre intervalos de $10n$ ciclos, $c_d = 0.82$ e $c_i = 1/0.82$.

2.3.3.3 Estratégia Evolutiva $(\mu+\lambda)$ -ES e (μ, λ) -ES

Como citado anteriormente, a classe de algoritmo (μ, λ) -ES considera uma população de λ indivíduos filhos em competição, diferente do algoritmo $(\mu+\lambda)$ -ES onde uma população era composta pelos filhos e também pelos pais que competiram na geração anterior, desta forma resultando em uma população de $(\mu+\lambda)$ indivíduos em competição. De qualquer forma, ambas as abordagens citadas são compatíveis e podem ser representadas através de notações semelhantes:

$$(\mu+\lambda)\text{-ES} = (P^0, \mu, \lambda, r, m, s, \Delta\sigma, f, g, t) \quad (2.83)$$

$$(\mu, \lambda)\text{-ES} = (P^0, \mu, \lambda, r, m, s, \Delta\sigma, f, g, t) \quad (2.84)$$

Onde P^0 corresponde à população inicial $(a_1^0, \dots, a_\mu^0) \in I^\mu$ com a_μ^0 representando os genomas dos indivíduos representados por tuplas $(\vec{x}, \vec{\sigma}) \in I$ com $I = \mathbb{R}^n \times \mathbb{R}^n$; $\mu \in N$ indica o número de pais; $\lambda \in N$ indica o número de filhos a serem gerados pelos pais selecionados da geração anterior onde $\lambda > \mu$; $r: I^\mu \rightarrow I$ corresponde ao operador de recombinação genética; $m: I \rightarrow I$ corresponde ao operador de mutação; $s: I^\lambda \rightarrow I^\mu$ (para a estratégia $(\mu, \lambda) - ES$) ou $s: I^{\lambda+\lambda} \rightarrow I^\mu$ (para a estratégia $(\mu, \lambda) - ES$) corresponde ao operador de seleção responsável pela remoção dos indivíduos menos aptos da população; $\Delta\sigma \in \mathbb{R}$ correspondendo a um parâmetro de controle de escala das operações de mutação; $f: \mathbb{R}^n \rightarrow \mathbb{R}$ corresponde à função objetivo; $g_j: \mathbb{R}^n \rightarrow \mathbb{R}$, $j=1, \dots, q$ corresponde às q funções de restrição do problema e $t: I^\mu \rightarrow \{0,1\}$ corresponde ao critério de parada do algoritmo [Bäck&Hammel&Schwefel-1991, 4].

Segundo estas abordagens genéricas, o operador de mutação m pode ser definido como:

$$a_i'^t = r(P^t) \quad (2.85)$$

$$m(a_i'^t) = a_i''^t = (x''^t, \sigma''^t) \quad (2.86)$$

$$\sigma''^t = \sigma'^t \exp N_o(\Delta\sigma) \quad (2.87)$$

$$x''^t = x'^t + N_0(\sigma''^t) \quad (2.88)$$

Com base nestas equações têm-se uma alteração do indivíduo e do seu desvio padrão associado. Desta forma não é necessária a regra de alteração do desvio padrão visto anteriormente.

Quanto aos operadores de recombinação r e seleção s , estes podem ser definidos por:

$$r(P^t) = a' = (x', \sigma') \in I, \vec{x}' \in \mathbb{R}^n, \vec{\sigma}' \in \mathbb{R}^n \quad (2.89)$$

$$x_i' = \frac{1}{2}(x_{a,i} + x_{b,i}), \forall i \in \{1, \dots, n\} \quad (2.90)$$

$$\sigma_i' = \frac{1}{2}(\sigma_{a,i} + \sigma_{b,i}), \forall i \in \{1, \dots, n\} \quad (2.91)$$

3. ANÁLISE COMPARATIVA ENTRE MODELOS DA INTELIGÊNCIA COMPUTACIONAL

Neste capítulo serão apresentadas as análises realizadas ao longo da pesquisa, que tiveram como foco identificar possíveis equivalências entre a Computação Neural e a Computação Evolutiva. A classe de problemas de otimização será avaliada em maiores detalhes, desencadeando em uma comparação direta entre a Máquina de Boltzmann e a Estratégia Evolutiva (1+1) – ES.

Para uma análise comparativa entre a Computação Neural (CN) e a Computação Evolutiva (CE), uma das questões a ser levada em consideração é a função que cada teoria se propõe a resolver (aproximação, classificação, agrupamento, otimização, etc.). Tal questão é de interesse para este trabalho, pois permite identificar funcionalidades comuns às duas áreas e desta forma iniciar uma análise detalhada partindo de uma semelhança já identificada em um nível de abstração geral (a rede neural ou algoritmo evolutivo vistos como um único módulo em forma de “caixa preta”). Uma vez identificada uma semelhança entre os módulos gerais, as análises poderão seguir com um procedimento de decomposição dos módulos principais em sub-módulos de menor complexidade.

Devido a restrições de tempo para o desenvolvimento deste trabalho, as análises comparativas entre a CN e a CE iniciarão com uma análise geral das funcionalidades comuns e com base nos resultados obtidos será então escolhido um estudo de caso específico para análise detalhada.

3.1 Análise Geral

Neste tópico a CN e CE serão analisadas sob uma ótica funcional, com o objetivo de identificar as utilizações comuns às duas áreas. Por se tratar de procedimento inicial, esta atividade irá realizar uma análise de forma abrangente, envolvendo os modelos de redes neurais e algoritmos evolutivos descritos no capítulo anterior, além de identificar algumas Classes de Utilização que as áreas são capazes de resolver. Com base nos resultados obtidos, será então escolhida uma Classe de Utilização de maior similaridade entre as diferentes áreas e um problema específico desta classe será analisada.

3.1.1 Visão Funcional

Atualmente pode-se encontrar na literatura uma grande diversidade de aplicações que fazem uso da CN e CE, sendo que estas aplicações são pertencentes a diversas áreas, como medicina [Walczak-2005], aeronáutica [Koper&Wood&Schneider-2004], robótica

[Lee&Shim&Han&Kim&Kim-2008], eletrônica, economia [Swales-1991], etc. Dada a grande variedade de aplicações existentes, como forma de viabilizar uma análise comparativa entre as diferentes áreas, estas aplicações serão agrupadas em um número reduzido de “Classes de Utilização”. Cada uma das “Classes de Utilização” representa uma finalidade de uso ou uma função dentro de uma aplicação. Como exemplo, a utilização de um Algoritmo Genético para a escolha de parâmetros de modelagem de aeronaves será considerada como um caso de otimização de aerodinâmica, ou seja, esta aplicação será considerada como pertencente a uma Classe de Utilização denominada “Otimização”. Com base neste conceito, algumas Classes de Utilização básicas podem ser sugeridas.

Computação neural	Computação Evolutiva
Aproximação de Funções	Otimização
Classificação	Outros
Agrupamento de Dados (<i>Clustering</i>)	
Memória associativa	
Otimização	
Reconhecimento de Padrões	
Outros	

Tabela 3.1 : Classes de utilização de Computação Neural e Computação Evolutiva.

Cabe ressaltar, que as Classes de Utilização sugeridas na Tabela 3.1 foram definidas de forma a auxiliar na busca de semelhanças entre as diferentes áreas. A avaliação se estas classes são ou não as mais adequadas para as análises pode dar abertura a uma discussão de caráter subjetivo e portanto não será realizado neste trabalho. Na tabela acima também é importante ressaltar a possibilidade de adição de novas classes de utilização em trabalhos futuros.

Com base nas Classes de Utilização listadas, será possível iniciar uma análise geral dos sistemas, com o objetivo de identificar os parâmetros de entradas e de saída dos modelos. As análises iniciais serão realizadas com base em definições existentes na literatura:

- Aproximação de Funções: Segundo [Sutton&Barto-1998] a aproximação de funções é um problema de predição, que inicia como a identificação de uma função V^π , criada a partir de experiências geradas por uma política π . Segundo o autor, a política visa gerar uma aproximação baseada em experiências mais prováveis de ocorrer e reduzir a influência de experiências menos prováveis. Para definição da “função de valor” V^π o autor sugere que esta seja baseada em uma “função de aproximação” representada por V_t , uma vez que a aproximação varia com o tempo t (as experiências são apresentadas para o sistema ao longo do tempo). Neste contexto, a função de aproximação é definida como uma função parametrizada em função de um vetor $\vec{\theta}_t$, de forma equivalente à função de aproximação resultante de uma rede neural Perceptron de Múltiplas Camadas (PMC) com algoritmo de aprendizado Backpropagation, onde os pesos sinápticos da rede são ajustados de acordo com o gradiente da função de erro (que equivale ao vetor $\vec{\theta}_t$). Segundo os autores, a aproximação de função faz como que um estado s desconhecido seja aproximado para um estado v , que o autor define como valor de backup, onde v é um estado gerado a partir das amostras de experiência com base na política π . Segundo as definições, as

experiências do sistema são dadas por s_t , $t=1,2,\dots,n$. Com base nesta definição, a aproximação de funções pode ser visualizada através de um módulo “caixa preta” como a apresentada na Figura 3.1:

Figura 3.1 - Diagrama de contexto para Aproximação de Funções.

- **Classificação:** Segundo [Aarts&Korst-1989, p.180] a Classificação pode ser vista como a divisão de elementos em conjuntos distintos denominados Classes. Levando em consideração as definições utilizadas pelo autor, um problema de classificação pode ser definido através de um par (O, S) , onde O denota um conjunto finito de objetos $O = \{o_1, \dots, o_n\}$ e S denota uma coleção de subconjuntos disjuntos $S_1, \dots, S_m$ pertencentes a O . Neste contexto, um problema de classificação terá como objetivo selecionar um dado objeto $o_i \in O$ como um membro de um dos subconjuntos $S_j \in S$. Com base nestes conceitos, as interfaces de entrada e saída do sistema podem ser definidas através do diagrama de contexto apresentado na Figura 3.2.

Figura 3.2 - Diagrama de contexto de sistemas de Reconhecimento de Padrões e Classificação.

- **Reconhecimento de Padrões:** Em [Verhagen-1975] é apresentada uma discussão sobre a definição precisa do termo Reconhecimento de Padrões. Segundo o autor [Verhagen-1975, p.111], as definições do termo “Reconhecimento” comumente se fundem com termos como “Classificação”, “Identificação”, “Análises”, etc. Analisando artigos diversos pode-se identificar esta mescla de conceitos: (i) [Sakataa&Kanekob&Kanekob&Takagic&Okudac-2005] utiliza termos como “Reconhecimento de Padrões” e “Classificação de Padrões” para descrever um mesmo

conceito; (ii) [Singh-1999] utiliza termos “Reconhecimento de Padrões” e “Casamento (*Matching*) de Padrões” com mesmo significado; (iii) [Kirbiz&Ulker&Gunsel-2007] descreve um modelo de Reconhecimento de Padrões como um sistema Classificador; (iv) [King&Azriel-1976] define o Reconhecimento de Padrões como um sistema cujo objetivo primário é de descrição e Classificação de medidas extraídas de processos; (v) [Bishop-2006-p.1] sustenta que a preocupação da área de Reconhecimento de Padrões é com a descoberta automática de regularidades em dados através de algoritmos computacionais e utilização destas regularidades para classificação. Segundo [Aarts&Korst-1989, p.179] os problema de Classificação são originário das áreas de Reconhecimento de Padrões e dadas as semelhanças semânticas entre o termo “Reconhecimento de Padrões” e “Classificação de Padrões” (segundo análise deste trabalho), o Reconhecimento de Padrão será considerado como um caso particular da Classificação, onde a saída do sistema resulta em um conjunto unitário composto por uma única classe S_1 , aquela que mais se aproxima, por algum critério de semelhança, com a classe do padrão original, conforme apresentado na Figura 3.3.

Figura 3.3 - Diagrama de contexto de sistemas de Reconhecimento de Padrões e Classificação.

- **Memória associativa:** Segundo [Haykin-1999, p.687], a memória associativa tem como objetivo a recuperação de um padrão armazenado dada a apresentação na entrada do sistema de um conjunto de informações completas, incompletas ou ruidosas de um determinado sinal. Segundo as definições apresentadas pelo autor, a memória associativa visa mapear uma memória fundamental $\vec{\xi}_\mu$ em um ponto fixo $\vec{x}_\mu$ do sistema dinâmico. Matematicamente este mapeamento é definido pelo autor através da relação $\vec{\xi}_\mu \Leftrightarrow \vec{x}_\mu$. Com base nesta definição, as interfaces do sistema podem ser visualizados através da Figura 3.4.

