Ciência & Engenharia (Science & Engineering Journal) ISSN 1983-4071
23 (1): 47 – 56, jan. – jun. 2014

ALGORITMO DE ESTIMAÇÃO DE DISTRIBUIÇÃO ASSOCIADO A ESQUEMAS DE MEMÓRIA EM ESPAÇOS DE BUSCA DINÂMICOS

ESTIMATION DISTRIBUTION ALGORITHM ASSOCIATED WITH MEMORY SCHEMES IN DYNAMIC SEARCH SPACES

Rubens Barbosa Filho
Professor do Curso de Ciência da Computação da Universidade Estadual de Mato Grosso do Sul, MS, Brasil
E-mail: rubens@comp.uems.br

RESUMO

Os Algoritmos Evolucionários (AE) são alvos de grandes estudos quanto às suas utilizações em problemas de otimização dinâmica. Esta observação reflete a real importância dos AE nas aplicações do mundo real. Muitos estudos têm sido desenvolvidos no intuito de se aprofundar em direção aos esquemas de memória. Este trabalho investiga a aplicação do método Multi-PBIL em espaços de busca dinâmicos, com a utilização de esquemas de memória. O Multi-PBIL é um algoritmo de otimização pertencente à classe de algoritmos evolucionários. Neste trabalho, realizou-se uma mudança no Multi-PBIL com o objetivo de se avaliar a adaptabilidade em ambientes dinâmicos. Nos esquemas de memória, os modelos de probabilidade são armazenados na memória conjuntamente com as melhores amostras criadas em um espaço de busca e são usados para reativar ambientes antigos a partir do momento em que ocorrer mudanças. São realizados estudos experimentais sobre uma série de ambientes dinâmicos onde são mostradas a eficiência e a adaptabilidade do método proposto. O trabalho apresenta também a relação entre os esquemas de memória e as multi populações em ambientes dinâmicos.
Palavras-Chave: Algoritmo de Estimação de Distribuição, Problema de Otimização Dinâmica, Esquemas de memória, Multi-PBIL, Computação Evolucionária.

ABSTRACT

The Evolutionary Algorithms (EA) are target of major studies regarding their use in dynamic optimization problems. This observation reflects the real importance of AE in real world applications. Many studies have been developed in order to have a better understand over the memory schemes. This work investigates the application of Multi-PBIL method on dynamic search spaces, using memory schemes. The Multi-PBIL is an optimization algorithm that belongs to the class of evolutionary algorithms. An adaptation is proposed to improve its adaptability in dynamic environments. In memory schemes, the probability models are stored in memory, along with the best samples created in a search space and they are used to reactivate old environments from the point when changes occur. Experimental studies based on a series of dynamic environments shows the efficiency of the memory scheme for the proposed method. This work presents also the relationship between the memory scheme and the multi-population in dynamic environments.
Keywords: Estimation Distribution Algorithm, Dynamic Optimization Problem, Scheme Memory, Multi-PBIL, Evolutionary computation.

1 – INTRODUÇÃO

Os Algoritmos de Estimação de Distribuição (EAD) são meta-heurísticas inspiradas nos princípios da seleção natural e, têm sido usados amplamente na solução de problemas de otimização. Porém, em problemas de otimização do mundo real onde os ambientes são dinâmicos, a função de aptidão, as variáveis e as condições gerais costumam mudar constantemente com o tempo.

Esta configuração mutante representa um desafio aos algoritmos evolucionários, pois estes não se adaptam facilmente a este tipo de ambiente. Atualmente, existem muitos estudos que abordam problemas reais dinâmicos. Uma lista bem diversificada pode ser encontrada em (BACK, 1998). Um conjunto considerável de estudos apresenta como metas a necessidade de se categorizar os problemas dinâmicos. Segundo (BRANKE, 2002) os problemas dinâmicos são categorizados em quatro tipos: (1) Aumento de diversidade após uma mudança no ambiente; (2) Mantendo a diversidade durante a execução; (3) Esquemas baseados em memória e (4) Abordagens multi populacionais.

Neste trabalho, o esquema baseado em memória é investigado com a aplicação do método Multi-PBIL (BARBOSA, 2005). Esta pesquisa é semelhante ao trabalho desenvolvido por Yang e Yao (2008), porém, enquanto estes trabalharam com sistemas de memória cíclicos, esta adaptação ao Multi-PBIL utiliza sistemas não cíclicos. Por se trabalhar com esquemas de memória, uma adaptação ao algoritmo original do Multi-PBIL foi necessária com o objetivo de melhorar a adaptabilidade em ambientes dinâmicos. Dentro deste esquema, a melhor amostra criada pelo modelo de

probabilidade é armazenada na memória em um determinado tempo e espaço. Quando uma mudança no ambiente (espaço de busca) é detectada, o modelo de probabilidade responsável por aquele ponto da memória que está sendo reavaliado é classificado como sendo o melhor, e posteriormente este ponto de memória será comparado com outros pontos armazenados em iterações futuras. Usando um gerador de problemas dinâmicos proposto nos trabalhos de Yang (2003) e, Yang e Yao (2005) foi possível construir uma série de ambientes de testes dinâmicos, a partir de duas funções estacionárias e, estudos experimentais foram feitos com o objetivo de se avaliar o desempenho realizado pelo Multi-PBIL. Estes estudos visaram a possibilidade de se validar a eficiência do esquema de memória quando aplicado ao Multi-PBIL em problemas de otimização dinâmica. Este trabalho analisa também a relação entre memória e esquemas de multi populações no Multi-PBIL em ambientes dinâmicos.

Este trabalho está organizado com as seguintes seções: a seção 2 apresenta os Algoritmos de Estimação de Distribuição; a seção 3 mostra o algoritmo PBIL; a seção 4 apresenta o método Multi-PBIL; a seção 5 mostra a junção do Multi-PBIL em conjunto com a distribuição Gaussiana; a seção 6 mostra os esquemas de memória em ambientes dinâmicos; a seção 7 mostra o algoritmo PBIL em conjunto com o esquema de memória; a seção 8 mostra o Multi-PBIL em conjunto com o esquema de memória; a seção 9 apresenta os ambientes de teste; a seção 10 mostra os estudos experimentais; a seção 11 apresenta os resultados obtidos e análises e a seção 12 apresentam as conclusões do trabalho.

