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 neuroevolução, 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 desejou 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, y'_1 e y'_2, a partir dos genitores y_1 e y_2 de acordo com a equação:

$$y'_1 = ry_1 + (1 - r)y_2 \tag{4-1}$$

$$y'_2 = ry_2 + (1 - r)y_1 \tag{4-2}$$

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 γ.

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 ζ.

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 (estes números de avaliações são usados em todos os experimentos para fazer com que a comparação entre os algoritmos seja justa). 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 γ	5
Taxa de Recombinação	66%
Intervalo de Atualização da População Quântica	10 gerações
Taxa de Atualização dos Indivíduos Quânticos ζ	16.6%

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

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 quão melhor o AEIQ–ℝ 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.

	Evolução Diferencial		Enxame de Partículas		AEIQ–ℝ		
	Erro Médio	Desvio Padrão	Erro Médio	Desvio Padrão	Erro Médio	Desvio Padrão	Perc.
f_1	$5.440E+04$	$7.208E+03$	$2.331E+04$	$4.886E+03$	**$2.286E+04$**	**$3.813E+03$**	1.95%
f_2	$8.592E+04$	$1.742E+04$	$6.041E+04$	$1.203E+04$	**$3.570E+04$**	**$5.594E+03$**	40.90%
f_3	$1.009E+09$	$2.390E+08$	$4.970E+08$	$1.517E+08$	**$3.553E+08$**	**$7.095E+07$**	28.51%
f_4	$9.984E+04$	$1.864E+04$	$6.810E+04$	$1.521E+04$	**$4.627E+04$**	**$8.760E+03$**	32.06%
f_5	$2.395E+04$	**$2.631E+03$**	**$1.876E+04$**	$3.792E+03$	$2.546E+04$	$3.381E+03$	26.32%
f_6	$2.600E+10$	$9.428E+09$	$6.354E+09$	$2.852E+09$	**$5.501E+09$**	**$1.846E+09$**	13.42%
f_8	$2.121E+01$	$6.801E-02$	**$2.118E+01$**	$7.595E-02$	$2.119E+01$	**$6.070E-02$**	0.03%
f_9	$4.045E+02$	$2.9405E+01$	$3.398E+02$	$2.191E+01$	**$3.072E+02$**	**$1.957E+01$**	9.59%
f_{10}	$4.696E+02$	$3.163E+01$	$3.521E+02$	**$2.498E+01$**	**$3.039E+02$**	$5.469E+01$	13.69%
f_{11}	$4.554E+01$	$1.282E+00$	$4.580E+01$	$1.448E+00$	**$4.186E+01$**	**$1.821E+00$**	8.08%
f_{12}	$1.515E+06$	$2.018E+05$	$1.240E+06$	$1.803E+05$	**$2.691E+05$**	**$9.929E+04$**	78.31%
f_{13}	$2.283E+05$	$1.061E+05$	$1.376E+04$	$1.158E+04$	**$1.192E+01$**	**$5.993E+01$**	99.91%
f_{14}	$1.416E+01$	**$1.483E-01$**	$1.408E+01$	$1.523E-01$	**$1.389E+01$**	$1.832E-01$	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–ℝ		
	Erro Médio	Desvio Padrão	Erro Médio	Desvio Padrão	Erro Médio	Desvio Padrão	Perc.
f_1	$1.279E+04$	$2.628E+03$	$1.025E+03$	$6.393E+02$	**$1.232E+01$**	**$7.060E+00$**	99.88%
f_2	$4.919E+04$	$9.254E+03$	$1.554E+04$	$4.128E+03$	**$1.097E+04$**	**$1.878E+03$**	29.41%
f_3	$3.268E+08$	$8.828E+07$	$7.558E+07$	$3.554E+07$	**$2.456E+07$**	**$5.439E+06$**	67.50%
f_4	$5.082E+04$	$8.594E+03$	$2.227E+04$	$5.343E+03$	**$2.052E+04$**	**$3.078E+03$**	7.86%
f_5	$1.047E+04$	**$1.345E+03$**	$1.026E+04$	$3.720E+03$	**$9.323E+03$**	$1.379E+03$	9.19%
f_6	$1.656E+09$	$8.344E+08$	$2.504E+07$	$2.063E+07$	**$8.364E+04$**	**$8.323E+04$**	99.67%
f_8	$2.109E+01$	$5.957E-02$	**$2.106E+01$**	**$5.543E-02$**	$2.107E+01$	$6.080E-02$	0.04%
f_9	$2.761E+02$	$2.175E+01$	**$1.040E+02$**	$2.698E+01$	$1.533E+02$	**$2.069E+01$**	32.18%
f_{10}	$3.349E+02$	$2.614E+01$	$1.962E+02$	$4.852E+01$	**$1.780E+02$**	**$2.698E+01$**	9.27%
f_{11}	$4.285E+01$	**$9.645E-01$**	$3.651E+01$	$3.835E+00$	**$3.333E+01$**	$3.142E+00$	8.70%
f_{12}	$7.100E+05$	$1.365E+05$	$1.234E+05$	$7.206E+04$	**$7.219E+04$**	**$3.421E+04$**	41.50%
f_{13}	$4.005E+03$	$1.868E+03$	$2.492E+01$	$5.348E+00$	**$1.026E+01$**	**$1.137E+00$**	58.82%
f_{14}	$1.390E+01$	**$1.323E-01$**	$1.358E+01$	$2.131E-01$	**$1.321E+01$**	$2.183E-01$	2.77%