Figura 3.4 - Diagrama de contexto de sistemas de Memória Associativa.

- Agrupamento de Dados (*Clustering*): Segundo [Jain&Murty&Flynn-1999, pag. 264] Agrupamento de Dados (*Clustering*) é uma classificação de padrões em grupos (*clusters*) de forma não supervisionada. Analisando esta classe de utilização com base na definição citada pelo autor, o problema de Agrupamento de Dados pode ser generalizado como um problema de classificação, onde as Classes não são conhecidas ou definidas no instante inicial do problema. Dentre as teorias apresentadas ao longo deste trabalho, esta classe de utilização é resultante da operação dos Mapas Auto-Organizáveis, que segundo [Haykin-1999, p.446] tem como objetivo transformar um padrão de sinal de dimensão arbitrário, denotado por $\vec{x} = [x_1, x_2, \dots, x_m]^T$, em um mapa discreto de uma ou duas dimensões, realizando esta transformação adaptativamente. Ao analisar as definições acima, pode-se interpretar que o mapa citado por [Haykin-1999, p.446] representa as classes definidas por [Jain&Murty&Flynn-1999, pag. 264] através de uma mapa visual, de uma ou duas dimensões. Com base nestas definições uma interface para esta Classe de Utilização pode ser definida através da Figura 3.5.

Figura 3.5 Diagrama de contexto de sistemas de Agrupamento de Dados (*Clustering*).

- Otimização Combinatorial: Segundo a definição apresentada em [Aarts&Korst-1989, p.153] a otimização pode ser definida como uma tupla $(\mathcal{S}, \mathcal{S}', f)$, onde $\mathcal{S}$ denota um conjunto finito de soluções, $\mathcal{S}' \subseteq \mathcal{S}$ representa o conjunto de soluções consistentes para o

problema e a função $f : \mathcal{S} \rightarrow \mathbb{R}$ representa a função de custo que atribui um número real para cada solução. Com base nesta definição, o problema visa identificar uma solução consistente cuja função de custo f seja ótima. Algebricamente uma solução de um problema de otimização é representada pela variável $\mathcal{X}$. Desta forma, as interfaces do sistema podem ser definidas conforme apresentadas na Figura 3.6.

Figura 3.6 - Diagrama de contexto de sistemas de Otimização Combinatorial.

Uma vez definidas as principais aplicações de interesse deste trabalho, uma análise comparativa entre os modelos pertencentes à CN e a CE pode ser realizada.

Sob a ótica da CN, quatro redes neurais distintas foram avaliadas nos capítulos anteriores:

- Perceptrons de Múltiplas Camadas: Rede neural com estrutura de propagação de sinal sem realimentação. O algoritmo de aprendizado desta rede neural é o algoritmo de Backpropagation, que visa gerar um ajuste dos pesos sinápticos de forma que a rede neural aprenda por exemplos. Com base nesta estrutura é possível solucionar problemas relacionados com Reconhecimento de Padrões, Classificação e Aproximação de Funções.
- Redes de Kohonen: Arquitetura neural com propagação de sinal sem realimentação. O algoritmo de aprendizado aplicado a esta rede é o aprendizado competitivo. Com base nesta estrutura neural é possível solucionar problemas relacionados com Classificação, Agrupamento de Dados (*Clustering*) e Otimização.
- Redes de Hopfield: Rede neural realimentada com neurônios determinísticos. O aprendizado aplicado a esta rede é o aprendizado Hebbiano e o aprendizado baseado em memória. Este rede apresenta uma estrutura comumente utilizada para aplicações de Memória Associativa e Classificação, porém também sendo possível a resolução de problemas de Otimização.
- Redes de Boltzmann: Rede neural realimentada com neurônios estocásticos. Com base nesta rede neural tem-se uma estrutura capaz de resolver problemas de Otimização, Classificação e Memória Associativa.

Para a área da CE, dado o algoritmo centrado em evolução, tem-se intrinsecamente uma estrutura capaz de resolver os problemas de otimização.

Avaliando as aplicações que cada teoria se propõe a resolver, uma Classe de Utilização comum pode ser identificada para ambas as áreas. Esta intersecção pode ser visualizada na figura abaixo, onde são exibidos os escopos (considerados neste trabalho) da CN e da CE.

Figura 3.7 - Intersecção de aplicações para CN e CE.

As visões apresentadas acima tiveram como objetivo analisar os sistemas em seu maior nível de abstração, onde são representados como um único macro processo funcional. As análises dos sistemas podem também ser realizadas em diferentes níveis de abstração, onde o processo fundamental pode ser avaliado como uma composição de processos funcionais de menor complexidade. A princípio, as análises comparativas entre as diferentes teorias serão inicialmente focadas nos problemas comuns às duas áreas. Esta abordagem pode ser vista como uma heurística que visa buscar por semelhanças partindo de uma equivalência já constatada em alto nível de abstração. No procedimento de decomposição dos sistemas em sub-módulos funcionais, as teorias serão analisadas mais detalhadamente sendo as sub-funções comparadas entre si para identificação de equivalências adicionais.

3.1.2 Visão Estrutural

A análise estrutural tem como objetivo possibilitar a identificação das principais entidades ou classes que definem os modelos. Em essência, este paradigma de visualização se assemelha ao paradigma Orientado a Objetos, comumente adotado na Engenharia de Software [Steidley-1993].

De forma análoga à análise funcional, a visão estrutural também pode ser visualizada segundo diferentes níveis de abstração, iniciando com uma visão de contexto e posteriormente identificando sub-módulos estruturais que o compõe. Esta visualização dos modelos será utilizada para análise uma vez que a estrutura de dados pode ser vista como um paradigma mais estável quando comparada à visão funcional, de forma equivalente à orientação a objetos [Yang&Butler-1998, pag.104] [Steidley-1993, p. 133].

Para o problema de interesse deste trabalho, ao analisar o que seria uma visão de contexto sob a ótica estrutural, pode-se identificar essencialmente uma classe que será definida como “Sistema”. Abaixo deste nível duas outras entidades podem ser definidas como comuns às duas áreas, uma classe “Algoritmo” e outra “Modelo”, que representa o modelo auxiliar

manipulado pelo algoritmo (essencialmente uma classe Neurônio para a Rede Neural, ou uma classe/estrutura qualquer manipulada pelos Algoritmos Evolutivos). Uma representação gráfica desta decomposição é apresentada na figura abaixo.

Figura 3.8 - Estrutura de classes da CN e CE.

Cabe ressaltar que as definições do sistema com base nas classes sugeridas acima representam uma dentre várias interpretações de classes possíveis. Em essência, a avaliação se estas classes são ou não as mais adequadas para análise do problema pode dar abertura a uma discussão de caráter subjetivo, portanto não sendo realizado neste trabalho. A estrutura de classes acima será utilizada como base para as próximas análises, pois permite uma visualização do sistema em 4 categorias distintas, dois níveis inicialmente conhecidos e dois níveis desconhecidos, que deverão ser classificados como “Classes com Equivalência” e “Classes sem Equivalência” entre as diferentes áreas.

Sob a ótica da CN, com base em uma análise das diferentes redes é possível identificar a existência de uma grande variedade de Algoritmos e propriedades, porém todas sendo baseadas na utilização de uma única entidade auxiliar, que é o próprio Neurônio Artificial. Para os Algoritmos Evolutivos, é possível identificar um cenário inverso, onde a entidade Algoritmo é bem definida enquanto que os Modelos manipulados pelo Algoritmo são flexíveis, uma vez que estruturas diversas podem ser mapeadas na forma de genes. Uma representação destes conceitos pode ser vista na figura abaixo.

Figura 3.9 - Estrutura dos sistemas neurais e evolutivos, segundo dois níveis distintos de abstração.

Com base nestas descrições, se for possível identificar um sistema neural com um algoritmo baseado nos operadores evolutivos, ou se for possível identificar um algoritmo evolutivo que manipula estruturas neurais, conseqüentemente será identificando um ponto comum de intersecção estrutural entre a CN e a CE. Uma visão desta intersecção é apresentada na figura abaixo. Os modelos que pertencerem a este ponto de intersecção serão modelos candidatos para análise de semelhanças e para avaliação sobre a possibilidade de conversão entre os domínios das diferentes áreas.

Figura 3.10 - Intersecção estrutural.

3.2 Análise Detalhada

Conforme citado anteriormente, a Classe de Utilização comum às duas áreas é a classe de Otimização (dentre as classes listadas na Tabela 3.1). Esta classe de problemas inclui o Problema do Caixeiro Viajante, que pode ser considerado um clássico da literatura conforme citado em [Haykin-199, p.723] e [Chatterjee&Carrera&Lynch-1996]. Dada a relevância deste problema no estudo aplicado à otimização, as análises comparativas entre a CN e a CE serão realizadas sobre uma aplicação prática deste problema.

3.2.1 Aplicação: Problema do Caixeiro Viajante

Dada a posição de um número de cidades distintas, o problema se resume em achar um menor percurso passando por cada cidade uma única vez e retornar ao ponto de partida inicial. O problema é comumente abordado como um caso de uso prático de Otimização Combinatorial do tipo NP completo dada a inexistência de método viável para achar a(s) solução(ões) ótima(s) do problema para um grande número de cidades.

Em geral os estudos utilizam como base o problema com distâncias simétricas, onde a distância entre duas cidades é a mesma independente do sentido do deslocamento. O problema do caixeiro viajante assimétrico é um caso onde a distância do percurso entre duas cidades i e

j possui valores distintos caso o sentido de deslocamento seja da cidade i para j ao invés de j para i . Para o problema a ser analisado neste trabalho, será abordado o Problema do Caixeiro Viajante com distâncias simétricas.

3.2.2 Análise Funcional

No contexto da CN, para a resolução do Problema do Caixeiro Viajante foram selecionadas da literatura soluções baseadas nos modelos de redes de Hopfield [Fausett-1993], Máquina de Boltzmann [Fausett-1993] [Aarts&Korst-1989] e rede de Kohonen [Fausett-1993].

Para as soluções baseadas nos Algoritmos Evolutivos é possível encontrar na literatura soluções baseadas nas três sub-áreas abordadas por este trabalho, conforme pode-se ver em [Cho&Oh&Choi-1998], [Wang&Qin-2007] e [Shengwu&Chengjun-2002] .

Nos próximos tópicos serão analisados cada um dos modelos da CN utilizados para resolução do Problema do Caixeiro Viajante, assim como serão analisados os Algoritmos Evolutivos de forma genérica, uma vez que todos são baseados nos mesmos operadores evolutivos básicos.

3.2.2.1 Redes neurais: Interfaces

Para a resolução do Problema do Caixeiro Viajante, a rede de Hopfield [Fausett-1993, p.349] e Máquina de Boltzmann [Fausett-1993, p.338] utilizam uma mesma arquitetura neural assim como utilizam uma mesma interface de representação das soluções do problema. Para a rede de Kohonen [Fausett-1993, p.182] é utilizada uma abordagem diferenciada em relação às duas primeiras redes neurais, tanto sob uma ótica de arquitetura quanto para representação das soluções do problema. Desta forma, a alteração da rede de Kohonen para utilização de uma interface comum às redes de Hopfield e Boltzmann deverá ser analisada.

Segundo as soluções baseadas na rede de Hopfield e máquina de Boltzmann, para o problema do caixeiro viajante com c cidades, serão utilizadas $m = c^2$ unidades neurais representando uma matriz de $c \times c$ sinais de saída. Nesta arquitetura, um percurso válido é representado através de uma única unidade ativada em cada linha e em cada coluna. Duas unidades ativadas em uma mesma linha indicam que uma cidade foi visitada duas vezes e duas unidades ativadas em uma coluna mostram que o viajante esteve em duas cidades ao mesmo tempo, sendo ambas as situações exemplos de soluções inconsistentes para o problema proposto. Uma visão desta estrutura de representação pode ser vista através da Figura abaixo.

Ordem Cidades	1	2	...	n
A (cidade 0)	$x_{0,0}$	$x_{0,1}$	...	$x_{0,c-1}$
B (cidade 1)	$x_{0,1}$	$x_{1,1}$	...	$x_{1,c-1}$
...	...	...	...	...
X (cidade $c-1$)	$x_{c-1,0}$	$x_{c-1,2}$	...	$x_{c-1,c-1}$

Figura 3.11 Arquitetura neural para problema do caixeiro viajante com c cidades.

