Introdução
Computação Evolutiva
Evolução Diferencial
Conclusão
Referências

Evolução Diferencial

Introdução e Conceitos Básicos

Levy Boccato
Romis Ribeiro de Faissol Attux
Fernando J. Von Zuben

DCA - UNICAMP

24 de Junho de 2009

Introdução
Computação Evolutiva
Evolução Diferencial
Conclusão
Referências

Resumo

Introdução

Computação Evolutiva

Evolução Diferencial

Conclusão

Referências

Introdução
Computação Evolutiva
Evolução Diferencial
Conclusão
Referências

Introdução

Meta-Heurísticas

- Uma meta-heurística é um conjunto de mecanismos de gerenciamento que atua sobre métodos heurísticos aplicáveis a um extenso conjunto de diferentes problemas. Em outras palavras, uma meta-heurística pode ser vista como uma estrutura algorítmica geral que pode ser aplicada a diferentes problemas de otimização com relativamente poucas modificações que possam adaptá-la a um problema específico.
  1. Simulated Annealing
  2. Busca Tabu
  3. Otimização por colônias de formigas
  4. Algoritmos evolutivos

Introdução
Computação Evolutiva
Evolução Diferencial
Conclusão
Referências

Introdução

Meta-Heurísticas

- Por que estudar meta-heurísticas?
- Características indesejáveis:
  1. Não garantem a obtenção da solução ótima.
  2. Não têm garantia de convergência.
  3. Não têm garantia de custo máximo para se chegar a uma solução.
- **O grande atrativo:** em diversas aplicações, ainda não foram concebidos algoritmos exatos de solução ou mesmo heurísticas específicas. Além disso, os métodos convencionais que garantem a localização da melhor solução são infactíveis. Nestas situações, as meta-heurísticas se tornam candidatas interessantes.

Introdução
Computação Evolutiva
Síntese
Esqueleto básico
Evolução Diferencial
Conclusão
Referências

Computação Evolutiva

Síntese

- A computação evolutiva se inspira em princípios da teoria da evolução e seleção natural e utiliza modelos destes processos naturais para a solução de problemas.
- Principais Ramos:
  1. Algoritmos Genéticos
  2. Estratégias Evolutivas
  3. Programação Evolutiva
  4. Programação Genética
  5. Sistemas Classificadores

Introdução
Computação Evolutiva
Síntese
Esqueleto básico
Evolução Diferencial
Conclusão
Referências

Computação Evolutiva

Esqueleto Básico

- Gere aleatoriamente uma população de soluções candidatas.
- Enquanto o critério de parada não for satisfeito, faça:
  1. recombine alguns indivíduos da população
  2. mute alguns indivíduos da população
  3. avalie todo o repertório de soluções candidatas
  4. selecione segundo algum critério quais soluções irão para a próxima geração

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Histórico

- 1995 - primeira publicação sobre *differential evolution* (DE).
- 1996 - DE participa da Primeira Competição Internacional em Computação Evolutiva, realizada em Nagoya, durante o IEEE Congress on Evolutionary Computation, e conquista o terceiro lugar geral.
- 1997 - Storn, R. e Price, K.,"Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", *Journal of Global Optimization*.
- 1999 - seção dedicada a DE no livro *New Ideas in Optimization*.
- 2005 - primeiro livro dedicado a DE, intitulado *Differential Evolution: A Practical Approach to Global Optimization*.
- 2006 - sessão especial sobre DE no WCCI-CEC'06.
- 2008 - *Advances in Differential Evolution*.
- 2009 - tópico dedicado a DE na *IEEE Transactions on Evolutionary Computation*.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Resumo

- Este algoritmo utiliza NP vetores de parâmetros D-dimensionais $\mathbf{x}_{i,G}, i = 1, \ldots, NP$, como população em cada geração G.
- O conjunto inicial de vetores é gerado aleatoriamente e deve cobrir todo o espaço de busca. Na ausência de qualquer conhecimento acerca do espaço de busca (regiões promissoras ou mesmo soluções parciais), utiliza-se uma distribuição uniforme para a população inicial.
- DE gera novos vetores de parâmetros através da adição da diferença ponderada entre dois vetores de parâmetros a um terceiro indivíduo. Considere esta operação como uma mutação.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Resumo

