UNISINOS

PIP/CA - ***Programa Interdisciplinar de Pós-Graduação***
Mestrado em Computação Aplicada da UNISINOS

2000/2 - 3o. Trimestre - AULA 07 / FSO

SISTEMAS ADAPTATIVOS INTELIGENTES

- **Professor Responsável:**

Prof. Dr. Fernando Osório

E-Mail: ***osorio@exatas.unisinos.br***

Web: ***http://www.inf.unisinos.br/~osorio/sistadap.html***

TEMAS DE ESTUDO: **Computação Evolutiva-** ***GA + GP***

Introdução

- **Evolutionary Computation (EC)**
- **Genetic Algorithms (GA)**
- **Genetic Programming (GP)**
- **Artificial Life (Alife)**
- **Evolutionary programming, Evolution Strategies, Cellular Automatas, Cellular Machines, Animats, ...**

Algoritmos Genéticos - GA

- Principais Conceitos, Aplicações e Exemplos
- Funcionamento, Algoritmos e Implementação prática
- Discussão sobre os algoritmos genéticos

Programação Genética - GP

- Principais conceitos e Exemplos
- Discussão sobre a programação genética

Discussão final
Bibliografia e Material Complementar

F. OSÓRIO - UNISINOS 2000

COMPUTAÇÃO EVOLUTIVA: Terminologia [Eberhart, Simpson, Dobbins 96]

Evolutionary Computation / Computação Evolutiva - EC

"Machine Learning optimization and classification paradigms that are based on evolution mechanisms such as biological genetics and natural selection"

Genetic Algorithms / Algoritmos Genéticos - GA

"Search algorithms that implement natural evolution mechanisms including crossover, mutation and survival of the fittest (Better individuals of population)". String manipulation (chromosome / genes) - Survival: Fitness Function Evaluation.

Genetic Programming / Programação Genética - GP

"Evolve computer programs genetically: Population = Computer Programs". Symbolic manipulation of programs (Usually: Lisp programs and Trees).

Evolutionary Programming / Programação Evolutipa - EP

"Similar to Genetic Algorithms, but do not implement crossover. Instead, they rely on survival of the fittest and mutation".

Artificial Life / Vida Artificial - Alife

"The study of man-made systems that exhibit behaviors characteristics of natural living systems". [FAQ Alife] (Animats = Animal Robots)

Cellular Automata, Cellular Machine, Cellular Programming, Complex Systems...

COMPUTAÇÃO EVOLUTIVA: Terminologia

Cellular Automata / Cellular Machine

Regular grids of cells, with *'K'* possible states, updated by a synchronous process based on a set of rules (same rules for all cells). Simulate behaviors of interacting cells: the state of a cell is determined by the previous state of a surrounding neighborhood of cells.

Cellular Programming

Evolution of parallel cellular machines. Systems involving the actions of simple, locally-interacting components => "global complex behavior"
Collective systems - Natural Selection - Dynamic Environment
Complex adaptive systems - Artificial Life

• **Sistemas Adaptativos Inteligentes:**

- Como nosso interesse nesta disciplina é centrado em métodos e algoritmos de aprendizado de máquinas (machine learning), vamos nos concentrar apenas nos tópicos referentes a ALGORITMOS GENÉTICOS e PROGRAMAÇÃO GENÉTICA.

***Refs. complementares* - FAQ Alife - http://www.faqs.org/faqs/ai-faq/alife/**
Alife Online - http://alifr.santafe.edu - What is Alife? http://lslwww.epfl.ch/~moshes/alife.html
Alife - http://arieldolan.com/ - Cellular Automata - http://lslwww.epfl.ch/~moshes/ca.html

ALGORITMOS GENÉTICOS: Conceitos Básicos

Origem:

Holland, J. H. (1975) - Adaptation in Natural and Artificial Systems. Univ. of Michigan
De Jong, K.A. (1975) - Na analisys of behavior of a class of genetic adaptive systems. University of Michigan - Ph.D. Thesis.
Goldberg, D.E.(1989) - Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.
Davis, L. D. (1991) - Handbook of Genetic Algorithms. Van Nostrand Reinhold.

