Abstract

This paper proposes a new Estimation of Distribution Algorithm (EDA) based on phylogenetic trees for the identification of Building Blocks. This new algorithm is tested on deceptive large scale problems of the literature and the results are promising.

A probabilistic phylogenetic model adapted within a genetic algorithm capable of solving generic problems was never seen in the literature. This new kind of EDA is called Phylo-Genetic Algorithm (PhGA).

Keywords: Genetic Algorithms, Probabilistic Model Building Genetic Algorithms, Estimation of Distribution Algorithms, Extended Compact Genetic Algorithms, Phylogeny, Phylogenetic Algorithms.

Resumo

Este trabalho propõe um novo Algoritmo de Estimação de Distribuição (EDA) baseado em árvores filogenéticas para a identificação de *Building Blocks*. Este novo algoritmo é testado em problemas deceptivos de larga escala advindos da literatura obtendo-se resultados promissores.

Um modelo probabilístico filogenético combinado com um algoritmo genético capaz de resolver problemas genéricos nunca foi antes encontrado na literatura. Este novo tipo de EDA é denominado Algoritmo File-Genético (PhGA)

Palavras Chaves: Algoritmos Genéticos, Algoritmos Genéticos Construidores de Modelos Probabilísticos, Algoritmos de Estimação de Distribuição, Algoritmo Genético Compacto Estendido, Filogenia, Algoritmos Filogenéticos.

Sumário

Lista de Abreviaturas

EDA – *Estimation of Distribution Algorithm*

(Algoritmo de Estimação de Distribuição)

ECGA – *Extended Compact Genetic Algorithms*

(Algoritmo Genético Compacto Extendido)

cGA – *Compact Genetic Algorithms*

(Algoritmo Genético Compacto)

BBs – *Building Blocks*

(Blocos de Construção)

PhGA – *Philo-Genetic Algorithm*

(Algoritmo Filo-Genético)

Lista de Figuras

1. INTRODUÇÃO

Os algoritmos evolutivos (EAs, do inglês *Evolutionary Algorithms) rep*resentam uma técnica de busca e otimização que tem recebido crescente atenção de pesquisadores nos últimos anos. Essa técnica tem sido aplicada com sucesso em diversas situações, atingindo algumas vezes resultados superiores àqueles conseguidos por técnicas tradicionais (Deb, 2001; Delbem, 2003; Gen & Cheng, 1997; Gen, Cheng & Oren, 2000; Sastry et al 2006; Hoolinger & Gwaltney, 2006; Preble, Lipson & Lipson, 2005). No entanto, poucos algoritmos evolutivos estão aptos a lidar com problemas combinatoriais realmente complexos, envolvendo variáveis que tem relações entre si (mutuamente dependentes), pois essas relações quando não identificadas dirigem o algoritmo para ótimos locais.

Problemas do mundo real exibem em geral um grande número de variáveis mutuamente dependentes. A linha de pesquisa de algoritmos de estimação de distribuição (EDA, do inglês *Estimation of Distribution Algorithm*) tem se mostrado uma das abordagens mais promissoras para resolver tais tipos de problemas. Este texto investiga o *Extended Compact Genetic Algorithm* (ECGA), um dos principais algoritmos da literatura de EDAs, que é capaz de resolver um problema de otimização combinatorial envolvendo conjuntos de variáveis com forte correlações , que formam os chamados blocos construtivos (BBs, do inglês *Building Blocks)*. Para lidar de forma eficiente, robusta e confiável com problemas complexos (com vários BBs relativamente grandes) o ECGA utiliza um modelo probabilístico construído a partir de amostras chamado de Modelo de Probabilidade Marginal (MPM).

Com base no estudo do ECGA, é proposto e implementado neste trabalho um novo EDA baseado na geração de BBs a partir de árvores filogenéticas, também chamadas de filogenias. As árvores de evolução das espécies proposta por Darwin são os exemplos mais conhecidos de filogenias. As árvores construídas por estudos filogenéticos são, na verdade, modelos probabilísticos que buscam a melhor representação hierárquica possível para relações evolutivas de um conjuntos de espécies de animais ou de outros tipos de elementos como proteínas, genes e vírus. O modelo probabilístico filogenético usado em um algoritmo de otimização para problemas genéricos nunca foi encontrado na literatura. Este trabalho propõe tal aplicação, gerando um novo

EDA, denominado algoritmo filogenético (PhGA, do ingês *PhyloGenetic Algorithm*) e mostra resultados promissores advindos do uso.

Orestante deste texto está organizado como descrito a seguir. Na Seção 2 é apresentada uma revisão bibliográfica, descrevendo o ECGA e outros conhecimentos necessários para entender este trabalho. Na Seção 3 é proposto o PhGA e os resultados são mostrados na Seção 4. Na Seção 5 são destacadas considerações finais sobre a relevância da proposta e perspectivas de trabalhos futuros.

2. REVISÃO BIBLIOGRÁFICA

Esta Seção apresenta uma introdução aos principais métodos envolvidos no estudo de EDAs. A Seção 2.1 descreve os conceitos gerais sobre EAs. A Seção 2.2 explica os EDAs seguida pela Seção 2.2.1 que introduz os CGAs. A Seção 2.2.2 que descreve os ECGAs.

A Seção 2.3 descreve possíveis modificações no ECGA para que este apresente melhor eficiência. Dessa maneira, a Seção 2.3.1 explica o *Binary Hillclimber* e a Seção 2.3.2 explica a Compressão de Cromossomo. A Seção 2.4 descreve as possíveis extensões do ECGA para suportar números inteiros ou reais.

2.2. Algoritmos Evolutivos

EAs são técnicas de solução de problemas complexos baseado em um processo evolutivo inspirado nas idéias Darwinianas. Em geral, um EA contém os seguintes elementos:

- Um conceito de indivíduo: uma representação computacional de um candidato à solução;
- Uma população de indivíduos: um conjunto de candidatos à solução;
- Uma noção de aptidão: uma forma de mensurar ou comparar a qualidade de dois indivíduos como solução;
- Um ciclo de "nascimento"/"morte" influenciado pela aptidão: um mecanismo de atualização da população;
- Uma noção de herança: um mecanismo pelo qual características promissoras de indivíduos aptos sejam passadas para seus descendentes.

O algoritmo típico para um EA é sintetizado no Algoritmo 1.

Algoritmo 1: Pseudocódigo de um EA.

Gere aleatoriamente uma população inicial

- Enquanto algum critério de parada não for atingido:
  - Selecione os pais (influenciado pela aptidão)
  - Produza novos filhos
  - Selecione indivíduos para morrer (influenciado pela aptidão)
- Fim enquanto

4. Retorne a população resultante

As diversas técnicas classificadas como EA diferem entre si pela forma com que especificam cada um dos seguintes parâmetros:

- Tamanho da população;
- Seleção dos "pais";
- Reprodução e Herança;
- Sobrevivência;
- Representação dos indivíduos;
- Função aptidão.