Com base nesta arquitetura em forma de matriz, as redes de Hopfield e Máquina de Boltzmann são modeladas de forma que todas as unidades de cada linha sejam interconectadas entre si, o mesmo ocorrendo para todas as unidades pertencentes a uma mesma coluna. Os pesos sinápticos das conexões são então definidos de forma que unidades na mesma linha, assim como as unidades de uma mesma coluna, tenham tendência de não serem ativadas simultaneamente. Para a representação das distâncias entre as cidades a arquitetura define conexões entre as unidades de colunas adjacentes e entre a primeira e última coluna da matriz, sendo o peso sináptico destas conexões definidas em função das distâncias entre as cidades do problema.

Para a resolução do problema utilizando rede de Kohonen, outra arquitetura neural é utilizada, onde são utilizados dois neurônios na camada de entrada e uma quantidade de neurônios $m_{Kohonen}$ superior ao número de cidades c para a camada de saída (vide tópico 2.2.5.2). Nesta modelagem da rede neural a representação das soluções está associada aos pesos sinápticos que ligam os neurônios da camada de entrada com os neurônios da camada de saída, representação diferenciada das utilizadas pelas redes de Hopfield e Boltzmann. Outra característica das soluções na rede de Kohonen é o caráter subjetivo dos resultados, que em algumas situações podem dar margem a interpretações distintas de soluções, o que não é desejável uma vez que busca-se uma unificação das representações para os diferentes modelos. Uma forma de compatibilizar as interfaces de saída entre as redes neurais seria de considerar a saída da rede de Kohonen após aplicação de uma função de interpretação $f_{Interpretacao} : D_{Kohonen} \rightarrow D_{Hopfield/Boltzmann}$ responsável pela conversão dos sinais ambíguos de saída do domínio $D_{Kohonen}$ para sinais precisos pertencentes ao domínio $D_{Hopfield/Boltzmann}$. Com esta abordagem é possível definir uma interface de saída comum para as teorias, atividade que não será realizada neste trabalho.

Uma vez analisadas as representações das soluções (que representam os sinais de saída), será avaliada a compatibilização das interfaces de entrada de dados do problema. Neste contexto, as soluções baseadas em Hopfield / Boltzmann processam as informações de conexões entre cidades e os custos de utilização dos caminhos, enquanto que as soluções baseada em Kohonen utilizam como sinal de entrada as coordenadas das cidades no mapa. Para o algoritmo executado pela rede de Kohonen, tem-se a impossibilidade de representação de restrições, como caminhos inexistentes entre cidades, além da distância ser uma medida definida no espaço euclidiano bidimensional do mapa. Para as soluções utilizando as redes de Hopfield / Boltzmann é possível levar em consideração tanto a inexistência de conexões entre

idades quanto é possível utilizar um conceito de distância baseado no custo para utilização de um caminho, o que permite que existam problemas assimétricos (custo de uma trajetória de uma cidade i para j é considerado diferente da trajetória de j para i). Uma vez que as análises visam unificar as interfaces entre os diferentes modelos da CN e da CE, esta incompatibilidade entre os sinais de entrada das diferentes redes neurais não é desejada.

Devido às diferenças nos sinais de entrada das redes Kohonen e Hopfield / Boltzmann e a falta de um sinal de saída adequado para a rede de Kohonen, serão consideradas para definição das interfaces apenas as redes de Hopfield e a Máquina de Boltzmann.

3.2.2.2 Algoritmos Evolutivos: Interfaces

Para os Algoritmos Evolutivos, diversas representações podem ser utilizadas para representação das soluções do Problema do Caixeiro Viajante. Em [Carballido&Ponzoni&Brignole-03] é apresentada uma abordagem ao problema através de duas representações distintas, uma representação ordenada dos caminhos utilizados e uma representação codificada das soluções. Outra representação possível é a listagem ordenada das cidades visitadas conforme se vê em [Bakhouya&Gaber-2007].

Dada a possibilidade de diversas representações de soluções nos Algoritmos Evolutivos, por simplicidade serão consideradas nas análises apenas as representações em forma de lista ordenada das cidades visitadas e lista ordenada dos caminhos seguidos pelo viajante. Outras representações para o problema não serão avaliados.

Para a representação (a) ordenada das cidades visitadas, tem-se uma interface diretamente compatível com a representação utilizada pelas redes neurais. Para a representação em função dos (b) caminhos percorridos tem-se uma forma de representação com interface não compatível, porém sendo possível uma conversão entre as duas representações.

Uma conversão entre as diferentes representações é possível uma vez que pode-se definir uma função bijetora que mapeia todos os elementos da solução (a) em elementos pertencentes à representação (b). Se for considerada uma solução ordenada de visitas $S_i = \{A, B, \dots, N\}$, onde A, B e N são cidades pertencentes a um problema com $c > 2$ cidades, então representar este percurso pode ser representado através da notação $\overline{S}_i = \{(A, B), (B, \dots), (\dots, N)\}$ uma vez que duas cidades visitadas implicarão em um caminho existente entre elas, ou seja, pode-se representar uma visita a duas cidades A, B como sendo uma aresta definida como $C_{A,B} = (A, B)$ e vice-versa. Dada esta equivalência e levando em consideração uma unificação das representações com as já utilizadas pelos modelos da CN, a representação ordenada de visitas será utilizada como representação base do problema.

Para representação dos sinais de entrada dos sistemas evolutivos, são necessárias as informações de distâncias (ou custo) entre as cidades e as conexões existentes entre estas.

Desta forma tem-se uma compatibilidade dos sinais de entrada entre a CE e as redes de Hopfield e Máquina de Boltzmann.

3.2.2.3 Redes Neurais vs. Algoritmos Evolutivos: Interfaces

Para o problema de otimização, cabe lembrar que este foi definido como uma tupla $(\mathcal{S}, \mathcal{S}', f)$, onde $\mathcal{S}$ denota um conjunto finito de soluções, $\mathcal{S}' \subseteq \mathcal{S}$ representa o conjunto de soluções consistentes para o problema e a função $f : \mathcal{S} \rightarrow \mathbb{R}$ representa a função de custo que atribui um número real para cada solução consistente. Cabe ressaltar que esta é uma definição matemática para um problema de otimização e não necessariamente uma interface a ser utilizada em aplicações práticas. Tal observação é importante, pois em um problema de otimização pode-se ter uma grande quantidade de soluções possíveis, o que poderia inviabilizar listagem de todas as soluções $\mathcal{S}$ e $\mathcal{S}'$ de forma explícita. Desta forma, para o problema do caixeiro viajante, as interfaces serão definidas levando em consideração aplicações práticas e não utilizando a interface teórica citada acima.

Antes de prosseguir com as análises, cabe lembrar que para a resolução do Problema do Caixeiro Viajante utilizando a rede de Kohonen, os sinais de entrada exigem informações sobre as coordenadas das cidades no mapa bidimensional do problema, informações diferentes das necessárias pelos algoritmos evolutivos e pelas redes de Hopfield e Boltzmann. Quanto aos sinais de saída, estes geram uma solução imprecisa, além de não possuir a capacidade de levar em consideração restrições do problema (cidades não conectadas) e levar em consideração a existência de custos assimétricos dos caminhos (as distâncias são baseadas na distância euclidiana e não há como representar distâncias distintas para trajetórias de i para j e j para i). Dadas estas características, esta rede neural não será considerada nas análises seguintes.

Quanto à representação das soluções, se for utilizada uma lista ordenada de cidades visitadas, uma forma de manter uma compatibilidade entre a CN e a CE será através das representações das visitas em forma binária, uma vez que as redes de Hopfield e Boltzmann já utilizam esta representação. Para o problema com c cidades, tem-se uma matriz de $c \times c$ saídas binárias, onde a π -ésima cidade visitada no tempo i do percurso é representada pela variável $x_{\pi i} = 1$. Cabe lembrar que nesta representação, um percurso válido é representado através de uma única unidade ativada em cada linha e em cada coluna. Duas unidades ativadas em uma mesma linha indicam que uma cidade foi visitada duas vezes e duas unidades ativadas em uma coluna mostram que o viajante esteve em duas cidades ao mesmo tempo, sendo ambas representam soluções inconsistentes. A representação do problema em forma de matriz é apresentada na Figura 3.12.

Figura 3.12 Representação unificada adotada para problema do caixeiro viajante com $c > 2$ cidades.

Com a adoção desta representação pode-se definir uma solução para o Problema do Caixeiro Viajante através da seguinte notação:

$$\bar{S} = \left\{ \{x_{0,0}, \dots, x_{0,c-1}\}, \dots, \{x_{c-1,0}, \dots, x_{c-1,c-1}\} \right\} \quad (3.1)$$

Onde a solução $\bar{x}$ é representada como um conjunto de linhas ordenadas da matriz.

Com base nesta definição, tem-se uma interface apropriada para representação das soluções do problema, que representam a interface de saída dos modelos.

Referente às interfaces de entrada, conforme evidenciado anteriormente, os três modelos necessitam das informações relacionadas com as conexões entre cidades e os pesos que estas possuem. Com base nestas semelhanças, para definição de uma interface de entrada comum, será utilizada como informação de entrada uma lista $\vec{C} = (C_1, \dots, C_m)$ de m caminhos $C_i = (\text{origem}, \text{destino}, D)$, que mapeiam cidades de “origem” e “destino” associados a um valor escalar $D \in \mathbb{R}$ de distância (ou custo). Estas informações deverão então ser processadas pelos sistemas para que possam ser gerados os resultados de saída. Pode-se então definir uma interface para o problema do caixeiro viajante como apresentado na figura abaixo.

Figura 3.13 - Interface comum para CN e CE visando resolver o problema do caixeiro viajante.

3.2.2.4 Redes neurais: Algoritmos

Neste tópico serão descritas as estruturas básicas dos algoritmos executados pela rede neural de Hopfield e da Máquina de Boltzmann para problemas de otimização. A intenção desta análise é de identificação dos principais sub-funções que compõe os algoritmos. Neste nível de abstração, os algoritmos serão analisados sem considerar as amarrações ao Problema do Caixeiro Viajante, ou seja, as análises para problemas de otimização serão mantidas genéricas.

Para a rede de Hopfield uma versão simplificada do algoritmo apresentado no tópico 2.2.6.2 é descrito:

Entradas:

$\Theta_{finalização}$ parâmetros genéricos para identificação da condição de parada.

m número de unidades neurais utilizadas na rede

Saídas:

$\vec{x}^*$ o melhor estado neural encontrado durante a execução do algoritmo

```

1      Inicialização de todas as unidades de acordo com o problema.
2      while (finalizar ( $\Theta_{finalização}$ )  $\neq$  true) do
3          while (Executar  $m$  vezes) do
4              Escolher uma unidade  $u_{\pi,i}$  aleatoriamente.
5              Alteração do sinal interno do neurônio  $v_{\pi,i}$  (novo).
6              Aplicar a função de saída  $x_{\pi,i}$ .
7          end while
8      end while

```

Analisando o funcionamento da Máquina de Boltzmann sequencial, pode-se descrever um algoritmo base para a resolução de problemas de Otimização:

Entradas:

$\Theta_{finalização}$ parâmetros genéricos para a função de finalização

$\Theta_{alteração}$ parâmetros genéricos para a função de alteração de estado

Saídas:

$\vec{x}^*$ o melhor estado neural encontrado durante a execução do algoritmo

```

1      inicializarRedeNeural ();
2      t  $\leftarrow$  0;
3       $\vec{x}(t) \leftarrow$  inicializar();
4      C( $\vec{x}(t)$ )  $\leftarrow$  avaliar( $\vec{x}(t)$ );
5      while (finalizar( $\Theta_{finalização}$ )  $\neq$  true) do

```

```

6       $\vec{x}^{\{u, \dots\}}(t) \leftarrow \text{calcularNovoEstado}(\vec{x}(t), \Theta_{\text{alteração}});$ 
7       $C(\vec{x}^{\{u, \dots\}}(t)) \leftarrow \text{avaliar}(\vec{x}^{\{u, \dots\}}(t));$ 
8       $\vec{x}(t+1) \leftarrow \text{selecionar}(\vec{x}(t), \vec{x}^{\{u, \dots\}}(t), C(\vec{x}(t)), C(\vec{x}^{\{u, \dots\}}(t)));$ 
9       $t \leftarrow t+1;$ 
10     end while

```

Onde $C: \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ é a função de consenso da Máquina de Boltzmann, $\vec{x}^{\{u_i, \dots\}}$ representa um estado $\vec{x}$ com os estados dos neurônios $\{u, \dots, v\}$ invertidos e t representa um contador de iterações do algoritmo.