2 – ALGORITMOS DE ESTIMAÇÃO DE DISTRIBUIÇÃO

Os AED são algoritmos heurísticos de otimização que baseiam sua busca no caráter estocástico. Semelhante aos Algoritmos Genéticos (AG), os AED também são baseados em populações que evoluem. Entretanto, a diferença entre os AG e os AED é que nestes a evolução das populações não se dá por meio de operadores de cruzamento. No lugar dos operadores de cruzamento, a nova população de indivíduos provém de uma distribuição de probabilidade, a qual é estimada na base de dados contendo o conjunto de indivíduos selecionados que constituíam a geração anterior.

Enquanto em outras heurísticas utilizadas em Computação Evolucionária, as inter-relações entre as variáveis que representam os indivíduos são mantidas implicitamente (*building block hypothesis*), nos AED as inter-relações são expressas explicitamente por meio de associações entre a distribuição de probabilidade e os indivíduos selecionados em cada interação (BENGOETXEA, 2002).

De fato, a tarefa de estimar a distribuição de probabilidade associada com a base de dados contendo os indivíduos selecionados a partir de gerações anteriores, constitui o trabalho mais difícil de executar.

Entre as vantagens desse novo tipo de algoritmo, está a ideia de tornar mais fácil a predição de movimentos da população no espaço de busca. Este tipo de algoritmo também se baseia na evolução da população, onde esta evolução representa um progresso na busca pela solução. Vale ressaltar que ambos possuem uma fundamentação teórica muito bem centrada na Teoria de Probabilidade. Desta forma, pode-se resumir que os AED são algoritmos de busca de base populacional baseados em um modelo probabilístico de soluções promissoras, que utiliza a simulação de modelos introduzidos para guiar o processo de busca (BENGOETXEA, 2002).

3 – POPULATION-BASED INCREMENTAL LEARNING (PBIL)

O PBIL (*Population-Based Incremental Learning*) pertence ao grupo dos AED. Neste grupo, as variáveis são interpretadas como sendo variáveis independentes, isto é, variáveis que não possuem interações significativas entre si. Em sua origem, o PBIL surgiu da combinação de Otimização Evolucionária e *Hillclimbing* (BALUJA, 1994).

O objetivo PBIL é utilizar um vetor de probabilidade composto de valores reais, que permita encontrar, por meio de uma amostra populacional, altos valores de aptidão de vetores soluções com alta probabilidade.

As fases de execução do PBIL são similares aos passos de execução de um AG. As diferenças estão no fato do PBIL utilizar vetores de probabilidade para descrever as populações, não fazer uso da técnica de crossover e, usar Aprendizagem Competitiva (BALUJA, 1994).

Enquanto os AG geram uma população implícita de indivíduos partindo-se de uma geração atual para uma geração seguinte, o PBIL cria as suas populações de indivíduos com base somente no vetor de probabilidade. É importante destacar que apenas o vetor de probabilidade é mantido durante as gerações. Em outras palavras, mais do que manter uma população de potenciais soluções, o PBIL mantém o modelo de probabilidade das regiões promissoras do espaço de busca (BALUJA, 1994).

4 – O MÉTODO MULTI-PBIL

O Multi-PBil é um algoritmo evolucionário baseado em modelos que não apresentam dependências entre variáveis. O modelo probabilístico usado é um vetor de valores reais onde cada elemento desse vetor representa, independentemente, a probabilidade de se gerar o valor 1 ou o valor 0 em cada posição gênica de cada indivíduo. Um vetor de probabilidade é simplesmente uma sequência de probabilidades.

Contextualizando, a representação básica de uma solução em um algoritmo Multi-PBIL tradicional pode ser a mesma solução de um algoritmo genético, porém, ao invés do algoritmo Multi-PBIL guardar cada possibilidade explicitamente, a população é substituída por uma distribuição de probabilidade.

O Multi-PBIL possui características que fazem dele um método robusto e eficiente. Uma delas é a possibilidade de se criar quantos modelos de probabilidade forem necessários para

uma busca eficiente em um espaço multimodal. Outra característica importante é que o método permite estabelecer as regras de inicialização dos modelos de probabilidade com valores de zero a um e, isto, significa poder inicializar os modelos próximos aos objetivos procurados pelo problema.

4.1 Inicialização da população inicial

A inicialização da população inicial dos Algoritmos de Estimação de Distribuição é feita por meio de uma função matemática obedecendo a um formato específico. Cada indivíduo possui um valor de aptidão o qual representa o seu potencial. E, é por meio destes valores que os indivíduos devem ser ordenados decrescentemente.

4.2 Criação dos modelos de probabilidade

O método Multi-PBIL utiliza o conjunto de indivíduos da população inicial para criar os modelos de probabilidade. Obrigatoriamente, o primeiro indivíduo da população é usado para criar o primeiro modelo de probabilidade.

Os modelos de probabilidade são criados por meio da Equação 1:

$$M_i = (1- \alpha) * X_i + \alpha * (\text{média da metade dos melhores indivíduos}) \quad (1)$$

Em que: M_i é o valor binário da posição corrente **i** no modelo criado; α é o parâmetro de aprendizagem; X_i é o valor binário da posição corrente do indivíduo.

O critério usado para criar os modelos subsequentes segue a seguinte estrutura condicional: a partir do segundo indivíduo da população até o último, realiza-se um teste com o objetivo de se verificar se tais indivíduos estão aptos a também criar os modelos de probabilidade seguintes.

A estrutura condicional utiliza um parâmetro chamado Fator de Criação (Fc) para verificar a probabilidade de cada indivíduo ser usado para criar um novo modelo de probabilidade. O Fc representa um termômetro na criação de poucos ou muitos modelos de probabilidade, isto é, quanto maior for o valor do fator de criação, maior será o número de modelos de probabilidade criados. E, inversamente, quanto menor for o valor do fator de criação, menor será o número de modelos criados.

Um teste comparativo mede a proximidade entre os indivíduos utilizando a distância de *Hamming*. Para cada indivíduo X_i da população, com $i > 1$, realiza-se um teste para detectar se esse indivíduo em destaque pode gerar o próximo modelo de probabilidade.

No algoritmo da Figura 1 é apresentada a sub-rotina responsável pela criação dos modelos.

