1. Evolução Diferencial, Introdução

Evolução Diferencial, conhecido como ED (ou do inglês DE, *Differential Evolution*) é uma técnica de otimização estocástica de funções não-lineares e não-diferenciável no espaço contínuo baseada em populações de indivíduos, desenvolvida pelo Rainer Storn e Kenneth Price em 1995, para tentar resolver problemas de ajuste polinomial de Chebychev [5]. Tal técnica se mostrou não só capaz de resolver os problemas de ajuste polinomial de Chebychev, como também para várias outras funções de testes, mesmo que possuam restrições do espaço de busca [4].

Por ter suas raízes na Computação Evolucionária, o algoritmo compartilha as suas inspirações com os algoritmos evolucionários, tais como, por exemplo, ser baseado na teoria da evolução e permitir que os indivíduos mais aptos tenham maiores chances de sobrevivência sobre os menos aptos. Embora seja semelhante aos outros algoritmos evolucionários, como Algoritmos Genéticos, o ED possui duas grandes diferenças que fazem com que ele seja considerado um novo algoritmo:

- O operador de mutação é aplicado primeiro para a criação de um vetor experimental, que depois será usado no operador de cruzamento para a criação da nova população da próxima geração

- A aleatoriedade da mutação não é feita a partir de um número aleatório de distribuição normal, e sim de dois indivíduos da própria população escolhidos aleatoriamente.

Já que o operador de mutação utiliza dos indivíduos da própria população, isso faz com que a inicialização dos indivíduos seja também um ponto chave para o sucesso, pois caso os indivíduos sejam inicializados em um único ponto (por exemplo), isso comprometerá a convergência do algoritmo, podendo ficar preso em locais sub-ótimos [6].

Devido a essas diferenças a técnica consegue evoluir muito mais rápido que outras, fazendo inclusive menos chamadas a função de *fitness*, podendo ser uma grande vantagem quando for utilizado de funções de avaliação muito custosas [1].

2. Histórico

O algoritmo de Evolução Diferencial originou-se a partir do algoritmo *Genetic Annealing*, desenvolvido por Kenneth Price em outubro de 1994. Baseado em populações, o *Genetic Annealing* é um algoritmo de otimização que implementa um critério de resfriamento (*annealing*) por limiares (*thresholds*). Após a publicação do algoritmo, Rainer Storn entrou em contato com Kenneth Price para discutir sobre o uso do algoritmo para solução do problema de ajuste do polinômio Chebyshev. Durante o estudo para conseguir solucionar o problema proposto por Rainer, acabaram criando um novo algoritmo, o qual eles chamaram de Evolução Diferencial.

Em 1995 Rainer Storn e Kenneth Price apresentaram alguns dos seus primeiros resultados em no relatório técnico TR-95-012 "*Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*" para o Instituto Internacional de Ciência da Computação (*International Computer Science Institute*, ICSI) [1].

O sucesso dos experimentos permitiram que, em maio de 1996, Rainer e Ken entrassem com o algoritmo de ED no Primeiro Concurso Internacional de Otimização Evolucionária (*First International Contest on Evolutionary Optimization*) que foi realizado em conjunto com a Conferência Internacional do IEEE em Computação Evolucionária (*IEEE International Conference on Evolutionary Computation*), ambos realizado em Nagoya, no Japão. Durante esse concurso, o algoritmo de Rainer e Ken finalizou em terceiro lugar, perdendo para dois algoritmos considerados "não tão versáteis" quanto o ED, pois o primeiro foi otimizado para trabalhar bem as funções do concurso enquanto que o segundo envolvia demasiados parâmetros de configuração [4].

Diante disso, Rainer e Ken percebendo o potencial do ED escreveram um dos principais artigos do algoritmo para o DDJ (*Dr. Dobb's Journal of Software Tools*), que foi publicado em abril de 1997 com o título "*Differential Evolution - A Simple Evolution Strategy for Fast Optimization*" [2]. Esse artigo foi muito bem aceito na comunidade, que fez com que o algoritmo ganhasse amplo conceito internacional.

Em dezembro de 1997, muitos pesquisadores em otimização conheceram o potencial do ED após ler o artigo "*Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*", de Rainer e Ken, onde traz diversas e extensivas evidências empíricas sobre a robustez e desempenho em uma grande variedade de funções de teste. Nesse mesmo momento, Rainer criou um site (http://www.icsi.berkeley.edu/~storn/code.html) contendo várias informações sobre ED, links para aplicações e atualizações do algoritmo [4].

3. Motivação

Problemas de otimização permeiam a ciência e engenharia. Um bom otimizador deve ser fácil de usar, robusto para conseguir convergir para uma solução ótima e não deve levar um tempo excessivo para encontrar uma solução. Devido a isso que pesquisadores vem estudando vários métodos de otimização e como aperfeiçoá-los para os problemas do mundo real. De acordo com as características do problema a ser otimizado, os métodos de otimização podem ser classificados em lineares e não-lineares, sendo o último subdividido em determinísticos que geralmente utilizam o cálculo de derivadas e não-determinístico que utilizam de alguma aleatoriedade na busca. Vale ressaltar que grande parte dos problemas reais de otimização são não-lineares e não-determinísticos, onde os mais difíceis são os não-determinísticos.

A Evolução Diferencial nasceu das modificações de Storn e Kenneth do algoritmo *Genetic Annealing* criado inicialmente por Rainer. Tais modificações foram produzidas na tentativa de resolver o problema de ajuste polinomial de Chebychev. No desenvolvimento do ED, Rainer e Ken levaram em consideração que o espaço de busca do problema é não-linear, não-diferenciável, contínuos e com restrições. Porém, mesmo possuindo essas características, seus criadores consideram que em espaços de busca "bem comportados" poucos pontos localizados em cada uma de suas dimensões são possíveis de expressar a sua topologia. Ou seja, dependendo da inicialização dos seus indivíduos, eles provêm por si só uma boa representação da função *fitness* [6].

Visto que a característica aleatória (estocástica) é por base utilizando os próprios indivíduos da população, isso faz com que o algoritmo convirja mais rapidamente, efetuando menos chamadas para a função *fitness*. Isso representa um importante ponto, pois se a função *fitness* for demasiadamente cara computacionalmente falando essa técnica se torna promissora nesse âmbito.

Outra característica importante é a menor quantidade de parâmetros, mostrando que a técnica é bastante flexível aos problemas impostos a ela. São necessários apenas três parâmetros: número de indivíduos, fator de escala e probabilidade de recombinação. Existem alguns estudos que provêm à auto-adaptação desses parâmetros ao problema a ser utilizado [7].

