Algoritmo Evolutivo para Solução de Problemas de Larga Escala

Thyago S. P. C. Duque[1], Kumara Sastry[2], Alexandre C. B. Delbem[1], David E. Goldberg[2]

[1]Instituto de Ciências Matemáticas e Computação – Universidade de São Paulo (USP)
Av. Trabalhador São-carlense, 400 – São Carlos – SP – Brazil

[2]Illinois Genetic Algorithms Laboratory – University of Illinois at Urbana-Champaign
Urbana – IL – USA.

{thyago,acbd}@icmc.usp.br, {kumara, deg}@illigal.ge.uiuc.edu

Abstract. *Evolutionary algorithms (EAs) are a largely used search and optimization technique. They have been successfully applied to a wide variety of problems, overcoming traditional algorithms in performance. However, few EAs and traditional algorithms are able to handle complex combinatorial problems involving a great number of variables (thousands or millions). This paper proposes a new EA, capable of solving combinatorial problems with great number of variables. This algorithm is the result of two extensions from the Extended Compact Genetic Algorithm, a state-of-the-art EA.*

Resumo. *Os algoritmos evolutivos são uma técnica de busca e otimização que tem sido aplicada com sucesso em diversas situações, atingindo algumas vezes resultados superiores àqueles conseguidos por técnicas tradicionais. No entanto, poucos algoritmos evolutivos estão aptos a lidar com problemas combinatoriais realmente complexos, envolvendo grande número de variáveis (milhares, ou até milhões). Este artigo descreve os resultados preliminares de um novo algoritmo evolutivo, capaz de resolver um problema de otmização envolvendo uma grande quantidade de variáveis. Este algoritmo é o resultado de duas extensões desenvolvidas a partir do Extended Compact Genetic Algorithm, um representante do estado da arte dos algoritmos evolutivos.*

1. Introdução

Muitos problemas do mundo real envolvem a otimização de parâmetros em conjuntos de dados que podem chegar a centenas, milhares ou até milhões de bits. Esses problemas apresentam complexidade computacional muito alta e não podem ser solucionados em tempo viável por técnicas de força bruta (Papadimitriou, 1995).

A abordagem mais comum para esses problemas é a utilização de técnicas que encontram resultados aproximados, ótimos locais. Dentre essas técnicas destacam-se os Algoritmos Evolutivos (EA), que apresentam como uma das vantagens o fato de que poderem ser aplicados tanto sem utilizar informações específicas do problema (comportando-se como uma técnica genérica) como utilizando informações específicas de domínio, gerando algoritmos de aplicabilidade mais restrita, porém com maior desempenho.

Por outro lado, existem poucos algoritmos capazes de resolver problemas envolvendo milhares de variáveis sem utilizar nenhuma informação específica de domínio. Este artigo propõe uma extensão do Extended Compact Genetic Algorithm (ECGA) (Harik, 1999) que o torna capaz de resolver problemas genéricos de alta complexidade.

Atualmente, restrições de memória e de tempo de processamento impedem que o ECGA seja aplicado para problemas de alta complexidade. Nossa proposta melhora o desempenho do ECGA utilizando uma técnica de compressão de cromossomo combinada com uma busca local que reduz significativamente o espaço de busca do algoritmo, tornando assim viável sua aplicação em problemas de larga escala.

Este artigo está organizado da seguinte forma: A Seção 2 apresenta uma breve introdução aos algoritmos evolutivos. A Seção 3 apresenta uma introdução aos algoritmos de estimação de distribuição e ao ECGA. A Seção 4 explica o funcionamento do novo algoritmo, descrevendo as extensões propostas. A Seção 5 apresenta os resultados preliminares obtidos com o algoritmo desenvolvido e a Seção 6 apresenta considerações finais a respeito deste trabalho.

2. Algoritmos Evolutivos

Algoritmos Evolutivos (EAs) (Goldberg, 1989; De Jong, 2006) são técnicas de solução de problemas complexos baseado em um processo evolutivo inspirado na teoria da evolução Neo-Darwinista (Darwin, 2000).