Figura 1 – Sub-rotina responsável pela criação dos modelos de probabilidade.

```
para indivíduo Xi == 2 até N faça
  se máximo(Prob(Xi | M(k)), k = 1,...,Ni) < Fc então
    cria-se um novo modelo de probabilidade;
  senão xi + 1;
fim_para
```

Uma distribuição Gaussiana sobre um espaço de busca tem por objetivo aplicar o produto de um conjunto Gaussiano multidimensional para cada modelo probabilístico.

Este modelo probabilístico inicia-se com uma distribuição geral mais equilibrada, onde o modelo probabilístico da equação Gaussiana é posicionado no centro do espaço de busca (TAKAHASHI e MARTINS, 2005). A cada geração, o modelo probabilístico é atualizado combinando o melhor indivíduo, o segundo melhor e o pior indivíduo. A equação 2 apresenta a equação de distribuição Gaussiana (onde **X** representa o indivíduo e **t** o tempo).

$$X^{t+1} = (1 - \alpha) * X^t + \alpha * (X^{melhor,1} + X^{melhor,2} - X^{pior}) \quad (2)$$

A aplicação da distribuição Gaussiana no Multi-PBIL introduz um novo parâmetro **δ** chamado desvio padrão univariante Gaussiano. O valor de **δ** permite determinar a diversidade populacional, em outras palavras, um valor pequeno de **δ** restringe os indivíduos a uma pequena área em torno do modelo probabilístico, enquanto um valor grande de **δ** abrange uma área maior.

Em ambos os casos, todos os pontos no espaço de busca apresentam uma chance de serem amostrados, com probabilidades concentradas em torno do valor de **δ**. Assim sendo, em teoria, o Multi-PBIL baseado em distribuição Gaussiana apresenta todas as propriedades de uma otimização global. Na prática, o valor de **δ** deve ser não tão grande, porque sendo o tamanho da população limitado, associada com um grande **δ**, implicaria em uma distribuição de uma população limitada por um grande espaço de busca.

Como resultado, as informações coletadas sobre as estruturas do problema pelos indivíduos podem ser imprecisas. Em relação às regras de atualização, ou taxa de aprendizagem, somente uma pequena fração da população é escolhida para atualizar o modelo de probabilidade. Entretanto, cada indivíduo da população pode fornecer alguma informação a respeito do cenário estrutural.

O ponto convergente para esse tipo de situação é usar um valor pequeno para a taxa de aprendizagem em uma posição A e, um valor maior de taxa de aprendizagem em uma posição B, sendo ambas equidistantes no espaço de busca.

A estratégia utilizada na taxa de aprendizagem possui o seguinte formato:

1. Inicializa a taxa de aprendizagem com um valor pequeno **p**;

2. Se a direção do movimento da geração corrente permanece a mesma da geração anterior, então aumente a taxa de aprendizagem em direção a 1,0:
   a. $\alpha_{(t+1)} = \alpha_{(t)} * (1 + q)$ para $(q > 0)$ (bônus) (3)
   b. Caso contrário, volte ao valor inicial de **p**:
   $\alpha_{(t+1)} = p.$ (penalidade) (4)

O conceito inserido nesta estratégia, diz que, se houver uma oscilação do modelo probabilístico, então é porque há múltiplos atratores nas áreas vizinhas. A movimentação do Multi-PBIL em uma mesma direção durante gerações consecutivas permite inferir que deve existir somente um atrator, e que um aumento gradual na taxa de aprendizagem trará um movimento acelerado e seguro. Em relação ao valor do parâmetro **q**, o qual controla o quão rápido se quer aumentar a taxa de aprendizagem, pode-se predefinir uma constante tal como **0,1** ou **0,2**, ou mesmo decidir por uma heurística dinâmica.

5.1 Etapas do Método Multi-PBIL

Cada modelo de probabilidade criado dá origem a uma população de indivíduos. Cada população de indivíduos tem como objetivo encontrar o ponto ótimo procurado por meio do espaço de busca.

Cada modelo faz sua atualização independente dos demais modelos, isto é, o modelo $\mathbf{M_1}$ seleciona um indivíduo de sua população utilizando um método de seleção, para ser usado na sua atualização; o modelo $\mathbf{M_2}$ seleciona outro indivíduo de sua população para usar em sua atualização; e esse processo de atualização segue até que todos os modelos tenham sido atualizados.

O ciclo de etapas do método Multi-PBIL é composto dos seguintes passos:

1. Geração dos indivíduos;
2. Avaliação da aptidão de cada indivíduo, baseada em uma função de aptidão de um problema em particular;
3. Criação dos modelos de probabilidade;
4. Para cada modelo de probabilidade criado:
   a. Inicializar os modelos de probabilidade por meio da equação (1);
   b. Gerar uma população de **N** indivíduos para cada modelo de probabilidade e, aleatoriamente determinar o gene de cada posição de cada indivíduo (0 ou 1);
   c. Avaliar a aptidão de cada indivíduo;
   d. Ordenar os indivíduos pelo valor de aptidão;
   e. Atualizar os valores probabilísticos de cada posição do modelo, baseado em um conjunto de indivíduos selecionados por uma função de seleção e utilizando uma taxa de aprendizagem;
   f. Verificar se o modelo de probabilidade converge. Se não converge, repetir os passos de (b) até (e).

6 – ESQUEMAS DE MEMÓRIA EM AMBIENTES DINÂMICOS

A aplicação de esquemas de memória em ambientes dinâmicos mostrou ser capaz de aprimorar o desempenho de algoritmos evolucionários. O princípio básico dos esquemas de memória reside na capacidade de poder armazenar informações (boas soluções) a partir do ambiente corrente em execução e, reutilizá-las posteriormente em novos ambientes. Segundo Branke (1999) a informação pode ser armazenada utilizando-se dois mecanismos: memória implícita ou memória explícita.

Para o esquema de memória implícita, os algoritmos evolucionários utilizam a representação do genótipo, o qual armazena informação redundante, podendo ser uma boa solução, que poderá ser reutilizada posteriormente. Neste caso, a representação redundante funciona como uma memória que, implicitamente os algoritmos evolucionários podem reutiliza-la de forma apropriada. Exemplos de esquemas memória implícita são os algoritmos genéticos diploides ou a representação multiplóide. Primeiramente, Goldberg e Smith (1987) estenderam de simples algoritmos genéticos haploides para algoritmos genéticos diploides com esquemas dominantes tri-alelos. Em seguida, Ng e Wong (1997) propuseram o esquema dominante com quatro alelos para algoritmos genéticos diploides. Posteriormente, Lewis (1998) investigou a adição de esquemas diploide em cenários onde os genes tornam-se 1 se a adição de todos os alelos excederem determinado limiar e 0 caso contrário. Yang e Yao (2005) propuseram um PBIL dual para problemas dinâmicos inspirados no dualismo presente na natureza. No PBIL dual, um vetor de probabilidade dual compete com o vetor de probabilidade principal na geração de amostras. O PBIL dual provou ter um desempenho muito bom em ambientes dinâmicos com significativas mudanças no espaço do genótipo.