4. Resumo da técnica

Em ED cada indivíduo é inicializado uniformemente pelas dimensões do espaço de busca. Os operadores utilizados são: mutação, cruzamento e seleção. Eles são executados nessa mesma ordem para cada um dos filhos da população, sendo assim representados:

1. O operador de mutação cria vetores experimentais utilizando a adição da diferença de dois indivíduos aleatórios a um terceiro indivíduo.
2. O operador de cruzamento escolhe quais as componentes do vetor experimental serão escolhidas para afetar o indivíduo escolhido.

3. O operador de seleção escolherá o melhor indivíduo para a próxima geração. Se o indivíduo gerado for melhor que o pai, ele será escolhido para a próxima geração, caso contrário o pai é que irá permanecer na próxima geração.

Quando o critério de parada for estabelecido, o algoritmo retorna o indivíduo que possui melhor *fitness* da população.

Para ilustrar melhor esse processo, um fluxograma (figura 4.1) foi criado com todo o comportamento do algoritmo.

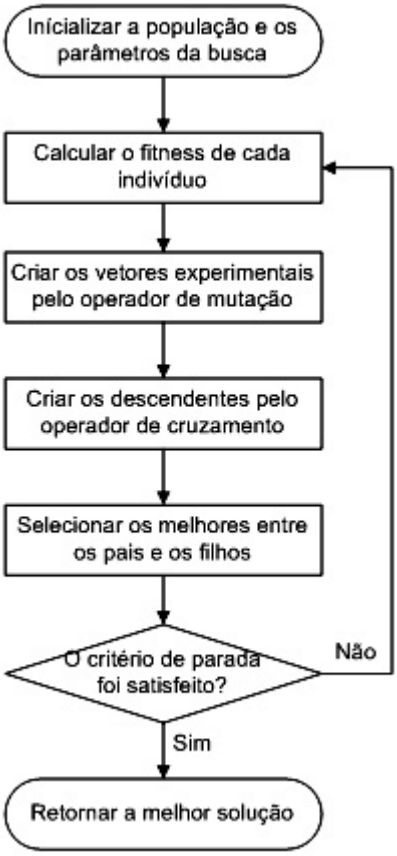

Figura 4.1 - Fluxograma do algoritmo de Evolução Diferencial.

4.1. Representação individual

De acordo com os outros algoritmos evolucionários, cada posição de um indivíduo da população representa uma possível solução no espaço de busca. Isso permite ao ED obter uma boa representação de todo o espaço de busca apenas com as posições iniciais dos indivíduos. Cada indivíduo é codificado em valores reais em forma de um vetor de posição, onde cada componente desse vetor representa uma das variáveis da solução.

O uso de números do tipo ponto flutuantes na representação das componentes dos valores codificados no ED traz vantagens como uma maior precisão dos resultados, sem a perda devido à representação e uma melhor capacidade de representação do domínio do problema. Essas vantagens fazem com que a representação ponto flutuante seja uma abordagem bastante atrativa.

4.2. Avaliação de sucesso

A avaliação dos indivíduos é baseada em uma função de avaliação *fitness*. Essa função deve ser capaz de identificar os indivíduos bons dos ruins, podendo inclusive incorporar restrições no espaço de busca.

As restrições podem ser consideradas como valores das componentes que não são factíveis nos problemas reais, sejam por restrições físicas como, por exemplo, dimensões negativas, ou seja, por restrições de preferências nas soluções, como lucro maior que um determinado valor, ou perdas abaixo de um limiar, etc.

Para descrever o ambiente de busca a função de avaliação pode ser representada por:

1. Uma função matemática que defina o problema ou que se aproxime do mesmo, caso seja conhecida;
2. Uma heurística que indica uma possível direção da solução do problema, sendo um guia para a solução;
3. Ou, pela simulação da solução proposta utilizando dos parâmetros do vetor em simulador ou da aplicação direta no problema real, avaliando assim seu comportamento.

4.3. Inicialização

Um dos pontos mais sensíveis do algoritmo de Evolução Diferencial é a inicialização dos indivíduos pelo espaço de busca. Para o ED funcionar satisfatoriamente, os indivíduos devem ser bem distribuídos pelo espaço de busca, podendo ser utilizado da distribuição aleatória e uniforme [4].

Caso a inicialização seja realizada de forma equivocada como, por exemplo, definida muito longe dos limites do espaço de busca, esta pode comprometer a qualidade da solução encontrada. Devido a isso, o algoritmo nem sempre irá convergir para o melhor ponto da função *fitness* [6].

4.4. Operadores

Nos algoritmos evolucionários são utilizados comumente os operadores de mutação e cruzamento. Quando ambos deles são utilizados, geralmente a ordem é a seguinte: primeiro são gerados os filhos com o operador de cruzamento e depois aplicado o operador de mutação sobre os filhos gerados para por fim o operador de seleção escolher os indivíduos que farão parte da próxima geração.

No caso do ED todos os operadores citados são utilizados, o que basicamente diferencia ED das outras técnicas é o operador de mutação que é aplicado primeiro em um vetor experimental para depois realizar o cruzamento e na produção da sua aleatoriedade que não é regido por um número aleatório de distribuição uniforme [6].

4.4.1. Operador de mutação

Este operador produz um vetor experimental $u_i(t)$ a partir de outros três indivíduos da população. O primeiro passo consiste na escolha do indivíduo destino $x_{i1}(t)$ como sendo um indivíduo da população diferente do atual. Os critérios dessa escolha podem ser a aleatoriedade, o melhor indivíduo (que possui o melhor valor de *fitness*) ou o pior indivíduo (que possui o pior valor de *fitness*), etc.

Uma vez que o indivíduo destino tenha sido escolhido, escolhem-se mais dois outros indivíduos diferentes $x_{i2}(t)$ e $x_{i3}(t)$ na população. Esses serão exclusivamente utilizados para o cálculo do vetor experimental, que é realizado como sendo a diferença vetorial desses indivíduos escolhidos (x_{i2} e x_{i3}), amplificada pelo fator de escala (β) sendo somada ao vetor da posição do indivíduo destino (x_{i1}).

Representando matematicamente, temos:

$$u_i(t) = x_{i1}(t) + \beta(x_{i2}(t) - x_{i3}(t))$$

Figura 4.2 - Fórmula matemática do operador de mutação.

onde β é o fator de escala, responsável por controlar a amplificação da variação diferencial.

A figura 4.3 mostra o funcionamento do operador de mutação para o cálculo do vetor experimental bem como o cálculo vetorial dos demais vetores.