2.3. Algoritmos de Estimação de Distribuição

Podemos enxergar um conjunto de indivíduos selecionados de uma população como uma amostra retirada a partir de uma distribuição desconhecida. Conhecer essa distribuição nos permitiria gerar novas soluções que de alguma forma estivessem relacionadas com aquelas selecionadas. Esse é o princípio do EDA (Pelikan, Goldberg & Lobo, 2002; Pelikan, Sastry & Cantú-Paz, 2006), isto é, este é um EA capaz de estimar a distribuição de probabilidade através do conjunto de indivíduos selecionados e, usando essa estimativa, gerar novas soluções.

Essa técnica funciona da seguinte forma: inicialmente, um conjunto aleatório de indivíduos é gerado. Em seguida, alguns deles são selecionados, como nos GAs. No entanto, ao invés de gerar novas soluções por meio de operadores reprodutivos, o EDA estima uma distribuição baseado nos indivíduos selecionados e gera (amostra) novos indivíduos usando essa distribuição. Posteriormente, os novos indivíduos são adicionados à população, substituindo alguns indivíduos antigos. O processo de seleção, estimação e amostragem é repetido até que algum critério de parada seja satisfeito.

Note que, assim com GA, ES e GP, os EDAs são um tipo particular de EA e, sob certas condições, podem ter comportamento idêntico a alguma dessas técnicas (cGA x *Simple* GA) (Harik, Lobo & Goldberg, 1998; Mühlenbein, 1997). A diferença básica é que os EDAs substituem a aplicação de operadores reprodutivos pelos seguintes passos:

- Construção de um modelo probabilístico (uma estimação da distribuição real) dos indivíduos selecionados;

- Geração de novos indivíduos baseado no modelo construído.

O grande problema dos EDA é que a construção de um modelo probabilístico não é uma tarefa trivial. Em geral, existe um compromisso entre a precisão do modelo construído e a eficiência do processo de estimação (Pelikan, Goldberg & Lobo, 2002).

A seguir são apresentados dois EDAs. O primeiro deles desconsidera qualquer interação entre as variáveis (posições da cadeia que representa o indivíduo). O Segundo, considera que as variáveis podem estar relacionadas em grupos, chamados de blocos construtivos (BBs, do inglês *Building Blocks*), porém desconsidera sobreposição entre esses grupos.

2.3.1 *Compact Genetic Algorithm*

A maneira mais simples de estimar a distribuição de uma população é assumir que todas as variáveis são independentes. É nesse principio que é baseado o cGA (Harik, Lobo & Goldberg, 1998).

No cGA, os indivíduos são representados como uma cadeia de *l bits*. A população é representada por um vetor de probabilidades de tamanho *l*. Cada posição desse vetor indica a probabilidade de um indivíduo possuir valor *1* na posição equivalente. Esse vetor representa o modelo da população ideal para o cGA.

O cGA pode ser sintetizado conforme mostra o Algoritmo 2.

Algoritmo 2: Pseudocódigo de um cGA.

```
1. Inicialize o Vetor de Probabilidades

    Para i := 1 até l : p[i] := 0.5

2. Gere dois indivíduos aleatórios:

    A := gerar(p)

    B := gerar(p)


3. Selecione o melhor entre A e B

    Melhor := melhor (A, B);

    Pior := pior (A, B)


4. Atualize o Vetor de Probabilidades influenciado pelo melhor

    Para i := 1 até l :
```

Se Melhor[i] ≠ Pior[i] :

Se Melhor[i] = 1 : p[i] := p[i] + 1/n

Senão: p[i] := p[i] – 1/n

5. Verifique o critério de convergência

Para i := 0 até l :

Se (p[i] ≠ 0 && p[i] ≠ 1) :

retorne para o Passo 2.

No Algoritmo 2 *l* é o comprimento da cadeia de *bits*, *n* é o tamanho da população. O procedimento "gerar" consiste simplesmente em preencher uma cadeia de tamanho l com bits aleatórios Em cada posição i da cadeia a probabilidade do bit gerado ser 1 é dada por p[i]. O procedimento "melhor" retorna aquele entre dois indivíduos que tiver melhor valor na função de avaliação e o procedimento "pior" retorna aquele que tem pior valor.

É possível demonstrar que o cGA pode ser comportar de maneira similar ao *Simple* GA (De Jong, 1975, Harik, Lobo & Goldberg, 1998) .

Além do cGA, outros EDA trabalham assumindo que todas as variáveis são independentes. Os principais algoritmos são o Population-Based Incremental Learning (PBIL) (Baluja, 1994; Kvasnicka, Pelikan & Pospichal, 1996) e o Univariate Marginal Distribution Algorithm (UMDA) (Mühlenbein, Paaß, 1996).

2.3.2. *Extended Compact Genetic Algorithm*

O cGA comporta-se de forma similar ao Simple GA e é capaz de resolver os mesmos problemas apresentando desempenho semelhante. No entanto, o Simple GA não é um EA eficiente

para solução de problemas que envolvam relações complexas entre variáveis e, nesse caso, cGA possui o mesmo desempenho insatisfatório.

O ECGA (Harik, 1999) procurar obter melhor desempenho nesse tipo de problemas através do aprendizado das relações entre variáveis (linkage learning) (Harik, 1997; 1999; Harik & Goldberg, 2000). Baseado nesse conhecimento, é possível usar operadores reprodutivos competentes (Goldberg, 2002), que não destroem um BB apropriado durante sua aplicação. O EGCA é baseado no fato de que encontrar uma boa distribuição de probabilidade é equivalente ao processo de *linkage learning*.

Para entender melhor a importância do processo de *linkage learning*, considere o conceito de problema deceptivo, um problema cuja função aptidão direciona a busca para uma direção contrária à direção do ótimo global. Esse conceito é definido em (Goldberg, 1989a; 2002; Whitley 1991). Considere o seguinte exemplo: suponha que uma determinada função de aptidão f sobre uma cadeia c de l bits seja da seguinte forma:

> *f(c)* = número de *bits* com valor 1 em *c* caso *c* possua algum *bit* diferente de 0.
>
> *f(c)* = *l*+1 caso todos os *bits* de *c* sejam 0.

Observe a função f premia qualquer algoritmo de busca pela adição de 1s à cadeia, no entanto o melhor resultado possível é uma cadeia composta somente por 0. Essa é uma função do tipo "agulha-no-palheiro" (*needle in a haystack*) e não se espera que nenhum método de busca resolva esse problema de maneira eficiente, além disso não há muito interesse prático em problemas dessa forma.

Considere agora uma função *F* que seja deceptiva por partes (Harik, 1999): uma função sobre uma cadeia c de l *bits* que trabalhe agrupando d *bits* em funções deceptivas. Um GA convencional poderia até ser capaz de encontrar uma solução para alguns dos sub-problemas deceptivos, no entanto os operadores reprodutivos teriam grande probabilidade de destruir essa solução pois nenhuma informação sobre a relação entre as variáveis envolvidas nesse sub-problema é utilizada.

O ECGA procura aprender e utilizar relações entre variáveis em um operador de recombinação BB-*wise*. Esse operador utilizará informações obtidas no processo de linkage learning para recombinar cada uma das soluções ótimas para os subproblemas de d bits em uma solução ótima para o problema de l bits.

