PROGRAMAÇÃO EVOLUTIVA COM DISTRIBUIÇÃO DE MUTAÇÕES AUTO-ADAPTATIVA APLICADA A REDES NEURAIS ARTIFICIAIS

Renato Tinós*

*Departamento of Física e Matemática, FFCLRP, Universidade de São Paulo (USP)
Ribeirão Preto, SP, 14040-901, Brasil

Email: rtinos@ffclrp.usp.br

Abstract— Evolutionary Programming with self-adaptation of the mutation distribution is proposed to optimize the synaptic weights of an Artificial Neural Network (ANN). For this purpose, the isotropic q-Gaussian distribution is employed in the mutation operator. The q-Gaussian distribution allows to control the shape of the distribution by setting a real parameter q. In the proposed method, the real parameter q of the q-Gaussian distribution is encoded in the chromosome of an individual and is allowed to evolve. An Evolutionary Programming algorithm with self-adaptation of the mutation distribution is proposed to adapt and re-adapt the synaptic weights of an ANN. Experiments were carried out to study the performance of the proposed algorithm.

Keywords— Evolutionary Programming, q-Gaussian, Evolutionary Computation, Neural Networks.

Resumo— Um algoritmo de Programação Evolutiva com auto-adaptação da distribuição de mutações é proposto para a otimização de pesos de uma Rede Neural Artificial (RNA) neste trabalho. Para este propósito, distribuições do tipo q-Gaussiana isotrópicas são utilizadas para gerar novas soluções através do operador de mutação. Mudando-se o parâmetro real q é possivel mudar o formato da distribuição q-Gaussiana, reproduzindo distribuições como a Gaussiana e a de Cauchy. No algoritmo proposto, o parâmetro q é codificado no cromossomo dos indivíduos e submetido à evolução. Experimentos envolvendo a adaptação e re-adaptação de pesos de uma RNA através do algoritmo proposto são utilizados para a análise do desempenho do algoritmo.

Keywords— Programação Evolutiva, Distribuição q-Gaussiana, Computação Evolutiva, Redes Neurais.

1 Introdução

Algoritmos Evolutivos (AEs) têm sido utilizados com sucesso em diversas tarefas relacionadas ao projeto de soluções baseadas em Redes Neurais Artificiais (RNAs). Entre estas tarefas, podem ser citadas (Yao, 1999): a definição de arquiteturas, o ajuste de pesos sinápticos, a adaptação das leis de aprendizado e a inicialização dos pesos sinápticos. O ajuste de pesos através de AEs é particularmente atrativo, já que diversas soluções são buscadas em paralelo, diminuindo assim a probabilidade, em relação a técnicas tradicionais como o *Backpropagation*, de o algoritmo ficar rapidamente preso em ótimos locais em superfícies de erros vastas, complexas, multimodais e/ou não-diferenciáveis (Yao, 1999). Além disso, para os AEs, não é obrigatório o conhecimento das saídas desejadas da RNA, sendo necessária somente uma medida de eficiência proveniente do ambiente. No entanto, o preço que se paga por estas vantagens é, em geral, o grande tempo necessário para o treinamento da RNA.

Quando AEs com codificação real são aplicados, como por exemplo na otimização dos pesos de uma RNA, o operador de mutação geralmente emprega distribuições Gaussianas isotrópicas para gerar novas soluções candidatas (Beyer and Schwefel, 2002). O uso de distribuições Gaussianas isotrópicas é interessante porque ela maximiza, sob certas condições, a Entropia de Boltzmann-Gibbs (e consequentemente a Entropia da Informação de Shannon para o caso contínuo) (Thistleton et al., 2006). Como a distribuição Gaussiana isotrópica não privilegia nenhuma direção de busca, a geração de novas soluções candidatas não requer nenhum conhecimento sobre a geometria do espaço de busca.