- Os vetores de parâmetros mutados são então combinados com outros vetores pré-determinados, denomidados *target vectors*, a fim de gerar os *trial vectors*. Esta combinação de parâmetros é referida como *crossover* em DE. É importante ressaltar que cada vetor presente na atual população deve ser usado uma vez como *trial vector*.
- Caso o *trial vector* forneça um valor de *fitness* maior (maximização) que aquele associado ao respectivo *target vector*, este último dará lugar ao primeiro na próxima geração. Esta operação corresponde à seleção.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Mutação

- Para cada *target vector* $\mathbf{x}_{i,G}$, $i = 1, \ldots, NP$, um novo vetor é gerado por meio da seguinte relação:

$$\mathbf{v}_{i,G+1} = \mathbf{x}_{r_1,G} + F \cdot (\mathbf{x}_{r_3,G} - \mathbf{x}_{r_2,G}) \tag{1}$$

- $r_1, r_2, r_3 \in 1, 2, \ldots, NP$ são índices mutuamente distintos e também diferentes do índice i.
- F é uma constante real $\in [0, 2]$ que determina o tamanho do passo a ser dado na direção definida pelo vetor diferença $\mathbf{x}_{r_3,G} - \mathbf{x}_{r_2,G}$.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Exemplo: Mutação

- Seja $\mathbf{x}_{i,G}$ o atual *target vector*.

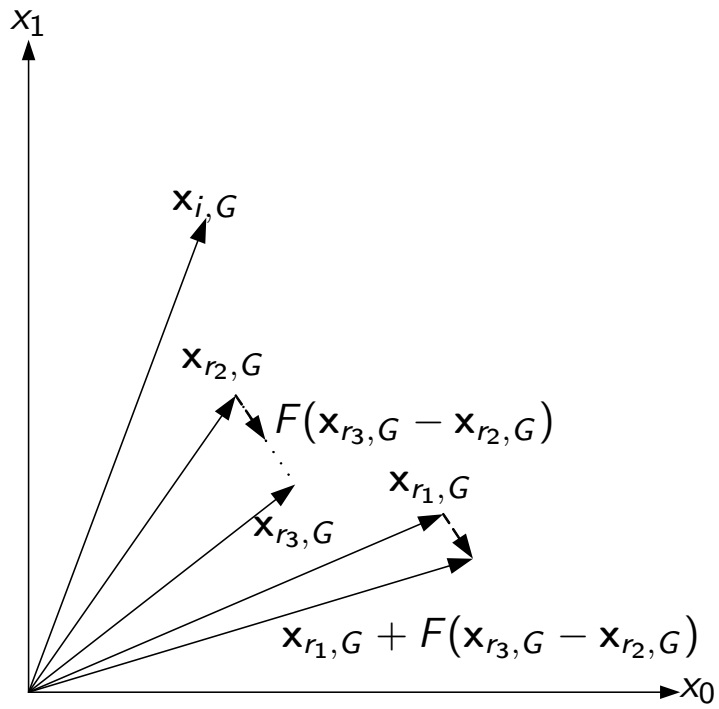

Figura: Exemplo bidimensional do processo de mutação.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Crossover