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

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

```
1. Gere aleatoriamente uma população inicial
2. Enquanto algum critério de parada não for atingido:
      a. Selecione os pais (influenciado pela aptidão)
      b. Produza novos filhos
      c. Selecione indivíduos para morrer (influenciado pela
         aptidão)
3. Fim enquanto
4.Retorne a população resultante
```

Algoritmo 1. Algoritmo evolutivo genérico

As diversas técnicas classificadas como EA diferem entre si pela forma com que especificam cada um dos seguintes parâmetros: Tamanho da população; Seleção dos "pais"; Reprodução e Herança; Sobrevivência; Representação dos indivíduos; Função aptidão.

3. Algoritmos de Estimação de Distribuição

Podemos enxergar um conjunto de indivíduos selecionados de uma população como uma amostra retirada a partir de uma distribuição desconhecida. Conhecer essa distribuição nos permitiria gerar novas soluções que de alguma forma estivessem relacionadas com aquelas selecionadas. Esse é o princípio dos Algoritmos de Estimação de Distribuição (EDA) (Larranaga & Lozano, 2001), isto é, este é um EA capaz de estimar a distribuição de probabilidade através do conjunto de indivíduos selecionados e, usando essa estimativa, gerar novas soluções.

Essa técnica funciona da seguinte forma: inicialmente, um conjunto aleatório de indivíduos é gerado. Em seguida, alguns deles são selecionados, como nos Algoritmos Genéticos (GAs) (Goldberg, 1989). No entanto, ao invés de gerar novas soluções por meio de operadores reprodutivos, o EDA estima uma distribuição baseado nos indivíduos selecionados e gera (amostra) novos indivíduos usando essa distribuição. Posteriormente, os novos indivíduos são adicionados à população, substituindo alguns indivíduos antigos. O processo de seleção, estimação e amostragem é repetido até que algum critério de parada seja satisfeito.

Note que, assim com GA, Estratégias Evolutivas (ES) (Beyer, 2001) e Programação Genética (GP) (Koza, 1992), os EDAs são um tipo particular de EA e, sob certas condições, podem ter comportamento idêntico a essas técnicas (Harik, Lobo & Goldberg, 1998). A diferença básica é que os EDAs substituem a aplicação de operadores reprodutivos pelos seguintes passos:

1. Construção de um modelo probabilístico (uma estimação da distribuição real) dos indivíduos selecionados;
2. Geração de novos indivíduos baseado no modelo construído.

A grande desvantagem dos EDA é que a construção de um modelo probabilístico não é uma tarefa trivial. Em geral, existe um compromisso entre a precisão do modelo construído e a eficiência do processo de estimação.

Como exemplos de EDA podemos citar o Compact Genetic Algorithm (cGA) (Harik, Lobo & Goldberg, 1998), que desconsidera qualquer interação entre as variáveis (posições da cadeia que representa o indivíduo), o ECGA, que considera que as variáveis podem estar relacionadas em grupos (Building Blocks, BB), porém desconsidera sobreposição entre esses grupos e o Bayesian Optimization Algorithm, BOA, que assume possibilidades de iteração mais complexas, representadas por Redes Bayesianas (Cheng, Bell & Liu, 1997)

Este artigo propõe duas extensões a partir do ECGA para melhorar seu desempenho e tornar solúveis problemas antes intratáveis por restrições de memória e tempo de processamento. A seguir, são apresentados os principais conceitos do ECGA.

3.1. Extended Compact Genetic Algorithm

O cGA, um dos EDA mais simples, comporta-se de forma similar ao Simple GA e é capaz de resolver os mesmos problemas apresentando desempenho semelhante (Harik, Lobo & Goldberg, 1998). No entanto, o Simple GA não é um EA eficiente para solução de problemas que envolvam relações complexas entre variáveis e, nesse caso, o cGA possui o mesmo desempenho insatisfatório. Esta limitação impulsionou o desenvolvimento de EDAs mais eficientes para problemas complexos.