Terminologia:

***Population* = Set of initial individuals (set of "start points"). Parameter: size**
***Generation* = Set of individuals of an *epoch*. Size of generations is constant**
***Genome* = Complete set of genetic material of a biological organism. Collection of chromosomes of an individual.**
***Genotype* = Particular set of genes contained in a genome (individual charact.)**
***Phenotype* = Characteristics determined by the genotype (eye color = blue)**

***Chromosomes* = String of genes (DNA) coding individual characteristics. String of bits coding candidate solutions (epoch members).**
***Genes* = Elements of a chromosome (bits). Grouped blocks of adjacent bits.**
***Locus* = Position of a gene on the chromosome.**
***Alleles* = Different possible "settings" of an trait (gene values). Ex.: 0 / 1.**

F. OSÓRIO - UNISINOS 2000

ALGORITMOS GENÉTICOS: Conceitos Básicos

Origem:

Holland, J. H. (1975) - Adaptation in Natural and Artificial Systems. Univ. of Michigan

DNA: Deoxyribonucleic Acid
Chemical Substance that forms chromosomes
Amino Acids: Adenine, Cytosine, Thymine, Uracil, Guanine

Nucleotides:
Elementary bits of DNA (A, C, T, U, G)

Gamete:
Results of combination of a pair of chromosomes (reproduction)

Offspring:
Descendents, progeny of an animal or plant.

Diploid / Haploid:
Chromosome paired (XY) individuals - Unpaired individuals

***Alleles* = Different possible "settings" of an trait (gene values). Ex.: 0 / 1.**

F. OSÓRIO - UNISINOS 2000

ALGORITMOS GENÉTICOS: Conceitos Básicos

Terminologia:

Population / Generation / Genome / Chromosomes / Genes / Alleles / Locus

Reproduction = Set of operations (crossover, mutation, ...) applied to an pair of individuals.

Selection = Select a pair of individuals for reproduction. *Selection Criteria*. Selection based upon a fitness function. Parameter: type of SC.
Fitness = Function used to evaluate each member of one generation, Measure individuals, obtaining a fitness value. Define: function.

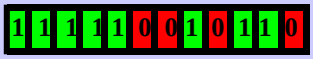
Crossover = Copying selected bits from each parent - Recombine genetic code. Parameters: single-point, double-point, frequency.
Mutation = Small random changes to the bit string. Parameter: frequency.
Inversion = Individual mutation (slice invertion). Parameters: nbits, frequency.

Schema / Schemas / Schemata = 0 / 1 / *Don't Care* - Similarity Templates
Set of bit strings that can be described by a template with *wild-cards*
Template => Bit strings / Bit strings => Template
Example: 1**001 = 100001 / 101001 / 110001 / 111001

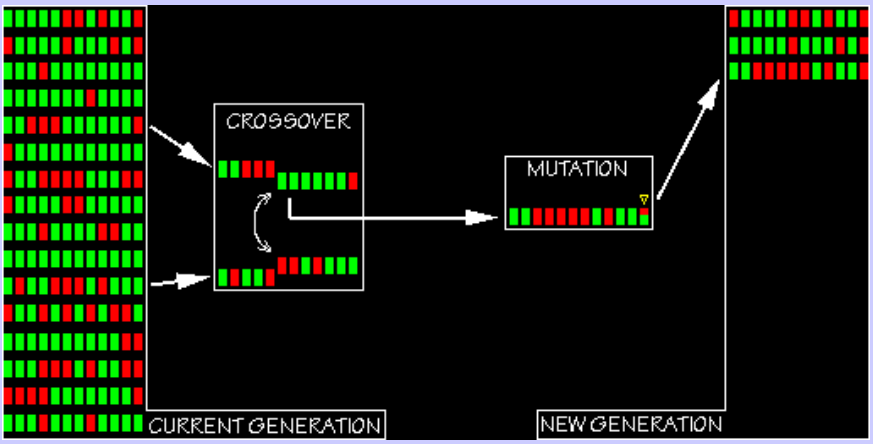
ALGORITMOS GENÉTICOS: Conceitos Básicos

Terminologia:

Population / Generation / Genome / Chromosomes / Genes / Alleles / Locus
Reproduction / Selection / Fitness / Crossover / Mutation / Inversion / Schema



