

Bacharelado em Ciência da Computação

Métodos Heurísticos

Prof. Diego Mello da Silva

Instituto Federal de Minas Gerais - Campus Formiga

13 de novembro de 2014





Sumário

- 1 Fundamentos
- 2 Pseudo-Código
- 3 Exemplo: Minimizando Funções
- 4 Variante: BRKGA
- 5 Referências Bibliográficas

Fundamentos

Algoritmos Genéticos¹

- **Algoritmos Genéticos:** Forma de realizar busca local que emprega métodos baseados em evolução para alterar cromossomos de uma população na tentativa de encontrar soluções ótimas ou pelo menos boas
- John Holland: *Adaptation in Natural and Artificial Systems*
- É um algoritmo populacional
 - Parte de uma população de indivíduos (soluções) gerados aleatoriamente
 - ‘Evolui’ a população no tempo, em busca de novos indivíduos (soluções) melhores adaptados ao problema combinatório
 - Após um certo número de gerações devolve o indivíduo mais apto como solução para o problema
- Mecanismos de evolução: **operadores genéticos**
 - Seleção Natural
 - Recombinação ou Cruzamento
 - Mutação

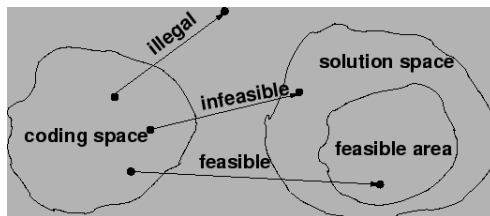
¹ Eiben G., Smith J. *Introduction to Evolutionary Computing*, (ISBN 3540401849)    

Algoritmos Genéticos

- Um indivíduo é representado por uma sequência de alelos (**cromossomo**)
- Os alelos codificam uma solução usando algum tipo de **representação**
- Cada indivíduo possui uma medida de adaptabilidade ao ambiente (i.e., ao problema), denominada de **fitness** ou **aptidão**, denotado por f
- O fitness diferencia soluções boas das ruins
 - Quanto maior o fitness, melhor é a qualidade da solução
 - A adaptabilidade de um indivíduo ao problema pode ser fator usado na seleção natural de uma população entre a geração k e a geração $(k + 1)$
 - Indivíduos mais aptos sobrevivem entre gerações, enquanto indivíduos menos aptos padecem
- Critérios de Parada
 - Evoluir população por quantidade pré-determinada k de gerações
 - Encerrar evolução após n gerações sem melhorar o fitness do melhor indivíduo
 - Desvio percentual em relação à um valor alvo

Algoritmos Genéticos: Representação

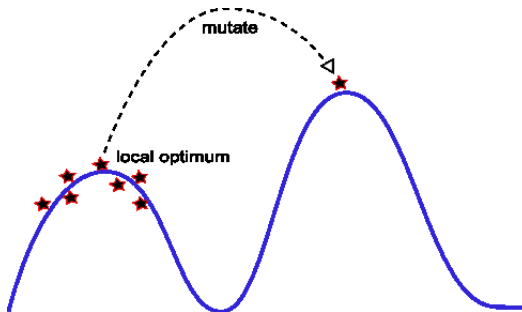
- Representação mapeia o genótipo dos indivíduos no **espaço de codificações** em soluções do **espaço de busca**



- Principais representações:
 - **String de bits:** adequada quando variáveis de decisão são binárias
 - **Inteira:** adequada quando variáveis podem admitir valores inteiros
 - **Ponto Flutuante:** $\langle x_0, x_1, x_2, \dots, x_k \rangle$, com $x_i \in \mathbb{R}, i = 1, 2, \dots, k$
 - **Permutações:** codifica ordem na qual uma sequência de eventos ocorrerá

Algoritmos Genéticos: Mutaç o

- Muta  o altera o c digo gen tico dos indiv duos da popula  o segundo a **taxa de muta  o**
- Insere variabilidade em uma popula  o em evolu  o
- Evita m nimos locais permitindo que indiv duos mutem e representem outras solu  es do espa o de busca
- Exemplo