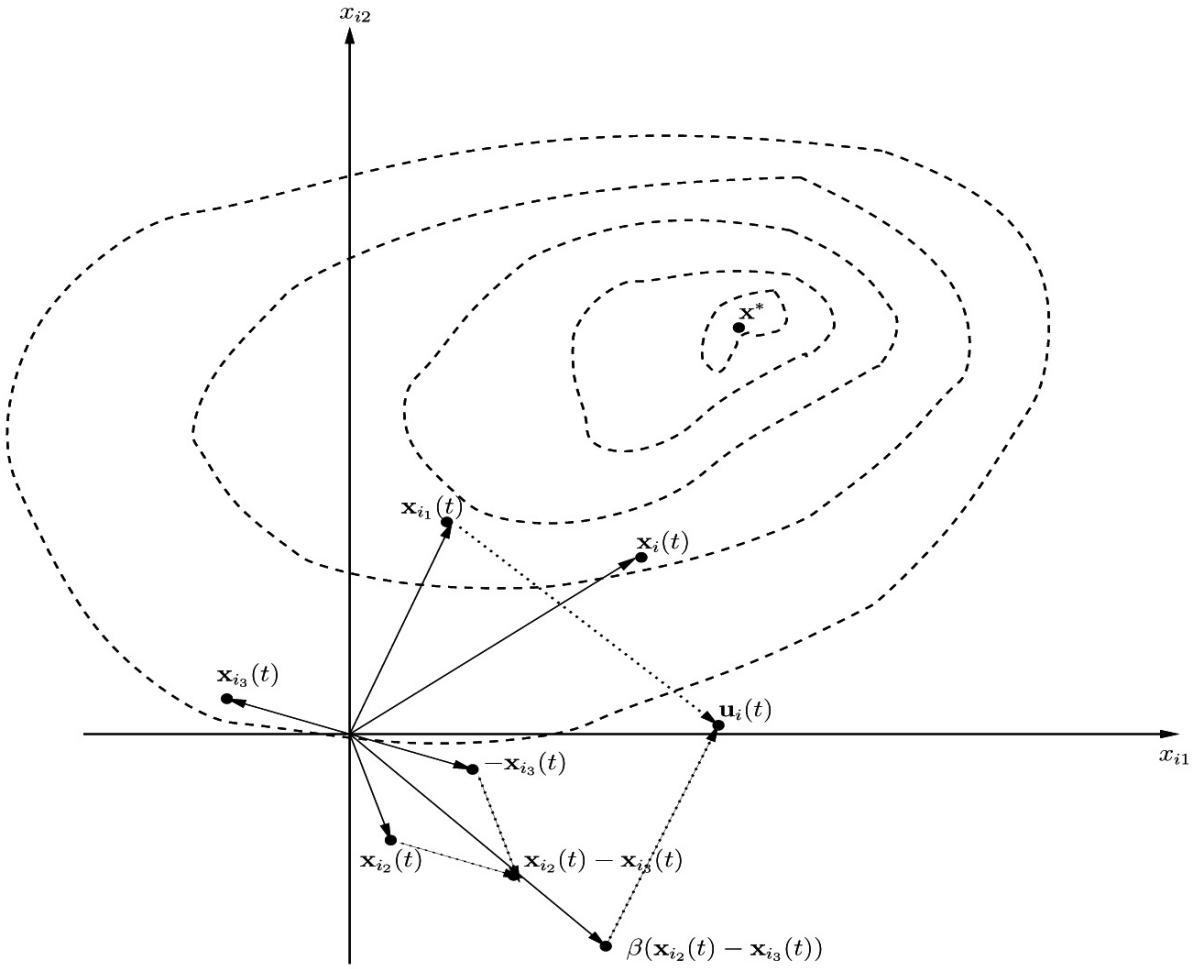

Figura 4.3 - Operador de mutação.

4.4.2. Operador de cruzamento

O operador de cruzamento proporciona a diversidade no ED, que é realizada através da combinação entre o vetor experimental $u_i(t)$ e o vetor pai $x_i(t)$, tendo como resultado um vetor filho $x'_i(t)$. A recombinação é feita através da seguinte equação:

$$x'_{ij} = \begin{cases} u_{ij}(t), & Se\ j \in J \\ x_{ij}(t), & Caso\ contrário \end{cases}$$

Figura 4.4 - Fórmula de seleção do operador de cruzamento.

Onde x_{ij} é referente a j-ésima componente do vetor $x_i(t)$, e J é o conjunto de pontos de cruzamentos, ou seja, os índices referentes às posições do vetor que sofrerão perturbação.

Para que a componente seja trocada, um número aleatório é sorteado e ele é comparado ao parâmetro de probabilidade de recombinação (***Pr***). Caso o número seja menor que o parâmetro de recombinação, a componente do vetor experimental será utilizada. Caso contrário, a própria componente do pai sem modificações será utilizada.

Para a escolha dessas componentes, existem duas técnicas comumente utilizadas, são elas:

- **Cruzamento Binomial:** Onde os pontos de cruzamento são selecionados aleatoriamente dentro da dimensão do problema. Ou seja, cada uma das dimensões gera um número aleatório e compara o número ao parâmetro de recombinação (***Pr***).

- **Cruzamento Exponencial:** Os pontos de cruzamento são selecionados seqüencialmente dentro das dimensões do problema, em forma de uma lista circular. Ou seja, para cada dimensão é sorteado um número aleatório, que caso seja menor que o parâmetro de recombinação (***Pr***) então todas as componentes a partir daquela em diante serão trocadas, até que o número sorteado seja maior que o parâmetro de recombinação.

A figura 4.5 mostra o funcionamento do operador de cruzamento para o cálculo vetorial do filho.

