



**André Vargas Abs da Cruz**

**Algoritmos Evolutivos com Inspiração Quântica  
para Problemas com Representação Numérica**

**Tese de Doutorado**

Tese apresentada ao Programa de Pós–graduação em Sistemas de Apoio à Decisão do Departamento de Engenharia Elétrica da PUC-Rio como requisito parcial para obtenção Do título de Doutor em Sistemas de Apoio à Decisão

Orientadores: Prof. Marco Aurélio C. Pacheco  
Prof. Marley M. B. R. Vellasco

Rio de Janeiro  
Março de 2007

# **Livros Grátis**

<http://www.livrosgratis.com.br>

Milhares de livros grátis para download.



**André Vargas Abs da Cruz**

**Algoritmos Evolutivos com Inspiração Quântica  
para Problemas com Representação Numérica**

Tese apresentada ao Programa de Pós-graduação em Sistemas de Apoio à Decisão do Departamento de Engenharia Elétrica do Centro Técnico Científico da PUC-Rio como requisito parcial para obtenção Do título de Doutor em Sistemas de Apoio à Decisão. Aprovada pela Comissão Examinadora abaixo assinada.

**Prof. Marco Aurélio C. Pacheco**

Orientador

Departamento de Engenharia Elétrica — PUC-Rio

**Prof. Marley M. B. R. Vellasco**

Orientador

Departamento de Engenharia Elétrica — PUC-Rio

**Prof. Carlos Roberto Hall Barbosa**

Pontifícia Universidade Católica do Rio de Janeiro

**Prof. Karla Tereza Figueiredo Leite**

Universidade Estadual do Rio de Janeiro

**Prof. Leandro dos Santos Coelho**

Pontifícia Universidade Católica do Paraná

**Prof. Renato Portugal**

Laboratório Nacional de Computação Científica

**Prof. Valmir C. Barbosa**

Universidade Federal do Rio de Janeiro

**Prof. José Eugênio Leal**

Coordenador Setorial do Centro Técnico Científico —  
PUC-Rio

Rio de Janeiro, 05 de Março de 2007

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

### **André Vargas Abs da Cruz**

Graduou-se em Engenharia de Computação na Pontifícia Universidade Católica do Rio de Janeiro. Mestrado na área de Sistemas de Apoio à Decisão no Departamento de Engenharia Elétrica da Pontifícia Universidade Católica do Rio de Janeiro.

#### Ficha Catalográfica

Abs da Cruz, André V.

Algoritmos Evolutivos com Inspiração Quântica para Problemas com Representação Numérica / André Vargas Abs da Cruz; orientadores: Marco Aurélio C. Pacheco; Marley M. B. R. Vellasco. — Rio de Janeiro : PUC-Rio, Departamento de Engenharia Elétrica, 2007.

v., 109 f: il. ; 29,7 cm

1. Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica.

Inclui referências bibliográficas.

1. Engenharia Elétrica – Tese. 2. Computação Evolutiva. 3. Algoritmos Genéticos. 4. Algoritmos Culturais. 5. Algoritmos com Inspiração Quântica. I. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. II. Título.

CDD: 510

À minha esposa Luciana

## **Agradecimentos**

Ao CNPq e à PUC-Rio pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus orientadores Prof. Dr. Marco Aurélio C. Pacheco e Prof<sup>a</sup>. Dr<sup>a</sup>. Marley M. B. R. Vellasco, pelo estímulo e parceria na realização deste trabalho.

Aos meus colegas e amigos da PUC-Rio e do ICA.

Aos amigos Dilza Sczwarcman, Douglas Mota Dias, Omar Paranaíba e Wilson Freitas pela paciência ao ouvir longas conversas sobre física quântica e computação evolutiva.

Aos meus familiares e amigos que de uma forma ou de outra me estimularam e ajudaram.

Aos meus pais, pela educação, os exemplos e o apoio que me foram dados ao longo de toda a minha vida.

À minha esposa Luciana, pelo amor, carinho, generosidade e compreensão.

## **Resumo**

Abs da Cruz, André V.; . **Algoritmos Evolutivos com Inspiração Quântica para Problemas com Representação Numérica.** Rio de Janeiro, 2007. 109p. Tese de Doutorado — Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Desde que foram propostos como método de otimização, os algoritmos evolutivos têm sido usados com sucesso para resolver problemas complexos nas mais diversas áreas como, por exemplo, o projeto automático de circuitos e equipamentos, planejamento de tarefas, engenharia de software e mineração de dados, entre tantos outros. Este sucesso se deve, entre outras coisas, ao fato desta classe de algoritmos não necessitar de formulações matemáticas rigorosas a respeito do problema que se deseja otimizar, além de oferecer um alto grau de paralelismo no processo de busca. No entanto, alguns problemas são computacionalmente custosos no que diz respeito à avaliação das soluções durante o processo de busca, tornando a otimização por algoritmos evolutivos um processo lento para situações onde se deseja uma resposta rápida do algoritmo (como por exemplo, problemas de otimização *online*). Diversas maneiras de se contornar este problema, através da aceleração da convergência para boas soluções, foram propostas, entre as quais destacam-se os Algoritmos Culturais e os Algoritmos Co-Evolutivos. No entanto, estes algoritmos ainda têm a necessidade de avaliar muitas soluções a cada etapa do processo de otimização. Em problemas onde esta avaliação é computacionalmente custosa, a otimização pode levar um tempo proibitivo para alcançar soluções ótimas. Este trabalho propõe um novo algoritmo evolutivo para problemas de otimização numérica (Algoritmo Evolutivo com Inspiração Quântica usando Representação Real – AEIQ–R), inspirado no conceito de múltiplos universos da física quântica, que permite realizar o processo de otimização com um menor número de avaliações de soluções. O trabalho apresenta a modelagem deste algoritmo para a solução de problemas *benchmark* de otimização numérica, assim como no treinamento de redes neurais recorrentes em problemas de aprendizado supervisionado de séries temporais e em aprendizado por reforço em tarefas de controle. Os resultados obtidos demonstram a eficiência desse algoritmo na solução destes tipos de problemas.

## **Palavras-chave**

Computação Evolutiva. Algoritmos Genéticos. Algoritmos Culturais.  
Algoritmos com Inspiração Quântica.

## **Abstract**

Abs da Cruz, André V.; ; . **Quantum-Inspired Evolutionary Algorithms for Problems based on Numerical Representation.** Rio de Janeiro, 2007. 109p. PhD Thesis — Department of Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Since they were proposed as an optimization method, the evolutionary algorithms have been successfully used for solving complex problems in several areas such as, for example, the automatic design of electronic circuits and equipments, task planning and scheduling, software engineering and data mining, among many others. This success is due, among many other things, to the fact that this class of algorithms does not need rigorous mathematical formulations regarding the problem to be optimized, and also because it offers a high degree of parallelism in the search process. However, some problems are computationally intensive when it concerns the evaluation of solutions during the search process, making the optimization by evolutionary algorithms a slow process for situations where a quick response from the algorithm is desired (for instance, in online optimization problems). Several ways to overcome this problem, by speeding up convergence time, were proposed, including Cultural Algorithms and Coevolutionary Algorithms. However, these algorithms still have the need to evaluate many solutions on each step of the optimization process. In problems where this evaluation is computationally expensive, the optimization might take a prohibitive time to reach optimal solutions. This work proposes a new evolutionary algorithm for numerical optimization problems (Quantum-Inspired Evolutionary Algorithm for Problems based on Numerical Representation – QIEA–R), inspired in the concept of quantum superposition, which allows the optimization process to be carried on with a smaller number of evaluations. The work presents the modelling for this algorithm for solving benchmark numerical optimization problems, and for training recurrent neural networks in supervised learning and reinforcement learning. The results show the good performance of this algorithm in solving these kinds of problems.

## **Keywords**

Evolutionary Computation.    Genetic Algorithms.    Cultural Algorithms.  
Quantum–Inspired Computing.

# Sumário

1	Introdução	<b>13</b>
1.1	Motivação	13
1.2	Objetivos	15
1.3	Contribuições	15
1.4	Descrição do Trabalho	16
1.5	Organização do Trabalho	18
2	Fundamentos	<b>19</b>
2.1	Mecânica Quântica	19
2.2	Um Exemplo de um Sistema Físico Quântico – O Modelo da Partícula na Caixa	24
2.3	Computação Quântica	26
2.4	Algoritmos com Inspiração Quântica	28
2.5	Algoritmos Evolutivos com Inspiração Quântica – Representação Binária	31
2.6	Algoritmos Culturais	34
2.7	Neuro-Evolução	37
3	Algoritmos Evolutivos com Inspiração Quântica e Representação Real – AEIQ-R	<b>39</b>
3.1	O Modelo de Algoritmo Evolutivo com Inspiração Quântica e Representação Real	39
3.2	Usando o Modelo da Partícula na Caixa com o AEIQ-R	51
3.3	Analogia entre o AEIQ-R e os Algoritmos Culturais	52
3.4	Modelo Neuro-Evolutivo – Aprendizado Supervisionado	54
3.5	Modelo Neuro-Evolutivo – Aprendizado por Reforço	58
4	Estudos de Caso	<b>60</b>
4.1	Otimização de Funções	60
4.2	Neuro-Evolução	72
5	Conclusões e Trabalhos Futuros	<b>82</b>
5.1	Aprendizado Online	84
5.2	Sistemas Multi-Agentes	85
	Referências Bibliográficas	<b>87</b>

## **Lista de figuras**

2.1	Diagrama do sistema físico hipotético da “partícula na caixa”	24
2.2	Gráfico mostrando os níveis de energia e a forma das funções de onda para $n = 1$ , $n = 2$ e $n = 3$ .	26
2.3	Gráfico da função densidade de probabilidade para o sistema da “partícula na caixa” quando $n = 1$	27
2.4	Representação gráfica das probabilidades de se observar os valores 0 e 1 para um $q$ -bit qualquer	32
2.5	Pseudo–código do algoritmo evolutivo com inspiração quântica usando representação binária.	34
2.6	Pseudo–código do algoritmo cultural.	35
2.7	Diagrama da estrutura geral do algoritmo cultural.	36
3.1	Listagem completa do algoritmo evolutivo com inspiração quântica usando representação real.	40
3.2	Exemplo de um gene quântico do AEIQ–IR .	42
3.3	Genes de uma população quântica usando pulsos quadrados como função densidade de probabilidade.	45
3.4	Funções cumulativas de probabilidade associadas aos genes de uma população quântica.	46
3.5	Diagrama completo do Sistema Evolutivo com Inspiração Quântica	50
3.6	Diagrama de um modelo de partícula na caixa para uso no AEIQ–IR.	51
3.7	Diagrama do modelo de partícula na caixa para uso no AEIQ–IR após a atualização do gene quântico.	53
3.8	Diagrama mostrando uma rede neural recorrente. Este diagrama não mostra as ligações dos <i>biases</i> .	55
3.9	Modelo de Aprendizado Supervisionado usando o AEIQ–IR.	56
3.10	Modelo de Aprendizado por Reforço usando o AEIQ–IR.	58
4.1	Esforço computacional para otimização da função $f_{sphere}$ .	67
4.2	Esforço computacional para otimização da função $f_{griewank}$ .	68
4.3	Gráfico de desempenho para uma função não separável usando taxas de atualização diferentes.	70
4.4	Gráfico de desempenho para uma função separável usando taxas de atualização diferentes.	71
4.5	Representação gráfica do problema do carro na montanha.	75
4.6	Variação da velocidade (linha vermelha) e da posição (linha azul) no problema do carro na montanha.	77
4.7	Exemplo do problema do pêndulo invertido.	77
4.8	Ângulo do pêndulo com relação ao tempo.	80
4.9	Ângulo do pêndulo e velocidade do carro com relação ao tempo.	81
5.1	Diagrama de um foguete sem estabilizadores.	84
5.2	Modelo de aprendizado multi-agentes usando o AEIQ–IR .	86
5.3	Mapa 3D da função $f_1$ .	91

5.4	Mapa 3D da função $f_2.$	92
5.5	Mapa 3D da função $f_3.$	93
5.6	Mapa 3D da função $f_4.$	94
5.7	Mapa 3D da função $f_5.$	95
5.8	Mapa 3D da função $f_6.$	96
5.9	Mapa 3D da função $f_8.$	97
5.10	Mapa 3D da função $f_9.$	98
5.11	Mapa 3D da função $f_{10}.$	99
5.12	Mapa 3D da função $f_{11}.$	100
5.13	Mapa 3D da função $f_{12}.$	101
5.14	Mapa 3D da função $f_{13}.$	102
5.15	Mapa 3D da função $f_{14}.$	103
5.16	Mapa 3D da função $f_{sphere}.$	104
5.17	Mapa 3D da função $f_{ackley}.$	105
5.18	Mapa 3D da função $f_{griewank}.$	106
5.19	Mapa 3D da função $f_{rastrigin}.$	107
5.20	Mapa 3D da função $f_{schwefel}.$	108
5.21	Mapa 3D da função $f_{rosenbrock}.$	109

## **Lista de tabelas**

2.1	Números da lista a ser ordenada distribuídos pelos múltiplos universos do modelo com inspiração quântica	30
2.2	Números da lista após uma primeira ordenação realizada em cada um dos 4 universos	30
2.3	Matriz de universos após a interferência diagonal	30
2.4	Matriz de universos após a interferência vertical	30
2.5	Probabilidades de observação de cada um dos possíveis estados do indivíduo quântico.	33
3.1	Exemplo de indivíduos que formam uma população quântica $Q(t)$ em uma geração $t$ qualquer.	44
4.1	Configuração do AEIQ–IR para comparação com evolução diferencial e enxame de partículas.	62
4.2	Resultado comparativo entre o AEIQ–IR, o algoritmo de evolução diferencial e o de enxame de partículas com 1000 avaliações de função.	62
4.3	Resultado comparativo entre o AEIQ–IR, o algoritmo de evolução diferencial e o de enxame de partículas com 10000 avaliações de função.	63
4.4	Tabela que indica a melhora percentual média ao se usar o AEIQ–IR, comparando-o com o segundo melhor algoritmo, em relação às características das funções de teste.	63
4.5	Configuração 1 do AEIQ–IR para comparação com programação evolutiva clássica, programação evolutiva rápida, algoritmos genéticos convencionais e o algoritmo evolutivo com inspiração quântica usando representação binária.	65
4.6	Configuração 2 do AEIQ–IR para comparação com programação evolutiva clássica, programação evolutiva rápida, algoritmos genéticos convencionais e o algoritmo evolutivo com inspiração quântica usando representação binária.	65
4.7	Resultado comparativo entre o AEIQ–IR , o AEIQ–B (QEA), programação evolutiva rápida (FEP), programação evolutiva clássica (CEP) e algoritmos genéticos convencionais (GA). Os itens com a designação “n.d.” indicam que o dado não está disponível.	65
4.8	Parâmetros do AEIQ–IR para aprendizado supervisionado.	73
4.9	Resultados do Treinamento da Rede Neural usando <i>Backpropagation</i> .	73
4.10	Resultados do Treinamento da Rede Neural usando o AEIQ–IR .	74
4.11	Parâmetros do AEIQ–IR usados para o problema do carro na montanha.	76
4.12	Resultados para o problema do carro na montanha.	76
4.13	Parâmetros usados para o problema do pêndulo invertido.	78
4.14	Parâmetros do AEIQ–IR usados para o problema do pêndulo invertido.	79

4.15 Parâmetros usados para o problema do pêndulo invertido.	80
--------------------------------------------------------------	----

# 1

## Introdução

### 1.1

#### Motivação

Os problemas de otimização numérica são uma importante área de pesquisa, tendo aplicações em problemas como otimização de plantas industriais, mistura de produtos, refino, otimização de carteiras, mineração de dados e diversos outros (Back97, Michalewicz94).

Os algoritmos evolutivos têm sido uma importante ferramenta para a solução destes tipos de problemas. Estes métodos de otimização apresentam um bom grau de paralelismo durante o processo de busca, são facilmente adaptáveis a diversos tipos de problemas e têm bom desempenho em problemas de otimização ruidosos, descontínuos, não-diferenciáveis ou multimodais (Back97).

Os algoritmos evolutivos podem ser divididos em várias classes. Novas classes e novas técnicas dentro de cada uma das classes têm sido constantemente desenvolvidas. Alguns tipos de algoritmos evolutivos são:

- Algoritmos Genéticos (Back97, Michalewicz94)
- Programação Genética (Koza92)
- Evolução Diferencial (Storn95)
- Algoritmos Culturais (Reynolds94, Reynolds04)
- Programação Evolutiva (Yao99)

Apesar de serem utilizados com sucesso em diversos problemas de otimização, os algoritmos evolutivos apresentam, em algumas ocasiões, características que podem prejudicar o seu desempenho. Como estes algoritmos necessitam avaliar diversas vezes se as soluções encontradas por ele são adequadas, problemas onde essa avaliação seja computacionalmente custosa podem tornar proibitivo o uso dos algoritmos evolutivos. Além disso, os algoritmos evolutivos podem ter dificuldades para otimizar funções quando o cromossomo apresenta genes com problemas de epistasia (quando a “qualidade” de um gene é dependente de um outro gene).

Neste sentido, os algoritmos genéticos com inspiração quântica (Narayanan96, Han02, Han04) representam um dos mais recentes avanços na

área de computação evolutiva. Estes algoritmos se baseiam em idéias inspiradas na física quântica, em particular no conceito de superposição de estados, apresentando melhor desempenho em diversos tipos de aplicações. Mais especificamente, o algoritmo evolutivo com inspiração quântica usando representação binária (AEIQ– $\mathcal{B}$ ) (Han02) foi usado com sucesso em problemas de otimização combinatorial, apresentando resultados superiores em relação aos algoritmos genéticos convencionais em termos de tempo de convergência (e, consequentemente, em termos do número de avaliações necessárias para se atingir bons resultados). O algoritmo genético com inspiração quântica para otimização do problema do caixeiro-viajante (Narayanan96) também apresenta bons resultados quando comparado com os algoritmos evolutivos convencionais.

No entanto, nenhum dos algoritmos evolutivos com inspiração quântica preencheram uma lacuna importante: a otimização de problemas numéricos. Apesar do AEIQ– $\mathcal{B}$  poder ser utilizado para otimização neste tipo de problemas e apresentar alguns resultados satisfatórios (Han04), em geral, o uso de genes com codificação real, aliado a operadores específicos para esse tipo de codificação, produz resultados superiores, mais consistentes de experimento para experimento e com maior precisão numérica (especialmente em domínios grandes, onde a codificação binária requer uma representação proibitivamente longa) (Michalewicz94).

Um exemplo de problema onde os algoritmos evolutivos tradicionais não apresentam bom desempenho é na otimização de pesos para redes neurais. Por necessitarem realizar, em muitos casos, um número elevado de avaliações da função objetivo, os algoritmos genéticos tradicionais têm, em geral, um desempenho inferior aos algoritmos de aprendizado tradicionais usados em redes neurais (Yao99a, Ilonen03). Além disso, na área de redes neurais, normalmente necessita-se fazer diversas seqüências de aprendizado, devido ao fato de que a topologia ideal da rede neural não é, geralmente, conhecida à priori (Haykin99).

Um outro ponto importante é que, com exceção dos algoritmos culturais (Reynolds94, Reynolds04), os algoritmos evolutivos não mantêm, normalmente, conhecimento normativo sobre o espaço de busca de forma não-pontual. Em outras palavras, os algoritmos evolutivos não armazenam informações sobre as regiões mais promissoras do espaço de busca, a não ser através dos próprios indivíduos que formam a população em uma determinada geração. Esta informação armazenada nos próprios indivíduos é pontual e não pode ser compartilhada diretamente entre os indivíduos. Ao contrário, nos algoritmos culturais, esta informação é guardada no *espaço de crenças* e é diretamente compartilhada entre os indivíduos. Além disso, a informação não é pontual, mas representada por intervalos dentro do espaço de buscas que, ao longo do processo evolutivo, irá indicar as regiões mais promissoras do espaço de buscas. Este tipo de informação pode, como no caso dos algoritmos

culturais, melhorar o desempenho do algoritmo evolutivo consideravelmente. Esta informação também pode ser usada como semente em problemas de otimização *online* (onde a função que se quer otimizar varia ao longo do tempo), ao invés de (ou além de) usar um conjunto de indivíduos para este fim.

## 1.2 Objetivos

Deste modo, baseado na discussão anterior, o objetivo principal desta tese é propor um novo modelo de algoritmo evolutivo, inspirado em paradigmas da física quântica, com as seguintes características:

- *Rápida Convergência* – Como a avaliação das soluções potenciais é, em geral, a responsável pela maior parte do tempo computacional gasto com o processo de otimização, deseja-se que o novo modelo seja capaz de realizar a otimização com um número menor de chamadas à função de avaliação do problema em questão;
- *Escalabilidade* – Em muitos problemas, o número de variáveis de entrada da função que se deseja otimizar é grande. Neste sentido, deseja-se que o algoritmo de otimização seja robusto com relação à dimensionalidade do problema, ou seja, que o aumento do número de variáveis não produza um aumento exponencial no esforço computacional ao se aumentar o número de variáveis da função que está sendo otimizada;
- *Conhecimento Normativo* – É interessante que o algoritmo de otimização seja capaz de armazenar conhecimento com relação às regiões promissoras do espaço de busca. Este conhecimento pode ser usado durante o processo de otimização convencional e em problemas de otimização *online* (onde a função que está sendo otimizada varia continuamente ao longo do tempo sem, no entanto, sofrer mudanças bruscas);
- *Aprendizado Compartilhado* – Como uma consequência do conhecimento normativo, é interessante que este conhecimento possa ser compartilhado entre os indivíduos da população que está sendo evoluída.

## 1.3 Contribuições

Em função dos objetivos apresentados na seção anterior, este trabalho se concentrou na definição de um novo modelo de algoritmo evolutivo com inspiração quântica que ofereça as seguintes características:

- Representação específica para problemas de otimização numérica (representação por números reais);

- Maior velocidade de convergência quando comparado com algoritmos genéticos convencionais;
- Menor complexidade computacional com relação ao número de variáveis (dimensões) do problema que se quer otimizar;
- Mecanismos de compartilhamento de conhecimento entre indivíduos da população;

Além disso, este trabalho apresenta outras contribuições:

- Desenvolvimento de um ambiente de testes para o AEIQ–R usando a ferramenta MATLAB 7.0 nos ambientes Windows XP e Linux;
- Análise comparativa do desempenho do AEIQ–R com outros algoritmos evolutivos tradicionalmente usados em problemas de otimização;
- Análise do desempenho do algoritmo com relação à dimensionalidade do problema que se quer otimizar;
- Desenvolvimento de um algoritmo de treinamento de redes neurais recorrentes usando o AEIQ–R para problemas de previsão de séries temporais e para problemas de controle;
- Desenvolvimento de um ambiente de testes para o algoritmo de treinamento de redes neurais recorrentes usando a ferramente MATLAB 7.0 nos ambientes Windows XP e Linux;
- Análise comparativa do desempenho das redes neurais treinadas com o AEIQ–R com outros algoritmos de treinamento de redes neurais e de aprendizado por reforço.

## 1.4

### Descrição do Trabalho

Este trabalho apresenta uma nova representação para os algoritmos genéticos com inspiração quântica. Esta representação, baseada em números reais, é uma poderosa ferramenta para a otimização de problemas numéricos e se mostra mais eficiente do que os algoritmos genéticos convencionais que usam o mesmo tipo de representação para otimizar funções matemáticas. Além disso, o algoritmo também se mostra mais eficiente na otimização de problemas numéricos do que o algoritmo evolutivo com inspiração quântica que usa representação binária. Entre outras propriedades importantes deste modelo está a sua capacidade de convergir rapidamente para uma boa solução usando poucos indivíduos na sua população. Isto reduz drasticamente o número de avaliações necessárias para a otimização, o

que é um importante fator de desempenho quando o modelo está sendo usado em problemas onde a avaliação consome muito tempo de processamento.

Além de apresentar o novo modelo, este trabalho também descreve uma série de testes realizados: alguns testes de otimização utilizando funções *benchmark* para otimização numérica, onde se pretende comparar o desempenho do algoritmo proposto com outros modelos de otimização inspirados na evolução natural, incluindo métodos já consagrados e métodos mais recentes; e alguns testes no treinamento de redes neurais (neuroevolução) recorrentes para previsão de séries temporais (aprendizado supervisionado) e para tarefas de controle (aprendizado por reforço).

Em especial, com relação ao uso do algoritmo para o treinamento de redes neurais, deseja-se mostrar que, ao contrário dos métodos de descida por gradiente, usados tradicionalmente para realizar o treinamento destas redes, e que se baseiam na otimização de uma única rede, o método de neuroevolução proposto é capaz de otimizar os pesos sinápticos de várias redes neurais ao mesmo tempo. Com isto, é possível diminuir a ocorrência de problemas de estagnação do processo de aprendizado por aprisionamento em mínimos locais, sem incorrer nos problemas tradicionais do uso de algoritmos evolutivos para otimização de pesos sinápticos já que, com o uso dos algoritmos evolutivos com inspiração quântica, o número de indivíduos que precisam ser avaliados para se atingir uma solução ótima é reduzido. Em outras palavras, com o uso de algoritmos evolutivos com inspiração quântica, pode-se acelerar o aprendizado ao mesmo tempo em que se evita uma convergência prematura para soluções sub-ótimas.

O método de neuroevolução proposto também apresenta outros benefícios que não estão diretamente ligados ao fato do algoritmo ter inspiração quântica. Estes benefícios se tornam evidentes, principalmente, quando se usa estes algoritmos para o treinamento de redes neurais totalmente recorrentes (*fully recurrent neural networks*), o que permite contornar problemas encontrados em métodos tradicionais de treinamento baseados em métodos de descida por gradiente para este tipo de rede tais como (Blanco01):

- Aprendizado Recorrente em Tempo Real (*Real Time Recurrent Learning* - RTRL) – este modelo tem como principal desvantagem o alto custo computacional para executar cada iteração;
- Retropropagação pelo Tempo (*Backpropagation Through Time* – BTT) – este modelo tem como principal desvantagem a necessidade de se saber o tamanho da seqüência de entrada à priori.

Finalmente, o modelo apresentado neste trabalho permite, entre outras coisas, a definição automática da topologia da rede neural final. Isto inclui, não só o número de processadores em cada camada, como também o número total de camadas que

compõem a rede neural. Além disso, devido à sua característica probabilística, o treinamento por algoritmos evolutivos com inspiração quântica evita a ocorrência de super-treinamento da rede neural, eliminando a necessidade do uso de conjuntos de validação para o processo de aprendizado.

Este trabalho tem como metas:

- Demonstrar que o novo modelo proposto tem um desempenho melhor do que os algoritmos evolutivos clássicos em diversos problemas de otimização numérica;
- Demonstrar a aplicabilidade e as vantagens deste novo modelo no aprendizado supervisionado de redes neurais em problemas de previsão e controle;
- Apresentar discussões sobre as características principais deste novo modelo e os resultados obtidos nos estudos de caso.

## 1.5

### Organização do Trabalho

Este trabalho está dividido da seguinte forma:

- No capítulo 2 é apresentado um resumo dos fundamentos teóricos necessários para a compreensão deste trabalho;
- No capítulo 3 o modelo proposto de algoritmo evolutivo com inspiração quântica e representação real é apresentado em detalhes;
- O capítulo 4 mostra os resultados obtidos em diversos estudos de caso;
- O capítulo 5 apresenta discussões sobre o modelo e os resultados encontrados, e aponta direções para trabalhos futuros.

## 2

# Fundamentos

### 2.1

#### Mecânica Quântica

##### 2.1.1

##### Introdução

Quando a luz incide sobre uma superfície de metal, elétrons são arrancados desta mesma superfície. Este problema não pode ser perfeitamente explicado pela mecânica clássica devido a três fatores (Green01):

- Não importa o quanto baixa é a intensidade da luz, os elétrons são arrancados instantaneamente da superfície do metal. De acordo com a física clássica, os elétrons precisariam acumular durante um certo tempo a energia necessária para escapar da superfície do metal, já que a luz deveria ter sua energia distribuída uniformemente ao longo da onda;
- Existe uma freqüência de corte para o feixe incidente de luz, abaixo da qual nenhum elétron é emitido. De acordo com as leis de Maxwell, a energia transportada pela luz depende apenas da amplitude da onda eletromagnética e não da freqüência;
- Se a energia cinética máxima dos elétrons emitidos for relacionada através de um gráfico com a freqüência da luz incidente, uma linha reta será obtida. Como a energia deveria estar relacionada apenas com a amplitude da onda, de acordo com as leis de Maxwell, não deveria existir uma relação linear entre freqüência e energia.

A explicação para estes fenômenos foi proposta em 1900 por Planck (Planck01, Green01) ao abandonar o conceito clássico para o comportamento da luz apenas como uma onda eletromagnética. Planck demonstrou que, se a energia da luz estiver concentrada em pacotes (ou *quanta*) de energia  $E = hv$  (onde  $v$  é a freqüência da luz e  $h$  é a constante de Planck), os problemas descritos anteriormente podem ser, então, facilmente explicados. Através desta explicação, a luz passa a ter, em determinadas situações, um comportamento de uma partícula (chamada *fóton*).

### 2.1.2

#### Funções de Onda

A idéia de que a luz se comporta em alguns casos como uma onda e em alguns casos como uma partícula leva imediatamente à questão sobre se o fóton tem outras propriedades comuns às partículas. Como energia e massa estão relacionadas pela equação  $E = mc^2$ , o fóton (que se move com a velocidade da luz  $c$ ) tem uma massa relativística  $m = h\nu/c^2$ . Portanto, como o fóton tem massa e velocidade, também deve ter um momento  $p = mc = h\nu/c = h/\lambda$ , onde  $\lambda$  é o comprimento de onda da luz.

De maneira inversa, assim como os fótons têm um momento característico  $h/\lambda$ , outras partículas (por exemplo, um elétron), que também têm um momento característico, devem possuir um comprimento de onda relacionado a este momento  $\lambda = h/p$ . Se uma partícula tem um comprimento de onda característico, então deve ser possível representar este comprimento de onda matematicamente da mesma maneira que uma onda é representada, ou seja, usando-se uma função de onda. Na física clássica, uma onda estacionária com comprimento de onda  $\lambda$ , propagando-se na direção positiva do eixo  $x$ , pode ser representada pela equação 2-1 (Green01).

$$\psi(x) = \exp(i2\pi x/\lambda) = \cos(2\pi x/\lambda) + i \sin(2\pi x/\lambda) \quad (2-1)$$

Onde  $i = \sqrt{-1}$ . Substituindo  $\lambda = h/p$ , obtém-se a equação (2-2).

$$\psi(x) = \exp(ipx/\hbar) \quad (2-2)$$

Onde  $\hbar = h/2\pi$ . A partir da função de onda, é possível determinar a probabilidade de que a partícula esteja em um determinado ponto do espaço ao se tentar observá-la. A densidade de probabilidade para a localização de uma partícula com função de onda  $\psi$  deve ser o quadrado da amplitude da função de onda, ou seja,  $|\psi|^2$  (Gillespie73, Green01). Em uma dimensão, uma partícula, representada pela função de onda  $\psi(x)$ , terá uma densidade de probabilidade de ser encontrada no intervalo  $x + dx$  igual a:

$$|\psi(x)|^2 dx = \psi^* \psi dx \quad (2-3)$$

Onde  $\psi^*$  representa o conjugado complexo de  $\psi$ . Em três dimensões, a densidade de probabilidade da partícula é  $|\psi(\mathbf{r})|^2$  (onde  $\mathbf{r}$  é um vetor de três dimensões  $(x, y, z)$ ). Isto significa que a probabilidade de se encontrar a partícula no elemento infinitesimal de volume  $d\tau = dxdydz$  no ponto  $\mathbf{r}$  é  $|\psi(\mathbf{r})|^2 d\tau$ . A integral deste valor sobre todo o espaço onde a partícula pode ser encontrada fornece a probabilidade de se encontrar esta partícula em algum lugar do espaço: consequentemente, este valor deve ser igual a 1 (equação 2-4).

$$\int_{-\infty}^{\infty} |\psi|^2 d\tau = 1 \quad (2-4)$$

As funções de onda introduzem um conceito importante e pouco intuitivo: a relação entre a função de onda e a localização da partícula é, por natureza, probabilística (Green01). Em um feixe de luz, por exemplo, a função de onda está espalhada por toda a frente de onda de forma homogênea mas, quando se tenta detectar os fótons, nota-se que os mesmos chegam de forma imprevisível ao detector, com uma densidade de probabilidade proporcional à intensidade do feixe de luz ou proporcional ao quadrado da amplitude da onda. Em outras palavras, o valor das propriedades da partícula que podem ser representadas por funções de onda (por exemplo, a posição da partícula) não pode ser conhecido à priori; sem efetivamente medir a propriedade que se deseja, tudo que se pode afirmar sobre esta propriedade é que ela tem diferentes probabilidades de assumir diferentes valores quando for observada. Assim, antes de se observar através de algum instrumento, por exemplo, a posição de um elétron confinado dentro de alguma região do espaço, o máximo que se pode determinar é a probabilidade de observá-lo em alguma região deste espaço de confinamento.

As funções de onda devem obedecer algumas propriedades. Estas propriedades são:

1.  $\psi$  deve ser finita;
2.  $\psi$  deve ser contínua;
3.  $\psi$  deve ser derivável duas vezes;
4.  $\psi$  deve ser integrável em todo o espaço.