= Chromosome: One individual of a specific generation



F. OSÓRIO - UNISINOS 2000

ALGORITMOS GENÉTICOS: Algoritmo

[Eberhart, Simpson, Dobbins]

1. **Inicializa a população**
2. **Calcula o valor do fitness para cada indivíduo da população**
   2.1 Se atingiu um "grau de evolução" satisfatório, então termina execução!
3. **Seleciona os indivíduos mais bem adaptados formando uma nova população**
   3.1 Aplica o operador de crossover aos paras de indivíduos selecionados
   3.2 Aplica o operador de mutação individualmente
4. **Retorna ao passo 2.**
   4.1 Se atingiu o número máximo de gerações então termina execução!

***Resumindo*:**

- **Initialize population**
- **Apply Fitness Function**
- **Individual Selection**
- **Reproduction: Crossover / Mutation**
- **New generation = new epoch**

F. OSÓRIO - UNISINOS 2000

ALGORITMOS GENÉTICOS: Algoritmo

[Melanie Mitchell]

1. **Init: set random values for n-bit chromosomes**
2. **Fitness F(X) for each individual chromosome X**
3. **Repeat until stop criteria is reached**
   3.1 Select a pair of chromosomes: (with or without replacement)
   Probability of selection being an increasing function of fitness.
   3.2 Crossover the selected pair, with probability *Pc* (crossover probab./rate)
   at a randomly chosen point, generating 2 new chromosomes.
   3.3 Mutate the new chromosomes with probability *Pm* (mutation prob./rate)
   and place the resulting chromosomes in the new population.
4. **Replace current population with new population**
5. **Go to step 2.**

***Steps*:**	***Specific Implementations*:**
- Init	**- Fitness function / Fitness Threshold**
- Fitness	**- Selection: Roulette Wheel, Tournament**
- Select a pair	**- Reproduction: Elitist strategy ("n" best), G_Gap**
- Crossover, Mutation	**- Population Size, Pc, Pm, Crossover points**
	- Initialization, String Coding, Stop

F. OSÓRIO - UNISINOS 2000

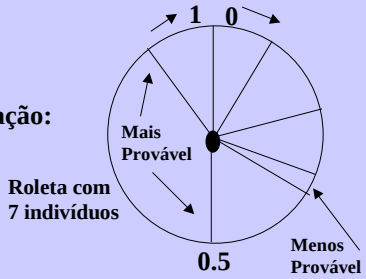
ALGORITMOS GENÉTICOS: Algoritmo

Selection - Roulette Wheel

- **Estimar o valor de Fitness = F(x)**
- **Normalizar o Fitness em relação a população:**

$$F_{norm}(x) = \frac{F(x)}{\Sigma_x F(x)}$$

$$\Sigma_x \; F_{norm}(x) = 1$$



- **Sortear um número aleatório e determinar em que ponto da roleta ele caiu, selecionando o indivíduo correspondente.**

***Tournament Selection*:**

- **Seleciona dois indivíduos aleatoriamente**
- **Compara o fitness value de cada um**
- **O melhor entre os dois ganha o torneio e "sobrevive"**
- **Repete o processo até completar a população da nova geração.**

ALGORITMOS GENÉTICOS: Exemplo de aplicação

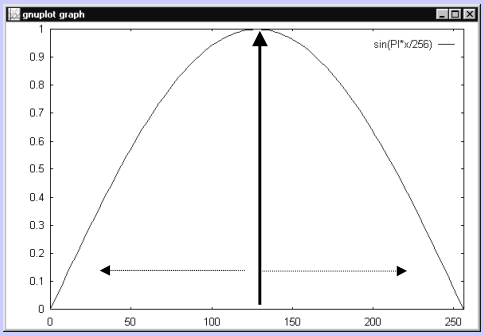
[EberHart, Simpson, Dobbins]

Função a ser otimizada:

F(X) = Sin (Pi * X / 256)

Intervalo: 0 <= x <= 255
Máximo: X = 128 .:. F(X) = 1.0

***Codificação:* String de Bits = X**
***Objetivo*: Achar X que mazimize F(X)**



População: 8 indivíduos
Código binário puro
Inicialização aleatória

Pc = 0.75 / Pm = 0.0
Size = 8 / Coding = Binary
Selection = Roulette Wheel

Search: Best X => Max. F(X)