Enquanto o esquema de memória implícita depende da representação redundante para armazenar informações úteis aos algoritmos evolucionários, para que estes possam explorar ambientes dinâmicos durante a execução, os esquemas de memória explícita utilizam representações precisas, onde um espaço de armazenamento extra é criado para armazenar as informações úteis a partir da geração corrente. Estas informações podem ser armazenadas e reutilizadas em gerações ou ambientes futuros. O esquema de memória explícita trabalha sobre três questões: O que armazenar na memória? Como organizar e atualizar a memória? E como recuperar a informação da memória?

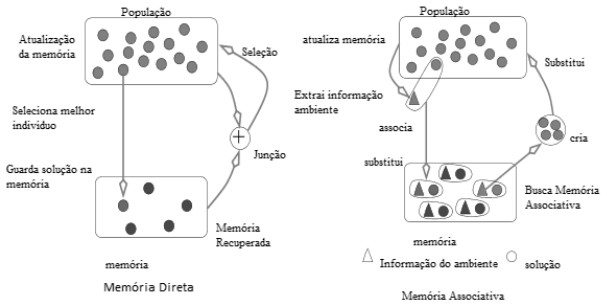
Para a primeira questão, a escolha natural é armazenar boas soluções e reutilizá-las quando uma mudança no ambiente for detectada. Este procedimento é chamado Esquema de Memória Direta e foi exemplificado no trabalho de Louis e Xu (1996), onde os autores estudaram o problema do re-agendamento da loja aberta (*open shop re-scheduling problem*). Neste problema, sempre que ocorre uma mudança, o AG é reiniciado a partir de uma população parcial (5 a

10%) de indivíduos herdados da última execução, enquanto os indivíduos restantes são gerados randomicamente. Os autores reportam avanços significativos de seu AG sobre outras implantações onde a população total é reiniciada randomicamente.

Ao invés de armazenar somente as boas soluções, as informações que permitem associar boas soluções com seus respectivos espaços de busca, também podem ser classificadas como sendo "boas soluções". Essas informações podem ser usadas como medida de similaridade ao associar um novo espaço de busca com boas soluções armazenadas e, a partir deste ponto reutilizar essas soluções associadas mais eficientemente. Este processo chama-se Esquema de Memória Associativa. Por exemplo, Ramsey e Greffenstette (1993) estudaram um AG para o problema de controle de um robô, onde as boas soluções candidatas são armazenadas em memória permanente junto com a informação referente ao ambiente corrente do robô. Quando o robô incorre a um ambiente similar ao ambiente armazenado anteriormente, a informação de controle armazenada é reativada.

Na Figura 2 é apresentado um esquema que permite diferenciar a memória direta da memória associativa.

Figura 2 – Memória direta *versus* memória associativa.



Quando se trata de busca computacional, o espaço de busca é limitado, e esta preocupação se reflete na organização da memória e dos mecanismos de atualização. Em relação à organização da memória, existem dois mecanismos: mecanismo local onde a memória é orientada ao individuo, e o mecanismo global onde a memória é orientada à população. Trojanowski e Michalewicz, (1999) introduziram um estudo sobre memória local, onde para cada indivíduo, armazena-se na memória o número de seus ancestrais. Quando o ambiente sofre uma alteração, o indivíduo corrente e seus ancestrais são reavaliados e acabam competindo juntos com os melhores indivíduos gerados na execução corrente. O mecanismo de memória global apresenta-se como um processo mais natural quanto à sua aplicação. Neste mecanismo de memória global, o melhor indivíduo da população é armazenado na memória a cada conjunto de gerações, enquanto um determinado indivíduo é retirado da memória de acordo com uma unidade de medida. Por meio deste mecanismo de atualização, um indivíduo é sempre removido da memória para dar lugar a outro com melhor aptidão. Desta forma, pode-se distribuir em várias áreas promissoras do espaço de busca uma população de indivíduos com aptidão acima da média.

Quanto ao processo de se recuperar informações da memória, a tendência natural é recuperar os melhores indivíduos e, recolocá-los na população com ajustes mínimos. Esta estratégia pode ser feita a cada geração ou somente quando houver mudança no ambiente.

7 – MEMÓRIA E PBIL

O pseudocódigo para o PBIL utilizando memória possui a seguinte estrutura:

1. Inicialize o modelo de probabilidade com valor M = 0,5 e a memória vazia.
2. **Repita**
   a. Gere um conjunto de população inicial (n – k) baseado no modelo de probabilidade M.
   b. Avalie o conjunto gerado e denote o melhor indivíduo B.
   c. Avalie a memória e denote o melhor conjunto de indivíduos B_m. Associe-os ao modelo de probabilidade gerador M_m.
   d. **Se** for detectado mudança no ambiente **então**
      i. Se f (B_m) > f (B) então
         1. Troque M por M_m
   e. **Senão** Evolua M em direção a B usando a taxa de aprendizagem gaussiana.
   f. **Se** memória não cheia **então**
      i. Armazene B e M na memória.
   g. **Senão** ache uma amostra na memória A_m próxima a B e, seu modelo de probabilidade associado M_c.
      i. Se f (B) > f (A_m) então
         1. Troque A_m e M_c por B e M.
3. **Até** fim das gerações

Segundo o pseudocódigo para o PBIL, **n** representa o número de avaliações por iteração incluindo os pontos de memória e, f(X) representa a aptidão do indivíduo X.

