Learning and Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais (SBRN), Vol. 6, No. 2, pp. 121-141, 2008

ALGORITMO AUXILIAR PARALELO PARA MELHORAR A PERFORMANCE DE ALGORITMOS GENÉTICOS COM CODIFICAÇÃO BINÁRIA

Celso G. Camilo Jr
Faculdade de Ciências Exatas e Tecnologia
Universidade Federal da Grande Dourados
Dourados, MS
celsocamilo@ufd.edu.br, celsocamilo@gmail.com

Keiji Yamanaka
Faculdade de Engenharia Elétrica
Universidade Federal de Uberlândia
Uberlândia, MG
keiji@ufu.br

Resumo

Várias são as técnicas aplicadas em problemas de otimização, no entanto, poucas alcançam desempenho satisfatório quando o problema é complexo como, por exemplo, multimodal ou multiobjetivo. As metaheurísticas, apesar de não garantirem o ótimo global, apresentam bons resultados e, por isso, são bastante utilizadas para esses cenários. Entre as metaheurísticas, os algoritmos evolucionários, especialmente os Algoritmos Genéticos (AG), apresentam ótimos resultados e, por isso, são bastante populares entre os pesquisadores. No entanto, o processo de melhoramento da solução de um AG pode ser lento, principalmente em casos de grande complexidade. Em vista disso, alguns trabalhos são desenvolvidos para melhorar o desempenho do AG. Quando se agiliza o processo de evolução em algoritmos evolucionários, normalmente, corre-se o risco de gerar soluções prematuras e fortes, que podem influenciar negativamente a população a máximos e mínimos locais. Sendo assim e percebendo a necessidade de soluções de aplicabilidade ampla, este trabalho propõe descrever e problematizar o Algoritmo Auxiliar Paralelo (AAP), utilizado para auxiliar a evolução da população dos AGs com codificação binária. O algoritmo proposto é um modulo que, acoplado aos AGs, alimenta a população de bons indivíduos. Quatro operadores foram criados para o AAP: AR, EAR-T, EAR-P e EAR-N, todos funcionalmente independentes. Foram efetuados experimentos para aferir a eficiência do AAP e seus operadores. Os resultados obtidos demonstram que o AAP cumpre o objetivo de auxiliar o AG sem o uso de conhecimento específico do problema.

Palavras Chave: Algoritmos Genéticos. Convergência Prematura. Velocidade de Convergência. Computação Evolucionária. Metaheurística.

Abstract

Some techniques are applied in the optimization problems, however, just a few achieve satisfactory performance when the problem is complex, for example, multimodal or multiobjective. The metaheuristics, although not guaranteeing a global optimum, have good results and, hence, are quite used to these scenarios. Among the metaheuristics, the evolutionary algorithms, especially the Genetic Algorithms (GA), have great results and, hence, one of the most popular. However, the process of improving the solution of an AG may be slow, especially in cases of great complexity. Hence, some papers are developed to improve the performance of the AG. However, when it speeds up the process of evolution in evolutionary algorithms, normally increases the risk of premature convergence, which can negatively influence the population to maximum and minimum locations. Therefore, this work suggests the Assistant Parallel Algorithm (AAP), an algorithm to assist the evolution process of binary encoding GAs. The proposed algorithm is a module attached to the AGs that feeds the population of good individuals. Four operators were created for the AAP: AR, EAR-T, EAR-P and EAR-N, all functionally independent. Experiments were done to measure the efficiency of the AAP and its operators. The results show that the AAP reach the objective of assist the good evolution without using specifics knowledges about the problem.

Learning and Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais (SBRN), Vol. 6, No. 2, pp. 121-141, 2008

Keywords: Genetic algorithms, Premature Convergence, Convergence Speed, Evolutionary Computation, Metaheuristics.

1. Introdução

Várias são as técnicas aplicadas em problemas de otimização, no entanto, poucas alcançam desempenho satisfatório quando o problema é complexo como, por exemplo, multimodal ou multiobjetivo. Os algoritmos da Programação Matemática, que utilizam como guia de busca o gradiente, têm grandes dificuldades e, quase sempre, não atingem o ótimo global em problemas multimodais. Já as metaheurísticas apresentam bons resultados e, por isso, são bastante utilizadas para estes cenários, apesar de não garantirem o ótimo global.