Indivíduos	X	F(X)	Norm
1 0 1 1 1 1 0 1	189	.733	.144
1 1 0 1 1 0 0 0	216	.471	.093
0 1 1 0 0 0 1 1	99	.937	.184
1 1 1 0 1 1 0 0	236	.243	.048
1 0 1 0 1 1 1 0	174	.845	.166
0 1 0 0 1 0 1 0	74	.788	.155
0 0 1 0 0 0 1 1	35	.416	.082
0 0 1 1 0 1 0 1	53	.650	.128

Σ Norm = 1

ALGORITMOS GENÉTICOS: Exemplo de aplicação

[EberHart, Simpson, Dobbins]

Função a ser otimizada:

F(X) = Sin (Pi * X / 256)

	Indivíduos	X	F(X)	Norm	Acumulado
					0.0
1	1 0 1 1 1 1 0 1	189	.733	.144	0.144
2	1 1 0 1 1 0 0 0	216	.471	.093	0.237
3	0 1 1 0 0 0 1 1	99	.937	.184	0.421
4	1 1 1 0 1 1 0 0	236	.243	.048	0.469
5	1 0 1 0 1 1 1 0	174	.845	.166	0.635
6	0 1 0 0 1 0 1 0	74	.788	.155	0.790
7	0 0 1 0 0 0 1 1	35	.416	.082	0.872
8	0 0 1 1 0 1 0 1	53	.650	.128	1.0

Os indivíduos com um melhor fitness tem mais chances de sobreviver (serem selecionados)

Seleção dos indivíduos para reprodução: *roulette wheel*

- Geração de nros. aleatórios

Valor		Indivíduo	
0.293	=>	3	0 1 1 0 0 0 1 1
0.971	=>	8	0 0 1 1 0 1 0 1
0.160	=>	2	1 1 0 1 1 0 0 0
0.469	=>	5	1 0 1 0 1 1 1 0
0.664	=>	6	0 1 0 0 1 0 1 0
0.568	=>	5	1 0 1 0 1 1 1 0
0.371	=>	3	0 1 1 0 0 0 1 1
0.109	=>	1	1 0 1 1 1 1 0 1

Nova geração antes da combinação

ALGORITMOS GENÉTICOS: Exemplo de aplicação

[EberHart, Simpson, Dobbins]

Função a ser otimizada:

F(X) = Sin (Pi * X / 256)

Seleção dos indivíduos para reprodução: *roulette wheel*

- Geração de nros. aleatórios

Valor		Indivíduo	
0.293	=>	3	0 1 1 0 0 0 1 1
0.971	=>	8	0 0 1 1 0 1 0 1
0.160	=>	2	1 1 0 1 1 0 0 0
0.469	=>	5	1 0 1 0 1 1 1 0
0.664	=>	6	0 1 0 0 1 0 1 0
0.568	=>	5	1 0 1 0 1 1 1 0
0.371	=>	3	0 1 1 0 0 0 1 1
0.109	=>	1	1 0 1 1 1 1 0 1

Crossover (2 points)
Fc = 0.75

Mutation
Fm = 0.0

```
      1     2
0 1 1 | 0 0 0 | 1 1
0 0 1 | 1 0 1 | 0 1
  1       2
1 | 1 0 1 1 | 0 0 0
1 | 0 1 0 1 | 1 1 0
    2           1
0 1 | 0 0 1 0 1 | 0
1 0 | 1 0 1 1 1 | 0

0 1 1 0 0 0 1 1
1 0 1 1 1 1 0 1
```

Reprodução dos selecionados Pares: Crossover Individual: Mutation

ALGORITMOS GENÉTICOS: Exemplo de aplicação

[EberHart, Simpson, Dobbins]

Função a ser otimizada:

F(X) = Sin (Pi * X / 256)