### 2.1.3 Operadores Hermitianos

As informações contidas nas funções de onda são as únicas de que se pode dispor a respeito de um sistema físico microscópico. Como já foi visto, é possível manipular estas funções de onda de modo a se obter a distribuição de probabilidade espacial do sistema físico representado por esta função. No entanto, esta informação espacial não é a única que se deseja obter sobre o sistema físico. Deseja-se também ser capaz de medir quantidades como o momento, a energia e o momento angular de uma partícula. Apesar da função de onda não poder ser medida diretamente, todas essas propriedades mecânicas podem ser medidas diretamente e são chamadas, na mecânica quântica, de *observáveis* (em inglês, *observables*) (Gillespie73). Os observáveis são representados matematicamente por *operadores*. Alguns exemplos de observáveis são a posição e o momento de uma partícula.

Um operador é um objeto matemático que age sobre uma função transformando-a em outra função. Um exemplo é o operador  $\hat{d}_x$ , que realiza a

operação de “diferenciar com respeito a variável  $x$ ”. Este operador transforma uma função  $f(x)$  em sua derivada. Neste trabalho, todos os operadores serão distinguidos de outros objetos matemáticos através do uso de um sinal  $\hat{\phantom{a}}$  sobre o operador.

Apesar de, em geral, os operadores transformarem uma função em outra, em alguns casos especiais um operador pode transformar uma função nela mesma. Um exemplo é o operador  $\hat{\partial}_x$  que transforma a função  $e^{\alpha x}$  em  $\alpha e^{\alpha x}$ , ou seja, a mesma função multiplicada por uma constante  $\alpha$ . A função que é transformada nela mesma é chamada de autofunção (ou autovetor) e a constante multiplicativa introduzida pelo operador é chamada de autovalor.

Os conceitos de autofunção e autovalor são extremamente importantes na mecânica quântica. Isto se deve ao fato de que todos os observáveis são representados por operadores, sendo que os autovalores, os quais podem ser encontrados através da aplicação de um operador a uma função, são os únicos valores que podem efetivamente ser observados ao se fazer uma medida. Em outras palavras, qualquer medição de um observável deve encontrar um dos possíveis autovalores do operador (Gillespie73). Por exemplo, os níveis de energia de um átomo são autovalores do operador de energia; qualquer medida realizada para se determinar o nível de energia de um átomo deve encontrar um desses autovalores. No entanto, é importante destacar que, mesmo na mecânica quântica, os valores possíveis para os autovalores de um operador não precisam, necessariamente, ser discretos, ou seja, é possível que para certos observáveis, os autovalores do operador correspondente formem um conjunto contínuo.

Os operadores que representam quantidades observáveis devem respeitar duas restrições: linearidade e hermiticidade. Um operador  $\hat{A}$  é linear quando, para quaisquer valores constantes  $\alpha$  e  $\beta$ , e para quaisquer funções de onda  $\psi(x)$  e  $\phi(x)$ , a equação 2-5 for válida.

$$\hat{A}(\alpha\psi(x) + \beta\phi(x)) = \alpha\hat{A}\psi(x) + \beta\hat{A}\phi(x) \quad (2-5)$$

Um operador  $\hat{A}$  é hermitiano se, para quaisquer funções de onda  $\psi(x)$  e  $\phi(x)$ , a equação 2-6 for válida.

$$\int_{-\infty}^{\infty} \psi^*(x)\hat{A}\phi(x)dx = \int_{-\infty}^{\infty} \hat{A}\psi^*(x)\phi(x)dx \quad (2-6)$$

Duas importantes consequências advêm destas restrições (Gillespie73): a primeira é que os autovalores de um operador hermitiano são números reais; a segunda é que as autofunções de um operador hermitiano correspondentes a autovalores diferentes, são ortogonais.

Supondo-se então que um observável  $\mathcal{A}$  é representado por um operador  $\hat{A}$  e que uma função de onda  $\psi$  é uma autofunção de  $\hat{A}$  com autovalor  $\alpha$  (ou seja,  $\hat{A}\psi = \alpha\psi$ ), então o sistema tem um valor definido  $\alpha$  para o observável  $\mathcal{A}$ . Uma

medida de  $\mathcal{A}$  irá resultar no valor  $\alpha$  com probabilidade 1. Por outro lado, se a função de onda  $\psi$  não for uma autofunção de  $\hat{A}$ , então uma medida de  $\mathcal{A}$  pode produzir qualquer um de diferentes autovalores do operador  $\hat{A}$  com diferentes probabilidades. O valor esperado de um observável  $\mathcal{A}$ , denotado por  $\langle A \rangle$ , é definido pela equação 2-7.

$$\langle A \rangle = \int \psi^* \hat{A} \psi d\tau \quad (2-7)$$

De modo a simplificar a notação matemática, pode-se usar a notação de Dirac para estes problemas (Green01). Neste caso, a função de onda  $\psi$  passa a ser representada pelo símbolo  $|\psi\rangle$  (chamado *ket*). O conjugado complexo  $\psi^*$  da função de onda é representado pelo símbolo  $\langle\psi|$  (chamado *bra*). As integrais do tipo  $\int \phi^* \psi d\tau$  são representadas pelo símbolo  $\langle\phi|\psi\rangle$  (é importante notar que este símbolo implica na integração da função, ou seja,  $\langle\phi|\psi\rangle = \int \phi^* \psi d\tau$ ). Portanto, uma função de onda  $\psi$  está normalizada se  $\langle\psi|\psi\rangle = 1$ .

A principal característica que se deseja ressaltar com relação às funções de onda e aos operadores, é a possibilidade de se representar as funções de onda como uma expansão por autofunções. Em outras palavras, além de serem ortogonais, as autofunções de um operador hermitiano formam uma base ortonormal completa que permite escrever qualquer função de onda para o sistema físico como uma superposição (combinação linear) destas autofunções. Ou seja, se um operador  $\hat{A}$  possui um conjunto de autofunções  $|\psi_i\rangle$  com autovalores  $\alpha_i$  (ou seja,  $\hat{A}|\psi_i\rangle = \alpha_i|\psi_i\rangle$ ), então qualquer função de onda  $\phi$  válida para o sistema pode ser representada pela superposição de autofunções  $|\psi_i\rangle$  conforme mostrado na equação 2-8.

$$\phi = \sum_i c_i |\psi_i\rangle \quad (2-8)$$

Onde  $c_i^* c_i$  (ou  $|c_i|^2$ ) é a probabilidade de se medir o autovalor  $\alpha_i$ . Isto significa que, ao fazer uma medida de um estado quântico, a função de onda é projetada em uma das autofunções que formam a base ortonormal e o autovalor correspondente a esta autofunção é o valor observado. Em outras palavras, a função de onda pode ser escrita como uma combinação linear dos autovetores associados à um determinado operador. Os coeficientes associados aos autovetores nesta combinação linear permitem determinar a probabilidade de que a função de onda seja projetada sobre cada um dos autovetores quando a observação for realizada. Vale ressaltar que, como  $|c_i|^2$  representa a probabilidade de um autovalor ser observado, e como a base ortonormal é completa,  $\sum_i |c_i|^2 = 1$ .

## 2.2

## Um Exemplo de um Sistema Físico Quântico – O Modelo da Partícula na Caixa

Esta seção tem como objetivo mostrar um exemplo de um sistema físico quântico hipotético (hipotético no sentido de não ser possível reproduzir as condições ideais nas quais o experimento deve ser realizado) de modo a ilustrar os conceitos da mecânica quântica apresentados anteriormente. Além disso, o sistema físico aqui apresentado será usado, posteriormente, como um modelo físico que permite implementar o algoritmo genético com inspiração quântica apresentado neste trabalho.

Considere-se uma partícula (um elétron por exemplo) confinada dentro de uma caixa cujas paredes têm um potencial infinito. Em outras palavras, não é permitido que a partícula saia de dentro da caixa onde a mesma está confinada. Além disso, o movimento desta partícula está limitado a uma dimensão (o sentido da largura da caixa). Um diagrama deste modelo pode ser visto na figura 2.1 (nesta figura,  $V = \infty$  indica a região de potencial infinito).

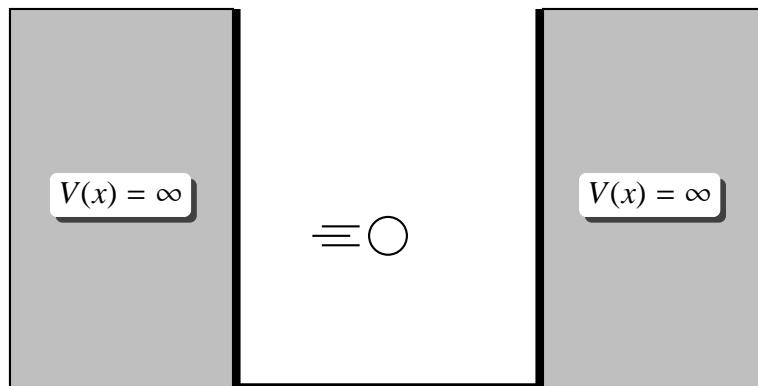


Figura 2.1: Diagrama do sistema físico hipotético da “partícula na caixa”

Para o problema unidimensional (ou seja, a partícula pode se mover apenas sobre o eixo  $x$ ), pode-se usar a equação de Schrödinger independente do tempo (Green01) para calcular a função de onda. Esta equação pode ser escrita como em 2-9.

$$-\frac{\hbar^2}{2m} \frac{d^2\psi(x)}{dx^2} + V(x)\psi(x) = E\psi(x) \quad (2-9)$$

Onde  $\hbar$  é a constante de Planck reduzida,  $m$  é a massa da partícula,  $\psi(x)$  é a função de onda estacionária da partícula (esta é a função que se deseja encontrar),  $V(x)$  determina o valor da energia potencial em função da posição e  $E$  é a energia da partícula.

Supondo-se que se deseja encontrar a função de onda para uma partícula em uma situação onde as barreiras de potencial estejam posicionadas, respectivamente, em  $x = 0$  e  $x = L$ , pode-se considerar que a equação anterior, dentro do intervalo  $[0, L]$  para  $x$  assume a forma mostrada em 2-10, já que o potencial dentro dessa região é igual a zero.

$$-\frac{\hbar^2}{2m} \frac{d^2\psi(x)}{dx^2} = E\psi(x) \quad (2-10)$$

Esta equação diferencial é bem conhecida e sua solução é da forma:

$$\psi(x) = A \sin(kx) + B \cos(kx) \quad (2-11)$$

$$E = \frac{k^2\hbar^2}{2m} \quad (2-12)$$

Nestas equações,  $A$  e  $B$  podem ser quaisquer números complexos e o número  $k$  deve ser um número real (já que a energia  $E$  deve ser um número real). De modo a definir a solução específica para este problema, deve-se determinar as condições de contorno e encontrar os valores de  $A$  e  $B$  que satisfaçam essas condições. Como já foi mencionado anteriormente, a equação deve garantir que a função de onda seja contínua ao longo de todo o eixo real. Como a partícula não pode estar na região em que a barreira de potencial é infinita, deduz-se que, quando  $x$  tende a 0 ou quando  $x$  tende a  $L$ , a probabilidade  $|\psi(x)|^2$  de encontrar uma partícula nesta região deve tender a zero. Isto está de acordo com a intuição física já que, ao se aproximar da barreira de potencial, a repulsão sobre a partícula aumenta. Assim,  $\psi(0) = \psi(L) = 0$ . De acordo com a equação 2-11, para que  $\psi(0) = 0$  o termo com o cosseno deve desaparecer (já que  $\sin(0) = 0$  e  $\cos(0) = 1$ ) e, portanto,  $B$  deve ser igual a zero. Já para a situação em que  $x = L$ ,  $\psi(L) = A \sin(kL) = 0$ . Para o caso não-trivial em que  $A \neq 0$ ,  $\sin(kL) = 0$  somente se  $k = n\pi/L$ , onde  $n \in \mathbb{N}^+$ .

Finalmente, para se encontrar o valor de  $A$ , deve-se lembrar que a função de onda deve ser normalizada, conforme mostrado na equação 2-4.

$$1 = \int_{-\infty}^{\infty} |\psi(x)|^2 dx = |A|^2 \int_0^L \sin^2(kx) dx = |A|^2 \frac{L}{2} \quad (2-13)$$

Rearranjando a equação anterior tem-se:

$$|A| = \sqrt{\frac{2}{L}} \quad (2-14)$$

Assim,  $A$  pode ser qualquer número complexo cujo valor absoluto  $|A|$  seja igual a  $\sqrt{2/L}$ . Como diferentes valores de  $A$  produzem o mesmo estado físico, e para fins de simplificação, define-se  $A = \sqrt{2/L}$ .

Substituindo-se estes valores nas equações 2-11 e 2-12 chega-se as equações:

$$\psi_n(x) = \sqrt{\frac{2}{L}} \sin\left(\frac{n\pi x}{L}\right) \quad (2-15)$$

$$E_n = \frac{n^2\hbar^2\pi^2}{2mL^2} = \frac{n^2\hbar^2}{8mL^2} \quad (2-16)$$

Onde  $n \in \mathbb{N}^+$ . O valor de  $n$  determina o número de ciclos da senóide, como pode ser visto na figura 2.2.

A função de onda deve ser estacionária e, por este motivo, o número  $n$  deve ser inteiro e maior do que zero. Isto, juntamente com a equação 2-16, explica porque

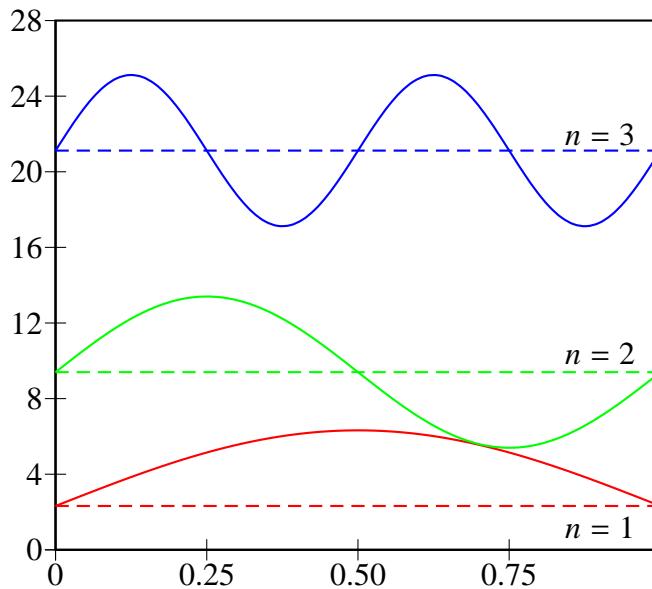


Figura 2.2: Gráfico mostrando os níveis de energia e a forma das funções de onda para  $n = 1$ ,  $n = 2$  e  $n = 3$ .

os níveis de energia da partícula são discretos.

Por sua vez, a função densidade de probabilidade  $p(x)$  para este sistema físico é dada pela equação:

$$p(x) = |\psi(x)|^2 = \frac{2}{L} \sin^2\left(\frac{n\pi x}{L}\right) \quad (2-17)$$

O gráfico desta equação quando  $n = 1$  é mostrado na figura 2.3.

O resultado observado neste gráfico é o mesmo esperado intuitivamente. A partícula tem mais probabilidade de ser observada na parte central da caixa do que nas bordas da mesma (onde a distância da barreira de potencial infinito começa a ser menor).

## 2.3 Computação Quântica

Um computador quântico é um dispositivo que faz uso direto de certos fenômenos da mecânica quântica para realizar operações com dados. Estes fenômenos permitem construir, em teoria, computadores que obedecem novas leis mais permissivas de complexidade computacional(Spector04). O termo “computação quântica” é, portanto, usado para descrever processos computacionais que se baseiam nestes fenômenos específicos e que são, assim, capazes de diminuir o esforço e a complexidade para se resolver determinados problemas. A principal perspectiva que se abre com o uso de computadores quânticos em relação ao seu poder de processa-

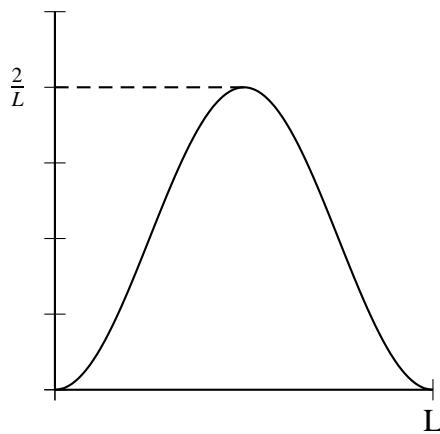


Figura 2.3: Gráfico da função densidade de probabilidade para o sistema da “partícula na caixa” quando  $n = 1$

mento é o fato de que “as possibilidades contam, mesmo que elas nunca venham a acontecer” (Spector04).

Em um computador clássico, o *bit* é a menor unidade de informação, podendo assumir os valores 0 ou 1. Em um computador quântico, a unidade de informação básica, chamada de *q-bit*, pode assumir os estados  $|0\rangle$ ,  $|1\rangle$  ou uma superposição dos dois estados. Esta superposição de dois estados é uma combinação linear dos estados  $|0\rangle$  e  $|1\rangle$  e pode ser representada pela equação 2-18.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (2-18)$$

Onde  $|\psi\rangle$  é o estado do *q-bit* e, conforme discutido na seção anterior,  $|\alpha|^2$  e  $|\beta|^2$  são as probabilidades de que os estados 0 ou 1 sejam observados quando o *q-bit* for medido.

Ao ser observado, o bit quântico será trazido para o nível clássico e o estado observado será o valor 0 ou 1. Esta superposição de estados oferece um grau de paralelismo incomparável aos computadores quânticos que, se bem explorado, permite que estes computadores realizem tarefas praticamente impossíveis de serem realizadas em computadores clássicos. Um exemplo de uma tarefa deste tipo é a fatoração inteira de grandes números. Neste problema, deseja-se encontrar os dois números primos cujo produto seja igual a um número  $x$  fornecido. Esta tarefa pode levar milhões de anos para ser resolvida em computadores clássicos usando os algoritmos conhecidos mais sofisticados. Em um computador quântico, no entanto, a mesma tarefa levaria apenas alguns segundos para ser concluída. Uma discussão mais detalhada sobre o funcionamento dos computadores quântico pode ser encontrada em (Rieffel00).

## 2.4

### Algoritmos com Inspiração Quântica

Embora a computação quântica ofereça uma boa promessa em termos de capacidade de processamento, dois problemas impedem, atualmente, que a mesma se torne uma ferramenta útil: a dificuldade de se implementar um computador quântico e a dificuldade de se criar algoritmos que tirem proveito da capacidade de processamento destes computadores. Deste modo, as pesquisas na área de computação quântica se concentram, atualmente, em dois pontos:

- Desenvolvimento de novos algoritmos que sejam mais eficientes nos computadores quânticos do que os algoritmos equivalentes para computadores clássicos;
- Desenvolvimento de um *hardware* que torne viável o uso dos computadores quânticos.

No entanto, no artigo (Moore95), uma nova abordagem é proposta. Ao invés de desenvolver novos algoritmos para computadores quânticos ou de tentar viabilizar o uso destes, a idéia de *computação com inspiração quântica* é apresentada. O objetivo desta nova abordagem é criar algoritmos clássicos (capazes, portanto, de serem executados em computadores clássicos) que tirem proveito dos paradigmas da física quântica, de forma a melhorar o desempenho dos mesmos na resolução de problemas.

Uma metodologia para a formulação inicial de um algoritmo com inspiração quântica também é apresentada em (Moore95). Esta metodologia consiste dos seguintes passos:

1. O problema deve ter uma representação numérica ou, caso não tenha, um método para sua conversão em representação numérica deve ser empregado;
2. A configuração inicial deve ser determinada;
3. Uma condição de parada deve ser definida;
4. O problema deve poder ser dividido em sub-problemas menores;
5. O número de universos (ou estados de superposição) deve ser identificado;
6. Cada sub-problema deve ser associado a um dos universos;
7. Os cálculos nos diferentes universos devem ocorrer de forma independente;
8. Alguma forma de iteração entre os múltiplos universos deve existir. Esta interferência deve, ou permitir encontrar a solução para o problema, ou

fornecer informação para cada sub-problema em cada universo ser capaz de encontrá-la.

Um exemplo de um algoritmo com inspiração quântica é o “Algoritmo de Ordenação com Inspiração Quântica” (Moore95), baseado no algoritmo de “ordenação por seleção”. Este algoritmo é usado para resolver o problema de ordenação numérica onde deseja-se ordenar uma lista com  $n$  elementos em ordem ascendente ou descendente. O algoritmo funciona da seguinte maneira:

1. Denomina-se a lista que se quer ordenar por  $L_1$  (esta lista deve conter pelo menos dois elementos) e inicializa-se uma segunda lista  $L_2$  sem nenhum elemento;
2. Encontra-se o menor elemento (caso se esteja ordenando de forma ascendente) em  $L_1$  e após remover este elemento da lista  $L_1$  acrescenta-se o mesmo à lista  $L_2$ ;
3. Repete-se o passo anterior até que  $L_1$  esteja vazia.  $L_2$  contém agora todos os elementos em ordem crescente.

Quando implementado de maneira eficiente, o tempo de execução deste algoritmo é da ordem de  $\frac{1}{2}N(N - 1)$  (Knuth73), onde  $N$  é o tamanho inicial de  $L_1$ . Por exemplo, caso se queira ordenar uma lista com 16 valores, serão necessários, em média, 120 buscas à lista vezes para encontrar o menor elemento. Usando elementos inspirados na física quântica, este algoritmo pode ser melhorado da seguinte maneira (Moore95) (usa-se aqui, como exemplo, uma lista  $L_1$  contendo 16 elementos  $L_1 = (16, 14, 12, 3, 8, 4, 15, 7, 11, 6, 2, 5, 1, 10, 9, 13)$ ):

1. Fixar  $n =$  número de elementos na lista = 16;
2. Fixar  $u =$  número de superposições (universos) necessárias =  $\sqrt{n} = 4$ ;
3. Fixar  $l =$  número de elementos da lista em cada universo =  $u = 4$ ;
4. Divide-se a lista  $L_1$  em 4 partes: os primeiros quatro números são colocados no universo 1, os próximos 4 números são colocados no universo 2, e assim por diante. Desta forma, para este exemplo, pode-se construir a tabela 2.1 com os universos e os elementos que estão armazenados em cada um deles:
5. Usa-se o algoritmo de ordenação por seleção em cada um dos universos, separadamente. Deste modo, obtém-se a tabela 2.2.

$u_0$	16	14	12	3
$u_1$	8	4	15	7
$u_2$	11	6	2	5
$u_3$	1	10	9	13

Tabela 2.1: Números da lista a ser ordenada distribuídos pelos múltiplos universos do modelo com inspiração quântica

$u_0$	3	12	14	16
$u_1$	4	7	8	15
$u_2$	2	5	6	11
$u_3$	1	9	10	13

Tabela 2.2: Números da lista após uma primeira ordenação realizada em cada um dos 4 universos

6. Realiza-se uma *interferência diagonal*: os universos são rearranjados pegando-se as diagonais que têm, como primeiro elemento, os números da primeira coluna da matriz. Estes elementos são reordenados e reposicionados na matriz. Em outras palavras, o primeiro universo irá conter os elementos da diagonal principal (3, 7, 6, 13), o segundo universo irá conter os elementos formados pela diagonal que se inicia na primeira coluna da segunda linha (4, 5, 10, 16), o terceiro universo irá conter os elementos formados pela diagonal que se inicia na primeira coluna da terceira linha (2, 9, 14, 15) e, finalmente, o quarto universo irá conter os elementos formados pela diagonal que se inicia na primeira coluna da quarta linha (1, 12, 8, 11). A matriz obtida por essa interferência diagonal é novamente processada pelo algoritmo de ordenação, separadamente por universo. Assim, forma-se a tabela 2.3.

$u_0$	3	6	7	13
$u_1$	4	5	10	16
$u_2$	2	9	14	15
$u_3$	1	8	11	12

Tabela 2.3: Matriz de universos após a interferência diagonal

7. Em seguida, realiza-se a *interferência vertical*, onde cada coluna da matriz de universos é ordenada e transposta. Com isto, obtém-se a tabela 2.4.

$u_0$	1	2	3	4
$u_1$	5	6	8	9
$u_2$	7	10	11	14
$u_3$	12	13	15	16

Tabela 2.4: Matriz de universos após a interferência vertical

8. Checa-se o último valor do universo  $u_i$  e verifica-se se é menor ou igual ao primeiro valor de  $u_{i+1}$  para  $i = 0, 1, \dots, n - 1$ . Caso não seja, repete-se os passos 6 e 7 novamente até que esta condição seja satisfeita.

O tempo de execução deste algoritmo é da ordem de  $\sqrt{N}[N + 2(\sqrt{N} - 1)]$  (Moore95). Portanto, uma lista com 16 elementos precisará, em média, de 88 buscas na lista usando-se a ordenação por seleção, o que é significativamente menor do que o número de buscas (120) necessárias para a execução do algoritmo sem inspiração quântica.

No artigo (Moore95), também é apresentado um algoritmo com inspiração quântica para determinar a solução para o problema do *15-puzzle*. Uma explicação mais detalhada sobre estes algoritmos e sobre os algoritmos com inspiração quântica pode ser encontrada em (Moore95).

## 2.5

### Algoritmos Evolutivos com Inspiração Quântica – Representação Binária

O algoritmo evolutivo com inspiração quântica usando representação binária, que neste trabalho será designado pela sigla AEIQ–B (*Quantum–Inspired Evolutionary Algorithm* – QEA na sigla original), foi proposto inicialmente em (Han00). Neste modelo, o algoritmo evolutivo proposto é também caracterizado por um cromossomo, uma função de avaliação e uma dinâmica populacional. Entretanto, ao invés de uma representação binária convencional, este algoritmo usa uma representação especial que simula um cromossomo formado por  $q$ -bits. Esta representação é feita da seguinte forma: um  $q$ -bit é formado por um par de números  $(\alpha, \beta)$ , como na equação 2-19.

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad (2-19)$$

Onde  $|\alpha|^2 + |\beta|^2 = 1$ . O valor dado por  $|\alpha|^2$  indica a probabilidade de que o  $q$ -bit terá o valor 0 quando for observado e o valor  $|\beta|^2$  indica a probabilidade de que o  $q$ -bit terá o valor 1 quando for observado. Pode-se visualizar graficamente esta relação entre  $\alpha$  e  $\beta$ , conforme mostrado na figura 2.4.

A partir da definição do  $q$ -bit acima, pode-se definir um indivíduo quântico formado por  $m$   $q$ -bits, conforme a equação 2-20.

$$\begin{bmatrix} \alpha_1 & | & \alpha_2 & | & \dots & | & \alpha_m \\ \beta_1 & | & \beta_2 & | & \dots & | & \beta_m \end{bmatrix} \quad (2-20)$$

De forma semelhante à mostrada na equação 2-19,  $|\alpha_i|^2 + |\beta_i|^2 = 1$  onde  $i = 1, 2, 3, \dots, m$ .

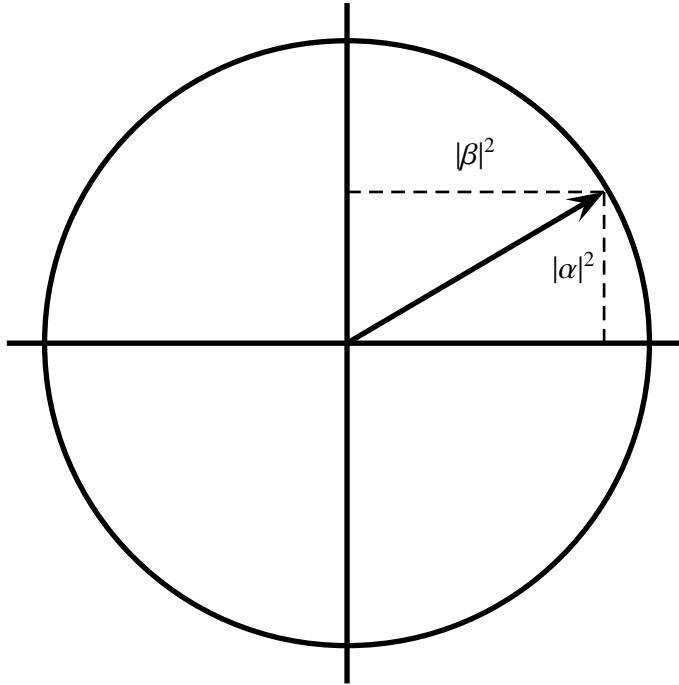


Figura 2.4: Representação gráfica das probabilidades de se observar os valores 0 e 1 para um  $q$ -bit qualquer

Deste modo, cada indivíduo quântico representa uma superposição de indivíduos formados por  $m$  genes. Seja, por exemplo, um indivíduo quântico formado por 3  $q$ -bits com os valores mostrados na equação 2-21.

$$\left[ \begin{array}{c|c|c} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} \end{array} \right] \quad (2-21)$$

Para se calcular a *amplitude de probabilidade* (o módulo da amplitude de probabilidade ao quadrado é igual à probabilidade de se observar o estado) do estado  $|000\rangle$ , multiplica-se as amplitudes de probabilidade de se observar os estados 0 em cada um dos bits ( $\alpha_1$ ,  $\alpha_2$  e  $\alpha_3$ ). Para calcular a amplitude de probabilidade de se observar o estado  $|001\rangle$ , multiplica-se as amplitudes de probabilidade de se observar estes bits ( $\alpha_1$ ,  $\alpha_2$  e  $\beta_3$ ). Estendendo-se isto para as outras possibilidades de observação dos estados, a superposição dos mesmos pode ser representada pela equação 2-22.

$$\frac{1}{4}|000\rangle + \frac{\sqrt{3}}{4}|001\rangle + \frac{1}{4}|010\rangle + \frac{\sqrt{3}}{4}|011\rangle + \frac{1}{4}|100\rangle + \frac{\sqrt{3}}{4}|101\rangle + \frac{1}{4}|110\rangle + \frac{\sqrt{3}}{4}|111\rangle \quad (2-22)$$

Este resultado significa que, cada um dos possíveis estados deste indivíduo, tem a probabilidade de ser observado mostrada na tabela 2.5.

O AEIQ- $\mathcal{B}$ , proposto em (Han00), é definido como na figura 2.5. Neste algoritmo,  $Q(t)$  é uma população com um ou mais indivíduos quânticos. No passo

Estado	Probabilidade
000	$\frac{1}{16}$
001	$\frac{3}{16}$
010	$\frac{1}{16}$
011	$\frac{3}{16}$
100	$\frac{1}{16}$
101	$\frac{3}{16}$
110	$\frac{1}{16}$
111	$\frac{3}{16}$

Tabela 2.5: Probabilidades de observação de cada um dos possíveis estados do indivíduo quântico.

de inicialização, estes indivíduos são inicializados de modo que os valores  $\alpha_i$  e  $\beta_i$  (onde  $i = 1, 2, 3, \dots, j$  e  $j$  é o comprimento do indivíduo quântico) sejam iguais a  $\frac{1}{\sqrt{2}}$ . Isto significa que, na geração inicial, todos os genes dos indivíduos terão a mesma probabilidade de gerarem os estados 0 ou 1.

A atualização da população, conforme indicado no algoritmo da figura 2.5, é realizada pelo operador  $q - gate$ , o qual é definido como a matriz de rotação dada pela equação 2-23.

$$U(\Delta\theta_i) = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \quad (2-23)$$

Esta matriz deve ser usada para multiplicar cada uma das colunas do indivíduo quântico, ou seja, os valores  $\alpha_i$  e  $\beta_i$  são tratados como um vetor bidimensional e são rotacionados usando a matriz especificada. Em outras palavras, considerando-se o indivíduo quântico  $Q(t) = \{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \dots, (\alpha_i, \beta_i), \}$  (onde  $i = 1, 2, 3, \dots, j$  e  $j$  é o comprimento do indivíduo quântico), a atualização deste indivíduo é feita de acordo com a equação 2-24.

$$(\alpha'_i, \beta'_i) = U(\Delta\theta_i) \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \quad (2-24)$$

Onde  $i = 1, 2, 3, \dots, j$ . Os valores de  $\Delta\theta$  são definidos através de uma tabela, de modo que esta matriz de rotação seja capaz de modificar os valores de  $\alpha_i$  e  $\beta_i$  aumentando as chances de que os indivíduos com as melhores avaliações sejam observados.

Finalmente, a estrutura  $B(t)$  é usada para armazenar os melhores indivíduos gerados pelo algoritmo ao longo do processo evolutivo. Os últimos dois passos do algoritmo servem para armazenar os melhores indivíduos gerados na população atual com os melhores indivíduos criados nas gerações anteriores.

Este algoritmo com representação binária foi usado com sucesso em problemas de otimização combinatorial (Han00, Han02) e detecção de faces (Jang04), apresentando resultados superiores aos algoritmos genéticos convencionais em ter-

```

iniciar
     $t \leftarrow 0;$ 
    inicializa  $Q(t)$ 
    gera  $P(t)$  observando estados de  $Q(t)$ 
    avalia  $P(t)$ 
    armazena as melhores soluções de  $P(t)$  em  $B(t)$ 
    enquanto não ocorrer condição de parada
         $t \leftarrow t + 1$ 
        gera  $P(t)$  observando estados de  $Q(t - 1)$ 
        avalia  $P(t)$ 
        atualiza  $Q(t)$  usando q-gate
        armazena as melhores soluções de  $B(t - 1)$  e  $P(t)$  em  $B(t)$ 
        armazena a melhor solução  $\mathbf{b}$  de  $B(t)$ 
    fim
fim

```

Figura 2.5: Pseudo–código do algoritmo evolutivo com inspiração quântica usando representação binária.

mos de tempo de convergência e qualidade das soluções encontradas. Uma descrição mais detalhada deste algoritmo pode ser encontrada em (Han00, Han02).