- Com a finalidade de aumentar a diversidade dos vetores de parâmetros mutados, um procedimento similar ao *crossover* é utilizado.
- Seja $\mathbf{x}_{i,G}$ o *target vector* sob análise e $\mathbf{v}_{i,G+1}$ o respectivo vetor mutado obtido por meio da relação (1) apresentada anteriormente.
- O vetor $\mathbf{u}_{i,G+1} = (u_{1i,G+1}\, u_{2i,G+1} \ldots u_{Di,G+1})$, denominado *trial vector*, é obtido da seguinte maneira:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, & \text{se } r_j \leq CR \text{ ou } j = I_i \\ x_{ji,G}, & \text{se } r_j > CR \text{ e } j \neq I_i \end{cases}, \quad (2)$$

onde $j = 1, \ldots, D$, $r_j \backsim \mathrm{U}(0,1)$, $CR \in [0,1]$ é uma constante definida pelo usuário e I_i é um índice aleatoriamente escolhido $\in 1, \ldots, D$, o que garante que $\mathbf{u}_{i,G+1}$ recebe pelo menos uma componente de $\mathbf{v}_{i,G+1}$.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Exemplo: Crossover

- Seja $\mathbf{x}_{i,G}$ o *target vector* sob análise e $\mathbf{v}_{i,G+1}$ o respectivo vetor mutado obtido por meio da relação (1). A Figura abaixo esboça o processo de geração do *trial vector* $\mathbf{u}_{i,G+1}$.

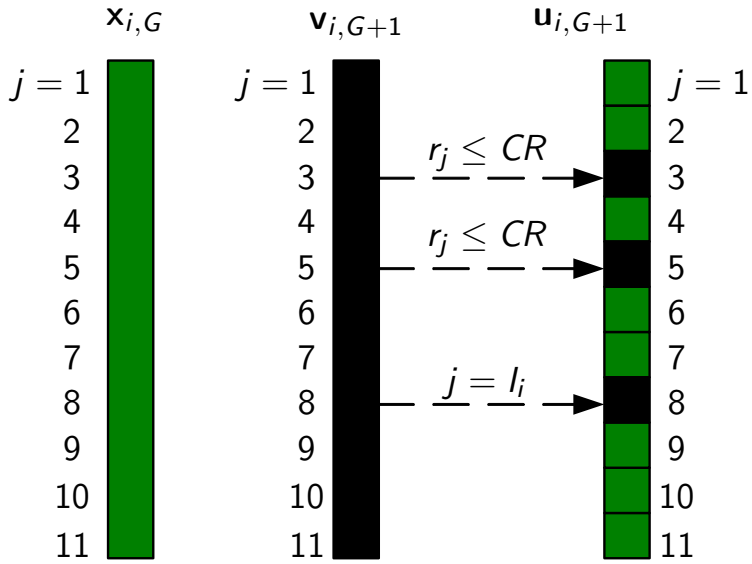

Figura: Exemplo do processo de *crossover*.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Seleção

- Após as etapas de mutação e *crossover*, nas quais todos os NP vetores serviram como *target vector*, a seleção dos vetores que serão preservados para a próxima geração é feita usando um critério guloso.
- Seja $\mathbf{x}_{i,G}$ o *target vector* sob análise e $\mathbf{u}_{i,G+1}$ seu respectivo *trial vector*.
  1. Se $f(\mathbf{u}_{i,G+1}) > f(\mathbf{x}_{i,G})$, então $\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G+1}$. (maximização)
  2. Caso contrário, $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Pseudo-Código

Quadro 1 Evolução Diferencial

Function $\mathbf{x}$ = DE(NP,CR,F,range,f)
$\mathbf{x} \Leftarrow$ random(range,NP)
$fit_x \Leftarrow$ f($\mathbf{x}$)
while critério de parada não for satisfeito **do**
  for $i = 1$ até NP **do**
    $\mathbf{v}_{i,G+1} \Leftarrow$ mutação($\mathbf{x}_{i,G}$,F)
    $\mathbf{u}_{i,G+1} \Leftarrow$ *crossover*($\mathbf{x}_{i,G}$,$\mathbf{v}_{i,G+1}$,CR)
  end for
  $fit_u \Leftarrow$ f($\mathbf{u}$)
  for $i = 1$ até NP **do**
    if $fit_u(i) > fit_x(i)$ **then**
      $\mathbf{x}_{i,G+1} \Leftarrow \mathbf{u}_{i,G+1}$
    else
      $\mathbf{x}_{i,G+1} \Leftarrow \mathbf{x}_{i,G}$
    end if
  end for
end while