Resultado...
Seleção + Reprodução

Crossover (2 points)
Fc = 0.75

Mutation
Fm = 0.0

0 1 1 | 0 0 0 | 1 1		0 1 1 | 1 0 1 | 1 1
0 0 1 | 1 0 1 | 0 1		0 0 1 | 0 0 0 | 0 1
1 | 1 0 1 1 | 0 0 0		1 | 0 1 0 1 | 0 0 0
1 | 0 1 0 1 | 1 1 0		1 | 1 0 1 1 | 1 1 0
0 1 | 0 0 1 0 1 | 0		1 0 | 0 0 1 0 1 | 0
1 0 | 1 0 1 1 1 | 0		0 1 | 1 0 1 1 1 | 0
0 1 1 0 0 0 1 1		0 1 1 0 0 0 1 1
1 0 1 1 1 1 0 1		1 0 1 1 1 1 0 1

	Indivíduos	X	F(X)	Norm
1	0 1 1 1 0 1 1 1	119	.994	0.157
2	0 0 1 0 0 0 0 1	33	.394	0.062
3	1 0 1 0 1 0 0 0	168	.882	0.139
4	1 1 0 1 1 1 1 0	222	.405	0.064
5	1 0 0 0 1 0 1 0	138	.992	0.157
6	0 1 1 0 1 1 1 0	110	.976	0.154
7	0 1 1 0 0 0 1 1	99	.937	0.148
8	1 0 1 1 1 1 0 1	189	.733	0.116

ALGORITMOS GENÉTICOS: Exemplo de aplicação

[EberHart, Simpson, Dobbins]

Função a ser otimizada:

F(X) = Sin (Pi * X / 256)

Resultado...
Seleção + Reprodução

Search:
X = 128

Crossover (2 points)
Fc = 0.75

Mutation
Fm = 0.0

0 1 1 | 0 0 0 | 1 1		0 1 1 | 1 0 1 | 1 1
0 0 1 | 1 0 1 | 0 1		0 0 1 | 0 0 0 | 0 1
1 | 1 0 1 1 | 0 0 0		1 | 0 1 0 1 | 0 0 0
1 | 0 1 0 1 | 1 1 0		1 | 1 0 1 1 | 1 1 0
0 1 | 0 0 1 0 1 | 0		1 0 | 0 0 1 0 1 | 0
1 0 | 1 0 1 1 1 | 0		0 1 | 1 0 1 1 1 | 0
0 1 1 0 0 0 1 1		0 1 1 0 0 0 1 1
1 0 1 1 1 1 0 1		1 0 1 1 1 1 0 1

	Indivíduos	X	F(X)	Norm
1	0 1 1 1 0 1 1 1	119	.994	0.157
2	0 0 1 0 0 0 0 1	33	.394	0.062
3	1 0 1 0 1 0 0 0	168	.882	0.139
4	1 1 0 1 1 1 1 0	222	.405	0.064
5	1 0 0 0 1 0 1 0	138	.992	0.157
6	0 1 1 0 1 1 1 0	110	.976	0.154
7	0 1 1 0 0 0 1 1	99	.937	0.148
8	1 0 1 1 1 1 0 1	189	.733	0.116

F. OSÓRIO - UNISINOS 2000

ALGORITMOS GENÉTICOS: Exemplo de aplicação / Discussão

[EberHart, Simpson, Dobbins]
Função a ser otimizada: F(X) = Sin (Pi * X / 256)

Aprendizado Genético - Discussão sobre o exemplo apresentado...

- Representação usada nos chromosomos: strings binárias / alfabetos alternativos

Inteiros => Codificados como nro. binários / Gray-Coded (muda 1 bit por vez)
Gray: 0000 / 0001 / 0011 / 0010 / 0110 / 0111 / 0101 / 0100 / ...

Reais => Intervalo + Resolução = representação binária
Exemplo: 2.5 a 6.5 (intervalo 4), se adotarmos 3 casas decimais, precisaremos usar uma string de 12 bits (4 x 3)

- Problemas da representação binária pura: 128 = 1000 0000 / 127 = 0111 1111

- Como determinar: tamanho da população, população inicial, função de fitness, taxa de crossover, taxa de mutação, critério de parada, função de fitness, tipo de seleção (roulette, tournament, elitist, gap), *função de fitness*, ... ???

Scaling window

Aprendizado Genético - Outros exemplos....