## 2.6 Algoritmos Culturais

Os algoritmos culturais são um tipo de algoritmo evolutivo que complementam a metáfora adotada pelos algoritmos genéticos tradicionais. Ao invés de usar apenas analogias dos conceitos de genética e de seleção natural, os algoritmos culturais são baseados também em teorias sociais e arqueológicas que modelam a evolução cultural. Estas teorias indicam que a evolução cultural pode ser vista como um processo de herança que ocorre em dois níveis: o nível micro-evolutivo e nível macro-evolutivo (Reynolds94).

No nível micro-evolutivo, os indivíduos são descritos em termos de características comportamentais (que podem ser socialmente aceitáveis ou não). Estas características são passadas de geração para geração usando uma série de operadores. No nível macro-evolutivo, os indivíduos são capazes de gerar "*mappas*" ou descrições gerais de suas experiências. Os *mappas* individuais podem ser agrupados através do uso de um conjunto de operadores (genéricos ou específicos para o problema). Os dois níveis evolutivos compartilham uma ligação através da qual podem se comunicar.

O objetivo do uso deste modelo em dois níveis é aumentar a velocidade de convergência do processo de otimização e permitir uma melhor resposta do algoritmo para uma gama maior de problemas.

### 2.6.1

#### Componentes de um Algoritmo Cultural

Os algoritmos culturais operam em dois espaços: o espaço de população, que modela as características comportamentais no nível micro-evolutivo, e o espaço de crenças, que modela os *mappas* de experiência no nível macro-evolutivo (Reynolds04). Inicialmente, os indivíduos no espaço de população são avaliados através do uso de uma função de desempenho *obj()*. Uma função de aceitação *acpt()* é então usada para determinar quais indivíduos do espaço de população irão influenciar o espaço de crenças. As características comportamentais destes indivíduos serão usadas para modificar as crenças existentes dentro do espaço de crenças, através de uma função de atualização *upd<sub>t</sub>()*. Em seguida, as crenças que formam o espaço de crenças são usadas para influenciar a evolução da população através de uma função de influência *infl<sub>t</sub>()*. Finalmente, alguns indivíduos da população são selecionados para a população da geração seguinte através de uma função de seleção *sel()*. As figuras 2.6 e 2.7 mostram, respectivamente, o algoritmo cultural em forma de pseudo–código e de um diagrama da estrutura geral de relacionamento entre os espaços de população e de crenças.

```

iniciar
     $t = 0;$ 
    inicializa espaço de crenças  $B^t$  e população  $P^t$ ;
repetir
     $obj(P^t);$ 
     $updt(B^t, acpt(P^t));$ 
     $infl(B^t, P^t);$ 
     $t = t + 1;$ 
     $P^t \Leftarrow sel(P^{t-1});$ 
até que condição de parada alcançada;
fim

```

Figura 2.6: Pseudo–código do algoritmo cultural.

O espaço de crenças do algoritmo cultural é a região onde o conhecimento é armazenado. As cinco categorias básicas de conhecimento cultural, de acordo com (Reynolds04), são:

- *Conhecimento Normativo* – um conjunto de faixas de valores promissores dentro do espaço de busca que fornece padrões para os comportamentos individuais serem ajustados. O conhecimento normativo guia os indivíduos na direção das regiões promissoras, caso estes indivíduos já não estejam dentro destas regiões;
- *Conhecimento Situacional* – fornece um conjunto de “casos exemplares” que são úteis para a interpretação de experiências individuais específicas. Este

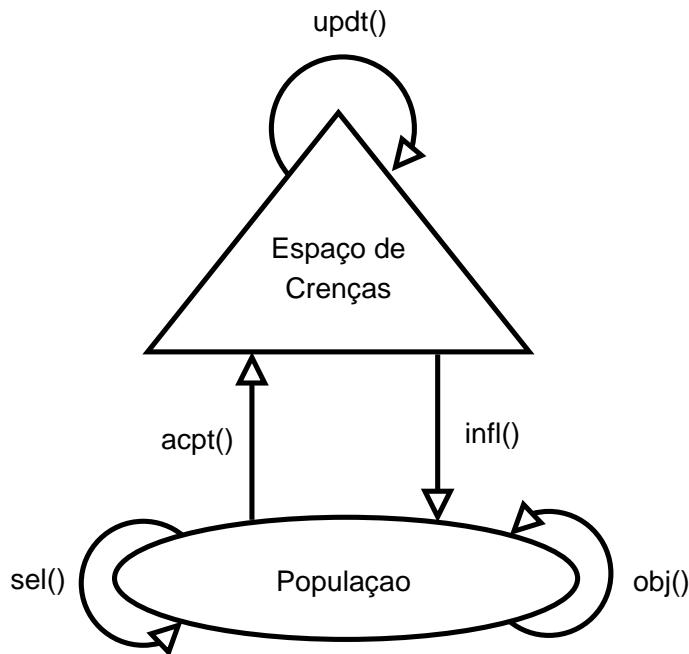


Figura 2.7: Diagrama da estrutura geral do algoritmo cultural.

tipo de conhecimento leva os indivíduos na direção dos melhores casos. Nos algoritmos genéticos convencionais (e também em várias implementações dos algoritmos culturais) este conhecimento é representado pelo melhor indivíduo da população;

- *Conhecimento Topográfico* – este conhecimento foi originalmente proposto para melhorar o conhecimento do processo de otimização sobre as estruturas locais do espaço de busca. O espaço de buscas é dividido em células e cada célula armazena o melhor indivíduo que está dentro dos seus limites. O conhecimento topográfico leva os indivíduos na direção dos melhores indivíduos locais;
- *Conhecimento do Domínio* – este tipo de conhecimento diz respeito ao domínio do problema que está sendo otimizado. Em geral, consiste em hibridizar, com o algoritmo cultural, alguma heurística particular ao problema que se quer otimizar;
- *Conhecimento Histórico* – o conhecimento histórico é usado, principalmente, em sistemas de aprendizado *online*. Ele é usado para armazenar o melhor indivíduo encontrado pelo algoritmo cultural antes de uma mudança na topologia da função de avaliação.

Os algoritmos culturais foram usados com sucesso em diversas áreas (Rychtyckyj03), como análise de fraudes em seguros de automóveis, especificação de estratégias para automóveis em um ambiente multi-agente e mineração de dados.

## 2.7

### Neuro-Evolução

A neuro-evolução é um modelo híbrido que explora a potencialidade de duas diferentes áreas inspiradas em processos biológicos: Redes Neurais Artificiais (RNA) e Algoritmos Genéticos (AG). A idéia básica da neuro-evolução é buscar, automaticamente, a melhor configuração para uma rede neural usando algoritmos genéticos. Em outras palavras, a neuro-evolução combina a capacidade de generalização e aproximação de funções das redes neurais artificiais com um método eficiente de busca paralela. O objetivo dos algoritmos genéticos é melhorar os algoritmos de aprendizado, automatizando, total ou parcialmente, o processo de configuração da rede neural, bem como o processo de treinamento e atualização dos pesos da mesma.

Para a neuro-evolução funcionar, não é necessário que o sistema satisfaça nenhuma restrição em particular: o mesmo pode ser contínuo e parcialmente observável. No que concerne os métodos de aprendizado por neuro-evolução, a única exigência é que os mesmos possam, de alguma forma, ter as suas soluções candidatas avaliadas relativamente umas às outras. Se o problema é suficientemente regular, de modo que o mesmo possa ser resolvido, e a representação do fenótipo<sup>1</sup> é suficientemente poderosa, então estes métodos de aprendizado serão, muito provavelmente, capazes de encontrar uma solução.

As abordagens possíveis para os sistemas neuroevolutivos diferem uma da outra, basicamente, pelo modo como as mesmas codificam os pesos e a topologia das redes neurais nos cromossomos. Os cromossomos podem codificar qualquer informação relevante para a parametrização da rede neural, incluindo os pesos sinápticos, o número de camadas escondidas, a topologia da rede, a taxa de aprendizado, etc. A escolha do esquema de codificação tem um papel significativo na formação do espaço de buscas, no comportamento do algoritmo de busca e em como os genótipos devem ser transformados nos fenótipos (representação direta ou indireta).

No modelo descrito em (Gomez97) somente os pesos sinápticos são codificados, mantendo fixa a topologia e o número de neurônios da rede neural. Já os modelos SANE (*Symbolic, Adaptive Neuroevolution*) (Moriarty97), NEAT (*Neuroevolution of Augmenting Topologies*) (Stanley02) e ESP (*Enforced Sub-Populations*) (Gomez03) otimizam, além dos pesos sinápticos, a topologia e o número de neurônios da rede neural. Além disso, neste último trabalho, diversas aplicações de neuro-evolução em problemas de controle são apresentadas, com resultados promissores tanto com relação à capacidade de evoluir corretamente como com relação

<sup>1</sup>Em biologia, o genótipo de um organismo vivo é constituído pelos genes e pela configuração dos mesmos, enquanto que o fenótipo é a expressão física do genótipo como, por exemplo, a cor dos olhos de uma pessoa.

ao desempenho do processo de otimização.

As principais desvantagens destes modelos estão relacionadas ao fato de que, ao usar modelos evolutivos baseados somente em uma população de indivíduos, o aprendizado do sistema pode ser muito lento, devido ao fato de que o espaço de buscas é mapeado pontualmente (cada indivíduo representa um único ponto no espaço de buscas). Isto pode retardar o processo de aprendizado e, consequentemente, tornar o aprendizado inviável em alguns sistemas que devem operar em tempo real ou que necessitem de atualizações *online*. Além disso, por se tratarem de algoritmos de busca global, os mesmos são mais ineficientes do que os algoritmos de busca baseados em gradiente. Finalmente, o número de pesos de uma rede neural cresce quadraticamente com o número de entradas. Isto implica no uso de indivíduos mais longos e, consequentemente, uma população maior para que a otimização seja bem-sucedida. O uso mais promissor de algoritmos genéticos para o treinamento de redes neurais foi feito usando-se os mesmos para encontrar os pesos iniciais da rede neural e posteriormente, através de um outro algoritmo, realizar o aprendizado propriamente dito (Skinner95). Este processo se mostrou eficiente pois o uso de algoritmos genéticos supre a deficiência que os algoritmos baseados em gradiente apresentam para fazer uma busca global eficiente (evitando mínimos locais), permitindo que os algoritmos de aprendizado das redes neurais façam a busca local de forma rápida (Skinner95).

O próximo capítulo detalha o funcionamento do algoritmo evolutivo com inspiração quântica e representação real, o seu uso para treinamento de redes neurais e também mostra como o modelo hipotético da partícula na caixa pode ser usado para simular este algoritmo.

### 3

## Algoritmos Evolutivos com Inspiração Quântica e Representação Real – AEIQ–R

Conforme mencionado nos capítulos 1 e 2, os problemas de otimização numérica são um importante assunto de pesquisa e oferecem desafios nas mais diversas áreas como, por exemplo, otimização de plantas industriais, controle, síntese de filtros digitais, treinamento de redes neurais e diversas outras aplicações. O uso de algoritmos evolutivos com inspiração quântica nesta classe de problemas de otimização pode, potencialmente, oferecer inúmeras vantagens. Mas, como acontece com outros tipos de algoritmos evolutivos (sendo os algoritmos genéticos o exemplo mais comum), a representação binária não é, necessariamente, a mais adequada para este tipo de problema, por apresentar algumas particularidades que restringem a capacidade de otimização do algoritmo. Com o objetivo de contornar este problema, este trabalho apresenta um modelo com inspiração quântica que usa uma representação baseada em números reais, sendo, portanto, mais adequada para os problemas de otimização numérica. Além de oferecer uma representação mais adequada, o modelo apresentado neste trabalho, denominado Algoritmo Evolutivo com Inspiração Quântica e Representação Real – AEIQ–R, tem as seguintes vantagens, em relação aos algoritmos genéticos tradicionais e ao AEIQ–B:

- Menor tempo para convergência;
- O conhecimento sobre o problema que está sendo otimizado é armazenado diretamente nos cromossomos quânticos;
- Populações com poucos indivíduos; capacidade de convergir de forma eficiente, mesmo com populações com poucos indivíduos.

### 3.1

#### O Modelo de Algoritmo Evolutivo com Inspiração Quântica e Representação Real

O interesse principal deste trabalho é desenvolver um modelo completo, que permita utilizar uma representação numérica em um algoritmo de otimização com inspiração na física quântica. Ao contrário do AEIQ–B, descrito na seção 2.5, deseja-se que este algoritmo permita representar uma superposição de estados

contínuos. Por este motivo, a abordagem usando funções de onda é usada como inspiração para o desenvolvimento do algoritmo.

É importante frisar que, no caso de algoritmos com inspiração quântica, a superposição dos estados é apenas simulada, ou seja, o algoritmo é executado em um computador convencional (daí a denominação "inspiração quântica").

As próximas seções deste capítulo apresentam o algoritmo completo do modelo proposto neste trabalho, bem como uma descrição detalhada de cada um dos passos do algoritmo, da representação dos indivíduos com inspiração quântica, dos operadores evolutivos e outros detalhes importantes referentes ao processo de otimização.

### 3.1.1

#### O Algoritmo Evolutivo com Inspiração Quântica com Representação Real

A figura 3.1 mostra a listagem completa do algoritmo evolutivo com inspiração quântica com representação real (AEIQ–R).

```

iniciar
1.    $t \leftarrow 1$ 
2.   Gerar população quântica  $Q(t)$  com  $N$  indivíduos com  $G$  genes
3.   enquanto ( $t \leq T$ )
4.        $E(t) \leftarrow$  gerar indivíduos clássicos usando indivíduos quânticos
5.       se ( $t=1$ ) então
6.            $C(t) \leftarrow E(t)$ 
7.       senão
8.            $E(t) \leftarrow$  recombinação entre  $E(t)$  e  $C(t)$ 
9.           avaliar  $E(t)$ 
10.           $C(t) \leftarrow K$  melhores indivíduos de  $[E(t) + C(t)]$ 
11.      fim se
12.       $Q(t+1) \leftarrow$  Atualiza  $Q(t)$  usando os  $N$  melhores indivíduos de  $C(t)$ 
13.       $t \leftarrow t + 1$ 
14.  fim enquanto
fim

```

Figura 3.1: Listagem completa do algoritmo evolutivo com inspiração quântica usando representação real.

Este algoritmo será explicado detalhadamente nas subseções a seguir.

### 3.1.2

#### A População Quântica

Da mesma forma que o algoritmo genético com inspiração quântica usando representação binária (AEIQ–B) necessita de uma população de indivíduos que representem a superposição dos possíveis estados que o indivíduo clássico pode assumir ao ser observado, o AEIQ–R também necessita de uma população com a

mesma funcionalidade. No entanto, esta representação deve respeitar a condição de que o conjunto de estados observáveis deve ser contínuo, e não discreto como no caso do AEIQ–B.

Para se conseguir esse objetivo, a população quântica  $Q(t)$ , em um instante  $t$  qualquer do processo evolutivo, é formada por um conjunto de  $N$  indivíduos quânticos  $q_i$  ( $i = 1, 2, 3, \dots, N$ ). Cada indivíduo quântico  $q_i$  desta população é formado por  $G$  genes  $g_{ij}$  ( $j = 1, 2, 3, \dots, G$ ) que, por sua vez, são formados por funções que representam uma densidade de probabilidade. Os indivíduos quânticos podem ser representados pela equação 3-1.

$$q_i \rightarrow [g_{i1} = p_{i1}(x), g_{i2} = p_{i2}(x), \dots, g_{iG} = p_{iG}(x)] \quad (3-1)$$

Onde  $i = 1, 2, 3, \dots, N$ ,  $j = 1, 2, 3, \dots, G$  e as funções  $p_{ij}$  representam as funções densidade de probabilidade. Esta função densidade de probabilidade é usada pelo AEIQ–R para gerar os valores para os genes dos indivíduos clássicos. Em outras palavras, a função  $p_{ij}(x)$  representa a densidade de probabilidade de se observar um determinado valor para o *gene quântico* quando a superposição do mesmo for colapsada. De fato, a função  $p_{ij}(x)$  pode ser definida como mostra a equação 3-2.

$$p_{ij}(x) = \psi_{ij}^*(x)\psi_{ij}(x) \quad (3-2)$$

Onde  $\psi_{ij}(x)$  representa a função de onda associada ao gene quântico  $j$  do indivíduo  $i$  da população quântica e  $\psi_{ij}^*(x)$  representa o conjugado complexo desta mesma função de onda. A função densidade de probabilidade deve respeitar a propriedade de normalização mostrada na equação 3-3.

$$\int \psi_{ij}(x)^*\psi_{ij}(x)dx = \int p_{ij}(x)dx = 1 \quad (3-3)$$

É importante ressaltar que a função densidade de probabilidade deve ser integrável na região do domínio dentro do qual as variáveis que se quer otimizar podem assumir valores. Isto permite calcular a distribuição cumulativa de probabilidade e assim usar a distribuição de probabilidade para se gerar valores para os genes clássicos.

Uma das funções mais simples que se pode usar como função densidade de probabilidade é o pulso quadrado. Esta função pode ser definida pela equação 3-4.

$$p_{ij}(x) = \begin{cases} \frac{1}{U_{ij}-L_{ij}} & \text{se } L_{ij} \leq x \leq U_{ij} \\ 0 & \text{caso contrário} \end{cases} \quad (3-4)$$

Onde  $L_{ij}$  é o limite inferior e  $U_{ij}$  o limite superior do intervalo no qual o gene  $j$  do  $i$ -ésimo indivíduo quântico pode assumir valores (colapsar) quando observado. Esta equação respeita a propriedade de normalização mencionada na equação 3-3. Além disso, corresponde a uma distribuição de probabilidade que pode

ser facilmente utilizada em algoritmos computacionais, já que a mesma corresponde a uma distribuição uniforme no intervalo  $[L_{ij}, U_{ij}]$ .

Um exemplo de um gene quântico formado por um pulso quadrado é mostrado na figura 3.2. Neste exemplo, os limites  $L_{ij}$  e  $U_{ij}$  da função  $p_{ij}(x)$  estão definidos como  $-1$  e  $1$  respectivamente.

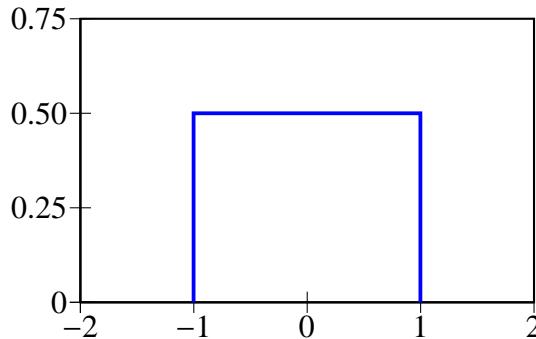


Figura 3.2: Exemplo de um gene quântico do AEIQ–R .

O passo 2 do algoritmo da figura 3.1 consiste, portanto, em gerar um conjunto de  $N$  indivíduos quânticos formados por genes que, por sua vez, são formados por funções  $p_{ij}(x)$ . Para o caso em que  $p_{ij}(x)$  é um pulso quadrado, pode-se representar o gene quântico armazenando-se os valores dos limites inferior e superior para cada gene, ou armazenando a posição do ponto central do pulso quadrado e a largura do mesmo. Por exemplo, considerando um indivíduo quântico  $q_i$  formado por dois pulsos quadrados  $g_{i1}(x)$  e  $g_{i2}(x)$ , e supondo que estes dois pulsos quadrados têm uma largura igual a 2 e estão posicionados de tal modo que o seu centro está localizado nas posições  $-0.5$  e  $0.5$  respectivamente, o cromossomo quântico pode ser representado, caso se esteja usando largura e centro como valores para os genes, por  $q_i = (\mu_{i1} = -0.5, \mu_{i2} = 0.5, \sigma_{i1} = 2, \sigma_{i2} = 2)$ , onde  $\mu_{i1}$  e  $\mu_{i2}$  indicam o centro e  $\sigma_{i1}$  e  $\sigma_{i2}$  representam a largura dos dois pulsos quadrados respectivamente. Obviamente a altura desses pulsos deve ser calculada de modo que a propriedade de normalização da função densidade de probabilidade formada pela soma desses pulsos quadrados seja respeitada. Assim, a altura dos pulsos quadrados deve ser tal que a área total dos mesmos seja igual a 1, e pode ser calculada usando-se a equação 3-4. No exemplo dado, cada pulso terá, portanto, uma altura igual a 0.5.

Ao usar o pulso quadrado, pode-se usar, pelo menos, duas estratégias diferentes de inicialização dos mesmos: na primeira, cria-se pulsos quadrados com uma largura igual a  $(U_{ij} - L_{ij})/N$  (onde  $N$  é o número de indivíduos quânticos) e com seus centros distribuídos uniformemente ao longo de todo o domínio das variáveis; na segunda, cria-se todos os pulsos com o limite inferior e superior igual ao limite inferior e superior do domínio, respectivamente, sob o qual se deseja otimizar a função objetivo. Como exemplo para a primeira estratégia, pode-se considerar um

problema com duas variáveis  $x$  e  $y$ , ambas no intervalo  $[-100, 100]$ . Considerando também que a função densidade de probabilidade é representada por um pulso quadrado e que a população quântica inicial  $Q(0)$  tem 5 indivíduos quânticos, pode-se representar estes indivíduos quânticos que formam a população inicial da seguinte maneira:

$$Q(0) = \left[ \begin{array}{l} q_1 = [(\mu = -80, \sigma = 40); (\mu = -80, \sigma = 40)] \\ q_2 = [(\mu = -40, \sigma = 40); (\mu = -40, \sigma = 40)] \\ q_3 = [(\mu = 0, \sigma = 40); (\mu = 0, \sigma = 40)] \\ q_4 = [(\mu = 40, \sigma = 40); (\mu = 40, \sigma = 40)] \\ q_5 = [(\mu = 80, \sigma = 40); (\mu = 80, \sigma = 40)] \end{array} \right]$$

Neste caso, a posição central de cada pulso foi determinada, de modo que os mesmos ficassem uniformemente distribuídos pelo domínio de cada variável. Além disso, a altura de cada pulso deve ser tal que a área total de cada um deles seja igual a 1. Assim sendo, a altura de cada pulso neste exemplo deve ser igual a  $1/40 = 0.025$ .

Já no caso da segunda estratégia, os indivíduos quânticos seriam representados como:

$$Q(0) = \left[ \begin{array}{l} q_1 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \\ q_2 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \\ q_3 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \\ q_4 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \\ q_5 = [(\mu = 0, \sigma = 200); (\mu = 0, \sigma = 200)] \end{array} \right]$$

Neste caso, a altura de cada pulso será igual a  $1/200 = 0.005$ .

### 3.1.3

#### Observação dos Indivíduos Quânticos

Após a inicialização da população quântica no passo 2, mostrado anteriormente, o algoritmo entra no laço principal do processo evolutivo. Este laço será executado por um determinado número  $T$  de gerações e é formado por várias tarefas.

O passo 4 é um dos mais importantes do algoritmo. É neste momento que se executa o processo de observação do estado quântico para se gerar indivíduos clássicos, ou seja, indivíduos cujos genes são números reais dentro do intervalo válido do domínio. Para se executar esta observação, usam-se as funções densidade de probabilidade  $p_{ij}(x)$  e um gerador uniforme de números aleatórios no intervalo  $[0, 1]$ . Para cada gene é realizada uma observação através do seguinte procedimento:

1. Gera-se um número aleatório  $r$  no intervalo  $[0, 1]$ ;

2. Identifica-se o ponto  $x$  tal que, dado que  $P_{ij}(x) = \int p_{ij}(x)dx$ ,  $x = P_{ij}^{-1}(r)$ ;
3.  $x$  é o valor observado para o gene  $j$  do indivíduo clássico  $i$ .

Este processo, a princípio, implica que o número de indivíduos clássicos gerados é igual ao número de indivíduos quânticos. No entanto, isto não é, necessariamente, verdade. O número de indivíduos clássicos pode ser igual ou maior ao número de indivíduos quânticos, bastando que, ao se gerar os indivíduos clássicos, não se dê preferência a certos indivíduos quânticos no processo de observação, afinal, os indivíduos quânticos não são, a princípio, avaliados como os indivíduos clássicos e, portanto, não podem ser considerados melhores ou piores uns que os outros. Em outras palavras, deve-se garantir que nenhum indivíduo quântico seja preterido ou privilegiado em relação aos outros no que diz respeito ao número de vezes que o mesmo será usado para o processo de observação (geração dos indivíduos clássicos). Uma opção, usada neste trabalho, consiste em garantir que o número de indivíduos clássicos seja um múltiplo inteiro do número de indivíduos quânticos. Assim, dado que o número de indivíduos quânticos é igual a  $N$  e supondo-se que o número de indivíduos clássicos seja igual a  $N_c$ , define-se  $N_c = kN$ , onde  $k \in \mathbb{N} - \{0\}$  indica quantos indivíduos clássicos serão gerados por cada indivíduo quântico.

É importante ressaltar que, o número de indivíduos quânticos não deve ser maior do que o número de indivíduos clássicos ( $N \leq N_c$ ). Esta restrição se deve ao fato de que os indivíduos clássicos serão usados para atualizar os indivíduos quânticos posteriormente, o que será explicado em detalhes adiante, na seção 3.1.4.

Como exemplo do processo de geração de indivíduos clássicos, pode-se considerar uma população quântica formada por dois indivíduos, cada um composto por 2 genes que usam pulsos quadrados como função densidade de probabilidade. A configuração destes indivíduos é dada na tabela 3.1 e a representação gráfica dos mesmos é mostrada na figura 3.3.

Indivíduo	Genes
$q_1$	$g_{11} = (-\mu_{11} = 5, \sigma_{11} = 20)$ , $g_{12} = (\mu_{12} = 0, \sigma_{12} = 20)$
$q_2$	$g_{21} = (\mu_{21} = 5, \sigma_{21} = 20)$ , $g_{22} = (\mu_{22} = 5, \sigma_{22} = 20)$

Tabela 3.1: Exemplo de indivíduos que formam uma população quântica  $Q(t)$  em uma geração  $t$  qualquer.

A função  $P_{ij}(x)$  (função cumulativa de probabilidade), associada a cada um dos genes quânticos mostrados anteriormente, pode ser facilmente representada por equações de retas, já que as funções  $p_{ij}(x)$  são constantes dentro dos intervalos onde o valor das mesmas é diferente de 0. A figura 3.4 mostra as funções  $P_{ij}(x)$  associadas às funções densidade de probabilidade  $p_{ij}(x)$  mostradas, anteriormente, na figura 3.3.

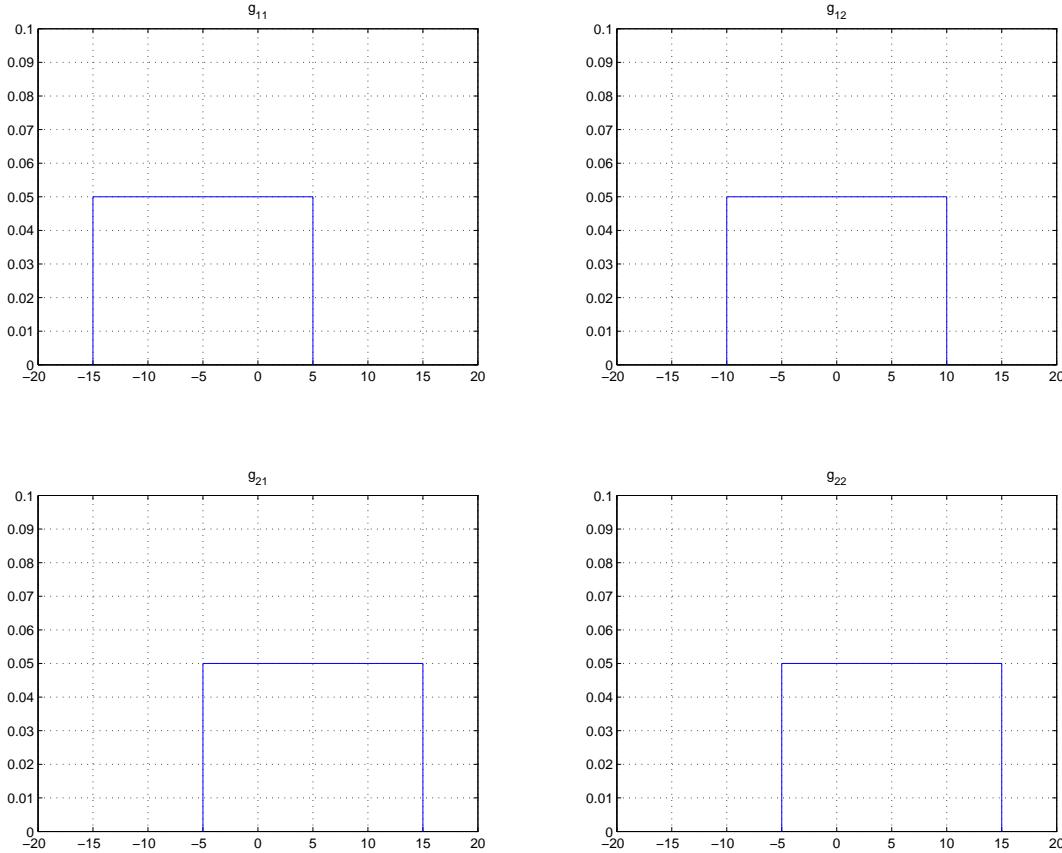


Figura 3.3: Genes de uma população quântica usando pulsos quadrados como função densidade de probabilidade.

De posse das funções  $P_{ij}(x)$  é possível efetuar a geração dos indivíduos clássicos propriamente dita (passo 4 do algoritmo). Esta observação é feita através de um processo aleatório, descrito anteriormente. Este processo aleatório é determinado pela função de onda de cada gene do indivíduo quântico. Assim, para cada gene  $j$  de cada indivíduo quântico  $i$ , deve-se gerar um número aleatório  $r_{ij}$  no intervalo  $[0, 1]$ . Com este número pode-se, então, gerar o valor observado para o gene  $j$  de cada indivíduo clássico. De fato, o cálculo dos indivíduos clássicos é extremamente simples quando se usa um pulso quadrado para representar os genes que formam o indivíduo quântico, já que a função cumulativa de probabilidade pode ser representada pela equação de uma reta, quando se usa pulsos quadrados. Dado o número  $r_{ij}$  gerado aleatoriamente, e usando-se a equação da reta  $y(x) = ax + b$ , pode-se calcular o valor do gene clássico usando-se a equação 3-5.

$$x_{mj} = r_{mj} * \sigma_{ij} + \left( \mu_{ij} - \frac{\sigma_{ij}}{2} \right) \quad (3-5)$$

Onde  $x_{mj}$  é o  $j$ -ésimo gene do  $m$ -ésimo indivíduo clássico que se está gerando,  $r_{mj}$  é o número aleatório sorteado para o gene do indivíduo que está sendo observado, e  $\mu_{ij}$  e  $\sigma_{ij}$  são a posição do centro e a largura do pulso quadrado que está sendo usado para observar o gene clássico.

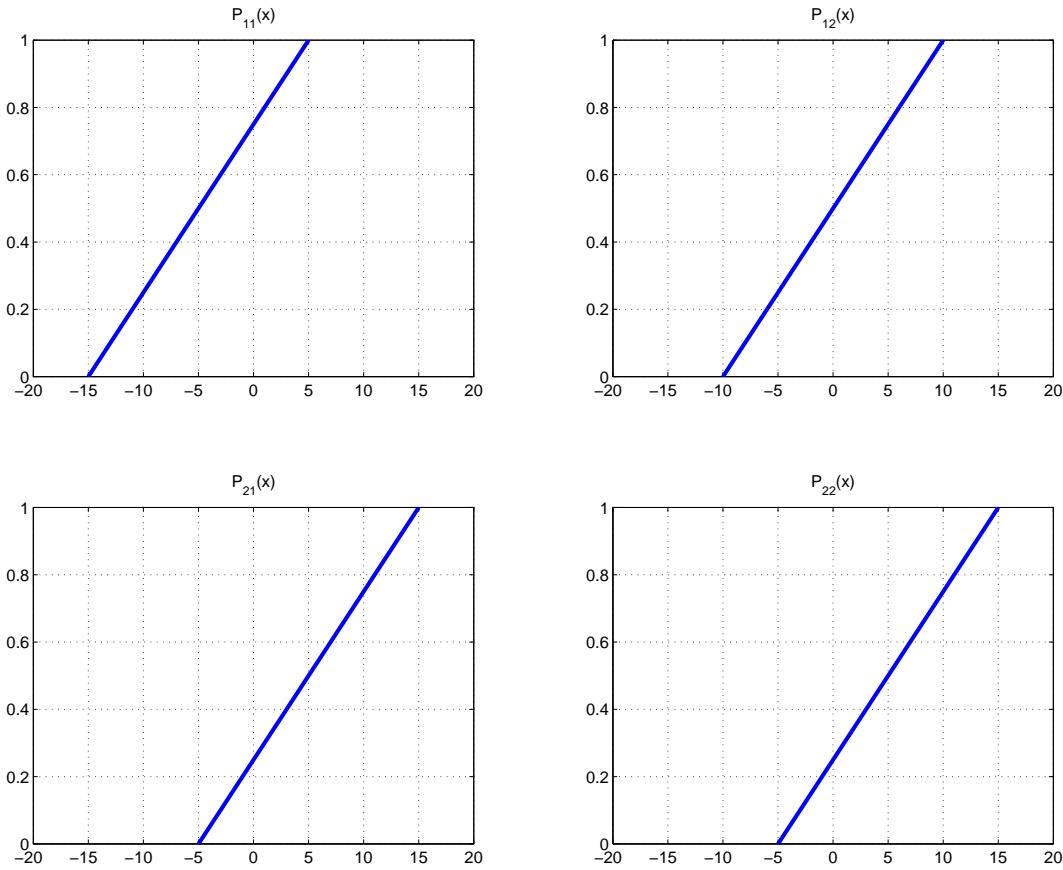


Figura 3.4: Funções cumulativas de probabilidade associadas aos genes de uma população quântica.