Algoritmos Genéticos: Mutação

- Operações de mutação comuns, de acordo com a representação utilizada

| Representação | Mutação | Descrição |
|----------------|-----------------|---|
| Binária | Bit-flipping | Cada gene em separado alterna seu valor binário |
| Inteira | Reset Aleatório | Assume novo valor dentro de um conjunto de valores permissíveis |
| Inteira | Creep Mutation | Adiciona valores positivos ou negativos sorteados aleatoriamente de alguma distribuição simétrica à cada gene |
| Pto. Flutuante | Random Change | Sortear aleatoriamente valores para cada gene do genótipo segundo alguma distribuição estatística |
| Permutação | Troca | Trocam-se os alelos de duas posições do genótipo escolhidas aleatoriamente |
| Permutação | Inserção | Escolhem-se duas posições aleatórias do genótipo e movendo-se uma para que fique adjacente à outra |
| Permutação | Inversão | Inverte-se a ordem dos alelos entre duas posições do genótipo escolhidas aleatoriamente |

Algoritmos Genéticos: Mutação

(a) Binária

■ Bit Flipping

$$\langle 1, 0, 1, 0, 0, 0, 0, 1, 0 \rangle \rightarrow \langle 1, 1, 0, 0, 0, 0, 0, 0, 0 \rangle$$

(b) Inteira

■ Random Resetting

$$\langle 1, 5, 3, 7, 2, 8, 6, 9, 0 \rangle \rightarrow \langle 1, 5, 7, 7, 2, 8, 6, 2, 0 \rangle, \text{ com } x_i \in \{0, 1, 2, \dots, 8, 9\}$$

■ Creep Mutation

$$\langle 1, 5, \underbrace{3}_{+1}, 7, \underbrace{2}_{+3}, 8, 6, \underbrace{9}_{-3}, 0 \rangle \rightarrow \langle 1, 5, 4, 7, 5, 8, 6, 6, 0 \rangle$$

(c) Ponto Flutuante

■ Random Change

$$\langle 1.5, 3.2, 1.0, 2.7, 5.9 \rangle \rightarrow \langle 1.5, 6.8, 1.0, 2.7, 2.9 \rangle, \text{ com } x_i \in [0, 7]$$

ou

$$\langle x_1, \dots, x_n \rangle \rightarrow \langle x'_1, \dots, x'_n \rangle \text{ onde } x_i, x'_i \in [L_i, U_i]$$

Algoritmos Genéticos: Mutação

(d) Permutação

■ Troca

$\langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle \rightarrow \langle 1, 5, 3, 4, 2, 6, 7, 8 \rangle$

■ Inserção

$\langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle \rightarrow \langle 1, 2, 5, 3, 4, 6, 7, 8 \rangle$

■ Inversão

$\langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle \rightarrow \langle 1, 5, 4, 3, 2, 6, 7, 8 \rangle$

Algoritmos Genéticos: Seleção dos Pais (Roleta)

- Seleção proporcional ao fitness:

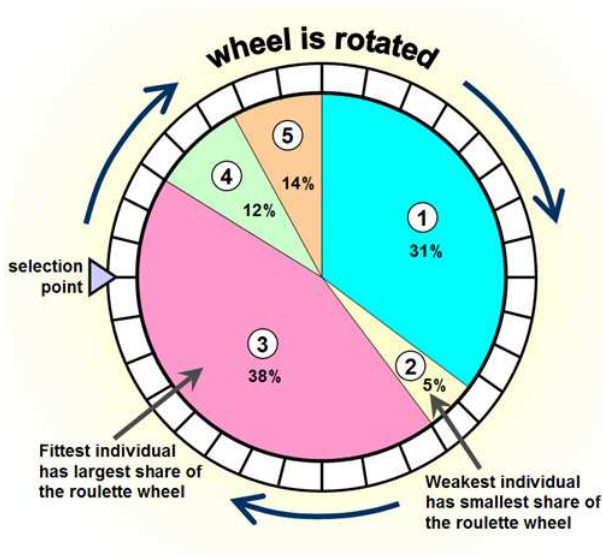
$$Prob(i) = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$$