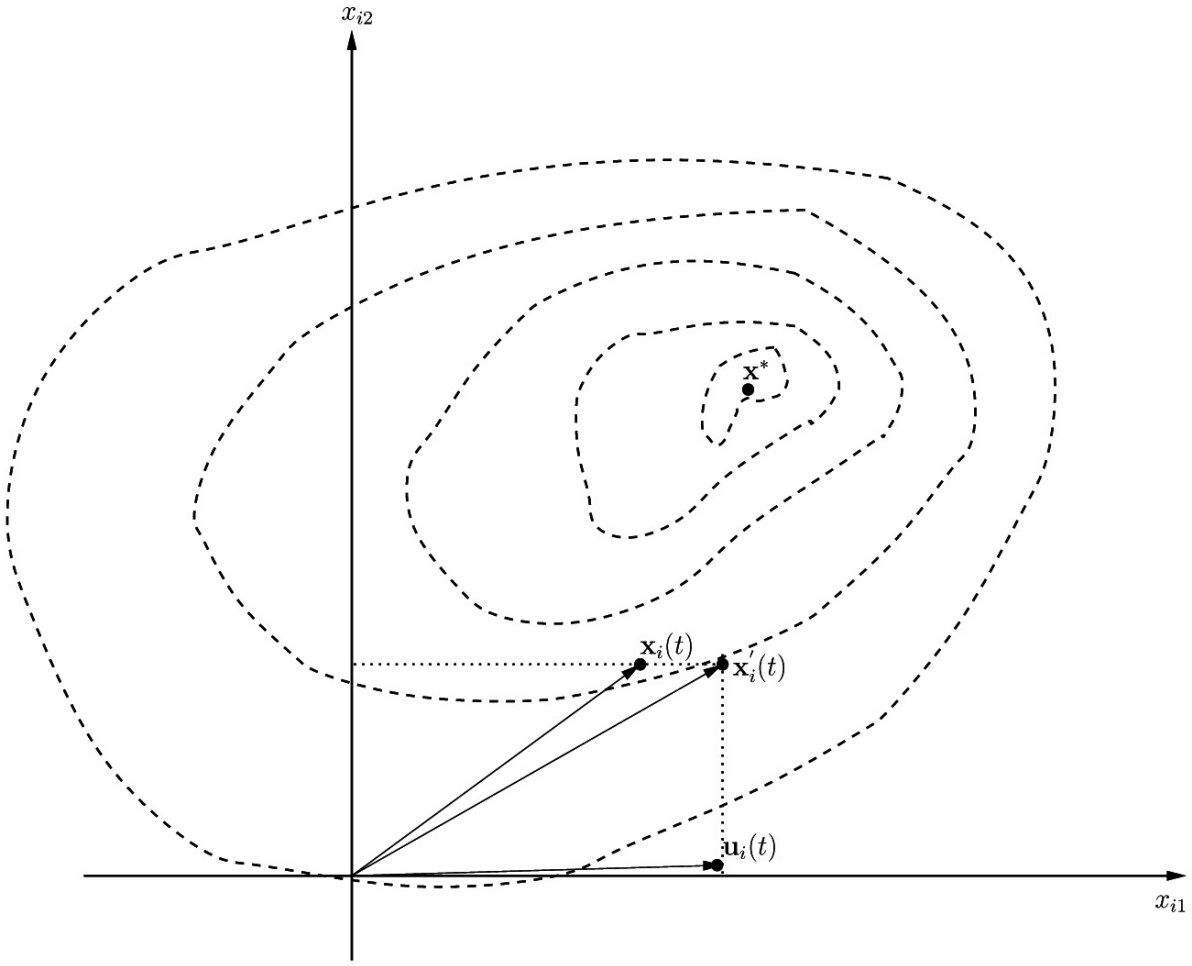

Figura 4.5 - Operador de cruzamento.

4.4.3. Operador de seleção

O operador de seleção é o operador mais simples da técnica, sendo utilizado apenas para escolha dos indivíduos que farão parte da próxima geração. A escolha é realizada através da avaliação do *fitness* do vetor escolhido com o vetor experimental:

- Caso o vetor experimental possuir *fitness* melhor que o do vetor escolhido, então o vetor escolhido será substituído pelo vetor experimental na próxima geração.

- Caso o vetor escolhido possua *fitness* melhor que o vetor experimental, então o vetor escolhido permanece na população para a próxima geração.

4.5. Parâmetros

Uma grande vantagem do ED é a baixa utilização de parâmetros, tornando o algoritmo bastante versátil [4]. Além do algoritmo possuir poucos parâmetros, boa parte deles não são sensíveis para a busca, podendo inclusive serem auto-adaptáveis [6].

4.5.1. Número de indivíduos na população

Esse parâmetro a quantidade de indivíduos na população que afeta diretamente a capacidade de exploração do algoritmo. Valores pequenos para esse parâmetro afetam a capacidade de exploração do algoritmo, todavia valores grandes aumentam a complexidade computacional.

Alguns estudos empíricos mostram que um bom valor inicial é dez vezes (10x) da quantidade de dimensões do problema [6]. Porém, para um melhor desempenho, esse número deve ser analisado para cada problema. Por exemplo, problemas com o espaço de busca relativamente comportado permite que esse número seja reduzido sem afetar a capacidade de exploração. O oposto também é válido, espaços de busca relativamente complexos exigem que mais indivíduos sejam definidos para uma melhor capacidade de exploração.

4.5.2. Fator de escala (β)

O fator de escala controla a amplificação da diferença vetorial no operador de mutação. Valores pequenos demais comprometem a convergência do algoritmo, já valores grandes demais facilitam a exploração do espaço de busca, porém o algoritmo pode ficar “saltando” do ponto ótimo.

Esse parâmetro deve ser grande o suficiente para permitir maior capacidade de exploração, como também deve ser pequeno o suficiente para garantir a exploração local. Alguns estudos empíricos mostram que um bom valor inicial para esse parâmetro seria 0,5.

Um fato importante é levar em consideração que esse parâmetro está diretamente ligado ao número de indivíduos da população. Uma população com muitos indivíduos faz com que o espaço de busca fique mais "denso", evitando assim que o fator de escala seja grande. Caso a densidade dos indivíduos seja pequena, o fator de escala deve ser grande o suficiente para facilitar a busca [6].

Uma estratégia interessante de ajuste desse parâmetro seria torná-lo auto-adaptável, iniciando o mesmo com valores altos, garantindo assim uma boa exploração global e na medida em que as gerações avançassem, esse valor iria decaindo linearmente (por exemplo) facilitando assim a exploração local ao final das iterações de busca.

4.5.3. Probabilidade de recombinação (Pr)

A probabilidade de recombinação influencia diretamente na diversidade do algoritmo controlando a quantidade de elementos que irão sofrer alterações. Valores altos demais aumentam a rapidez na

convergência, porém, comprometendo a qualidade da busca. Valores baixos demais aumentam a robustez da busca, porém, aumentam também o tempo da busca e por conseqüência a quantidade de chamadas a função *fitness*.

4.6. Critérios de parada

Semelhante a outras abordagens evolucionárias, os critérios de parada para ED compartilham as características de outras técnicas, tais como:

- **Número máximo de gerações for atingido:** Para a execução do algoritmo quando um determinado número de gerações for atingido. Esse critério não é um critério satisfatório, pois se esse valor for muito pequeno, poderá comprometer a qualidade da solução buscada por não haver tempo suficiente para o algoritmo encontrá-la.

- **Não for observada nenhuma melhoria no melhor indivíduo em um número consecutivo de gerações:** Para a execução quando após uma quantidade determinada de consecutivas gerações o melhor indivíduo não apresentar nenhuma melhoria, o que representa alguma estagnação.

- **Não for observada nenhuma alteração na população em um número de gerações:** Para a execução quando após uma quantidade determinada de consecutivas gerações a média de melhoria da população for muito pequena.

- **Uma solução aceitável for encontrada:** Para a execução quando uma determinada solução aceitável seja encontrada, independente se essa é a melhor de todas ou não. Essa abordagem é bastante útil para casos onde o tempo é um fator crucial.