3.2.2.5 Algoritmos Evolutivos: Algoritmos

Dada a similaridade entre as diferentes sub-áreas da CE, será utilizada nas análises o algoritmo genérico descrito em [Jong&Lawrence&Schwefel-1997, p. B1.1:2]:

Entradas:

- μ número de pais da população
- λ número de descendentes da população
- $\Theta_{\text{finalização}}$ parâmetros genéricos para função de finalização
- $\Theta_{\text{recombinação}}$ parâmetros genéricos para operador de recombinação
- $\Theta_{\text{mutação}}$ parâmetros genéricos para operador de mutação
- $\Theta_{\text{seleção}}$ parâmetros genéricos para operador de seleção

Saídas:

- a^* , a melhor solução encontrada na execução do algoritmo
- P^* , a melhor população de soluções encontrada na execução do algoritmo

```

1       $t \leftarrow 0;$ 
2       $P(t) \leftarrow \text{inicializar}(\mu);$ 
3       $F(t) \leftarrow \text{avaliar}(P(t), \mu);$ 
4      while (finalizar( $P(t)$ ,  $\Theta_{\text{finalização}}$ )  $\neq$  true) do
5           $P'(t) \leftarrow \text{recombinar}(P(t), \Theta_{\text{recombinação}});$ 
6           $P''(t) \leftarrow \text{mutacionar}(P'(t), \Theta_{\text{mutação}});$ 
7           $F(t) \leftarrow \text{avaliar}(P''(t), \lambda);$ 
8           $P(t+1) \leftarrow \text{selecionar}(P''(t), F(t), \mu, \Theta_{\text{seleção}});$ 
9           $t \leftarrow t+1;$ 
10     end while

```

Onde t representa um contador de iterações do algoritmo.

3.2.2.6 Redes Neurais vs. Algoritmos Evolutivos: Algoritmos

Considerando o algoritmo base descrito para as redes de Hopfield tem-se que este apresenta um ciclo simples de seleção randômica de neurônios, cálculo de seu sinal interno $v_{\pi,i}$ e cálculo de sua saída $x_{\pi,i}$. Este é um algoritmo básico de operação das redes neurais, que em si não é o responsável pela resolução dos problemas de Otimização. No contexto da rede de Hopfield a capacidade de resolução do problema fica então associada à função de inicialização da rede e à implementação de uma função diferenciada de cálculo do sinal interno dos neurônios (que pode ser consultado no tópico 2.2.6.2). De forma abrangente, pode-se identificar neste algoritmo uma estrutura básica composta pelas seguintes secções:

- 1) Cabeçalho: utilizado para identificar as entradas e saídas dos algoritmos (sem considerar parâmetros específicos dos problemas).
- 2) Inicialização: utilizado para inicialização da rede neural.
- 3) Ciclo: utilizado para determinar o instante de interrupção do algoritmo.
- 4) Propagação de sinal: utilizado para recalcular o sinal de saída de uma unidade da rede.

Comparando a rede de Hopfield com os algoritmos da Máquina de Boltzmann e dos Algoritmos Evolutivos é possível identificar apenas uma semelhança no nível de cabeçalho e no controle de ciclo.

Considerando uma análise comparativa envolvendo unicamente a Máquina de Boltzmann e o algoritmo base da CE, é possível identificar uma estrutura com maior semelhança do que a identificada levando em consideração os três modelos em conjunto. Comparando os dois algoritmos diretamente (da Máquina de Boltzmann e dos Algoritmos Evolutivos) é possível identificar uma estrutura básica semelhante composta pelas seguintes secções:

- 1) Cabeçalho: utilizado para identificar as entradas e saídas dos algoritmos.
- 2) Inicialização: utilizado para criação e avaliação de uma solução inicial.
- 3) Ciclo: utilizado para determinar o instante de interrupção do algoritmo.
- 4) Busca: processo de busca de uma nova solução no espaço de versões de soluções consistentes do problema.
- 5) Competição: procedimento utilizado para seleção de uma solução ou um conjunto de soluções para uma próxima iteração do algoritmo.

Uma comparação entre os algoritmos da Máquina de Boltzmann e os Algoritmos Evolutivos genéricos é apresentada na Tabela abaixo.

	Máquina de Boltzmann	Algoritmos Evolutivos
CABEÇALHO	Entradas: $\Theta_{finalização}$, $\Theta_{alteração}$ Saídas: $\bar{x}^*$	Entradas: μ , λ , $\Theta_{finalização}$, $\Theta_{recombinação}$, $\Theta_{mutação}$, $\Theta_{seleção}$ Saídas: a^* ou P^*
INICIALIZAÇÃO	1 inicializarRedeNeural(); 2 $t \leftarrow 0$; 3 $\bar{x}(t) \leftarrow \text{inicializar}()$; 4 $C(\bar{x}(t)) \leftarrow \text{avaliar}(\bar{x}(t))$;	1 $t \leftarrow 0$; 2 $P(t) \leftarrow \text{inicializar}(\mu)$; 3 $F(t) \leftarrow \text{avaliar}(P(t), \mu)$;
CICLO	5 while (finalizar(Θ_{fin}) $\neq$ true) do	4 while (finalizar ($P(t)$, Θ_{fin}) $\neq$ true) do
BUSCA	6 $\bar{x}^u(t)$ $\leftarrow \text{calcularNovoEstado}(\bar{x}(t), \Theta_{alt})$;	5 $P'(t) \leftarrow \text{recombinar}(P(t), \Theta_{rec})$; 6 $P''(t) \leftarrow \text{mutacionar}(P'(t), \Theta_{mut})$;
COMPETIÇÃO	7 $C(\bar{x}^u(t)) \leftarrow \text{avaliar}(\bar{x}^u(t))$; 8 $\bar{x}(t+1) \leftarrow \text{selecionar}(\bar{x}(t), \bar{x}^u(t), C(\bar{x}(t)), C(\bar{x}^u(t)))$;	7 $F(t) \leftarrow \text{avaliar}(P''(t), \lambda)$; 8 $P(t+1) \leftarrow \text{selecionar}(P''(t), F(t), \mu, \Theta_{sel})$;
CICLO	9 $t \leftarrow t+1$; 10 end while	9 $t \leftarrow t+1$; 10 end while

Tabela 3.2 : Comparativo entre algoritmos da Máquina de Boltzmann e Algoritmos Evolutivos.

Dada a semelhança entre a estrutura dos algoritmos da Máquina de Boltzmann e os Algoritmos Evolutivos, as análises seguintes irão focar nestes dois modelos apenas.

Cabeçalho

Uma vez que os estados da Máquina de Boltzmann e os indivíduos dos Algoritmos Evolutivos representam soluções de um problema de Otimização, as saídas destes modelos podem ser vistos como soluções binárias ótimas $\bar{S}^*$, e não como indivíduos e estados de rede. Esta percepção nos ajuda a identificar uma mesma natureza dos algoritmos, i.e. o uso para problemas de otimização com soluções binárias.

Outra diferença entre os algoritmos nesta camada, está na cardinalidade dos resultados retornados, onde a Máquina de Boltzmann retorna uma única solução enquanto que o Algoritmo Evolutivo genérico retorna um conjunto de soluções. Neste contexto, o retorno de uma única solução pelo Algoritmo Evolutivo pode ser considerado um caso específico desta

área que se assemelha à cardinalidade da Máquina de Boltzmann. Neste caso a Máquina de Boltzmann pode ser considerado similar aos Algoritmos Evolutivos com único retorno.

Considerando as entradas, ambos os algoritmos descrevem uma forma genérica de passar os parâmetros auxiliares utilizados pelas funções. Desta forma, podem-se generalizar estas entradas auxiliares através de um parâmetro genérico $\Theta_{controle}$.

Como resultado destes conceitos, pode-se visualizar a Máquina de Boltzmann para Otimização e um Algoritmo Evolutivo com saídas únicas como semelhantes. Uma representação desta semelhança pode ser vista na Figura abaixo:

Figura 3.14 Comparação dos algoritmos genéricos da Máquina de Boltzmann para Otimização e Algoritmos Evolutivos binários com retorno único.

Inicialização

Para o processo de inicialização, a Máquina de Boltzmann inicializa uma única solução candidata, enquanto que os Algoritmos Evolutivos inicializam um conjunto de soluções. Se for considerado um caso específico de Algoritmos Evolutivos com $\mu=1$ pais, então o número de soluções candidatas para ambos os modelos serão os mesmos.

Referente à função de avaliação, mesmo que a implementação possa ser diferente entre os modelos, cabe ressaltar que em ambos os casos as funções são utilizadas com mesmo propósito, i. e. de identificação de um valor de aptidão para a solução candidata. Também considerando um caso específico dos Algoritmos Evolutivos com $\mu=1$ pais, o número de avaliações retornadas pelas funções será o mesmo.

Com base nestas considerações, as funções de inicialização e de avaliação da Máquina de Boltzmann podem ser considerada semelhantes às mesmas funções dos Algoritmos Evolutivos binários com $\mu=1$ pais.

A representação deste processo genérico de inicialização pode ser vista na Figura abaixo.

Figura 3.15 Comparação dos procedimentos de Inicialização da Máquina de Boltzmann para Otimização e dos Algoritmos Evolutivos binários com único pai.

Ciclo

Um dos conceitos básicos por trás dos Algoritmos Evolutivos é a existência das Gerações de indivíduos. Para cada Geração, o mesmo processo de Recombinação, Mutação, Avaliação e Seleção são executados. Para uma Máquina de Boltzmann, a existência da realimentação na arquitetura da rede indica que as saídas dos neurônios também são recalculadas por ciclo. Com base nestes dois conceitos, os algoritmos utilizam um procedimento cíclico controlado por uma função de identificação do instante de finalização dos algoritmos. Mesmo que as funções tenham implementações distintas, eles podem ser baseados em conceitos comuns, como número máximo de iterações, limites de tempo de execução, etc. Desta forma, a função de finalização de ambos os modelos podem ser considerados similares.

Uma visão geral deste procedimento de controle de iterações é exibida na Figura abaixo.

Figura 3.16 Comparação dos procedimentos de controle de Ciclo da Máquina de Boltzmann para Otimização e dos Algoritmos Evolutivos genéricos.

Busca

O procedimento de Busca visa procurar por soluções no espaço de versões de soluções possíveis de um problema. Para os Algoritmos Evolutivos este procedimento depende de dois operadores principais, um operador de Recombinação e um operador de Mutação. Para a Máquina de Boltzmann a busca depende de uma única função, que é responsável pela inversão probabilística de bits da solução. Se for considerado um caso específico de soluções binárias nos Algoritmos Evolutivos, então operador de mutação executará basicamente o mesmo procedimento da função `calcularNovoEstado(.)` da Máquina de Boltzmann.

Também considerando um caso específico dos Algoritmos Evolutivos com $\mu=1$ pais e $\lambda=1$ descendentes, então o operador de recombinação não será mais utilizado pelo algoritmo (uma vez que este operador necessita de dois pais) e conseqüentemente a criação de uma nova solução nas Máquinas de Boltzmann e nos Algoritmos Evolutivos será baseado em uma única função e esta será equivalente, ou seja, será utilizada para inversão probabilística de bits.

Se for considerado um caso específico dos Algoritmos Evolutivos com $\lambda=1$ descendentes, então o retorno do operador de mutação terá a mesma cardinalidade que a função `calcularNovoEstado(.)` da Máquina de Boltzmann.

Com base nestas considerações, a criação de novas soluções na Máquina de Boltzmann pode ser considerada similar a um caso específico de Algoritmo Evolutivo binário com $\mu=1$ pais e $\lambda=1$ descendentes.

Uma visão geral do procedimento de busca é exibida na Figura abaixo.

Figura 3.17 Comparação dos procedimentos de controle de Ciclo da Máquina de Boltzmann para Otimização e dos Algoritmos Evolutivos genéricos.

Competição

Nos Algoritmos Evolutivos, o processo competitivo é utilizado para seleção de um conjunto de melhores soluções enquanto que para a Máquina de Boltzmann este processo é utilizado para seleção de uma única solução. Para ambos os modelos, o processo competitivo é baseado nas funções de avaliação e de seleção. Como citado anteriormente, a função de avaliação identifica um valor de aptidão para cada solução candidata, enquanto que a função de seleção é utilizada para selecionar uma ou mais soluções para o próximo ciclo do algoritmo.