Contudo, pesquisadores têm proposto recentemente o uso de distribuições com caudas mais longas e com segundo momento infinito em AEs, em oposição à distribuição Gaussiana isotrópica, que tem segundo momento finito. Como exemplos, podem ser citados o *Fast Evolutionary Programming* (FEP), que utiliza a distribuição de Cauchy, e a Programação Evolutiva com distribuição de Lévy (LEP) (Lee and Yao, 2004). O uso de mutações com distribuições com segundo momento infinito implica em saltos com tamanhos livres de escala, permitindo que regiões mais distantes sejam alcançadas no espaço de busca. Esta propriedade pode ser interessante em problemas multimodais ou de otimização dinâmica já que pode fazer com que as soluções candidatas escapem com mais facilidades dos ótimos locais. Contudo, alguns pesquisadores argumentam que, em geral, os algoritmos propostos utilizam distribuições anisotrópicas, ou seja, em que algumas direções do espaço de busca são privilegiadas (Obuchowicz, 2003), o que seria interessante somente para problemas com funções de aptidão separáveis. Além disso, em geral, é difícil atingir uma região interessante do espaço de busca com um salto longo partindo de um ótimo local (Hansen et al., 2006), já que a aptidão da solução corrente é em geral maior que a aptidão média do

espaço de busca.

Neste trabalho, o uso de AEs, mais especificamente Programação Evolutiva (PE), com auto-adaptação da distribuição de mutações é proposto para a otimização de pesos de uma RNA. Desta forma, a decisão de escolher qual distribuição de mutações é mais indicada para o problema, ou para um determinado momento do processo evolutivo, fica a cargo do algoritmo, e não mais do projetista. No algoritmo proposto, distribuições do tipo q-Gaussiana isotrópicas, derivadas do conceito de entropia generalizada de Tsallis (Thistleton et al., 2006), são utilizadas. Mudando-se o parâmetro real q é possivel mudar o formato da distribuição q-Gaussiana, reproduzindo distribuições com segundo momento finito, como a distribuição Gaussiana, e com segundo momento infinito, como a distribuição de Cauchy. No algoritmo proposto neste trabalho, o parâmetro q que define a distribuição de mutações é codificado no cromossomo dos indivíduos e é submetido à evolução.

Na próxima seção, a distribuição q-Gaussiana é brevemente introduzida. Na Seção 3, o algoritmo proposto é descrito. Estudos experimentais envolvendo a aplicação do algoritmo proposto para o ajuste de pesos em RNAs em problemas estacionários e não-estacionários são apresentados na Seção 4. Finalmente, o artigo é concluído na Seção 5.

2 A Distribuição q-Gaussiana

Apesar de a distribuição Gaussiana ser um atrator para sistemas com componentes independentes e com variância finita, ela não representa bem sistemas correlacionados com segundo momento infinito (Thistleton et al., 2006). A distribuição q-Gaussiana pode ser vista como uma generalização da distribuição Gaussiana, aparecendo quando a entropia generalizada de Tsallis é maximizada. Um parâmetro real q controla a forma da distribuição q-Gaussiana, o que lhe confere propriedades interessantes. Para valores de q menores que 5/3, a distribuição apresenta segundo momento finito e para $q = 1$ ela reproduz a distribuição Gaussiana. Quanto $q < 1$, a distribuição q-Gaussiana apresenta uma forma compacta, e decai assintoticamente como uma lei de potência para $1 < q < 3$. Para $q = 2$, a distribuição reproduz a distribuição de Cauchy.

Uma variável aleatória x com distribuição q-Gaussiana com q-média $\bar{\mu}_q$ e q-variância $\bar{\sigma}_q^2$ é denotada aqui por $x \sim \mathcal{N}_q(\bar{\mu}_q, \bar{\sigma}_q)$. Neste trabalho, o método de Box-Müller generalizado proposto em (Thistleton et al., 2006), o qual é muito simples (ver pseudo-código em (Thistleton et al., 2006)), é empregado para a geração de variáveis aleatórias $x \sim \mathcal{N}_q(0, 1)$ para $q < 3$.

3 PE com Auto-Adaptação da Distribuição de Mutações

Geralmente em PE, cada indivíduo $\vec{x}_i$, sendo $i = 1, \ldots, \mu$ e μ o número de indivíduos da população, gera um descendente por mutação através de

$$\tilde{\vec{x}}_i = \vec{x}_i + \mathbf{C}\ \vec{z} \tag{1}$$

na qual $\vec{z}$ um vetor m-dimensional gerado através de uma distribuição multivariada, e $\mathbf{C} = \text{diag}(\vec{\sigma})$, ou seja, a matriz diagonal obtida através de $\vec{\sigma}^{\mathrm{T}} = [\ \sigma(1)\ \ \ldots\ \ \sigma(m)\ ]$, sendo $\sigma(j)$ a força de mutação em cada componente do vetor m-dimensional $\vec{x}_i$. Geralmente, utiliza-se a distribuição Gaussiana isotrópica ou a distribuição de Cauchy anisotrópica para gerar o vetor m-dimensional $\vec{z}$