- **O delta da função objetivo for aproximadamente zero:** Para a execução quando o percentual de variação da diferença do melhor indivíduo da geração anterior com o indivíduo da geração atual seja próximo de zero. Ou seja, caso a melhoria entre gerações seja abaixo de um limiar, o algoritmo irá parar. Essa abordagem pode fazer com que o algoritmo pare antes de encontrar o ótimo global.

5. Variações

Os algoritmos de Evolução Diferencial são conhecidos na literatura com a notação **ED/*x*/*y*/*z***, onde:

- ***x*** indica o método de seleção do indivíduo de destino, podendo ser, por exemplo, aleatório (*rand*), ou o melhor (*best*), ou do aleatório para o melhor (*rand-to-best*), ou do atual para o melhor (*current-to-best*) ou outros métodos;

- ***y*** indica o número de vetores diferenciais utilizados, podendo utilizar de 1 até *n* vetores;

- ***z*** indica o método de cruzamento utilizado, podendo ser binomial (*bin*) ou exponencial (*exp*).

As variações do método de seleção do indivíduo de destino são:

- **Aleatório (*rand*):** O indivíduo de destino é escolhido aleatoriamente dentro dos indivíduos da população.

- **Melhor (best):** O melhor indivíduo da população é escolhido como sendo o indivíduo de destino.

- **Aleatório para o melhor (*rand-to-best*):** Parte do melhor indivíduo é escolhida e a outra parte é escolhida de um indivíduo aleatório. Essa variação introduz um novo parâmetro que

controla quanto de cada um dos indivíduos envolvidos serão utilizados. Geralmente usa-se a estratégia de geração de um número aleatório para saber quanto de cada um será obtido.

- **Atual para o melhor (*current-to-best*):** Nessa estratégia, parte do vetor é calculado utilizando o melhor indivíduo e a outra parte é calculada utilizando o indivíduo atual.

A tabela 5.1 mostra algumas das variações de ED conhecidas:

Tabela 5.1 - Tipos de variações do algoritmo de Evolução Diferencial.

Notação	Método de seleção	Número de vetores	Método de cruzamento
ED/rand/1/bin	Aleatório	1	Binomial
ED/best/1/bin	Melhor indivíduo	1	Binomial
ED/rand/2/bin	Aleatório	2	Binomial
ED/best/2/bin	Melhor indivíduo	2	Binomial
ED/rand-to-best/2/bin	Aleatório para o melhor	2	Binomial
ED/current-to-best/2/bin	Atual para o melhor	2	Binomial
ED/rand/1/exp	Aleatório	1	Exponencial
ED/best/1/exp	Melhor indivíduo	1	Exponencial
ED/rand/2/exp	Aleatório	2	Exponencial
ED/best/2/exp	Melhor indivíduo	2	Exponencial
ED/rand-to-best/2/exp	Aleatório para o melhor	2	Exponencial
ED/current-to-best/2/ exp	Atual para o melhor	2	Exponencial

Estudos empíricos mostram que o ED/*rand*/*1*/*bin* garante uma boa diversidade, enquanto que o ED/*rand-to-best*/*2*/*bin* garante uma boa convergência.

6. Funcionamento

A fim de demonstrar o funcionamento do algoritmo, foi utilizada a função "*peaks*" (figura 6.1) que é não-linear, contínua e multimodal (possui mais de um mínimo ou máximo). Embora seja multimodal, existem pontos da função que são chamados de máximos ou mínimos globais, pois são de todos os "picos" os mais expressivos. A função em questão está sendo representada graficamente pela figura 6.2.

$$f(x_1,x_2) = 3(1-x_1)^2 \cdot \exp\left(x_1^2 + (x_2+1)^2\right) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) \cdot$$
$$\cdot \exp\left(x_1^2 + x_2^2\right) - \frac{1}{3} \cdot \exp\left((x_1+1)^2 + x_2^2\right)$$

Figura 6.1 - Fórmula matemática da função "*peaks*".

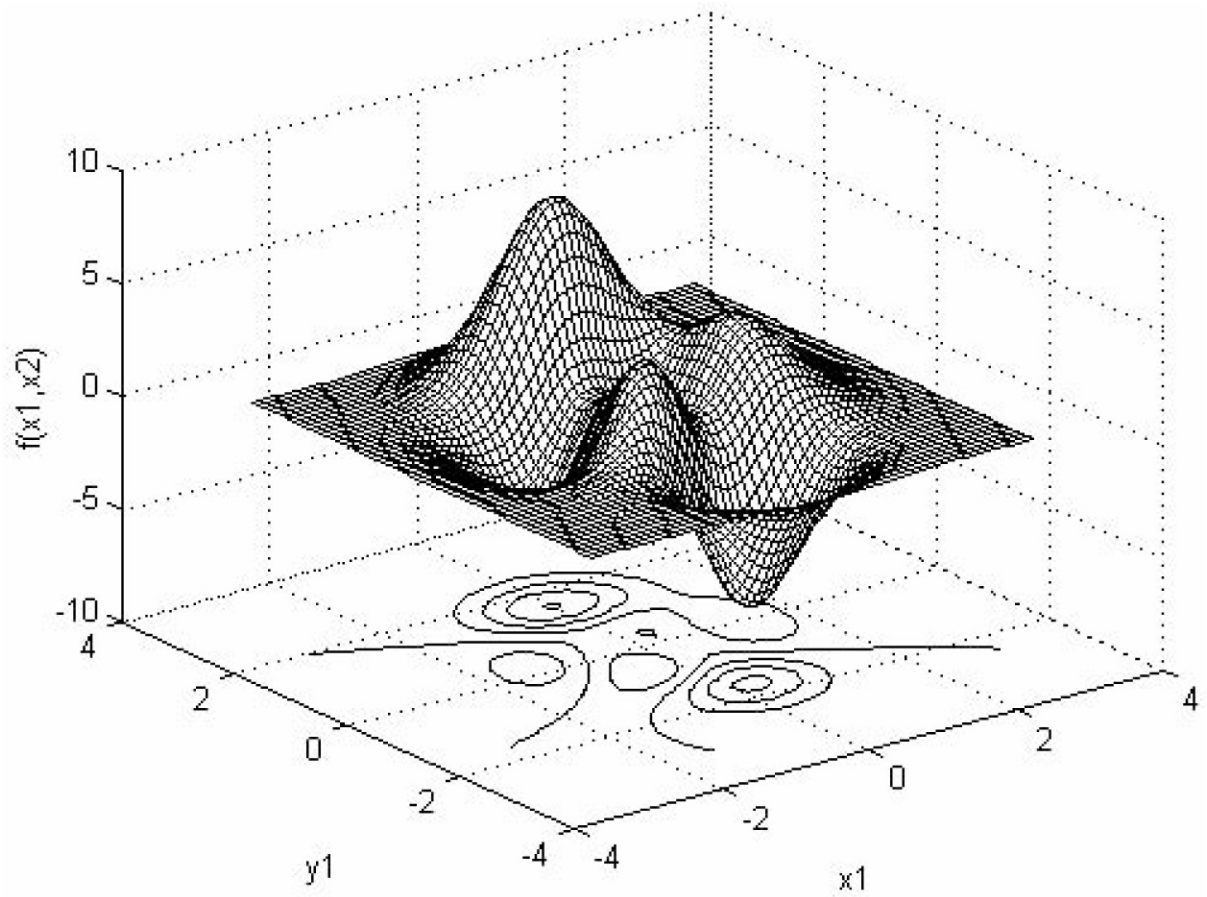

Figura 6.2 - Gráfico da função "*peaks*".