Para tornar o processo mais claro, considere o exemplo de uma população quântica como a mostrada na figura 3.3. Considere também que, para este exemplo, deseja-se gerar uma população clássica com 4 indivíduos. Deste modo, o número de indivíduos  $N_c = kN \rightarrow k = N_c/N$  é tal que  $k = 2$  e, portanto, cada indivíduo quântico será usado para gerar dois indivíduos clássicos. Supondo-se que o gerador de números aleatórios forneceu 2 números diferentes para gerar os dois genes do primeiro indivíduo clássico – {0.1, 0.8} – e, usando-se a equação 3-5 ( $x_{11} = 0.1 * 20 + (-5 + 10)$  e  $x_{12} = 0.8 * 20 + (0 + 10)$ ), o indivíduo clássico observado é, portanto, {-13, 6}.

Neste ponto existe a possibilidade de se realizar uma operação de recombinação entre os indivíduos da população nova (geração atual) e da população antiga (geração anterior), equivalente ao passo 8 do algoritmo da figura 3.1. O uso desta função de recombinação é interessante para permitir o aproveitamento do material genético já presente na população clássica, melhorando a capacidade do algoritmo explorar regiões próximas dentro do espaço de busca. Neste trabalho, é usada uma operação de recombinação similar a que é usada no algoritmo de evolução diferencial, descrito em (Storn95).

Em seguida, os indivíduos da população nova são avaliados (passo 9 do

algoritmo da figura 3.1) pela função de avaliação que se deseja otimizar. Apesar de ser possível, não é, provavelmente, conveniente usar operações de mutação nos indivíduos novos ou antigos, já que os indivíduos quânticos, por si só, já introduzem um efeito aleatório (*exploration*) nas populações sendo evoluídas.

Com a nova população clássica gerada, deve-se tomar uma decisão sobre como substituir os indivíduos de uma eventual população pré-existente (passo 10 do algoritmo da figura 3.1). De fato, a estratégia de substituição dos indivíduos da população da geração anterior pelos indivíduos criados na nova geração deve ser usada a todo momento, com exceção da primeira geração, onde não existe uma população anterior. Existem várias abordagens possíveis para abordar esta etapa do algoritmo. Estas abordagens podem ser:

1. Substituir todos os indivíduos da população antiga pelos indivíduos da população nova (equivalente a ausência de elitismo e *steady-state* dos algoritmos genéticos convencionais);
2. Substituir todos os indivíduos da população antiga pelos indivíduos da população nova, mas preservando o melhor indivíduo da população nova (equivalente ao elitismo dos algoritmos genéticos tradicionais);
3. Substituir os  $n$  piores indivíduos da população antiga pelos  $n$  melhores indivíduos da população nova (equivalente ao *steady-state* dos algoritmos genéticos tradicionais);
4. Substituir os  $n$  piores indivíduos da população antiga por  $n$  indivíduos quaisquer da população nova.

Cada uma dessas abordagens apresenta vantagens e desvantagens. A opção 1 diminui o tempo de processamento pois elimina a necessidade de ordenação dos indivíduos da população antiga e da população nova. No entanto, esta abordagem tende a introduzir muito ruído, já que os indivíduos da população antiga são completamente substituídos e a avaliação do melhor indivíduo pode piorar significativamente, dependendo do problema que se deseja otimizar. A opção 2 resolve este problema preservando o indivíduo mais apto da população clássica na população nova. A opção 3, apesar de exigir a ordenação dos elementos da população antiga e da população nova, apresenta a alternativa mais conservadora de busca, sem provocar grandes alterações na população que está sendo evoluída. Finalmente, a opção 4 combina o conservadorismo da opção 3, mantendo os indivíduos mais bem avaliados na população em evolução, mas diminuindo o tempo de processamento por não necessitar avaliar todos os  $N_c$  indivíduos da população nova.

### 3.1.4

#### Atualização da População Quântica

Finalmente, após a geração dos indivíduos clássicos, é necessário atualizar os indivíduos da população quântica (passo 12 do algoritmo da figura 3.1). Este procedimento depende, até certo ponto, do tipo de função densidade de probabilidade que se definiu para usar nos genes quânticos. Em geral, este processo deve:

- Reduzir o espaço de busca da função que se quer otimizar. No AEIQ-R isto é feito reduzindo-se o tamanho da região onde a função densidade de probabilidade (genes quânticos) tem probabilidade diferente de 0;
- Mapear as regiões mais promissoras do espaço de busca. Isto deve ser feito aumentando-se a probabilidade de se observar um determinado conjunto de valores para o gene clássico nas proximidades dos indivíduos mais bem-sucedidos da população clássica.

Fica claro que, na primeira geração, por não se ter encontrado ainda as regiões mais promissoras do espaço de busca, o tamanho do domínio para a função que se quer otimizar deve ser o maior possível para que todas as regiões do espaço de busca sejam mapeadas. Em outras palavras, supondo-se que se deseja otimizar uma função  $f(x)$  no intervalo  $x \in [-100, 100]$ , os genes de todos os indivíduos quânticos devem, preferencialmente, estar definidos de tal forma que todos os valores possíveis dentro do intervalo  $[-100, 100]$  tenham chance de serem observados. Para o caso em que a função densidade de probabilidade é um pulso quadrado, estes valores devem ter, idealmente, a mesma chance de serem observados. Obviamente que, no caso de se ter informações *a priori* sobre regiões mais promissoras, pode-se redefinir as funções densidade de probabilidade de tal modo que os indivíduos quânticos ofereçam maiores chances de que genes na região mais promissora do espaço de busca sejam observados com mais freqüência do que em outras regiões.

Supondo-se, então, que uma função densidade de probabilidade com a forma de pulso quadrado foi usada para os genes quânticos e, para garantir que as duas condições citadas acima sejam satisfeitas, pode-se considerar o seguinte processo de atualização dos genes quânticos:

1. Modificar a largura dos pulsos de modo que o espaço de buscas seja reduzido;
2. Modificar a posição dos pulsos de modo que o ponto central dos mesmos coincida com o valor dos genes de um conjunto de indivíduos da população clássica.

Mais uma vez é possível adotar diversas estratégias para realizar cada uma das duas tarefas descritas anteriormente. Para o passo 1, pode-se usar um decaimento

exponencial da largura dos pulsos quadrados, um decaimento linear ou, ainda, um processo similar ao usado em algoritmos de estratégia evolutiva chamado “*regra do 1/5*” (Michalewicz94). A regra do 1/5 faz com que a modificação de largura dos pulsos seja executada de forma homogênea para todos os genes quânticos e para todos os indivíduos quânticos. A heurística usada para determinar se a largura do gene será aumentada ou diminuída é a seguinte (Michalewicz94): se menos de 20% da população clássica criada na geração atual tiver uma avaliação melhor do que na geração anterior, a largura do gene é reduzida; se esta taxa for maior do que 20%, a largura do gene é aumentada; caso a taxa seja exatamente igual a 20%, nenhuma alteração é feita. Matematicamente, isto pode ser representado pela equação 3-6.

$$\sigma_{ij} = \begin{cases} \sigma_{ij} \cdot \delta & \varphi < 1/5 \\ \sigma_{ij}/\delta & \varphi > 1/5 \\ \sigma_{ij} & \varphi = 1/5 \end{cases} \quad (3-6)$$

Onde  $\sigma_{ij}$  é a largura do  $j$ -ésimo gene do  $i$ -ésimo indivíduo quântico em  $Q(t)$ ,  $\delta$  é um valor arbitrário, em geral no intervalo  $]0, 1[$  e  $\varphi$  é a taxa que indica quantos indivíduos da nova população clássica foram melhores do que os indivíduos da população clássica na geração anterior.

Esta operação de redimensionamento da largura dos genes da população quântica pode ser feita em cada geração ou pode usar um intervalo de tempo maior (por exemplo, a cada 5 gerações). Intervalos menores para o redimensionamento aceleram o processo de convergência, mas podem levar o algoritmo a ficar preso em mínimos locais. Por outro lado, intervalos grandes podem evitar que o aprisionamento aconteça, mas podem retardar o processo de otimização.

O argumento para o uso desta regra se baseia na seguinte heurística: se menos de 20% dos indivíduos da população clássica gerada tiverem melhorado de uma geração para outra, é provável que o algoritmo esteja fazendo a busca em uma região muito grande e, neste caso, pode ser interessante reduzir o espaço de buscas; se mais de 20% dos indivíduos tiverem melhorado de uma geração para outra, é provável que o algoritmo esteja fazendo a busca em uma região muito pequena (e portanto achando, facilmente, indivíduos com avaliações melhores) e o mesmo deve tentar ampliar a região de busca; a taxa de 20% é considerada ideal e, portanto, nenhuma alteração é feita caso essa taxa seja medida.

Já no passo 2 da atualização dos genes quânticos, o processo pode ser definido em termos de quais indivíduos da população clássica serão usados para atualizar os pulsos da população quântica. Pode-se escolher os indivíduos mais aptos, os indivíduos menos aptos, um conjunto aleatório de indivíduos ou usar um processo de roleta para selecionar quais os indivíduos que farão parte do *pool* de indivíduos clássicos que serão usados para atualizar os indivíduos quânticos. Após escolher quais indivíduos serão usados (o número de indivíduos clássicos escolhidos deve

ser igual ao número de indivíduos quânticos e, portanto, o número de indivíduos clássicos não pode ser menor do que o número de indivíduos quânticos), o centro dos pulsos de cada indivíduo quântico deve ser modificado em relação ao valor de cada gene quântico. Em geral, pode-se usar um cálculo que desloque o centro dos pulsos na direção do ponto indicado pelo gene do indivíduo clássico. Por exemplo, supondo-se que o centro do gene quântico na geração  $t$  seja dado por  $\mu_{ij}(t)$  e o valor do gene clássico seja dado por  $x_{ij}$ , então, pode-se calcular a nova posição do gene quântico na geração  $t + 1$ , através da equação  $\mu_{ij}(t + 1) = \mu_{ij}(t) + \lambda(x_{ij} - \mu_{ij})$ , onde  $\lambda$  indica o percentual que se quer deslocar o centro do gene quântico na direção do gene clássico.

Todos esses passos do algoritmo devem ser repetidos por um número  $T$  de gerações, de acordo com o que foi mostrado na figura 3.1.

Além do algoritmo definido na figura 3.1, é possível definir um diagrama que mostra como as partes que constituem o modelo se relacionam entre si. Este diagrama pode ser visto na figura 3.5.

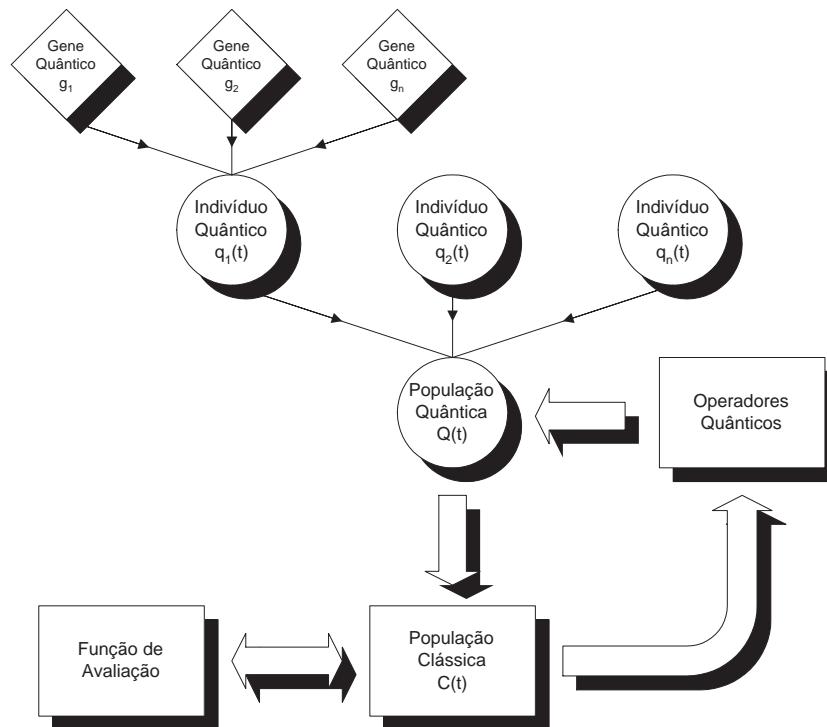


Figura 3.5: Diagrama completo do Sistema Evolutivo com Inspiração Quântica

Este diagrama mostra que a população quântica  $Q(t)$  é composta de indivíduos quânticos que, por sua vez, são formados por genes quânticos. Esta população quântica é usada para gerar uma população clássica que é avaliada e usada para modificar os indivíduos da população quântica em um processo iterativo.

### 3.2

#### Usando o Modelo da Partícula na Caixa com o AEIQ-R

Conforme mostrado no capítulo 2, é possível idealizar um modelo de uma partícula confinada em uma caixa com barreiras de potencial infinito e calcular a função de onda para este sistema físico. Nesta seção, pretende-se mostrar como é possível usar este modelo hipotético para simular um gene quântico do AEIQ-IR.

Para isto, deve-se supor que, para cada gene quântico que se deseja representar, deve-se confinar uma partícula (por exemplo, um elétron) em um poço com barreiras de potencial infinito. Se estas barreiras de potencial infinito forem posicionadas em relação a um referencial qualquer, que permita relacionar a posição desta barreira com os limites do domínio da função que se quer otimizar, pode-se, então, usar a função de onda para a posição do elétron como o gene quântico.

Como exemplo, pode-se imaginar que se deseja otimizar uma função qualquer de duas variáveis  $x$  e  $y$  e que estas variáveis podem assumir valores no intervalo  $[-100, 100]$ . O AEIQ-IR que se pretende usar para otimizar esta função tem dois indivíduos quânticos, cada um com dois genes quânticos. Neste caso, pode-se criar um conjunto de 4 poços de potencial com um elétron confinado (um para cada gene quântico, conforme mencionado anteriormente) e com um nível energético tal que o valor de  $n$  (através do qual é possível definir o número de picos da função de onda, conforme explicado no capítulo 2) seja igual a 1. As barreiras de potencial infinito devem ser, caso se deseje que as funções densidade de probabilidade associadas a cada elétron se estendam por todo o domínio do problema na geração inicial, posicionadas de modo que, em relação a algum referencial pré-definido, as mesmas estejam na posição  $-100$  e  $100$ . Um exemplo de um elétron preso entre barreiras de potencial, sendo utilizado como gene quântico, pode ser visto na figura 3.6.

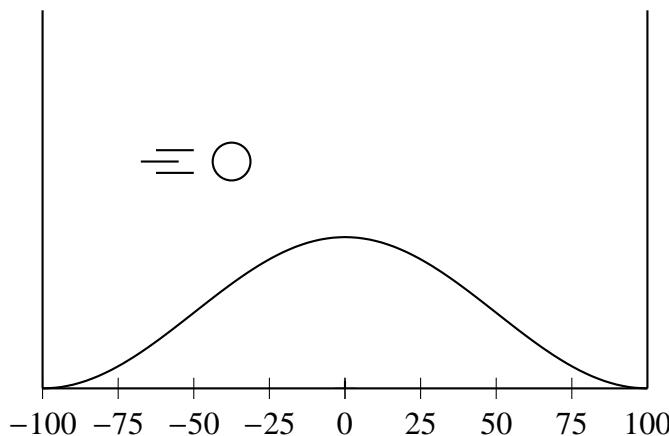


Figura 3.6: Diagrama de um modelo de partícula na caixa para uso no AEIQ-IR.

Nesta figura pode-se ver, além das barreiras de potencial nas posições  $-100$  e  $100$ , a função densidade de probabilidade associada à função de onda desta partí-

cula. O fato desta função de onda ser uma senóide faz com que a probabilidade de que as posições próximas às barreiras de potencial tenham menos chance de serem observadas ao se medir a posição do elétron. Isto, no entanto, não oferece problemas. É sempre possível definir as barreiras de potencial em posições além das fronteiras do domínio, de modo a permitir que as chances das posições mais próximas às bordas tenham maiores chances de serem observadas, descartando-se os genes clássicos que venham a ser, eventualmente, observados fora do domínio. Uma outra opção consiste em criar vários indivíduos quânticos diferentes, posicionando-se as barreiras de potencial ao longo do domínio (por exemplo, poderia-se usar três indivíduos quânticos com as barreiras de potencial nos intervalos  $[-200, 0]$ ,  $[-100, 100]$  e  $[0, 200]$ ).

Para se gerar os indivíduos clássicos, basta observar a posição do elétron dentro de cada um dos poços de potencial. Em outras palavras, a observação da posição do elétron dentro da caixa é equivalente ao processo de observação da população quântica no AEIQ-R descrito na seção anterior. Com os valores obtidos para a posição do elétron nessas observações, é possível ter indivíduos que possam ser avaliados da maneira tradicional. Ou seja, é possível, de fato, substituir a parte com inspiração quântica do algoritmo, por um sistema físico que possa fornecer o efeito quântico desejado.

Após a geração dos indivíduos clássicos e a avaliação dos mesmos, a atualização dos indivíduos quânticos é realizada. Esta atualização consiste, basicamente, em alterar o posicionamento e a distância entre as barreiras de potencial. Por exemplo, pode-se imaginar que, após as avaliações dos indivíduos clássicos, identifique-se que o indivíduo mais bem avaliado tem o gene relativo à variável  $x$  igual a 25. Além disso, define-se que, a cada geração, a largura do gene quântico deva ser igual a 60% da largura na geração anterior (neste caso, portanto, a nova largura será igual a 120). Assim, a nova configuração do gene quântico será aquela na qual as barreiras de potencial estejam localizadas na posição -35 e 85, como mostrado na figura 3.7.

Portanto, através deste exemplo, verifica-se que é possível usar as propriedades e características de um modelo quântico hipotético como um método de implementar os genes quânticos do AEIQ-R. Deseja-se assim tornar mais clara a relação entre o algoritmo implementado e os elementos da física quântica nos quais o mesmo algoritmo se inspira.

### **3.3**

#### **Analogia entre o AEIQ-R e os Algoritmos Culturais**

Uma importante característica do AEIQ-R faz com que o mesmo tenha um comportamento semelhante ao apresentado pelos algoritmos culturais: as funções densidade de probabilidade usadas para representar os genes dos indivíduos quânti-

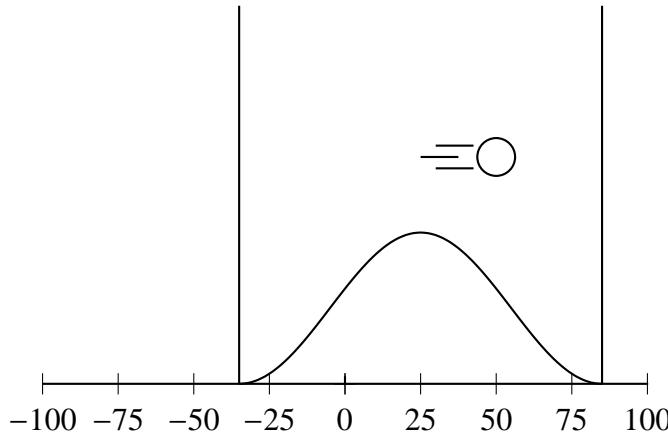


Figura 3.7: Diagrama do modelo de partícula na caixa para uso no AEIQ–R após a atualização do gene quântico.

cos em  $Q(t)$  representam uma forma de conhecimento normativo (esta forma de conhecimento é explicada no capítulo 2). Em outras palavras, este tipo de informação guarda semelhanças funcionais com o modelo de espaço de crenças dos algoritmos culturais. As principais diferenças entre os dois algoritmos estão no fato de que, nos algoritmos culturais, o espaço de crenças é usado para influenciar a avaliação dos indivíduos da população e influenciar também, indiretamente a geração de novos indivíduos. Além disso, os indivíduos da população, por sua vez, atuam no espaço de crenças modificando a sua forma. Já no caso do AEIQ–R, a população quântica (que nesta comparação teria uma funcionalidade equivalente ao espaço de crenças) gera novos indivíduos para a população clássica, influenciando diretamente a criação dos novos indivíduos. De maneira análoga, os indivíduos da população clássica atuam na população quântica, alterando a forma e posição dos genes quânticos.

Além de guiar melhor o processo de otimização, da mesma maneira que nos algoritmos culturais, esta característica sugere uma importante possibilidade do uso do AEIQ–R: como o conhecimento normativo fica armazenado no indivíduo quântico, este indivíduo pode ser utilizado para tarefas de otimização onde seja importante alimentar a população inicial com algum tipo de conhecimento. Um exemplo deste tipo de tarefa de otimização é o aprendizado online. Mais especificamente, pode-se imaginar o seguinte cenário: supondo que se deseja otimizar a fila de embarque de produtos em um porto através do planejamento do uso dos equipamentos e dos píeres que compõem o porto. Uma primeira otimização é feita e uma solução é produzida pelo algoritmo. Em seguida, pequenas alterações no cenário do porto (por exemplo, a quebra de um equipamento) fazem com que uma nova otimização seja necessária. Obviamente, o fato da mudança no cenário não ter sido drástica, faz com que a solução encontrada na otimização anterior, apesar de não ser mais a melhor possível, seja, provavelmente, uma solução acima da média em termos de

avaliação. Em um algoritmo genético convencional, pode-se usar o melhor (ou os melhores) indivíduo encontrado para alimentar a população inicial da nova tarefa de otimização. Com o AEIQ-R, pode-se usar, não só os indivíduos da população clássica final, como também os indivíduos quânticos finais (sendo que os pulsos estão, provavelmente, com a largura próxima de zero e, portanto, devem ser aumentados antes de reiniciar o processo). Desta forma, a realimentação do processo evolutivo deixará de ser pontual e passará a englobar toda uma região promissora com respeito à solução procurada, de maneira similar aos espaços de crença nos algoritmos culturais. A principal diferença entre a população quântica e o modelo de espaço de crenças é que o conhecimento normativo é tratado de maneira probabilística no primeiro modelo e de maneira uniforme no segundo.

### **3.4**

#### **Modelo Neuro-Evolutivo – Aprendizado Supervisionado**

O AEIQ-R pode ser utilizado para otimizar qualquer tipo de função numérica. Em geral, os resultados obtidos pelo algoritmo são superiores aos obtidos pelos algoritmos genéticos clássicos com relação ao tempo de convergência do processo de otimização (alguns resultados do uso deste modelo para a otimização deste tipo de problemas são mostrados no capítulo 4). Além de apresentar este melhor desempenho em diversas situações, o modelo aqui proposto se destaca também nos processos neuro-evolutivos, devido ao fato de não apresentar problemas de super-treinamento, não necessitar de um conjunto de dados para validação e por ser capaz de realizar o aprendizado de forma *online*, conforme mencionado anteriormente.

O modelo aqui proposto usa o AEIQ-R para treinar uma rede neural recorrente (FRNN – *Fully-Recurrent Neural Network*), otimizando os pesos e a topologia da mesma, de modo a minimizar o erro na saída, aprendendo, assim, a reconhecer os padrões de entrada fornecidos.

A rede neural recorrente é uma rede que tem realimentação nas suas ligações, da saída dos processadores para as entradas dos mesmo, além das ligações das entradas para os processadores e dos processadores para as saídas, como nas redes *multilayer perceptron* (Haykin99). Este tipo de rede apresenta bons resultados em problemas de previsão de séries (Haykin99) e em problemas de controle (Gomez03), pelo fato de que as realimentações introduzem um efeito de memória nestas redes (Gomez03). A figura 3.8 mostra um diagrama de uma rede neural recorrente com duas entradas, três processadores e uma saída (os *biases* não são mostrados para não sobrecarregar o diagrama).

Outra importante característica de uma rede neural recorrente é que, através da manipulação dos pesos, em particular ao se zerar alguns deles, é possível redefinir a topologia da rede neural, e até mesmo reduzir a rede recorrente a uma rede

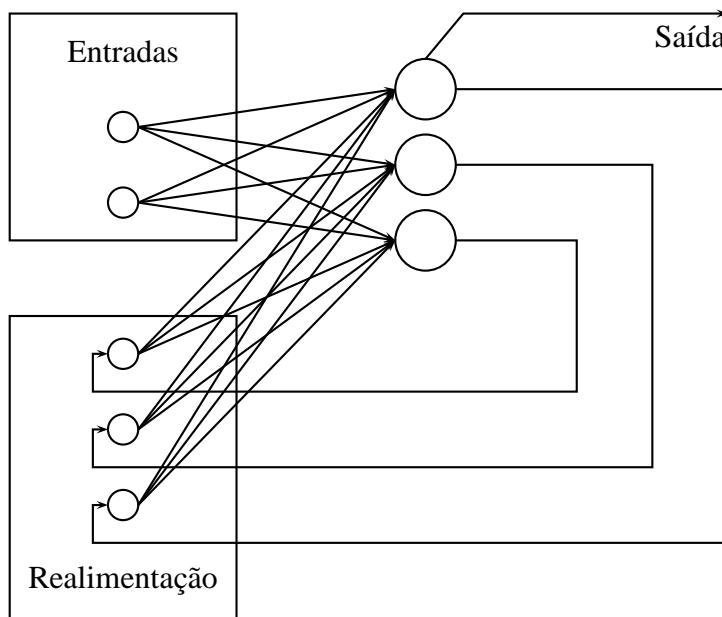


Figura 3.8: Diagrama mostrando uma rede neural recorrente. Este diagrama não mostra as ligações dos *biases*.

*multilayer perceptron*. Por exemplo, na rede mostrada na figura 3.8, caso todos os pesos das entradas para o primeiro neurônio sejam zero e caso todos os pesos de realimentação sejam iguais a zero, a rede irá se transformar numa rede *multilayer perceptron* com duas entradas, dois processadores na camada escondida e um processador de saída.

O cromossomo clássico usado para fazer o treinamento desta rede contém todos os pesos necessários para todas as conexões da rede neural que se quer treinar. O cálculo do número de genes necessários para o cromossomo que representa os pesos da rede neural pode ser feito através da equação 3-7:

$$num\_genes = N_e \times N_p + N_p^2 + N_p \quad (3-7)$$

Onde  $N_e$  indica o número de entradas e  $N_p$  indica o número de processadores. O primeiro termo desta equação calcula o número de pesos necessários para realizar as ligações entre as entradas e os processadores. O segundo termo, calcula o número de pesos necessários para realizar as ligações recorrentes entre os processadores e o último termo é o número de *biases* necessários (um para cada processador).

Assim, o indivíduo quântico necessário para gerar os indivíduos clássicos, deve ser inicializado de modo que a função densidade de probabilidade permita que os valores dos genes clássicos observados estejam dentro de uma faixa aceitável como pesos para uma rede neural. Neste trabalho, em todos os estudos de caso mostrados no capítulo 4, o domínio usado para os pesos está no intervalo  $[-1, 1]$ .

Cada cromossomo clássico gerado deve, naturalmente, ser avaliado. Esta

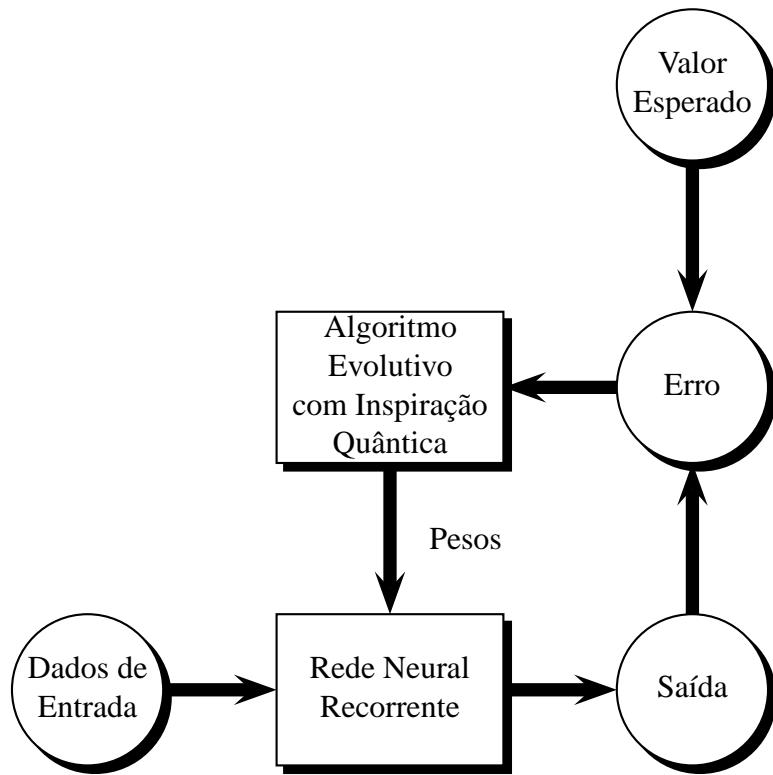


Figura 3.9: Modelo de Aprendizado Supervisionado usando o AEIQ–R.

avaliação é feita construindo-se a rede neural recorrente com os pesos determinados pelo indivíduo clássico. Os padrões de entrada são então apresentados e a saída obtida pela rede é comparada com as saídas desejadas. Esta comparação permite que se faça o cálculo do erro médio, que pode ser usado como o valor da avaliação do indivíduo clássico em questão.

Na figura 3.9 é mostrado um diagrama de blocos, que ilustra o funcionamento do modelo de aprendizado supervisionado para redes neurais recorrentes usando o AEIQ–R.

O número de neurônios e a topologia de uma rede neural são dois importantes parâmetros na configuração da mesma. Em geral, a topologia deve ser determinada empiricamente ou pela experiência pessoal do usuário. Quando se usa algoritmos como o *backpropagation* (Haykin99) para realizar o treinamento das redes, este parâmetro pode afetar seriamente o desempenho das mesmas. Processadores em excesso podem fazer com que a rede memorize o conjunto de treinamento ao invés de aprender a função que mapeia as entradas nas saídas, resultando em uma má capacidade de generalização por parte da mesma. Por outro lado, o uso de poucos processadores pode retardar o processo de aprendizado ou até mesmo inviabilizá-lo.

Para se contornar este problema, o modelo de treinamento usando o AEIQ–

R usa um método de aprendizado descrito em (Gomez03). Neste método, um operador especial chamado *lesion* é utilizado durante o processo de aprendizado. Este operador funciona da seguinte forma: quando a evolução pára de progredir por um certo número de gerações (este número de gerações  $\eta$  é definido pelo usuário), a melhor rede encontrada é reavaliada, removendo cada um de seus neurônios, um por vez. Se a avaliação desta nova rede não piorar mais do que um determinado nível após a remoção do neurônio  $i$ , pode-se considerar que este neurônio não é importante para a rede e o mesmo é removido. Se nenhum neurônio puder ser removido, isto pode significar que a rede está precisando de mais neurônios para aprender corretamente. Sendo assim, mais um neurônio é inserido na rede, com todos os pesos inicialmente iguais a zero. A inserção de neurônios é feita com peso igual a zero, para que isso não altere a avaliação dos indivíduos imediatamente (zerar os pesos da entrada e da saída de um neurônio equivale a remover o neurônio da rede). Além disso, conforme já foi mencionado anteriormente, dado que a rede neural recorrente pode se transformar em uma rede com múltiplas camadas, conclui-se que, além do número de neurônios, o número de camadas da rede neural também será otimizado automaticamente, de acordo com os pesos que forem encontrados para determinadas ligações entre os processadores.

Através do uso deste operador e da capacidade do processo de treinamento de encontrar a topologia mais adequada automaticamente, espera-se que o AEIQ–R seja capaz de otimizar os pesos da rede neural com um tempo computacional muito menor, já que através deste método de aprendizado, não é necessário realizar o processo de identificação do número ideal de processadores da rede, o que exige um esforço computacional relativamente alto em relação ao processo de aprendizado da rede como um todo.

Finalmente, é importante destacar que, o fato de se usar o AEIQ–R como método de treinamento, torna desnecessário o uso de um conjunto de dados para validação da rede neural. De acordo com (Bishop95), devido ao fato de as redes bayesianas usarem distribuições de probabilidade para a determinação dos pesos, a otimização dos valores ideais para os coeficientes de regularização (os coeficientes de regularização são usados para minimizar a ocorrência de super-treinamento) da rede pode ser feita sem a necessidade do uso de conjuntos de validação. Como o AEIQ–R também usa distribuições de probabilidade para encontrar estes pesos, pode-se concluir que este método de aprendizado também não necessita do uso destes conjuntos de validação para o aprendizado.

### 3.5

### Modelo Neuro-Evolutivo – Aprendizado por Reforço

Nesta seção, pretende-se apresentar um modelo de aprendizado por reforço, onde se usa redes neurais para aprender a realizar tarefas de controle e para as quais o reforço só é fornecido quando o agente alcança o estado final ou o estado de falha. Neste caso, o objetivo é treinar a rede para que a mesma seja capaz de levar um determinado sistema de um estado a outro (por exemplo, equilibrar um pêndulo, regular a temperatura de um ambiente, entre outros casos).