Pode-se dividir as metaheurísticas, quanto há estratégia de busca em dois grupos. O primeiro é de busca populacional e o segundo não populacional. A estratégia populacional inicia a busca com vários pontos no espaço de busca e, por meio da interação desses, tenta levar os pontos para um de maior valor da função objetivo, a cada iteração. Esta, portanto, explora o espaço de busca em vários pontos simultaneamente, fazendo assim um paralelismo na busca. Já a não populacional baseia-se em um único ponto para efetuar a exploração do espaço, quase sempre munida de técnicas para fugir de ótimos locais. Ambas as estratégias demonstram bons resultados, dependendo do problema uma ou outra é mais adequada.

Como exemplo de metaheurística, pode-se citar: *Simulated Anneling*, *Tabu Search*, GRASP, VND, VNS, Colônia de Formigas e Algoritmos Genéticos. Entre esses exemplos, os algoritmos evolucionários, especialmente os Genéticos, apresentam ótimos resultados, por isso, é um dos mais populares entre os pesquisadores.

Algoritmos Genéticos, segundo Goldberg (1989), são métodos de otimização e busca inspirados nos mecanismos de evolução de população de seres vivos. Esses algoritmos seguem o princípio da seleção natural e sobrevivência do mais apto, conforme a teoria da evolução de Charles Darwin.

Uma das vantagens de um algoritmo genético simples é a sua ampla aplicabilidade, dado que não usa conhecimento específico sobre o problema, e a simplificação na formulação e solução de problemas de otimização. AGs simples normalmente trabalham com descrições de entrada formadas por cadeias de *bits* de tamanho fixo. Outros tipos de AGs podem trabalhar com cadeias de *bits* de tamanho variável, como por exemplo AGs usados para Programação Genética (RODRIGUES, 2003; FERREIRA, 2001a; FERREIRA, 2001b).

O AG é indicado para a solução de problemas de otimização complexos que envolvem um grande número de variáveis e, conseqüentemente, espaços de soluções de dimensões elevadas. Além disso, em muitos casos, em que outras estratégias de otimização falham na busca de uma solução, os AGs são boas opções. Em alguns casos, no entanto, a performance, relação tempo/qualidade da solução, não é satisfatória (GEN; CHENG, 1997), por isso, trabalhos são desenvolvidos com o intuito de melhorá-la (PARK *et al.*, 2000; RONG-LONG; KOZO, 2005; RAJAN *et al.*, 2002; WU *et al.*, 2004; RUTTKAY *et al.*, 1995; YEN et al., 1998; YANG; DOUGLAS, 1998; MUSIL et al., 1999; CHAINATE *et al.*, 2007). Alguns trabalhos focam na velocidade, embora a maioria busque a eficácia. Neles, a manutenção da diversidade genética para evitar convergência prematura é um ponto bastante pesquisado (TACKETT, W.A.; CARMI, A., 1994; MAHFOUD, 1992; SHIMODAIRA, 2002; MAHFOUD, 1995).

Várias são as alterações propostas que provocam melhorias no AG para as mais diversas aplicações (CHANG *et al.*, 2008; MATHIAS; WHITLEY, 1992). Essas modificações, contudo, são direcionadas a uma aplicação ou, no máximo, a uma classe de problemas. Por isso, deve-se estudar novos operadores ou modificações que, de preferência, tenham uma maior abrangência e, assim, possam ser largamente utilizadas, assim como o próprio AG, conforme afirmam Mitchell e Forrest (1994). Por isso, este trabalho procura apresentar uma aplicabilidade mais ampla desse algoritmo.

Com o objetivo de auxiliar, agilizar a evolução dos AGs e evitar a perda na qualidade da solução final, este trabalho aborda o Algoritmo Auxiliar Paralelo (AAP). O AAP é um algoritmo auxiliar executado em um fluxo paralelo dos AGs. Ele recombina cromossomos para aproveitar ao máximo as informações presentes nos indivíduos, oriundos das populações criadas pelos AGs ou gerados pelo operador do AAP. Ressalta-se que o paralelismo atribuído ao AAP é em relação ao fluxo de execução do AG e não quanto ao paralelismo de processamento.