Baseado em (Obuchowicz, 2003) e (Thistleton et al., 2006), é proposto aqui a geração de $\vec{z}$ com distribuição q-Gaussiana isotrópica na Eq. 1 através de

$$\vec{z} \sim r\vec{u} \tag{2}$$

sendo $r \sim \mathcal{N}_q(0, 1)$, i.e., uma variável aleatória com distribuição q-Gaussiana e $\vec{u}$ um vetor aleatório com distribuição uniforme. Neste trabalho, um vetor m-dimensional aleatório com distribuição q-Gaussiana isotrópica é denotado por $\vec{z} \sim \mathcal{N}_q^m$.

Propõe-se aqui, também, a auto-adaptação do parâmetro q de cada descendente i da população através de

$$\tilde{q}_i = q_i \exp\left(\tau_a \mathcal{N}(0, 1)\right) \tag{3}$$

sendo que τ_a denota o desvio padrão da distribuição Gaussiana $\mathcal{N}(0, 1)$.

Vale ressaltar que o uso da distribuição q-Gaussiana em AEs não é novo (Moret et al., 2006), (Iwamatsu, 2002). Contudo, nestes algoritmos, como na maioria dos AEs que utilizam mutações com distribuições com caudas longas, as novas soluções são geradas com distribuições anisotrópicas. O uso de distribuições anisotrópicas pode ser útil para problemas com funções de avaliação separáveis, contudo, podem ter baixo desempenho para problemas com funções não-separáveis (Thistleton et al., 2006). Além disso, neste trabalho, auto-adaptação da forma da distribuição de mutações é utilizada, permitindo alterar a distribuição durante o processo de evolução.

O Algoritmo 1 apresenta as idéias propostas neste trabalho. As principais diferenças em relação aos algoritmos de PE Gaussiana, FEP e LEP são que: i) a mutação q-Gaussiana isotrópica é utilizada (passo 6), ao invés das mutações Gaussianas (PE tradicional), de Cauchy (FEP) e de Lévy (GEP); ii) um procedimento para adaptar o parâmetro q é adotado (passo 5).

Utilizado as sugestões obtidas por trabalhos teóricos e empíricos (Beyer and Schwefel, 2002),

são definidos

$$\tau_b = \frac{1}{\sqrt{2m}}, \qquad \tau_c = \frac{1}{\sqrt{2\sqrt{m}}}, \qquad \tau_a = \frac{1}{\sqrt{m}} \tag{4}$$

Algoritmo 1

1: Inicialize a população composta pelos indivíduos ($\vec{x}_i$, $\vec{\sigma}_i$, q_i) para $i = 1, \ldots, \mu$
2: **enquanto** (critério de parada não é satisfeito) **faça**
3: **para** $i \leftarrow 1, \ldots, \mu$ **faça**
4: $\tilde{\sigma}_i(j) = \sigma_i(j) \exp\left(\tau_b \mathcal{N}(0,1) + \tau_c \mathcal{N}(0,1)_j\right)$ para $j = 1, \ldots, m$
5: $\tilde{q}_i = q_i \exp\left(\tau_a \mathcal{N}(0,1)\right)$
6: $\tilde{\vec{x}}_i \leftarrow \vec{x}_i + \mathbf{C}_i\, \mathcal{N}_q^m$, sendo $\mathcal{N}_q^m$ um vetor aleatório gerado a partir da distribuição q-Gaussiana isotrópica com parâmetro $\tilde{q}_i$ e $\mathbf{C}_i = \mathrm{diag}(\tilde{\vec{\sigma}}_i)$
7: **fim para**
8: Compute o $fitness$ dos pais ($\vec{x}_i$, $\vec{\sigma}_i$, q_i) e descendentes ($\tilde{\vec{x}}_i$, $\tilde{\vec{\sigma}}_i$, $\tilde{q}_i$) para $i = 1, \ldots, \mu$
9: Compute a função usada na seleção por torneio ($winning\ function$) (Lee and Yao, 2004) para todos os indivíduos que compõem a população formada por μ pais e μ filhos
10: Faça a seleção por torneio entre os μ pais e μ filhos. A nova população gerada é formada pelos μ indivíduos com os maiores valores da função computada no passo anterior
11: **fim enquanto**