Ao contrário do modelo de aprendizado supervisionado, neste caso não existem dados de saída para comparação e cálculo do erro, mas sim um estado (ou um conjunto de estados) finais que se deseja alcançar. A rede neural então lê o estado inicial deste sistema, e dá uma resposta na sua saída. Essa resposta é usada pelo atuador para mudar o estado do sistema que então será usado como nova entrada para a rede neural. Esse processo é repetido sucessivamente até que a rede neural consiga levar o sistema ao estado final desejado ou que um certo número de passos seja executado. O diagrama da figura 3.10 resume o funcionamento deste modelo.

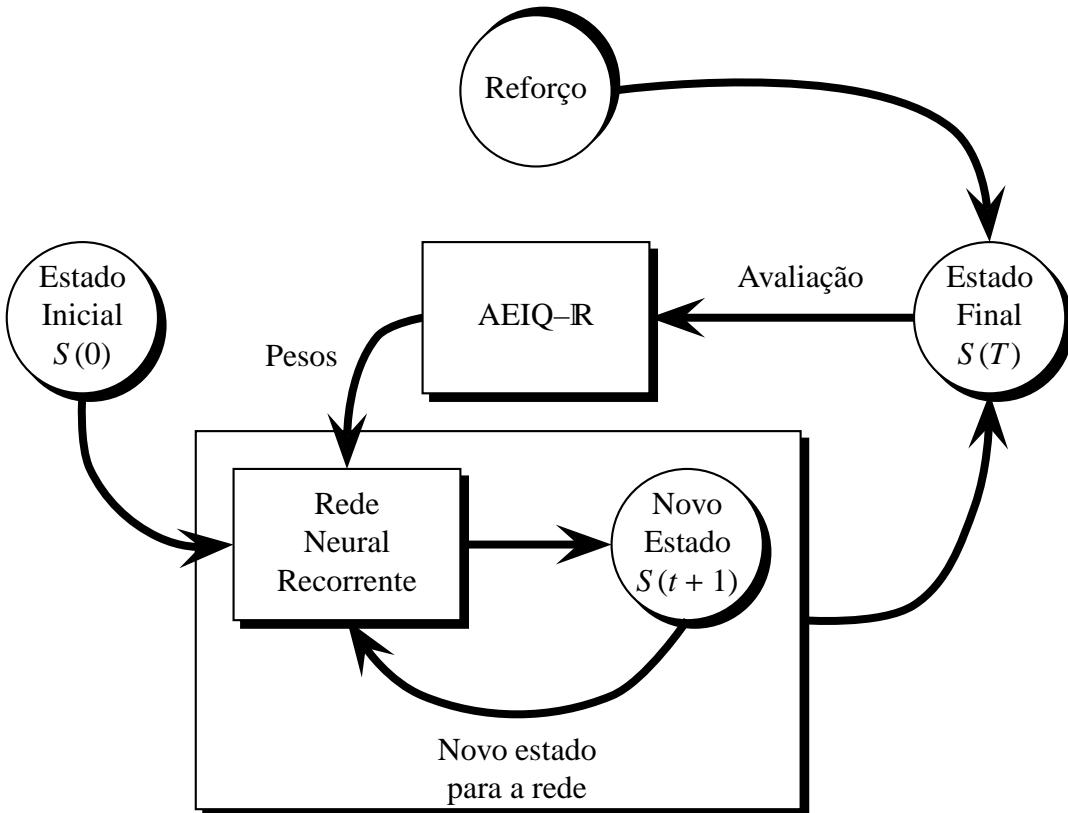


Figura 3.10: Modelo de Aprendizado por Reforço usando o AEIQ-R.

Neste diagrama, pode-se observar que, a partir de um estado inicial, a rede neural é usada para gerar uma saída que indica qual ação o sistema deve tomar. Esta ação muda o estado corrente do sistema e este novo estado é usado como entrada

para a mesma rede neural. Uma nova saída é gerada e desta forma o processo é repetido continuamente até que um certo número limite de passos tenha sido executado ou até que o sistema atinja o estado final desejado. O sistema é então avaliado pelo AEIQ–R (em geral, usando-se o número de passos necessários para se atingir o objetivo) que, por sua vez, através do processo de evolução, gera um novo conjunto de pesos para a rede neural. Estes novos pesos são então usados para repetir o processo de controle novamente.

Este modelo de aprendizado usa o mesmo método de treinamento apresentado na seção anterior. Assim, o algoritmo de aprendizado também é capaz de definir a topologia e o número de neurônios necessários para a rede neural.

## 4

# Estudos de Caso

Neste capítulo são apresentados diversos resultados em ensaios realizados com o AEIQ–R. Estes ensaios foram realizados com os objetivos principais de comparar o desempenho do AEIQ–R com outros algoritmos tradicionais de otimização numérica global e medir o desempenho do AEIQ–R aplicado à neuro-evolução tanto em problemas de aprendizado supervisionado (previsão de séries), quanto aprendizado por reforço. Além disso, deseja-se medir a influência dos parâmetros usados no AEIQ–R no processo de otimização.

Para o teste de desempenho em otimização de funções e o teste de influência dos parâmetros, foi selecionado um conjunto de funções de *benchmark* tradicionalmente usadas para medir o desempenho de algoritmos de otimização. Este conjunto de funções foi escolhido também em função da disponibilidade de resultados de cada um dos algoritmos com os quais se deseja realizar uma comparação de desempenho.

Para o teste do aprendizado supervisionado, um problema de previsão de vazões em bacias hidrográficas foi selecionado, para permitir a fácil comparação com um treinamento supervisionado utilizando o algoritmo mais usado, o *backpropagation* (Haykin99).

Finalmente, para o teste do aprendizado por reforço, dois problemas tradicionais da área de controle foram usados. Nestes dois problemas, os resultados obtidos pelo AEIQ–R também são comparados com outros métodos baseados em aprendizado por reforço.

### 4.1

#### Otimização de Funções

Os testes de desempenho de otimização foram realizados de modo a permitir a comparação de resultados do AEIQ–R com outros algoritmos de otimização global tradicionais baseados em evolução. Os algoritmos usados para comparação são:

- Algoritmos Genéticos Tradicionais com Representação Real
- Algoritmos Genéticos com Inspiração Quântica e Representação Binária
- Programação Evolutiva

- Programação Evolutiva Rápida
- Enxame de Partículas
- Evolução Diferencial

Além dos parâmetros já descritos no capítulo 3, o AEIQ–R usado nesta seção também exige que alguns parâmetros extras sejam configurados. Em particular, para a tarefa de recombinação, é necessário especificar a taxa na qual a recombinação ocorre. O AEIQ–R implementado usa recombinação aritmética (Michalewicz94) entre os indivíduos gerados e os indivíduos da população anterior. Este operador sorteia um número  $r$  no intervalo  $[0, 1]$  e gera dois indivíduos novos,  $p'_1$  e  $p'_2$ , a partir dos genitores  $p_1$  e  $p_2$  de acordo com a equação:

$$p'_1 = rp_1 + (1 - r)p_2 \quad (4-1)$$

$$p'_2 = rp_2 + (1 - r)p_1 \quad (4-2)$$

Com relação ao processo de substituição dos indivíduos da população clássica a cada geração (passo 10 do algoritmo descrito no capítulo 3), decidiu-se optar pela solução mais tradicional de usar uma técnica de *steady-state*, que consiste em manter parte da população da geração anterior na nova população gerada. Neste trabalho, esta técnica elimina os  $n$  piores indivíduos e os substitui por  $n$  novos indivíduos gerados. Assim, apesar da população clássica ter um tamanho maior do que  $n$ , apenas  $n$  indivíduos são avaliados a cada geração do algoritmo (com exceção da primeira, onde a população clássica é criada). Este conjunto de indivíduos que é substituído a cada geração é chamado de *gap* e será representado pela letra  $\gamma$ .

Finalmente, durante o processo de atualização dos indivíduos quânticos utilizando-se os indivíduos clássicos (passo 12 do algoritmo descrito no capítulo 3), decidiu-se usar uma *taxa de atualização* que determina a probabilidade que um determinado gene quântico tem de ser atualizado pelo gene clássico de referência. Em outras palavras, deve-se definir a probabilidade de um gene quântico ter sua posição modificada a cada geração. Por exemplo, uma taxa de 85% indica que, em média, 15% dos genes de um indivíduo quântico não terão seus centros e larguras modificados em uma geração. Esta taxa será representada pela letra  $\zeta$ .

#### 4.1.1

#### **Comparação com Evolução Diferencial e Enxame de Partículas**

A comparação com evolução diferencial e enxame de partículas é feita a partir de resultados retirados de (Tasgetiren05) e (Tasgetiren05A), respectivamente. O objetivo deste teste é analisar o desempenho do AEIQ–R quando comparado com dois importantes algoritmos de otimização global. Os resultados são comparados executando-se dois experimentos diferentes, com um número total de avaliações

igual a  $10^3$  e  $10^4$ . Os dois experimentos são necessários para permitir a identificação de métodos que convirjam muito rapidamente para um mínimo local e que apresentem resultados melhores em poucas gerações, mas que não consigam levar a otimização adiante quando se aumenta o número de avaliações totais. Cada função é testada com 30 variáveis de entrada. As funções usadas para comparação são descritas no Anexo 1, onde são também apresentados os respectivos gráficos para as funções com duas variáveis.

O conjunto de parâmetros de configuração do AEIQ–R usado nestes testes, foi selecionado através de alguns experimentos para determinar a melhor configuração, usando-se a função  $f_1$  como avaliação. Em outras palavras, a parametrização utilizada não é, necessariamente, a melhor parametrização para cada uma das funções otimizadas. Isto foi feito devido ao fato do conjunto de funções de teste ser muito extenso. A parametrização do AEIQ–R é mostrada na tabela 4.1. Os resultados obtidos realizando-se um total de  $10^3$  e  $10^4$  avaliações são mostrados nas tabelas 4.2 e 4.3 respectivamente. Estas tabelas mostram o erro médio obtido após 25 rodadas em relação ao mínimo global conhecido da função, o desvio padrão dos erros obtidos e a diferença percentual entre o valor obtido pelo melhor algoritmo com o valor obtido pelo segundo melhor algoritmo.

Função Densidade de Probabilidade	Pulso Quadrado
Número de Indivíduos Quânticos	5
Número de Indivíduos Clássicos	10
$Gap \gamma$	5
Taxa de Recombinação	66%
Intervalo de Atualização da População Quântica	10 gerações
Taxa de Atualização dos Indivíduos Quânticos $\zeta$	16.6%

Tabela 4.1: Configuração do AEIQ–R para comparação com evolução diferencial e enxame de partículas.

	Evolução Diferencial		Enxame de Partículas		AEIQ–R		Perc.
	Erro Médio	Desvio Padrão	Erro Médio	Desvio Padrão	Erro Médio	Desvio Padrão	
$f_1$	$5.440E + 04$	$7.208E + 03$	$2.331E + 04$	$4.886E + 03$	<b><math>2.286E + 04</math></b>	<b><math>3.813E + 03</math></b>	1.95%
$f_2$	$8.592E + 04$	$1.742E + 04$	$6.041E + 04$	$1.203E + 04$	<b><math>3.570E + 04</math></b>	<b><math>5.594E + 03</math></b>	40.90%
$f_3$	$1.009E + 09$	$2.390E + 08$	$4.970E + 08$	$1.517E + 08$	<b><math>3.553E + 08</math></b>	<b><math>7.095E + 07</math></b>	28.51%
$f_4$	$9.984E + 04$	$1.864E + 04$	$6.810E + 04$	$1.521E + 04$	<b><math>4.627E + 04</math></b>	<b><math>8.760E + 03</math></b>	32.06%
$f_5$	$2.395E + 04$	<b><math>2.631E + 03</math></b>	<b><math>1.876E + 04</math></b>	$3.792E + 03$	$2.546E + 04$	$3.381E + 03$	26.32%
$f_6$	$2.600E + 10$	$9.428E + 09$	$6.354E + 09$	$2.852E + 09$	<b><math>5.501E + 09</math></b>	<b><math>1.846E + 09</math></b>	13.42%
$f_8$	$2.121E + 01$	$6.801E - 02$	<b><math>2.118E + 01</math></b>	$7.595E - 02$	$2.119E + 01$	<b><math>6.070E - 02</math></b>	0.03%
$f_9$	$4.045E + 02$	$2.9405E + 01$	$3.398E + 02$	$2.191E + 01$	<b><math>3.072E + 02</math></b>	<b><math>1.957E + 01</math></b>	9.59%
$f_{10}$	$4.696E + 02$	$3.163E + 01$	$3.521E + 02$	<b><math>2.498E + 01</math></b>	<b><math>3.039E + 02</math></b>	$5.469E + 01$	13.69%
$f_{11}$	$4.554E + 01$	$1.282E + 00$	$4.580E + 01$	$1.448E + 00$	<b><math>4.186E + 01</math></b>	<b><math>1.821E + 00</math></b>	8.08%
$f_{12}$	$1.515E + 06$	$2.018E + 05$	$1.240E + 06$	$1.803E + 05$	<b><math>2.691E + 05</math></b>	<b><math>9.929E + 04</math></b>	78.31%
$f_{13}$	$2.283E + 05$	$1.061E + 05$	$1.376E + 04$	$1.158E + 04$	<b><math>1.192E + 01</math></b>	<b><math>5.993E + 01</math></b>	99.91%
$f_{14}$	$1.416E + 01$	<b><math>1.483E - 01</math></b>	$1.408E + 01$	$1.523E - 01$	<b><math>1.389E + 01</math></b>	$1.832E - 01$	1.37%

Tabela 4.2: Resultado comparativo entre o AEIQ–R, o algoritmo de evolução diferencial e o de enxame de partículas com 1000 avaliações de função.

	Evolução Diferencial		Enxame de Partículas		AEIQ-IR		
	Erro Médio	Desvio Padrão	Erro Médio	Desvio Padrão	Erro Médio	Desvio Padrão	Perc.
$f_1$	$1.279E + 04$	$2.628E + 03$	$1.025E + 03$	$6.393E + 02$	<b><math>1.232E + 01</math></b>	<b><math>7.060E + 00</math></b>	99.88%
$f_2$	$4.919E + 04$	$9.254E + 03$	$1.554E + 04$	$4.128E + 03$	<b><math>1.097E + 04</math></b>	<b><math>1.878E + 03</math></b>	29.41%
$f_3$	$3.268E + 08$	$8.828E + 07$	$7.558E + 07$	$3.554E + 07$	<b><math>2.456E + 07</math></b>	<b><math>5.439E + 06</math></b>	67.50%
$f_4$	$5.082E + 04$	$8.594E + 03$	$2.227E + 04$	$5.343E + 03$	<b><math>2.052E + 04</math></b>	<b><math>3.078E + 03</math></b>	7.86%
$f_5$	$1.047E + 04$	<b><math>1.345E + 03</math></b>	$1.026E + 04$	$3.720E + 03$	<b><math>9.323E + 03</math></b>	$1.379E + 03$	9.19%
$f_6$	$1.656E + 09$	$8.344E + 08$	$2.504E + 07$	$2.063E + 07$	<b><math>8.364E + 04</math></b>	<b><math>8.323E + 04</math></b>	99.67%
$f_8$	$2.109E + 01$	$5.957E - 02$	<b><math>2.106E + 01</math></b>	<b><math>5.543E - 02</math></b>	$2.107E + 01$	$6.080E - 02$	0.04%
$f_9$	$2.761E + 02$	$2.175E + 01$	<b><math>1.040E + 02</math></b>	$2.698E + 01$	$1.533E + 02$	<b><math>2.069E + 01</math></b>	32.18%
$f_{10}$	$3.349E + 02$	$2.614E + 01$	$1.962E + 02$	$4.852E + 01$	<b><math>1.780E + 02</math></b>	<b><math>2.698E + 01</math></b>	9.27%
$f_{11}$	$4.285E + 01$	<b><math>9.645E - 01</math></b>	$3.651E + 01$	$3.835E + 00$	<b><math>3.333E + 01</math></b>	$3.142E + 00$	8.70%
$f_{12}$	$7.100E + 05$	$1.365E + 05$	$1.234E + 05$	$7.206E + 04$	<b><math>7.219E + 04</math></b>	<b><math>3.421E + 04</math></b>	41.50%
$f_{13}$	$4.005E + 03$	$1.868E + 03$	$2.492E + 01$	$5.348E + 00$	<b><math>1.026E + 01</math></b>	<b><math>1.137E + 00</math></b>	58.82%
$f_{14}$	$1.390E + 01$	<b><math>1.323E - 01</math></b>	$1.358E + 01$	$2.131E - 01$	<b><math>1.321E + 01</math></b>	$2.183E - 01$	2.77%

Tabela 4.3: Resultado comparativo entre o AEIQ-IR, o algoritmo de evolução diferencial e o de enxame de partículas com 10000 avaliações de função.

Os resultados usando  $10^3$  avaliações foram, em média, 23.2% melhores, enquanto que os resultados usando  $10^4$  avaliações foram, em média, 30.95% melhores do que os resultados dos outros algoritmos. A tabela 4.4 mostra em termos percentuais, o quanto melhor o AEIQ-IR foi em relação ao segundo melhor algoritmo (a maioria das comparações foi feita com o PSO), em relação às características das funções usadas como teste (unimodal, multimodal, separável, não-separável, rotacionada e não-rotacionada). É importante destacar que o resultado obtido com relação a separabilidade das funções não é representativo, já que apenas duas funções do conjunto de teste são separáveis. De qualquer modo, os resultados são colocados na tabela para fins de ilustração.

	$10^3$ avaliações	$10^4$ avaliações
Unimodal	15.42%	42.76%
Multimodal	28.04%	23.53%
Separável	5.77%	33.85%
Não-Separável	26.35%	30.42%
Rotacionada	10.32%	17.64%
Não-Rotacionada	31.22%	39.26%

Tabela 4.4: Tabela que indica a melhora percentual média ao se usar o AEIQ-IR, comparando-o com o segundo melhor algoritmo, em relação às características das funções de teste.

Estes resultados mostram que, ao se aumentar o número de avaliações permitidas para o algoritmo em funções unimodais, o desempenho do mesmo melhora consideravelmente. Isto sugere que o algoritmo tem um bom comportamento com relação à buscas locais. Por outro lado, o algoritmo apresenta uma pequena perda de desempenho quando as funções são multimodais. Ainda assim, o seu desempenho é bem superior quando funções multimodais estão sendo otimizadas. Isto reforça a sua característica de ser um algoritmo de busca global. Comparando-se também os resultados obtidos com  $10^3$  e  $10^4$  avaliações, pode-se notar que, em geral, ao permi-

tir que o AEIQ–R faça mais avaliações, o seu desempenho melhora com relação aos outros algoritmos. Isto sugere que o AEIQ–R usa, quando bem configurado, uma estratégia de busca mais conservadora, evitando convergências prematuras para mínimos locais. Este comportamento pode ser controlado através da manipulação do valor que determina de quantas em quantas gerações a largura dos pulsos deve ser modificada. Os resultados apresentados nesta subseção serão discutidos em mais detalhes na subseção 4.1.4.

#### 4.1.2

#### **Comparação com o AEIQ-B, Programação Evolutiva Clássica, Programação Evolutiva Rápida e Algoritmos Genéticos Convencionais**

A comparação com a programação evolutiva clássica (*Classical Evolutionary Programming* - CEP), programação evolutiva rápida (*Fast Evolutionary Programming* - FEP) e com o AEIQ–B é feita a partir de resultados tirados de (Yao99) (CEP e FEP) e (Han04) (AEIQ–B). Os resultados usando algoritmos genéticos foram, por sua vez, obtidos usando-se o software GACOM, desenvolvido no ICA. Os resultados são comparados da seguinte forma: para cada função são executados testes com um número total de avaliações previamente definido e com um número de experimentos igual a 50. Cada função é testada com 30 variáveis de entrada. As funções usadas para comparação são descritas no anexo 2, onde são também apresentados os respectivos gráficos para as funções com duas variáveis.

Foram usados dois conjuntos de configurações diferentes para o AEIQ–R neste teste, já que o número de funções de teste é mais reduzido. Estas configurações são mostradas nas tabelas 4.5 e 4.6 e foram determinadas através de testes. A principal diferença na parametrização está nas taxas de recombinação e na taxa de atualização dos indivíduos quânticos. Durante os testes, foi observado que para funções separáveis, estas duas taxas podem ter valores menores enquanto que, para funções não-separáveis, estas duas taxas devem ter valores maiores. A configuração 1 foi usada nas funções  $f_{griewank}$ ,  $f_{schwefel}$  e  $f_{rosenbrock}$ , enquanto que a configuração 2 foi usada nas funções  $f_{sphere}$ ,  $f_{ackley}$ ,  $f_{rastrigin}$ . Os resultados da otimização são mostrados na tabela 4.7.

O algoritmo genético convencional foi executado usando-se os operadores de mutação uniforme e não-uniforme (Michalewicz94), os operadores de cruzamento aritmético e uniforme. Os operadores de mutação foram usados com uma taxa de 10%, enquanto que os operadores de cruzamento foram usados com uma taxa de 80%. O método de seleção usado foi a roleta e também foi usado um operador de *steady-state* com um *gap* de 20%.

Os resultados apresentados mostram que o AEIQ–R obteve um resultado nitidamente superior ao dos outros algoritmos utilizados. O resultado foi, em média,

Função Densidade de Probabilidade	Pulso Quadrado
Número de Indivíduos Quânticos	5
Número de Indivíduos Clássicos	100
$Gap \gamma$	99
Taxa de Recombinação	66%
Intervalo de Atualização da População Quântica	10 gerações
Taxa de Atualização dos Indivíduos Quânticos $\zeta$	66.6%

Tabela 4.5: Configuração 1 do AEIQ–R para comparação com programação evolutiva clássica, programação evolutiva rápida, algoritmos genéticos convencionais e o algoritmo evolutivo com inspiração quântica usando representação binária.

Função Densidade de Probabilidade	Pulso Quadrado
Número de Indivíduos Quânticos	5
Número de Indivíduos Clássicos	100
$Gap \gamma$	99
Taxa de Recombinação	3.33%
Intervalo de Atualização da População Quântica	1 geração
Taxa de Atualização dos Indivíduos Quânticos $\zeta$	3.33%

Tabela 4.6: Configuração 2 do AEIQ–R para comparação com programação evolutiva clássica, programação evolutiva rápida, algoritmos genéticos convencionais e o algoritmo evolutivo com inspiração quântica usando representação binária.

Função	AEIQ–B	FEP	CEP	GA	AEIQ–R	Perc.
$f_{sphere}$	Média	$1.8E - 04$	$5.7 - 04$	$2.2E - 04$	$4.71E + 00$	<b>1.99E - 06</b>
Aval.= 150000	Desvio Padrão	$1.3E - 04$	$1.3E - 04$	$5.9E - 04$	n.d.	<b>9.50E - 06</b>
$f_{ackley}$	Média	$2.5E - 03$	$1.8E - 02$	9.2	$4.71E - 01$	<b>4.26E - 05</b>
Aval.= 150000	Desvio Padrão	$8.1E - 04$	$2.1E - 03$	2.8	n.d.	<b>1.39E - 04</b>
$f_{Griewank}$	Média	$3.6E - 02$	$1.6E - 02$	$8.6E - 02$	$1.03E + 00$	<b>9.42E - 06</b>
Aval.= 200000	Desvio Padrão	$3.2E - 02$	$2.2E - 02$	0.12	n.d.	<b>1.55e - 05</b>
$f_{Rastrigin}$	Média	$3.9E - 02$	$4.6E - 02$	89.0	$4.40E - 01$	<b>1.65E - 11</b>
Aval.= 500000	Desvio Padrão	$1.9E - 01$	$1.2E - 02$	23.1	n.d.	<b>3.39E - 12</b>
$f_{schwefel}$	Média	$3.8E - 04$	14.987	4652.3	$1.77E + 00$	<b>8.13E - 09</b>
Aval.= 900000	Desvio Padrão	$3.0E - 09$	52.6	634.5	n.d.	<b>0</b>
$f_{Rosenbrock}$	Média	11.73	5.06	6.17	$14.15E + 00$	<b>2.52</b>
Aval.= 2000000	Desvio Padrão	18.36	5.87	13.61	n.d.	<b>4.43</b>

Tabela 4.7: Resultado comparativo entre o AEIQ–R , o AEIQ–B (QEA), programação evolutiva rápida (FEP), programação evolutiva clássica (CEP) e algoritmos genéticos convencionais (GA). Os itens com a designação “n.d.” indicam que o dado não está disponível.

95.95% melhor do que os resultados observados no segundo melhor algoritmo. Em particular, na comparação com estes algoritmos, o AEIQ–R obteve valores finais muito mais consistentes, o que pode ser observado comparando-se os desvios padrões obtidos. Estes resultados são discutidos em mais detalhes na subseção 4.1.4.

#### 4.1.3

#### Testes de Esforço Computacional

Esta série de experimentos se propõe a identificar a razão na qual o esforço computacional de otimização pelo AEIQ–R cresce quando se aumenta o número de variáveis do problema que se deseja otimizar. Para se alcançar este objetivo, os experimentos foram realizados da seguinte maneira:

- Foram selecionadas duas funções diferentes para otimizar: uma unimodal, com variáveis separáveis ( $f_{sphere}$  – fácil), e outra multimodal, com variáveis não-separáveis ( $f_{griewank}$  – difícil);
- Definiu-se o melhor conjunto de parâmetros para otimizar cada uma das funções com 10 variáveis;
- Definiu-se um valor-alvo para a função que se quer otimizar;
- Executou-se o algoritmo até alcançar o valor-alvo definido. A execução é feita 20 vezes e o valor médio do número de avaliações necessárias para alcançar o valor-alvo é calculado;
- Aumentou-se o número de variáveis de modo a se obter resultados para as funções com 10, 50, 100, 250, 500, 1000 e 2000 variáveis. Para cada um destes testes, calcula-se a média de avaliações necessárias em 20 experimentos para alcançar o mesmo valor-alvo.

As funções utilizadas para teste foram a  $f_{sphere}$  e a  $f_{griewank}$  e o valor alvo utilizado para as duas funções foi 1 (o mínimo global destas funções é 0). Estas funções foram escolhidas, por se tratarem de duas funções com características diferentes. A primeira é unimodal e separável, enquanto que a segunda é multimodal e não-separável. A parametrização da função não é alterada ao longo do teste, de modo a evitar a influência positiva desta parametrização nos testes com maior número de variáveis. Em outras palavras, uma parametrização mal feita para o problema com 10 variáveis e uma parametrização melhor executada para o problema com 2000 variáveis pode produzir uma distorção no resultado. As figuras 4.1 e 4.2 mostram os resultados obtidos neste teste.

Estes gráficos sugerem que, para as duas funções apresentadas e para um conjunto de até 2000 variáveis, o esforço computacional cresce, aproximadamente, de forma linear com o número de variáveis. Em outras palavras, a complexidade

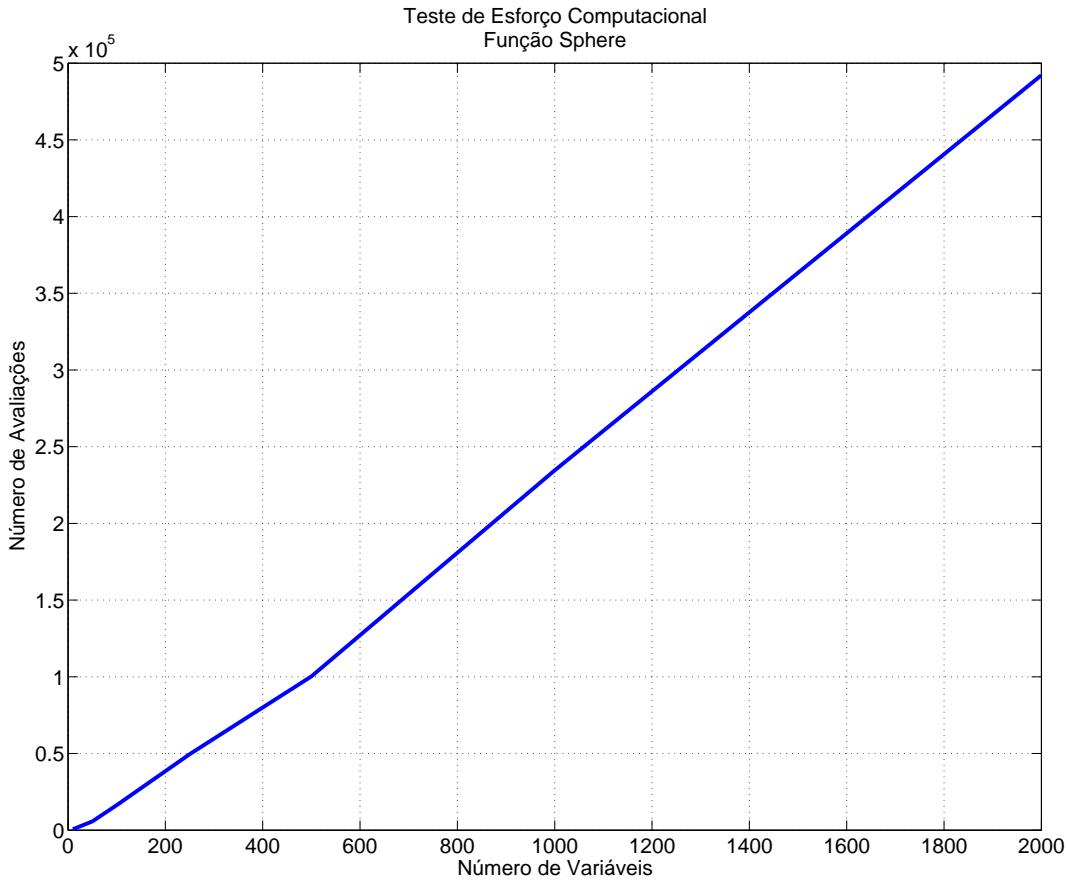


Figura 4.1: Esforço computacional para otimização da função  $f_{sphere}$ .

computacional do algoritmo para estas funções e para o número de dimensões utilizadas é, aproximadamente,  $O(n)$ . Este teste sugere que o AEIQ–R não tenha, talvez, uma complexidade exponencial na otimização de problemas. Em outras palavras, a complexidade do AEIQ–R não parece ser do tipo  $O(a^n)$ . Um exame analítico desta propriedade e uma extensão dos testes aqui realizados podem ajudar a determinar se esta propriedade é válida para qualquer função que se deseja otimizar ou para que grupos de funções esta propriedade é válida.

#### 4.1.4 Discussão dos Resultados

##### Desempenho

Com relação ao desempenho do AEIQ–R quando comparado aos outros algoritmos, pode-se observar que, no geral, o desempenho do algoritmo apresentado aqui é superior aos métodos tradicionalmente usados. O primeiro ponto importante é o fato de que o AEIQ–R obteve um desempenho muito superior ao AEIQ–B para a otimização dos problemas de otimização numérica aqui mostrados. Isto demonstra a importância de um algoritmo com uma representação específica para números reais.

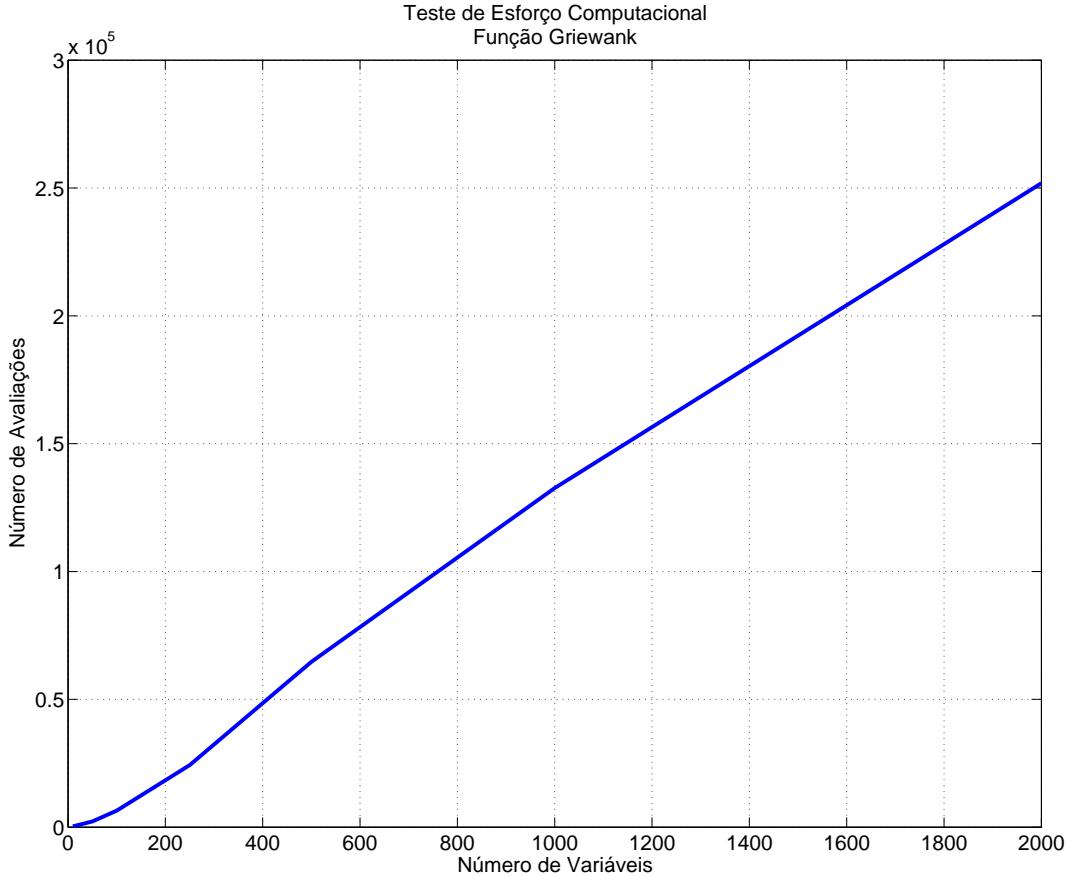


Figura 4.2: Esforço computacional para otimização da função  $f_{griewank}$ .