Este PBIL baseado em memória apresenta uma memória de tamanho (m = 0,1 * n) usada para armazenar amostras e modelos de probabilidade. Cada posição de memória consiste do seguinte par: uma amostra e um modelo de probabilidade associado. O sistema de medida usado na estratégia de atualização da memória funciona da seguinte forma: quando a memória está para ser atualizada, primeiramente acha-se o ponto de memória com a amostra mais próxima da melhor amostra populacional. Se a melhor amostra populacional possuir uma aptidão maior que a amostra da memória então, ocorre a troca pela melhor amostra populacional; caso contrário, a memória fica intacta. Quando a melhor amostra populacional é armazenada na memória, o modelo de probabilidade que a gerou também é armazenado na memória. Similarmente, quando ocorre a atualização do ponto de memória, tanto a amostra quanto o modelo de probabilidade associado são atualizados pela

amostra da melhor população corrente e seu respectivo modelo de probabilidade gerador.

A memória é reavaliada a cada iteração. Se qualquer amostra presente na memória sofrer alteração no seu valor de aptidão, então o ambiente sofrerá uma mudança. Assim sendo, o modelo de probabilidade associado presente na memória em conjunto com a melhor amostra da memória reavaliada substituirá o modelo de probabilidade corrente, se e somente se, a amostra associada da memória superar a melhor amostra criada pelo modelo de probabilidade corrente. Se nenhuma mudança no ambiente for detectada, o PBIL segue a sua execução padrão. O ponto chave é associar boas soluções ao ambiente referente.

8 – MEMÓRIA E MULTI-PBIL

Para estudar o comportamento do Multi-PBIL sobre o esquema de memória em ambientes dinâmicos, foi necessário adaptar o método original Multi-PBIL ao esquema de memória. A adaptação é mostrada a seguir:

1. Geração dos indivíduos;
2. Avaliação da aptidão de cada indivíduo, baseada em uma função de aptidão de um problema em particular;
3. Criação dos modelos de probabilidade;
4. **Para** cada modelo de probabilidade criado:
   a. Inicializar os modelos de probabilidade por meio da equação (1);
   b. Gerar uma população de **N** indivíduos para cada modelo de probabilidade e, aleatoriamente determinar o gene de cada posição de cada indivíduo (0 ou 1);
   c. Avaliar a aptidão de cada indivíduo;
   d. Avalia as amostras populacionais criadas e identifica a melhor. Associar esta amostra ao modelo de probabilidade que a criou.
   e. **Se** alguma mudança no ambiente for detectada **então**
      i. Denota B_w como a pior amostra das já geradas e associa esta amostra ao seu respectivo modelo de probabilidade gerador.
      ii. **Se** $f(B_m) > f(B_w)$ **então**
         1. Substitui M_w por M_m.
      iii. Senão
         1. Aplica a função de aprendizagem gaussiana aos modelos de probabilidade.
      iv. Identifique a melhor amostra B das populações geradas e seu modelo probabilístico associado.
      v. **Se** memória não estiver cheia **então**
         1. Armazene a melhor amostra e seu modelo probabilístico gerador na memória.
      vi. Senão
         1. Encontre na memória a amostra M_c mais próxima a B e seu modelo probabilístico associado.
         2. **Se** $f(B) > f(M_c)$ **então**
            a. Substitua M_c e M_m por B e M_b.
   f. Ordenar os indivíduos pelo valor de aptidão;
   g. Atualizar os valores probabilísticos de cada posição do modelo, baseado em um conjunto de indivíduos selecionados por uma função de seleção e uma taxa de aprendizagem;
5. Verificar se o modelo de probabilidade convergiu. Se não, repetir os passos a partir de (b) até (e).

M_w representa o modelo de probabilidade associado à pior amostra gerada B_w, e M_b representa o modelo de probabilidade associado à melhor amostra gerada.

Cada um destes modelos gera uma amostra populacional independente e, cada uma destas amostras caminha em direção ao melhor indivíduo (melhor solução). O tamanho das amostras geradas por cada modelo probabilístico é igual às demais geradas pelos outros modelos. Quando é detectada uma mudança no ambiente, o melhor modelo de probabilidade armazenado na memória competirá com o pior modelo de probabilidade em execução. O mecanismo de atualização de memória é similar ao utilizado no método do PBIL, isto é, o modelo de probabilidade vencedor que esteja em execução e sua amostra gerada serão armazenados.

9 – AMBIENTES DE TESTES DINÂMICOS

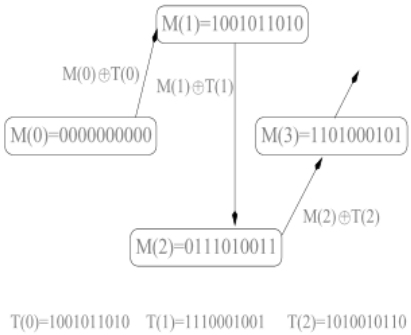
O gerador de problemas dinâmicos proposto por Yang (2003) e Yang e Yao (2005) foi utilizado para construir ambientes dinâmicos a partir de funções estacionárias codificadas binariamente, como por exemplo, $f(x)$ $(x \in \{0, 1\}^l)$, por bit a bit do tipo operador or-exclusivo (XOR).

Na Figura 3 é mostrado um esquema exemplificando a utilização do gerador de problemas dinâmicos. Nesta Figura 3, M(0) representa o estado inicial, M(1) representa o estado 1, M(2) o estado 2, e assim sucessivamente. Para a aplicação do gerador, suponha que o ambiente mude a cada $\boldsymbol{\tau}$ gerações.
Para cada período $\boldsymbol{\kappa}$ do ambiente, uma máscara XOR $M(\kappa)$ é incrementada da seguinte forma:

$$M(\kappa) = M(\kappa - 1) \text{ XOR } T(\kappa) \quad (5)$$

Em que: $T(\kappa)$ é um modelo binário intermediário criado com $(\rho \times l)$ uns (1s) para o período κ do ambiente.

Figura 3 – Gerador de problemas dinâmicos.



Para o primeiro período κ = 1, M(1) é configurado para um vetor zero. Então, a população na geração **t** é avaliada conforme a equação:

$$f(x,t) = f(x \text{ XOR } M(\kappa)) \quad (6)$$

Em que: $\kappa = \sup(t / \tau)$ é o índice do período do ambiente.