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Notação

- A fim de facilitar a discriminação das principais variantes de DE, a notação *DE/x/y/z* foi introduzida, onde:
  1. *x* - especifica o vetor a ser mutado, isto é, $\mathbf{x}_{r_1,G}$.
  2. *y* - determina o número de vetores diferença (direções) utilizados na etapa de mutação.
  3. *z* - indica o esquema de *crossover* adotado.
- *DE/rand/1/bin* corresponde ao algoritmo apresentado nas seções anteriores e representa a proposta clássica da evolução diferencial.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Outros operadores de mutação

- *DE/rand/2*
  - mutação:
    $\mathbf{v}_{i,G+1} = \mathbf{x}_{r_1,G} + F \cdot (\mathbf{x}_{r_3,G} - \mathbf{x}_{r_2,G}) + F \cdot (\mathbf{x}_{r_5,G} - \mathbf{x}_{r_4,G})$
- *DE/best/2*
  - mutação:
    $\mathbf{v}_{i,G+1} = \mathbf{x}_{best,G} + F \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_1,G}) + F \cdot (\mathbf{x}_{r_4,G} - \mathbf{x}_{r_3,G})$
- *DE/target-to-best/1* - não parece o *particle swarm*?
  - mutação:
    $\mathbf{v}_{i,G+1} = \mathbf{x}_{i,G} + F \cdot (\mathbf{x}_{best,G} - \mathbf{x}_{i,G}) + F \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_1,G})$

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Outros operadores de *crossover*

- *DE/exp*

Seja $\mathbf{x}_{i,G}$ o *target vector* sob análise e $\mathbf{v}_{i,G+1}$ o respectivo vetor mutado. O *trial vector* $\mathbf{u}_{i,G+1}$, é obtido da seguinte maneira:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, & \text{para } j = \langle n \rangle_D, \ldots, \langle n + L - 1 \rangle_D \\ x_{ji,G}, & \text{para todos os outros } j \in [1, D] \end{cases}, \quad (3)$$

onde n é um inteiro aleatoriamente escolhido $\in 1, \ldots, D$, L denota o número de componentes que $\mathbf{u}_{i,G+1}$ recebe de $\mathbf{v}_{i,G+1}$ e $\langle \cdot \rangle_D$ denota a função *mod* com módulo D.
O valor de L é determinado da seguinte maneira: Dado $\mathbf{a} = [a_1 \ldots a_D]$, onde $a_i \backsim U(0, 1)$, então

$$L = \max_i \{ i \in [1, D] \mid a_i \leq CR \text{ e } i \neq D \}.$$

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

DE satisfaz alguns requisitos interessantes

- capacidade de lidar com funções custo não-lineares, não-diferenciáveis e multimodais.
- passível de paralelização.
- acessibilidade - poucas variáveis de controle cujos valores são ajustados de maneira relativamente simples.
- auto-ajuste do passo de adaptação - conforme a população converge, os passos são cada vez menores.
- boas propriedades de convergência.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Exemplos

- Maximizar a função
  $f(x,y) = xsin(4\pi x) - ysin(4\pi y + \pi) + 1$, $x,y \in [-1,2]$.
  O máximo global situa-se no ponto $(1{,}62888, 1{,}62888)$.
- Parâmetros DE: $NP = 100$, $CR = 0{,}9$, $F = 0{,}5$.

