Instituto de Ciências Matemáticas e de Computação

ISSN -

Otimização Bayesiana com Detecção de Comunidades

MARCIO KASSOUF CROCOMO
ALEXANDRE CLÁUDIO BOTAZZO DELBEM

Nº

RELATÓRIO TÉCNICO

São Carlos
MAIO/2011

Resumo

Algoritmos de Estimação de Distribuição compõem uma frente de pesquisa em Computação Evolutiva que tem apresentado os resultados mais promissores principalmente para lidar com problemas complexos de larga escala. Nesse contexto, destaca-se o Algoritmo de Otimização Bayesiana (BOA) que usa um modelo probabilístico multivariado. Baseado neste, este trabalho propõe um novo algoritmo chamado de Otimização Bayesiana com Detecção de Comunidades (OBDC) que utiliza um algoritmo de detecção de estrutura de comunidades para melhorar o modelo multivariado (utilizado para orientar as operações de reprodução desse tipo de algoritmo evolutivo). Este documento detalha o funcionamento de uma implementação do algoritmo proposto e mostra experimentos com resultados favoráveis, que mostram a viablidade de se utilizar técnicas de detecção de comunidades para melhorar o modelo estatístico usado em EDAs.

Agradecimento: FAPESP (processo nº 2010/01634-5)

Sumário

Capítulo 1

Introdução

A Computação Evolutiva é a área de pesquisa que estuda o desenvolvimento dos Algoritmos Evolutivos (EAs, do inglês *Evolutionary Algorithms*). Esses algoritmos baseiam-se na teoria da evolução natural das espécies. Com o objetivo de encontrar uma solução adequada para um problema em computação, indivíduos de uma população representam soluções candidatas à resolverem o problema. Uma função de avaliação - capaz de retornar o quão boa é uma solução - é utilizada para selecionar os indivíduos mais aptos. Pela combinação das soluções mais aptas selecionadas geram-se novo indivíduos. Esses compõem uma nova população, completando uma etapa evolutiva chamada de geração. A cada nova geração, existe uma tendência de que soluções melhores sejam encontradas.

Embora baseados nesta simples ideia, EAs cresceram em diversas áreas da engenharia e ciências por mostraram-se capazes de resolver de forma eficiente problemas complexos. Como exemplo das diversas áreas de aplicação nas quais EAs obtiveram sucesso, encontram-se sistemas de redes elétricas (Mansour et al., 2010; Hadi e Rashidi, 2005), robótica (Simões e Barone, 2002; Teo et al., 2008), projetos de redes (Lima, 2009; Rajagopalan et al., 2008), jogos (Crocomo, 2008; Baba e Handa, 2007), predição de estruturas de proteínas (Lima, 2006; Cotta, 2003) entre outras.

Os EAs que se tornaram mais conhecidos foram propostos por Holland em 1975 (Holland, 1975), chamados Algoritmos Genéticos (GAs, do inglês *Genetic Algorithms*). GAs utilizam operadores reprodutivos como crossover e mutação, que combinam soluções duas a duas e alteram pontualmente soluções existentes, respectivamente, formando as soluções que compõem a nova população. Esse método obteve resultados relevantes em diversos problemas (Ogata, 2006; Moura et al., 2010; Pessin et al., 2010), difundindo esses algoritmos.

Embora GAs apresentem sucesso para uma grande variedade de problemas, pesquisadores começaram a deparar-se com casos em que a eficiência atingida pelos mesmos não era suficiente (Rana e Whitley, 1998; Harik et al., 1999). Com o objetivo de superar essas limitações, uma nova classe de EAs tem sido investigada: os Algoritmos de Estimação de Distribuição (EDAs, do inglês *Estimation of Distribution Algorithms*). Segundo estudos, os EDAs apresentam desempenho bastante superior ao dos EAs convencionais quando aplicados em vários tipos

de problemas complexos (Goldberg, 2002). Contrariamente aos GAs, que em geral combinam soluções duas a duas por meio do crossover, a ideia principal dos EDAs é a de construir um modelo probabilístico baseado nas soluções existentes na população e, então, gerar as novas soluções a cada geração a partir das informações captadas pelo modelo. O uso do modelo possibilita focar nas formas de combinação de soluções que são mais promissoras, aumentando significativamente a capacidade de solução de um EDA em relação a um EA convencional. Dessa forma, o desempenho de um EDA é diretamente dependente da qualidade do modelo probabilístico gerado.

O Algoritmo de Otimização Bayesiana (BOA, do inglês *Bayesian Optimization Algorithm*) (Pelikan et al., 1999) é um exemplo de um EDA bem sucedido, talvez o mais bem sucedido considerando-se a quantidade de problemas complexos que tem resolvido. Esse algoritmo utiliza redes Bayesianas (Niedermayer, 2009) para montar o modelo probablístico utilizado. Proposto em (Pelikan et al., 1999), o BOA aparece resolvendo difíceis problemas deceptivos linearmente separáveis (PDLSs) (PDLSs compõem classes de problemas difíceis em que os melhores EAs convencionais não conseguem resolver adequadamente). Recentemente, variações nesse algoritmo têm sido utilizadas para resolver problemas deceptivos hierárquicos (mais difíceis que os PDLS) (Pelikan e Goldberg, 2001; Pelikan et al., 2005) mostrando-se capazes de resolver difíceis problemas da literatura, como o *Isis Spin Glasses* e problemas MAXSAT (Pelikan e Goldberg, 2003), sendo este último apontado por pesquisadores como exemplo típico de problemas para os quais GAs não apresentam desempenho satisfatório (Rana e Whitley, 1998).

Neste trabalho é proposta a criação de um novo EDA, a partir da modificação do BOA. O ponto inovador da técnica proposta é o de investigar algoritmos de detecção de comunidades em Redes Complexas (Newman e Girvan, 2004) para criar melhores modelos probabilísticos de uma população para EDAs. Os algoritmos de detecção de estruturas de comunidades possuem como finalidade encontrar unidades comportamentais e funcionais e, desta forma, identificam as variáveis relacionadas em nosso problema, possibilitando a criação de um modelo probabilístico mais representativo de um certo conjunto de soluções. Com isso, pode-se gerar EDAs capazes de resolver problemas mais difíceis computacionalmente e de forma mais eficiente.

Este documento encontra-se organizado como descrito a seguir: o Capítulo 2 apresenta os principais conceitos relacionados a EDAs e explica o BOA. No Capítulo 3 são explicadas técnicas de detecção de comunidades, e a técnica implementada é descrita. No Capítulo 4, o algoritmo OBDC proposto é explicado. O Capítulo 5 mostra os experimentos realizados, assim como os resultados. Finalmente, o Capítulo 6 apresenta as conclusões do trabalho.

Capítulo 2

Algoritmos de Estimação de Distribuição

Este Capítulo apresenta os principais conceitos relacionados com EDAs, bem como o estado da arte sobre esses algoritmos. A Seção 2.1 apresenta conceitos fundamentais sobre EAs. A Seção 2.2 introduz o conceito de Blocos de Construção (BBs, do inglês *Building Blocks*) e ideias importantes para compreender o funcionamento dos EDAs. Por sua vez, esses algoritmos são explicados na Seção 2.3. Por fim, a Seção 2.4 explora o EDA no qual se baseia o algoritmo proposto: o Algoritmo de Otimização Bayesiana (BOA, do inglês *Bayesian Optimization Algorithm*).

2.1 Algoritmos Evolutivos

Os EAs são baseados em aspectos presentes na Teoria da Evolução Natural das Espécies (Darwin, 1859, 2004). Basicamente, dado um conjunto inicial de soluções, chamado de população, este tende a evoluir até convergir para soluções aceitáveis. Tal tarefa é realizada por operadores de evolução que, em geral, correpondem a dois procedimentos: recombinação e mutação (Goldberg, 1989). Esses operadores são aplicados a cada elemento da população, os quais são denominados de cromossomos (também conhecidos como indivíduos). Um cromossomo pode ser representado por um *array* de símbolos (por exemplo: *array* binário). Cada *array* é tratado como se fosse o próprio indivíduo. Em geral, cada elemento do cromossomo, chamado de gene, corresponde a uma característica (ou variável) do problema. Quando um *array* é decodificado ("traduzido"), o resultado é uma possível solução para um problema. Assim, cada novo *array* equivale a uma nova solução proposta. Em linhas gerais, um EA possui os seguintes passos:

1. É criada uma população inicial;
2. Avaliam-se as soluções da população atual, através de uma função de aptidão, que atribui a cada solução uma pontuação indicando o quão adequada é a mesma;
3. Aplica-se um procedimento de seleção para escolher os indivíduos a serem usados na criação das novas soluções que irão compor a população;

4. A partir da utilização de operadores de reprodução, as soluções selecionadas no Passo 3 são utilizadas para criar uma nova população de soluções;

5. Retorna ao Passo 2, executando o algoritmo repetidamente (de forma iterativa) até que alguma condição de parada seja atendida.

No passo 1, a população criada inicialmente é normalmente aleatória. No entanto, é possível incluir soluções com "bias"de características esperadas serem mais adequadas, caso exista conhecimento prévio sobre o problema sendo otimizado. No passo 2, é utilizada uma função de aptidão para avaliar cada solução existente na população. Dessa forma, a cada solução é atribuída uma pontuação chamada de *fitness*. No passo 3, um mecanismo de seleção é utilizado para escolher os indivíduos usados na criação de novas soluções que irão compor a próxima geração. Abaixo, encontram-se explicados alguns desses mecanismos (Jong, 2006):

- **Seleção Proporcional**: Cada indivíduo possui uma probabilidade de ser selecionado relativa a seu próprio *fitness*, definida por f_i/f_{pop}, em que f_i é o *fitness* da própria solução e e f_{pop} é a soma de todos os *fitness* da população. Com base nessa probabilidade, são selecionados todos os indivíduos necessários para o próximo passo do algoritmo;

- **Torneio de j indivíduos**: São sorteados, com igual probabilidade, j indivíduos da população. O indivíduo selecionado é o que possui maior *fitness* entre os sorteados. O processo é repetido até que todos os indivíduos necessários para o próximo passo sejam selecionados;

- **Seleção por truncamento**: São selecionados os w indivíduos de maior fitness, sendo w o número de indivíduos necessários para o próximo passo do algoritmo.

Durante o passo 4, são utilizados mecanismos de reprodução. A seguir, encontram-se explicados dois dos mecanismos mais comuns: operadores de recombinação e de mutação.