4 Experimentos

Com o objetivo de avaliar o algoritmo proposto, experimentos com duas bases de dados foram realizados. Três algoritmos obtidos pela alteração do método de ajuste do parâmetro q são comparados na tarefa de otimização do vetor de pesos de um *Perceptron* Multicamadas (MLP) do tipo $feed-forward$ utilizado para a classificação dos dados de cada base. Os algoritmos são

- Algoritmo GEP: uso da mutação Gaussiana. O Algoritmo GEP é obtido pela troca da linha 5 do Algoritmo 1 por $\tilde{q}_i = 1$, i.e, a distribuição q-Gaussiana utiliza um parâmetro fixo $q = 1$ para todos os indivíduos e, assim, reproduz a distribuição Gaussiana;
- Algoritmo ICEP: uso da mutação isotrópica de Cauchy. O Algoritmo ICEP é obtido pela troca da linha 5 do Algoritmo 1 por $\tilde{q}_i = 2$;
- Algoritmo qGEP: uso de um parâmetro q para cada indivíduo, i.e., o Algoritmo 1 descrito na última seção é utilizado

As duas bases de dados utilizadas neste trabalho são descritas na Seção 4.1. A descrição dos experimentos realizados encontra-se na Seção 4.2, enquanto que os resultados experimentais são apresentados e analisados na Seção 4.3.

4.1 Bases de Dados Utilizadas

As bases de dados utilizadas neste experimento foram obtidas no Repositório de Aprendizado de Máquina da *University of California* - Irvine (Fogel et al., 1995). A primeira é a base de dados *Wisconsin Breast Cancer*, contendo 699 exemplos. Para cada exemplo, nove atributos referentes a exames histopatológicos obtidos em biopsia estão presentes. Em (Fogel et al., 1995), um MLP do tipo $feed-forward$ com pesos otimizados por um algoritmo de PE com mutações Gaussianas foi utilizado para diagnosticar dados desta base de dados entre duas classes (maligno e benigno). Foram utilizados os 683 exemplos completos, ou seja, com todos os atributos presentes, sendo 400 deles utilizados na fase de treinamento e os 283 restantes para a fase de testes. Utilizando um MLP com uma única camada oculta com apenas 2 neurônios e população com 500 indivíduos evoluída por 400 gerações, obteve-se uma média de 98,05% de acerto no conjunto de teste, o que foi melhor que os resultados encontrados na literatura.

A segunda base de dados é a *Pima Indians Diabetes Database*, consistindo de 768 exemplos com 8 atributos obtidos de exames clínicos e informações sobre índias de uma determinada tribo norte americana. Em (Smith et al., 1988), os exemplos desta base de dados foram separados em um conjunto de treinamento com 576 exemplos e outro de teste com 192 exemplos. Em seguida, o algoritmo ADAP foi treinado para indicar a presença ou ausência de diabetes no exemplo dado, sendo que os testes indicaram uma média de 76% de acerto.

4.2 Descrição dos Experimentos

Nos experimentos realizados, cada indivíduo da população é um vetor de números reais contendo os valores dos pesos sinápticos do MLP, que tem uma arquitetura fixa. O MLP possui uma única camada oculta e neurônios com ativação dada pela função degrau. Em cada execução de um algoritmo, os componentes de cada indivíduo (solução) da população inicial são gerados aleatoriamente com distribuição uniforme no intervalo [-0,5:0,5], ou seja, os pesos iniciais de cada MLP são aleatórios. Para cada indivíduo gerado durante a evolução, é calculado o $fitness$ correspondente, que neste caso é dado pelo erro de classificação para os dados de treinamento do MLP gerado. Neste trabalho, utilizou-se a mesma divisão (com o mesmo número de exemplos) das bases de dados entre conjuntos de treinamento e teste apresentados em (Fogel et al., 1995) para a Base *Breast Cancer* e (Smith et al., 1988) para a Base *Pima*, relatados na seção anterior. Para a Base *Breast Cancer*, utilizou-se também a mesma arquitetura do MLP que obteve os melhores resultados em (Fogel et al., 1995), ou seja, com uma camada oculta com 2 neurônios. Desta forma, o número de componentes do vetor de cada indiví-

duo (i.e., o número de pesos do MLP) é $m = 23$. Já para a Base *Pima*, utilizou-se MLPs com uma camada oculta com 8 neurônios (mesmo número de atributos), fazendo com que m seja igual a 81.