O conceito de *Building Block* (BB), fundamental para o ECGA, é definido formalmente em Goldberg (1989, 2002) e em Holland (1975). Para o entendimento deste trabalho é suficiente assumir que um BB é um conjunto de variáveis que estão de alguma forma relacionadas e cuja contribuição na aptidão do indivíduo se dá analisando os valores de todas as variáveis simultaneamente. Em outras palavras, um BB é um conjunto de variáveis para as quais a contribuição de cada variável na aptidão é dependente valor das outras variáveis do conjunto. O processo de determinação das relações entre variáveis que formam um BB é denominado *linkage learning*.

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

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

$f(c) =$ número de *bits* com valor 1 em *c* caso *c* possua algum *bit* diferente de 0;

$l+1$ caso todos os *bits* de *c* sejam 0.

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

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

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

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

O conceito de simplicidade é definido em termos de complexidade de representação. O critério da complexidade combinada (*Combined Complexity Criterion*) (CCC) (Harik, 1999) é empregado nesse sentido. Esse critério diz que a soma da complexidade do modelo (*Model Complexity*) (MC) com a complexidade da população comprimida (*Compressed Population Complexity*) (CPC) deve ser mínima. Essas duas métricas são definidas em (Harik, 1999). Definidos esses conceitos, o ECGA segue o pseudocódigo apresentado no Algoritmo 2.

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

Algoritmo 2. Pseudocódigo do ECGA

O ECGA usa um algoritmo de busca gulosa para a criar um modelo da população. Este algoritmo funciona da seguinte forma: inicie assumindo que todas as variáveis são independentes. Então tente unir dois conjuntos de variáveis e verifique se há melhora na CCC. A melhor de todas as uniões (menor CCC) é mantida. Mais detalhes sobre o ECGA e discussões sobre os resultados obtidos por esse algoritmo podem ser encontrados em (Harik, 1999).

4. Chromosome Compression Extended Compact Genetic Algorithm

Nesta seção são descritas extensões realizadas no ECGA gerando o novo algoritmo evolutivo, denominado Chormosome Compression Extended Compact Genetic Algorithm, ccECGA..

4.1. Binary Hillclimber

O ccECGA incorpora um *Binary Hillclimber* como método de busca local ao ECGA.

O *Binary Hillclimber* proposto funciona da seguinte forma: dada uma cadeia *c* de comprimento *l* e uma função de aptidão *F* aplica-se o Algoritmo 3.

```
Para i := 1 até l :
   Fanterior = F(c)
   inverter(c[i])
   Fatual = F(C)
   Se (Fatual > Fanterior) :
      Mantenha a modificação
   Senão :
      inverter(c[i])
```

Algoritmo 3. Hillclimber

O procedimento "`inverter(c[i])`" consiste simplesmente em inverter o *bit* na posição *i* da cadeia *c*. Com a aplicação desse procedimento, cada um dos BB que compõe a cadeia *c* será transformado em um ótimo local para o subproblema ao qual o BB está associado. Com isso, o ECGA terá seu desempenho melhorado por passar a trabalhar com uma quantidade menor de instancias diferentes para um dado BB.

4.2. Compressão de Cromossomo

O princípio de funcionamento de ECGA considera que a determinação de uma distribuição adequada é equivalente ao processo de *linkage learning*. Baseado nesse princípio, o ECGA tenta aprender relações entre as variáveis para determinar quais são os BB responsáveis por cada subproblema. O algoritmo, então, recombina soluções ótimas para os subproblemas (BBs inteiros) buscando encontrar a solução ótima para o problema global. A aplicação do *Hillclimber* elimina todas as soluções intermediárias para os subproblemas, mantendo apenas os ótimos locais.

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

4.3. Extendendo o ECGA

Utilizando um *Binary Hillclimber* e o conceito de compressão de cromossomos, em conjunto com o ECGA, obtemos um novo algoritmo evolutivo. Este algoritmo é capaz resolver problemas que eram intratáveis por restrições de memória ou tempo de processamento.

O ccECGA é descrito pelo o Algoritmo 4.

```
1. Gere uma população aleatória
2. Aplique o Binary Hillclimber
3. Realize seleção sobre essa população
4. Modele a população usando uma busca gulosa
5. Comprima a população usando o modelo
6. Aplique o ECGA sobre a população comprimida
```

Algoritmo 4. O ccECGA

O ccECGA pode ser visto como um pré-processamento da população para o ECGA. O *Hillclimber* e a compressão de cromossomos são aplicados antes da execução do ECGA, possibilitando que esse trabalhe sobre um espaço de busca reduzido, melhorando assim seu desempenho.