Operadores de recombinação: São responsáveis pela criação de novas soluções a partir da combinação de dois ou mais indivíduos selecionados. Um exemplo de operador de recombinação é o crossover de 1 ponto, o qual gera novas soluções a partir de dois indivíduos selecionados e da definição aleatória de um ponto de corte, quebrando o cromossomo (*array*) em duas subcadeias. As soluções resultantes são a permutação das subcadeias definidas por este ponto entre os indivíduos selecionados. A Figura 2.1 ilustra este processo para cromossomos que são *arrays* de 6 *bits*.

Outros operadores de recombinação podem ser encontrados. Dentre os mais comumente usados, podem ser destacados: o crossover de dois pontos, em que dois pontos de corte são definidos ao invés de um, trocando-se o trecho entre esses pontos; e o crossover uniforme, em que cada gene possui uma dada probabilidade de provir de uma das duas soluções selecionadas (Bäck et al., 2000).

Mutação: A mutação é responsável por inserir modificações em um ou mais genes de uma solução que irá compor a próxima geração. Diversos operadores de mutação podem ser

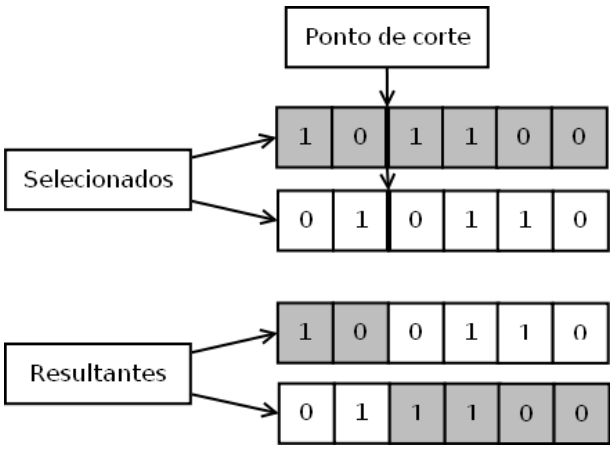

Figura 2.1: Exemplo de crossover de 1 ponto.

encontrados na literatura (Jong, 2006; Fogel, 2005). Cada gene possui uma probabilidade, de ter seu valor alterado. Essa probabilidade é chamada de taxa de mutação. A forma como o valor é modificado depende da representação do cromossomo. Se os genes são representados por valores reais, um possível operador de mutação pode adicionar ao gene um valor obtidao com base em uma distribuição probabilística (Bäck et al., 2000). Para soluções representadas por valores binários, um operador de mutação pode inverter o valor de um *bit*. A Figura 2.2 ilustra um exemplo de mutação em uma solução representada por um *array* binário. O Algoritmo 1 mostra o pseudocódigo típico de um EA.

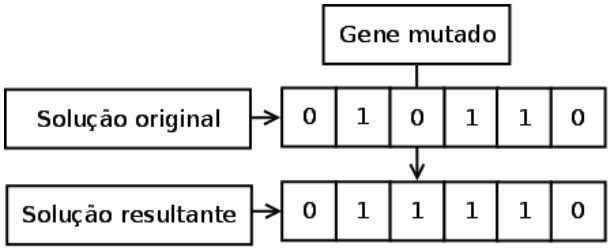

Figura 2.2: Exemplo de mutação em um *array* binário.

Algoritmo 1: Um Algoritmo Evolutivo.

```
1 InicializaPopulação(P_g);
  // avalia a aptidão dos indivíduos da população P_g
2 AvaliaPopulacao(P_g);
3 enquanto critério de parada não atingido faça
      // incrementa o contador da geração
4     g = g+1;
      // seleciona os indivíduos para a geração dos
         descendentes
5     S = Seleção(P_g-1);
      // são aplicados os operadores reprodução sobre S,
         gerando a nova população
6     P_g = Reprodução(S);
      // avalia a aptidão dos indivíduos da população
7     Avalia(P_g);
8 fim
```

Atualmente, existem diversos tipos de EAs, sendo os primeiros tipos chamados de EAs canônicos. São EAs canônicos: GAs, (Goldberg, 1989), Estratégias Evolutivas (Beyer e Arnold, 2001) e Programação Evolutiva (Bäck, 1996). Desses, os mais difundidos academicamente são os GAs, que são encontrados com eficientes resultados em diversos trabalhos na literatura (Ogata, 2006; Moura et al., 2010; Pessin et al., 2010). Em estudos recentes os EDAs têm mostrado resultados significativamente melhores que os GAs convencionais para diversos problemas complexos (Goldberg, 2002). A Seção 2.3 explica o funcionamento desses algoritmos, mas antes, a Seção 2.2 introduz conceitos importantes para se entender a questão central que motiva o desenvolvimento de EDAs.

2.2 Blocos de Construção e Problemas Difíceis

Ao se analisar a dificuldade de problemas, é preciso verificar como essa varia em escala, isto é, com o crescimento do tamanho do problema. É importante caracterizar também o tamanho do espaço de busca. Isso depende não somente do número de variáveis do problema, como em geral é apresentado em análises de complexidade computacional, mas também da cardinalidade das variáveis. Por exemplo, ao trabalhar com variáveis binárias, tem-se um espaço de busca de tamanho 2^n, onde n é o número de variáveis. Assim, para variáveis χ-árias, o espaço de busca possui tamanho χ^n.

Embora o tamanho do espaço de busca seja um fator muito importante para se determinar a dificuldade de um problema, limitantes superiores de complexidade significativamente menores que χ^n podem ser encontrados. Em (Goldberg, 2002) é verificado que dois fatores muito im-

portantes são: A quantidade m de Blocos de Construção (BBs, do inglês *Building Blocks*) e o número k de variáveis que compõem cada BB.

Para ilustrar a importância desses fatores, considere três problemas distintos, representados na Figura 2.3, todos referentes a um conjunto de 5 variáveis binárias (5 *bits*). Os problemas ilustrados são dados por funções unitárias, o que significa que a entrada para a função é um valor u que corresponde à quantidade de variáveis binárias que possuem o valor '1'. Os problemas são os seguintes:

1. Problema UmMax (Goldberg, 2002): consiste em maximizar a quantidade de variáveis iguais a 1. O *fitness* é a quantidade de variáveis com valor 1;

2. Problema de senha: *fitness* é 0 para todas as soluções, exceto para o caso em que todas as variáveis são 1, para o qual o *fitness* possui valor 5;

3. Problema armadilha (Goldberg, 2002): o *fitness* cai com o aumento de u, exceto para o maior u que possui o valor de *fitness* máximo como, por exemplo, a seguinte função armadilha de 5 *bits*:

$$ftrap5(u) = \begin{cases} 4-u & se\ u < 5, \\ 5 & caso\ contrário. \end{cases} \tag{2.1}$$

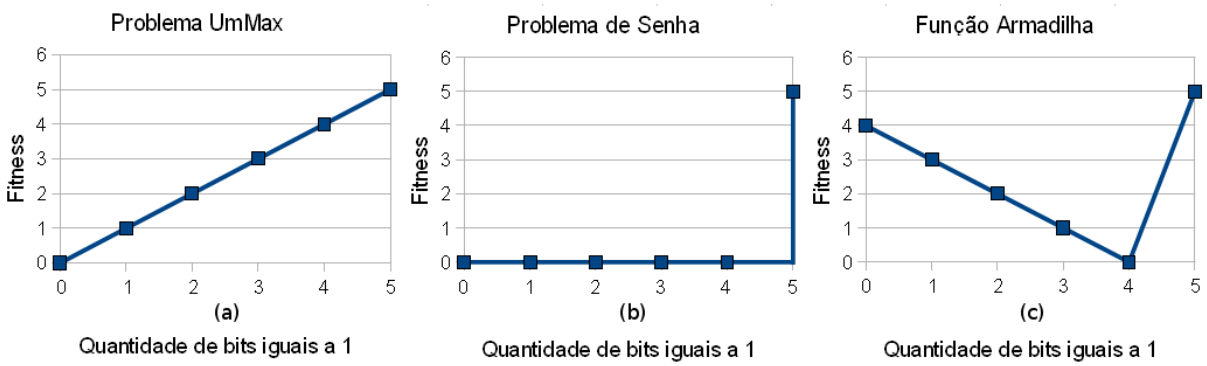

Figura 2.3: Problemas de funções unitárias: (a)UmMax, (b)Problema de senha, (c)Função armadilha.

Ao modificar o valor de um único *bit* de entrada no problema UmMax, pode-se saber se a alteração gerou uma solução mais próxima do ótimo global apenas verificando se a alteração aumentou ou diminuiu o *fitness*. Observe que isso independente do valor de qualquer uma das outras variáveis. Devido a esse fato, pode-se considerar que cada variável é, isoladamente, um BB. Assim, este problema é composto por 5 BBs de tamanho um, que são 5 subproblemas independentes. No problema de senha ou no problema de armadilha, para se saber se a modificação de um *bit* gerou uma solução mais próxima do ótimo global, é preciso saber os valores dos outros 4 *bits* de entrada. Em outras palavras, os 5 *bits* são dependentes entre si, formando um BB. Portanto, esses dois problemas possuem um único BB de tamanho 5. A partir deste ponto, é possível observar que, embora todos os problemas tenham o mesmo espaço de busca: 2^5. Os problemas com BBs de tamanho maior são mais difíceis.

No problema UmMax, algoritmos de busca local (Hoos e Stützle, 2004) são suficientes para se encontrar o ótimo global do problema (11111). No entanto, para o problema de senha, qualquer meta-heurística (Talbi, 2009) é no máximo tão eficiente quanto uma busca exaustiva ou aleatória, uma vez que não existe informação em nenhum local do espaço de busca que possa guiar o algoritmo para o ótimo global (com exceção do próprio ótimo global). Por outro lado, no problema armadilha, há informação. Com isso, meta-heurísticas que utilizem alguma informação local para orientar a busca, tendem a orientar a busca para soluções com poucos 1s, podendo a solução 00000 (ótimo local da $ftrap5$ dominar a população e impedir a descoberta do ótimo global (11111). Portanto, para essas metaheurísticas, a informação existente é na verdade desinformação, uma armadilha.