Percebida a importância do processo de *linkage learning*, o ECGA tenta aprender as relações entre variáveis por meio da construção de uma função adequada de distribuição de probabilidade que não assuma que as varáveis são independentes. No entanto, buscar uma função desse tipo sem a utilização de nenhum bias é uma tarefa inútil. O princípio do ECGA é que, mantidas todas as condições, as distribuições mais simples são preferidas em relação às mais complexas.

O conceito de simplicidade é definido em termos de complexidade de representação. O critério da complexidade combinada (CCC, do inglês *Combined Complexity Criterion*) (Harik, 1999) é empregado nesse sentido. Esse critério diz que a soma da complexidade do modelo (MC, do inglês *Model Complexity*) com a complexidade da população comprimida (CPC, do inglês *Compressed Population Complexity*) deve ser mínima. Essas duas métricas são definidas em (Harik, 1999). Definidos esses conceitos, o ECGA está descrito no algoritmo 3.

Algoritmo 3: Pseudocódigo de um ECGA.

1. Gere uma população aleatória
2. Realize seleção sobre essa população
3. Modele a população usando uma busca gulosa
4. Se o modelo convergiu, pare
5. Gere uma nova população usando o modelo
6. Volte para o passo 2

O algoritmo de busca gulosa usado para a criação do modelo funciona da seguinte forma: inicie assumindo que todas as variáveis são independentes. Então tente unir dois conjuntos de variáveis e observe se há melhorar na CCC. A melhor de todas as uniões (menor CCC) é mantida.

Mais detalhes sobre o ECGA e discussões sobre os resultados obtidos por esse algoritmo podem ser encontrados em (Harik, 1999).

2.4. Melhora de Eficiência no ECGA

ECGA é um algoritmo de grande complexidade, principalmente por tentar criar um modelo probabilístico, e por isso apresenta preocupações quanto a sua eficiência.

Tentando resolver esse problema várias soluções foram propostas, tais soluções podem em alguns casos serem usadas em conjunto, podendo assim aumentar ainda mais o desempenho.

Em seguida será apresentado dois métodos já implementados na literatura que conseguiram melhorar a eficiência deste algoritmo.

2.4.1. *Binary Hillclimber*

O *Binary Hillclimber* funciona da seguinte forma: dada uma cadeia *c* de comprimento *l* e uma função de aptidão *F* aplique o algoritmo 4.

Algoritmo 4: Pseudocódigo de um *Binary Hillclimbing*.

```
Para i := 1 até l :
        Fanterior = F(c)
        inverter(c[i])
```

```
                Fatual = F(C)

                Se (Fatual > Fanterior) :

                        Mantenha a modificação

                Senão :

                        inverter(c[i])

        Fim Para
```

O procedimento “inverter(c[i])” consiste simplesmente em inverter o *bit* na posição i da cadeia c.

Com a aplicação desse procedimento, cada um dos BB que compõe a cadeia c será transformado em um ótimo local para o subproblema ao qual o BB está associado. Com isso, o ECGA terá seu desempenho melhorado por passar a trabalhar com uma quantidade menor de instancias diferentes para um dado BB.

Assim como os EDA, o *Hillclimber* pode ser estendido para representações χ-árias. Para isso, basta substituir o procedimento inverter por um procedimento que gera todos os valores possíveis para a posição i e escolhe aquele que representa melhor valor da função aptidão.

Essa melhoria foi implementada e analisada por (Duque, Delbem & Goldberg 2007a), (Duque, Delbem & Goldberg 2007b).

2.4.2. Compressão de Cromossomo

O princípio de funcionamento de ECGA é que encontrar uma distribuição adequada é equivalente ao processo de *linkage learning*. Baseado nesse princípio, o ECGA tenta aprender relações entre as variáveis para determinar quais são os BB responsáveis por cada subproblema. O algoritmo, então, recombina soluções ótimas para os subproblemas (BBs inteiros) buscando

encontrar a solução ótima para o problema global. A aplicação do *Hillclimber* elimina todas as soluções intermediárias para os subproblemas, mantendo apenas os ótimos locais.

Aproveitando essas duas características para implementar um método de compactação de cromossomo (Duque, Delbem & Goldberg 2007a), (Duque, Delbem & Goldberg 2007b). Esse método consiste em substituir todo o BB que representa a solução de um subproblema por uma única variável χ-ária. Cada um dos valores dessa variável representará um dos ótimos locais do subproblema associado. As representações que não são ótimos locais serão desconsideradas e não terão representação χ-ária. O resultado desse procedimento é uma redução do espaço de busca do ECGA que passa a trabalhar com cadeias χ-árias de menor ordem (menor número de variáveis).

2.5. ECGA com Números Reais ou Inteiros

ECGAs foram primeiramente construídos para otimizar cadeias binárias de variáveis, porém há o interesse de adaptar este algoritmo para que ele seja capaz de otimizar problemas com números inteiros e reais. Pois um grande número de problemas de otimização são modelados em sobre números reais ou inteiros.

Para tanto foi desenvolvido várias modificações nos ECGAs recentemente, de forma a conferir a este algoritmo a capacidade de tratar problemas envolvendo números inteiros e reais.

Em (Ping-chu Hung, Ying-ping Chen, Ying-ping Chen 2006) é implementado o iECGA, que é uma modificação do ECGA para números inteiros. Este algoritmo funciona da mesma maneira que o ECGA mudando apenas a representação e a função de calculo de CCC. O algoritmo resultante consegue encontrar BB em problemas envolvendo inteiros mas falha em encontrar BB em problemas com *bits*, enquanto o ECGA é o inverso. Então os dois algoritmos resolvem problemas em específicos domínios.

Um método denominado *Split on Demand* foi implementado em

(Chao-hong Chen, Wei-nan Liu, Ying-ping Chen 2006), no qual acrescenta-se a possibilidade de tratar números reais dentro do ECGA. O método divide o espaço de busca nas regiões mais promissoras. Esse método é adicionado ao ECGA em conjunto com outras modificações para melhorar o desempenho (buscas locais usando Simplex (Nelder, Mead 1965)

resultando no rECGA (*real-coded* ECGA). Os resultados conseguidos foram próximos a outros algoritmos no estado da arte.

2.6. *Neighbor Joining*

Neighbor Joining é um método de *clustering* utilizado em bioinformática, para a construção de árvores filogenéticas. Esse método baseia-se no uso de distâncias entre cada par de sequências de proteínas ou DNA, na forma de uma matriz de distâncias, para a construção da respectiva árvore filogenética.

A árvore resultante apresenta em suas arestas as distâncias distribuídas de forma que o caminho de um nó folha até outro, se somado as arestas, será o mesmo valor da matriz de distâncias passada para construir a árvore.

É desconhecida qualquer aplicação deste método fora das áreas de bioinformática, no entanto este projeto visa uma aplicação mais genérica deste método, considerando ele como um algoritmo de *clustering* genérico entre variáveis, de acordo com uma métrica de distância.

2.7. Considerações Finais

*Nesta Seção foi introduzido os conhecimentos necessários para o entendimento do algoritmo e apresentando o estado-da-arte n*a área. Na Seção 3 o PhGA é apresentado, bem como experimentos relativos ilustrando suas contribuições para problemas complexos de larga-escala.