Com este gerador, é possível aplicar dois parâmetros aos ambientes dinâmicos. O primeiro parâmetro é o $\boldsymbol{\tau}$ que controla a variação da velocidade, e o segundo parâmetro é o $\boldsymbol{\rho} \in (0,0; 1,0)$ que controla a gravidade com que o ambiente muda. Valores grandes para **ρ** significa mudanças severas no ambiente e, por consequência grandes desafios ao algoritmo. Neste trabalho foram usadas duas funções estacionárias. A primeira função é uma função de 120-bit *OneMax*, cujo objetivo é maximizar o número de uns (1s) em um cromossomo binário. A segunda função, chamada NK(30,4), consiste de 30 *building blocks* contíguos de ordem 4. Cada *building block* contribui com um valor 4 à função de aptidão se todos os quatro bits forem iguais a 1, e contribui com zero caso contrário. O valor de aptidão de um *bit String* é a soma das contribuições de todos os *building blocks.* O valor de aptidão ótimo da função NK(30,4) é 120. Os pontos de máximo e mínimo mudam periodicamente a cada $\boldsymbol{\tau}$ gerações durante a execução, e $\boldsymbol{\tau}$ é configurado para os valores 10, 50 e 100 respectivamente. O parâmetro **ρ** é configurado com os valores 0,1; 0,2; 0,4; 0,6 e 0,9 respectivamente.

10 – ESTUDOS EXPERIMENTAIS

Os testes foram conduzidos com o intuito de se comparar as diferentes versões dos algoritmos evolucionários: PBIL utilizando memória, Multi-PBIL e Multi-PBIL utilizando memória. O ambiente de teste foi elaborado utilizando-se as funções *OneMax* e NK(30,4). Essas funções foram usadas com o objetivo de se criar os espaços de busca. Nesses espaços de busca foram executados os algoritmos evolucionários destacados no problema com o objetivo de encontrar a solução ótima. Para todos os métodos, o tamanho total de amostras *n* (incluindo as amostras de memória quando usadas) foi configurado em 120. O tamanho da memória, quando usada, foi $m = 0,1 * n = 10$ e, a taxa de aprendizagem para todos os modelos probabilísticos foi configurada em 0,05.

Para cada experimento algorítmico sobre um problema de teste dinâmico, foram feitas 20 execuções independentes utilizando as mesmas sementes randômicas. E para cada execução, 100 mudanças no ambiente foram feitas e a melhor aptidão de cada geração foi armazenada. O desempenho global do algoritmo sobre o problema é formulado da seguinte maneira:

$$F_{BOG} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N} F_{BOG}ij\right) \quad (7)$$

Em que: G é o número total de gerações em uma execução (isto é, $G = 100 * \tau$), $N = 20$ é o número total de execuções, $F_{BOG}ij$ representa a melhor aptidão da geração *i* durante a execução *j*, e F_{BOG} é o desempenho *off-line*, isto é, a aptidão média das gerações durante as 20 execuções.

11 – RESULTADOS EXPERIMENTAIS E ANÁLISES

Os resultados experimentais obtidos são apresentados na Tabela 1. Os resultados estatísticos de comparação dos algoritmos por um ***τ*-teste** encadeado com 38 graus de liberdade e com nível de significância de 0,05 são dados na Tabela 2. Na Tabela 2, o resultado de ***τ*-teste** mostra que Alg.1 – Alg.2 possui os seguintes símbolos "+", "-" ou "~", onde Alg.1 é significativamente melhor que, significativamente pior que, ou estatisticamente equivalente ao Alg.2. Nas Figuras 4 e 5 são apresentados os resultados para os ambientes dinâmicos das funções *OneMax* e NK(30,4), respectivamente.

Primeiramente, os resultados obtidos com a aplicação dos esquemas de memória são significantemente melhores que os obtidos sem a aplicação de memória nos problemas dinâmicos testados. Estes resultados validam a eficiência ao se introduzir esquemas de memória em AED. Interpretando o efeito dos esquemas de memória sobre os AED em ambientes dinâmicos, pode-se concluir que o comportamento dinâmico dos algoritmos testados em relação à melhor aptidão avaliada durante todas as gerações, quando $\boldsymbol{\tau = 50}$ e $\boldsymbol{\rho = 0,1}$ mostram que o desempenho do Multi-PBIL normal (sem memória) após passar pelas primeiras pequenas mudanças no ambiente, cai consistentemente com o passar das gerações até chegar um estado estável. Em contraste, o PBIL usando memória e o Multi-PBIL usando memória possuem seus desempenhos aumentados durante as primeiras e pequenas mudanças no ambiente e, posteriormente, tem seu desempenho piorado à medida que as mudanças no ambiente se tornam mais severas. Entretanto, a partir do momento em que ocorre a estabilização da memória (diminuição na frequência de gravação e recuperação) os métodos PBIL e Multi-PBIL, ambos usando memória, se adaptam às mudanças no ambiente. E então, seu desempenho se mantém estável ao invés de cair consistentemente.

Para cada valor fixo de **ρ**, à medida que o valor de τ aumenta, o desempenho do Multi-PBIL sem memória tende a diminuir, enquanto o desempenho do PBIL e do Multi-PBIL usando memória tende a aumentar ou se manter constante. Isto acontece porque quando o valor de τ é muito grande, as mudanças no ambiente se tornam mais lentas, e agindo desta forma, o algoritmo "ganha" mais tempo para realizar as buscas e convergir para um ponto antes da próxima mudança. Esta constatação permite concluir que existe uma menor capacidade de adaptação do Multi-PBIL sem memória e, um pior desempenho sempre que ocorrerem mudanças no ambiente.

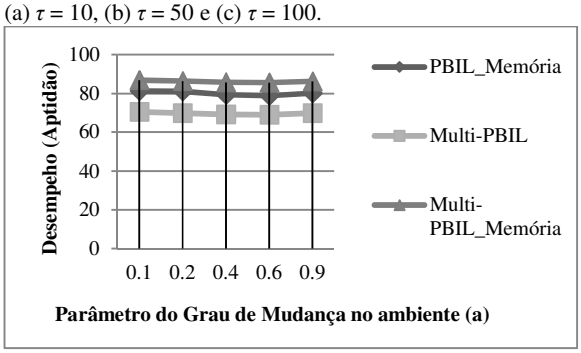
Tabela 1 – Resultados experimentais em relação ao desempenho dos métodos.