Uma forma de construir problemas mais difíceis que o problema armadilha é simplesmente pelo somatório de subproblemas em que cada um é uma função armadilha. Como exemplo, considere um problema composto pelo somatório de 3 funções $ftrap5$. Nesse exemplo, um cromossomo é formado então por um conjunto de 15 *bits*, os quais são particionados em 3 subconjuntos de 5 *bits* e uma $ftrap5$ é aplicada a cada subconjunto. O *fitness* do cromossomo é a soma dos resultados obtidos para cada $ftrap5$. Dessa forma, o problema é formado por 3 BBs de tamanho 5 ($k = 5$ e $m = 3$). Como discutido anteriormente, em funções armadilha, a busca dentro de um BB, é induzida para ótimos locais diferente do ótimo global. Uma alternativa apresentada em (Goldberg, 2002) para vencer as armadilhas é que o processo de busca não ocorra dentro dos BBs, mas entre os BBs. Em outras palavras, parte-se do princípio de que a população inicial do algoritmo possua pelo menos uma representação de cada instância possível de um BB, então o operador de recombinação utilizado não mais tem como objetivo combinar as variáveis do problema, mas sim combinar os BBs existentes na população. Reforça-se que, para isso, é necessário que a população inicial tenha uma grande diversidade de soluções provendo pelo menos uma instância ótima de cada BB na população de forma que essas instâncias possam ser combinadas gerando a solução ótima do problema.

Foi demonstrado em (Goldberg, 2002) que a Equação 2.2 estima adequadamente o tamanho da população inicial para que essa amostre com alta probabilidade, todas as possíveis instâncias de um BB.

$$n = \chi^k(k * \ln\ \chi + \ln\ m), \tag{2.2}$$

em que n é o tamanho da população, χ a cardinalidade das variáveis, k o tamanho do BB e m a quantidade de BBs. Como k é expoente positivo na Equação 2.2 e os demais termos não são exponenciais, χ^k domina a estimativa de n conforme k cresce.

A Figura 2.4 ilustra como o tamanho da população inicial necessária para que todas as possíveis instâncias dos BBs sejam representadas na população variam com k e m. As linhas contínuas são obtidas pela Equação 2.2; enquanto os pontos marcados foram obtidos experimentalmente através dos passos explicados abaixo, reproduzidos a partir dos experimentos relatados em 2.2.

1. Crie uma população inicial com 0 indivíduo;

2. Enquanto houverem instancias não representadas de algum BB na população, inclua um novo individuo aleatório;

3. Retorne o tamanho da população final (que contém todas as instâncias de cada BB).

Cada ponto representado no gráfico é obtido para um dado valor de k e de m, e é encontrado a partir de 30 execuções dos passos acima, sendo o ponto exibido dado pela média dos resultados encontrados. Dessa forma, ilustra-se teórica e experimentalmente como a complexidade de um problema aumenta conforme k e m aumentam. Além disso, a Figura 2.4 mostra claramente que o tamanho do BB é o fator mais influente a ser considerado.

No entanto, uma amostra inicial grande (contendo amostras com o ótimo global de cada sub-problema) não é o suficiente para que o ótimo global do problema seja construído. É necessário identificar previamente quais são as variáveis relacionadas que compõem os BBs, para que se possa combinar instâncias de BBs presentes em diferentes indivíduos ao longo das gerações até atingir a combinação correspondente ao ótimo global. EDAs que trabalham com esse princípio, são explicados na Seção 2.3.

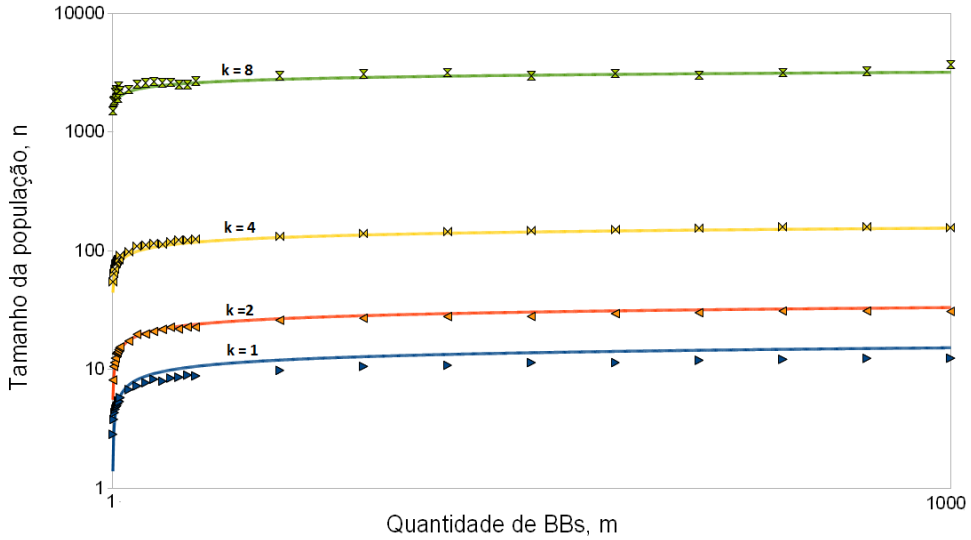

Figura 2.4: Tamanho da população inicial necessária para que exista pelo menos uma de cada possível instância de um BB.

2.3 Características dos EDAs

Um conjunto de indivíduos (ou soluções) selecionados (promissores) de uma população pode ser visto como uma amostra de dados, os quais podem ser modelados por uma função de distribuição de probabilidades. Essa função modela as relações entre variáveis presentes nas soluções mais promissoras até então encontradas. Com essa função, pode-se amostrar novos indivíduos a príncipio com altos valores de *fitness*. Dessa forma um modelo probabilístico

multivariado possibilita gerar indivíduos com grande probabilidade de terem altos valores de *fitness*. Esse é o princípio dos EDAs (Larrañaga e Lozano, 2001; Pelikan et al., 2002).

Essa técnica funciona da seguinte forma: inicialmente, gera-se um conjunto de indivíduos através de uma função de inicialização (normalmente são gerados indivíduos aleatórios). Em seguida, alguns desses indivíduos são selecionados, como em um EA comum (Ver Seção 2.1. No entanto, ao invés de gerar novas soluções por meio de operadores de reprodução como crossover e mutação, o EDA estima uma distribuição baseada nos indivíduos selecionados e, a partir dessa distribuição, gera (amostra) novos indivíduos. Posteriormente, os novos indivíduos são adicionados à população, substituindo indivíduos antigos. O processo de seleção, estimação e amostragem são repetidos até que algum critério de parada seja satisfeito. O Algoritmo 2 mostra o pseudocódigo de um EDA.

Algoritmo 2: Algoritmo de Estimação de Distribuição.

```
   // Inicializa o contador de gerações e a população incial.
1  g=1;
2  InicializaPopulação(P_g);
3  enquanto critério de parada não atingido faça
       // Avalia a aptidão dos indivíduos da população P_g.
4      AvaliaPopulacao(P_g);
       // Incrementa o contador da geração.
5      g = g+1;
       // Seleciona os indivíduos para a geração do modelo.
6      S = Seleção(P_g-1);
       // Tendo os indivíduos selecionados como entrada,
          constrói uma modelo M.
7      M = CriaModelo(S);
       // Gera a nova população utilizando o modelo construído.
8      P_g = NovaPopulacao(M);
9  fim
```

A diferença básica com relação a um EA comum, é que os EDAs substituem a aplicação de operadores reprodutivos pelos seguintes passos:

1. Construção de um modelo probabilístico que representa de forma sintética os indivíduos selecionados;

2. Geração de novos indivíduos baseada no modelo construído.

Note que os EDAs são um tipo particular de EA com um procedimento de reprodução diferenciado. Assim, sob certas condições, um EDA poderia ter comportamento equivalente ao de um EA convencional. Um exemplo é o *Compact Genetic Algorithm* (CGA), um EDA que constrói modelos monovariados (supõe BBs de tamanho 1) e possui desempenho similar ao de um GA simples (Harik et al., 1999; Mühlenbein, 1997).

É importante observar que a construção de um modelo probabilístico pode não ser uma tarefa trivial. Em geral, existe um compromisso entre a qualidade do modelo construído e a eficiência do método de construção do modelo (Pelikan et al., 2002). Assim, pode-se dizer que um dos grandes desafios de pesquisa na área de EDA é determinar o método que possui a melhor relação entre o custo computacional do modelo e a qualidade do mesmo.

A seguir são apresentadas duas linhas importantes de desenvolvimento de EDAs: o Algoritmo Genético Compacto Estendido (ECGA, do inglês *Extended Compact Genetic Algorithm*) e o BOA. Também é apresentado o Algoritmo Filogenético (PhyGA, do inglês *Phylogenetic Algorithm*), e o Filogenético com Evolução Diferencial (PhyDE, do inglês *Phylogenetic Differential Evolution*).

***Algoritmo Genético Compacto Extendido*:**

O ECGA (Harik et al., 2006) busca encontrar as variáveis relacionadas através de um processo chamado de *linkage learning*. Esse método considera que as variáveis relacionam-se em grupos (BBs) sem que estes sejam sobrepostos, isto é, considera que não existe uma ou mais variáveis pertencentes a dois grupos distintos. Uma vez obtido um modelo que representa os BBs para um problema, é possível usar operadores reprodutivos competentes (Goldberg, 2002) que não misturam valores de variáveis de instâncias de um BB de indivíduos diferentes ao combinar duas ou mais soluções. Observe que, sem o modelo de BBs, o operador de recombinação pode desagrupar combinações promissoras de valores de variáveis se o ponto de corte for interno a um BB, ou seja, pode destruir uma solução ótima de um subproblema (ver Seção 2.2).

***Algoritmo de Otimização Bayesiana*:**

A ideia central do BOA (Pelikan, 2005; Pelikan et al., 1999) é a utilização de Redes Bayesianas (BNs, do inglês *Bayesian Networks*) (Niedermayer, 2009) para representação de informações a respeito das dependências entre variáveis. As variáveis são representadas como vértices em uma rede (um grafo) e a existência de uma aresta entre dois vértices significa que o valor de uma dessas variáveis depende do valor da outra. Em outras palavras, as componentes conexas (Diestel, 2005) da rede podem ser vistas como BBs. Diferentemente do ECGA, a BN é capaz de representar sobreposição de BBs. A partir de uma BN e de indivíduos selecionados da população, é possível gerar novos indivíduos que irão compor a nova geração. O BOA é explicado em mais detalhes na Seção 2.4.