- Aprendizado match-strings, truck demo, tsp - travel salesman problem (java demos)
- Aprendizado puzzle (Melanie Mitchell, p.14)
- Aprendizado play-tennis (Tom Mitchell, p. 252)

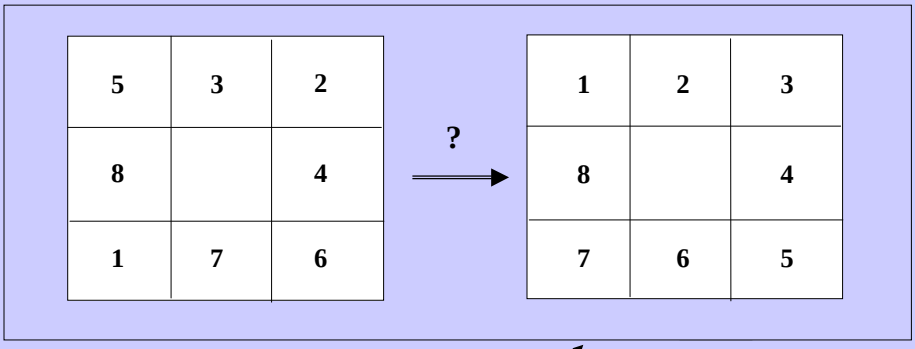
F. OSÓRIO - UNISINOS 2000

ALGORITMOS GENÉTICOS: Exemplo de aplicação / Discussão

[EberHart, Simpson, Dobbins]
Função a ser otimizada: F(X) = Sin (Pi * X / 256)

Aprendizado Genético - Discussão sobre o exemplo apresentado...

- Representação usada nos chromosomos / Fitness Function



Aprendizado Genético - Outros exemplos....

- Aprendizado match-strings, truck demo, tsp - travel salesman problem (java demos)
- Aprendizado puzzle (Melanie Mitchell, p.14)
- Aprendizado play-tennis (Tom Mitchell, p. 252)

F. OSÓRIO - UNISINOS 2000

ALGORITMOS GENÉTICOS: Exemplo de aplicação

[Tom Mitchell] - Do GA ao GP...

Função a ser otimizada: Play Tennis (Quinlan problem)

- Representação de regras do tipo IF-THEN usando strings de bits

Outlook = { Sunny, Overcast, Rain } Wind = { Strong, Weak }
Outlook = 3 bits (um para cada valor)

IF Outlook = Sunny **String de Bits: 100**
IF Outlook = Overcast or Outlook = Rain **String de Bits: 011**

String de bits = 111 **(*don't care* about outlook...)**

Exemplo: (outlook = overcast or rain) and (wind = strong)
Bits: Outlook = 011 / Wind= 10

Pre-conditions + Post-Conditions:
IF Wind = Strong Then PlayTennis = Yes
Bits: Outlook = 111 (don't care) / Wind = 10 / PlayTennis = 10

- Fittness: Regras que cobrem o maior número de exemplos possíveis corretamente

F. OSÓRIO - UNISINOS 2000

PROGRAMAÇÃO GENÉTICA: GP - Definições

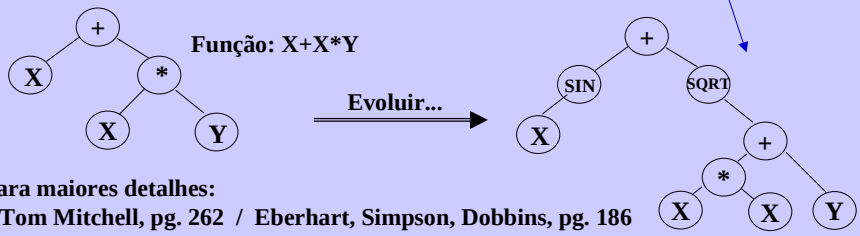
Função a ser otimizada: programas!
String de bits são substituídas por programas que evoluem...

- Representação típica:
Árvores - correspondendo a árvore de parse do programa

- Exemplo: Aprender a função SIN(X) + SQRT (X * X + Y)

Funções primitivas = SIN, COS, SQRT, +, -, *, /, ...
Terminais = X, Y, valores constantes, ...