É importante observar que há outras formas de incorporar a compressão de cromossomo no ECGA. É possível ainda aplicar a compressão de cromossomo toda vez que um novo modelo for gerado.

5. Resultados

Esta seção descreve dois experimentos realizados com o ccECGA e os resultados obtidos. O primeiro experimento tem como objetivo comparar o desempenho do ccECGA com o ECGA. O segundo experimento tenta estimar a escalabilidade do método proposto.

5.1 Comparação entre ECGA, ECGA com *Hillclimber* e ccECGA

Para analisar a contribuição das extensões propostas e verificar se podem de alguma forma melhorar o desempenho do ECGA, foi realizado o teste descrito a seguir. Cada um dos algoritmos foi empregado solucionar um problema deceptivo por partes (ver seção 3.1) com k (tamanho de um BB) igual a 4, m (número de BBs) igual a 25 e l ($k*m$) igual a 100. Utilizou-se tamanho da população igual a 400, consequentemente, não se pode esperar que o ECGA tenha bom desempenho uma vez que o tamanho da população estimado para o ECGA usando os modelos analíticos (Sastry & Goldberg, 2004) é 1.600. O número máximo de gerações para cada um dos algoritmos foi restringido para 50 gerações.

Para cada um dos algoritmos foram realizadas 300 execuções. Para cada uma dessas execuções, foi anotado o número de gerações para convergência, o número de BB "adivinhados" corretamente e o *fitness* obtido. Estes dados foram utilizados para traçar gráficos comparativos do desempenho dos algoritmos.

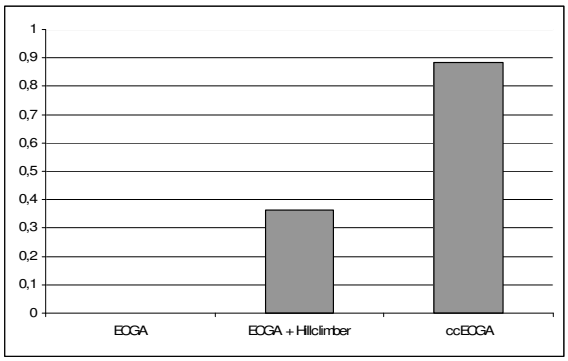

Figura 1. Proporção das execuções que encontraram o ótimo global

O gráfico da Figura 1 mostra a proporção das execuções em que cada um dos algoritmos encontrou o ótimo global. Este gráfico indica que o ECGA com *Hillclimber* é superior ao ECGA tradicional, que nos testes realizados não foi capaz de encontrar o máximo global nenhuma vez. O ccECGA também apresenta desempenho superior ao do ECGA com *Hillclimber*, o que demonstra que tanto o *Hillclimber* quanto a compressão de cromossomo são responsáveis por mudanças positivas no desempenho do ECGA.

Além da proporção de execuções que atingiram o máximo global, foram observadas a proporção das execuções em que nenhum dos BBs do indivíduo foi encontrado corretamente (ECGA 1%, ECGA + Hillclimber 0%, ccECGA 0%) e a proporção das execuções em que apenas um BB foi encontrado de forma incorreta (ECGA 0%, ECGA + Hillclimber 21%, ccECGA 12%).

Estes dados indicam que o ccECGA matem as características de um EA competente e ainda é capaz de manipular eficientemente uma população de tamanho menor do que a requerida pelo ECGA, podendo assim ser aplicado em problemas que antes eram intratáveis devido a restrições de memória ou tempo de processamento.

5.2 Escalabilidade do ccECGA

Na seção anterior foi demonstrado que em uma determinada situação o ccECGA tem desempenho muito superior ao do ECGA. No entanto, mais do que comparar o

desempenho do algoritmo para um caso específico, deve-se estudar a escalabilidade do método proposto.