3. O Algoritmo Filogenético

Esta pesquisa buscou a criação e implementação de um novo algoritmo genético (GA, do inglês *Genetic Algorithm*) usando conceitos da área da filogenia. O funcionamento do Algoritmo Filogenético é ilustrado pelo diagrama da Figura 1.

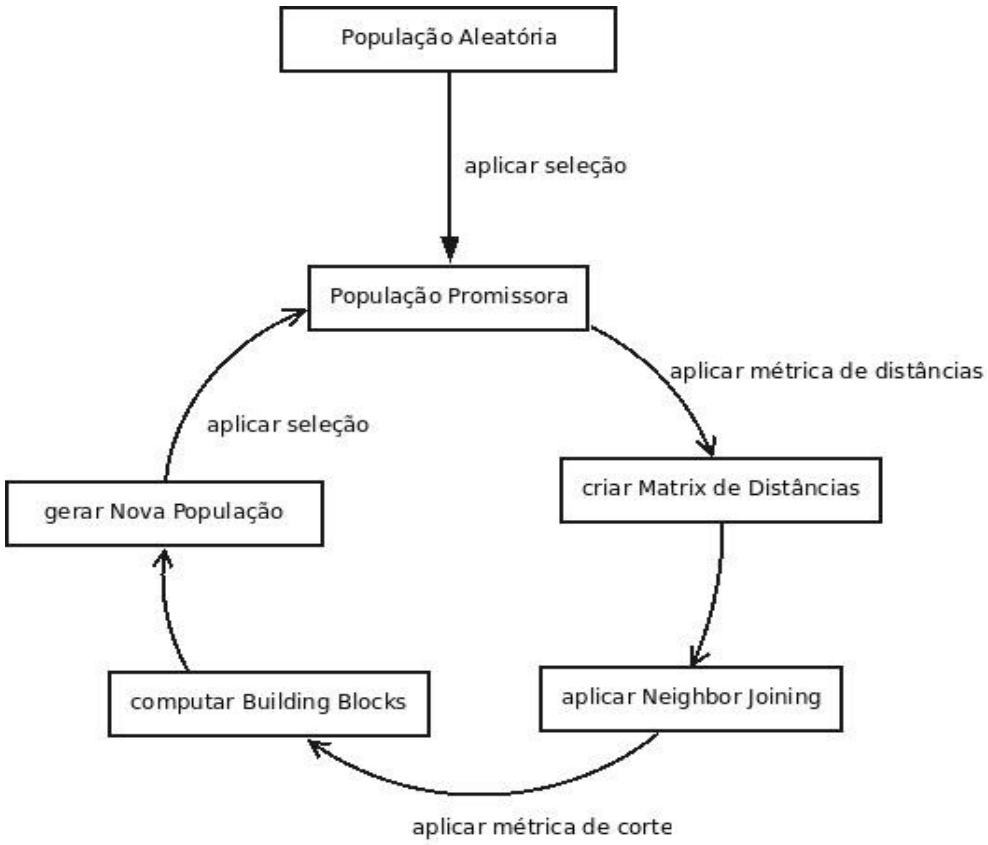

Figura 1: Diagrama descrevendo o PhGA.

O algoritmo inicia com uma população aleatória e a partir dela seleciona uma população promissora. Essa população promissora é selecionada baseada em torneios de 16 indivíduos advindos da população aleatória, em que o vencedor é colocado na população promissora. A população promissora pode conter a presença de um único indivíduo varias vezes.

A população promissora pode ser vista como um conjunto de amostras, as quais possuem um conjunto probabilisticamente melhor que o anterior (considerando melhor a população que contem indivíduos com maior *fitness*). Dessa forma, é aplicado em seguida uma métrica de distância comparando as variáveis.

Essa métrica pode ser modificada, alterando assim o subseqüente agrupamento. Sabe-se que o problema de otimização torna-se mais fácil ao se conhecer os BBs (grupos de variáveis mutuamente dependentes), pois evita-se a quebra de configurações importantes no *crossing over.*

Assim, com o objetivo de agrupar variáveis que são mutuamente dependentes, uma métrica é aplicada para cada par de variáveis, resultando numa matriz NxN, onde N é o número de variáveis.

Essa matriz é passada para o método de *clustering* denominado *Neighbor Joining*. Esse método por sua vez produz uma árvore binária contendo as variáveis do problema nas folhas de forma com que as variáveis mais próximas fiquem também próximas nesta árvore.

Na árvore binária usa-se uma heurística de corte almejando a separação dos BBs. Observe que a heurística de corte pode variar de acordo com métrica de distância.

Os BBs resultantes são então usados, assim como no ECGA, como um modelo da população e são usados ao aplicar o *crossing over.* De forma que os BBs não sejam quebrados.

Por fim, o processo é reiniciado a partir dessa nova população gerada.

3.2 Métricas de Distância

A seguir encontram-se as métricas de distância codificadas e testadas.

- *Mutual Information*: Em Teoria da Informação, esta métrica é comunmente utilizada para calcular a mutual dependência entre duas variáveis. A formula é apresentada a seguir:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right),$$

onde p(x,y) é a Probabilidade de Distribuição Conjunta de *X* e *Y*, e *p1(x)* e *p2(y)* são as probabilidades marginais de *x* e *y* dadas as distribuições *X* e *Y* respectivamente.

- Diferença entre médias: Esta métrica calcula a média do conjunto de amostras de duas variável e retorna a diferença entre estas duas variáveis. Note que a média de uma variável no caso binário é idêntico ao valor da probabilidade da mesma variável ser 1.

- *ix Metric:* Métrica que mistura ambas as métricas de cima de forma normalizada.

3.3 Heurísticas de Corte

A seguir são especificadas cada uma das técnicas de corte da árvore filogenética empregadas para determinação dos BBs:

- *Average Threshold*: Heurística de corte baseada no valor médio das distâncias entre os nós da árvore. Esta heurística corta as arestas que possuem valor maior do que o valor médio;

- *Average Threshold Extended*: A mesma heurística descrita acima com a exceção de que esta heurística não corta os nós folhas. Ela foi criada com o objetivo de evitar com que os nós folhas fossem cortados, formando BBs de apenas uma variável;

- *BB Function*: Heurística baseada no agrupamento seqüencial a partir de um nó folha. A função utilizada é: F(y)= K * 1/y . Aonde y é o número de variáveis presentes na subárvore embaixo do nó escolhido (que por sua vez é igual ao tamanho do BB formado caso fosse efetivado o corte) e K é uma variável arbitrária. A função resulta na mínima distância necessária para cortar a árvore no nó;

- *Layer Function*: Heurística que corta a árvore inteira numa camada a partir do nó raiz. Dessa maneira, ela forma BBs e pode usar várias possibilidades de BBs (pois cada camada propicia uma possibilidade de corte) juntos ou escolher um deles.

3.4 *Crossver*

O *crossing over* pode ocorrer basicamente de duas formas:

- Pai e Mãe: Primeiramente é escolhido dois cromossomos (abstratamente um pai e uma mãe) e depois é trocado blocos de variáveis respeitando os BBs e formando um terceiro indivíduo que fará parte da próxima população;

- Probabilístico: Para cada bloco dentro dos BBs é escolhido aleatoriamente um indivíduo da população que deve passar aquele bloco para o indivíduo da próxima geração. Este processo é repetido até que todos os blocos sejam passados.