Rainer e Ken mostraram em "***Differential Evolution: A Practical Approach to Global Optimization.***" (Springer, 2005) [4] de forma simples e clara o comportamento do algoritmo para a função *peaks* demonstrado abaixo. Vale ressaltar que o algoritmo não conheçe seu espaço de busca e usa apenas o cálculo da função *fitness* para cada ponto do gráfico.

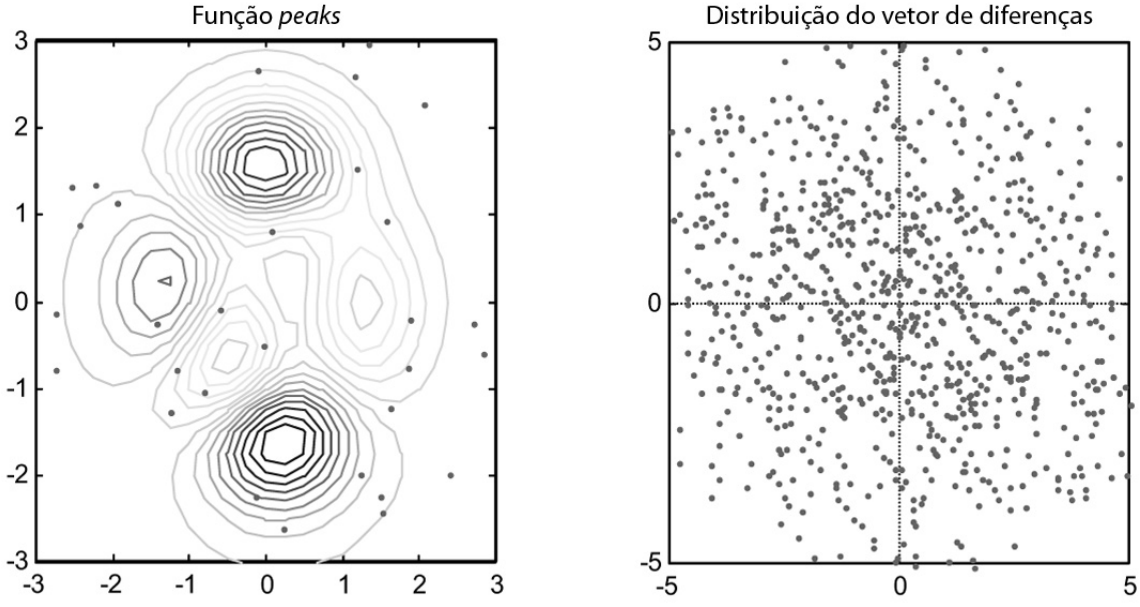

Figura 6.3 - Geração 1: população inicial e distribuição dos vetores de diferenças do ED.

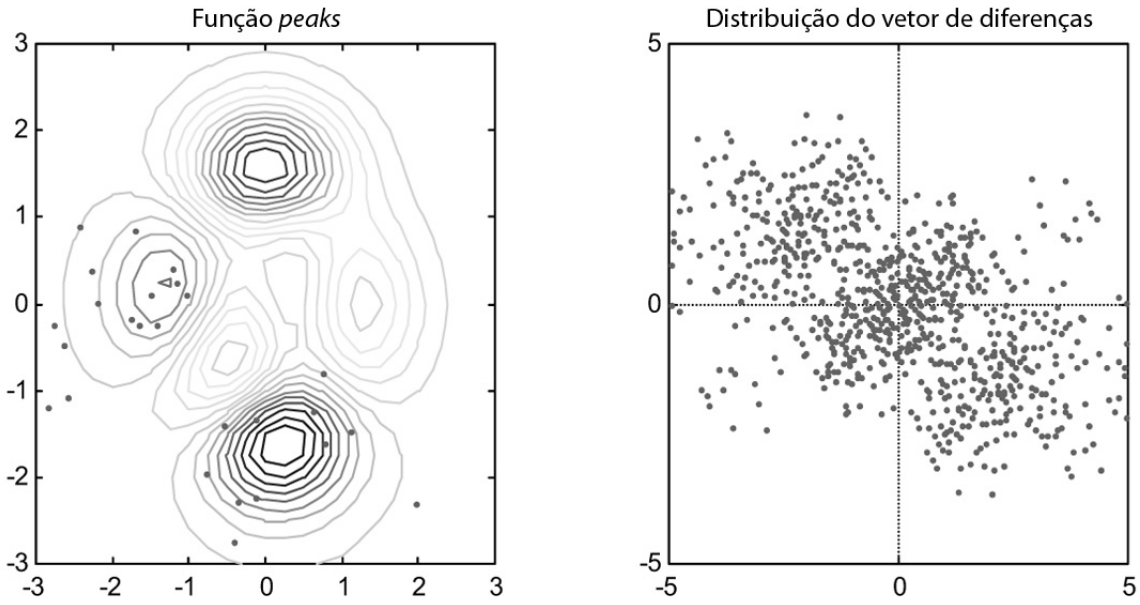

Figura 6.4 - Geração 6: A população permeia os dois mínimos.