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

	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–ℝ, comparando-o com o segundo melhor algoritmo, em relação às características das funcões 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 permitir que o AEIQ–$\mathbb{R}$ faça mais avaliações, o seu desempenho melhora com relação aos outros algoritmos. Isto sugere que o AEIQ–$\mathbb{R}$ 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–$\mathcal{B}$ é feita a partir de resultados tirados de (Yao99) (CEP e FEP) e (Han04) (AEIQ–$\mathcal{B}$). 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–$\mathbb{R}$ 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.

Função Densidade de Probabilidade	Pulso Quadrado
Número de Indivíduos Quânticos	5
Número de Indivíduos Clássicos	100
Gap γ	99
Taxa de Recombinação	66%
Intervalo de Atualização da População Quântica	10 gerações
Taxa de Atualização dos Indivíduos Quânticos ζ	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 γ	99
Taxa de Recombinação	3.33%
Intervalo de Atualização da População Quântica	1 geração
Taxa de Atualização dos Indivíduos Quânticos ζ	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.

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, 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,

Função		AEIQ–$\mathcal{B}$	FEP	CEP	GA	AEIQ–$\mathbb{R}$	Perc.
f_{Sphere}	Média	$1.8E-04$	$5.7-04$	$2.2E-04$	$4.71E+00$	**1.99E – 06**	98.99%
Aval.= 150000	Desvio Padrão	$1.3E-04$	$1.3E-04$	$5.9E-04$	n.d.	**9.50E – 06**	n.d.
f_{Ackley}	Média	$2.5E-03$	$1.8E-02$	9.2	$4.71E-01$	**4.26E – 05**	98.3%
Aval.= 150000	Desvio Padrão	$8.1E-04$	$2.1E-03$	2.8	n.d.	**1.39E – 04**	n.d.
$f_{Griewank}$	Média	$3.6E-02$	$1.6E-02$	$8.6E-02$	$1.03E+00$	**9.42E – 06**	99.94%
Aval.= 200000	Desvio Padrão	$3.2E-02$	$2.2E-02$	0.12	n.d.	**1.55e – 05**	n.d.
$f_{Rastrigin}$	Média	$3.9E-02$	$4.6E-02$	89.0	$4.40E-01$	**1.65E – 11**	100.00%
Aval.= 500000	Desvio Padrão	$1.9E-01$	$1.2E-02$	23.1	n.d.	**3.39E – 12**	n.d.
$f_{Schwefel}$	Média	$3.8E-04$	14.987	4652.3	$1.77E+00$	**8.13E – 09**	100.00%
Aval.= 900000	Desvio Padrão	$3.0E-09$	52.6	634.5	n.d.	**0**	n.d.
$f_{Rosenbrock}$	Média	11.73	5.06	6.17	$14.15E+00$	**2.52**	78.52%
Aval.= 2000000	Desvio Padrão	18.36	5.87	13.61	n.d.	**4.43**	n.d.