Comparado com o AEIQ– $\mathcal{B}$ , o AEIQ–R obteve, pelo menos, um resultado duas ordens de grandeza menor e, em alguns casos, como no teste com a função  $f_{rastrigin}$ , o AEIQ–R obteve um resultado nove ordens de grandeza menor. Os resultados na comparação com os dois algoritmos de programação evolutiva também foram expressivamente melhores para todas as funções examinadas.

Os resultados comparativos com a evolução diferencial e o enxame de partículas também mostraram um desempenho superior do AEIQ–R. Apesar de alguns resultados não terem sido tão expressivamente superiores em alguns dos problemas, como foram na comparação com o AEIQ– $\mathcal{B}$  e com os algoritmos de programação evolutiva, é importante ressaltar que a otimização das 14 funções foi feita com a mesma parametrização. Mesmo assim, na maioria dos problemas, o AEIQ–R obteve um desempenho maior do 20% quando comparado com os outros algoritmos.

Com relação ao aspecto do desempenho, a questão principal consiste em determinar por qual motivo o AEIQ–R tem um bom desempenho em problemas de otimização. Uma das possíveis razões é o fato de que a população quântica, à medida que o processo evolutivo avança, é capaz de descrever, cada vez melhor, as regiões mais promissoras do espaço de busca. Deve-se lembrar que as funções densidade de probabilidade que representam os genes quânticos são reposicionadas durante o

processo evolutivo na direção dos indivíduos mais promissores da população clásica. Além disso, ao terem suas larguras modificadas, essas funções especializam o gene quântico, fazendo com que o mesmo convirja, conforme o número de gerações tende a infinito, para um valor único no espaço de buscas. Este processo funciona de forma similar às técnicas de *hill-climbing*, só que de maneira mais sofisticada. Em conjunto com a população clássica, os indivíduos quânticos podem fornecer valores aproximados de avaliação para toda uma região do espaço de busca. Obviamente, este valor aproximado apresenta um erro muito grande quando os genes quânticos têm uma largura muito grande (por exemplo, no início do processo de otimização, quando as funções densidade de probabilidade cobrem todo o espaço de busca) e um erro cada vez menor à medida que a largura dos genes diminui. Este erro tende a zero à medida que o número de gerações tende a infinito. Em outras palavras, a população quântica do AEIQ–R é capaz de descrever *schemata* diretamente, ao contrário dos algoritmos genéticos convencionais, onde os indivíduos representam, unicamente, pontos no espaço de busca. Os *schemata* representam todo um conjunto de indivíduos e, deste modo, a avaliação deste conjunto de indivíduos, observados a partir destes *schemata*, pode fornecer, não somente a avaliação exata de um ponto no espaço de busca, como também uma aproximação da avaliação em uma determinada região deste mesmo espaço. Esta habilidade de avaliar regiões do espaço de busca, ao invés de avaliar apenas pontos, pode permitir que o algoritmo identifique mais rapidamente as regiões promissoras deste espaço, acelerando o tempo de convergência.

## Parametrização

Um outro aspecto importante a ser discutido diz respeito aos parâmetros do AEIQ–R e o impacto dos mesmos no processo evolutivo. Não existe um valor ou uma fórmula "mágica" para se definir esses valores, sendo a experimentação o processo mais adequado para se identificar os valores ideais. No entanto, algumas observações importantes podem ser feitas a respeito dos vários parâmetros que definem o AEIQ–R.

Em primeiro lugar, deve-se ressaltar a importância do parâmetro relacionado à recombinação dos indivíduos clássicos e à taxa de atualização dos genes quânticos. Em geral, nos testes preliminares, realizados para se determinar a melhor configuração a ser usada nas comparações, a parametrização mais bem-sucedida se observou quando as duas taxas eram mantidas iguais. Isto, no entanto, não é um argumento para descartar a possibilidade de se usar valores diferentes para estes parâmetros (nos testes com evolução diferencial e enxame de partículas foram usados valores diferentes para estas taxas). De qualquer modo, é importante destacar que esta parametrização está intimamente relacionada ao grau de epistasia (ou o grau de

separabilidade das variáveis) da função que se quer otimizar. Funções cujas variáveis são separáveis (uma função é separável se, por exemplo, puder ser escrita como  $f(x, y) = g(x) + h(y)$  ou  $f(x, y) = g(x)h(y)$  ou, em outras palavras, se a correlação entre as variáveis for baixa) podem ser otimizadas com taxas de recombinação baixas. Deste modo, poucas variáveis são mudadas em cada recombinação. Por outro lado, funções cujas variáveis não são separáveis (correlação alta entre as variáveis) são melhor otimizadas quando se usa taxas de recombinação elevadas. Isto faz com que diversas variáveis sejam alteradas simultaneamente. A figura 4.3 mostra as curvas de desempenho para uma mesma função ( $f_{griewank}$ ) com variáveis não separáveis quando se usa uma taxa de recombinação para a população clássica e uma taxa de atualização  $\zeta$  para a população quântica iguais a 3.33% e quando se usa taxas iguais a 66.66%. Já a figura 4.4 mostra a curva de desempenho para uma função ( $f_{sphere}$ ) com variáveis separáveis usando as mesmas taxas com valores iguais a 3.33% e 66.66%.

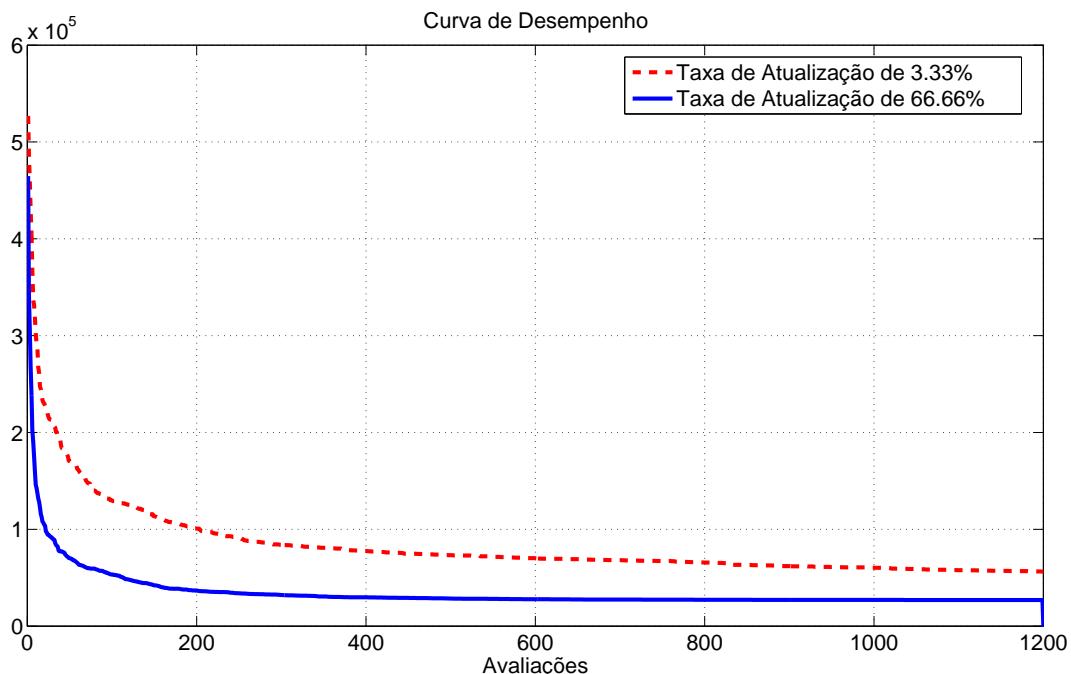


Figura 4.3: Gráfico de desempenho para uma função não separável usando taxas de atualização diferentes.

### Tamanho das Populações

Com relação ao número de indivíduos que formam a população quântica, deve-se levar em consideração que um número muito pequeno pode levar o algoritmo a uma convergência prematura. Isto acontece devido ao fato de que os poucos pulsos que formam a população acabam convergindo rapidamente para uma região de mínimo local, ficando presos nesta região indefinidamente. Por outro lado, um

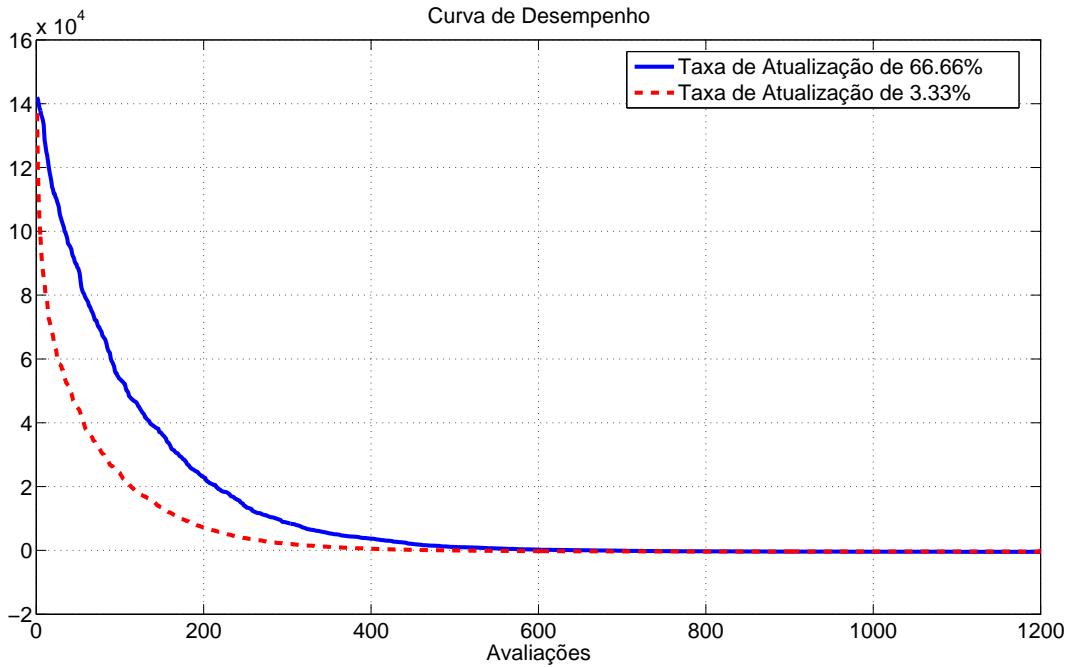


Figura 4.4: Gráfico de desempenho para uma função separável usando taxas de atualização diferentes.

número grande de indivíduos quânticos pode fazer com que a convergência do processo de otimização seja mais demorada do que o necessário, já que as funções densidade de probabilidade vão cobrir grandes regiões do espaço de busca. Um bom valor inicial para esta parametrização está em torno de 5 indivíduos quânticos com uma função densidade de probabilidade por gene.

Nos testes realizados para determinação dos melhores parâmetros, observou-se que o tamanho da população clássica (população observada) não é crítico para o algoritmo. Poucos indivíduos por geração são, em geral, suficientes para que o algoritmo convirja. Muitos indivíduos observados não geram uma grande melhoria na evolução e aumentam, desnecessariamente, o número de avaliações necessárias. Deste modo, sugere-se que sejam usados poucos indivíduos nesta população.

Com relação ao tamanho  $\gamma$  do *gap*, sugere-se usar valores no intervalo  $[N/2, N - 1]$ , onde  $N$  é o total de indivíduos clássicos da população. Aqui também sugere-se iniciar a parametrização com valores pequenos para o *gap* de modo a não aumentar o esforço computacional desnecessariamente.

O valor que indica quantas gerações devem se passar até que a taxa de melhoria do algoritmo (explicada na regra do 1/5 no capítulo 3) seja verificada é muito importante para o bom desempenho da otimização. Este valor, no entanto, também deve ser encontrado através de um processo de experimentação. Cada função que se deseja otimizar terá um comportamento diferente com relação a este parâmetro. Valores pequenos para este parâmetro farão com que a largura dos pulsos seja modificada muito rapidamente. Em alguns casos, isto pode fazer com

que o tempo que o algoritmo tem para explorar o entorno de um determinado ponto não seja suficiente para o mesmo encontrar o percentual adequado de mutações desejadas, acelerando, eventualmente, a convergência para um mínimo local. Por outro lado, valores grandes para este parâmetro farão com que o algoritmo perca muito tempo explorando regiões do espaço de busca com o mesmo tamanho.

Finalmente, com relação ao percentual com que a largura do pulso deve variar, o valor usado em todos os experimentos neste trabalho é igual a 0.9 (ou seja, cada vez que a largura dos pulsos tem que ser atualizada, eles irão aumentar ou diminuir 10%). Este é um valor empírico determinado experimentalmente mas, nos testes realizados para determinação dos melhores parâmetros para o algoritmo, observou-se que valores elevados (acima de 0.85) produzem os melhores resultados.

## 4.2

### Neuro-Evolução

Nesta seção são descritos os resultados obtidos em problemas de otimização de pesos de uma rede neural recorrente (neuro-evolução). Foram realizados testes de aprendizado supervisionado, usando-se um problema de previsão de séries temporais, e testes de aprendizado por reforço, usando-se um conjunto de problemas *benchmark* de controle.

#### 4.2.1

##### Aprendizado Supervisionado

Para testar o AEIQ–IR em problemas de previsão de séries, usou-se uma base de dados de vazão média semanal na bacia hidrográfica do Paraná. Esta base contém 8 atributos com informações sobre a vazão média nos 3 dias anteriores à previsão (3 atributos), a vazão atual (1 atributo), a previsão acumulada de chuvas para os próximos 7 dias (1 atributo) e as medições realizadas em um posto fluviométrico nos 3 dias anteriores (3 atributos). Os parâmetros usados para a otimização da rede neural são mostrados na tabela 4.8. A rede inicial utilizada no processo evolutivo com o AEIQ–IR tem, além das 8 entradas descritas anteriormente, 16 processadores e uma saída (associada ao processador 1). Os processadores usam a função sigmóide logística como função de ativação.

O número de genes igual a 400 corresponde ao número necessário para representar uma rede neural com 128 pesos das 8 entradas para um conjunto de 16 neurônios ( $8 \times 16 = 128$ ), 256 pesos das realimentações ( $16 \times 16 = 256$ ) e 16 *biases* dos neurônios. O intervalo  $\eta$  é o número de gerações que devem se passar para verificar se o operador *lesion* (definido no capítulo 3, e responsável por eliminar neurônios quando o erro não diminuir consideravelmente, ao se tentar retirar este neurônio) deve ser usado. Neste caso, a cada 10 gerações o algoritmo verifica se

Número Inicial de Genes	400
Tipo de Rede	Recorrente
Número de Entradas da Rede	8
Número Inicial de Processadores	16
Função de Ativação	logsig
Número de Indivíduos na População Quântica	5
Número de Indivíduos na População Clássica	10
$Gap \gamma$	5
Taxa de recombinação	0.5
Taxa de Atualização dos Genes Quânticos $\zeta$	0.5
Número de Gerações	250
Total de Experimentos	30
Intervalo de Atualização da População Quântica	5 gerações
Número de Gerações $\eta$ para Verificação do Número de Neurônios na Camada Escondida	10 gerações
Percentual Máximo de Queda de Desempenho ao Remover Neurônio	10%

Tabela 4.8: Parâmetros do AEIQ–R para aprendizado supervisionado.

deve retirar ou acrescentar um neurônio. Um neurônio é retirado quando o erro não se degradar mais do que 10% quando um neurônio for retirado.

Os resultados obtidos foram comparados com os resultados de (ICA05). Os resultados apresentados em (ICA05) foram gerados usando-se uma rede neural *multilayer perceptron* com uma camada escondida. Foram avaliadas 21 configurações diferentes para esta rede (variando o número de neurônios na camada escondida de 1 a 21) com o objetivo de determinar a melhor configuração para a rede neural. Para cada uma destas configurações, foram executadas 2500 épocas de treinamento (totalizando 52500 épocas) e a verificação do erro para validação foi feita a cada 5 épocas. Foram usados 4 anos de dados para treinamento, 1 ano para validação e 1 ano para teste. O melhor resultado obtido em (ICA05) é mostrado na tabela 4.9. Deve-se observar que os pesos finais, usados para o cálculo do erro com os dados de teste, são os pesos para os quais o processo de treinamento encontrou o menor erro de validação. Deste modo, os pesos utilizados não são, necessariamente, os pesos encontrados na última época do processo de treinamento.

Melhor Topologia	16 neurônios
Erro de Treinamento	12.62%
Erro de Teste	8.66%
Número Total de Épocas	52500

Tabela 4.9: Resultados do Treinamento da Rede Neural usando *Backpropagation*.

Os resultados com o treinamento realizado pelo AEIQ–R são mostrados na tabela 4.10.

Melhor Topologia	10 neurônios
Erro de Treinamento	9.26%
Erro de Teste	8.62%
Número Total de Épocas	5000

Tabela 4.10: Resultados do Treinamento da Rede Neural usando o AEIQ–R .

É importante ressaltar que, mesmo sem usar um conjunto de validação durante o treinamento do AEIQ–R, os dados referentes ao período usado como validação no *backpropagation* não foram utilizados. Como se pode verificar pelos resultados, a melhor topologia encontrada pelo algoritmo usa 6 neurônios a menos do que a encontrada usando *backpropagation*. Os resultados obtidos em termos de erro percentual são semelhantes mas, com relação ao número de épocas, o resultado apresentado pelo AEIQ–R é muito superior. O número total de épocas usado pelo AEIQ–R é calculado considerando-se que cada avaliação feita pelo algoritmo é uma época.

Além do resultado em termos de número de épocas, deve-se levar em consideração o resultado em termos do tempo computacional para a execução do algoritmo. Em testes realizados em um computador Pentium IV 3.0GHz, o tempo médio para o cálculo de 10000 épocas para o AEIQ–R é da ordem de 160 segundos. Para o *backpropagation*, o tempo médio é da ordem de 1090 segundos, ou seja, aproximadamente 6 vezes maior do que o AEIQ–R.

#### 4.2.2

#### Problemas de Aprendizado por Reforço

##### Carro na Montanha

O experimento do carro na montanha (Moore91) pode ser descrito da seguinte forma (Figueiredo03): um carro deve subir até o topo de uma montanha; entretanto, o carro não possui a potência necessária para vencer a força da gravidade. Dessa forma, para alcançar o seu objetivo, o carro precisa inicialmente se mover no sentido contrário ao alvo para poder adicionar a aceleração da gravidade à sua própria aceleração.

O problema possui duas variáveis de estado contínuas: a posição do carro  $x_t \in [-1.2, 0.5]$  e sua velocidade  $v_t \in [-0.07, 0.07]$ . Além disso, o motor do carro pode aplicar sobre o mesmo 3 ações discretas: uma impulsão para a esquerda ( $F = -1$ ), para a direita ( $F = 1$ ) e ficar parado ( $F = 0$ ). A dinâmica do sistema é descrita pelas equações 4-3 e 4-4.

$$v_{t+1} = \min(0.07, \max(-0.7, v_t + 0.001F_t - 0.0025 \cos(3x_t))) \quad (4-3)$$

$$x_{t+1} = \min(0.5, \max(-1.2, x_t + v_{t+1})) \quad (4-4)$$

A montanha é definida pela equação 4-5 e a figura 4.5 representa a mesma graficamente.

$$f(x) = \sin(3x) \quad (4-5)$$

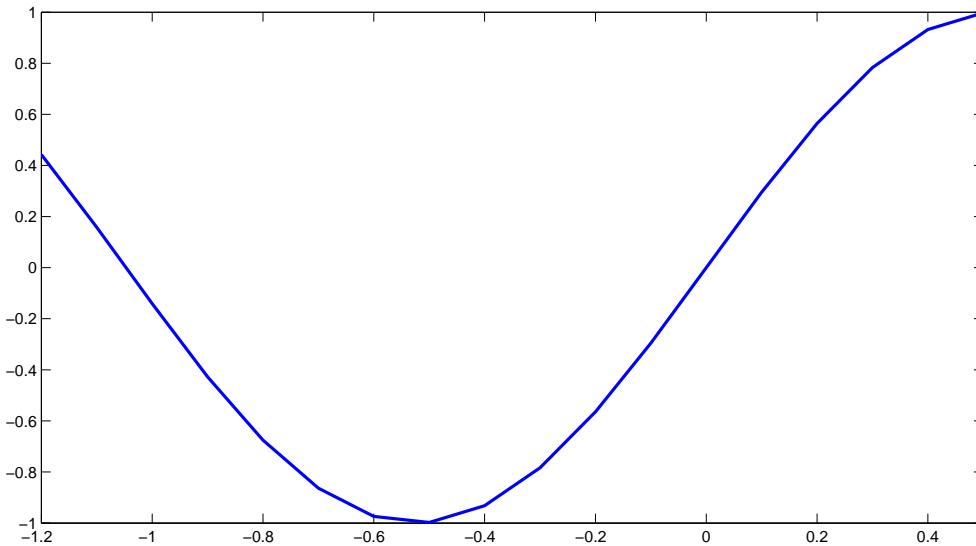


Figura 4.5: Representação gráfica do problema do carro na montanha.

Neste problema, a rede neural tem 2 entradas (posição e velocidade do carro). A configuração inicial utilizada tem 5 neurônios na camada escondida, uma saída associada ao primeiro processador e usa a função sigmóide como função de ativação dos processadores. Para este problema, os parâmetros listados na tabela 4.11 foram usados para o AEIQ–R.

A função de avaliação consiste na soma do número de passos necessários para que o carro atinja o topo da montanha à partir de duas posições iniciais diferentes ( $x_0 = -1.2$  e  $x_0 = -0.5$ ), com velocidade inicial igual à zero. Após a fase de aprendizado, foram gerados 1000 casos com posições e velocidades iniciais aleatórias e o valor médio do número de passos necessários para alcançar o topo da montanha foi calculado. Para fins de medida de desempenho, este resultado foi comparado com os modelos Neuro-Fuzzy Hierárquicos (RL-NFHB e RL-NFHP), Neural Q-Learning, CMAC Q-Learning e FQL, mostrados em (Figueiredo03) na tabela 4.12. É importante destacar que, nos modelos usados para comparação, o treinamento foi feito com posições e velocidades iniciais aleatórias e o conjunto de ações possíveis na saída é diferente ( $F = -10, -5, -1, 0, 1, 5, 10$ ).

Estes resultados mostram uma pequena melhoria de desempenho em relação ao número de passos por parte do AEIQ–R, mesmo aprendendo a partir de apenas dois casos iniciais diferentes e tendo apenas 3 ações possíveis como resultado.

Número Inicial de Genes	40
Tipo de Rede	Recorrente
Número de Entradas da Rede	2
Número Inicial de Processadores	5
Função de Ativação	logsig
Número de Indivíduos na População Quântica	3
Número de Indivíduos na População Clássica	6
$Gap \gamma$	3
Taxa de recombinação	0.5
Taxa de Atualização dos Genes Quânticos $\zeta$	0.5
Número de Gerações	200
Total de Experimentos	5
Intervalo de Atualização da População Quântica	5 gerações
Número de Gerações $\eta$ para Verificação do Número de Neurônios na Camada Escondida	2 gerações
Percentual Máximo de Queda de Desempenho ao Remover Neurônio	10%

Tabela 4.11: Parâmetros do AEIQ–R usados para o problema do carro na montanha.

Método	Número Médio de Passos na Fase de Teste
Neural Q-Learning	2189
CMAC Q-Learning	85
FQL	61
RL-NFHB	71
RL-NFHP	69
AEIQ–R	62

Tabela 4.12: Resultados para o problema do carro na montanha.

Deve-se ressaltar, no entanto, que o AEIQ–R e os modelos neuro-fuzzy usam conceitos diferentes de aprendizado por reforço.

O gráfico da figura 4.6 mostra a variação da velocidade e da posição do carro para a condição inicial  $x_0 = -0.5$  e  $v_0 = 0$  (pior caso).

### Pêndulo Invertido

O problema do pêndulo invertido é um *benchmark* clássico para sistemas de controle. O problema consiste em um carro motorizado com um pêndulo apoiado sobre a sua parte superior. Este pêndulo gira em torno de um eixo fixo no carro e o veículo desliza sobre um trilho em cima de uma superfície plana. Em geral, o problema é formulado de modo que o motor do carro esteja sempre acelerado em algum sentido. Somado ao fato do pêndulo estar invertido, isto faz com que o problema seja intrinsecamente instável. A figura 4.7 mostra um exemplo de um carro com um pêndulo invertido.

A formulação matemática do problema (Koza92) é dada pelas equações 4-6,

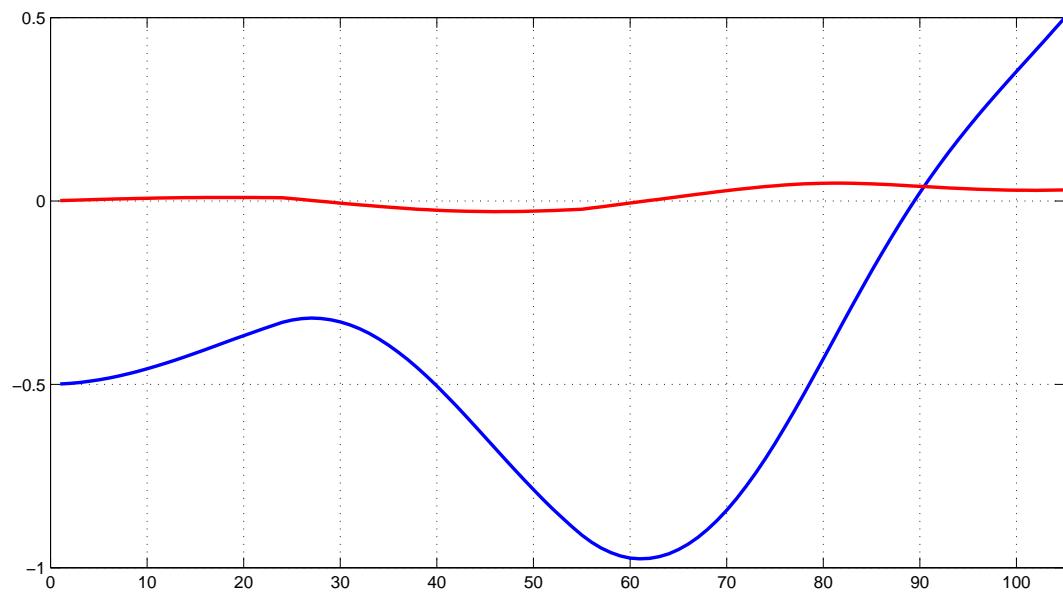


Figura 4.6: Variação da velocidade (linha vermelha) e da posição (linha azul) no problema do carro na montanha.

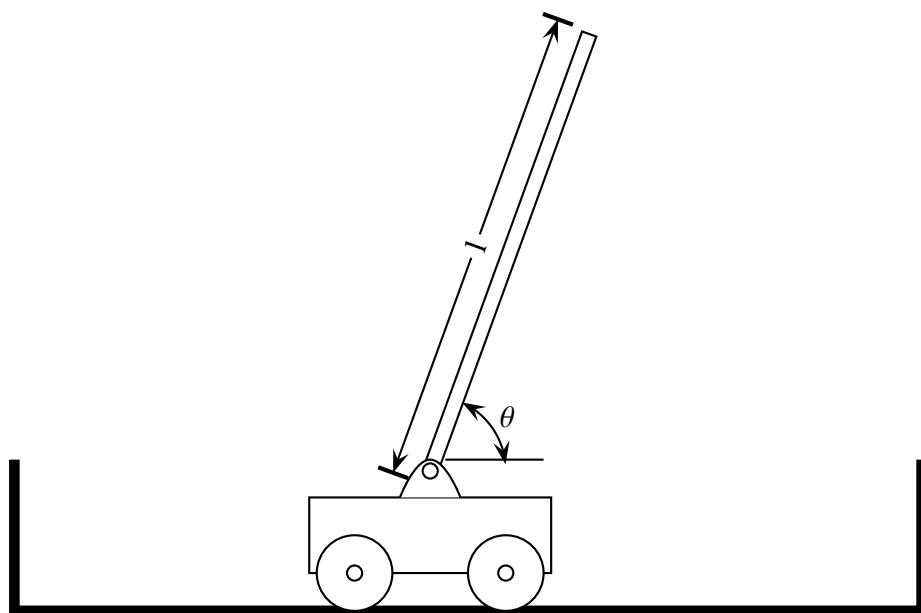


Figura 4.7: Exemplo do problema do pêndulo invertido.

4-7, 4-8 e 4-9.

$$\ddot{x} = \frac{F - \mu_c sgn(\dot{x}) + \sum_{i=1}^N \tilde{F}_i}{M + \sum_{i=1}^N \tilde{m}_i} \quad (4-6)$$

$$\ddot{\theta}_i = -\frac{3}{4l_i} \left( \ddot{x} \cos \theta_i + g \sin \theta_i + \frac{\mu_{pi}\dot{\theta}_i}{m_il_i} \right) \quad (4-7)$$

$$\tilde{F}_i = m_il_i\dot{\theta}_i^2 \sin \theta_i + \frac{3}{4}m_i \cos \theta_i \left( \frac{\mu_{pi}\dot{\theta}_i}{m_il_i} + g \sin \theta_i \right) \quad (4-8)$$

$$\tilde{m}_i = m_i \left( 1 - \frac{3}{4} \cos^2 \theta_i \right) \quad (4-9)$$

Onde  $x$  é a posição do carro no trilho,  $\theta_i$  é o ângulo do pêndulo  $i$  (para problemas onde se usa mais de um pêndulo) com o eixo vertical,  $F$  é a força aplicada ao carro,  $l_i$  é a metade do comprimento do pêndulo,  $g = 10m/s^2$ ,  $M$  é a massa do carro,  $m_i$  é a massa do  $i$ -ésimo pêndulo sobre o carro,  $\mu_c$  é o coeficiente de atrito do carro com o trilho e  $\mu_{pi}$  é o coeficiente do  $i$ -ésimo pêndulo com o eixo no qual ele está preso.

Os valores usados para o problema são os mesmos utilizados em (Gomez03) e são mostrados na tabela 4.13.

Parâmetro	Valor
$x$	$[-2.4, 2.4]m$
$\theta$	$[-0.209, 0.209]rad$
$F$	$[-10, 10]$
$l$	$0.5m$
$M$	$1kg$
$m$	$0.1kg$

Tabela 4.13: Parâmetros usados para o problema do pêndulo invertido.

A parametrização do AEIQ–R é mostrada na tabela 4.14.

O valor de 26 para o número inicial de genes é equivalente a uma rede neural com 5 neurônios na camada escondida. A configuração inicial desta rede, portanto, tem 4 entradas (posição, velocidade, ângulo do pêndulo com o eixo vertical e velocidade angular do pêndulo), 5 neurônios na camada escondida e uma saída.

O tempo é discretizado em unidades de 0.01 segundos. Todos os valores usados como entrada na rede neural são normalizados entre -1 e 1. A saída da rede é mapeada do domínio  $[-1, 1]$  para o domínio  $[-10, 10]$ . A tarefa de controlar o pêndulo é considerada bem-sucedida quando o controle é capaz de manter o pêndulo equilibrado por mais de 100000 unidades de tempo (1000 segundos). A função de avaliação consiste no número de unidades de tempo em que o carro consegue manter o pêndulo equilibrado (e, portanto, este é um problema onde se quer maximizar a função de avaliação). Quando o ângulo do pêndulo é maior do que 0.209 radianos ou a posição do carro for maior do que os limites definidos para  $x$  o algoritmo

Número Inicial de Genes	50
Tipo de Rede	Recorrente
Número de Entradas da Rede	4
Número Inicial de Processadores	5
Função de Ativação	logsig
Número de Indivíduos na População Quântica	5
Número de Indivíduos na População Clássica	10
$Gap \gamma$	5
Taxa de recombinação	0.5
Taxa de Atualização dos Genes Quânticos $\zeta$	0.5
Número de Gerações	250
Total de Experimentos	5
Intervalo de Atualização da População Quântica	3 gerações
Número de Gerações $\eta$ para Verificação do Número de Neurônios na Camada Escondida	10 gerações
Percentual Máximo de Queda de Desempenho ao Remover Neurônio	10%

Tabela 4.14: Parâmetros do AEIQ–R usados para o problema do pêndulo invertido.

considera que o controlador falhou em manter o pêndulo equilibrado. Foram feitos 50 experimentos selecionando-se as condições iniciais para o problema de forma aleatória dentro das faixas de valores definidas na tabela 4.13. Esta configuração é a mesma usada em (Gomez03) e, a partir dela, é possível comparar os resultados com diversos outros métodos. Os métodos usados em (Gomez03) e que são usados aqui para fins de comparação são:

- *Q-Learning* com MLP (Q-MLP);
- Sarsa( $\lambda$ ) com Aproximador de Função baseado em Casos (SARSA-CABA);
- Sarsa( $\lambda$ ) com Aproximador de Função CMAC (SARSA-CMAC);
- Neuro–Evolução Simbiótica e Adaptativa (SANE);
- Neuro–Evolução Convencional (CNE);
- Neuro–Evolução de Topologias Crescentes (NEAT);
- Subpopulações Evolutivas (ESP).

Os resultados comparativos são mostrados na tabela 4.15.