Algoritmo Filogenético:

O PhyGA (Vargas e Delbem, 2009) é um EDA cuja ideia é identificar os BBs de um problema binário uma única vez, e então realizar uma busca em cada BB separadamente. Para a identificação dos BBs, é utilizado um algoritmo de *clustering*, denominado *Neighbor Joining* (Saitou e Nei, 1987). Este algoritmo é também utilizado na biologia, para a construção de árvores filogenéticas (Felsenstein, 2003). Após a identificação dos BBs, o PhyGA realiza uma busca exaustiva em cada BB para compor a solução final.

Filogenético com Evolução Diferencial:

O PhyDE é uma extensão do PhyGA que utiliza o EA chamado Evolução Diferencial (DE, do inglês *Differential Evolution*) para encontrar os valores ótimos para os BBs, ao invés da busca

exaustiva utilizada pelo PhyGA. O algoritmo em questão vem apresentando bons resultados para problemas com BBs grandes, tendo resolvido problemas com BBs de tamanho 8 ($k = 8$), enquanto algoritmos da literatura normalmente tratam problemas com $k < 6$ (Harik et al., 2006; Pelikan et al., 1999).

2.4 Algoritmo de Otimização Bayesiana

O BOA utiliza um modelo complexo que pode identificar relacionamentos de ordem n entre os genes da população, em que n é o número de variáveis do problema. Um grande desafio desses algoritmos é justamente encontrar o melhor modelo que represente o conjunto de indivíduos selecionados da população, com o menor custo computacional possível (Emmendorfer, 2007). A Seção 2.4.2 explica um procedimento para a criação de BNs.

O BOA, proposto em (Pelikan et al., 1999), utiliza BNs para modelar as soluções promissoras e, consequentemente, orientar a exploração do espaço de busca. De maneira simplificada, a Figura 2.5 e o Algoritmo 3 ilustram o pseudocódigo do BOA.

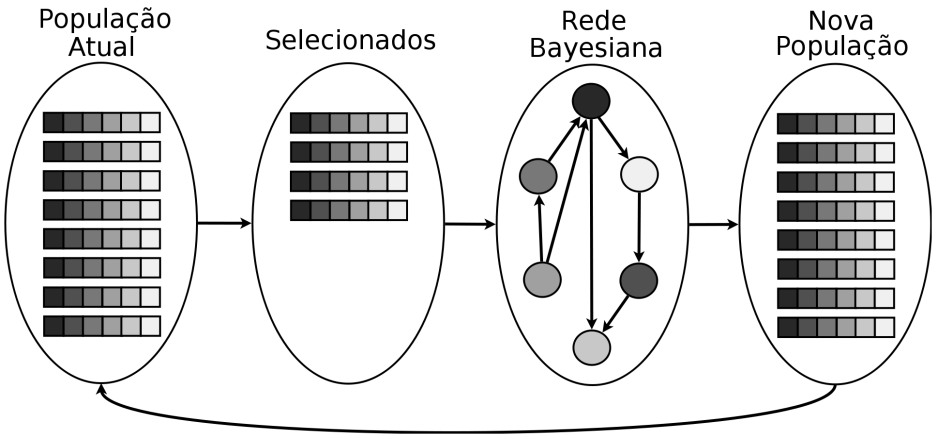

Figura 2.5: Princípio de funcionamento do BOA.

Algoritmo 3: Pseudocódigo do BOA.

```
// Inicializa o contador de gerações e a população incial.
g=1;
InicializaPopulação(P_g);
enquanto critério de parada não atingido faça
    // Avalia a aptidão dos indivíduos da população P_g.
    AvaliaPopulacao(P_g);
    // Incrementa o contador da geração.
    g = g+1;
    // Seleciona os indivíduos para a geração do modelo.
    S = Seleção(P_g-1);
    // Tendo os indivíduos selecionados como entrada,
       encontra uma BN apropriada usando uma métrica para
       avaliar cada rede considerada.
    BN = RedeBayesiana(S);
    // Gera a nova população utilizando a BN construída.
    P_g = NovaPopulacao(BN);
fim
```

2.4.1 Redes Bayesianas

Uma BN (Niedermayer, 2009; Pelikan et al., 1999; Santos, 2007) é um modelo probabilístico multivariado baseado em grafos, o qual é usado para representar as relações existentes entre as variáveis. Cada possível grafo modelando tais relações entre as variáveis é chamado de rede. Essas redes são usadas para representar dados multivariados em vários domínios de aplicação, em situações nas quais a estrutura de dependência entre as variáveis é desconhecida ou apenas parcialmente conhecida a priori.

As BNs podem ser usadas para descrever a estrutura dos dados bem como para gerar novos dados das variáveis com propriedades similares aos anteriores. Cada vértice na rede corresponde a uma variável. Tanto a variável quanto o vértice correspondente são denotados X_i. Cada variável X_i corresponde ao valor na posição de um *array* (cromossomo) que representa uma solução. O relacionamento entre duas variáveis é representado por uma aresta entre os dois vértices correspondentes. Formalmente, uma BN é um grafo acíclico com arestas orientadas às quais se associam à distribuições de probabilidades conjuntas. Isso pode ser escrito como:

$$P(X) = \prod_{i=0}^{n-1} P(X_i|\pi_i), \tag{2.3}$$

em que:

- $X = (X_0, ..., X_{n-1})$ é um vetor de variáveis e n é o tamanho do vetor;

- π_i é o conjunto das variáveis pais de X_i na rede (vértices que contém arestas apontando para X_i);
- $P(X_i|\pi_i)$ é a probabilidade de X_i condicional aos valores de seus pais (π_i).

A Equação 2.3 pode ser melhor compreendida a partir do exemplo abaixo.

Exemplo de uma BN

O exemplo apresentado a seguir é baseado no texto disponível em (Niedermayer, 2009). Considere as duas variáveis binárias relativas ao problema de prever chuva em uma região, em que:

- $X_1 = 1$ indica que "chove hoje"; caso contrário $X_1 = 0$;
- $X_2 = 1$ indica que "chove amanhã"; caso contrário $X_2 = 0$.

Para esse problema, sabe-se que a probabilidade de chover em um dia é condicional ao fato de ter ou não chovido no dia anterior. Suponha que a chance de chover em um dia seja de 20%, dado que não houve chuva no dia anterior. Caso tenha chovida no dia anterior, a probabilidade de chover é de 70%. As relações entre as variáveis X_1 e X_2 pode ser ilustrada pela Figura 2.6, ilustrando a relação de que "chover hoje" depende do fato de ter ou não chovido ontem.

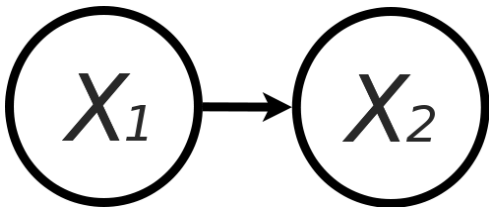

Figura 2.6: BN que mostra a relação entre duas variáveis (X_2 depende de X_1).

Supondo que ontem não choveu. Então, pode-se calcular a probabilidade de que chova hoje (X_1=1) e chova amanhã (X_2=1) utilizando o cálculo da probabilidade condicional e a informação dada pela BN da Figura 2.6, isto é, de que $\pi_2 = \{X_1\}$, obtem-se:

$P(X_1 = 1, X_2 = 1) = P(X_1 = 1) * P(X_2 = 1|X_1 = 1) = 0.20 * 0.70 = 0.14$

A Tabela de Probabilidades

A Tabela de Probabilidades indica a frequência dos valores encontrados para cada variável no grupo de indivíduos selecionados considerando as variáveis pais. Para exemplificar como construir tal tabela, considere o exemplo composto por:

1. Um problema de duas variáveis de forma que cada indivíduo pode ser representado por um *array* binário: $X = [X_1, X_2], X_i \in 0, 1$;
2. Um conjunto de 4 ndivíduos selecionados, mostrados na Tabela 2.1;
3. Uma BN como a mostrada na Figura 2.6;

Tabela 2.1: Conjunto de 4 indivíduos selecionados de uma população.

Indivíduo	X_1	X_2
1	1	1
2	1	1
3	1	0
4	0	1

4. A função $P(H)$ que retorna a frequência de ocorrências de um evento H (ver Equação 2.3).

A partir desses dados pode-se calcular as probabilidades de todos os possíveis eventos envolvendo X_1 e X_2 com base nas 4 amostras da Tabela 2.1. A Tabela 2.2 mostra tais probabilidades.

Tabela 2.2: Exemplo de uma Tabela de Probabilidades obtida a partir de uma BN (Figura 2.6) e de uma amostra de soluções promissoras (Tabela 2.1).

$P(X_1 = 0)$	$0,25$
$P(X_1 = 1)$	$0,75$
$P(X_2 = 0 \mid X_1 = 0)$	0
$P(X_2 = 1 \mid X_1 = 0)$	1
$P(X_2 = 0 \mid X_1 = 1)$	$0,33...$
$P(X_2 = 1 \mid X_1 = 1)$	$0,66...$

Para criar os indivíduos da próxima geração, utiliza-se a tabela de probabilidades. Considerando a Tabela 2.2, a chance de um indivíduo da nova geração possuir o valor 1 atribuído a sua variável X_1 é de 75% e de 25% para o valor 0. Admitindo que a variável X_1 tenha recebido o valor 1, a chance de tal indivíduo possuir o valor 0 atribuído a variável X_2 é aproximadamente de 33%, e de 67% para o valor 0.

O BOA também utiliza o conjunto de soluções promissoras (como as da Tabela 2.1) para gerar a própria BN. Para isso, é necessário um algoritmo para a criação de uma BN dado um conjunto de dados de entrada. O algoritmo K2 (Cooper e Herskovits, 1992) é um dos principais métodos para a construção de tais redes, conforme mostra a Seção 2.4.2.

2.4.2 O Algoritmo K2

Há dois componentes básicos no algoritmo que constrói a topologia de uma BN (Pearl, 1988):

1. A métrica de pontuação;

2. O procedimento de busca.

A métrica de pontuação avalia a qualidade com que a rede modela os dados. O conhecimento prévio sobre o problema também pode ser incorporado na métrica. É importante destacar que encontrar a melhor BN (que possui a melhor pontuação dada pela métrica) para um conjunto de variáveis é um problema NP (Santos, 2007). Dessa forma, busca-se na prática um procedimento que construa uma rede com o valor de métrica de pontuação tão alto quanto possível em um tempo computacional aceitável (observe que uma nova BN deve ser construída a cada geração para cada novo conjunto selecionado). A pontuação de uma BN (B_s) dado um conjunto de dados (D), chamada de $P(B_s, D)$, pode ser dada pela Equação 2.4.

$$P(B_s, D) = P(B_s) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!)} \prod_{k=1}^{r_i} N_{ijk}!, \tag{2.4}$$

em que:

- $P(B_s)$ é um fator utilizado quando se tem uma informação prévia sobre a qualidade da rede. Quando não há tal informação, este valor é 1 (valor usado neste trabalho);
- n é número de variáveis X_i;
- q_i é o número de possíveis combinações encontradas na base de dados para os valores dos pais de X_i;
- R_i é o número de possíveis valores associados à variável X_i;
- N_{ijk} é o número de casos em D, com $X_i = v_{ik}$ e $\pi_i = w_{ij}$;
- N_{ij} é o número de ocorrências que possuem o conjunto de pais de X_i instanciados como w_{ij}. Dado por: $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.