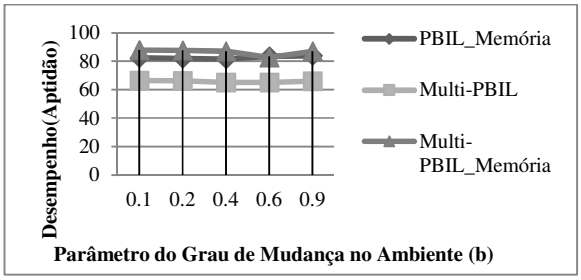
Desempenho		*OneMax*			NK(30,4)		
τ	ρ	PBIL_Memória	Multi-PBIL	Multi-PBIL_Memória	PBIL_Memória	Multi-PBIL	Multi-PBIL_Memória
10	0.1	81,2	70,5	86,8	31,1	28,7	36,8
10	0.2	81,1	69,8	86,4	30,7	28,2	36,3
10	0.4	79,4	69,2	85,7	30,4	27,4	34,1
10	0.6	78,8	69,1	85,6	29,4	26,9	34,0
10	0.9	80,1	69,8	86,2	30,6	28,0	34,7
50	0.1	82,3	66,4	87,8	36,7	24,8	41,3
50	0.2	82,0	66,1	87,7	36,2	24,3	40,0
50	0.4	81,5	65,0	86,9	35,0	23,2	40,0
50	0.6	83,4	65,0	82,6	35,0	23,1	35,7
50	0.9	83,9	66,0	87,0	36,1	23,8	41,6
100	0.1	82,4	61,7	86,7	36,5	20,4	41,1
100	0.2	82,3	60,4	86,3	36,2	20,0	40,6
100	0.4	80,6	60,2	84,2	36,1	19,7	40,1
100	0.6	81,7	58,3	84,1	35,5	18,3	39,2
100	0.9	84,2	61,3	84,6	36,7	19,9	40,8

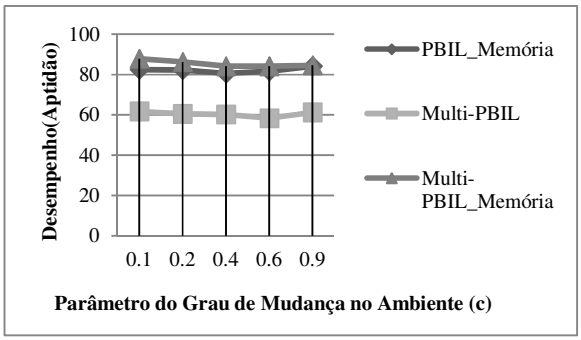
Tabela 2 – Resultados estatísticos comparando os métodos EAD em ambientes dinâmicos.

Resultado τ-Teste	*OneMax*					NK(30,4)				
τ = 10, ρ =>	0,1	0,2	0,4	0,6	0,9	0,1	0,2	0,4	0,6	0,9
PBIL_memória – Multi-PBIL	+	+	+	+	+	+	+	+	+	+
PBIL_memória – Multi-PBIL_memória	-	-	-	-	-	-	-	-	-	-
Multi-PBIL – Multi-PBIL_memória	-	-	-	-	-	-	-	-	-	-
τ = 50, ρ =>	0,1	0,2	0,4	0,6	0,9	0,1	0,2	0,4	0,6	0,9
PBIL_memória – Multi-PBIL	+	+	+	+	+	+	+	+	+	+
PBIL_memória – Multi-PBIL_memória	-	-	-	+	~	-	-	-	~	~
Multi-PBIL – Multi-PBIL_memória	-	-	-	-	-	-	-	-	-	-
τ = 50, ρ =>	0,1	0,2	0,4	0,6	0,9	0,1	0,2	0,4	0,6	0,9
PBIL_memória – Multi-PBIL	+	+	+	+	+	+	+	+	+	+
PBIL_memória – Multi-PBIL_memória	-	-	-	-	+	-	-	-	-	~
Multi-PBIL – Multi-PBIL_memória	-	-	-	-	-	-	-	-	-	-

Figura 4 – Resultados experimentais sobre os métodos EAD sobre ambientes dinâmicos sobre a função *OneMax* com valores (a) $\tau = 10$, (b) $\tau = 50$ e (c) $\tau = 100$.



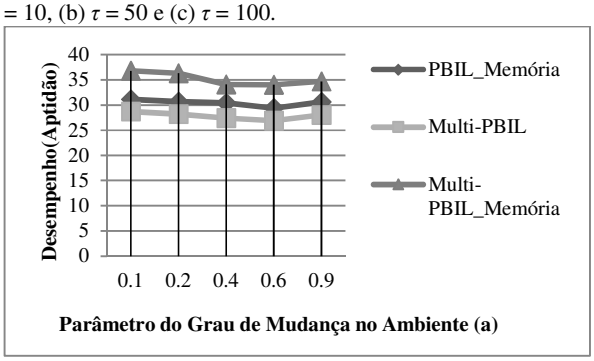


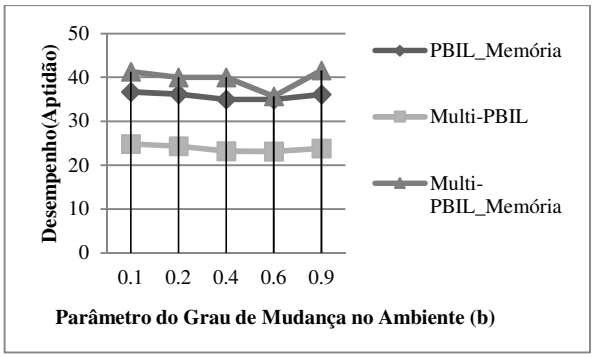


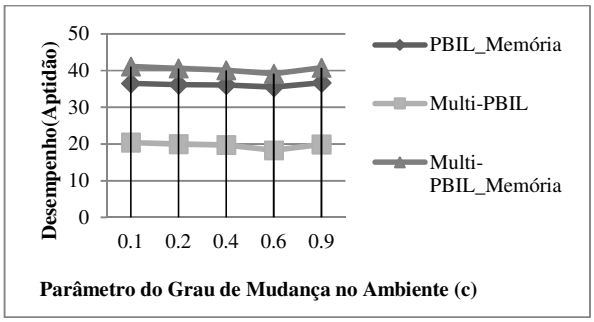
Para o método PBIL usando memória, as buscas por um período longo (valor de τ muito grande) fazem com que os modelos de probabilidade armazenados na memória simbolizem com maior precisão os ambientes mais antigos armazenados. Este fato leva a uma melhor eficiência no esquema de memória e por consequência, um melhor desempenho do método PBIL usando memória.