Se for considerado um caso específico de Algoritmo Evolutivo com $\mu=1$ pais, então o retorno da função de avaliação terá a mesma cardinalidade para ambos os modelos.

Se for considerado um caso específico de Algoritmo Evolutivo com $\lambda=1$ descendentes, então o retorno da função de seleção terá a mesma cardinalidade para ambos os modelos.

Com base nestes conceitos, o processo competitivo da Máquina de Boltzmann pode ser considerado similar ao caso específico de um Algoritmo Evolutivo binário com $\mu=1$ pais e $\lambda=1$ descendentes.

Uma visão geral do procedimento de Competição é exibida na Figura abaixo.

Figura 3.18 Comparação dos procedimentos de controle de Ciclo da Máquina de Boltzmann para Otimização e dos Algoritmos Evolutivos genéricos.

3.2.3 Análise Estrutural

Para o procedimento de busca por semelhanças entre as duas teorias, as análises seguintes terão como objetivo expandir a árvore estrutural inicialmente identificada como composta por duas entidades distintas, o Modelo Auxiliar de representação e a entidade Algoritmo.

3.2.3.1 Computação Neural

Neste tópico serão analisadas as estruturas de classes dos modelos de forma mais detalhada (segundo visão própria deste trabalho). Neste contexto todas as redes neurais descritas ao longo deste trabalho serão analisadas.

3.2.3.1.1 Perceptrons de Múltiplas Camadas

Referente à entidade Algoritmo, cabe lembrar que os Perceptrons de Múltiplas Camadas (PMCs) possuem dois modos de operação distintos, desta forma identificando duas entidades abaixo do Algoritmo. Em um dos modos de operação a rede neural opera através de um algoritmo de Aprendizado, onde são realizados ajustes nos parâmetros livres da rede (Pesos Sinápticos e *Bias*). No segundo modo de operação, o modo Execução, a única função executada pela rede é a de propagação de sinal.

Para o modo de Aprendizado, a rede realiza um procedimento de Minimização da função de erro através de um método de Gradiente Descendente, algoritmo que é denominado Backpropagation.

Abaixo da entidade Modelo, são descritas as estruturas principais das redes neurais, ou seja, as classes Neurônio Determinístico, Pesos Sinápticos, Bias e a própria entidade Rede Neural.

Uma visão expandida das classes principais que compõe a rede PMC pode ser vista na figura abaixo.

Figura 3.19 - Estrutura expandida da rede Perceptron de Múltiplas Camadas.

3.2.3.1.2 Rede de Kohonen

Esta rede neural é utilizada tanto para a resolução de problemas de classificação quanto de otimização, utilizando para isso modos de operação para Aprendizado e Execução, ambos baseados no conceito de Competição entre as unidades de saída. Para o processo de Competição foram então destacadas duas entidades consideradas relevantes, as entidades Avaliação e Seleção.

Abaixo da entidade Modelo, serão descritas as estruturas principais das redes neurais, ou seja, as classes Neurônio Determinístico, Pesos Sinápticos, Bias e a própria entidade Rede Neural.

Uma visão expandida das classes principais que compõe a rede de Kohonen pode ser vista na figura abaixo.

Figura 3.20 - Estrutura expandida da rede de Kohonen.

3.2.3.1.3 Rede de Hopfield

Lembrando que a rede de Hopfield é utilizada para resolver problemas de Otimização e de Memória Associativa, tem-se que a entidade Algoritmo é baseada em dois modos de operação distintos, um modo de Aprendizado e um modo de Execução. Para as diferentes aplicações, diferentes modos de Aprendizado são utilizados, onde o modo de Aprendizado Dinâmico (utilizado pela Memória Associativa), define os pesos sinápticos através de um procedimento cíclico de apresentação de exemplos de treinamento, e o modo de Aprendizado Fixo (utilizado para Otimização) define os pesos sinápticos sem um procedimento genérico, devendo ser analisado especificamente para cada cenário ou problema.

De forma equivalente às outras redes neurais descritas anteriormente, abaixo da entidade Modelo serão descritas as estruturas principais das redes neurais, desta vez considerando também duas novas entidades, as entidades Estado da Rede e Realimentação, comumente referenciadas e manipuladas pelos algoritmos.

Uma visão expandida das classes principais que compõe a rede de Hopfield pode ser vista na figura abaixo.

Figura 3.21 - Estrutura expandida da rede de Hopfield.

3.2.3.1.4 Máquina de Boltzmann

Lembrando que a Máquina de Boltzmann é utilizada para problemas de Otimização, Classificação, Memória Associativa, etc., têm-se nesta rede neural os dois modos de operação básicos, os modos de Aprendizado e de Execução. De forma equivalente às redes de Hopfield, a Máquina de Boltzmann utiliza as entidades Aprendizado Fixo e Aprendizado Dinâmico. Para as operações de Execução, a rede passa a manipular um maior número de entidades, dada a maior complexidade do algoritmo utilizado para os problemas de Otimização. Abaixo da entidade Execução, podem ser identificadas as seguintes classes:

- Controle – utilizado para controle do ciclo do algoritmo.
- Avaliação – utilizado para identificar um valor de aptidão das soluções candidatas.
- Seleção – utilizado para selecionar os estados com melhor aptidão.
- Operador Max/Min – operador utilizado pela entidade de Seleção.
- Mutação Bit a Bit – entidade responsável pela inversão probabilística de neurônios da rede.
- Probabilidade – Dada a natureza estocástica desta rede neural, a probabilidade é utilizada para cálculo dos sinais de saída dos neurônios. Dado a natureza probabilística das inversões de bits da entidade de Mutação, esta entidade também é utilizada.

De forma equivalente às outras redes neurais descritas anteriormente, abaixo da entidade Modelo é possível descrever as estruturas principais das redes neurais, desta vez considerando o Neurônio Estocástico e a Função de Consenso como entidades diferenciais, se comparado com a rede de Hopfield.

Uma visão expandida das classes principais que compõe a Máquina de Boltzmann pode ser vista na figura abaixo.

Figura 3.22 - Estrutura expandida da rede de Boltzmann.

3.2.3.2 Computação Evolutiva

Dadas as semelhanças existentes entre os algoritmos evolutivos, será analisada a estrutura de classes desta área computacional através de um modelo único e não individualmente para cada uma das sub-áreas analisadas para este trabalho.

Com base nos modelos evolutivos têm-se teorias apropriadas para problemas de otimização, onde a representação dos problemas não se restringe a nenhum modelo específico, podendo ser utilizadas representações através de seqüências de bits, grafos, etc. Para a classe estrutural Algoritmo, que só possui modo de Execução, tem-se uma representação desta entidade como composta pelos operadores evolutivos de Recombinação, Mutação, Avaliação e Seleção, onde a entidade Probabilidade é utilizada como base em diversos destes operadores.

Abaixo da entidade Modelo, são descritas as principais entidades envolvidas com os Algoritmos Evolutivos, ou seja, as classes Geração, Indivíduo, População, Genótipo, Cromossomo, Gene, Alelo, Fenótipo, etc.

Figura 3.23 - Estrutura expandida da Computação Evolutiva.

3.2.3.3 Redes Neurais vs. Algoritmos Evolutivos: Classes

Analisando as estruturas de classes de cada uma das redes neurais e dos algoritmos evolutivos, é possível identificar classes comuns:

a) Modelo

Uma das características das sub-áreas da CN, está na manipulação de Neurônios artificiais, não sendo necessária nenhuma estrutura algorítmica padrão. Dadas estas características, com o intuito de destacar a principal propriedade da CN (manipula modelos de neurônios artificiais) foi utilizada esta definição de classe (classe “Modelo”).

b) Algoritmo

Uma das características dos Algoritmos Evolutivos está na utilização de um procedimento cíclico baseando em quatro operadores evolutivos: Recombinação, Mutação, Avaliação e Seleção. Em essência, este procedimento é um padrão geral dos Algoritmos Evolutivos. Dada esta característica, com o intuito de destacar esta propriedade principal da CE (dependência quanto ao uso de operadores evolutivos) foi utilizada esta definição de classe (classe “Algoritmo”).

c) Algoritmo de “Execução”

Comumente as teorias neurais envolvem dois modos distintos de operação, um modo onde a rede é treinada e um modo de operação livre, o modo de “Execução”. Para os algoritmos evolutivos, não existe um procedimento de aprendizado, desta forma sendo considerada apenas a categoria de “Execução”. Se comparadas às entidades abaixo da classe “Execução” (para todas as redes neurais e algoritmos evolutivos descritos acima), será possível identificar algumas entidades comuns, como por exemplo as entidades “Probabilidade”, “Seleção”, “Mutação*”, “Controle” e “Avaliação”, que serão avaliados nos tópicos seguintes.

d) Probabilidade

A Probabilidade é um dos conceitos básicos utilizados pelos algoritmos evolutivos. De forma geral, diversos operadores evolutivos fazem uso desta entidade como por exemplo o operador de “Seleção” (os indivíduos são selecionados para recombinação de acordo com uma probabilidade proporcional a avaliação que tiveram), “Recombinação” (um número aleatório define pontos de corte para operações de *crossover*) e “Mutação” (alterações nos indivíduos são probabilísticas). Para as redes neurais, a probabilidade comumente é utilizada para seleção de neurônios a serem processados, seleção da probabilidade de inversão dos neurônios estocásticos, etc.

e) “Mutação” e “Mutação Bit a Bit”

Esta entidade tem como objetivo realizar uma alteração de características dos indivíduos de forma aleatória e probabilística. Para os algoritmos evolutivos, a mutação pode ser representada como a alteração de um bit do genótipo, alteração de um estado ou transição de um grafo ou alguma outra propriedade dos indivíduos. Para a máquina de Boltzmann, que faz uso de um procedimento probabilístico de inversão de sinais de neurônios, é possível considerar este procedimento (“Mutação Bit a Bit”) como equivalente ao operador de Mutação dos Algoritmos Evolutivos.

f) Seleção

Para os algoritmos evolutivos, o conceito de Seleção representa a operação de selecionar os indivíduos de uma população para serem pais da próxima geração. Para a máquina de Boltzmann, a seleção tem como objetivo identificar qual estado neural será utilizado como estado vigente da rede no próximo ciclo de processamento. Em essência, esta entidade representa operadores comuns para ambas as áreas.

g) Avaliação

Para os algoritmos evolutivos, a seleção de indivíduos é realizada com base em um valor de aptidão que é fornecido através do operador de Avaliação. Para a máquina de Boltzmann, a operação de seleção também é realizada sobre um valor de aptidão, que é calculado com base na Função de Consenso da rede e está representado no diagrama através da classe Avaliação (a Avaliação utiliza a Função de Consenso). Independente das

diferenças práticas entre os operadores utilizados pelas teorias, conceitualmente ambos os operadores possuem mesmas propriedades e podem ser vistas como semelhantes.

h) Controle

Esta classe foi representada como uma forma de evidenciar a preocupação das teorias com relação ao controle no processo de Mutação e na parada dos algoritmos. Para os algoritmos evolutivos, a mutação pode ser definida como um valor representando uma probabilidade constante ou variável de mutação. Para a máquina de Boltzmann, o controle do parâmetro denominado “pseudo-temperatura” tem influência na probabilidade de alteração dos sinais de saída, que foi definido como “Mutação Bit a Bit”. Desta forma a máquina de Boltzmann pode ser vista como um caso específico de operador evolutivo com mutação variável em função do tempo. Outra função dos parâmetros de controle é de identificação de um critério de parada, lógica existente tanto na CE quanto na Máquina de Boltzmann e rede de Hopfield. Dadas estas características, é possível considerar esta entidade de Controle como comum para ambas as áreas.