Por isso é necessário saber se houve modificação na complexidade do algoritmo ou se a melhoria obtida foi em relação às constantes do problema. Por constantes, entende-se qualquer modificador da complexidade do algoritmo que seja desconsiderado na análise assintótica de algoritmos (Papadimitriou, 1995). Esta característica foi verificada utilizando o método da bisseção para determinação da população mínima (Sastry, 2001) com os parâmetros a descritos seguir: o problema alvo escolhido foi um problema deceptivo por partes com BB de comprimento (*k*) 10 *bits* e o número de BBs variando nos experimentos. Utilizou-se taxa de erro máxima de 1/*m*, ou seja, no máximo um BB errado. Foi usado *nRuns* 10 e *tolerância* 1. O limitante inferior e superior foi ajustado dinamicamente para cada *m* com base em previsões empíricas.

Cada um dos BB foi avaliado utilizando a seguinte função:

$$f = \frac{(d-1)*x}{k_c} + (1-d), \text{ se } x < k_c$$

$$f = \frac{(x-k_c)}{(k-k_c)}, \text{ caso contrário}$$

onde *x* é o número de *bits* do BB com valor 1, *k* é o comprimento do BB, k_c é o valor crítico da função deceptiva e *d* é o parâmetro de ajuste. Essa função é chamada *v-shape.* Nos testes realizados foram utilizados os seguintes valores: $k = 10$, $k_c = 8$, $d = 0.20$. Para esses valores *f* é uma função deceptiva (Goldberg, 2002).

Para cada execução do método da bisseção foi anotado o tamanho mínimo de população encontrado, o número médio de avaliações de indivíduos e o número médio de gerações para convergência. Os resultados reportados nesta seção são referentes à média de dez execuções do método da bisseção.

A Figura 2 mostra a escalabilidade do ccECGA com relação ao tamanho da população obtido pelo método da bisseção. No gráfico são apresentados os dados obtidos experimentalmente, a escalabilidade prevista obtida por meio do ajuste aos dados e o resultado teórico para o ECGA.

Como podemos observar, a escalabilidade do ccECGA continua muito próxima a do ECGA ($\theta(m \ln m)$, (Sastry & Goldberg, 2004)), indicando que ambos escalam praticamente da mesma maneira. No entanto, o ccECGA precisa de uma população duas ordens de grandeza menor do que do ECGA. Apesar de não ter havido melhora na escalabilidade, essa diferença de duas ordens de complexidade é suficiente para permitir, por exemplo, que o ccECGA seja executado para m=60 em uma máquina convencional, com 1Gb de RAM. Isso seria impossível para o ECGA convencional, devido a restrições de memória.

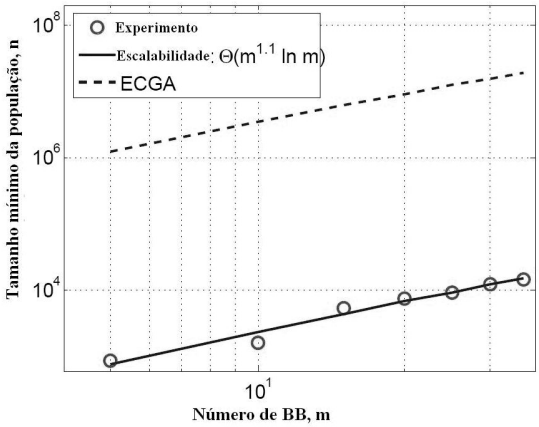

Figura 5. Escalabilidade do ccECGA com relação ao tamanho da população

A escalabilidade do ccECGA foi verificada também considerando o número de gerações para convergência e o número de avaliações da função aptidão. Em ambos os casos o ccECGA não apresentou variação significativa de escalabilidade, porém apresentou novamente variação positiva com relação as constantes envolvidas na complexidade. O número de avaliações da função aptidão é três ordens de complexidade menor no ccECGA em relação ao ECGA, representando uma variação de 10^5 para 10^8 avaliações no caso de m=5.