- Algoritmo da Roleta: cada fatia da roleta é associada à um indivíduo
- Tamanho da fatia: probabilidade do indivíduo ser selecionado para reprodução

Algoritmo 1 ROULETTE-WHEEL()

```
1:  $T \leftarrow \sum_{i=1}^{\mu} Prob(i)$ 
2: Sorteie um número aleatório  $r$  a partir de  $U(0, T)$ 
3:  $S \leftarrow 0$ 
4: for ( $i \leftarrow 1$  to  $\mu$ ) do
5:    $S \leftarrow S + Prob(i)$ 
6:   if ( $S \geq r$ ) then
7:     return  $i$ 
8:   end if
9: end for
```

Algoritmos Genéticos: Seleção dos Pais (Roleta)



Fonte: <http://www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php/> (Nov/2014)

Algoritmos Genéticos: Seleção dos Pais (Torneio)

- Quantidade $\tau < \mu$ indivíduos é selecionada aleatoriamente da população
- Vence o torneio o indivíduo que possuir maior fitness dentre os τ
- Indivíduo vencedor é selecionado para reprodução

Algoritmo 2 TOURNAMENT-SELECTION()

```
1: Selecione  $\tau$  ( $1 < \tau \leq \mu$ ) indivíduos da população, com ou sem substituição
2:  $best \leftarrow 1$ 
3:  $val \leftarrow fitness(i)$ 
4: for ( $i \leftarrow 2$  to  $\tau$ ) do
5:   if  $fitness(i) > val$  then
6:      $val \leftarrow fitness(i)$ 
7:      $best \leftarrow i$ 
8:   end if
9: end for
10: return  $best$ 
```

Algoritmos Genéticos: Seleção dos Pais (Torneio)



Fonte: <http://geneticprogramming.us/>. Acessado em Nov/2014.

Algoritmos Genéticos: Recombinação

- Dados dois ancestrais da k -ésima geração gerar novos indivíduos da geração $(k + 1)$ recombinao os materiais genéticos dos ancestrais
- A partir desta recombinação o algoritmo genético irá visitar novas soluções no espaço de busca
- Operações de recombinação comuns, de acordo com a representação utilizada

| Representação | Recombinação | Descrição |
|----------------|--------------------|---|
| Binária | Crossover n ptos | Escolhe-se n posições aleatórias na faixa $[0, L - 1]$ trocando as caudas em cada ponto |
| Binária | Crossover uniforme | Trata cada gene separadamente, fazendo escolha sobre de qual ancestral irá herdar o gene |
| Inteira | Crossover n ptos | Idem representação binária |
| Inteira | Crossover uniforme | Idem representação binária |
| Pto. Flutuante | Discreta | Idem crossover n ptos e uniforme |
| Pto. Flutuante | Aritmética | Alelo do descendente é gerado a partir de combinação linear de alelo dos ancestrais segundo fórmula $z_i = \alpha x_i + (1 - \alpha)y_i$ para $\alpha \in [0, 1]$. Pode ser simples, única ou completa |