De forma sucinta, as classes abaixo da entidade Algoritmo evidenciaram semelhanças diretas, onde as classes de cada modelo possuem basicamente o mesmo nome. Ao realizar uma comparação entre as classes abaixo da entidade Modelo, a princípio não serão identificadas semelhanças explícitas. Em uma análise mais detalhada, é possível identificar como estas classes podem ser relacionadas:

a) “Modelo” → “Estado da Rede” vs. “Modelo” → “Indivíduo”

Uma vez que os estados da Máquina de Boltzmann e os indivíduos dos Algoritmos Evolutivos representam soluções de um problema de Otimização, ambas estas entidades podem ser vistas como comuns, para esta aplicação específica.

b) “Modelo” → “Neurônio Estocástico” vs. “Algoritmo” → “Operador Mutação”

Para a Máquina de Boltzmann, o neurônio estocástico permite que um estado da rede sofra com uma inversão de sinal probabilística, que pode ser visualizado como base para o equacionamento de um Operador de Mutação para Algoritmos Evolutivos binários. Cabe ressaltar que um Operador de Mutação genérico pode possuir uma lógica mais complexa do que a equação do neurônio estocástico, desta forma fazendo com que o Neurônio Estocástico represente um caso específico dos Algoritmos Evolutivos.

c) “Modelo” → “Pesos Sinápticos” / “Modelo” → “Função de Consenso” vs. Algoritmo → “Função Objetivo”

Um dos conceitos utilizados por [Aarts&Korst-1989] na descrição da Máquina de Boltzmann é a denominada Função de Consenso, que possui papel importante nos problemas de otimização. Basicamente esta função é utilizada para avaliar a performance de um estado da rede, desta forma fazendo com que esta função represente um caso específico de Função Objetivo dos algoritmos evolutivos. Também é relacionada neste tópico a entidade “Pesos Sinápticos” dada a relação direta que esta possui na representação

da função de consenso. Neste contexto, é possível interpretar que os pesos sinápticos e a função de consenso representam um caso específico de Função Objetivo para Algoritmos Evolutivos binários.

d) Modelo → “Pseudo Temperatura” vs. Algoritmo → “Controle”

Uma das características dos neurônios estocásticos, está na existência de um parâmetro de controle denominado “pseudo-temperatura” (vide tópico 2.2.1.2), que influencia na probabilidade de inversão dos sinais de saída dos neurônios. Na resolução de problemas de otimização utilizando a Máquina de Boltzmann, este parâmetro deve ser controlado para que a rede neural consiga chegar a uma solução ótima para o problema. Para os algoritmos evolutivos, funções de controle das taxas de mutação também podem ser aplicadas. Desta forma, a “pseudo-temperatura” pode ser vista como um caso específico de controle de mutação para os Algoritmos Evolutivos.

e) Modelo → “Realimentação” vs. Modelo → “Geração”

Para uma rede neural, a realimentação de sinais faz com que o algoritmo da rede apresente um comportamento cíclico. Para um Algoritmo Evolutivo a passagem das Gerações também implica em um algoritmo iterativo. Dadas estas características, é possível interpretar uma equivalência entre a Realimentação das redes neurais com a Geração dos algoritmos evolutivos.

Com base nas análises realizadas neste tópico, foi possível identificar diversas semelhanças entre classes da CN e CE para problemas relacionados com Otimização. Até este nível de abstração, características específicas dos problemas não precisaram ser avaliadas. Como as implementações dos operadores da CE e das funções da CN necessitam ser realizadas de forma específica para os problemas sendo resolvidos, no tópico seguinte serão analisadas as implementações de ambas as áreas para um problema específico de Otimização.

3.2.4 Mapeamentos

Uma vez analisadas as entidades básicas das estruturas e dos algoritmos, pode-se agora considerar a análise das implementações das funções de cada área. Uma vez que uma implementação depende de características específicas dos problemas sendo solucionados, então esta análise será realizada sobre o caso prático de interesse, o Problema do Caixeiro Viajante.

Com o intuito de identificar equivalências entre a CN e a CE, este trabalho irá inicialmente considerar a possibilidade de conversão de implementações entre os modelos de ambas as áreas. Para esta abordagem, é necessário levar em consideração um fato importante: para um modelo ser considerado parte da CN, este deverá obrigatoriamente ser baseado em conceitos neurais, ou seja, deverá manipular neurônios artificiais e outros conceitos associados. Neste caso, o mapeamento de implementações da CE para a CN só poderá ser realizado se o Algoritmo Evolutivo for baseado em modelos neurais. Dada esta restrição, por simplicidade neste trabalho serão analisados mapeamentos da Máquina de Boltzmann para a CE.

Com base nas análises apresentadas ao longo deste texto, foi sugerido que a Máquina de Boltzmann pode ser considerada um caso específico dos Algoritmos Evolutivos. Como forma de evidenciar esta similaridade entre os modelos, o procedimento a ser apresentado irá converter as equações da Máquina de Boltzmann em operadores evolutivos. Como forma de descrever estes operadores, será utilizada a notação padrão das Estratégias Evolutivas.

Segundo as análises realizadas anteriormente, a Máquina de Boltzmann pode ser vista como um Algoritmo Evolutivo binário com $\mu=1$ pais e $\lambda=1$ descendentes. Dada esta similaridade, nas análises seguintes será utilizada a definição da Estratégia Evolutiva $(1+1)-ES$ conforme descrita em [Bäck&Hammel&Schwefel-1991] (vide tópico 2.3.3.1). Com base nesta definição é possível sugerir a seguinte aproximação:

$$\text{Máquina de Boltzmann} \approx (1+1)-ES = (P^0, m, s, c_d, c_i, f, g, t) \quad (3.2)$$

Onde P^0 é a população, m é o operador de mutação, s é o operador de seleção, c_d e c_i são os controles do algoritmo, f é a função objetivo, g representa as funções de restrição e t é critério de parada do algoritmo.

A seguir cada um destes parâmetros será analisado separadamente.

População P^0

$P^0 = (\vec{x}^0, \sigma^0) \in I$ representa a população com um único pai, onde $I = \{0,1\}^{N \times N} \times \mathbb{R}$ é o domínio do indivíduo para o caso específico de uma solução binária com controle de mutação $\sigma \in \mathbb{R}$ (perceba que nesta modelagem está sendo considerado um caso específico do parâmetro de controle, que é descrito em [Bäck&Hammel&Schwefel-1991] como sendo $\sigma \in \mathbb{R}^n$). Uma vez que a população é composta por um único indivíduo pai, pode-se descrever:

$$P^0 = a^0 = (\vec{x}^0, \sigma^0) \quad (3.3)$$

Onde a^0 é o primeiro indivíduo pai do algoritmo com genótipo $\vec{x}^0$, que é baseado na representação das soluções em forma de matriz $c \times c$ (onde c representa o número de cidades do problema). O genótipo dos indivíduos $\vec{x}^t$ pode ser representado através da notação definida pela equação (3.1):

$$\vec{x}^t = \vec{S}(t) = \left\{ \{x_{0,0}^t, \dots, x_{0,N-1}^t\}, \dots, \{x_{N-1,0}^t, \dots, x_{N-1,N-1}^t\} \right\} \quad (3.4)$$

No mapeamento entre modelos, o controle de mutação σ^t será considerado equivalente à pseudo-temperatura do neurônio estocástico:

$$\sigma^t = T^t \quad (3.5)$$

Operador de mutação m

$m: I \rightarrow I$ corresponde ao operador de mutação e pode ser definido por uma inversão probabilística de sinal do neurônio estocástico:

$$m(P^t) = m\left(\left(\overrightarrow{x^t}, \sigma^t\right)\right) \quad (3.6)$$

$$m\left(\left(\overrightarrow{x^t}, \sigma^t\right)\right) = \left(m'\left(\overrightarrow{x^t}\right), \sigma^t\right) \quad (3.7)$$

$$m'\left(\overrightarrow{x^t}\right) = \left\{\left\{m''\left(x_{0,0}^t\right), \dots\right\}, \dots, \left\{\dots, m''\left(x_{N-1,N-1}^t\right)\right\}\right\} \quad (3.8)$$

$$m''\left(x_{ip}^t\right) = \begin{cases} x_{ip}^t & \text{if } N_0 > \frac{1}{1 + \exp\left(-v_{x_{ip}}^t / \sigma^t\right)} \\ \sim x_{ip}^t & \text{else} \end{cases} \quad (3.9)$$

$$\sim x_{ip}^t = \begin{cases} 1 & \text{if } x_{ip}^t = 0 \\ 0 & \text{else} \end{cases} \quad (3.10)$$

Onde,

$$m': \{0,1\}^n \rightarrow \{0,1\}^n,$$

$$m'': \{0,1\} \rightarrow \{0,1\},$$

$N_0 \in \mathbb{R}$ é um número aleatório,

$v_{x_{ip}}^t$ é dado pela equação do sinal interno de um neurônio estocástico e

σ^t é a pseudo-temperatura.

Dada a representação adotada para a solução (em forma de matriz), é possível reescrever $v_{x_{ip}}^t$ como sendo:

$$v_{x_{ip}}^t = \sum_{j=1}^N \sum_{q=1}^N w_{\{x_{ip}, x_{jq}\}} x_{jq}^t \quad (3.11)$$

Onde,

$w_{\{u_{ip}, u_{jq}\}}$ representa o peso sináptico conectando neurônios da matriz que possuem coordenadas (i, p) e (j, q) .

Finalmente, é possível descrever o procedimento de uso do operador de mutação:

$$\begin{aligned} P^{t'} &= (a_1^{t'}, a_2^{t'}) \in I \times I \\ a_1^{t'} &= P^t = (\vec{x}^t, \sigma^t) \\ a_2^{t'} &= m(P^t) = (\vec{x}^{t'}, \sigma^{t'}) \end{aligned} \quad (3.12)$$

Operador de Seleção s

$s : I \times I \rightarrow I$ representa o operador de Seleção, que é utilizado para determinar o indivíduo vencedor para ser o pai da geração $t+1$:

$$\begin{aligned} P^{t+1} &= s(P^{t'}) \\ s(P^{t'}) &= \begin{cases} a_2^{t'} & \text{if } f(\vec{x}^{t'}) \leq f(\vec{x}^t) \\ a_1^{t'} = P^t & \text{else} \end{cases} \end{aligned} \quad (3.13)$$

Onde, $f : \{0,1\}^{N \times N} \rightarrow \mathbb{R}$ é a função objetivo, descrito como parâmetro 6 do algoritmo (1+1)-ES.

Neste procedimento, é importante notar o uso do operador de maximização, representado pela comparação $f(\vec{x}^{t'}) \leq f(\vec{x}^t)$ da equação (3.12), seleciona a solução com maior valor de aptidão. Cabe ressaltar que outras implementações poderiam ser utilizadas com o intuito de evitar máximos locais, como soluções baseadas em seletividade probabilística, etc.

Parâmetro de Controle c_d

$c_d \in \mathbb{R}$ representa um parâmetro de controle para o operador de mutação. Uma vez que o operador é baseado na equação do neurônio estocástico, este parâmetro será considerado para o controle do resfriamento da Máquina de Boltzmann, que deve ter sua pseudo-temperatura reduzido para que o sistema convirja para uma solução ótima. Para esta redução, será considerado um procedimento simples baseado na equação:

$$\sigma^t = (c_d)^t \sigma^0 \quad (3.14)$$

Onde t representa as gerações.

Parâmetro de Controle c_i

c_i representa o segundo parâmetro de controle do operador de mutação. Para a estratégia evolutiva descrita em [Bäck&Hammel&Schwefel-1991], o operador de mutação é controlado por dois diferentes parâmetros, como forma de obedecer a regra do 1/5 de Rechenberg (vide tópico 2.3.3.1). Como a Máquina de Boltzmann utilize um cronograma de esfriamento que decrementa as mutações para zero quando $t \rightarrow \infty$, então este parâmetro não é necessário e não será utilizado neste exemplo.

Função Objetivo f

$f : \{0,1\}^{N \times N} \rightarrow \mathbb{R}$ representa a função objetivo e é baseado na função de consenso da Máquina de Boltzmann.

$$f(\vec{x}) = C(\vec{x}) = \sum_{\{x_{ip}, x_{ip}\} \in \mathbb{C}_{\text{bias}}} w_{\{x_{ip}, x_{ip}\}} (x_{ip}^t)^2 + \sum_{\{x_{ip}, x_{jq}\} \in \mathbb{C}_{\text{distance}}} w_{\{u_{ip}, u_{jq}\}} x_{ip}^t x_{jq}^t \quad (3.15)$$

Onde, $i, j, p, q = 0, \dots, N-1$ representa as coordenadas da matriz da solução, onde i e j representam o eixo das cidades e p com q representam o eixo da ordem de visita.