Evoluir uma população de indivíduos até encontrar a função correta:



Para maiores detalhes:
*** Tom Mitchell, pg. 262 / Eberhart, Simpson, Dobbins, pg. 186**
*** GP Tutorial - http://www.geneticprogramming.com/Tutorial/**

F. OSÓRIO - UNISINOS 2000

EC: GA + GP: Considerações Finais

Evolutionary Computation *versus* Search & Optimization:

- EC utiliza uma "população de indivíduos" (pontos = soluções potenciais) na sua busca do ótimo;
- EC utiliza uma função de "fitness" baseada diretamente na informação, ao invés de derivadas (estimativa da curva do erro);
- O "paradigma EC" é probabilístico, ao invés de muitos outros métodos determinísticos. Evita os mínimos locais... cria "túneis" de uma região a outra do espaço de busca;
- Busca paralela => Algoritmo pode ser implementado em paralelo ("boa granularidade")
- "Occam's Razor": Simples, valor do fitness é direto, sem derivadas, ...

Dificuldades / Requisitos:

- Representar os parâmetros que definem o problema (encode): strings;
- Usualmente trabalha com strings de bits;
- Problema: como representar valores contínuos e funções não lineares (otimização);
- Precisamos definir / conhecer uma função de fitness (direta!);
- Devemos evitar uma explosão combinatória na busca (random search);

Conclusão: GA pode ser muito interessante, GP pode ser muito interessante
GA e GP podem ser difíceis de implementar e "pesados demais"

F. OSÓRIO - UNISINOS 2000

EC: GA + GP: Considerações Finais

Evolutionary Computation *versus* Search & Optimization:

- EC utiliza uma "população de indivíduos" (pontos = soluções potenciais) na sua busca do ótimo;
- EC utiliza uma função de "fitness" baseada diretamente na informação, ao invés de derivadas (estimativa da curva do erro);
- O "paradigma EC" é probabilístico, ao invés de muitos outros métodos determinísticos. Evita os mínimos locais... cria "túneis" de uma região a outra do espaço de busca;
- Busca paralela => Algoritmo pode ser implementado em paralelo ("boa granularidade")
- "Occam's Razor": Simples, valor do fitness é direto, sem derivadas, ...

Dificuldades / Requisitos:

- Representar os parâmetros que definem o problema (encode): strings;
- Usualmente trabalha com strings de bits;
- Problema: como representar valores contínuos e funções não lineares (otimização);
- Precisamos definir / conhecer uma função de fitness (direta!);
- Devemos evitar uma explosão combinatória na busca (random search);

Conclusão: **Aplicação de Algoritmos Genéticos junto a outras técnicas**
Híbridos: GA + ANN / GA + Fuzzy / GA-GP + ML tools

F. OSÓRIO - UNISINOS 2000

TEMAS DE PESQUISA SOBRE ALGORITMOS GENÉTICOS: EC=GA + GP + Alife...

*** LIVROS / PAPERS / DOCUMENTAÇÃO SOBRE ÁRVORES DE DECISÃO:**

- **Obra clássica:**
  Goldberg, D. E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989.
- **Tom M. Mitchell. Machine Learning. McGraw-Hill, 1997.**
- **Melanie Mitchell. An Introduction to Genetic Algorithms. MIT Press, Cambridge, 1999.**
- **Eberhart, R; Simpson, P; Dobbins, R. Computational Intelligence PC Tools. Academic Press,1996.**
- **Lacerda, E. & Carvalho, A. Mini-Curso do JAI 1999 - Congresso da SBC. RJ, 1999.**
- **Holland 1975 / 1986, Davis 1991, De Jong 1993 => GA # Koza 1992, Fogel 1966, 1991, 1993 => GP**
- **Citeseer Nec: http://www.researchindex.com/**
- **FAQs: comp.ai.* (Genetic, Alife, ...)**
  http://www.faqs.org/faqs/ai-faq/genetic/
  ftp://ftp.cerias.purdue.edu/pub/doc/EC/Welcome.html

*** INTERNET:**

• GA-GP.Com	http://www.geneticprogramming.com/ga/index.htm
• GA Introduction	http://lslwww.epfl.ch/~moshes/ga.html
• GA Demo (Java)	http://www.taygete.demon.co.uk/java/ga/index.html
• GA - TSP	http://www.aridolan.com/ga/gaa/gaa.html
• GA Truck Demo	http://www.handshake.de/user/blickle/Truck/index.html