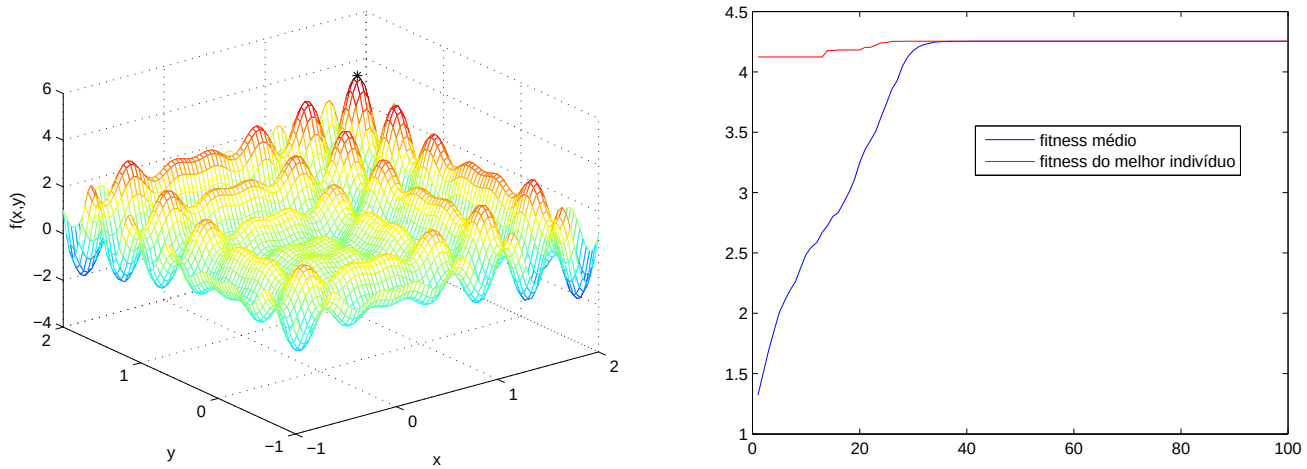

Figura: População final e curvas de *fitness*.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Exemplos

- Minimizar a função de Michalewicz definida como

$$f(x,y) = -\sum_{i=1}^{D} \sin(x_i) \sin\left(i x_i^2/\pi\right)^{2p},$$

com x,$y \in [0,\pi]$. O parâmetro p define a superfície da função. Neste exemplo, usamos $p = 10$. Com $D = 2$, o valor mínimo de $f(x,y)$ é igual a $-1{,}8013$.

- Parâmetros DE: $NP = 100$, $CR = 0{,}9$, $F = 0{,}5$.

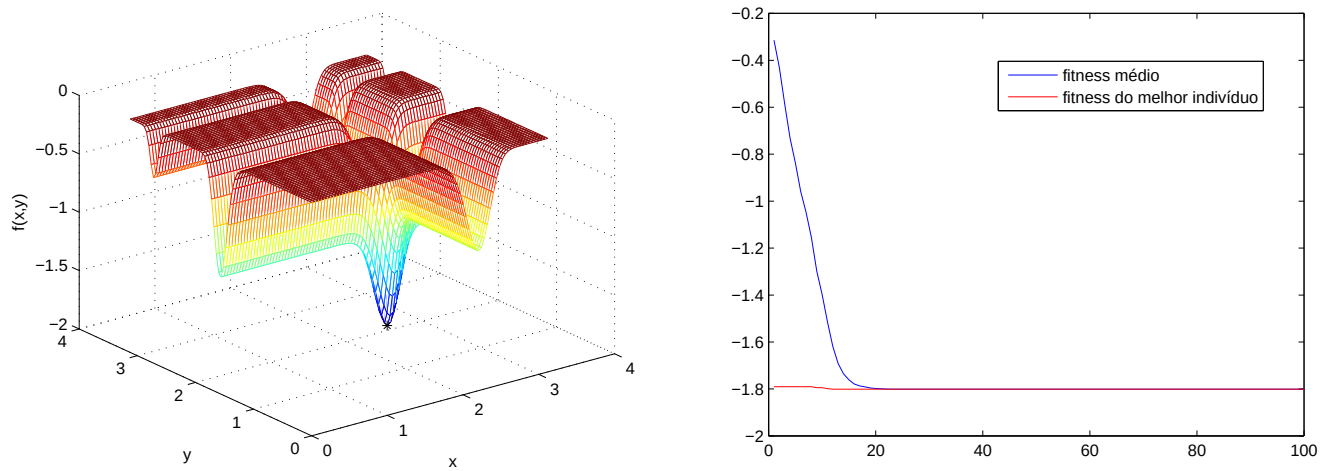

Figura: População final e curvas de *fitness*.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Exemplos