Algoritmos Genéticos: Recombinação

- Exemplo: crossover com 2 ptos de troca, sorteados nas posições 3 e 7

$$P_1 = \langle 0, 1, 1 \mid 0, 1, 0, 1 \mid 1, 0, 1 \rangle$$

$$P_2 = \langle 0, 0, 1 \mid 1, 0, 0, 1 \mid 0, 1, 1 \rangle$$

$$F_1 = \langle 0, 1, 1 \mid 1, 0, 0, 1 \mid 1, 0, 1 \rangle$$

$$F_2 = \langle 0, 0, 1 \mid 0, 1, 0, 1 \mid 0, 1, 1 \rangle$$

- Exemplo: crossover uniforme com probabilidade $p = 0.5$, e valores sorteados de $U(0, 1)$: 0.35, 0.62, 0.18, 0.42, 0.83, 0.76, 0.39, 0.51, 0.36

$$P_1 = \langle 0, 0, 0, 0, 1, 0, 0, 0, 0 \rangle$$

$$P_2 = \langle 1, 1, 0, 1, 0, 0, 0, 0, 1 \rangle$$

$$F_1 = \langle 0, 1, 0, 0, 0, 0, 0, 0, 0 \rangle$$

$$F_2 = \langle 1, 0, 0, 1, 1, 0, 0, 0, 1 \rangle$$

Algoritmos Genéticos: Recombinação

- Exemplo: recombinação aritmética **simples**, com $k = 6$ e $\alpha = 0.5$:

$$P_1 = \langle 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 \mid 0.7, 0.8, 0.9 \rangle$$

$$P_2 = \langle 0.3, 0.2, 0.3, 0.2, 0.3, 0.2 \mid 0.3, 0.2, 0.3 \rangle$$

$$F_1 = \langle 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 \mid 0.5, 0.5, 0.6 \rangle$$

$$F_2 = \langle 0.3, 0.2, 0.3, 0.2, 0.3, 0.2 \mid 0.5, 0.5, 0.6 \rangle$$

- Exemplo: recombinação aritmética **única**, para $k = 8$ e $\alpha = 0.5$

$$P_1 = \langle 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 \rangle$$

$$P_2 = \langle 0.3, 0.2, 0.3, 0.2, 0.3, 0.2, 0.3, 0.2, 0.3 \rangle$$

$$F_1 = \langle 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.5, 0.9 \rangle$$

$$F_2 = \langle 0.3, 0.2, 0.3, 0.2, 0.3, 0.2, 0.3, 0.5, 0.3 \rangle$$

- Exemplo: recombinação aritmética **completa**, para $\alpha = 0.5$

$$P_1 = \langle 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 \rangle$$

$$P_2 = \langle 0.3, 0.2, 0.3, 0.2, 0.3, 0.2, 0.3, 0.2, 0.3 \rangle$$

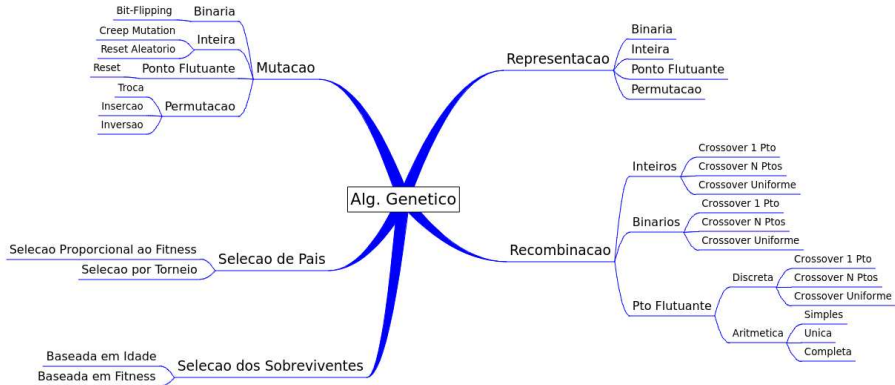
$$F_1 = \langle 0.2, 0.2, 0.3, 0.3, 0.4, 0.4, 0.5, 0.5, 0.6 \rangle$$