Tabela 4.7: Resultado comparativo entre o AEIQ–$\mathbb{R}$, o AEIQ–$\mathcal{B}$ (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.

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 computacional do algoritmo para estas funções e para o número de dimensões

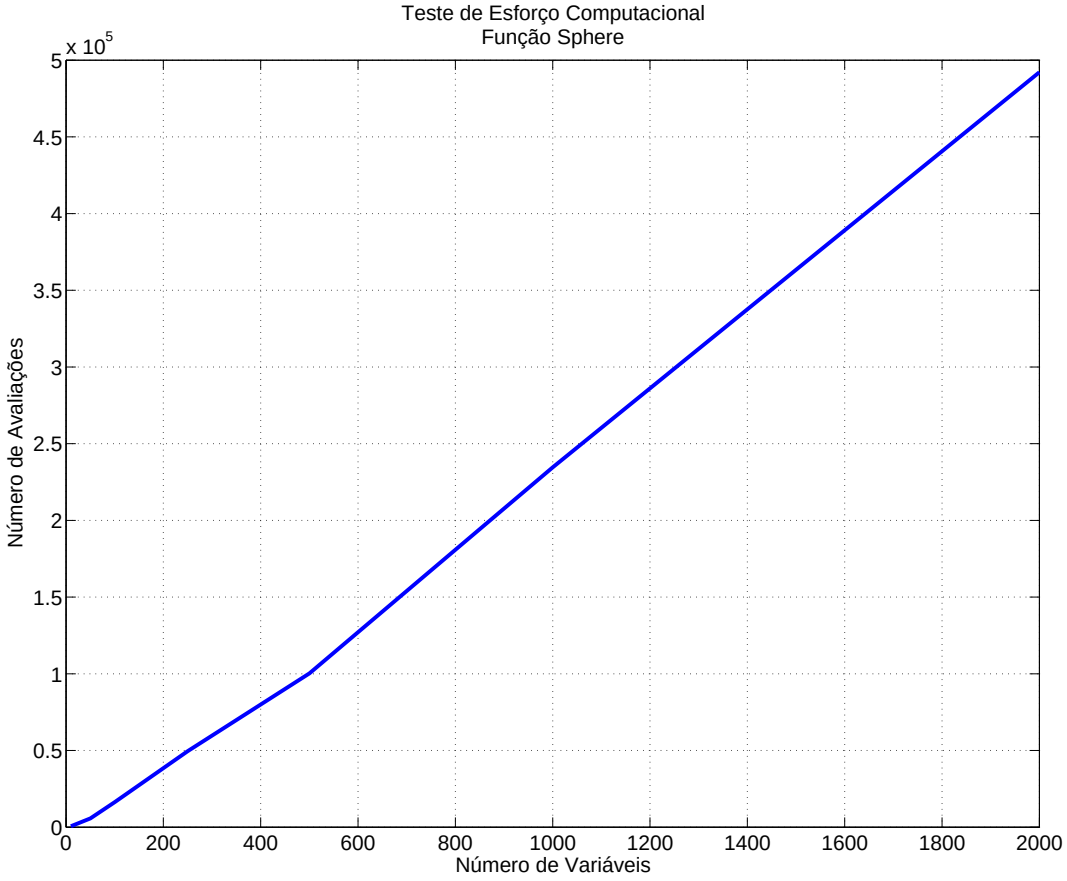

Figura 4.1: Esforço computacional para otimização da função f_{Sphere}.

utilizadas é, aproximadamente, $O(n)$. Este teste sugere que o AEIQ–$\mathbb{R}$ não tenha, talvez, uma complexidade exponencial na otimização de problemas. Em outras palavras, a complexidade do AEIQ–$\mathbb{R}$ 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–$\mathbb{R}$ 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–$\mathbb{R}$ obteve um desempenho muito superior ao AEIQ–$\mathcal{B}$ para a otimização dos problemas de otimização numérica aqui mostrados. Isto demonstra a importância de um algoritmo com uma representação específica para números reais. Comparado com o AEIQ–$\mathcal{B}$, o AEIQ–$\mathbb{R}$ obteve, pelo menos, um resultado duas