Os resultados preliminares mostram a maior eficiência do AEIQ–R com relação aos outros algoritmos. O algoritmo encontrou o melhor indivíduo com 4 neurônios na camada escondida. A figura 4.8 mostra o uso da melhor rede neural em um teste de generalização (os resultados usados para comparação (Gomez03) não fazem considerações sobre a capacidade de generalização da rede). Este gráfico mostra a variação do ângulo do pêndulo para uma posição inicial aleatória, diferente das posições usadas para o treinamento, para as 10000 primeiras unidades de tempo.

Método	Número Médio de Avaliações
Q-MLP	2056
SARSA-CMAC	540
SARSA-CABA	965
CNE	352
SANE	302
NEAT	743
ESP	289
AEIQ-R	210

Tabela 4.15: Parâmetros usados para o problema do pêndulo invertido.

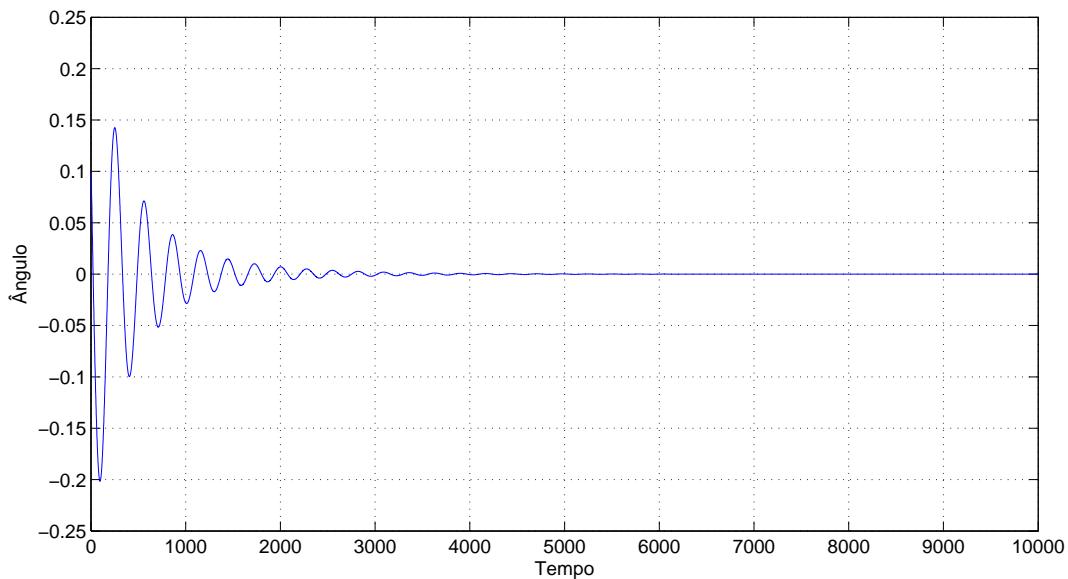


Figura 4.8: Ângulo do pêndulo com relação ao tempo.

A figura 4.9 mostra o mesmo gráfico mas, além do ângulo, também é mostrado a velocidade horizontal do carro.

Todos os resultados mostrados nos problemas de benchmark sugerem que o AEIQ-R é um algoritmo com ótimo potencial de uso em vários tipos de aplicações, oferecendo melhor desempenho em termos do número de avaliações e, em vários casos, resultados superiores em termos do melhor valor encontrado. No entanto, ainda existe a necessidade de se realizar diversos experimentos que comprovem o bom desempenho do algoritmo nos mais diversos tipos de aplicação.

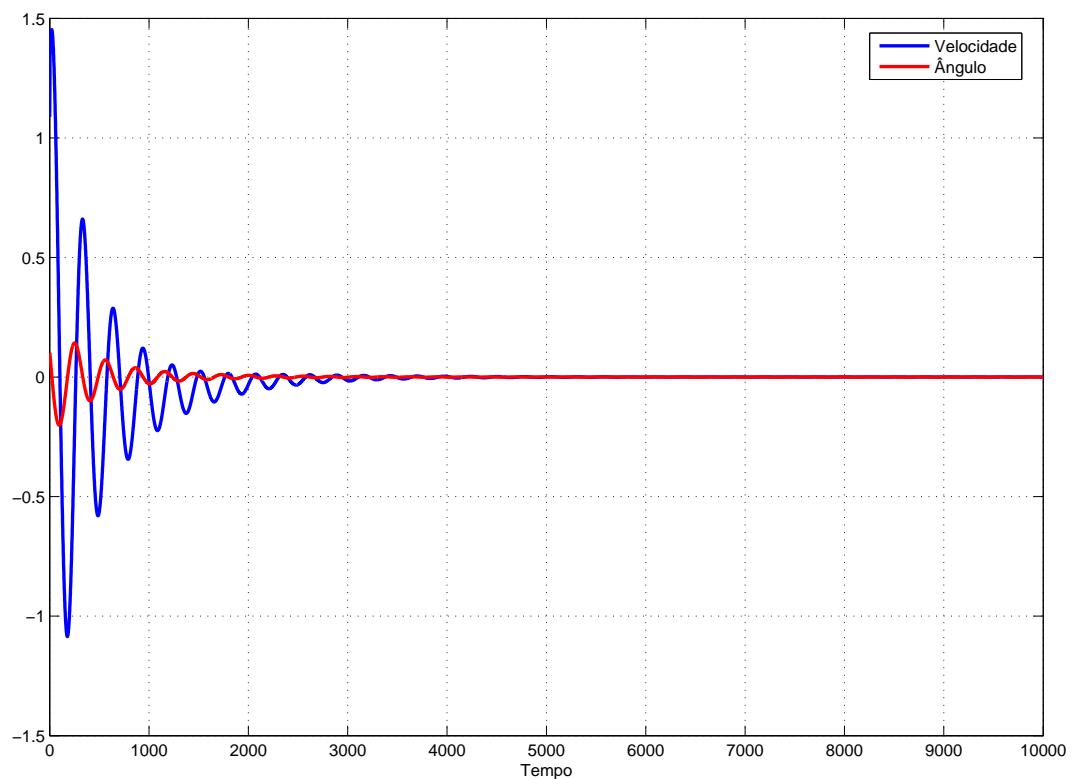


Figura 4.9: Ângulo do pêndulo e velocidade do carro com relação ao tempo.

## 5

### Conclusões e Trabalhos Futuros

Este trabalho apresentou a proposta de um novo modelo de otimização, inspirado nos conceitos da física quântica de superposição de estados e funções de onda, capaz de oferecer: uma maneira de representar variáveis numéricas de forma direta; melhor desempenho na otimização; escalabilidade; capacidade de armazenamento de conhecimento sobre regiões do espaço de busca; e capacidade de aprendizado compartilhado. Foi apresentado um algoritmo evolutivo com inspiração em física quântica que usa uma representação para números reais (AEIQ- $\mathbb{R}$ ), em contrapartida ao algoritmo apresentado em (Han04) que usa uma representação binária para um algoritmo evolutivo com inspiração quântica.

Também foi apresentado um modelo neuro-evolutivo, utilizando o AEIQ- $\mathbb{R}$  para o treinamento dos pesos de uma rede neural recorrente. Este modelo permite, não só o treinamento da rede, como também a otimização da topologia da mesma, através da variação do número de processadores e da distribuição destes processadores em camadas, fazendo-se a manipulação dos pesos da rede neural.

Com relação à capacidade de representar variáveis numéricas de forma direta mostrou-se que, através do uso de funções de onda, foi possível construir uma analogia contínua ao modelo quântico de estados discretos. Também foi apresentado um modelo físico hipotético capaz de simular o funcionamento do AEIQ- $\mathbb{R}$ .

Com relação a maior velocidade de convergência foram apresentados, no capítulo 4, diversos estudos de caso de otimização numérica. Estes estudos de caso mostraram a importância de se usar uma representação específica para números reais ao se tratar problemas de otimização numérica. A comparação de resultados com o AEIQ- $\mathbb{R}$  na otimização deste tipo de problemas mostrou a melhor capacidade deste algoritmo no tratamento desta classe de problemas. Além disso, quando comparado a outros algoritmos de otimização, o AEIQ- $\mathbb{R}$  mostrou ser consistente na otimização de problemas e mostrou ser capaz de produzir bons resultados em diversos tipos de funções, com diferentes características.

Para demonstrar a escalabilidade do algoritmo foram mostrados diversos resultados que mostram a capacidade do algoritmo AEIQ- $\mathbb{R}$  em otimizar problemas com até 1000 variáveis sem, no entanto, ter um crescimento exponencial no esforço computacional necessário para se atingir um determinado resultado. Apesar de não

serem testes conclusivos, estes sugerem a necessidade de uma investigação mais detalhada já que, um algoritmo com complexidade não-exponencial é interessante para a otimização de problemas com grande dimensionalidade.

Para explicar o desempenho do algoritmo foram apresentadas diversas hipóteses, em particular, a hipótese de que a população quântica se comporta como uma representação direta dos melhores *schemata* que representam os melhores indivíduos da população. Uma outra hipótese apresentada sugere que o armazenamento de conhecimento normativo, sob a forma de funções densidade de probabilidade ou funções de onda pode, indiretamente, determinar valores aproximados de avaliação para diferentes regiões do espaço de busca e, assim, ser responsável pelo melhor desempenho do algoritmo. Além disso, este tipo de conhecimento pode, eventualmente, ser compartilhado em um sistema de aprendizado multi-agentes e uma investigação mais detalhada precisa ser realizada nesta área.

Como trabalhos futuros é possível destacar alguns pontos importantes, que podem trazer novas possibilidades de aplicação para o algoritmo:

- Implementação de outros operadores de algoritmos genéticos clássicos com o objetivo de tentar melhorar a capacidade de otimização do AEIQ–R ;
- Testes de desempenho usando-se outros tipos de função densidade de probabilidade, diferentes de um pulso quadrado (por exemplo, distribuições normais);
- Testes do modelo neuro-evolutivo para classificação de padrões;
- Novos testes em previsão de séries usando neuro–evolução;
- Novos experimentos em problemas de controle, principalmente com relação à capacidade de generalização da rede neural encontrada pelo algoritmo;
- Investigação mais detalhada da complexidade computacional do algoritmo proposto;
- Investigação mais detalhada da capacidade de se usar a população quântica como uma forma de conhecimento compartilhado.

Além desses testes, também se inclui nos próximos passos a inserção de novas formas de conhecimento similares aos usados em algoritmos culturais (como por exemplo o conhecimento topográfico) e a utilização do AEIQ–R para a resolução de outros tipos de problemas, descritos nas seções a seguir.

## 5.1

### Aprendizado Online

Sistemas de aprendizado online são aqueles onde a tarefa que se deseja aprender ou otimizar é não-estacionária, ou seja, a configuração do problema varia com o tempo (Saad98). A idéia por trás do uso de algoritmos evolutivos para aprendizado *online* é usar as melhores soluções, encontradas até o momento anterior a esta mudança de configuração do problema, para alimentar a população inicial que será usada durante o aprendizado da nova configuração do problema. A expectativa é que as pequenas mudanças provocadas na configuração do problema provoquem também pequenas mudanças nas soluções consideradas ótimas. O uso do AEIQ–R para aprendizado *online* tem, deste modo, um potencial promissor: os pulsos que formam o indivíduo quântico representam, ao invés de apenas pontos no espaço de busca, toda uma região promissora do espaço de buscas. Ao usar estas regiões como realimentação para o AEIQ–R, espera-se conseguir um aprendizado *online* mais eficiente.

Para se experimentar este modelo, propõe-se utilizar um simulador de dinâmica de vôos chamado JSBSim (Jsbsim). O objetivo é fazer com que o sistema seja capaz de controlar em tempo real um foguete sem estabilizadores verticais, como o mostrado na figura 5.1.

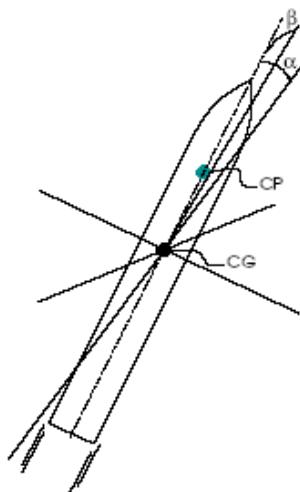


Figura 5.1: Diagrama de um foguete sem estabilizadores.

Nesta figura, CG é o centro de gravidade do foguete, CP é o centro de pressão e  $\alpha$  e  $\beta$  representam os ângulos de inclinação do foguete com o eixo horizontal.

Neste tipo de foguete existem apenas duas forças atuando sobre o mesmo: o empuxo gerado pelos propulsores do foguete e o arrasto exercido pela atmosfera sobre o centro de pressão do foguete. O controle deste tipo de foguete oferece um grande desafio, devido ao fato da ausência dos estabilizadores verticais fazer com que o centro de pressão do foguete fique localizado acima do seu centro

de gravidade, tornando o sistema naturalmente instável, de maneira similar ao problema do pêndulo invertido. No entanto, ao contrário do pêndulo invertido, este problema oferece um grau de complexidade muito maior pois, à medida que o foguete se movimenta, diversas variáveis relacionadas à configuração do ambiente se alteram (como, por exemplo, a densidade do ar, o peso do foguete, etc), exigindo que o sistema se adapte rapidamente à nova configuração do ambiente.

## 5.2 Sistemas Multi-Agentes

De acordo com (Franklin96), um agente autônomo é um sistema que está situado e é parte de um ambiente, e que é capaz de perceber este ambiente e agir sobre ele no decorrer do tempo em busca de um determinado objetivo. O mesmo artigo cita quatro pontos chaves que diferenciam um agente de um programa:

- Reação ao ambiente: o agente é capaz de reagir a mudanças no ambiente;
- Autonomia: exerce controle sobre suas próprias ações;
- Orientado a um objetivo: suas ações não são simples respostas ao ambiente;
- Temporalmente contínuo: um agente é um processo em contínua execução.

Quando diversos agentes interagem de forma colaborativa ou competitiva, os mesmos podem formar um sistema multi-agente. Em geral, os agentes que formam este tipo de sistema não têm disponíveis todos os dados ou todos os métodos necessários para se alcançar um objetivo e, deste modo, precisam colaborar entre si para alcançá-lo. Além disso, existe pouco ou nenhum controle central nestes sistemas responsável por gerenciar os agentes (nos casos de interações competitivas, não existe nenhum tipo de controle central entre os agentes que estão competindo).

O aprendizado de sistemas multi-agentes constitui um importante aspecto desta área de pesquisa. O uso do AEIQ-R para este aprendizado pode ser feito baseado no modelo mostrado na figura 5.2.

A idéia é que o indivíduo quântico funcione como um espaço que armazene conhecimento normativo sobre o aprendizado e que este conhecimento seja compartilhado entre os agentes. Além disso, a população clássica é dividida em subpopulações e cada uma dessas subpopulações evolui o conhecimento de cada um dos agentes. A expectativa é que o compartilhamento do conhecimento e a separação das subpopulações seja capaz de produzir a especialização dos agentes e a capacidade de cooperação dos mesmos na realização da tarefa.

Para a experimentação deste modelo, propõe-se utilizar um sistema chamado Robocode (Robocode). Este sistema consiste em uma arena onde um conjunto de robôs, similares a tanques de combate, simulam uma disputa. Cada vez que um dos

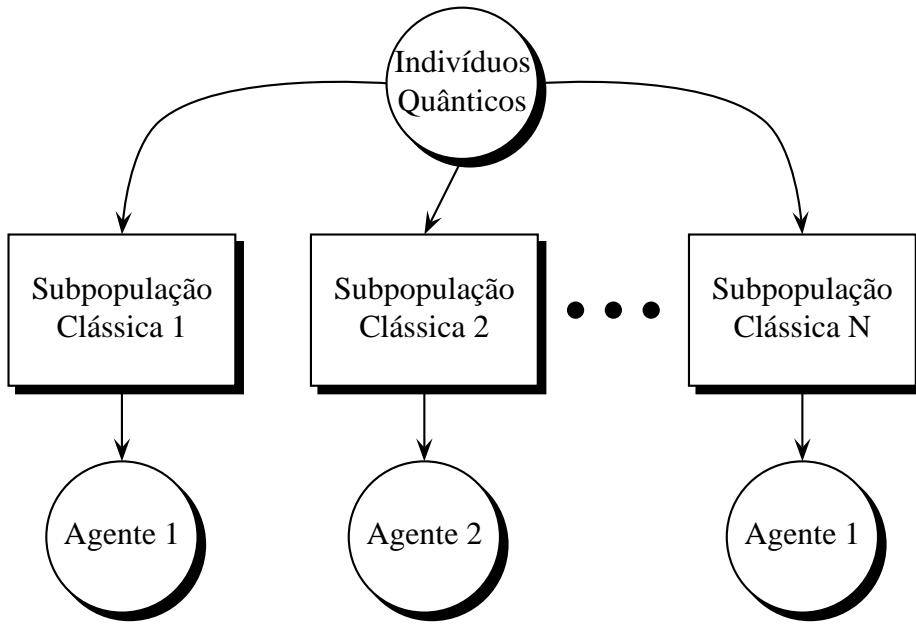


Figura 5.2: Modelo de aprendizado multi-agentes usando o AEIQ- $\mathbb{R}$ .

robôs é atingido por outro, o mesmo é eliminado da competição. O último robô a ficar na arena é o vencedor da disputa. O sistema permite a definição de times e os agentes que formam os times devem colaborar entre si para ganhar a disputa. Além disso, existem diversos algoritmos disponíveis, desenvolvidos para o controle dos robôs (inclusive algoritmos desenvolvidos com o uso de programação genética), que permitem a comparação dos resultados diretamente.

## **Referências Bibliográficas**

- [Back97] Back, T.; Fogel, D. B. ; Michalewicz, Z., editors. **Handbook of Evolutionary Computation**. Institute of Physics Publishing, 1997. 1.1
- [Bishop95] BISHOP, C. M.. **Neural Networks for Pattern Recognition**. Oxford University Press, 1995. 3.4
- [Blanco01] BLANCO, A.; DELGADO, M. ; PEGALAJAR, M. C.. **A real-coded genetic algorithm for training recurrent neural networks**. Neural Networks, 14:93–105, 2001. 1.4
- [Figueiredo03] FIGUEIREDO, K.. **Novos modelos neuro-fuzzy hierárquicos com aprendizado por reforço para agentes inteligentes**. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2003. 4.2.2, 4.2.2
- [Franklin96] FRANKLIN, S.; GRAESSER, A.. **Is it an agent, or just a program?: A taxonomy for autonomous agents**. In: INTELLIGENT AGENTS III. AGENT THEORIES, ARCHITECTURES AND LANGUAGES (ATAL'96), volumen 1193, Berlin, Germany, 1996. Springer-Verlag. 5.2
- [Gillespie73] GILLESPIE, D. T.. **A Quantum Mechanics Primer – An Elementary Introduction to the Formal Theory of Non-Relativistic Quantum Mechanics**. Open University Set Book, 1973. 2.1.2, 2.1.3, 2.1.3
- [Gomez97] GOMEZ, F.; MIKKULAINEN, R.. **Incremental evolution of complex general behavior**. In: ADAPTIVE BEHAVIOR, volumen 5, p. 317–342, 1997. 2.7
- [Gomez03] GOMEZ, F.. **Robust Non-Linear Control through Neuroevolution**. PhD thesis, The University of Texas at Austin, 2003. 2.7, 3.4, 3.4, 4.2.2, 4.2.2, 4.2.2
- [Green01] GREEN, N. J. B.. **Quantum Mechanics 1: Foundations**. Oxford Science Publications, 2001. 2.1.1, 2.1.2, 2.1.2, 2.1.2, 2.1.3, 2.2
- [Han00] HAN, K.; KIM, J.. **Genetic quantum algorithm and its application to combinatorial optimization problem**. In: PROCEEDINGS OF THE 2000

- CONGRESS ON EVOLUTIONARY COMPUTATION, volumen 2, p. 1354–1360, 2000. 2.5, 2.5, 2.5
- [Han02] HAN, K.; KIM, J.. **Quantum-inspired evolutionary algorithm for a class of combinatorial optimization.** In: IEEE TRANS. EVOL. COMPUT. 6, p. 580–593, 2002. 1.1, 2.5
- [Han04] HAN, K.-H.; KIM, J.-H.. **Quantum-inspired evolutionary algorithms with a new termination criterion,  $h_e$  gate and two-phase scheme.** IEEE Transactions on Evolutionary Computation, 8, 2004. 1.1, 4.1.2, 5
- [Haykin99] HAYKIN, S.. **Neural Networks – A Comprehensive Foundation.** Prentice Hall, 2 edition, 1999. 1.1, 3.4, 3.4, 4
- [ICA05] **Projeto de previsão de vazões**, 2005. 4.2.1
- [Ilonen03] ILONEN, J.; KAMARAINEN, J.-K. ; LAMPINEN, J.. **Differential evolution training algorithm for feed-forward neural networks.** Neural Processing Letters, 17:93–105, 2003. 1.1
- [Jang04] JANG, J.-S.; HAN, K.-H. ; KIM, J.-H.. **Face detection using quantum-inspired evolutionary algorithm.** In: PROCEEDINGS OF CONGRESS ON EVOLUTIONARY COMPUTATION, volumen 2, p. 2100–2106, 2004. 2.5
- [Jsbsim] <http://jsbsim.sourceforge.net/>. 5.1
- [Knuth73] KNUTH, D. E.. **Searching and Sorting.** Addison Wesley, 1973. 2.4
- [Koza92] Koza, J. R., editor. **Genetic Programming: on the programming of computers by means of natural selection.** MIT Press, 1992. 1.1, 4.2.2
- [Michalewicz94] MICHALEWICZ, Z.. **Genetic algorithms + data structures = evolution programs (2nd, extended ed.).** Springer-Verlag New York, Inc., New York, NY, USA, 1994. 1.1, 3.1.4, 4.1, 4.1.2
- [Moore91] MOORE, A. W.. **Variable resolution dynamic programming:efficiently learning action maps in multivariate real-valued state-spaces.** In: PROCEEDINGS OF THE EIGHTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 333–337, 1991. 4.2.2
- [Moore95] MOORE, M.; NARAYANAN, A.. **Quantum-inspired computing,** 1995. 2.4, 2.4, 2.4, 2.4
- [Moriarty97] MORIARTY, D. E.. **Symbiotic Evolution of Neural Networks in Sequential Decision Tasks.** PhD thesis, The University of Texas at Austin, 1997. 2.7

- [Narayanan96] NARAYANAN, A.; MOORE, M.. **Quantum inspired genetic algorithms.** In: INTERNATIONAL CONFERENCE ON EVOLUTIONARY COMPUTATION, p. 61–66, 1996. 1.1
- [Planck01] PLANCK, M.. **On the law of distribution of energy in the normal spectrum.** Annalen der Physik, 4:553, 1901. 2.1.1
- [Reynolds94] REYNOLDS, R. G.. **An introduction to cultural algorithms.** In: PROCEEDINGS OF THE 3RD ANNUAL CONFERENCE ON EVOLUTIONARY PROGRAMMING, p. 131–139, 1994. 1.1, 2.6
- [Reynolds04] REYNOLDS, R. G.; PENG, B.. **Cultural algorithms: Modeling of how cultures learn to solve problems.** In: PROCEEDINGS OF THE 18TH IEEE INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE (ICTAI 2004), 2004. 1.1, 2.6.1, 2.6.1
- [Rieffel00] RIEFFEL, E.; POLAK, W.. **An introduction to quantum computing for non-physicists**, 2000. 2.3
- [Robocode] <http://robocode.sourceforge.net/>. 5.2
- [Rychtyckyj03] RYCHTYCKYJ, N.; OSTROWSKI, D.; SCHLEIS, G. ; REYNOLDS, R. G.. **Using cultural algorithms in industry.** In: PROCEEDINGS OF IEEE SWARM INTELLIGENCE SYMPOSIUM, Indianapolis, Indiana, 2003. IEEE Press. 2.6.1
- [Saad98] Saad, D., editor. **On-line Learning in Neural Networks**. Cambridge University Press, 1998. 5.1
- [Skinner95] SKINNER, A.; BROUGHTON, J. Q.. **Neural networks in computational material science: Training algorithms.** Modelling and Simulation in Material and Engineering, 3:371–390, 1995. 2.7
- [Spector04] SPECTOR, L.. **Automatic Quantum Computer Programming – A Genetic Programming Approach.** Springer, 2004. 2.3
- [Stanley02] STANLEY, K. O.; MIIKKULAINEN, R.. **Evolving neural networks through augmenting topologies.** In: EVOLUTIONARY COMPUTATION, 2002. 2.7
- [Storn95] STORN, R.; PRICE, K.. **Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces.** Technical Report TR-95-012, 1995. 1.1, 3.1.3

- [Tasgetiren05] TASGETIREN, M. F.; GENCYLMAZ, G.; LIANG, Y.-C. ; EKER, I.. **A differential evolution algorithm for continuous function optimization.** In: CONGRESS ON EVOLUTIONARY COMPUTATION, 2005. 4.1.1
- [Tasgetiren05A] TASGETIREN, M. F.; GENCYLMAZ, G.; LIANG, Y.-C. ; EKER, I.. **Global optimization of continuous functions using particle swarm optimization.** In: CONGRESS ON EVOLUTIONARY COMPUTATION, 2005. 4.1.1
- [Yao99] YAO, X.; LIU, Y. ; LIN, G. M.. **Evolutionary programming made faster.** IEEE Trans. Evolutionary Computation, 3:82–102, 1999. 1.1, 4.1.2
- [Yao99a] YAO, X.. **Evolving artificial neural networks.** In: PROCEEDINGS OF THE IEEE, volumen 87, p. 1423–1447, 1999. 1.1

## Anexo 1

### Funções para Medida de Desempenho

#### Função Sphere Deslocada

$$f_1(\mathbf{x}) = \sum_{i=1}^D z_i^2 + bias_1$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

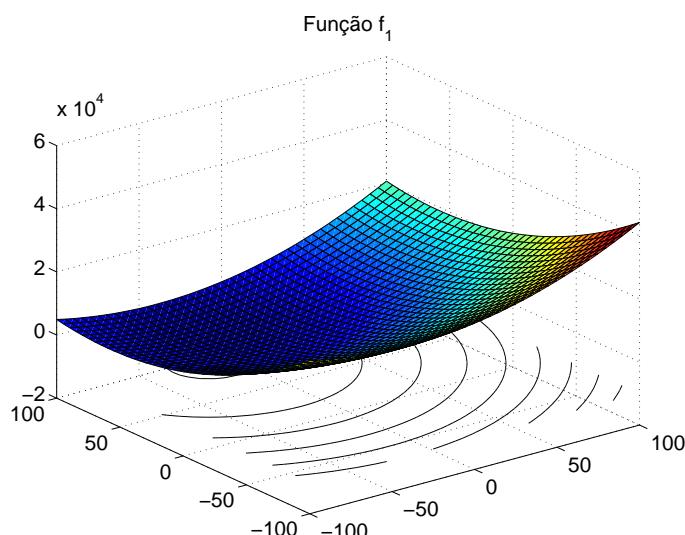


Figura 5.3: Mapa 3D da função  $f_1$ .

- Unimodal
- Deslocamento da origem
- Separável
- Escalonável
- $\mathbf{x} \in [-100, 100]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}, F_1(\mathbf{x}^*) = bias_1 = -450$

### Função Schwefel Deslocada

$$f_2(\mathbf{x}) = \sum_{i=1}^D \left( \sum_{j=1}^i z_j \right)^2 + bias_2$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

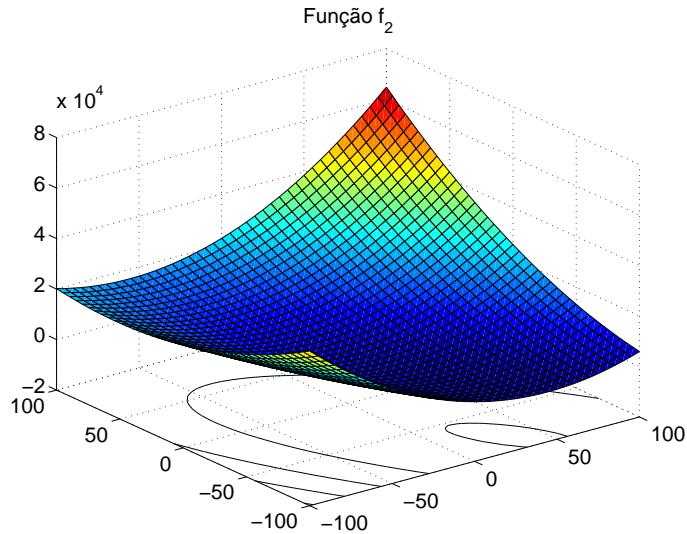


Figura 5.4: Mapa 3D da função  $f_2$ .

- Unimodal
- Deslocamento da origem
- Não-separável
- Escalonável
- $\mathbf{x} \in [-100, 100]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}, F_2(\mathbf{x}^*) = bias_2 = -450$

### Função Elíptica Rotacionada e Deslocada

$$f_3(\mathbf{x}) = \sum_{i=1}^D \left(10^6\right)^{\frac{i-1}{D-1}} z_i^2 + bias_3$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) \cdot \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

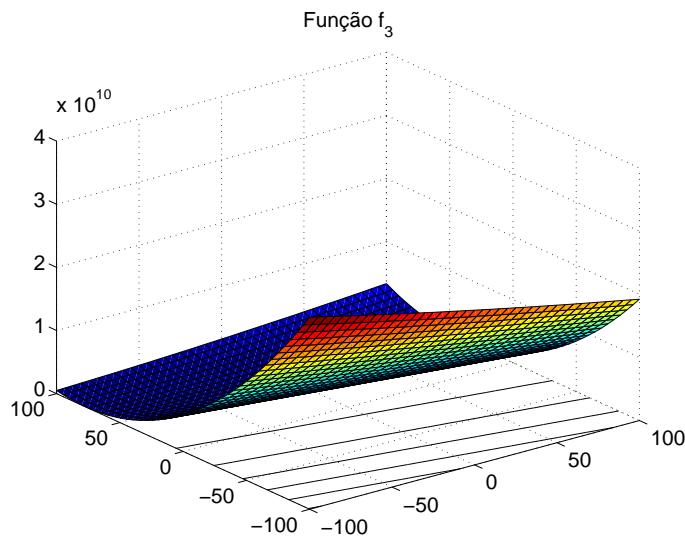


Figura 5.5: Mapa 3D da função  $f_3$ .

- Unimodal
- Deslocamento da origem
- Rotacionada
- Não-separável
- Escalonável
- $\mathbf{M}$  é uma matriz ortogonal qualquer pré-definida
- $\mathbf{x} \in [-100, 100]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}, F_3(\mathbf{x}^*) = bias_3 = -450$

### Função Schwefel 1.2 Deslocada com Ruído

$$f_4(\mathbf{x}) = \left( \sum_{i=1}^D \left( \sum_{j=1}^i z_j \right)^2 \right) * (1 + 0.4 |N(0, 1)|) + bias_4$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

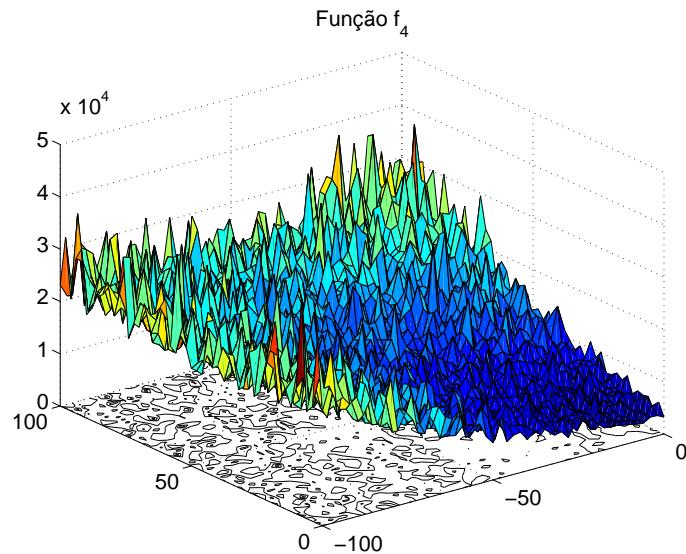


Figura 5.6: Mapa 3D da função  $f_4$ .

- Unimodal
- Deslocamento da origem
- Não-separável
- Escalonável
- Ruído aleatório
- $\mathbf{x} \in [-100, 100]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}$ ,  $F_4(\mathbf{x}^*) = bias_4 = -450$

### Função Schwefel 2.6

$$f_5(\mathbf{x}) = \max \{|\mathbf{A}_i \mathbf{x} - \mathbf{B}_i|\} + bias_5$$

$\mathbf{A}$  é uma matriz  $DxD$ , onde  $a_{ij}$  são números inteiros aleatórios no intervalo  $[-500, 500]$ ,  $\det(\mathbf{A}) \neq 0$ ,  $\mathbf{A}_i$  é a  $i$ -ésima linha de  $\mathbf{A}$ .

$\mathbf{B}_i = \mathbf{A}_i * \mathbf{o}$ ,  $\mathbf{o}$  é um vetor  $Dx1$ , onde  $o_i$  é um número aleatório no intervalo  $[-100, 100]$ .