Neste trabalho, chamamos de algoritmo K2 o algoritmo que usa a métrica apresentada anteriormente e um método de busca gulosa (*greedy search*, (Cook et al., 1997)) para maximizar P(Bs|D). Assim, a contribuição de um vértice i da rede no valor da pontuação $P(Bs, D)$ (efeito local) depende do conjunto de pais deste vértice (i, π_i) e pode ser explicitado pela Equação 2.4, definida a partir da Equação 2.5:

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!. \tag{2.5}$$

O pseudocódigo do algoritmo K2 (descrito no Algoritmo 4, adaptado de (Cooper e Herskovits, 1992)) pode ser explicado como segue. Partindo de um grafo sem arestas, uma busca gulosa é utilizada para, a cada etapa, escolher uma nova aresta que será inserida na rede. A modificação na pontuação da BN pela inserção desta aresta, é dada pela Equação 2.5. Esta equação torna-se uma complicação, pois, o uso de fatoriais produz números muito grandes, podendo causar *overflow* quando o algoritmo é executado. Portanto, deve-se utilizar a forma logarítmica da expressão, substituindo $g(i, \pi_i)$ por $log(g(i, \pi_i))$, conforme mostrado pela Equação 2.6.

$$log(g(i,\pi_i)) = \sum_{j=1}^{q_i} ln((r_i-1)!) - ln((N_{ij}+r_i-1)!) + \sum_{k=1}^{r_i} N_{ijk}!. \tag{2.6}$$

Algoritmo 4: Pseudocódigo do algoritmo K2.

Entrada: Um conjunto de n vértices ordenados, limite superior v (número máximo de pais por vértice), base de dados D, contendo t casos.

Saída: Para cada vértice, a identificação de seus pais.

para $i = 1$ *até* t **faça**
  π_i = {};
  `// Função calculada usando a fórmula logaritmica, dada pela Equação 2.6`
  P_{old} = g(i,π_i);
  OKparaProceder = sim;
  enquanto *OKparaProceder e* |π|<v **faça**
    `// z aponta para o vértice que, entre todos os vértices que antecedem` X_i `dada a ordenação de entrada, maximiza g(`i`,`$\pi_i \cup \{z\}$`)`
    z = maxPred(X_i);
    P_{new} = g(i,$\pi_i \cup \{z\}$);
    se $P_{new} > P_{old}$ **então**
      $P_{old} = P_{new}$;
      $\pi_i = \pi_i \cup z$;
    fim
    senão OKparaProceder = não;
  fim
  Escreva("Vértice:"X_i, "Pais deste vértice:", π_i);
fim

Capítulo 3

Detecção de Estruturas de Comunidades

Devido ao fato de pessoas, em geral, dividirem-se em grupos de acordo com seus interesses, trabalho, idade e outras características, redes sociais normalmente encontram-se subdivididas em uma estrutura de comunidades (Newman e Girvan, 2004). Essas podem ser formalmente representadas por grupos de vértices de uma rede que são densamente conectados entre si, e esparsamente conectados com o resto da rede, que contém as outras comunidades. A Figura 3.1 ilustra uma possível estrutura de comunidades para uma rede.

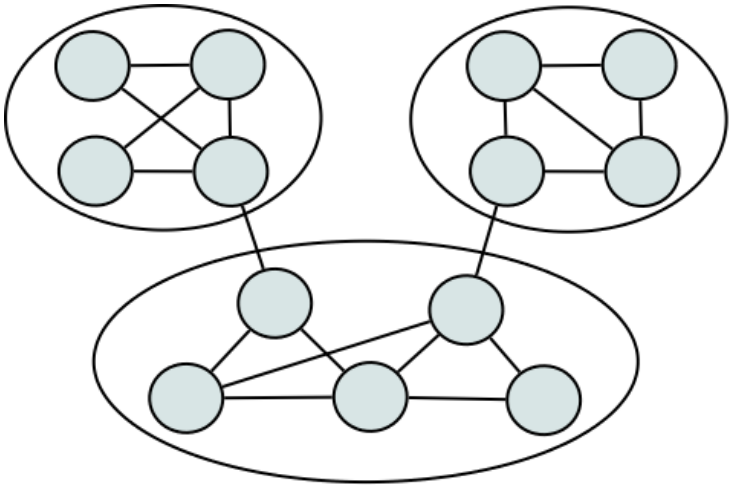

Figura 3.1: Exemplo de estrutura com três comunidades em uma rede.

Comunidades são de grande interesse, pois, geralmente correspondem a unidades comportamentais e funcionais. Devido a isso, ferramentas matemáticas e algoritmos têm sido pesquisados para detectar estruturas de comunidades na área de Redes Complexas (Donetti e Muñoz, 2004; Duch e Arenas, 2005). É importante ressaltar que algoritmos que verificam todas as possíveis estruturas de comunidade em uma dada rede são NP (Duch e Arenas, 2005) e, portanto, inviáveis de serem utilizados para redes grandes. Por isso, a literatura recente apresenta diversas novas técnicas que buscam encontrar a melhor estrutura de comunidade possível em um tempo aceitável.

Dessas técnicas, pode-se destacar as baseadas na métrica Q, definida por Newman (Newman e Girvan, 2004) que representa a modularidade de uma divisão em comunidades. Quanto maior

o valor de Q, melhor a estrutura de comunidades encontrada. A partir da modularidade, diversos pesquisadores têm proposto o uso de algoritmos de otimização para encontrar estruturas de comunidades. Entre esses, podem-se destacar o *Fast Algorithm* (FA) (Newman e Girvan, 2004) (Seção 3.1), o Adaptive Clustering (adClust) (Ye et al., 2008) e o Otimização Extrema (EO, do inglês *Extreme Optimization*) (Duch e Arenas, 2005)).

Os algoritmos citados podem ser comparados considerando dois diferentes fatores: (i) O valor de modularidade encontrado e (ii) o tempo de execução. Com relação ao primeiro item, os artigos que apresentam os algoritmos criados após o FA (adClust e EO) realizam comparações com o mesmo, em que conseguem obter melhores resultados. Dentre os três algoritmos estudados, os melhores resultados referentes a modularidade são obtidos pelo adClust, como mostra a Tabela 3.1, que apresenta os valores de modularidade obtidos pelos algoritmos para dois problemas de *benchmark* (Zachary, 1977; Newman, 2001). No entanto, com relação ao tempo de execução, tanto o adClust quanto o EO exigem mais tempo do que o necessário pelo FA, como apontado pelos autores das técnicas (Duch e Arenas, 2005; Ye et al., 2008).

Tabela 3.1: Comparação dos valores de modularidade Q obtidos pelos algoritmos EO, adClust e FA para duas redes diferentes. Resultados extraídos de (Duch e Arenas, 2005) e (Ye et al., 2008).

Redes	Q_{FA}	Q_{EO}	$Q_{adClust}$
Zachary (Zachary, 1977)	0.3810	0.4188	0.4198
Rede de cientistas (Newman, 2001)	0.6683	0.6790	0.7610

No algoritmo OBDC proposto neste trabalho, é necessário executar um algoritmo de detecção de comunidades a cada nova geração. Desta forma, é de interesse que o algoritmo escolhido seja de rápida execução. Devido a este fato e também a sua simples implementação, o algoritmo escolhido para implementação foi o FA, que encontra-se explicado na Seção 3.1. Por outro lado, é importante ressaltar que qualquer algoritmo de detecção de comunidades pode ser utilizado na implementação do OBDC, sendo inclusive o adClust um bom candidato, pois, permite atingir melhores valores de modularidade.

3.1 *Fast Algorithm.*

O FA é baseado na ideia de modularidade, que é uma métrica para qualificar as estruturas das comunidades (Newman e Girvan, 2004). Para isso, define-se o termo e_{ij} como a metade do número de arestas que conectam as comunidades i e j em relação ao total de arestas da rede. Dessa forma, $e_{ij} + e_{ji}$ corresponde ao total de arestas que conectam os vértices de ambas as comunidades. Já e_{ii} corresponde ao número de arestas dentro da comunidade i em relação ao total de arestas. Um algoritmo de detecção de comunidades deve maximizar a fração de arestas que conectam vértices de uma mesma comunidade, ou seja, maximizar $\sum_i e_{ii}$. Entretanto, essa medida não avalia bem a qualidade das comunidades, uma vez que seu valor máximo é facil-

mente atingido quando todos os vértices da rede pertencem à mesma comunidade. Para isso, mais um componente é utilizado: $a_i = \sum_j e_{ij}$, a fração das arestas conectadas a pelo menos um vértice da comunidade i. Com isso, é define-se o índice de modularidade Q ponderando entre ambas frações conforme mostra a Equação 3.1.

$$Q = \sum_i (e_{ii} - a_i^2). \tag{3.1}$$

De posse de um índice para quantificar a qualidade das comunidades, pode-se construir um algoritmo que otimize o valor de Q sobre todas as possíveis divisões da rede em comunidades. Foi desenvolvido em (Newman, 2004) um algoritmo guloso para esse fim. Inicialmente, esse algoritmo considera cada vértice da rede como uma comunidade. Repetidamente, as comunidades são agrupadas em pares considerando todos os pares possíveis. Então, o agrupamento com maior valor de Q é preservado. O Algoritmo 5 apresenta o funcionamento do FA.