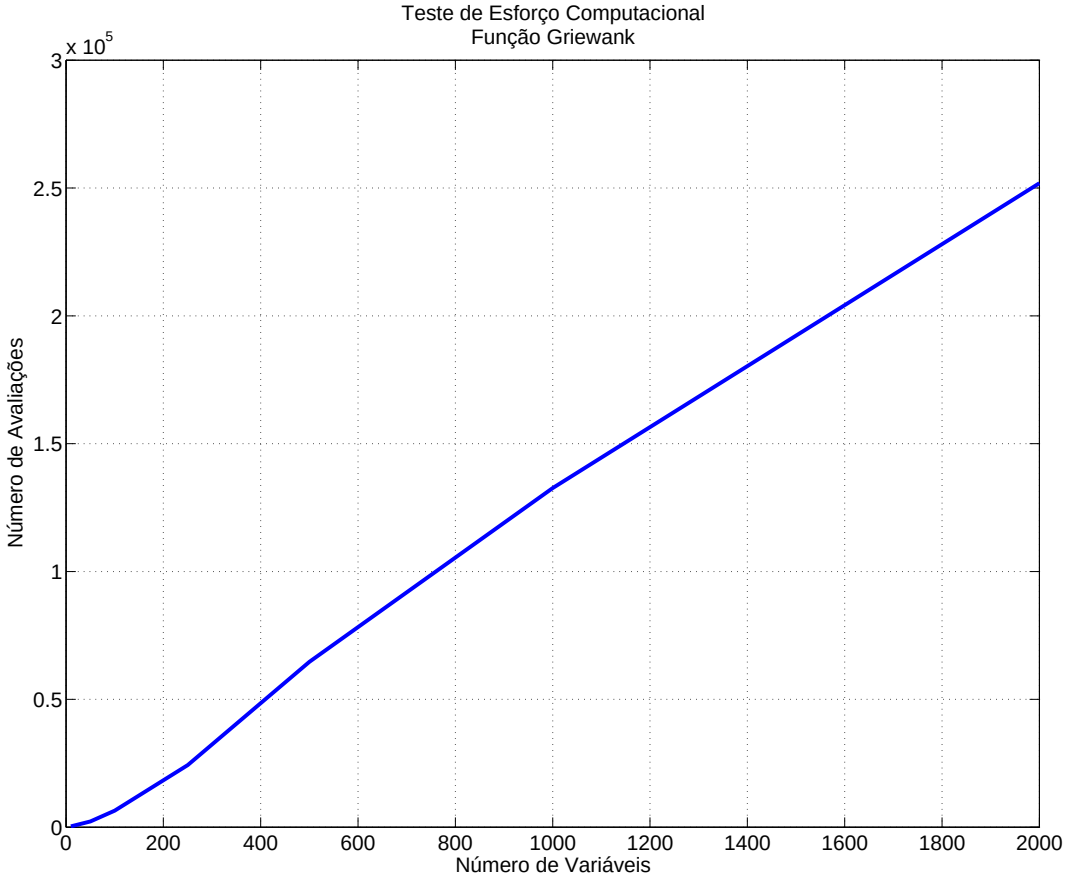

Figura 4.2: Esforço computacional para otimização da função $f_{Griewank}$.

ordens de grandeza menor e, em alguns casos, como no teste com a função $f_{Rastrigin}$, o AEIQ–$\mathbb{R}$ 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–$\mathbb{R}$. 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–$\mathbb{R}$ 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–$\mathbb{R}$ 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 clás-

sica. 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 variá-

veis 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 mais bem otimizadas quando se usam 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 ζ 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%.