$$F_2 = \langle 0.2, 0.2, 0.3, 0.3, 0.4, 0.4, 0.5, 0.5, 0.6 \rangle$$

Algoritmos Genéticos: Seleção dos Sobreviventes

- Equivale a seleção natural da evolução natural
- Seleção de sobreviventes mantêm uma 'memória genética' sobre indivíduos da geração k na geração $(k + 1)$
- Alguns indivíduos da geração k devem sobreviver na próxima geração
- A cada geração, λ novos indivíduos são gerados por recombinação, totalizando $(\mu + \lambda)$
- É preciso descartar λ indivíduos dentre $(\mu + \lambda)$
- Substituição baseada em
 - (a) Idade: indivíduo sobrevive apenas durante a sua geração
 - (b) Fitness: λ piores indivíduos são substituídos
 - (c) Elitismo + Idade: indivíduo mais apto sobrevive entre gerações
- Substituição baseada em fitness pode levar à **convergência prematura**

Algoritmo Genético: Resumo



Pseudo-Código

Algoritmo Genético (Adap. [Eiben e Smith])

Algoritmo 3 ALGORITMO-GENETICO(μ , λ , P_{mut})

- 1: **Inicialize** a população com μ soluções candidatas aleatórias
 - 2: **Avalie** o *fitness* de cada candidato
 - 3: **repeat**
 - 4: **Selecione** pares de pais
 - 5: **Recombine** pares de pais para gerar λ descendentes
 - 6: Aplique a **mutação** nos descendentes com probabilidade P_{mut}
 - 7: **Avalie** os novos candidatos
 - 8: **Selecione** os novos indivíduos para a próxima geração
 - 9: **until** (ocorrer uma condição de parada)
-

- μ : Tamanho da população
- λ : quantidade de novos indivíduos da população após uma nova geração
- P_{mut} : probabilidade de ocorrer mutação

Exemplo: Minimizando Funções

Algoritmo Genético: Minimizando Funções

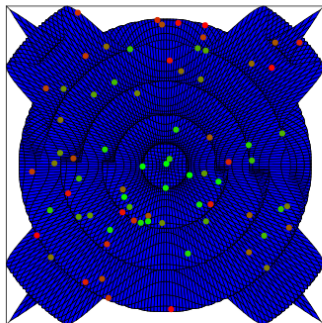
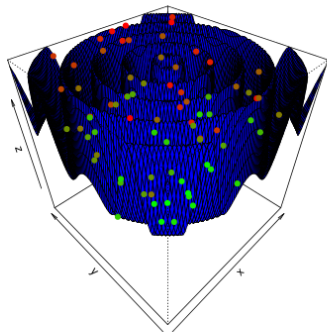
| Definição | Escolha |
|--------------------------|---|
| Representação | Números em Ponto Flutuante: $\{x_1, x_2\}$ |
| Mutação | Reset Aleatório com P_{mut} baixa |
| Recombinação | Recombinação Aritmética Completa |
| Seleção de Pais | Seleção por Torneio ($\tau = 3$) |
| Seleção de Sobreviventes | Substituição dos λ piores ($\lambda < \mu$) |

Exemplo: Minimizando $z = f(x_1, x_2)$

■ DropWave: $f(x_1, x_2) = -\frac{1 + \cos \left[12\sqrt{x_1^2 + x_2^2} \right]}{0.5(x_1^2 + x_2^2) + 2}$, com $-2.0 \leq x_i \leq 2.0$

DropWave: # 0 - F(0.02451,0.03386) = -0.93801

DropWave: # 0 - F(0.02451,0.03386) = -0.93801

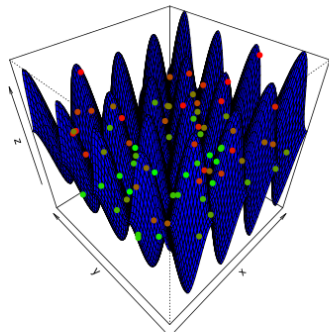


■ Mínimo Global: $f(0,0) = -1$

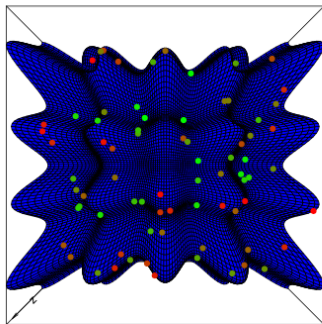
Exemplo: Minimizando $z = f(x_1, x_2)$

- Rastrigin: $f(x_1, x_2) = 20 + [x_1^2 - 10 \cos(2\pi x_1)] + [x_2^2 - 10 \cos(2\pi x_2)]$, com $-2.0 \leq x_i \leq 2.0$