Algoritmo 5: *Fast Algorithm*

```
1 Separe os vértices em n módulos;
2 enquanto número de comunidades > 1 faça
3 |   Junte as duas comunidades i e j cuja aglomeração forneça o maior acréscimo (ou
  |   menor decréscimo) no valor da modularidade;
4 |   Salve a divisão resultante e sua modularidade;
5 fim
6 Escolha a divisão com a melhor modularidade encontrada;
```

É importante ressaltar que apenas comunidades que possuam pelo menos uma aresta ligando seus vértices podem ser possivelmente unidas pelo algoritmo. Isso limita a um máximo de p pares de comunidades, onde p é o número de arestas do grafo. A variação em Q ao se unir duas comunidades é dada pela equação 3.2.

$$\Delta Q = e_{ij} + e_{ji} - 2a_i a_j = 2(e_{ij} - a_i a_j) \tag{3.2}$$

O valor inicial de e_{ij} é igual à metade do grau de cada vértice, uma vez que inicialmente cada comunidade é formada por apenas um vértice. Após a junção de duas comunidades os valores de e_{ij} devem ser atualizados. Por fim, vale destacar que o FA proposto em (Newman, 2003) quando aplicado em redes de pequena escala, possui um desempenho similar ao algoritmo proposto em (Newman e Girvan, 2004). Porém, para redes de larga-escala, o FA possui desempenho superior.

Capítulo 4

Otimização Bayesiana com Detecção de Comunidades

Esse novo EDA consiste basicamente em uma modificação substancial do BOA, em que se utiliza um algoritmo de detecção de comunidades como, por exemplo, o FA, com o objetivo de construir melhores modelos probabilísticos. Dessa forma, esse novo algoritmo possui mais dois novos passos em relação ao BOA original:

1. Aplica o algoritmo de detecção de comunidades sobre a rede Bayesiana para identificar as possíveis relações entre variáveis. Lembrando que cada comunidade representa um grupo de variáveis relacionadas (um BB);

2. Adiciona conexões entre arestas de uma mesma comunidade. Com isso, é possível aumentar a força de relação entre as variáveis dentro de um mesmo BB. Esse é um ponto chave, pois a BN obtida é gerada a partir de uma base de dados amostrada (conjunto de indivíduos selecionados pelo EDA) e, portanto, algumas relações entre variáveis em geral podem não ser bem amostradas conforme k e m aumentam (ver Seção 2.2) ou algumas relações podem não ser representadas na BN uma vez que a busca gulosa usada pelos algoritmos de construção da BN não analisam todas as potenciais redes, gerando uma rede subótima, isto é, com imperfeições. Dessa forma, a detecção de comunidades funciona como uma técnica de reparo efetuando algumas mudanças que possibilitam construir BNs mais confiáveis.

Vale destacar que, nas comunidades encontradas, são adicionadas arestas até um máximo de v_f (grau incidente máximo possível para um vértice) arestas por vértices para que não ocorra *overflow* de memória. Observe que durante o processo de criação da Tabela de Probabilidades (ver Seção 2.4) do BOA, a mesma terá um tamanho máximo de 2^{v_f} para cada um dos vértices, assim, é crucial a restrição v_f para manter o algoritmo de detecção de comunidades eficiente na construção da BN.

O algoritmo OBDC é ilustrado na Figura 4.1 e sintetizado no Algoritmo 6.

Algoritmo 6: Otimização Bayesiana com Detecção de Comunidades

1. 1. Gere uma população aleatória inicial;
2. **enquanto** *critério de parada não atingido* **faça**
3. 2. Selecione as boas soluções da população atual;
4. 3. Construa uma rede Bayesiana usando uma métrica escolhida;
5. 4. Use um algoritmo de detecção de estrutura de comunidades sobre a Rede Bayesiana (BN);
6. 5. Adicione conexões entre arestas de uma mesma comunidade;
7. 6. Substitua parte da população original por novas soluções criadas a partir da BN;
8. **fim**

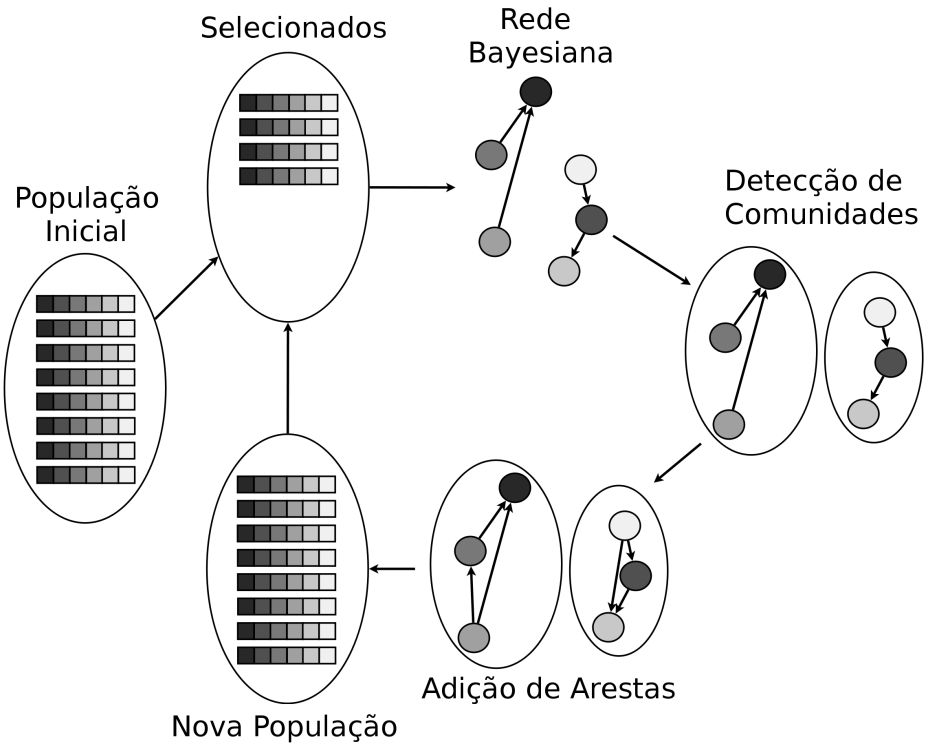

Figura 4.1: Ilustração do funcionamento do algoritmo OBDC.

Capítulo 5

Experimentos

Para testar a eficiência das versões dos algoritmos BOA e OBDC implementados, foram realizados os seguintes testes: o conjunto de variáveis que se procura otimizar é representado por um *array* binário composto por m BBs de tamanho 5 (k=5) a serem aplicados em funções $ftrap5$ (ver Seção 2.2). Foram realizados 30 testes para cada tamanho de problema (ℓ) utilizado: 30, 60 e 120. O tamanho da população (n) utilizada para cada um desses casos foi, respectivamente, de 1300, 3000 e 9000 indivíduos (determinados empiricamente). Os aspectos comparados entre estas duas técnicas são os seguintes:

- Taxa de acerto de BBs, dada pelo número de BBs para os quais foram encontrados o máximo global, dividido pelo total de BBs do problema;
- Número de avaliações até que o critério de convergência tenha sido atingido. Esse critério é o mais utilizado para se medir a eficiencia de um EDA. É admitido que a operação mais custosa é a avaliação de cada solução. Quanto menor for o número de avaliações realizadas para se obter uma boa solução, mais eficiente é o algoritmo.

Os resultados mostrados a seguir foram todos obtidos realizando uma média de 30 experimentos. Os Algoritmos testados foram:

1. BOA, com parâmetro v igual a 2;
2. BOA, com parâmetro v igual a 4;
3. OBDC, parâmetro v igual a 2, e v_f igual a 4.

O valor ideal de v (número máximo de variáveis pais associadas, ver Seção 2.4.2) a ser utilizado para resolver problemas $ftrap5$ é 4, dado que uma variável encontra-se associada à outras 4 variáveis. O parametro $v = 2$ foi escolhido por não permitir a criação de uma BN que identifique todas as relações existentes em uma $ftrap5$. Dessa forma, busca-se verificar a capacidade da técnica de detecção de comunidades utilizada pelo OBDC em identificar corretamente os BBs a partir de uma BN em que nem todas as conexões entre as variáveis estão

representadas. A Figura 5.1 mostra a comparação entre o número de avaliações necessárias para convergir utilizando o BOA com $v = 2$ e utilizando o OBDC com $v_f = 4$ (número máximo de variáveis pais associadas após a estrutura de comunidades ser detectada, ver Seção 4).

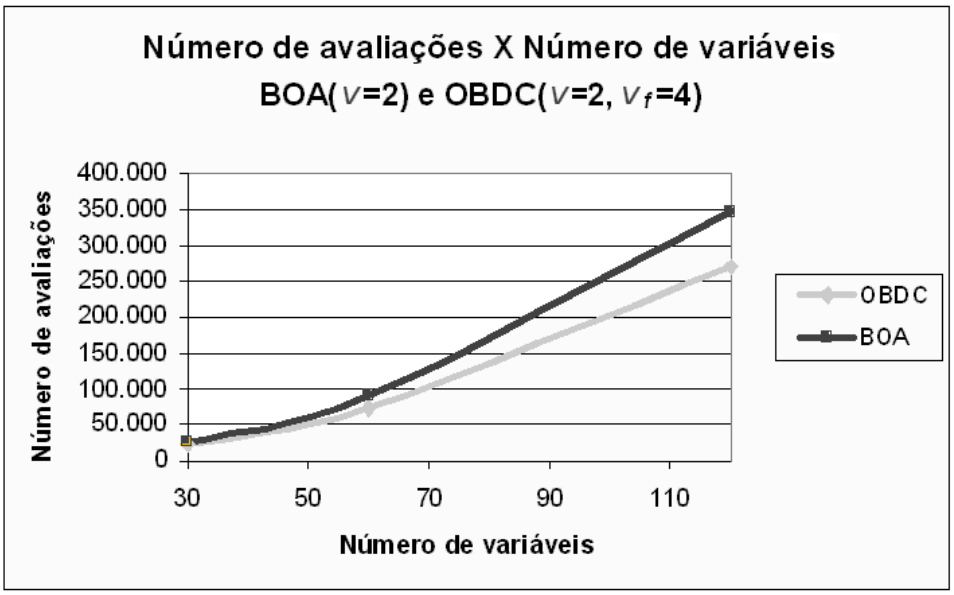

Figura 5.1: Comparação entre BOA com $v = 2$ e OBDC com $v = 2$ e $v_f = 4$.

Os resultados favorecem o uso do ODBC, pois o número de gerações necessárias foi sempre menor quando utilizando a nova técnica proposta. Observa-se também que a vantagem do OBDC aumenta com o tamanho (ℓ) do problema. A Figura 5.2 mostra a fração de BBs, para o qual o ótimo global foi atingido ao se utilizar ambas as técnicas. Mais uma vez, é possível notar que a técnica proposta mostrou-se mais eficiente, atingindo em todos os casos uma maior taxa de acerto quando comparada à técnica BOA.