- Minimizar a função de Rosenbrock:

$$f(\mathbf{x}) = \sum_{i=1}^{D-1} (1 - x_i)^2 + 100(x_{i+1} - {x_i}^2)^2,$$

  com $x_i \in [-1, 2]$. Neste exemplo, adotamos $D = 10$. O mínimo global de $f(\mathbf{x})$ está situado no ponto $(x_1, \ldots, x_D) = (1, \ldots, 1)$.
- Parâmetros DE: $NP = 100$, $CR = 0{,}9$, $F = 0{,}5$.
- Após 1000 gerações, a DE foi capaz de localizar o ótimo global precisamente. Na verdade, todos os NP indivíduos atingiram o ótimo global.
- A Figura abaixo mostra a superfície desta função no caso $D = 2$.

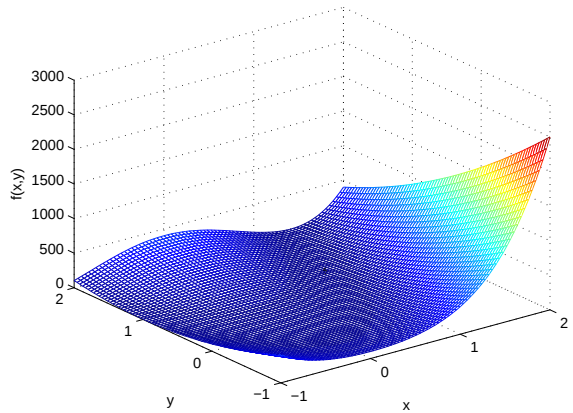

Figura: Superfície da função de Rosenbrock.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Exemplos de Aplicações

- **Projeto de Filtros:**
  1. Storn, R., "Designing Nonstandard Filters with Differential Evolution",*IEEE Signal Processing Magazine*, 2005, pp. 103 - 106.
  2. Storn, R., "Differential Evolution Design of an IIR-Filter",*Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 268 - 273, 1996.
- **Redes Neurais:**
  1. Masters, T. e Land, W., "A new training algorithm for the general regression neural network", *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1990 - 1994, 1997.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Exemplos de Aplicações

- **Telecomunicações:**
  1. Mendes, S.P., Gomez Pulido, J.A., Vega rodriguez, M.A., Jaraiz simon, M.D. e Sanchez Perez, J.M.,"A Differential Evolution Based Algorithm to Optimize the Radio Network Design Problem", *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, 2006.
- **Otimização e Controle:**
  1. Babu, B.V. e M.M.L. Jehan, "Differential Evolution for Multi-Objective Optimization", *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2696-2703, 2003.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Exemplos de Aplicações

- **Otimização Discreta:**
  1. Onwubolu, G. e Davendra, D., "Scheduling flow shops using differential evolution algorithm", *European Journal of Operational Research*, vol. 171, no. 2, pp. 674-692, 2006.
  2. Sauer, J.G. e Coelho, L.S. "Discrete Differential Evolution with local search to solve the Traveling Salesman Problem: Fundamentals and case studies", *IEEE International Conference on Cybernetic Intelligent Systems*, 2008.
  3. Tasgetiren, M.F., Pan, Q.K., Suganthan, P.N. e Liang, Y.C., "A discrete differential evolution algorithm for the total earliness and tardiness penalties with a common due date on a single-machine", *Proceedings of IEEE Symposium on Computational Intelligence in Scheduling*, pp. 271-278, 2007.
  4. Tasgetiren, M.F., Pan, Q.K., Suganthan, P.N. e Liang, Y.C., "A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion", *Proceedings of IEEE Symposium on Computational Intelligence in Scheduling*, pp. 271-278, 2007.
  5. Tasgetiren, M.F., Suganthan, P.N. e Pan, Q.K., "A discrete differential evolution algorithm for the permutation flowshop scheduling problem", *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 158-167, 2007.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Aprendizado baseado em Oposição