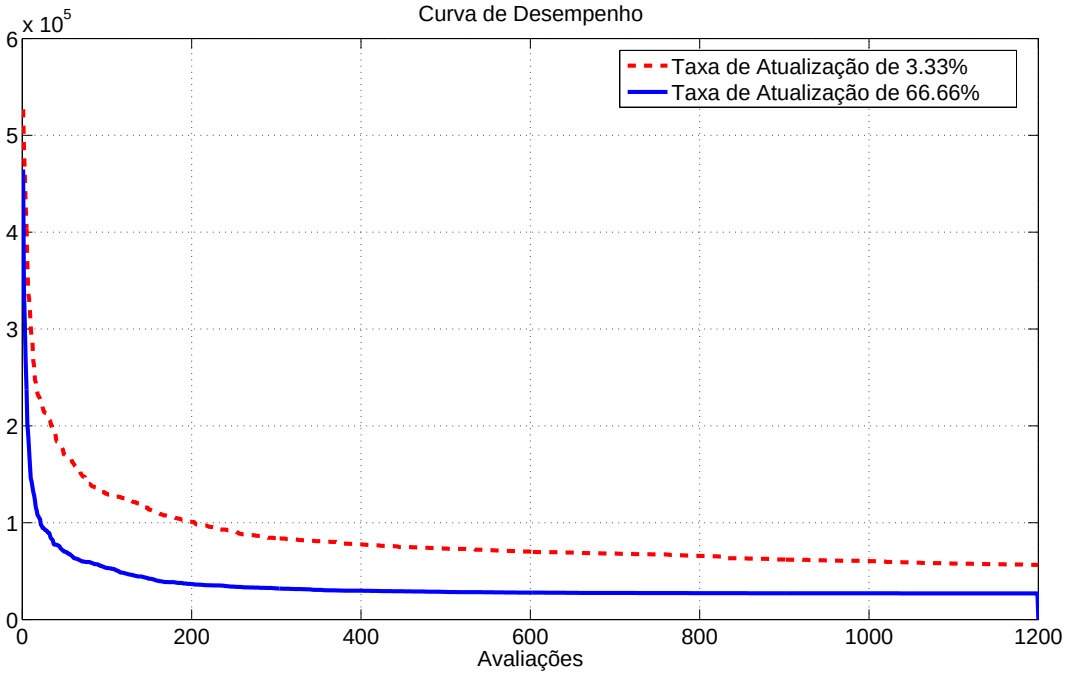

Figura 4.3: Gráfico de desempenho para uma função não separável usando taxas de atualização diferentes.

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 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

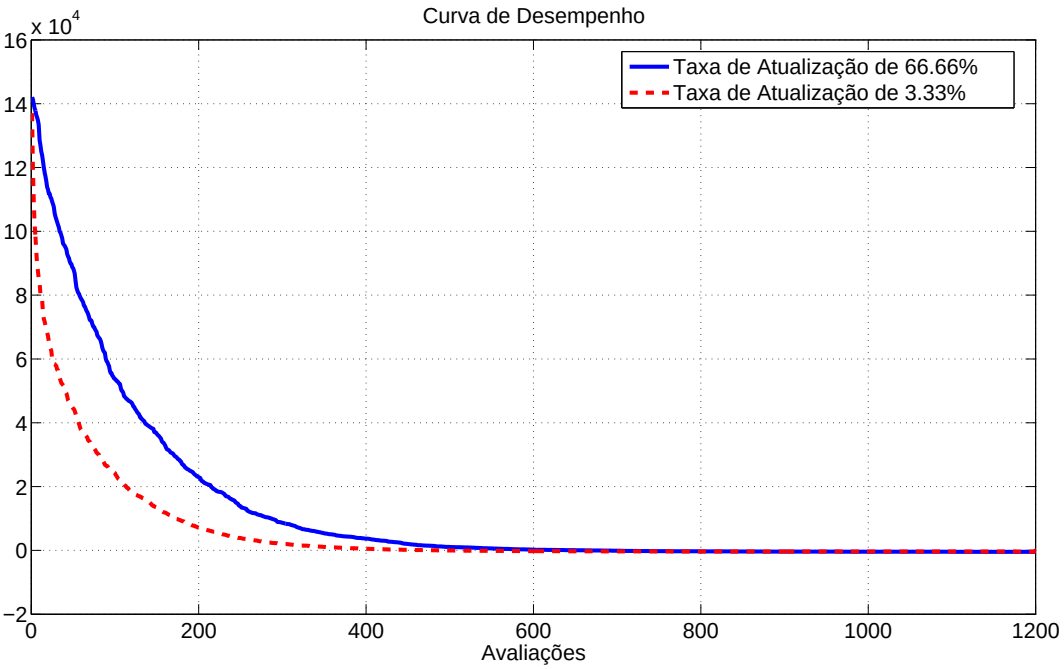

Figura 4.4: Gráfico de desempenho para uma função separável usando taxas de atualização diferentes.