Os experimentos foram repetidos utilizando $v = 4$ com o BOA, que é um parâmetro mais adequado para se tratar problemas $ftrap5$. As Figuras 5.3 e 5.4 mostram a comparação dos resultados. É importante notar que ao utilizar a técnica FA, o parâmetro v utilizado para o OBDC continuou sendo 2. As Figuras 5.3 e 5.4 mostram que, embora similares, os resultados da nova técnica proposta foi superior à técnica BOA em todos os casos. Tal comportamento mostra a capacidade do algoritmo de detecção de estrutura de comunidade em detectar os BBs a partir da BN gerada pelo algoritmo K2, mesmo quando essa BN não indentifica todas as relações existentes entre as variáveis.

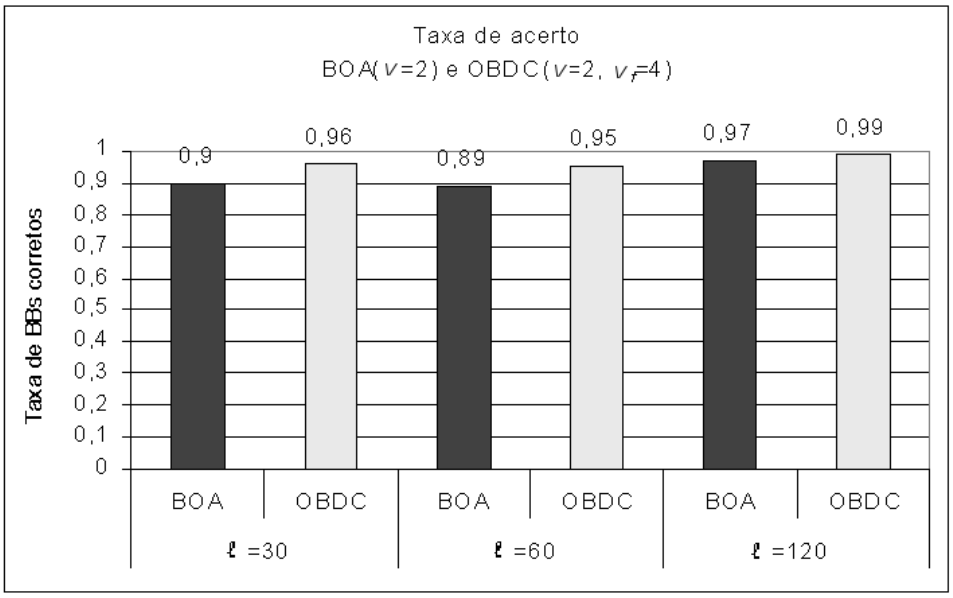

Figura 5.2: Proporção de BBs que atingiram o máximo local utilizando a técnica BOA, com $v = 2$ e a técnica OBDC, com $v = 2$ e $v_f = 4$. Os Resultados apresentados foram obtidos realizando a média de 30 testes, em problemas de tamanho 30, 60 e 120.

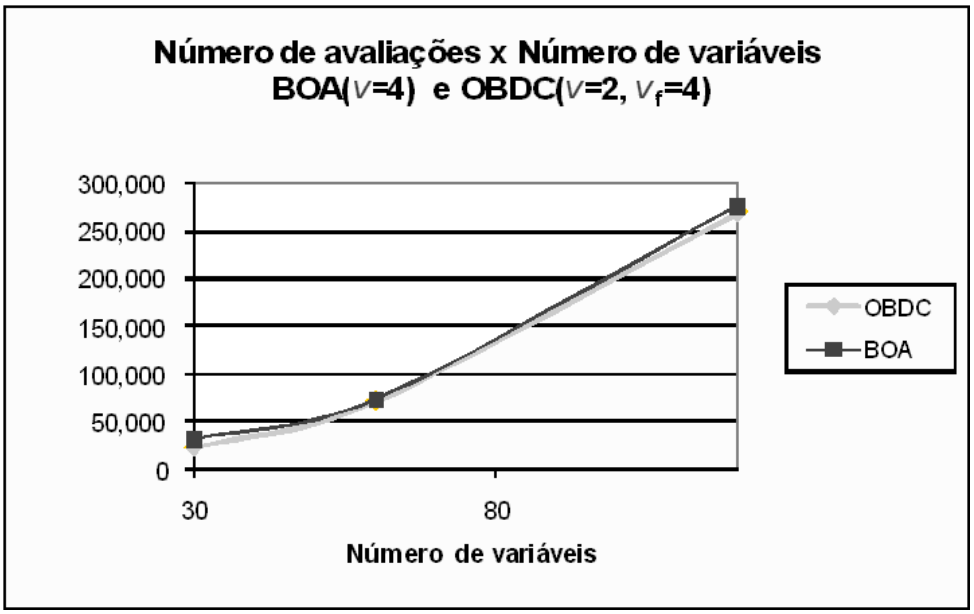

Figura 5.3: Número de avaliações necessárias até que a convergência seja atingida (95% dos *bits* em cada posição seja o mesmo em cada membro da população). Comparação entre BOA com $v = 4$ e OBDC com $v = 2$ e $v_f = 4$.

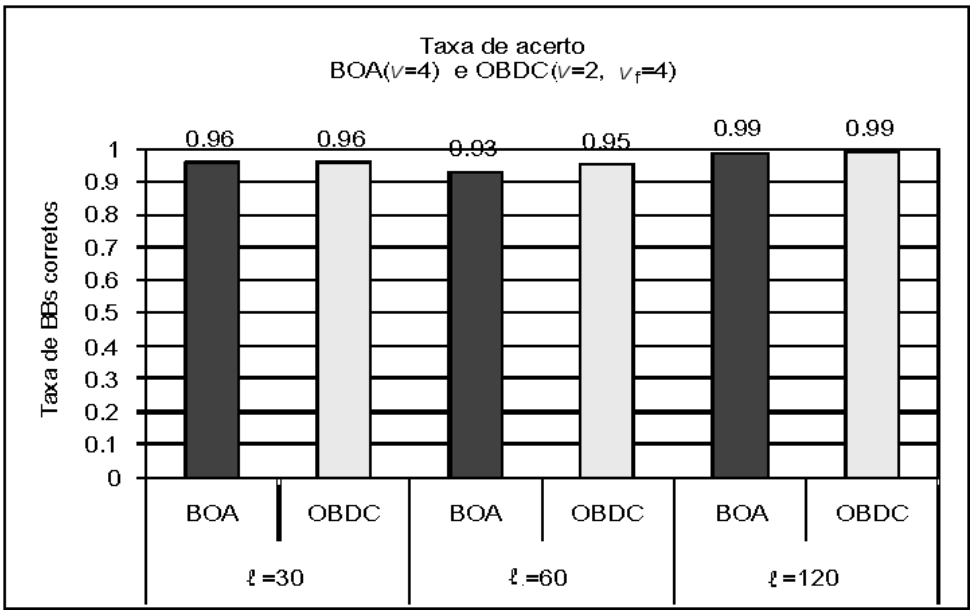

Figura 5.4: Proporção de BBs que atingiram o máximo local utilizando a técnica BOA, com $v = 4$ e a técnica OBDC, com $v = 2$ e $v_f = 4$. Os Resultados apresentados foram obtidos realizando a média de 30 testes, em problemas de tamanho 30, 60 e 120.

Capítulo 6

Conclusões

Os resultados mostram que é possível utilizar técnicas de detecção de estruturas de comunidades para gerar um melhor modelo a ser utilizado em um novo EDA. Ao utilizar um parametro v = 2 para o algoritmo K2 apresentado em um problema com 5 variáveis associadas por BB, limitamos a qualidade do modelo encontrado por não permitir que 4 variáveis sejam diretamente associadas a uma quinta variável, como espera-se no problema $ftrap5$ apresentado. Ao invés disso, com o parâmetro utilizado, a BN encontrada apresenta no máximo as 2 variáveis mais fortemente relacionadas a uma terceira. No entanto, a técnica de detecção de comunidades quando aplicada à BN encontrada desta forma, permite identificar comunidades que identificam conjuntos de variáveis relacionadas não limitadas pelo parâmetro v. Esta informação adicional pode ser utilizada na resolução do problema como, por exemplo, melhorando a BN encontrada, como foi feito neste trabalho.

Como em problemas do mundo real o tamanho dos BBs não são normalmente conhecidos a priori, é possível estimar um limitante inferior para o parâmetro v, e deixar que a técnica de detecção de estrutura de comunidades estime o número de variáveis associadas, representada pelo tamanho das comunidades encontradas, colaborando com a construção do modelo probabilístico a ser utilizado. Esta é a principal contribuição deste trabalho para o campo de computação evolutiva.

Possíveis continuações deste trabalho, são:

1. **Estudar o uso de algoritmos de detecção de comunidades com sobreposição**, isto é, onde uma variável pode pertencer a mais do que uma comunidade. Desta forma, é possível encontrar informações adicionais para problemas onde diferentes BBs possuam variáveis comuns.

2. **Realizar experimentos comparando o algoritmo OBDC e o BOA assumindo que o tamanho dos BBs não é conhecido a priori**, como normalmente ocorre em problemas do mundo real. É esperado que os resultados obtidos pelo OBDC sejam mais robustos que os obtidos pelo BOA ao se variar o tamanho dos BBs do problema sendo tratado. Ou

seja, a qualidade da solução encontrada é menos afetada, já que o OBDC permite estimar um intervalo de valores aceitáveis para o tamanho do BB.

3. **Realizar experimentos para problemas que contenham BBs de tamanhos diferentes**. Neste caso, também espera-se encontrar vantagens com a utilização do OBDC, pelos mesmos motivos apontados no item anterior. É esperado que os resultados obtidos pelo OBDC sejam mais robustos que os obtidos pelo BOA ao se variar o tamanho dos BBs do problema sendo tratado. Ou seja, a qualidade da solução encontrada é menos afetada, já que o OBDC permite estimar um intervalo de valores aceitáveis para o tamanho do BB.

Referências Bibliográficas

BABA, N.; HANDA, H. Commons Game Made More Exciting by an Intelligent Utilization of the Two Evolutionary Algorithms. In: BABA, N.; JAIN, L. C.; HANDA, H., eds. *Advanced Intelligent Paradigms in Computer Games*, New York: Springer. (Studies in Computational Intelligence, 71), p. 1–16, 2007.

BÄCK, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford: Oxford University Press, 328 p., 1996.