- H.R. Tizhoosh, "Opposition-Based Learning: A New Scheme for Machine Intelligence", *Int. Conf. on Computational Intelligence for Modelling Control and Automation (CIMCA)*, vol. I, pp. 695-701, 2005.
- A idéia principal por trás do aprendizado baseado em oposição é considerar um candidato e o seu oposto (uma estimativa e a contrária) ao mesmo tempo com a finalidade de atingir uma melhor aproximação para a solução candidata. Esta idéia pode ser aplicada a uma grande variedade de métodos de otimização, inclusive a outros algoritmos evolutivos.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Aprendizado baseado em Oposição

- Seja $\mathbf{x} = [x_1 \ldots x_D]$ um ponto no espaço D-dimensional, onde $x_i \in \Re$, $x_i \in [a_i, b_i]$, $i = 1, \ldots, D$. As coordenadas do ponto oposto $\check{\mathbf{x}} = [\check{x}_1 \ldots \check{x}_D]$ são dadas por: $\check{x}_i = a_i + b_i - x_i$, $i = 1, \ldots, D$.
- A proposta é empregar este procedimento em dois momentos: na definição da população inicial e durante a execução da DE. Neste último caso, em algumas gerações, ao invés de aplicar os operadores tradicionais de mutação e *crossover*, a população oposta é construída e procede-se à etapa de seleção gulosa (os NP melhores indivíduos de todo o repertório de soluções são preservados). A probabilidade de ocorrência deste procedimento é determinada pelo parâmetro JR (*jump rate*). É interessante destacar que os limites de cada variável x_i são alterados dinamicamente de acordo com a distribuição atual da população no espaço de busca.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Aprendizado baseado em Oposição

- A Figura abaixo mostra um indivíduo da população e seu oposto.

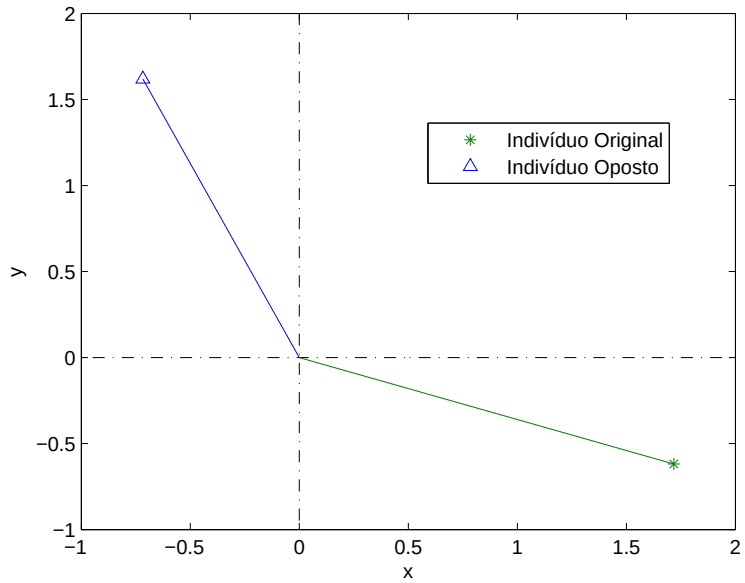

Figura: Aprendizado baseado em oposição - exemplo 2D.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Aprendizado baseado em Oposição - População inicial

- Abaixo, apresentamos a população inicial empregando o aprendizado baseado em oposição.