3.5 *Tournament*

Duas estratégias de torneio foram investigadas:

- Sem Reposição: O mesmo torneio usado no ECGA, aonde o indivíduo selecionado por um torneio pode ser selecionado novamente pelo próximo torneio;

- Com Reposição: Este torneio não permite que um mesmo indivíduo seja selecionado duas vezes. É importante notar que neste caso a população de indivíduos selecionados precisa ser necessariamente menor que a população de indivíduos presentes, pois caso contrário o efeito de seleção não ocorre.

3.6 Outras Variáveis

Além desses aspectos investigados, outros elementos importantes para o desempenho adequado de EDAs e, por sua vez, do PhGA podem ser mais intensamente pesquisados:

- Tamanho da População;

- Tamanho da População de Selecionados;

- Tamanho do número de indivíduos usado nos Torneios;

- Presença de Ruído nos novos indivíduos criados;

- *Hill Climbing;*

- Uso de múltiplos BBs para a geração dos indivíduos, decidindo probabilisticamente em qual usar para cada indivíduo.

4. Resultados Experimentais com o PhGA

Para a implementação do PhGA foi primeiramente implementado uma biblioteca para Grafos que possibilitasse a criação de árvores binárias. Essa biblioteca foi elaborada com a capacidade de imprimir grafos na linguagem DOT (Koutsofios, North, Intset, Sparcmcemit, Joinwitharg 1996). De forma, que esses pudessem ser vistos em ferramentas de visualização de grafos.

Na Figura 2 mostra-se uma visualização obtida a partir de uma árvore binária criada pelo PhGA.

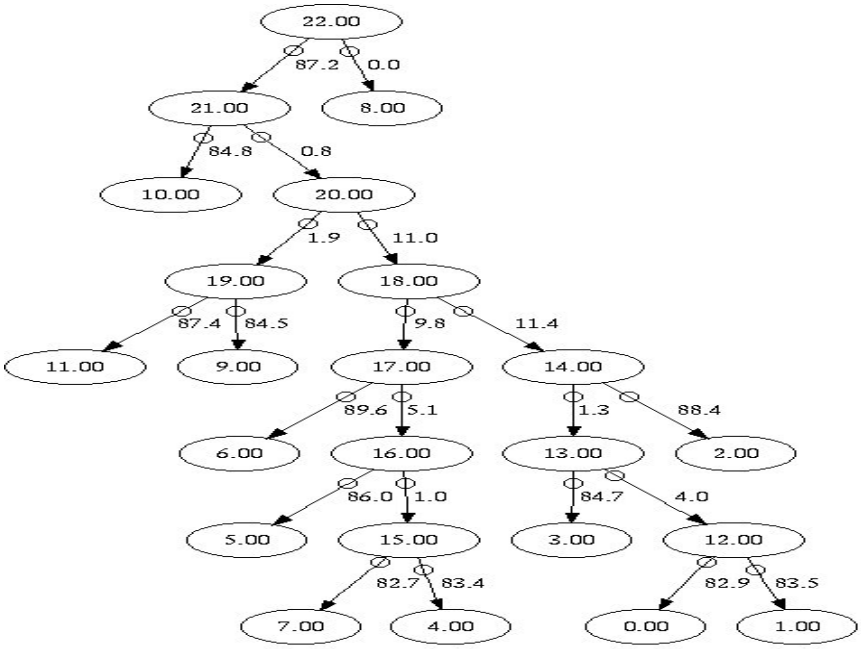

Figura 2: Ilustração de árvore gerada pelo PhGA.

Com a possibilidade de visualizações como a acima, foi mais fácil o desenvolvimento e teste do algoritmo. Além da biblioteca para grafos, foi criado os algoritmos cGA e ECGA para o aprendizado dos conceitos e fixação do aprendizado. Subseqüentemente foi implementado o PhGA. Todos os algoritmos foram programados em C/C++.

Nos resultados espera-se que o algoritmo atinga na maior parte das vezes o ótimo ou fique próximo. Outro ponto importante é a eficiência, que pode ser observada por dois fatores: número de avaliações e tempo de execução.

O número de avaliações é o número de vezes que o algoritmo testa um cromossomo verificando o seu *fitness*. Este número pode ser calculado nos exemplos a seguir multiplicando o número de gerações pelo tamanho da população. (note que na presença de alguns procedimentos como o *Hill Climbing,* este número irá aumentar). A importância de usar o mínimo número de avaliações possíveis advém do fato, que calcular o *fitness,* pode ser um processo custoso em algumas aplicações.

O tempo de execução indica a eficiência do algoritmo como um todo e por isso é importante ser observado. Há casos aonde a eficiência do algoritmo pode ser até mais importante que sua eficácia (sistema de comites de algoritmos). Para os testes são utilizados funções deceptivas por partes, que são conhecidas e normalmente utilizadas na literatura.

4.1 Comparações entre as Variações

Para os testes serão fixados algumas variáveis, estas são (note que para todos os testes foram realizados 30 execuções e tirado a média):

- População = 4000;
- Função Deceptiva por Partes, com cada parte tendo tamanho 4;
- Número de variáveis do Problema = 80 (Problema médio);
- Torneios sem reposição;
- sem *Hill Climbing;*
- população seleta igual a população inicial;
- *Crossing Over* Probabilístico.

Para esse teste o ótimo global é 20. O método de convergência utilizado foi de

4.1.1 Comparações de Heurísticas de Corte e Métricas de Distância

Caso de Teste 1.

- Métrica de Distância: *Mutual Information;*

•Heurística de Corte: *Layer Function;*

Média de Gerações: 15.6

Média de Resultados: 19.96

Média de Tempo: 12.9 segundos

Caso de Teste 2.

•Métrica de Distância: *Mutual Information;*

•Heurística de Corte: *BB Function;*

Média de Gerações: 14.43

Média de Resultados: 16.08

Média de Tempo: 11 segundos

Caso de Teste 3.

•Métrica de Distância: *Mutual Information;*

•Heurística de Corte: *Average Threshold Extended.*

Média de Gerações: 8.86

Média de Resultados: 13.22

Média de Tempo: 6.2 segundos

Caso de Teste 4.

•Métrica de Distância: Diferença entre médias;

•Heurística de Corte: *Average Threshold.*

Média de Gerações: 14.56

Média de Resultados: 18.78

Média de Tempo: 1.8 segundos

Caso de Teste 5.

•Métrica de Distância: *Mix Metric*

•Heurística de Corte: *Average Threshold Extended*

Média de Gerações: 9.93

Média de Resultados: 13.25

Média de Tempo: 6.75 segundos

Dessa maneira o mais rápido foi o Teste 4, o mais eficaz foi o Teste 1 e o que executou menos avaliações (número de gerações vezes o número da população) foi o Teste 3 (porém como o resultado dele foi abaixo do esperado, podemos desconsiderar este teste).

Os destaques de cada teste estão em negrito.