Função de Restrição g

$g = \{g_1, g_2\}$ representa a função de restrições do problema, onde $g_1 : \{0,1\}^{N \times N} \rightarrow \mathbb{R}$ e $g_2 : \{0,1\}^{N \times N} \rightarrow \mathbb{R}$. Para a Máquina de Boltzmann, as restrições são utilizadas na inicialização da rede neural, i.e. definição dos pesos sinápticos e *bias*, e são definidos pelas restrições de linha e coluna, onde apenas um elemento por linha e coluna pode estar ativado por vez:

$$g = \{g_1, g_2\}$$

$$g_1(\vec{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^N x_{ip} = 1 \\ -1 & \text{else} \end{cases} \quad (3.16)$$

$$g_2(\vec{x}) = \begin{cases} 1 & \text{if } \sum_{p=1}^N x_{ip} = 1 \\ -1 & \text{else} \end{cases}$$

Critério de Parada t

$t: I \times I \rightarrow \{0,1\}$ representa o critério de parada do algoritmo. Para a Máquina de Boltzmann, comumente o algoritmo finaliza quando o procedimento de resfriamento reduz a pseudo-temperatura para um valor baixo o suficiente que mudanças de estado não são mais perceptíveis ao longo das iterações. Desta forma o critério de parada comumente utilizado é a interrupção do algoritmo quando não ocorrer nenhuma alteração no estado da rede após um número de iterações predefinido. Outros critérios também podem ser utilizados.

3.2.5 Considerações Finais do Capítulo

Nas análises realizadas até o momento, foi possível identificar tanto semelhanças estruturais quanto funcionais entre a CE e CN para problemas de Otimização, levando em consideração uma aplicação prática para o Problema do Caixeiro Viajante. As principais semelhanças identificadas, envolveram a Máquina de Boltzmann e os Algoritmos Evolutivos e desta forma apenas estes dois modelos serão considerados nesta etapa final de conclusões. Para avaliação das semelhanças identificadas segundo os conceitos definidos pelo diagrama de Gajski & Kuhn, as classes dos dois modelos serão mapeadas como pontos discretos ao longo dos eixos Estrutural (representando a Computação Neural) e Físico/Geométrico (representando a Computação Evolutiva) do gráfico proposto em [Andrade-2002, pp. 43] (figura abaixo).

Figura 3.24 Análise do Diagrama de Gajski & Kuhn [Andrade-2002, pp. 43]

Levando em consideração que o foco deste trabalho está na análise da Computação Neural e Computação Evolutiva, será desconsiderado o eixo Comportamental, que faz referência à Computação Nebulosa. Com base nas análises realizadas, é possível sugerir os seguintes mapeamentos:

Nível “0” de Abstração: Visão de Contexto

Máquina de Boltzmann (Otimização)		Computação Evolutiva
Sistema CN	↔	Sistema CE

Nível “1” de Abstração: Classes Principais

Máquina de Boltzmann (Otimização)		Computação Evolutiva
Modelo	↔	Modelo
Algoritmo	↔	Algoritmo

Nível “2” de Abstração: Classes equivalentes entre Máquina de Boltzmann para Otimização e os Algoritmos Evolutivos binários com um pai e um descendente.

Máquina de Boltzmann (Otimização)		Computação Evolutiva
Equacionamento Neurônio Estocástico	↔	Operador de Mutação
Função de Consenso	↔	Operador de Avaliação (ou Função Objetivo)
Operador Max/Min	↔	Operador de Seleção

Nível “3” de Abstração: Classes equivalentes entre Máquina de Boltzmann para Otimização e os Algoritmos Evolutivos binários com um pai e um descendente, ambos visando resolver a aplicação específica do Problema do Caixeiro Viajante.

Máquina de Boltzmann (Otimização)		Computação Evolutiva
Pseudo-Temperatura	↔	Controle de Mutação
Estado da Rede Neural	↔	Indivíduo
Critério de parada da Realimentação	↔	Critério de parada das Gerações

Após a definição dos mapeamentos entre as entidades de cada modelo, é possível sobrepor estas informações com os eixos do diagrama de Gajski & Kuhn da IC conforme apresentado na figura abaixo.

Figura 3.25 Equivalências identificadas entre Máquina de Boltzmann para Otimização e Algoritmos Evolutivos binários com um pai e um descendente para o Problema do Caixeiro Viajante.

É importante notar que as relações mapeando os elementos da CE para a CN só podem ser realizadas se os Algoritmos Evolutivos forem baseados em conceitos neurais. Esta restrição pode então ser visualizada graficamente na figura acima através das setas menores mapeando os elementos da CE com a CN.

4. CONSIDERAÇÕES FINAIS

Levando em consideração os objetivos propostos, as estratégias previstas e as análises realizadas, serão abordadas neste capítulo as seguintes considerações finais:

- Cumprimento dos Objetivos Propostos
- Contribuições do Trabalho
- Conclusões
- Trabalhos Futuros

4.1 Cumprimento dos Objetivos Propostos

O objetivo deste trabalho foi de realizar uma análise comparativa entre a Computação Evolutiva e a Computação Neural, de forma a identificar semelhanças entre as teorias e sustentar mapeamentos entre as semelhanças identificadas. Como forma de realizar estas análises, a estratégia proposta foi de utilização de uma abordagem “*top-down*”, onde os modelos de cada área seriam analisados como um módulo único e posteriormente descompostos em módulos de menor complexidade.

Com base nos objetivos propostos e nas análises realizadas nos Capítulos 2 e 3, pode-se afirmar que os objetivos foram atingidos. Isto se sustenta baseado no fato de que foi realizada uma análise em diferentes níveis de abstração e foi proposto um mapeamento entre elementos da Máquina de Boltzmann para um Algoritmo Evolutivo com características específicas. Podem-se descrever em maiores detalhes cada um dos níveis avaliados:

- Nível 1: Neste nível de abstração foram listadas Classes de Utilização básicas que cada modelo é capaz de resolver. Cada Classe identificada foi utilizada para descrição dos módulos em forma de “caixa preta” onde as interfaces de entrada e saída foram definidas com base em definições existentes na literatura (tópico 3.1.1).
- Níveis 2 e 3: Nestes níveis de abstração as redes neurais e algoritmos evolutivos foram analisados visando resolver problemas da Classe de otimização combinatorial. As análises realizadas compararam as estruturas dos algoritmos e das classes de cada modelo. Não foram realizadas análises comparativas detalhadas sobre as funções e equações de cada modelo (tópicos 3.2.2.*,* e 3.2.3.*,*).
- Nível 4: Como cada rede neural e algoritmo evolutivo necessita ser modelado de acordo com o problema sendo resolvido, neste nível de abstração foi analisado o caso específico do Problema do Caixeiro Viajante. Nas análises realizadas, as equações de uma Máquina de Boltzmann modelado para o Problema do Caixeiro Viajante foram mapeadas em operadores de um Algoritmo Evolutivo binário com um pai e um descendente (tópicos 3.2.4 e 3.2.5).

4.2 Contribuições do Trabalho

Além da análise realizada, podem ser consideradas como principais contribuições deste trabalho os seguintes itens:

- Definição do conceito de “Classes de Utilização”, onde cada classe representa uma finalidade de uso ou uma função dentro de uma aplicação (tópico 3.1.1).
- Listagem de “Classes de Utilização” básicas dos modelos neurais e evolutivos abordados no trabalho, ou seja, as classes de Aproximação de Funções, Classificação, Agrupamento de Dados (*Clustering*), Memória associativa, Otimização e Reconhecimento de Padrões (tópico 3.1.1 - Tabela 3.1).
- Definição de interfaces tipo “caixa preta” para cada uma das “Classes de Utilização” listadas (tópico 3.1.1).
- Descrição de um algoritmo genérico para a Máquina de Boltzmann seqüencial de otimização (tópico 3.2.2.4).
- Comparação entre a estrutura do algoritmo da Máquina de Boltzmann com o algoritmo genérico dos Algoritmos Evolutivos (tópico 3.2.2.6 - Tabela 3.2).
- Elaboração de um Diagrama de Classes para cada um dos modelos neurais abordados no trabalho e um Diagrama de Classes genérico para os modelos da Computação Evolutiva (tópicos 3.2.3.1.*).
- Padronização de uma representação comum para as entradas e as saídas do Problema do Caixeiro Viajante para a Máquina de Boltzmann e os Algoritmos Evolutivos (tópico 3.2.2.3).
- Mapeamentos entre equações da Máquina de Boltzmann modelado para o Problema do Caixeiro Viajante e operadores evolutivos de um Algoritmo Evolutivo binário com um pai e um descendente utilizando a notação da Estratégia Evolutiva (1+1)-ES (tópicos 3.2.4 e 3.2.5).

4.3 Conclusões

Ao longo das análises realizadas, foi possível identificar que a CN e a CE são capazes de resolver problemas de otimização combinatorial, que engloba o Problema do Caixeiro Viajante. Com base na estrutura dos algoritmos da Máquina de Boltzmann para problemas de otimização, estrutura dos algoritmos dos Algoritmos Evolutivos e na estrutura de classes destes modelos, foi possível identificar uma estrutura equivalente entre a CN e a CE para os casos específicos de otimização.

Com base nas equações da Máquina de Boltzmann para o Problema do Caixeiro Viajante, foi possível definir operadores evolutivos de um Algoritmo Evolutivo binário com um pai e um descendente. Tal abordagem evidencia uma semelhança entre as diferentes áreas para o caso específico do Problema do Caixeiro Viajante.

A análise de mapeamentos genéricos (não restritos aos problemas de otimização e/ou não restritos ao Problema do Caixeiro Viajante) entre a CN e a CE não foi realizada neste trabalho. Dentre as principais dificuldades encontradas para o mapeamento genérico pode-se destacar:

- Necessidade de modelagem das redes neurais e dos Algoritmos Evolutivos de forma específica para cada problema.
- Necessidade de inicialização das redes neurais de forma específica para cada problema de otimização combinatorial.
- Existência de um número restrito de redes neurais descritas na literatura para a resolução de problemas de otimização combinatorial.
- Dificuldade de representação unificada de problemas para todos os modelos (conforme analisado para a rede de Kohonen, existe uma dificuldade de representação do Problema do Caixeiro Viajante assimétrico, etc.).
- Maior número de Classes de Uso relacionadas com a CN do que com a CE. Tal característica restringe a possibilidade de conversões da CE para a CN.
- Natureza das redes neurais, que necessitam manipular neurônios artificiais e agregados, restringe a possibilidade de conversão de modelos da CE para a CN.

4.4 Trabalhos Futuros

Visando uma continuidade dos trabalhos dentro da área de pesquisa, outros cenários poderão ser analisados, como por exemplo:

- Estudo de cenários com três ou mais soluções competindo entre si. Neste trabalho foi analisada a Máquina de Boltzmann para problemas de otimização com um Algoritmo Evolutivo com população composta por dois indivíduos (ou soluções), um indivíduo pai e um indivíduo descendente. Neste contexto uma alteração dos algoritmos ou da arquitetura da Máquina de Boltzmann teria que ser realizada para suportar mais do que dois estados de rede.
- Estudo de conversões de problemas de otimização NP completos de forma geral (não aplicado a um caso específico, como por exemplo o Problema do Caixeiro Viajante). Segundo [Aarts&Korst-1989, pag.155], todos os problemas da classe NP completo (classe englobada pela otimização combinatorial) são mapeáveis entre si e podem ser resolvidos por uma Máquina de Boltzmann. Desta forma, outro cenário de estudo teria como objetivo identificar esta forma genérica da Máquina de Boltzmann para resolução de problemas de otimização NP completos e identificar um mapeamento entre este modelo e os Algoritmos Evolutivos. Neste contexto, seria também necessário identificar regras para conversões entre problemas distintos na Máquina de Boltzmann.
- Estudo de conversões de problemas de otimização genéricos. Este trabalho abordou um exemplo de otimização relacionado com a classe de otimização denominada NP completo. Uma visão genérica da otimização envolveria o estudo das classes Polinomiais e NP incompletas de complexidade.
- Análise entre a Máquina de Boltzmann para otimização e os Algoritmos Evolutivos binários com um pai e um descendente visando identificar homomorfismos entre os modelos.

5. REFERÊNCIAS

[Ackley&Hinton&Sejnowski-1985] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski,. A Learning Algorithm for Boltzmann Machines, Cognitive Science, 147–169, 1985.

[Alderighi & D’Angelo & Sechi-98] Alderighi, M.; Dapos;Angelo, S.; Sechi, G.R.; dapos;Ovidio, F.; “Experimenting genetic algorithms for training a neural networkprototype for photon event identification”; System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on Volume 3, Issue , 1998 Page(s):283 - 291 vol.3

[Albizuri&Anjou&Grana-1996] Albizuri F.X.1; d'Anjou A.; Grana M.; Lozano J.A, “Convergence Properties of High-order Boltzmann Machines”, Neural Networks, vol. 9, Number 9, pp. 1561-1567(7), Dec 1996.