BÄCK, T.; FOGEL, D.; MICHALEWICZ, Z., eds. *Evolutionary computation 1: Basic algorithms and operators*. Bristol: Institute of Physics Publishing, 339 p., 2000.

BEYER, H. G.; ARNOLD, D. V. *Theory of evolution strategies: A tutorial* London: Springer-Verlag, p. 109–133, 2001.

COOK, W. J.; CUNNINGHAM, W. H.; PULLEYBLANK, W. R.; SCHRIJVER, A. *Combinatorial optimization*. New York: John Wiley, 1997.

COOPER, G. F.; HERSKOVITS, E. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, v. 9, p. 309–347, 1992.

COTTA, C. Protein structure prediction using evolutionary algorithms hybridized with backtracking. In: *International Work-Conference on Artificial and Natural Neural Networks: Part II: Artificial Neural Nets Problem Solving Methods*, IWANN '03, Berlin: Springer-Verlag, 2003, p. 321–328 (*IWANN '03*,).

CROCOMO, M. K. *Um algoritmo evolutivo para aprendizado on-line em jogos eletrônicos*. Dissertação de Mestrado, Universidade de São Paulo, São Carlos, 2008.

DARWIN, C. *On the origin of species*. London: John Murray, 502 p., 1859.

DARWIN, C. *A origem das espécies*. 1 ed. Brasil: Martin Claret, 640 p., 2004.

DIESTEL, R. *Graph Theory*, v. 173 de *Graduate Texts in Mathematics*. 3 ed. Springer-Verlag, Heidelberg, 2005.
Disponível em: <http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/GraphTheoryIII.pdf> (Acessado em 08 fev. 2011)

DONETTI, L.; MUÑOZ, M. A. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, v. 2004, n. 10, p. P10012, 2004.

DUCH, J.; ARENAS, A. Community detection in complex networks using extremal optimization. *Physical Review E*, v. 72, n. 2, p. 027104+, 2005.

EMMENDORFER, L. R. *Aprendizado da ligação entre genes em computação evolutiva: Uma nova abordagem baseada em estatísticas de baixa ordem*. Tese de Doutoramento, Universidade Federal do Paraná, 2007.

FELSENSTEIN, J. *Inferring phylogenies*. Connecticut: Sinauer Associates, 664 p., 2003.

FOGEL, D. B. *Evolutionary computation : Toward a new philosophy of machine intelligence*. IEEE Press, 296 p., 2005.

GOLDBERG, D. E. *Genetic algorithms in search, optimization, and machine learning*. Boston: Addison-Wesley Professional, 432 p., 1989.

GOLDBERG, D. E. *The design of innovation: Lessons from and for competent genetic algorithms*. Norwell: Kluwer Academic Publishers, 272 p., 2002.

HADI, A.; RASHIDI, F. Design of optimal power distribution networks using multiobjective genetic algorithm. In: *Advances in Artificial Intelligence*, New York: Springer, 2005, p. 203–215.

HARIK, G. R.; LOBO, F. G.; GOLDBERG, D. E. The compact genetic algorithm. *IEEE Trans. Evolutionary Computation*, v. 3, n. 4, p. 287–297, 1999.

HARIK, G. R.; LOBO, F. G.; SASTRY, K. Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ecga). In: *Scalable Optimization via Probabilistic Modeling*, p. 39–61, 2006.

HOLLAND, J. H. *Adaptation in natural and artificial systems*. Michigan: The University of Michigan Press, 1975.

HOOS, H. H.; STÜTZLE, T. *Stochastic local search: Foundations & applications*. Burlington: Elsevier: Morgan Kaufmann, 672 p., 2004.

JONG, K. A. D. *Evolutionary computation : A unified approach*. Cambridge: MIT Press, 256 p., 2006.

LARRAÑAGA, P.; LOZANO, J. A. *Estimation of distribution algorithms: A new tool for evolutionary computation (genetic algorithms and evolutionary computation)*. New York: Springer, 382 p., 2001.

LIMA, T. W. D. *Algoritmos evolutivos para predição de estruturas de proteínas*. Dissertação de Mestrado, Universidade de São Paulo, São Carlos, 2006.

LIMA, T. W. D. *Estruturas de dados eficientes para algoritmos evolutivos aplicados a projeto de redes*. Tese de Doutoramento, Universidade de São Paulo, São Carlos, 2009.

MANSOUR, M. R.; SANTOS, A. C.; J. B. A, L.; B., D. A. C.; BRETAS, N. G. Representação nó-profundidade e algoritmos evolutivos aplicados ao problema de restabelecimento de energia em sistemas de distribuição de energia elétrica. In: *Congresso Brasileiro de Automática*, Bonito, 2010, p. 1215 – 1221.

MOURA, A.; RIJO, R.; SILVA, P.; CRESPO, S. A multi-objective genetic algorithm applied to autonomous underwater vehicles for sewage outfall plume dispersion observations. *Applied Software Computing*, v. 10, n. 4, p. 1119–1126, 2010.

MÜHLENBEIN, H. The equation for response to selection and its use for prediction. *Evolutionary Computation*, v. 5, n. 3, p. 303–346, 1997.

NEWMAN, M. E. J. The structure of scientific collaboration networks. 2001.
Disponível em: <http://www.pubmedcentral.gov/articlerender.fcgi?artid=14598> (Acessado em 09 fev. 2011)

NEWMAN, M. E. J. Fast algorithm for detecting community structure in networks. 2003.
Disponível em: <http://arxiv.org/abs/cond-mat/0309508> (Acessado em 09 fev. 2011)

NEWMAN, M. E. J.; GIRVAN, M. Finding and evaluating community structure in networks. *Physical Review E*, v. 69, n. 2, p. 026113+, 2004.
Disponível em: <http://dx.doi.org/10.1103/PhysRevE.69.026113> (Acessado em 09 fev. 2011)

NIEDERMAYER, D. An introduction to bayesian networks. 2009.
Disponível em: <http://www.niedermayer.ca/papers/bayesian/index.html> (Acessado em 09 fev. 2011)

OGATA, A. K. O. *Multialinhamento de seqüências biológicas utilizando algoritmos genéticos*. Dissertação de Mestrado, Universidade de São Paulo, São Carlos, 2006.

PEARL, J. *Probabilistic reasoning in intelligent systems:networks of plausible inference*. Burlington: Morgan Kaufmann, 552 p., 1988.

PELIKAN, M. *Hierarchical bayesian optimization algorithm: Toward a new generation of evolutionary algorithms*. Studies in Fuzziness and Soft Computing, 1 ed. New York: Springer, 2005.

PELIKAN, M.; GOLDBERG, D. E. Escaping hierarchical traps with competent genetic algorithms. In: *Genetic and Evolutionary Computation Conference (GECCO2001)*, San Francisco: Morgan Kaufmann, 2001, p. 511–518.

PELIKAN, M.; GOLDBERG, D. E. Hierarchical boa solves ising spin glasses and maxsat. In: *Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartII*, GECCO'03, Berlin: Springer-Verlag, 2003, p. 1271–1282 (*GECCO'03*,).
Disponível em: <http://portal.acm.org/citation.cfm?id=1756582.1756586> (Acessado em 09 fev. 2011)

PELIKAN, M.; GOLDBERG, D. E.; CANTÚ-PAZ, E. Boa: The bayesian optimization algorithm. In: *Conference on Genetic and Evolutionary Computation*, Orlando: Morgan Kaufmann, 1999, p. 525–532.

PELIKAN, M.; GOLDBERG, D. E.; LOBO, F. G. A survey of optimization by building and using probabilistic models. *Comput. Optim. Appl.*, v. 21, n. 1, p. 5–20, 2002.

PELIKAN, M.; SASTRY, K.; GOLDBERG, D. E. Multiobjective hboa, clustering, and scalability. In: *Conference on Genetic and evolutionary computation*, GECCO '05, New York: ACM, 2005, p. 663–670 (*GECCO '05*,).

PESSIN, G.; OSÓRIO, F. S.; WOLF, D. F.; BRASIL, C. R. S. Improving efficiency of a genetic algorithm applied to multi-robot tactic operation. In: MORALES, Á. F. K.; SIMARI, G. R., eds. *Advances in Artificial Intelligence, 12th Ibero-American Conference on AI, Bahía Blanca, Argentina, November 1-5, 2010. Proceedings*, Springer, 2010, p. 50–59 (*Lecture Notes in Computer Science*, v.6433).

RAJAGOPALAN, R.; MOHAN, C. K.; MEHROTRA, K. G.; VARSHNEY, P. K. Multi-Objective Evolutionary Algorithms for Sensor Network Design. In: BUI, L. T.; ALAM, S., eds. *Multi-Objective Optimization in Computational Intelligence: Theory and Practice*, Hershey: Information Science Reference, p. 208–238, 2008.

RANA, S.; WHITLEY, D. Genetic algorithm behavior in the MAXSAT domain. Berlin: Springer, 1998, p. 785–794.

SAITOU, N.; NEI, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, v. 4, n. 4, p. 406–425, 1987.
Disponível em: <http://mbe.oxfordjournals.org/content/4/4/406.abstract> (Acessado em 09 fev. 2011)

SANTOS, E. B. *A ordenação das variáveis no processo de otimização de classificadores bayesianos: Uma abordagem evolutiva*. Dissertação de Mestrado, Universidade Federal de São Carlos, São Carlos, 2007.

SIMÕES, E. V.; BARONE, D. A. C. Predation: An approach to improving the evolution of real robots with a distributed evolutionary controller. In: *ICRA*, IEEE, 2002, p. 664–669.

TALBI, E.-G. *Metaheuristics : from design to implementation*, v. 10 de *The Sciences Po series in international relations and political economy*. San Francisco: John Wiley & Sons, 1–6 p., 2009.

TEO, J.; NERI, L. D.; NGUYEN, M. H.; ABBASS, H. A. Walking with EMO: Multi-Objective Robotics for Evolving Two, Four, and Six–Legged Locomotion. In: BUI, L. T.; ALAM, S., eds. *Multi-Objective Optimization in Computational Intelligence: Theory and Practice*, Hershey: Information Science Reference, p. 300–332, 2008.

VARGAS, D. V.; DELBEM, A. C. B. *Algoritmo filogenético*. Relatório Técnico, Universidade de São Paulo, 2009.

YE, Z.; HU, S.; YU, J. Adaptive clustering algorithm for community detection in complex networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, v. 78, n. 4, p. 046115+, 2008.

ZACHARY, W. W. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, v. 33, p. 452–473, 1977.