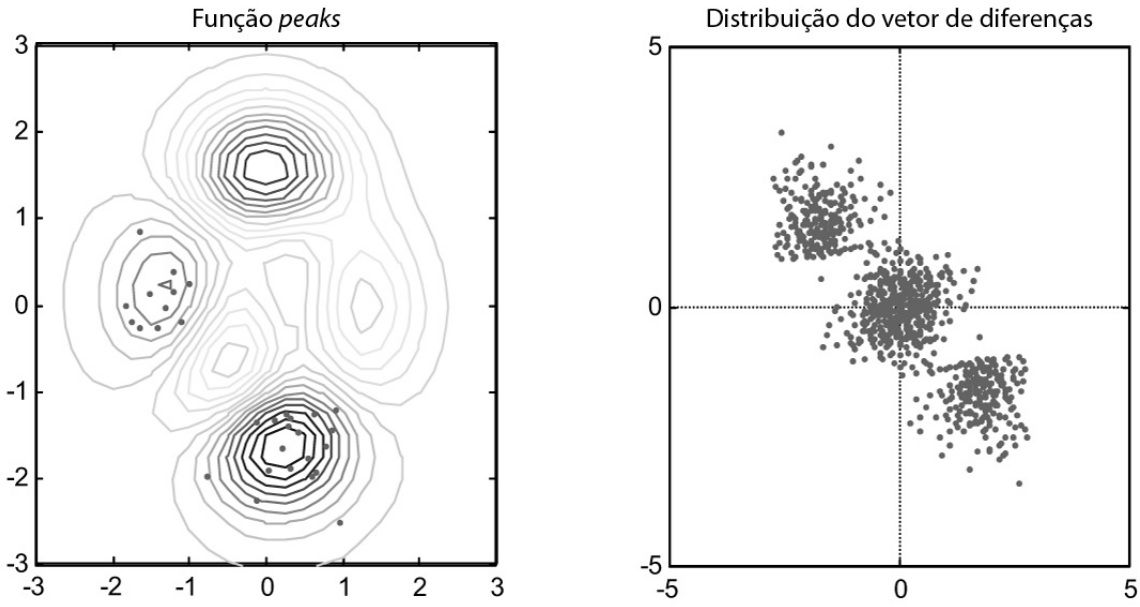

Figura 6.5 - Geração 12: Os vetores de diferenças estão distribuídos em três grandes nuvens.

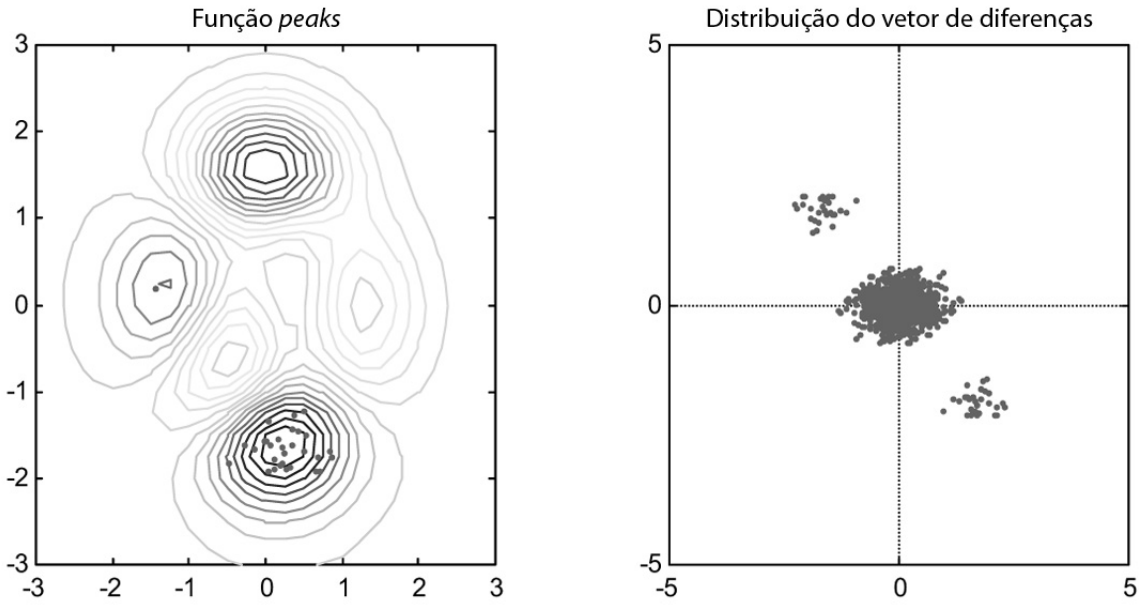

Figura 6.6 - Geração 16: População se concentra no mínimo principal.

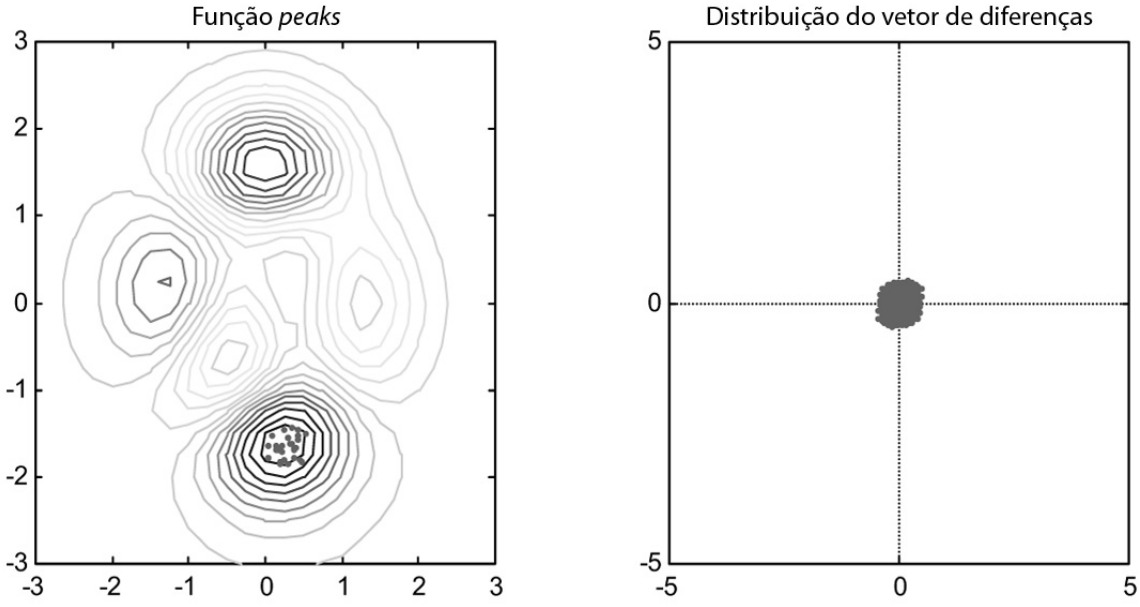

Figura 6.7 - Geração 20: Fase de convergência. Os vetores de diferenças fazem buscas locais.