4.1.2 Comparações de variações de *Crossing Over*

Para analisar os efeitos de mudança do *Crossing Over*, será modificado este nos dois melhores exemplos anteriormente (Teste 1 e 4). Note que os testes anteriores usaram *Crossing Over* Probabilístico, então os apresentados a seguir usarão *Crossing Over* Pai e Mãe.

Teste (Reexecução do teste 1, mudando o *Crossing Over*)

Média de Gerações: 15.3 (valor original 15.6)

Média de Resultados: 19.97 (valor original 19.96)

Média de Tempo: 12.73 segundos (valor original 12.9)

4. Teste (Reexecução do teste 4, mudando o *Crossing Over*)

Média de Gerações: 13.5 (valor original 14.56)

Média de Resultados: 19.13 (valor original 18.78)

Média de Tempo: 1.63 segundos (valor original 1.8)

No Teste 1 e Test 4 houveram pequenas melhorias em todos os quesitos, tendo maior impacto no Teste 4. Pode-se dizer que o melhor método de *Crossing Over* é o de Pais e Mães.

4.1.3 Comparações de variações dos métodos de Torneios

Para analisar os efeitos de mudança nos métodos de Torneios, será modificado este nos dois melhores exemplos anteriormente (Teste 1 e 4). Note que os testes anteriores usaram Torneios sem reposição, então os apresentados a seguir usarão Torneios com reposição. O tamanho da população selecionada precisa ser necessariamante menor que a população padão, por isso utilizou-se uma população selecionada de valor igual a 10% da população padrão (o motivo disto já foiexplicado anteriormente, ao descrever os métodos de Torneio).

1. Teste (Reexecução do teste 1, mudando o método de Torneio)

Média de Gerações: 17.43 (valor original 15.6)

Média de Resultados: 19.98 (valor original 19.96)

Média de Tempo: 2.8 segundos (valor original 12.9)

4. Teste (Reexecução do teste 4, mudando o método de Torneio)

Média de Gerações: 15.93 (valor original 14.56)

Média de Resultados: 17.36 (valor original 18.78)

Média de Tempo: 0.63 segundos (valor original 1.8)

Primeiramente é importante notar que o aumento da velocidade e do número de gerações está associado ao fato de ter-se diminuído a população selecionada, pois é sobre a população selecionada que é aplicado as funções de probabilidade para encontrar os BBs. Consequentemente como esta é menor, fica mais rápido este procedimento, porém a precisão do método também é diminuída, necessitando assim de mais gerações ou até mesmo podendo acarretar na diminuição do resultado final (como pode ser observado no Teste 4). Baseado nessas premissas e nos

resultados pode-se dizer, que este Torneio deve ser utilizado para problemas nos quais a velocidade é importante e o tempo de avaliação do *fitness* é pequeno.4.2 Primeiras comparações com o ECGA

Tendo comparado as variações do algoritmo deste projeto entre si, agora deseja-se avaliar sua classificação em relação a um algoritmo já estabelecido na literatura.

Para a comparação será usado o algoritmo deste projeto com duas configurações promissoras e o ECGA desenvolvido por (Illinois *Genetic Algorithm Laboratory* - *http://www.illigal.uiuc.edu/web/*) .

4.2.1 Definição dos Problemas e Configurações

As 2 configurações do PhGA e as 3 configurações dos Problemas são:

Configuração do PhGA 1

- Métrica de Distância: *Mutual Information*
- Heurística de Corte: *Layer Function*
- Torneios sem reposição
- sem *Hill Climbing*
- população seleta igual a população inicial
- *Crossing Over* Pai e Mãe

Configuração do PhGA 2

- Métrica de Distância: Diferença entre médias
- Heurística de Corte: *Average Threshold*
- Torneios sem reposição
- sem *Hill Climbing*

•população seleta igual a população inicial

•*Crossing Over* Pai e Mãe

Configurações do Problema 1

•População = 3000

•Função Deceptiva por Partes, com cada parte tendo tamanho 4.

•Número de variáveis do Problema = 40

Configurações do Problema 2

•População = 4000

•Função Deceptiva por Partes, com cada parte tendo tamanho 4.

•Número de variáveis do Problema = 80

Configurações do Problema 3

•População = 8000

•Função Deceptiva por Partes, com cada parte tendo tamanho 4.

•Número de variáveis do Problema = 192

4.2.2 Resultados das Comparações

Problema 1:

	PhGA 1*	PhGA 2*	ECGA
Resultado(ótimo=10)	10	9.9	10
Tempo	1.73s	0.6s	2.8s
Gerações	9.93	11.53	6

*Média de 30 execuções

Problema 2:

	PhGA 1*	PhGA 2*	ECGA
Resultado(ótimo=20)	19.98	19.35	20
Tempo	12.2s	1.73s	21s
Gerações	14.36	14.1	8

*Média de 30 execuções

Problema 3:

	PhGA 1	PhGA 2*	ECGA
Resultado(ótimo=48)	42.75	41.44	47.75
Tempo	46.5s	10.1s	643s
Gerações	28	15.66	14

*Média de 30 execuções

4.3. Dificuldades e Limitações

Como este trabalho apresenta técnicas bastante complexas e a composição destas nunca antes vistas na literatura, uma grande dificuldade foi encontrar os procedimentos adequados para tornar o PhGA competitivo com os algoritmos já estudados na literatura.

O algoritmo a princípio já mostrava grandes vantagens, com velocidades extremamente rápidas, mas para aumentar a precisão um grande número de mudanças tiveram que ser consideradas e testadas. Um procedimento árduo e muitas vezes difícil de conduzir, pois sempre apareciam muitas possibilidades de resolver o problema, mas com apenas uma pessoas conduzindo a codificação, demorava sempre bastante para ver os resultados e tornava impraticável o teste de todas as possibilidades.

A abordagem adotada foi no meu ponto de vista muito boa porém árdua. O processo adotado para desenvolver tal algoritmo foi primeiramente o conhecimento da área na literatura, seguida pela construção de um algoritmo clássico e depois a criação de um algoritmo básico que

expressasse em parte nossa nova idéia, de forma com que poderíamos prosseguir incrementalmente.

4.4 Considerações Finais

O algoritmo apresenta grandes resultados e incentiva novas abordagens, como por exemplo a de mudar o algoritmo de *clustering* (*Neighbor Joining*) por um algoritmo mais atual da filogenia entre outras mudanças. Tudo, apesar de promissor é ainda muito novo.

O projeto teve algumas dificuldades, mas nada que impossibilitasse sua execução.

A seguir será apresentado minhas visões sobre sua contribuição, possíveis trabalhos futuros e considerações sobre o curso.

5. CONCLUSÕES

Este trabalho é o primeiro a tentar utilizar métodos da área de filogenia na área de algoritmos genéticos. Espero que este seja o primeiro de muitos outros trabalhos envolvendo tais princípios.

Pessoalmente, este projeto seria mais uma ponte do que uma finalidade. Tudo começou quando eu queria um algoritmo genético mais eficaz para rodar num projeto cerebral. E depois de algumas conversas o meu orientador me convenceu a implementar uma idéia que ele tinha. Mas este projeto me proporcionou o entendimento de BBs, um conceito que considero muito importante para qualquer pessoa interessada em inteligência artificial.