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 γ 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 atualiza, 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–ℝ em problemas de previsão de séries, usou-se uma base de dados de vazão média semanal na bacia hidrográfica do Paranaíba. 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–ℝ 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 η 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 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.

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 γ	5
Taxa de recombinação	0.5
Taxa de Atualização dos Genes Quânticos ζ	0.5
Número de Gerações	250
Total de Experimentos	30
Intervalo de Atualização da População Quântica	5 gerações
Número de Gerações η 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.

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.

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

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 .

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.

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

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

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

$$f(x) = \sin(3x) \tag{4-5}$$

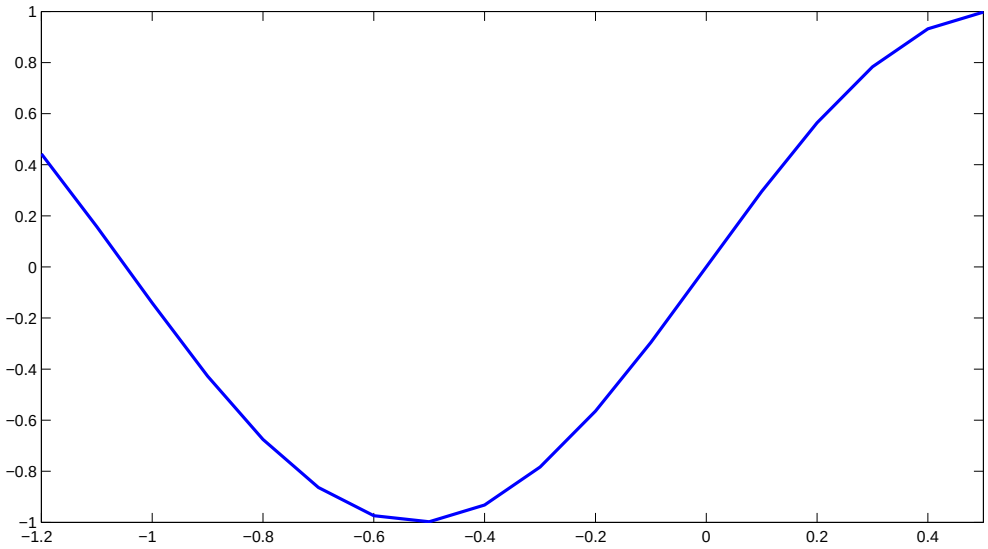

Figura 4.5: Representação gráfica do problema do carro na montanha.

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

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 γ	3
Taxa de recombinação	0.5
Taxa de Atualização dos Genes Quânticos ζ	0.5
Número de Gerações	200
Total de Experimentos	5
Intervalo de Atualização da População Quântica	5 gerações
Número de Gerações η 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.