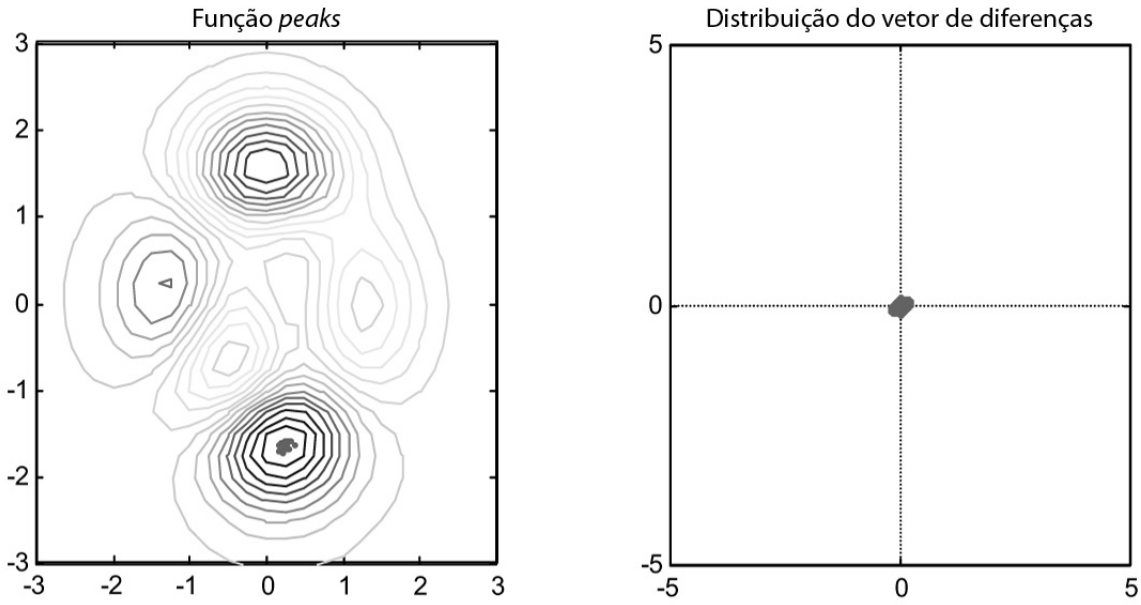

Figura 6.8 - Geração 26: População próxima da convergência.

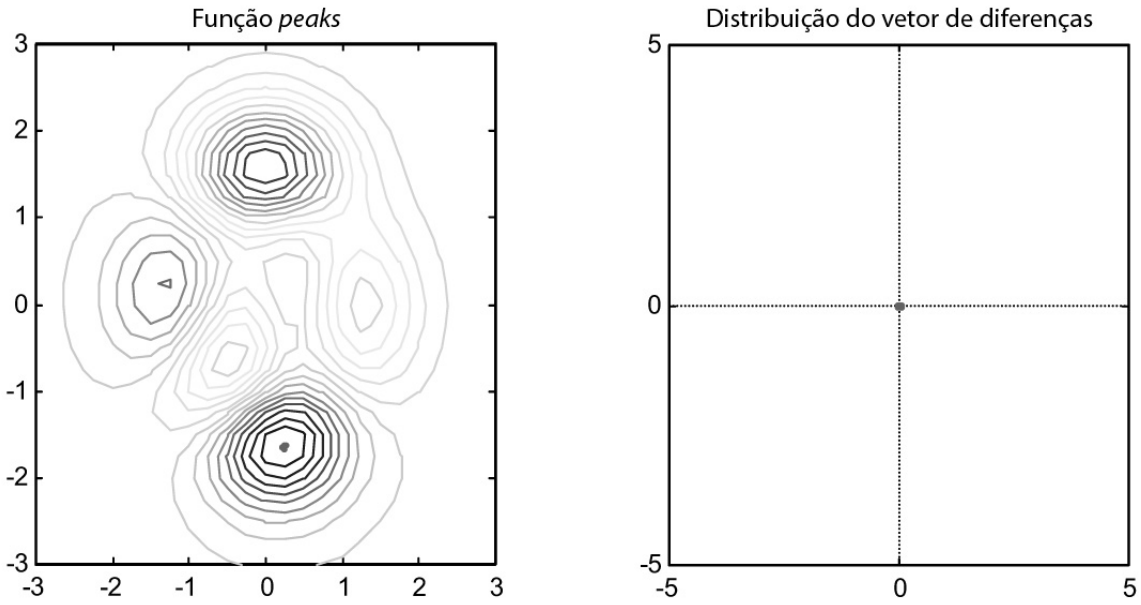

Figura 6.9 - Geração 34: ED encontra o mínimo global.

Alguns fatores observados importantes:

1. A convergência do algoritmo se mostrou bastante rápida que outras estratégias evolucionárias, tais como os algoritmos genéticos;
2. Os raios dos vetores diferenciais diminuem ao longo das iterações devido a mutação ser gerada a partir dos próprios indivíduos e existir o critério de "elitismo" embutido no próprio operador de seleção. Esse comportamento adaptativo faz com que o algoritmo diminua a sua capacidade de exploração em largura e aumente a capacidade de exploração em profundidade.

7. Considerações

Evolução Diferencial apesar de ser uma técnica recente já ganhou respeito da comunidade científica e hoje pode ser encontrada em aplicações de treinamento de redes neurais, análise de imagens, clusterização, projeto de sistemas dentre outras aplicações de otimização de funções não-lineares e não-diferenciável no espaço contínuo. Lembrando que para todos os casos restrições no espaço de busca podem existir, como por exemplo, o uso de funções descontínuas.

Seus pontos fortes são o uso de números reais para representação da solução, da sua rapidez na convergência realizando menos chamadas a função *fitness* e do uso de espaço de busca restritos. Porém, mesmo sendo uma técnica versátil, sua performace está diretamente relacionada ao quanto comportado é o seu espaço de busca, pois é apartir dele que o ED utiliza-o como base de sua busca.

8. Referências

[1] STORN Rainer, PRICE Kenneth. **Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces.** International Computer Science Institute, 1995.

[2] STORN Rainer, PRICE Kenneth. **Differential Evolution – A simple evolution strategy for fast optimization.** Dr. Dobb's Journal of Software Tools. 1997

[3] STORN Rainer, PRICE Kenneth. **Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces.** Journal of Global Optimization, 11(4):431–359, 1997.

[4] PRICE, Kenneth.; STORN, Rainner; LAMPINEN, Jouni A. **Differential Evolution: A Practical Approach to Global Optimization.** Springer, 2005.

[5] STORN, Rainer; PRICE Kenneth. **Differential Evolution for Continuous Function Optimization.** Disponível em: http://www.icsi.berkeley.edu/~storn/code.html Acesso em 20 de março. 2010.

[6] ENGELBRECHT A. **Computational intelligence: An introduction.** 2nd ed. John Wiley & Sons, 2007. 597p.

[7] CHAKRABORTY, Uday K. **Advances in Differential Evolution.** 1st ed. Springer, 2008. 340p.