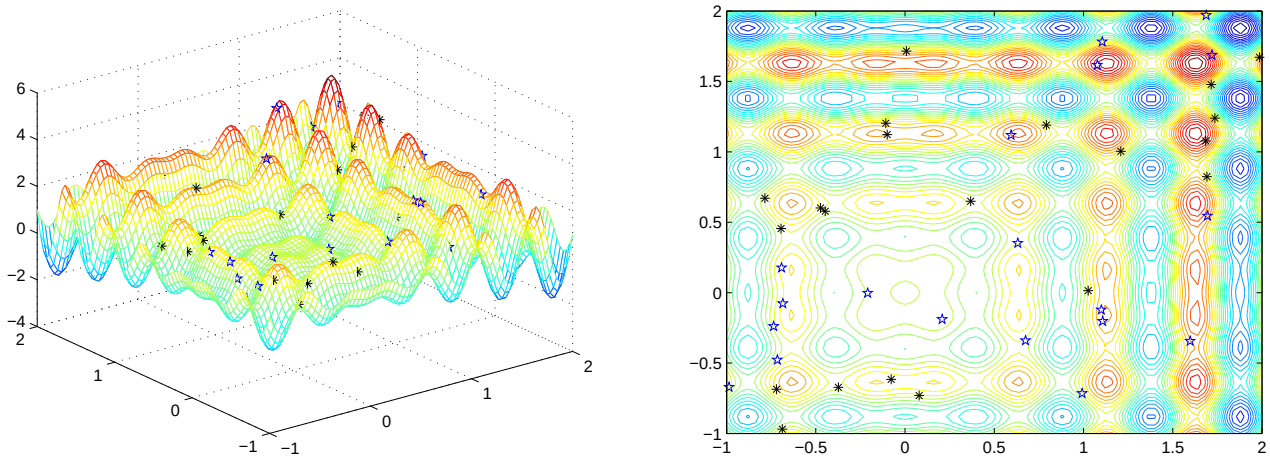

Figura: Aprendizado baseado em oposição na população inicial.

Introdução
Computação Evolutiva
Evolução Diferencial
Histórico
Resumo
Mutação
Crossover
Seleção
Pseudo-Código
Notação
Outros operadores
Requisitos
Exemplos
Aplicações
Aprendizado baseado em Oposição
Auto-adaptação
Conclusão
Referências

Evolução Diferencial

Auto-adaptação

- Estratégias Evolutivas: os parâmetros da mutação gaussiana são incorporados ao genótipo de cada indivíduo da população.
- Idéia: adaptar os parâmetros CR e F juntamente com os vetores de parâmetros $\mathbf{x}_{i,G}$.
- Cada indivíduo passa a ser representado da seguinte maneira: $[x_{1i,G} \ldots x_{Di,G}\ CR_{i,G}\ F_{i,G}]$.

Conclusão

Considerações finais

- Evolução Diferencial constitui uma vertente interessante dentro da Computação Evolutiva. Tal abordagem se mostra bastante simples e eficiente em diversos contextos.
- Tópicos de interesse na sessão dedicada a DE (WCCI 2006):

  1. Theory of differential evolution.
  2. Analysis of parameter settings (scale factor, crossover rate, population size).
  3. Multi-objective differential evolution.
  4. Differential evolution for noisy problems.
  5. Differential evolution for constrained optimization.
  6. Hybridization (with local search and other soft computing approaches).
  7. Applications in diverse domains.

- Novas contribuições certamente serão úteis e pesquisas nesta área são encorajadas.

Referências

Evolução Diferencial

- Storn, R. e Price, K., "Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces" , Technical Report TR-95-012, ICSI, 1995.
- Storn, R. e Price, K.,"Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", *Journal of Global Optimization*, Kluwer Academic Publishers, vol. 11, pp. 341 - 359, 1997.
- Rahnamayan, S., Tizhoosh, H. R., Salama, M. M. A., "Opposition-Based Differential Evolution Algorithms", *Proceedings of IEEE Congress on Evolutionary Computation*, 2006.
- Dasrupta, S., Das, S., Biswas, A. e Abraham, A., "On Stability and Convergence of the Population-Dynamics in Differential Evolution", *AI Communications*, vol. 22, 2009.
- Brest,J., Greiner, S., Bošković, B., Mernik, M. e Zumer, V.,"Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems", *IEEE Transactions on Evolutionary Computation*, 2006.