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

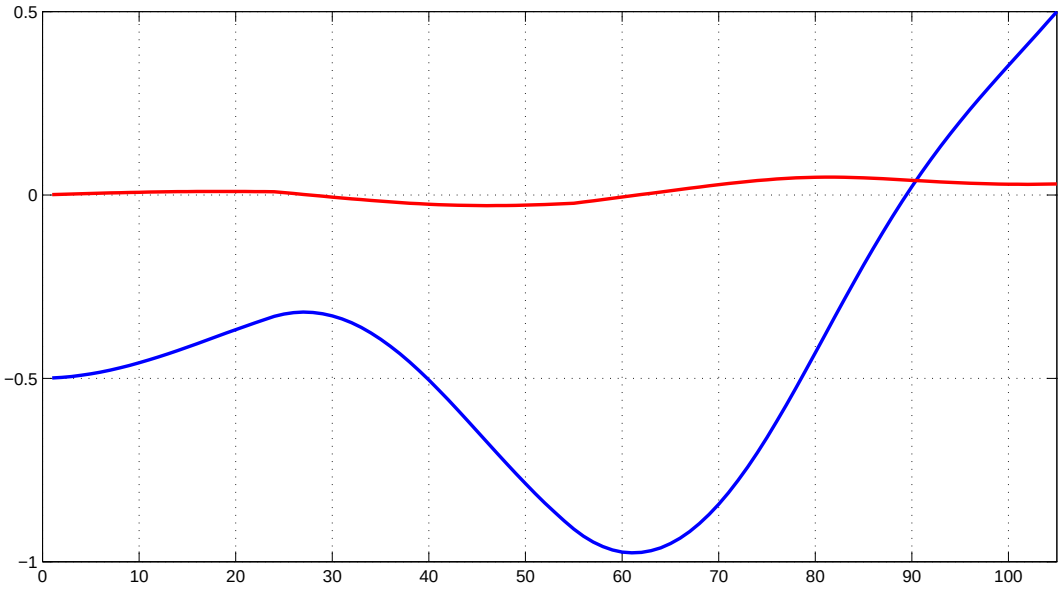

Figura 4.6: Variação da velocidade (linha vermelha) e da posição (linha azul) no problema do carro na montanha.

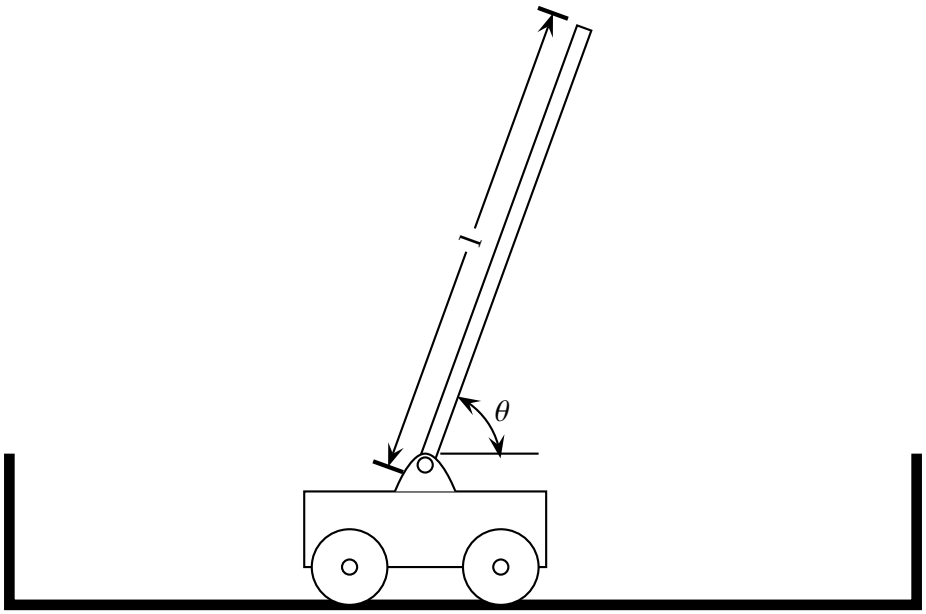

Figura 4.7: Exemplo do problema do pêndulo invertido.

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

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

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

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

Onde x é a posição do carro no trilho, θ_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, μ_c é o coeficiente de atrito do carro com o trilho e μ_{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	$[-2.4, 2.4]m$
θ	$[-0.209, 0.209]rad$
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 considera que o controlador falhou em manter o pêndulo equilibrado. Foram feitos

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 γ	5
Taxa de recombinação	0.5
Taxa de Atualização dos Genes Quânticos ζ	0.5
Número de Gerações	250
Total de Experimentos	5
Intervalo de Atualização da População Quântica	3 gerações
Número de Gerações η 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.

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(λ) com Aproximador de Função baseado em Casos (SARSA-CABA);
- Sarsa(λ) 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.