Este trabalho possui inúmeras possibilidades de extensão. Citando algumas delas:

- Aplicar novos algoritmos de *clustering* da filogenia para substituir o *Neighbor Joining*
- Novas métricas de distâncias
- Novas heurísticas de corte

É possível que este trabalho venha a ser usado na minha próxima pesquisa sobre comites.

6. REFERÊNCIAS

(Aggarwal, Jin & O'Reilly, 2006) V. Aggarwal, W. O. Jin, U.-M. O'Reilly, (2006), Filter Approximation Using Explicit Time and Frequency Domain Specifications, in Proceedings of Genetic And Evolutionary Computation Conference 2006, 753-760.

(Bäeck, 2006) T. Bäeck, (2006), *Evolutution Strategies*, Tutorial in Genetic and Evolutionary Computation Conference 2006.

(Bäeck, 1996), T. Bäeck, (1996), *Evolutionary algorithms in theory and practice : evolution strategies, evolutionary programming, genetic algorithms.*

(Baluja, 1994) S. Baluja, (1994), *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning*, Tech. Rep. No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA.

(Beyer, 2001) H. G. Beyer, (2001), *Theory of Evolution Strategies*.

(Chao-hong Chen, Wei-nan Liu, Ying-ping Chen 2006) Chao-hong Chen, Wei-nan Liu, Wei-nan Liu, Ying-ping Chen (2006), *Adaptive discretization for probabilistic model building genetic algorithms*, Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference GECCO-2006.

(Cheng, Bell & Liu, 1997) J. Cheng, D. A. Bell, W. Liu, (1997), *Learning belief networks from data: An information theory based approach*, in Proceedings of ACM CIKM'97. http://citeseer.ist.psu.edu/56468.html

(Coello, Veldhuizen & Lamont, 2002) C. A. C. Coello, D. A. Van Veldhuizen, G. B. Lamont, (2002), *Evolutionary Algorithms for Solving Multi-Objective Problems,* Kluwer Academic Publishers; ISBN: 0306467623.

(Darwin, 2000) Darwin, C., (2000), *A Origem das Espécies e a Seleção Natural*. Hemus S.A..

(Deb, 2001) K. Deb, (2001), *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons.

(Delbem, 2003) A. C. B Delbem, et al. (2003), *A Forest Encoding for Evolutionary Algorithms Applied to Desing Problems*. Genetic and Evolutionary Computation Conference 2003, Lecture Notes in Computer Science**,** Springer-Verlag Heidelberg, vol. 2723, p. 634-635.

(Dorigo, 2004) M. Dorigo and T. Stützle, (2004), *Ant Colony Optimization*.

(Duque, Delbem & Goldberg 2007a) DUQUE, T. S. P. C. ; SASTRY, K. ; DELBEM, A. C. B. ; GOLDBERG, D. E. . *Algoritmo Evolutivo para Solução de Problemas de Larga Escala.* In: VI ENIA - Encontro Nacional de Inteligência Artificial, 2007, Rio de Janeiro. Anais do VI ENIA, p. 1-6, 2007.

(Duque, Delbem & Goldberg 2007b) DUQUE, T. S. P. C. ; SASTRY, K. ; DELBEM, A. C. B. ; GOLDBERG, D. E. .*Evolutionary Algorithm for Large Scale Problems*. In: 7th International Conference on Intelligent Systems Design and Applications, 2007, Rio de Janeiro. IEEE Proceedings of ISDA 2007, p. 1-4, 2007.

(Epstein, 1995) R. Epstein, (1995), *The Theory of Gambling and Statistical Logic*.

(Farmer, Packard & Perelson, 1986) J.D. Farmer, N. Packard, A. Perelson, (1986), *The immune system, adaptation and machine learning*, Physica D, vol. 22, pp. 187—204

(Fogel, 1964) L. J. Fogel, (1964), *On the organization of intellect*. PhD thesis, University of California.

(Fogel, 1999), L. J. Fogel, (1999), *Intelligence through simulated evolution, forty years of evolutionary computation.*

(GECCO, 2000) D. Whitley, D. E. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, H. G. Beyer, Proceedings of the Genetic and Evolutionary Computation Conference 2000

(Gen & Cheng, 1997) M. Gen, R. Cheng, (1997), *Genetic Algorithms and Engineering Design*, Ashikaga Institute of Technology, Japan.

(Gen, Cheng & Oren, 2000) M. Gen, R. Cheng, S. S. Oren, (2000), *Network design techniques using adapted genetic algorithms*. Advances in Engineering Software.

(Goldberg, 1989) D. E. Goldberg, (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*.

(Goldberg, 1989a) D. E. Goldberg, (1989), *Genetic algorithms and Walsh Functions: Part I, a gentle introduction,* Complex Systems, 3(2), 129-152

(Goldberg, 1989b) D. E. Goldberg, (1989), *Genetic algorithms and Walsh Functions: Part II, deception and its analysis,* Complex Systems, 3(2), 129-152

(Goldberd, 2002) D. E. Goldberg, (2002), *The Design of Innovation*.

(Goldberg, Deb & Clark, 1992) D. E. Goldberg, K. Deb, Clark, (1992), *Genetic algorithms, noise, and the sizing of populations*, Complex Systems, 6 , 333{362. (Also IlliGAL Report No. 91010).

(Harik, 1997) G. R. Harik, (1997), *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algotithms*, Doctoral dissertation, University of Michigan, Ann Arbor, MI. (Também Illigal Report No. 97005).

(Harik, 1999) G. R. Harik, (1999), *Linkage Learning via probabilistic modeling in the ECGA*, Illigal Report No. 99010, Urbana, IL: University of Illinois at Urbana Champaign, Illinois Genetic Algorithm Laboratory.

(Harik & Goldberg, 2000) G. R. Harik, D. E. Goldberg, (2000), *Learning Linkage through probabilistic expression,* Computer Methods in Applied Mechanics and Engineering, 186(2-4), 295-310.

(Harik, Lobo & Goldberg, 1998) G.R. Harik, F.G. Lobo, D.E. Goldberg, (1998), *The compact genetic algorithm*, in Proceedings of the International Conference on Evolutionary Computation (ICEC'98), Piscataway, NJ, 1998, pp. 523–528.

(Heckerman, Geiger, Chickering, 1995) D. Heckerman, D. Geiger, D. M. Chickering, (1995), *Learning Bayesian networks: the combination of knowledge and statistical data,* Machine Learning, Springer.

(Holland, 1975) J. H. Holland, (1975), *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*.

(Hoolinger & Gwaltney, 2006) G. A. Hollinger, D. A. Gwaltney, (2006), *Evolutionary Design of Fault-Tolerant Analog Control for a Piezoelectric Pipe-Crawling Robot,* in Proceedings of the Genetic and Evolutionary Computation 2006, 761-768.

(Illigal, 2006) Illinois Genetic Algorithms Laboratory, (2006). Disponível em http://www-illigal.ge.uiuc.edu/index.php3

(Illinois *Genetic Algorithm Laboratory* - *http://www.illigal.uiuc.edu/web/*) - Illinois *Genetic Algorithm Laboratory, http://www.illigal.uiuc.edu/web/.*

(Jensen, 1996) F. V. Jensen, (1996), *Introduction to Bayesian Networks*, Springer-Verlag.