A escalabilidade do algoritmo com relação ao tamanho do BB não foi estimada, porém uma análise inicial indica que este apresentará os mesmos problemas que o ECGA. Ao analisar o funcionamento do *Hillclimber* percebeu-se que este era muito sensível ao valor crítico de decepção. Modelando a probabilidade de o *Hillclimber* encontrar o máximo global de um determinado BB chegamos à conclusão que este segue um modelo de *Gambler's Ruin* (Goldberg, 2002). Este segue a seguinte equação de probabilidade: $p = 1 - \sum_{i=1}^{k_c-1} \left[\frac{\binom{k}{i}}{2^k} \right] + \frac{\binom{k}{k_c}}{2^k * 2}$, onde k é o comprimento de um BB e k_c é o valor crítico desse BB. Como pode ser observado pela equação, a probabilidade de o *Hillclimber* encontrar o ótimo global do BB decresce exponencialmente em relação ao tamanho do BB. Este fato dá indícios de que assim como o ECGA, o ccECGA escale exponencialmente em relação ao tamanho do BB (k).

6. Considerações Finais

O algoritmo proposto mostrou-se eficaz e apresentou melhora significativa em relação ao ECGA nos testes preliminares. Esta melhora, apesar de não implicar em aumento da escalabililidade é suficiente para tornar tratáveis problemas antes intratáveis devido a restrições de tempo e memória. Desta perspectiva o ccEGCA atingiu sucesso em seu objetivo.

Os testes realizados ainda não são suficientes para determinar definitivamente o desempenho do ccECGA. Outros testes precisam ser realizados para determinar a escalabilidade do ccECGA com relação ao tamanho do BB (k). Este é o ponto crítico do ECGA, pois este escala exponencialmente em relação a k. Apesar de as suposições iniciais indicarem um crescimento exponencial para o ccECGA, este resultado ainda não foi confirmado experimentalmente.

O *Hillclimber* limita o desempenho do algoritmo devido à dificuldade de otimização local quando k aumenta e k_c aumenta proporcionalmente. No entanto, é possível otimizar o processo de *Hillclimbing* através de paralelização deste processo. Como o ccECGA e o *Hillclimber* lidam com uma população de indivíduos e o *Hillclimber* trabalha com cada um deles independentemente, é possível paralelizar facilmente este algoritmo obtendo melhoria significativa devido ao baixo custo de comunicação.

O procedimento de paralelização proposto como trabalho futuro consiste em distribuir entre N processadores disponíveis a população de indivíduos para que cada processador realize o processo de busca local e de seleção independentemente. Apenas os indivíduos selecionados pelo processador serão transmitidos para o processador responsável pela execução da compactação e do ECGA. Com isso, espera-se conseguir uma melhora significativa no desempenho.

Outra questão em aberto, a ser solucionada em trabalhos futuros, é o efeito da aplicação da compressão de cromossomo não como um pré-processamento, mas como um passo efetivo do algoritmo, a ser executado a cada iteração, permitindo assim a redução do espaço de busca sempre que for possível. Esta extensão pode melhorar ainda mais o desempenho do ccECGA.

O resultado deste projeto foi o desenvolvimento de uma técnica inovadora de solução de problemas de otimização combinatória que conseguiu sucesso na tarefa de melhorar o desempenho de uma técnica do estado da arte da computação evolutiva. Os resultados preliminares foram promissores e as perspectivas de trabalhos futuros indicam que estes resultados podem ainda ser melhorados.

Referências

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

(Cheng, Bell & Liu, 1997) J. Cheng, D. A. Bell, W. Liu, (1997), *Learning belief networks from data: An information theory based approach*, in Proceedings of ACM.

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

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

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

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

(Harik, 1999) G. Harik, (1999), *Linkage Learning via probabilistic modeling in the ECGA*, Illigal Report No. 99010

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

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

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

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

(Papadimitriou, 1995) C. H. Papadimitriou, (1995), *Computational Complexity*,

(Sastry, 2001) K. Sastry, (2001), *Evaluation-relaxation schemes for genetic and evolutionary algorithms,* Master's thesis, Urbana, IL

(Sastry & Goldberg, 2004) K. Sastry, D. E. Goldberg, (2004), *Designing competent mutation operators via probabilistic model building of neighborhoods*, Illigal Report No. 2004006