Rastrigin: # 0 - $F(-0.43802, 1.01593) = 2.02281$



Rastrigin: # 0 - $F(-0.43802, 1.01593) = 2.02281$



- Mínimo Global: $f(0, 0) = 0$

Biased Random Key Genetic Algorithm (BRKGA)

Biased Random Key Genetic Algorithm

- RKGA: Introduzido por (Bean, 1994) para problemas de sequenciamento
- Indivíduos são string de números reais (chaves aleatórias) no intervalo $[0, 1]$
- Ordenação das chaves aleatórias gera resultados na ordem da sequência

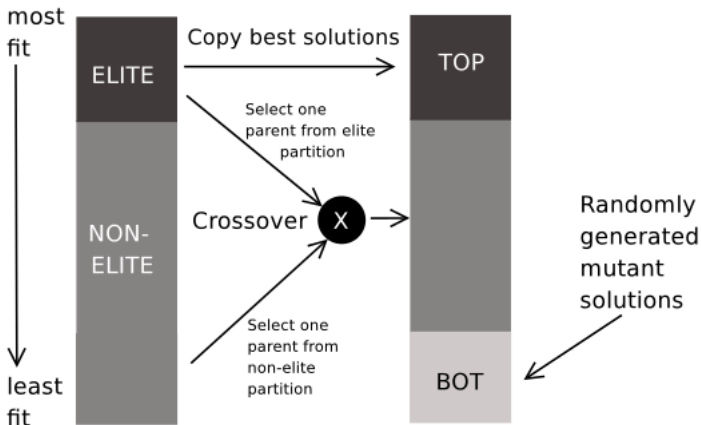
$\langle 0.25, 0.19, 0.67, 0.05, 0.89 \rangle \rightsquigarrow 4 - 2 - 1 - 3 - 5$

- Cruzamento é feito usando crossover uniforme parametrizado
- Diferenças entre as variantes RKGA e BRKGA

| RKGA | BRKGA |
|--|---|
| Ambos os ancestrais são escolhidos aleatoriamente da população inteira | Ambos os ancestrais são escolhidos aleatoriamente, porém um deles é escolhido a partir da população elite |
| Qualquer ancestral pode ser Ancestral 1 no crossover parametrizado | Ancestral mais apto é o Ancestral 1 no crossover uniforme parametrizado |