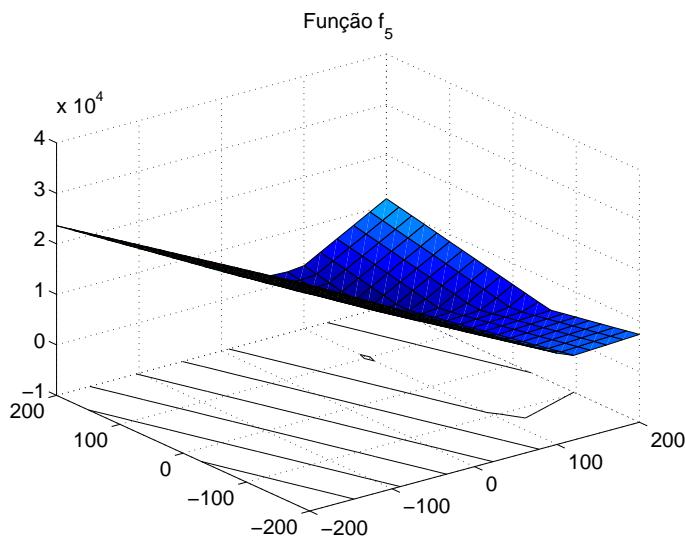


Figura 5.7: Mapa 3D da função  $f_5$ .

- Unimodal
- Não-separável
- Escalonável
- Ótimo global na borda do domínio
- $\mathbf{x} \in [-100, 100]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}, F_5(\mathbf{x}^*) = bias_5 = -310$

### Função Rosenbrock Deslocada

$$f_6(\mathbf{x}) = \sum_{i=1}^D \left( 100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2 \right) + bias_6$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o} + 1, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

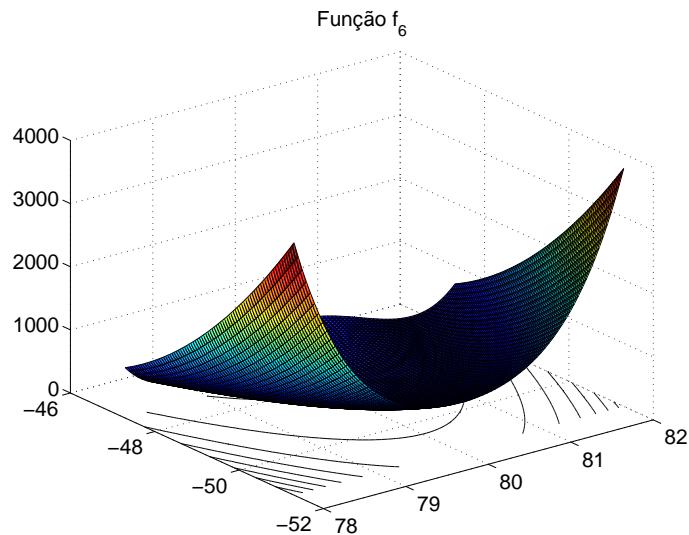


Figura 5.8: Mapa 3D da função  $f_6$ .

- Multi-modal
- Deslocamento da origem
- Não-separável
- Escalonável
- Esta função tem um vale muito estreito entre o mínimo local e o mínimo global
- $\mathbf{x} \in [-100, 100]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}$ ,  $F_6(\mathbf{x}^*) = bias_6 = 390$

### Função Ackley Rotacionada e Deslocada

$$f_8(\mathbf{x}) = -20 \exp \left( -0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2} \right) - \exp \left( \frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i) \right) + 20 + e + bias_8$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) \cdot \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

$\mathbf{M}$  é uma matriz de transformação linear com um número de condição igual a 100.

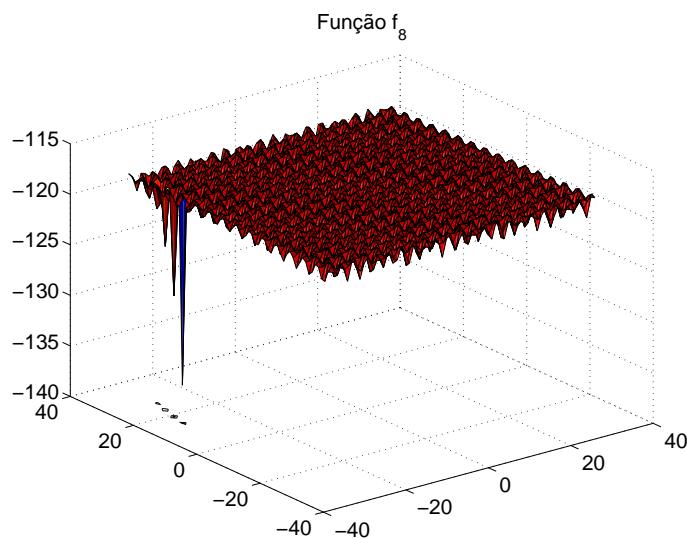


Figura 5.9: Mapa 3D da função  $f_8$ .

- Multi-modal
- Deslocamento da origem
- Rotacionada
- Não-separável
- Escalonável
- Ótimo global na borda do domínio
- $\mathbf{x} \in [-32, 32]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}$ ,  $F_8(\mathbf{x}^*) = bias_8 = -140$

## Função Rastrigin Deslocada

$$f_9(\mathbf{x}) = \sum_{i=1}^D \left( z_i^2 - 10 \cos(2\pi z_i) + 10 \right) + bias_9$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

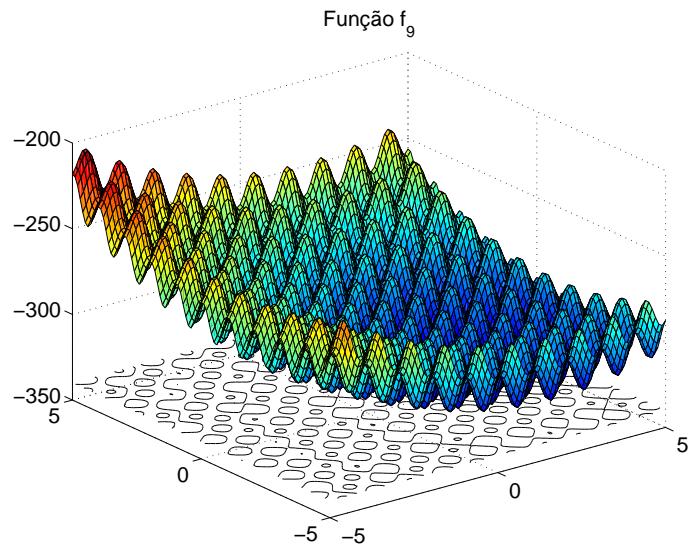


Figura 5.10: Mapa 3D da função  $f_9$ .

- Multi-modal
- Deslocamento da origem
- Separável
- Escalonável
- Grande número de ótimos locais
- $\mathbf{x} \in [-5, 5]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}, F_9(\mathbf{x}^*) = bias_9 = -330$

### Função Rastrigin Deslocada e Rotacionada

$$f_{10}(\mathbf{x}) = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) + bias_{10}, [-5, 5]^D$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) \cdot \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

$\mathbf{M}$  é uma matriz de transformação linear com um número de condição igual a 2.

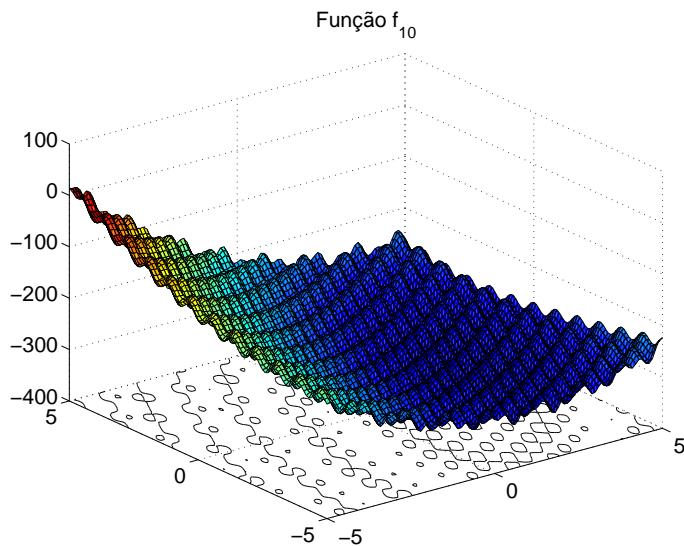


Figura 5.11: Mapa 3D da função  $f_{10}$ .

- Multi-modal
- Deslocamento da origem
- Rotacionada
- Não-Separável
- Escalonável
- Grande número de ótimos locais
- $\mathbf{x} \in [-5, 5]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{0}, F_{10}(\mathbf{x}^*) = bias_{10} = -330$

### Função Weierstrass Deslocada e Rotacionada

$$f_{11}(\mathbf{x}) = \sum_{i=1}^D \left( \sum_{k=0}^{20} [0.5^k \cos(2\pi 3^k (z_i + 0.5))] \right) - D \sum_{k=0}^{20} [0.5^k \cos(2\pi 3^k \cdot 0.5)] + bias_{11}$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) \cdot \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

$\mathbf{M}$  é uma matriz de transformação linear com um número de condição igual a 5.

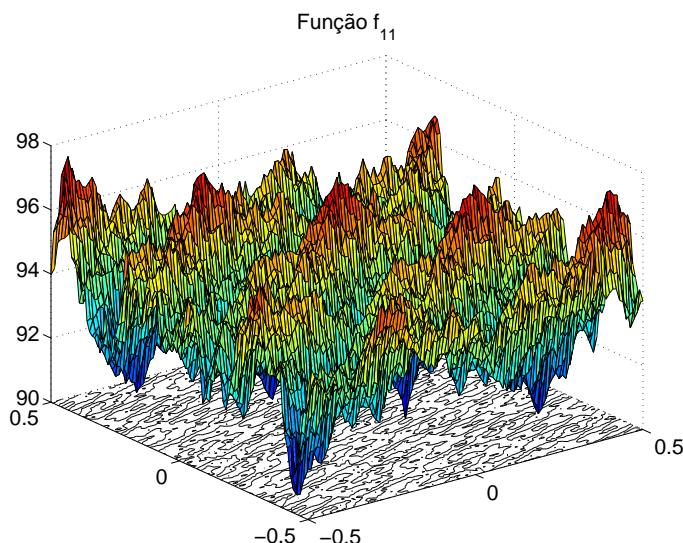


Figura 5.12: Mapa 3D da função  $f_{11}$ .

- Multi-modal
- Deslocamento da origem
- Rotacionada
- Não-Separável
- Escalonável
- Função contínua mas diferenciável apenas em alguns pontos
- $\mathbf{x} \in [-0.5, 0.5]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}, F_{11}(\mathbf{x}^*) = bias_{11} = 90$

### Função Schwefel 2.13

$$f_{12}(\mathbf{x}) = \sum_{i=1}^D (\mathbf{A}_i - \mathbf{B}_i(\mathbf{x}))^2 + bias_{12}$$

$$\mathbf{A}_i = \sum_{j=1}^D (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j)$$

$$\mathbf{B}_i(x) = \sum_{j=1}^D (a_{ij} \sin x_j + b_{ij} \cos x_j), i = 1, \dots, D$$

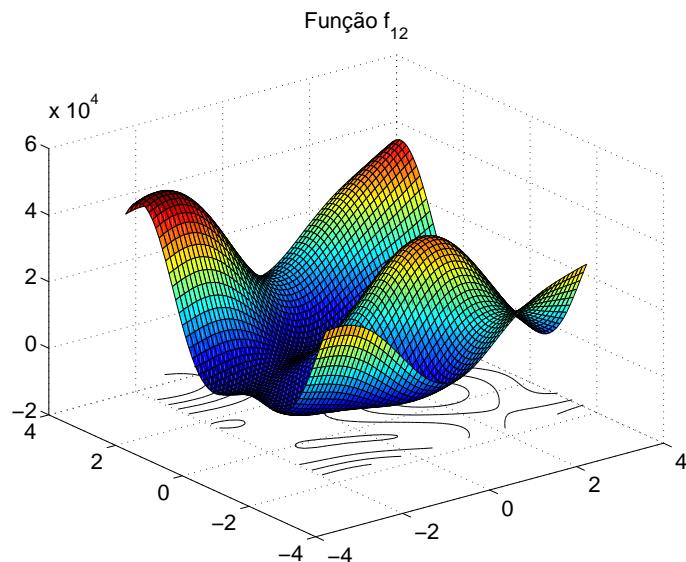


Figura 5.13: Mapa 3D da função  $f_{12}$ .

- Multi-modal
- Deslocamento da origem
- Não-Separável
- Escalonável
- $\mathbf{x} \in [-\pi, \pi]^D$ , Ótimo global:  $\mathbf{x}^* = \alpha, F_{12}(\mathbf{x}^*) = bias_{12} = -460$

### Função Griewank e Rosenbrock Expandida e Deslocada

Considerando-se as funções:

$$F8 - \text{Função Griewank} : F8(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$F2 - \text{Função Rosenbrock} : F2(x) = \sum_{i=1}^{D-1} \left( 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right)$$

$$f_{13}(\mathbf{x}) = F8(F2(z_1, z_2)) + F8(F2(z_2, z_3)) + \cdots + F8(F2(z_{D-1}, z_D)) + \\ F8(F2(z_D, z_1)) + bias_{13}$$

$$\mathbf{z} = \mathbf{x} - \mathbf{o} + 1, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

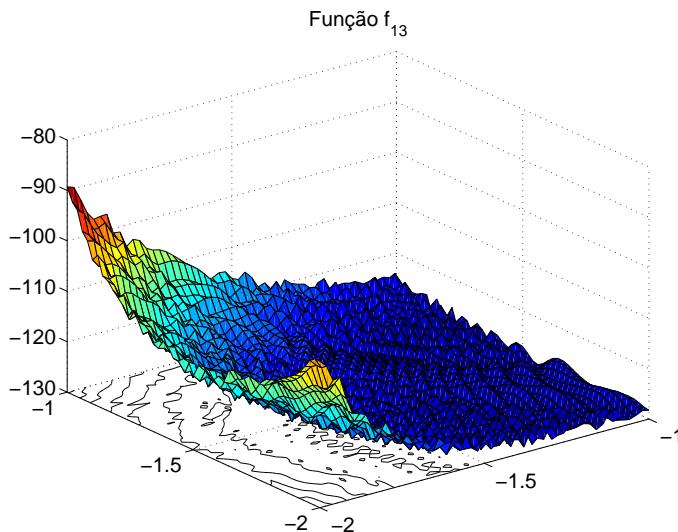


Figura 5.14: Mapa 3D da função  $f_{13}$ .

- Multi-modal
- Deslocamento da origem
- Não-separável
- Escalonável
- $\mathbf{x} \in [-5, 5]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}, F_{13}(\mathbf{x}^*) = bias_{13} = -130$

### Função F6 Expandida, Rotacionada e Deslocada

Considerando-se a função:

$$F(x, y) = 0.5 + \frac{\left(\sin^2\left(\sqrt{x^2 + y^2}\right) - 0.5\right)}{(1 + 0.001(x^2 + y^2))^2}$$

$$f_{14}(\mathbf{x}) = F(z_1, z_2) + F(z_2, z_3) + \cdots + F(z_{D-1}, z_d) + F(z_D, z_1) + bias_{14}$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) \cdot \mathbf{M}, \mathbf{x} = [x_1, x_2, \dots, x_D]$$

$\mathbf{M}$  é uma matriz de transformação linear com um número de condição igual a 5.

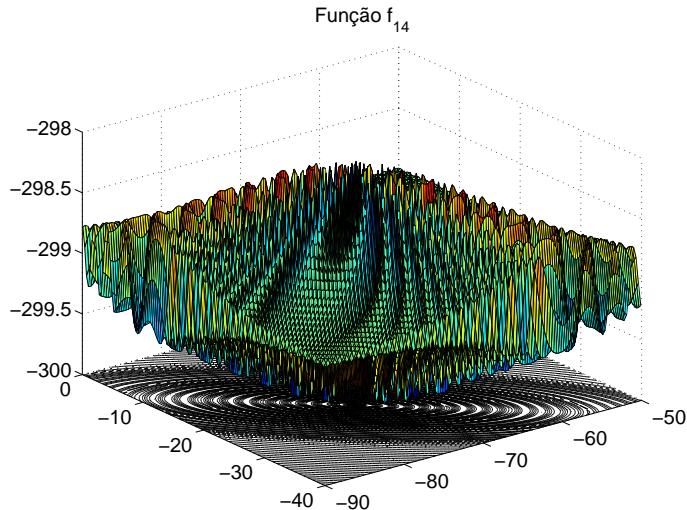


Figura 5.15: Mapa 3D da função  $f_{14}$ .

- Multi-modal
- Deslocamento da origem
- Não-separável
- Escalonável
- $\mathbf{x} \in [-100, 100]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{o}, F_{14}(\mathbf{x}^*) = bias_{14} = -300$

## Anexo 2

### Funções para Medida de Desempenho

#### Função Sphere

$$f_{sphere}(\mathbf{x}) = \sum_{i=1}^D x_i^2$$

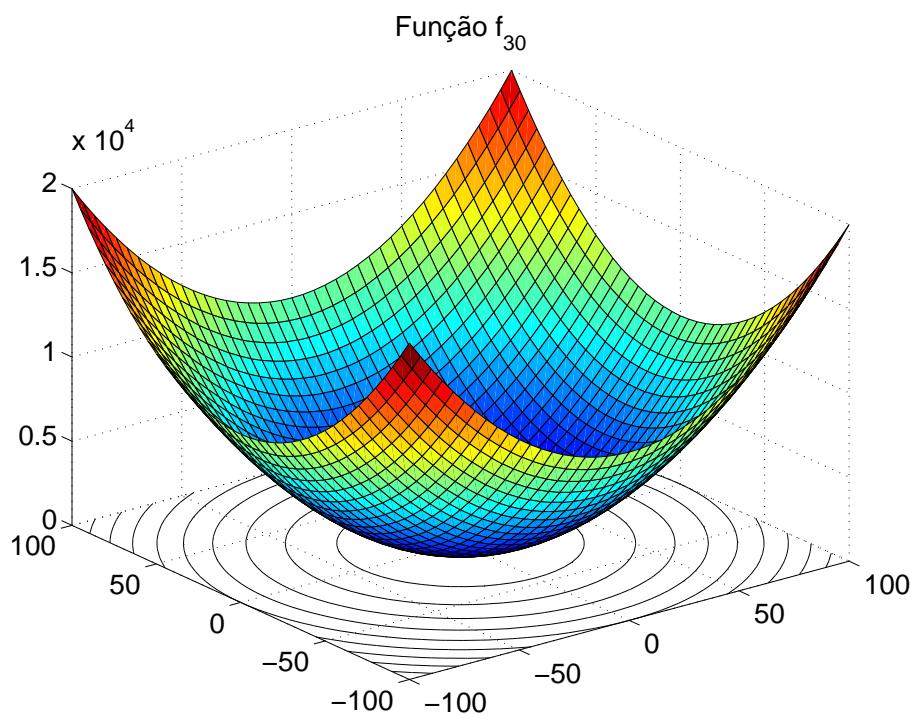


Figura 5.16: Mapa 3D da função  $f_{sphere}$ .

- Unimodal
- $\mathbf{x} \in [-100, 100]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{0}, f_{sphere}(\mathbf{x}^*) = 0$

### Função Ackley

$$f_{\text{ackley}}(\mathbf{x}) = -20 \exp \left( -0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) - \exp \left( \frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i) \right) + 20 + e$$

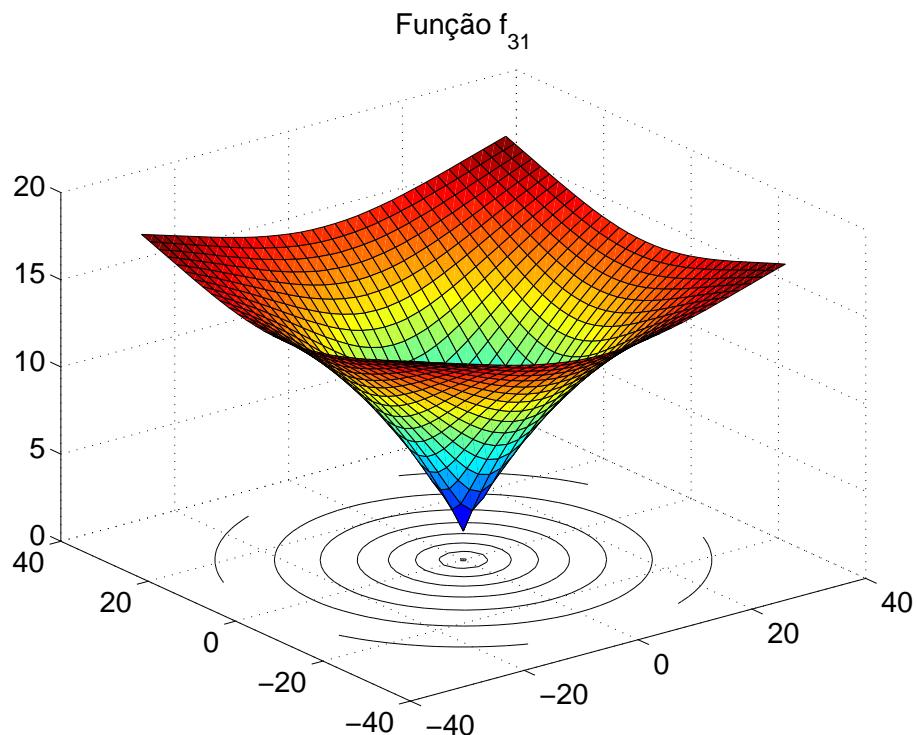


Figura 5.17: Mapa 3D da função  $f_{\text{ackley}}$ .

- Multi-modal
- $\mathbf{x} \in [-32, 32]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{0}$ ,  $f_{\text{ackley}}(\mathbf{x}^*) = 0$

## Função Griewank

$$f_{griewank}(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

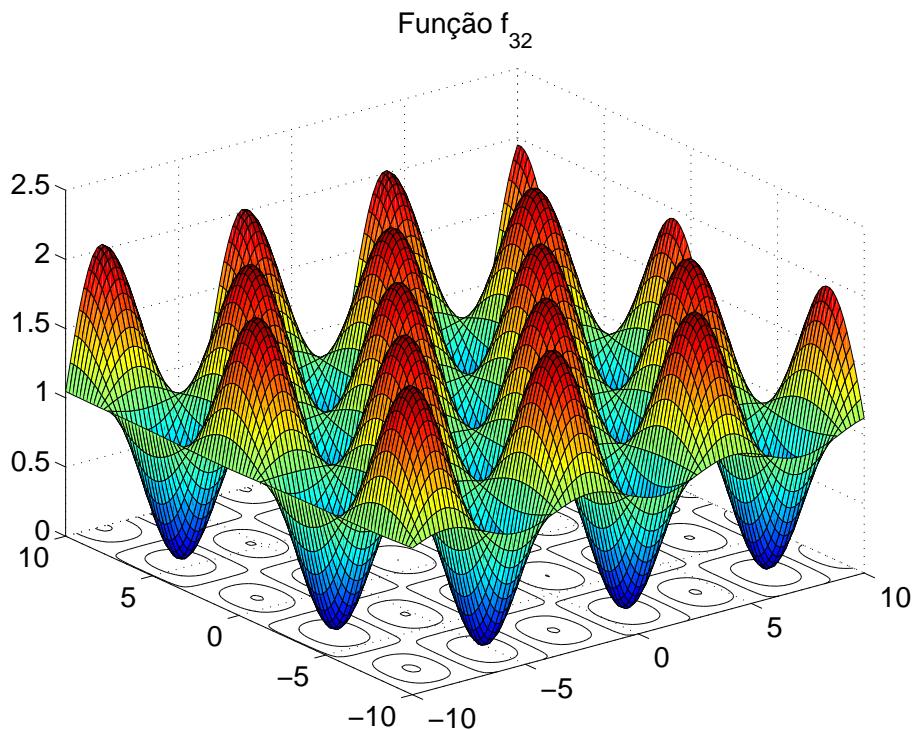


Figura 5.18: Mapa 3D da função  $f_{griewank}$ .

- Multi-modal
- $\mathbf{x} \in [-600, 600]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{0}$ ,  $f_{griewank}(\mathbf{x}^*) = 0$

### Função Rastrigin

$$f_{\text{rastrigin}}(\mathbf{x}) = \sum_{i=1}^D \left[ x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$$

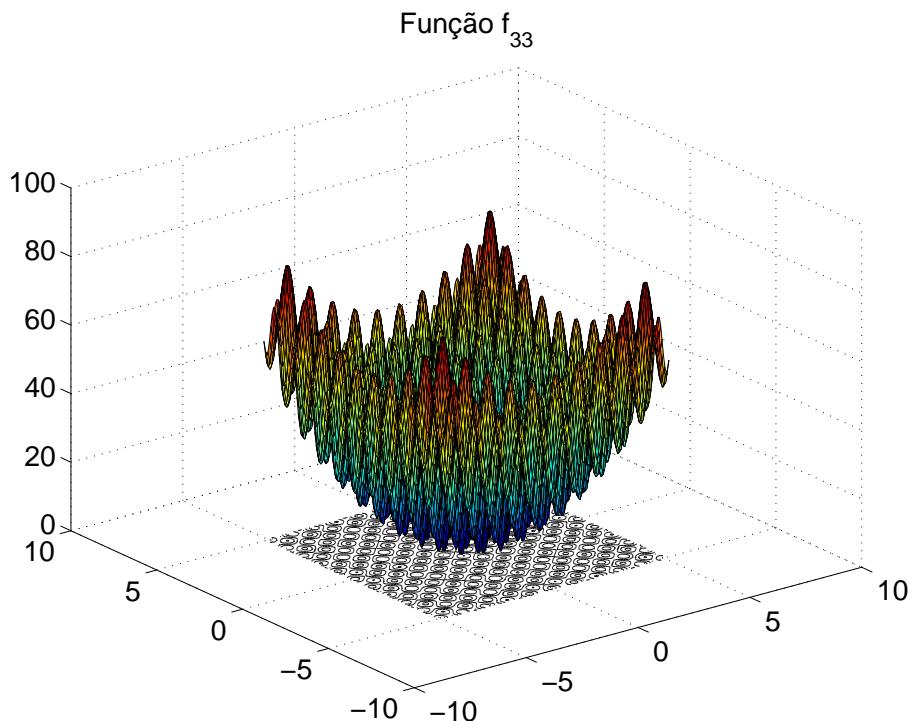


Figura 5.19: Mapa 3D da função  $f_{\text{rastrigin}}$ .

- Multi-modal
- $\mathbf{x} \in [-5.12, 5.12]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{0}$ ,  $f_{\text{rastrigin}}(\mathbf{x}^*) = 0$

### Função Schwefel 2.26

$$f_{schwefel}(\mathbf{x}) = - \sum_{i=1}^D \left( x_i \sin \left( \sqrt{|x_i|} \right) \right) + 12569.5$$

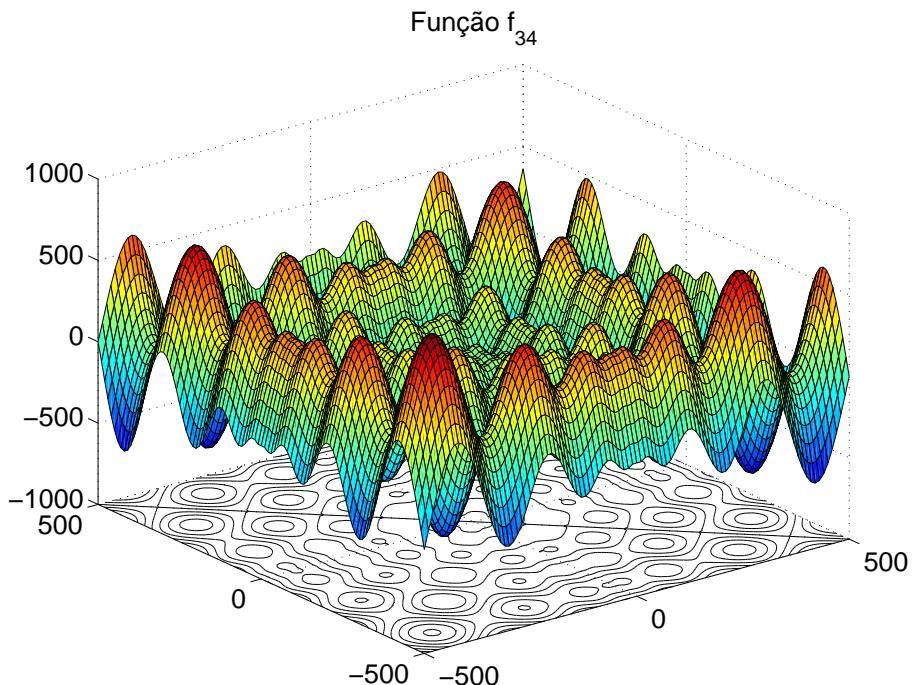


Figura 5.20: Mapa 3D da função  $f_{schwefel}$ .

- Multi-modal
- $\mathbf{x} \in [-500, 500]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{420.9687}$ ,  $f_{schwefel}(\mathbf{x}^*) = 0$

## Função Rosenbrock

$$f_{\text{rosenbrock}}(\mathbf{x}) = \sum_{i=1}^{D-1} \left[ 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$$

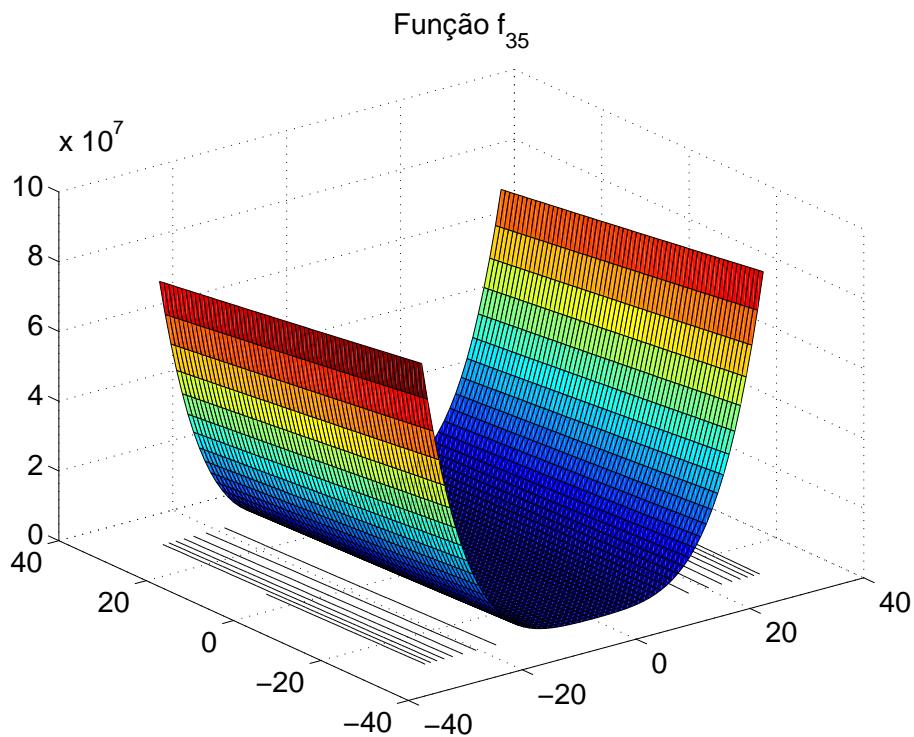


Figura 5.21: Mapa 3D da função  $f_{\text{rosenbrock}}$ .

- Multi-modal
- $\mathbf{x} \in [-30, 30]^D$ , Ótimo global:  $\mathbf{x}^* = \mathbf{1}$ ,  $f_{\text{rosenbrock}}(\mathbf{x}^*) = 0$

# Livros Grátis

( <http://www.livrosgratis.com.br> )

Milhares de Livros para Download:

[Baixar livros de Administração](#)

[Baixar livros de Agronomia](#)

[Baixar livros de Arquitetura](#)

[Baixar livros de Artes](#)

[Baixar livros de Astronomia](#)

[Baixar livros de Biologia Geral](#)

[Baixar livros de Ciência da Computação](#)

[Baixar livros de Ciência da Informação](#)

[Baixar livros de Ciência Política](#)

[Baixar livros de Ciências da Saúde](#)

[Baixar livros de Comunicação](#)

[Baixar livros do Conselho Nacional de Educação - CNE](#)

[Baixar livros de Defesa civil](#)

[Baixar livros de Direito](#)

[Baixar livros de Direitos humanos](#)

[Baixar livros de Economia](#)

[Baixar livros de Economia Doméstica](#)

[Baixar livros de Educação](#)

[Baixar livros de Educação - Trânsito](#)

[Baixar livros de Educação Física](#)

[Baixar livros de Engenharia Aeroespacial](#)

[Baixar livros de Farmácia](#)

[Baixar livros de Filosofia](#)

[Baixar livros de Física](#)

[Baixar livros de Geociências](#)

[Baixar livros de Geografia](#)

[Baixar livros de História](#)

[Baixar livros de Línguas](#)

[Baixar livros de Literatura](#)

[Baixar livros de Literatura de Cordel](#)

[Baixar livros de Literatura Infantil](#)

[Baixar livros de Matemática](#)

[Baixar livros de Medicina](#)

[Baixar livros de Medicina Veterinária](#)

[Baixar livros de Meio Ambiente](#)

[Baixar livros de Meteorologia](#)

[Baixar Monografias e TCC](#)

[Baixar livros Multidisciplinar](#)

[Baixar livros de Música](#)

[Baixar livros de Psicologia](#)

[Baixar livros de Química](#)

[Baixar livros de Saúde Coletiva](#)

[Baixar livros de Serviço Social](#)

[Baixar livros de Sociologia](#)

[Baixar livros de Teologia](#)

[Baixar livros de Trabalho](#)

[Baixar livros de Turismo](#)