Cada um dos algoritmos utilizados foi executado 30 vezes (com 30 diferentes sementes aleatórias) para cada uma das bases de dados. A população teve tamanho de $\mu = 100$ indivíduos, e cada indivíduo foi comparado com 10 outros durante a seleção por torneio. Na geração inicial, o parâmetro $\sigma_i(j)$ foi dado por $0,1\sqrt{m}$ e o parâmetro da distribuição q-Gaussiana para o Algoritmo qGEP foi igual a 1, i.e, a distribuição de mutações inicial reproduz a distribuição Gaussiana. Ainda no Algoritmo qGEP, os valores mínimo e máximo de q foram, respectivamente, 0,9 e 2,5.

Nos experimentos apresentados aqui, a comparação do desempenho dos algoritmos foi realizada para duas condições do problema. Na primeira, considerou-se o problema de classificação padrão, no qual não ocorrem mudanças durante o processo de evolução, i.e., otimização estacionária. Na segunda, o poder de adaptação frente a mudanças no problema foi testado, i.e., otimização não-estacionária. Nesta condição, mudou-se o problema de classificação após η gerações do processo de evolução com o intuito de verificar a capacidade de adaptação a mudanças de cada algoritmo. Nos problemas do mundo real, a função mapeada pela RNA pode ser modificada por diversos aspectos, como mudanças no ambiente, definição de novos objetivos e falhas. A busca por novas soluções baseada naquelas já existentes, quando possível, pode representar uma importante redução no tempo de computação. Para estes experimentos, utilizou-se, para gerar as alterações no problema, o Gerador de Problemas Dinâmicos Contínuos descrito em (Tinós and Yang, 2007), no qual a mudança do ambiente é simulada pela rotação dos indivíduos da população após η gerações do processo de evolução. O parâmetro ρ controla o ângulo de rotação dos indivíduos, sendo que o aumento da valor de ρ geralmente implica em uma maior severidade da mudança.

Nos experimentos aqui apresentados, considerou-se três valores de ρ, correspondentes a problemas com nenhuma mudança ($\rho = 0,0$), e com severidades média ($\rho = 0,1$), e alta ($\rho = 0,5$). Nos experimentos em que ocorrem mudanças no ambiente, variou-se também o período de ocorrência da mudança (parâmetro η). Os valores foram $\eta = 10$, $\eta = 100$ e $\eta = 300$ correspondentes, respectivamente, a ocorrências de mudanças em uma etapa inicial, média ou final na evolução. Nesta forma, cada algoritmo foi executado 30 vezes para cada uma das combinações dos valores ρ, que controla a severidade da mudança, e η, que define o estágio de ocorrência da mudança.

4.3 Resultados

Os resultados experimentais médios, sobre as 30 execuções, do fitness do melhor indivíduo na última geração do algoritmo são apresentados na Tabela 1. Nesta tabela, são apresentados também os valores médios das porcentagens de erros de classificação sobre o conjunto de teste. Entre parêntesis, são apresentadas as comparações estatísticas, utilizando-se o teste de hipóteses com distribuição $t-student$ com 58 graus de liberdade e nível de significância de 0,05, para os valores de fitness do melhor indivíduo na última geração do algoritmo. O resultado referente a comparação entre o Alg. X e o Alg. qGEP é apresentado como "+", "−", ou "∼", quando o Alg. qGEP é significantemente melhor, pior, ou estatisticamente equivalente ao Alg. X.

A Figura 1 apresenta os valores médios do $fitness$ do melhor indivíduo de cada geração para as 30 execuções de cada algoritmo para a base de dados *Pima*. Observe que, após η gerações, o problema muda, fazendo com que, em geral, o $fitness$ aumente. Os resultados para $\rho = 0,0$ não são mostrados pois eles são iguais àqueles obtidos até a geração 300 dos experimentos com $\eta = 300$. Já a Figura 2 mostra os valores médios do parâmetro q da distribuição q-Gaussiana para os melhores indivíduos de cada geração. Observe que $q = 1$ para o Algoritmo GEP e $q = 2$ para o Algoritmo ICEP.