[Andrade-2002b] ANDRADE, Marco Túlio Carvalho de. “Uma contribuição à pesquisa em inteligência computacional”. 2002. 71 p. Tese (Livre Docência) - ESCOLA POLITECNICA, Universidade de São Paulo, São Paulo, 2002.

[Aarts&Korst-1989] Aarts, Emile & Korst, Jan; “Simulated Annealing and Boltzmann Machines”, ohn Wiley & Sons, Inc., 1988

[Bäck&Hammel&Schwefel-1997] Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 1, NO. 1, APRIL 1997, Evolutionary Computation: Comments on the History and Current State

[Bäck&Hammel&Schwefel-1991] Bäck T., Hoffmeister F. and Schwefel H.P. 1991. A survey of evolutionary strategies Proc. 4th Int. Conf. on Genetic Algorithms (San Diego, CA, July 1991) ed R K Belew and L B Booker (San Mateo, CA: Morgan Kaufmann) pp 2–9

[Bakhouya&Gaber-2007] M. Bakhouya and J. Gaber; “An Immune Inspired-based Optimization Algorithm Application to the Traveling Salesman Problem”; AMO - Advanced Modeling and Optimization, Volume 9, Number 1, 2007.

[Bellarmine&Bhuvaneswari&Subramanian-2005] Bellarmine, G.T. Bhuvaneswari, R. Subramanian, S.; “Radial Basis Function Network based Design Optimization of Induction Motor”; SoutheastCon, 2006. Proceedings of the IEEE; March 31 2005-April 2 2005; p.75-80

[Bishop-2006] Christopher M. Bishop; “Pattern Recognition and Machine Learning”; Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2006.

[Bourbakis-98] Xusheng Lu; Bourbakis, N.; Neural network training using genetic algorithms in ATM trafficcontrol; Intelligence and Systems, 1998. Proceedings., IEEE International Joint Symposia on Volume , Issue , 21-23 May 1998 Page(s):396 - 401

[Bruck-1990] Bruck, J.; “On the convergence properties of the Hopfield model”, Proceedings of the IEEE, vol. 78, Issue 10, pp. 1579-1585, Oct 1990.

[Carballido&Ponzoni&Brignole-03] [13] Carballido, J. A., Ponzoni, I. y Brignole, N. B., MCPC and MCMP Evolutionary Algorithms for the TSP, V ASAI, XXXII JAIIO, Bs.As., Argentina, Sep. 2003.

[Caversan&Andrade-2005] Caversan, F.L.; Andrade, M.T.C. Exploração de relações isomórficas entre técnicas simbólicas e conexonistas da inteligência computacional. INFOCOMP Journal of Computer Science, v. 4, n. 1, p. 19-25, 2005.

[Caversan&Fialho&Andrade-2007] F.L. Caversan, A.R.S. Fialho, M.T.C. de Andrade, “Exploration of Relations between Symbolic and Conexonist Techniques of Computational Intelligence”, Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on Volume, Issue, 20-24 Oct. 2007 Page(s):70 – 78 Digital Object Identifier 10.1109/ISDA.2007.124.

[Chatterjee&Carrera&Lynch-1996] S. Chatterjee, C. Carrera, L. A. Lynch, “Genetic algorithms and traveling salesman problems”, European Journal of Operational Research., vol. 93, Issue 3, pp. 490-510, Sep 1996.

[Cho&Oh&Choi-1998] Hyeon-loong Cho, Se-Young Oh, Doo-Hyun Choi; “A new evolutionary programming approach based on simulatedannealing with local cooling schedule”; Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on; 4-9 May 1998; On page(s): 598-602.

[Duch-2007] Duch, Włodzisław; “What is Computational Intelligence and what could it become?” Department of Informatics, Nicolaus Copernicus University, Grudziadzka 5, 87-100 Torun, Poland. School of Computer Engineering, Nanyang Technological University, Singapore, 2007.

[Fausett-1993] Fausett, Laurene V.; “Fundamentals of Neural Networks: Architectures, Algorithms And Applications”, Prentice Hall 1st edition, December 1993.

[Fialho-2007] Fialho, Álvaro Roberto Silvestre; Exploração de relações entre as técnicas nebulosas e evolutivas da inteligência computacional. Dissertação Mestrado Escola Politécnica (EP) 2007.

[Gajski&Kuhn-1983] D. D. Gajski, R. Kuhn. “New VLSI tools.” IEEE Computer, pp. 11-14, 1983.

[Gloria&Faraboschi&Olivieri-1993] A. De Gloria, P. Faraboschi, and M. Olivieri. Clustered Boltzmann Machines: Massively parallel architectures for constrained optimization problems. *Parallel Computing*, 19(3):163-175, March 1993.

[Haykin-1999] Simon Haykin, *Neural Networks: A Comprehensive Foundation* (2nd Edition), Prentice Hall PTR, Upper Saddle River, NJ, 1999.

[Jain&Murty&Flynn-1999] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review", *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, 1999.

[Jeong& Park-1989] Hong Jeong; Jeong-Ho Park, "Lower bounds of annealing schedule for Boltzmann and Cauchy machines," *Neural Networks*, 1989. *IJCNN.*, International Joint Conference on , vol., no., pp.581-586 vol.1, 18-22 Jun 1989

[Jong&Lawrence&Schwefel-1997] Jong, Kenneth, Fogel, Lawrence, Schwefel Hans-Paul; "The Handbook of Evolutionary Computation"; IOP Publishing Ltd and Oxford University Press, 1997.

[King&Azriel-1976] KING-SUN FU, FELLOW, IEEE, AND AZRIEL ROSENFELD, FELLOW, IEEE "Pattern Recognition and Image Processing", *IEEE TRANSACTIONS ON COMPUTERS*, VOL. C-25, NO. 12, DECEMBER 1976

[Kirbiz&Ulker&Gunsel-2007] Serap Kirbiz, Yener Ulker, Bilge Gunsel, A pattern recognition framework to blind audio watermark decoding, *AEU - International Journal of Electronics and Communications*, Volume 63, Issue 2, 4 February 2009, Pages 92-102

[Koper&Wood&Schneider-2004] Koper, E.M. Wood, W.D. Schneider, S.W., "Aircraft antenna coupling minimization using genetic algorithms and approximations", Dept. of Electr. & Comput. Eng., Air Force Inst. of Technol., Wright-Patterson AFB, OH, USA; Aerospace and Electronic Systems, *IEEE Transactions on Publication Date: April 2004* Volume: 40, Issue: 2 p.742-751.

[Lee&Shim&Han&Kim&Kim-2008] Kang-Hee Lee, Hyun-Sik Shim, Woo-Sup Han, Kwang-Choon Kim, Jong-Hwan Kim; "Evolutionary algorithm for a genetic robot's personality; Robot and Human Interactive Communication", 2008. *RO-MAN 2008*. The 17th IEEE International Symposium on, Publication Date: 1-3 Aug. 2008, On page(s): 506-513, Location: Munich

[Russel&Norvig-2003] S. Russel, P. Norvig, "Artificial Intelligence. A Modern Approach, Second edition" Prentice Hall, 2003

[Sakataa&Kanekob&Kanekob&Takagic&Okudac-2005] Yukinobu Sakataa, Shuni'chi Kanekob, Yuji Takagic and Hirohito Okudac; "Successive pattern classification based on test feature classifier and its application to defect image classification"; *Pattern recognition*; 2005, vol. 38, no11, pp. 1847-1856.

[Shengwu&Chengjun-2002] Xiong Shengwu, Li Chengjun; “A distributed genetic algorithm to TSP”; Intelligent Control and Automation, 2002. Proceedings of the 4th World Congress on; 2002; Volume: 3, On page(s): 1827- 1830 vol.3.

[Singh-1999] S. Singh; “A Long Memory Pattern Modelling and Recognition System for Financial Time-Series Forecasting”, Pattern Analysis & Applications, Volume 2, Number 3 / August, 1999, p.264-273.

[Smith-1999] Kate A. Smith, Neural Networks for Combinatorial Optimization: a Review of More Than a Decade of Research, INFORMS Journal on Computing, v.11 n.1, p.15-34, January 1999.

[Steidley-1993] Steidley, C.W.; “Object-oriented software: theory, practice, and implementation”; Northcon/93. Conference Record; 12-14 Oct 1993; p.132–135

[Sutton&Barto-1998] Sutton, R. S. and Barto, A. G.; Reinforcement Learning: An Introduction; MIT Press, Cambridge, MA, 1998; A Bradford Book.

[Swales-1991] Youngohc Yoon Swales, G.; “Predicting stock price performance: a neural network approach”; System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on, Publication Date: 8-11 Jan 1991, Volume: iv, On page(s): 156-162 vol.4.

[Tesar&Kapenga&Trenary-1989] Tesar, B.B. Kapenga, J. Trenary, R., “A Boltzmann machine solution of the traveling salesperson problem:a study for parallel implementation,” in TENCON '89. Fourth IEEE Region 10 International Conference, Bombay, India, 1989, pp. 142-144.

[Verhagen-1975] C. J. D. M. Verhagen; “Some general remarks about pattern recognition; its definition; its relation with other disciplines; a literature survey”, October 1974.

[Walczak-2005] Walczak, S.; “Artificial neural network medical decision support tool: predicting transfusion requirements of ER patients”, Information Technology in Biomedicine, IEEE Transactions on, Publication Date: Sept. 2005, Volume: 9, Issue: 3, p.468-474.

[Wang&Schuermans&Peng&Zhao] Shaojun Wang, Dale Schuurmans, Fuchun Peng, Yunxin Zhao; "Learning Mixture Models with the Latent Maximum Entropy Principle"; Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA, p.784-791

[Wang&Qin-2007] Yuping Wang, Jinhua Qin; “A Memetic-Clustering-Based Evolution Strategy for Traveling Salesman Problems”; Springer Berlin / Heidelberg, 2007, p.260-266.

[Yang&Butler-1998] Yang, Q P e Butler, C, An Object-Oriented Model of Measurement Systems, IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 47, NO. 1, FEBRUARY 1998

[Zomaya-2001] Zomaya, A.Y, “Natural and Simulated Annealing”, Computing in Science & Engineering, vol. 3, Issue 6, pp. 97–99, Nov/Dec 2001.

6. LISTA DE ABREVIATURAS E PALAVRAS RESERVADAS

AE – Algoritmo Evolutivo

AG – Algoritmo Genético.

CE - Computação Evolutiva.

CF - Computação Fuzzy/Nebulosa.

CN - Computação Neural.

EE – Estratégia Evolutiva

IC – Inteligência Computacional.

PE – Programação Evolutiva

PMC – Perceptron de Múltiplas Camadas

Knoma – Knowledge Management Group.

RNA – Rede Neural Artificial, ou Redes Neurais Artificiais.

7. LISTA DE SÍMBOLOS

- $\emptyset$ - Conjunto vazio.
- $\supset$ - Contém a.
- $\supseteq$ - Contém, ou igual a.
- $\subset$ - Contido em.
- $\subseteq$ - Contido em, ou igual a.
- $\rightarrow$ - Implicação.
- $\leftarrow$ - Atribuição.
- $\cap$ - Intersecção.
- $\notin$ - Não pertence a.
- $\neg$ - Negação.
- $\oplus$ - Ou exclusivo.
- $\in$ - Pertence a.
- α - Peso de uma regra.
- $\exists$ - Quantificador – Existe pelo menos um elemento.
- $\forall$ - Quantificador – Qualquer que seja.
- $\cup$ - União.

8. GLOSSÁRIO

- Fuzzy – Antônimo de “Crisp”. Significa: vago, subjetivo, impreciso, incerto, ambíguo.
- Inteligência Artificial – Conjunto de técnicas que explora a possibilidade de criar programas que possam competir com seres Humanos em certas atividades. Para tal faz uso, por exemplo, de procedimentos heurísticos, conhecimento encapsulado, lógica simbólica, etc.
- Inteligência Computacional - Conjunto de técnicas que explora o potencial para criar “Máquinas Inteligentes” por meio da modelagem dos comportamentos e mecanismos que são subjacentes aos organismos biologicamente inteligentes.
- Teoria: teoria é uma síntese aceita de um vasto campo de conhecimento, consistindo de hipóteses que foram devidamente testadas, através de leis e fatos científicos que descrevem os fenômenos naturais (mas refutável)
- Modelo: representação ou interpretação simplificada da realidade através de equações, fórmulas matemáticas, etc.
- Sistema: conjunto de elementos interconectados, de modo a formar um todo organizado