(De Jong, 1975) K. de Jong, 1975, *An analysis of the behavior of a class of genetic adaptive systems*.

(De Jong, Watson & Thierens 2005) E. D. de Jong, R. A. Watson and D. Thierens (2005), *On the Complexity of Hierachical Problem Solving, GECCO' 05*.

(De Jong, 2006) K. de Jong, (2006), *Evolutionary Computation, A Unified Approach*.

(Kennedy, 2001) J. Kennedy, R. C. Eberhart and Y. Shi, (2001) *Swarm Intelligence*.

(Koutsofios, North, Intset, Sparcmcemit, Joinwitharg, 1996) Eleftherios Koutsofios, Stephen C. North, Sortedlist Intset, Sparcascode Sparcmcemit, Lrparser Joinwitharg, (1996) *Drawing Graphs with Dot.*

(Koza, 1992) J. R. Koza, (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*.

(Kvasnicka, Pelikan & Pospichal, 1996) V. Kvasnicka, M. Pelikan, J. Pospichal, (1996), *Hill climbing with learning (an abstraction of genetic algorithm)*, Neural Network World, vol. 6, 773–796.

(Larranaga & Lozano, 2001) P. Larrañaga and J. A. Lozano, (2001), *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*.

(Margulis 1970) L. Margulis (1970), *Origin of Eukaryotic Cells,* Yale University Press.

(Michalewicz, 1998) Z. Michalewicz, (1998), *Genetic Algorithms + Data Structures = Evolution Programs,* Springer.

(Mühlenbein, 1997) H.Mühlenbein, (1997), *The equation for response to selection and its use for prediction*, Evolutionary Computation, vol. 5, no. 3, 303–346.

(Mühlenbein, Paaß, 1996) H. Mühlenbein, G. Paaß, (1996), *From recombination of genes to the estimation of distributions I. Binary parameters*, in Parallel Problem Solving from Nature—PPSN IV, Berlin, A. Eiben, T. Bäck, M. Shoenauer, and H. Schwefel (Eds.), 178–187.

(Nelder, Mead 1965) J. A. Nelder and R. Mead, *A simplex method for function minimization,* Computer Journal 7 (1965), 308-313.

(Papadimitriou, 1995) C. H. Papadimitriou, (1995), *Computational Complexity*, Addison-Wesley Publishing Company.

(Pelikan, 1999) M. Pelikan, (1999), *A simple implementation of the Bayesian optimization algorithm (BOA) in C++*, (Illigal Report No. 99011), Urbana, IL: University of Illinois at Urbana Champaign, Illinois Genetic Algorithm Laboratory.

(Pelikan, 2005) M. Pelikan, (2005), *Hierarchical Bayesian Optimization Algorithm*, Toward a New Generation of Evolutionary Algorithms Series: Studies in Fuzziness and Soft Computing , Vol. 170 2005, XVIII, 166 p., ISBN: 3-540-23774-7

(Pelikan & Goldberg, 2000) M. Pelikan, D. E. Goldberg, (2000), *Hierarchical problem solving and the Bayesian optimization algorithm,* in Proceedings of the Genetic and Evolutionary Computation Conference 2000, 267-274, (Também Illigal Report No. 2000002).

(Pelikan & Goldberg, 2001) M. Pelikan, D. E. Goldberg, (2001), *Escaping hierarchical traps with competent genetic algorithms,* in Proceedings of the Genetic And Evolutionary Computation Conference 2001, 511-518, (Também Illigal Report No. 2001003).

(Pelikan, Goldberg & Cantú-Paz, 1999) M. Pelikan, D. E. Goldberg, E. Cantú-Paz, (1999), *BOA: The Bayesian optimization algorithm,* in Proceedings of the Genetic And Evolutionary Computation Conference 1999: Volume 1, 524-532, (Também Illigal Report No. 99003).

(Pelikan, Goldberg & Cantú-Paz, 2000a) M. Pelikan, D. E. Goldberg, E. Cantú-Paz, (2000), *Bayesian Optimization algorithm, population sizing and time to convergence,* in Proceedings of the Genetic And Evolutionary Computation Conference 2000, 275-282, (Também Illigal Report No. 2000001).

(Pelikan, Goldberg & Cantú-Paz, 2000b) M. Pelikan, D. E. Goldberg, E. Cantú-Paz, (2000), *Linkage problem, distribution estimation and Bayesian networks,* Evolutionary Computation, 8(3), 311-340, (Também Illigal Report No. 98003).

(Pelikan, Goldberg & Lobo, 2002) M. Pelikan, D. E. Goldberg, F. Lobo, (2002), *A survey of optimization by building and using probabilistic models*, Computational Optimization and Applications, 21(1), 5-20.

(Pelikan & Goldberg 2003) M. Pelikan, D. E. Goldberg (2003), *A hierarchy machine: learning to optimize from nature and humans.* Complexity, 8(5), 36-45.

(Pelikan, Sastry & Cantú-Paz, 2006) M. Pelikan, K. Satry, E. Cantú-Paz, (2006), *Scalable optimization by probabilistic modeling*.

(Ping-chu Hung, Ying-ping Chen, Ying-ping Chen 2006), Ping-chu Hung, Ying-ping Chen, Ying-ping Chen, (2006) iECGA: Integer extended compact genetic algorithm, In Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference - GECCO-2006

(Preble, Lipson & Lipson, 2005) S. Preble, H. Lipson M, Lipson, (2005), Two-dimensional photonic crystals designed by evolutionary algorithms, in Proceedings of the Genetic and Evolutionary Computation Conference 2005

(Rechenberg, 1971) I. Rechenberg, (1971), *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Dissertation.

(Sastry, 2001), K. Sastry, (2001), *Evaluation-relaxation schemes for genetic and evolutionary algorithms,* Master's thesis, Urbana, IL: University of Illinois at Urbana Champaign, Illinois Genetic Algorithm Laboratory. (Também Illigal Report No. 2002004).

(Sastry et al., 2006), K. Sastry, D. D. Johnson, A. L. Thompson, D. E. Goldberg, T. J. Martinez, J. Leiding, J. Owens, (2006), *Multiobjective genetic algorithms for multiscaling excited state direct dynamics in photochemistry,* in Proceedings of the Genetic And Evolutionary Computation Conference 2006, 1745-1752.

(Sastry & Goldberg, 2004), K. Sastry, D. E. Goldberg, (2004), *Designing competent mutation operators via probabilistic model building of neighborhoods*, (Illigal Report No. 2004006), Urbana, IL: University of Illinois at Urbana Champaign, Illinois Genetic Algorithm Laboratory.

(Schwefel, 1977) H. P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, (1977), volume 26 of Interdisciplinary systems research. Birkhauser, Basel.

(Hammond 2007) Simon P. Hammond (2007)., *Adaptive Scaling of Evolvable Systems*, PhD thesis, University of Birmingham, United Kingdom.

(Watson 2002) R. A. Watson (2002), *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity and Symbiosis.* PhD thesis, Brandeis University.

(Whitley 1991) L. D. Whitley, (1991), *Fundamental Principles of deception in genetic search,* in Foundations of Genetic Algorithms, 221-241.