Quando os resultados dos algoritmos GEP e ICEP são comparados, pode-se observar que o Algoritmo ICEP geralmente apresenta os melhores resultados de $fitness$ para a Base *Breast Cancer*. Para esta base de dados, a distribuição de Cauchy isotrópica (Algoritmo ICEP), que possui uma cauda mais longa quando comparada com a distribuição Gaussiana, propiciou o aparecimento de saltos maiores eventualmente gerados pelo operador de mutação, principalmente na etapa inicial e após a mudança do problema. Desta forma, conseguiu-se alcançar valores menores de erro para o conjunto de treinamento, e consequentemente, de $fitness$, em um tempo menor. Observe ainda que, para esta base de dados, os resultados de $fitness$ do Algoritmo ICEP são geralmente melhores do que para o Algoritmo qGEP, indicando que o valor alto de q mantido durante toda a evolução foi mais interessante que o valor de q auto-adaptável. Contudo observa-se que o valor de q aumenta ao longo da evolução para o Algoritmo qGEP, tendo um salto (especialmente para valores altos de ρ e η) após a mudança no problema. Isto ocorre porque indivíduos com valores mais altos de q foram sendo selecionados pelo algoritmo. Desta forma, o Algoritmo qGEP consegue se adaptar para a superfície de $fitness$, buscando valores de q mais apropriados. Para esta base de

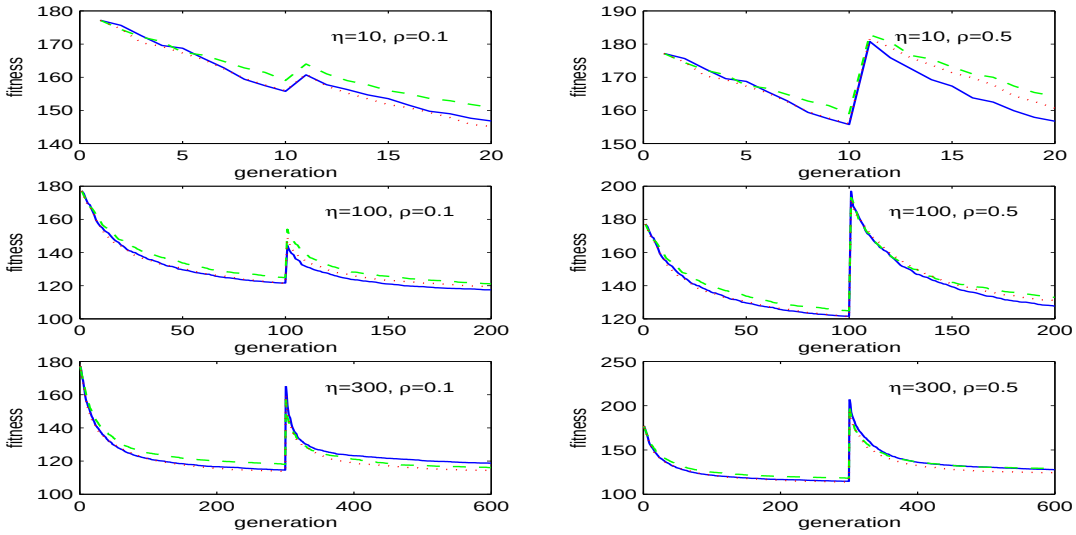

Figura 1: Fitness médio do melhor indivíduo de cada geração para os experimentos com a base de dados *Pima*. Algoritmos: GEP (linha sólida azul), ICEP (linha tracejada verde), e qGEP (linha pontilhada vermelha)