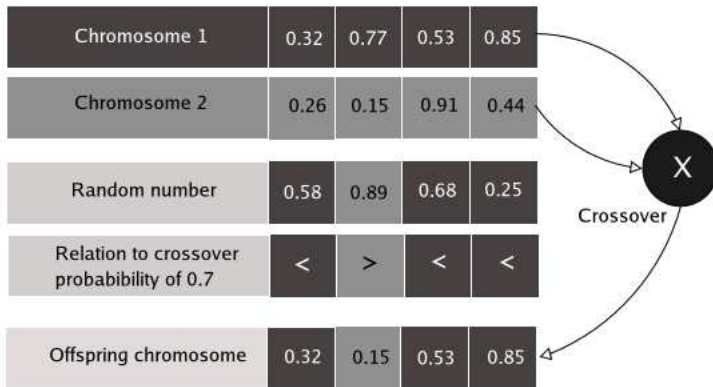
Biased Random Key Genetic Algorithm

- Gerando população ($k + 1$) com população k mais operadores genéticos



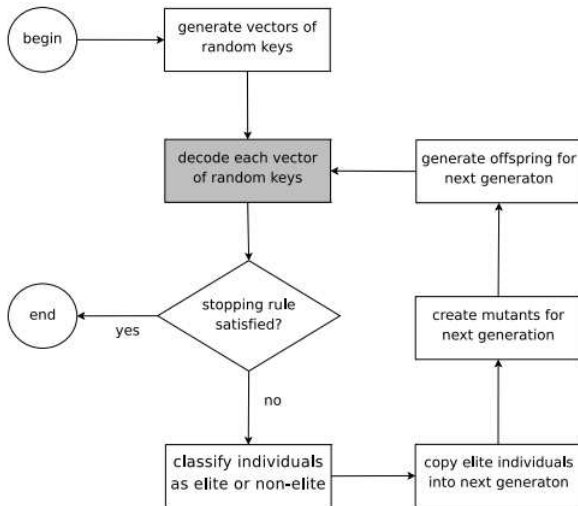
Biased Random Key Genetic Algorithm

- Geração de novos descendentes: crossover uniforme



Biased Random Key Genetic Algorithm

■ Framework do RKGA



Biased Random Key Genetic Algorithm: Recursos

- Genetic Algorithms and Random Keys for Sequencing and Optimization (Bean, 1994)
<http://www.cs.cinvestav.mx/~constraint/papers/Vol006No02Paper05.pdf>
- An Experimental Comparison of Biased and Unbiased Random Key Genetic Algorithms (Gonçalves et al)
<http://www2.research.att.com/~mgcr/doc/brkga-vs-rkga-long.pdf>
- Biased Random Key Genetic Algorithm: A tutorial (Resende, 2012)
<http://www2.research.att.com/~mgcr/talks/2012-09-CLAI02012-brkga-tutorial-both-days.pdf>
- Biased Random Key Genetic Algorithms with applications to optimization problems in telecommunications (Resende, 2012)
http://www.inf.ufrgs.br/elavio2012/elavio2012/Downloads_files/slides_elavio_mauricio2.pdf
- A C++ application programming interface for Biased Random-Key Genetic Algorithm (Toso e Resende, 2011)
http://www.optimization-online.org/DB_FILE/2011/10/3200.pdf

Referências Bibliográficas

Referências Bibliográficas



CAMPELLO R. E, MACULAN N.

Algoritmos e Heurísticas: Desenvolvimento, Avaliação e Performance.
Editora da Universidade Federal Fluminense.



NETTO P. O. B.

Grafos: Teoria, Modelos e Algoritmos, 4a. edição.
Editora Blucher. ISBN: 9788521203919



GLOVER F., KOCHENBERG G. A.

Handbook of Metaheuristics.
Editora Springer. ISBN: 1-4020-7263-5



GAREY M. R., JOHNSON D. S.

Computers and Intractability - A Guide to the Theory of NP-Completeness.
Editora Freeman and Company.



GOLDBARG M. C., LUNA H. P.

Otimização Combinatória e Programação Linear: Modelos e Algoritmos.
Editora Campus.



EIBEN A., SMITH J.

Introduction to Evolutionary Computing.
Editora Springer (Natural Computing Series). ISBN: 3540401849.



PARDALOS P., RESENDE M. G.

Handbook of Applied Optimization.
Editora Oxford.

Referências Bibliográficas



RIVEST R. L., LEIRSON C. E., CORMEN, T. H., STEIN, C.
Algoritmos: Teoria e Prática, 3a. edição.
Editora Campus. ISBN: 9788535236996



LOPES, H. S., RODRIGUES L. C. A., STEINER M. T. A.
Meta-heurísticas em Pesquisa Operacional
Editora Ominipax. ISBN: 978-85-64619-10-4 [recurso eletrônico]
DOI: 10.7436/2013.mhpo.0



RUSSEL, S., NORVIG, P.
Inteligência Artificial, 2a. Edição.
Editora Elsevier, 2004. ISBN: 978-85-352-1177-1