Com o intuito de aferir a proposta quanto à eficácia, aqui definida como a capacidade de atingir o ótimo global, e à eficiência, aqui concebida como a capacidade de acelerar a evolução do AG, optou-se por abordar problemas *benchmarks,* que possuem alto nível de complexidade. Eles foram usados neste trabalho para estabelecer os cenários de teste.

Este artigo está organizado da seguinte forma: a seção 2 descreve o AG canônico; a seção 3 descreve o Algoritmo Auxiliar Paralelo e seus operadores; a seção 4 discorre sobre a convergência prematura; a seção 5 apresenta os experimentos e os resultados analisados e sintetizados na seção 5.2; por fim, são apresentadas as conclusões e as perspectivas para pesquisas futuras.

2. AG Canônico

A implementação de um algoritmo genético, segundo Whitley (1994), começa com uma população formada por cromossomos gerados aleatoriamente. Cada cromossomo da população é avaliado e associado a uma probabilidade de reprodução, de tal forma que as maiores probabilidades são associadas aos cromossomos que representam uma melhor solução para o problema. O cromossomo também recebe um valor de aptidão chamado *fitness,* que é tipicamente definido com relação à população corrente.

A execução do AG pode ser dividida em duas fases. Na primeira, a seleção é aplicada na população corrente, criando uma população intermediária; já na segunda fase, a reprodução (recombinação) e a mutação são aplicadas na população intermediária para criar a próxima geração. Em Goldberg (1989), essa implementação básica é referenciada como *Simple Genetic Algorithm* (SGA). Neste trabalho, é chamada também de AG canônico. A figura 1, abaixo, mostra o fluxo de execução do AG. Nela é possível identificar as principais etapas e a arquitetura cíclica do AG.

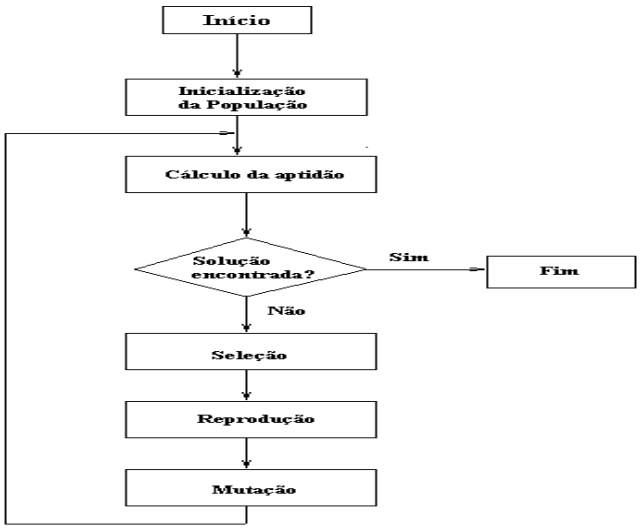


Figura 1: Fluxo de execução do AG canônico.

3. Algoritmo Auxiliar Paralelo

O processo evolutivo dos AGs é composto de um fluxo cíclico. A cada nova geração, indivíduos são gerados e introduzidos na população em substituição a outros existentes, que são descartados. Muitos dos indivíduos eliminados, porém, contêm, nos seus genes, informações importantes para a busca. São descartados sem, ao menos, passarem pela reprodução. Foram, portanto, gerados e eliminados e não contribuíram para a evolução. Em vista disso, pode-se dizer que a cada iteração, ou nova geração, várias informações são descartadas sem análise.

Apesar de esse processo imitar a evolução das espécies, em que um indivíduo pode nascer e morrer sem gerar descendentes, a perda de informação é notória. A informação perdida pode não voltar para a população, causando, em alguns casos, uma redução da eficácia e caracterizando a perda de oportunidade. Pode também voltar pelo processo da mutação, de baixa probabilidade. Nesse caso, perde-se eficiência, pois é necessário um tempo até que a informação retorne e seja aproveitada. Nesse sentido, propõe-se um Algoritmo Auxiliar Paralelo (AAP) que, baseado na população corrente do AG e em novos indivíduos gerados, recombina os cromossomos para melhor aproveitar as informações.