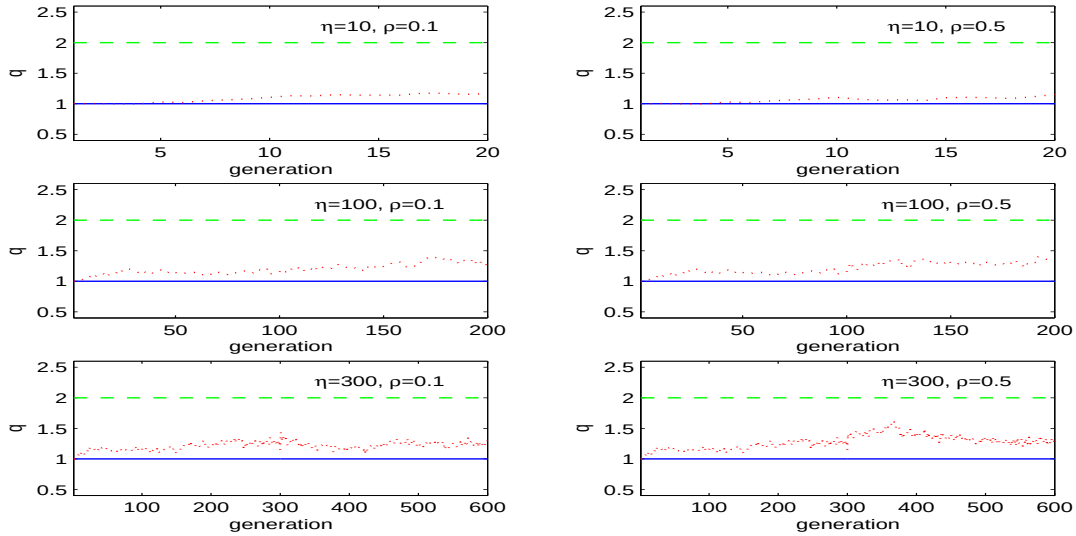

Figura 2: Valor médio do parâmetro q do melhor indivíduo de cada geração para os experimentos com a base de dados *Pima*.

dados, o Algoritmo qGEP apresentou resultados melhores que o Algoritmo GEP, pois conseguiu produzir saltos eventualmente maiores através do operador de mutação, o que se mostrou benéfico para este problema.

Já quando os resultados dos algoritmos GEP e ICEP para a Base *Pima* são comparados, pode-se observar que o Algoritmo GEP geralmente apresenta melhores resultados de $fitness$ quando comparado ao Algoritmo ICEP. Neste problema, saltos maiores gerados pela distribuição de Cauchy isotrópica (Algoritmo ICEP) não se mostraram benéficos, indicando que a busca local realizada na vizinhança foi mais apropriada. No Algoritmo GEP, mais indivíduos são gerados na vizinhança próxima das soluções atuais, o que se mostrou benéfico para este problema. No entanto, para este problema, o Algoritmo qGEP apresentou em geral os melhores resultados de $fitness$, indicando que valores de q maiores que 1 foram benéficos.

Para as duas bases, os resultados de classificação para o conjunto de testes foram próximos àqueles relatados na Seção 4.1. No caso da Base *Pima*, pode se observar que o número de avaliações de $fitness$ realizados pelos algorimos investigados aqui é bem menor. Observe também na Tabela 1, que resultados melhores de $fitness$, computado sobre o conjunto de treinamento, não necessariamente implicam em valores menores de erros de classificação para o conjunto de testes.

Tabela 1: Fitness do melhor indivíduo na última geração para cada algoritmo e % de erros de classificação do conjunto de teste.

Medida	Ambiente		Algoritmo		
	η	ρ	GEP	ICEP	qGEP
	300	0,0	9,5 (∼)	9,1 (−)	9,5
Fitness	10	0,1	12,1 (∼)	12,7 (+)	12,0
(*Breast*	10	0,5	20,3 (∼)	18,3 (∼)	17,8
Cancer)	100	0,1	9,8 (∼)	9,2 (−)	9,7
	100	0,5	21,6 (∼)	11,0 (∼)	12,2
	300	0,1	15,8 (∼)	9,5 (∼)	9,6
	300	0,5	25,4 (∼)	12,0 (∼)	16,4
	300	0,0	2,07	2,26	2,06
% erros	10	0,1	2,41	2,56	2,43
(*Breast*	10	0,5	3,51	3,38	2,96
Cancer)	100	0,1	2,05	2,28	2,25
	100	0,5	3,65	2,46	2,99
	300	0,1	2,98	2,23	2,51
	300	0,5	4,58	2,94	3,56
	300	0,0	114,5 (∼)	118,0 (+)	113,7
	10	0,1	146,8 (∼)	150,9 (+)	145,2
Fitness	10	0,5	156,7 (−)	164,3 (+)	160,7
(*Pima*)	100	0,1	117,4 (∼)	121,0 (∼)	119,2
	100	0,5	127,5 (∼)	132,8 (∼)	130,9
	300	0,1	118,7 (+)	116,0 (∼)	114,3
	300	0,5	127,6 (∼)	129,0 (+)	124,2
	300	0,0	22,67	22,74	22,90
	10	0,1	26,53	26,75	25,92
% erros	10	0,5	27,34	30,80	29,53
(*Pima*)	100	0,1	22,59	23,39	23,52
	100	0,5	24,57	25,43	25,03
	300	0,1	24,17	22,71	23,82
	300	0,5	25,83	25,00	24,08