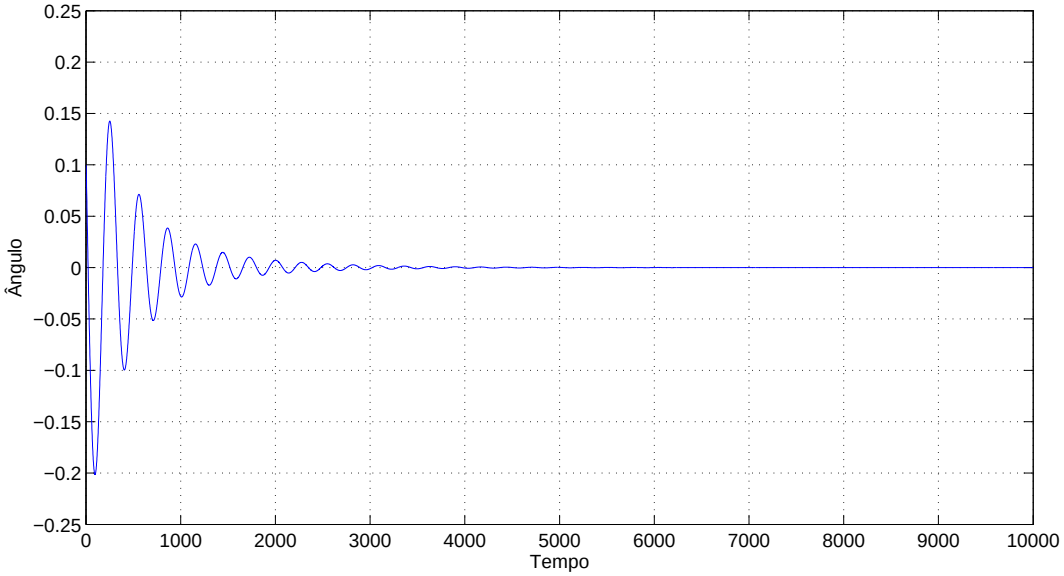

Figura 4.8: Ângulo do pêndulo com relação ao tempo.

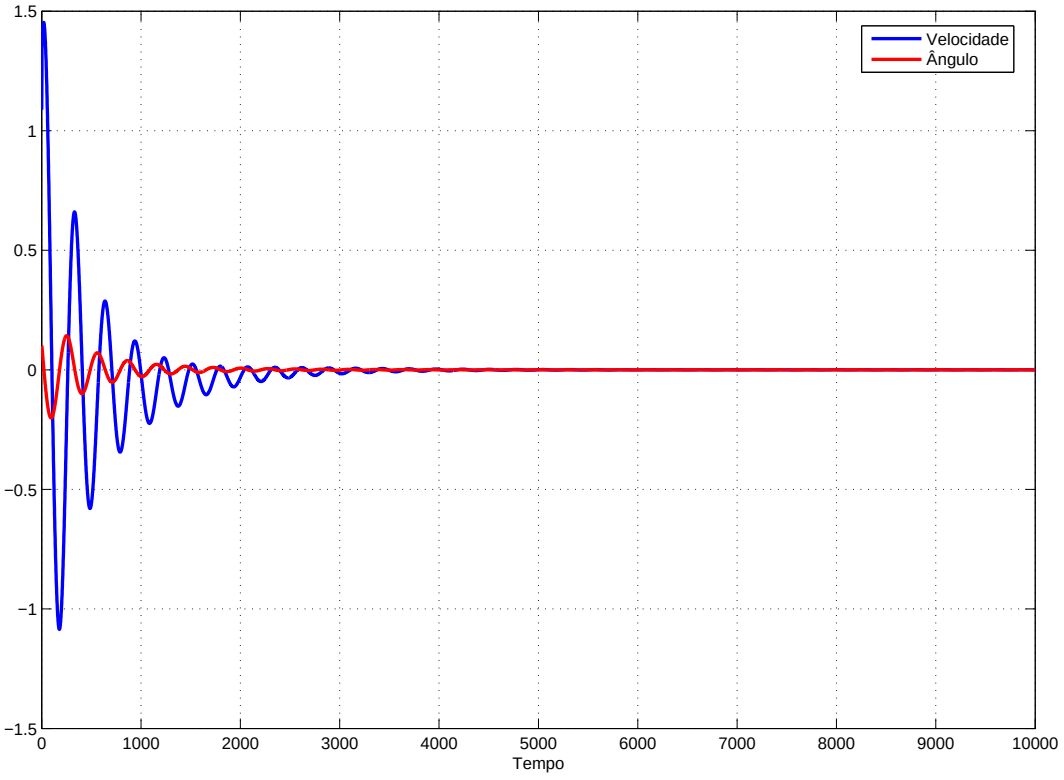

Figura 4.9: Ângulo do pêndulo e velocidade do carro com relação ao tempo.

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.

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.