O AAP é um algoritmo executado em um fluxo paralelo ao do AG. Ele recebe uma parcela da população corrente como entrada e emite um indivíduo como saída, que pode ser melhor que o melhor corrente (Figura 2). Desta forma, o AAP abastece o AG de bons indivíduos, agilizando assim, o processo evolutivo do AG.

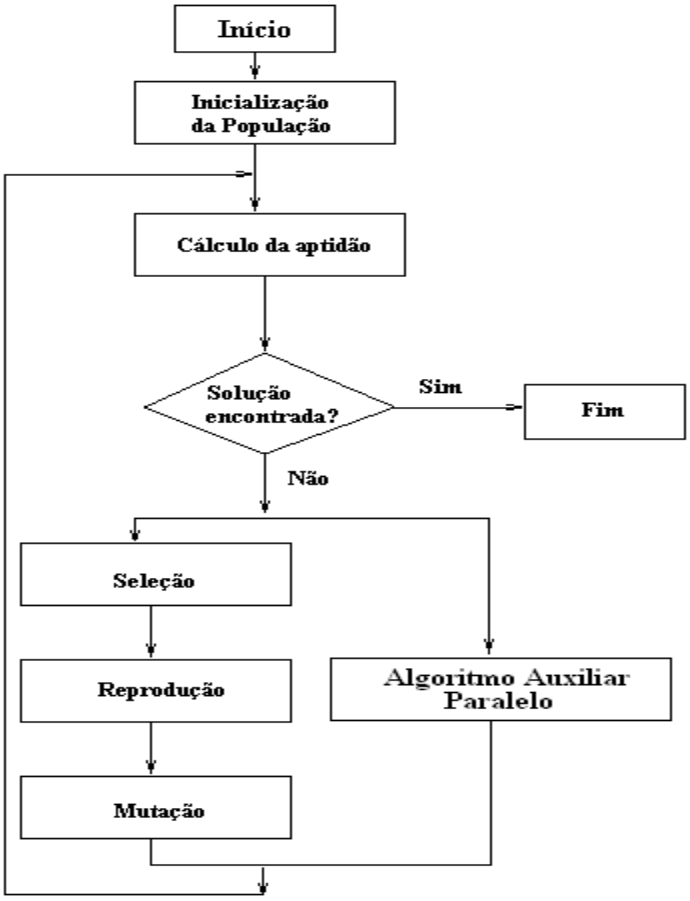


Figure 2: Fluxo paralelo de execução com o Algoritmo Auxiliar Paralelo – AAP.

A forma ideal de aproveitar todas as informações presentes nos indivíduos seria recombiná-los entre si, gene a gene, até encontrar a melhor combinação. Isso seria inviável computacionalmente, principalmente para cromossomos maiores. Devido a isso, optou-se por gerar filhos a partir da recombinação de partes dos cromossomos, de uma parcela da população (*N* indivíduos), com o melhor (seção 3.4). Caso o processo encontre melhor indivíduo e opte por elitismo, esse indivíduo se insere na população em substituição ao melhor corrente. Na ausência do elitismo, o indivíduo gerado substitui qualquer um da população. Caso o operador não produza melhor indivíduo, não há interferência na população.

Existem dois grupos de operadores até o momento para o AAP. O primeiro é constituído de operadores que utilizam somente a população gerada pelo AG como material genético. O segundo é constituído de operadores que alteram cromossomos da população que será recombinada. A estratégia do segundo grupo é enriquecer a população do AAP com informações que podem ser benéficas para o processo de recombinação. O operador AR faz parte do primeiro grupo e os operadores EAR-T, EAR-P e EAR-N do segundo.

Nas seções seguintes, o AAP será apresentado mais detalhadamente, quanto ao seu mecanismo de funcionamento. A seção 3.1 demonstra o fluxo de execução do algoritmo. A 3.2 discorre sobre a divisão do material genético. Já a seção 3.3 apresenta os operadores e a seção 3.4 o processo de recombinação.

3.1 O Fluxo de execução do AAP

No início da execução do AG, a divisão do material genético (seção 3.2) e a quantidade de indivíduos que são usados pelo AAP (*qtdIndiv*) são definidas. Essa configuração é feita uma única vez, no início da execução do algoritmo.