5 Conclusões

Neste trabalho, o uso da distribuição q-Gaussiana isotrópica com auto-adaptação do parâmetro q é proposto para gerar mutações em PE utilizada para otimizar os pesos de RNAs. Mudando o parâmetro q, muda-se a forma da distribuição q-Gaussiana, permitindo o controle da distribuição das mutações. Desta forma, a decisão de escolher qual distribuição de mutações utilizar, e.g. uma que permita saltos maiores ou menores, em cada momento da evolução fica a cargo do algoritmo, e não do programador. Os experimentos realizados indicam que o desempenho do algoritmo de PE fica dependente da escolha da distribuição de mutações. O melhor desempenho de uma distribuição em relação à outra depende do problema a ser otimizado, principalmente das caracterísiticas da superfície de $fitness$. Desta forma, a auto-adaptação da distribuição de mutações durante a evolução é interessante, permitindo buscar melhores distribuições em diferentes etapas da evolução.

Como trabalhos futuros, deve-se aplicar o algoritmo proposto em diversos outros problemas, e.g. em RNAs utilizadas para o controle de robôs móveis. Novos algoritmos baseados nas idéias propostas aqui e com outros métodos de controle do parâmetro q devem ser investigados.

Agradecimentos

O autor agradece à FAPESP (Proc. 04/04289-6) pelo apoio financeiro e ao Prof. Shengxiang Yang da *University of Leicester* pelos comentários a respeito dos algoritmos.

Referências

Beyer, H.-G. and Schwefel, H. S. (2002). Evolution strategies: a comprehensive introduction, *Natural Computing* **1**: 3–52.

Fogel, D. B., Wasson, E. D. and Boughton, E. M. (1995). Evolving neural networks for detecting breast cancer, *Cancer Letters* **96**: 49–53.

Hansen, N., Gemperle, F., Auger, A. and Koumoutsakos, P. (2006). When do heavy-tail distributions help?, *9th Int. Conf. on Parallel Problem Solving from Nature (PPSN IX)* **4193 LNCS**: 62 – 71.

Iwamatsu, M. (2002). Generalized evolutionary programming with levy-type mutation, *Computer Physics Com.* **147**(1-2): 729 – 732.

Lee, C.-Y. and Yao, X. (2004). Evolutionary programming using mutations based on the levy probability distribution, *IEEE Transactions on Evolutionary Computation* **8**(1): 1 – 13.

Moret, M. A., Pascutti, P. G., Bisch, P. M., Mundim, M. S. P. and Mundim, K. C. (2006). Classical and quantum conformational analysis using generalized genetic algorithm, *Physica A: Statistical Mechanics and its Applications* **363**(2): 260 – 268.

Obuchowicz, A. (2003). Multidimensional mutations in evolutionary algorithms based on real-valued representation, *Int. Journal of Systems Science* **34**(7): 469 – 483.

Smith, J. W., Everhart, J. E., Dickson, W. C. and Knowler, W. C. (1988). Using the adap learning algorithm to forecast the onset of diabetes mellitus, *Proc. of the Symp. on Computer Applications and Medical Care*, pp. 261–265.

Thistleton, W., Marsh, J. A., Nelson, K. and Tsallis, C. (2006). Generalized Box-Muller method for generating q-Gaussian random deviates, *ArXiv Condensed Matter e-prints, cond-mat/0605570* .

Tinós, R. and Yang, S. (2007). Continuous dynamic problem generators for evolutionary algorithms, *In The Proc. of the 2007 IEEE Cong. on Evolutionary Computation*.

Yao, X. (1999). Evolving artificial neural networks, *Proc. of the IEEE* **87**(9): 1423–1447.