Fixando o valor de τ, à medida que o valor de ρ aumenta (aumento severo das mudanças frequentes no ambiente) o desempenho de todos os métodos analisados tendem a diminuir. Para o valor $\rho = 0.9$, o desempenho dos métodos analisados se mostram melhores que os desempenhos obtidos quando aos valores de ρ são 0,6 e 0,4. Isto significa que, o método Multi-PBIL usando memória possui uma capacidade de adaptação mais rápida a novos ambientes, sempre que estes sofrerem mudanças significativas.

Figura 5: Resultados experimentais sobre os métodos EAD sobre ambientes dinâmicos sobre a função NK(30,4) com valores (a) $\tau = 10$, (b) $\tau = 50$ e (c) $\tau = 100$.







Uma observação importante deve ser feita em relação à Tabela 1, onde τ possui o valor 50 e ρ o valor 0,6. Nesta observação, o PBIL usando memória se mostra melhor que Multi-PBIL usando memória. Os resultados do PBIL e Multi-PBIL especificamente para esses valores representam um comportamento anômalo do algoritmo perante a função randômica. Uma mesma semente randômica foi utilizada para todas as execuções, porém, neste caso específico, o vetor de probabilidade do PBIL teve um desempenho melhor (convergência) em relação aos modelos de probabilidade do Multi-PBIL.

Concluindo, tanto o método PBIL usando memória quanto o Multi-PBIL usando memória possui um desempenho muito melhor que o método Multi-PBIL sem memória no trato de problemas dinâmicos. Ou seja, multi populações são características interessantes aos AEDs quando utilizados com memória e, desinteressantes quando

utilizados sem memória. Introduzir um modelo de probabilidade em um ambiente (espaço de busca) aumenta consideravelmente a diversidade e, por consequência a sua adaptabilidade aos ambientes dinâmicos.

12 – CONCLUSÕES E RECOMENDAÇÕES

Este trabalho apresentou um estudo sobre memórias associativas aplicadas a ambientes dinâmicos, com o objetivo de se avaliar o desempenho dos Algoritmos de Estimação de Distribuição. Neste ambiente de teste, o melhor modelo de probabilidade e sua melhor amostra populacional são armazenados na memória. Quando ocorre uma mudança no ambiente, o modelo de probabilidade correspondente àquele ponto de memória no espaço de busca é armazenado e classificado como sendo o melhor neste novo ambiente. Este modelo pode ser recuperado da memória com o intuito competir com outros modelos de probabilidade, paralelamente, durante as iterações no espaço de busca. Os resultados experimentais validam a eficiência dos esquemas de memória em ambientes dinâmicos.

Este trabalho mostra também o desempenho do método Multi-PBIL aplicado a memórias associativas. Os resultados indicam que essa relação favorece o desempenho do método porque a existência de modelos de probabilidade adicionais acelera o processo de busca (e por consequência o desempenho) do modelo armazenado na memória. Esses estudos podem ser estendidos para ambientes dinâmicos cíclicos, onde as configurações de ambientes mais antigos retornam ao cenário de tempos em tempos, ou seja, a cada número de gerações. Uma pergunta importante a se fazer nesse caso é: diante de tais ambientes dinâmicos, pode-se esperar que o uso de esquemas de memória seja mais benéfico ao método estudado? Estudar outros algoritmos de estimação de distribuição também é uma possibilidade. Ou mesmo outros modelos de gerenciamento da memória.

REFERÊNCIAS

BACK, T. On the behavior of evolutionary algorithms in dynamic fitness landscape. In Proc. of the **1998 IEEE International Conference on Evolutionary Computation**, p. 446-451, 1998. PMid:9737454.

BALUJA, S. **Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning**. Technical Report, Carnegie Mellon University, 1994.

BARBOSA FILHO, R. **Multi-PBil: Um algoritmo de estimação de distribuição aplicado a problemas de otimização multimodais**. Dissertação de Mestrado apresentada ao Departamento de Informática da Universidade Federal do Paraná, Curitiba, PR, Brasil, 2005.

BENGOETXEA, E. **Inexact graph matching using estimation of distribution algorithms**. PhD Thesis, Ecole Nationale Superieure des Telecommunications, Paris, France, December 2002.

BRANKE, J. Memory enhanced evolutionary algorithms for changing optimization problems. In Proc. of the **1999 Congress on Evolutionary Computation**, v. 3, p. 1875–1882, 1999.

BRANKE, J. **Evolutionary optimization in dynamic environments**. Kluwer Academic Publishers, 2002. http://dx.doi.org/10.1007/978-1-4615-0911-0.

GOLDBERG, D. E.; SMITH R. E. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In Proc. of the **2nd International Conference on Genetic Algorithms**, p. 59-68. Lawrence Erlbaum Associates, 1987.

LEWIS, E. H. J.; RITCHIE, G. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In Proc. of the **5th International Conference on Parallel Problem Solving from Nature**, p. 139-148, 1998. PMid:9520518.

NG, K. P.; WONG, K. C. A new diploid scheme and dominance change mechanism for non-stationary function optimization. In Proc. of The **6th International Conference on Genetic Algorithms**. Morgan Kaufmann Publishers, 1997.

PAPADIMITRIOU, C. H., STEIGLITZ, K. **Combinatorial Optimization: Algorithms and Complexity**. Prentice-Hall, Inc., 1982.

RANSEY, C. L.; GRENFFENSTETTE, J. J. Case-based initialization of genetic algorithms. In Proc. of the **5th International Conference on Genetic Algorithms**. Morgan Kaufmann Publishers, 1993.

TAKAHASHI, R. H. C.; MARTINS, F. V. C. Sobre a Inércia de Populações de Algoritmos Genéticos, **XXXVII Simpósio Brasileiro de Pesquisa Operacional**, Gramado RS, 2005.

YANG, S. Non-stationary problem optimization using the primal-dual genetic algorithm. In Proc. of the **2003 Congress of Evolutionary Computation**, v. 3, p. 2246–2253, 2003.

YANG, S.; YAO, X. **Experimental study on population-based incremental learning algorithms for dynamic optimization problems**. Soft Computing, 2005. http://dx.doi.org/10.1007/s00500-004-0422-3.

YANG, S.; YAO, X. Population-based incremental learning with associative memory for dynamic environment. **IEEE Transactions on Evolutionary Computation**, v. 12, n. 5, October, 2008.