Todos os outros procedimentos do algoritmo são executados a cada geração do AG, após receber a população corrente. Isso pode ser observado no processo abaixo descrito:

1. Encontra o mais apto e rotula-o como Pai;
2. Recebe como parâmetros de entrada: os *N* indivíduos (*qtdIndiv*), que são manipulados no processo de melhoramento genético, e os indivíduos que poderão ser substituídos;
3. No caso das estratégias EAR, dos *N* indivíduos selecionados, a metade (*N/2*) sofre alterações de todo ou de parte(s) dos cromossomos;
4. É feita uma recombinação genética entre o pai (melhor indivíduo) e os indivíduos estipulados na variável *qtdIndiv*. Como o pai é o mesmo para todo o procedimento de recombinação, os filhos gerados são irmãos;
5. Caso haja bons indivíduos gerados, eles são inseridos na nova população no lugar dos indivíduos escolhidos pelo processo de elitismo. Caso não tenha elitismo, os indivíduos são sobrepostos aos determinados no item 2 acima.

A tarefa 5, acima, explica a forma de interferência do AAP na população, gerada pelo AG. A interferência só acontece quando bons indivíduos, melhores que o melhor existente, forem gerados pelo operador proposto.

3.2 Divisão do material genético

Antes da recombinação (cruzamento ou troca de material genético), o cromossomo é dividido em grupos de genes ou gene a gene. A isso se intitula "Divisão do Material Genético", que será trocado. Essa é uma etapa importante do algoritmo, pois se acredita que o conhecimento do problema ajuda o projetista a escolher a melhor forma de dividir o cromossomo.

Uma das opções é que o cromossomo seja dividido segundo as variáveis codificadas. Se, por exemplo, os 3 primeiros genes representam a variável *x* e os 2 últimos a *y*, então, o cromossomo é dividido em dois grupos, um contendo 3 genes e o outro 2 (Figura 3). Dessa forma, durante a recombinação, a troca é feita entre características fenotípicas, que são grupos de genes que representam valores decodificados.

Assim, pode-se dizer que esse processo disponibiliza ao projetista uma ferramenta de transferência de conhecimento para o algoritmo.

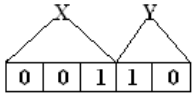


Figure 3: Dois grupos selecionados como material genético de troca.

3.3 Os operadores do AAP

Quatro operadores foram criados para o AAP, como estratégia para auxiliar o AG a explorar o espaço de busca. O AR recombina somente os cromossomos da população corrente sem alterações prévias. O EAR-T muta todos os genes de alguns cromossomos antes de fazer as recombinações. O EAR-P muta parte dos genes de alguns cromossomos antes da recombinação e o EAR-N gera novos indivíduos para participar do processo

de recombinação. Todos os EAR (seção 3.3.2) fazem algum tipo de alteração na população antes da recombinação. Já o AR (seção 3.3.1) utiliza somente a população corrente como material genético. Os operadores são independentes, por isso, não se testou, até o momento, o uso concomitante deles.

3.3.1 Operador *Assisted Recombination*

Dada a constatação de baixo aproveitamento, por parte do AG, do material genético produzido, propõe-se o *Assisted Recombination*, que é um operador do AAP elaborado para melhor explorar as informações genéticas presentes nos cromossomos das populações geradas pelo AG. Espera-se, por consequência, que o operador contribua para um melhoramento genético mais ágil.

O operador é constituído de uma única fase: a recombinação (ver detalhes na seção 3.4). Nela, o operador, como foi descrito no fluxo de execução do AAP (Figura 2), recebe a mesma população que a etapa de seleção e recombina, sem fazer alterações, os *N* indivíduos recebidos do AG com o melhor da mesma população e gera filhos que são analisados. Caso o processo encontre melhor indivíduo e opte-se por elitismo, ele é inserido na população em substituição ao melhor corrente. Na ausência do elitismo, o indivíduo gerado substitui qualquer indivíduo da população. Caso o operador não produza melhor indivíduo, não há interferência na população.

Uma das principais características desse operador é a capacidade de assistir, por isso, o nome *Assisted*, as recombinações feitas e identificar indivíduos com aptidão superior ao melhor indivíduo corrente.

3.3.2 Operadores *Exploratory Assisted Recombination*

Os operadores *Exploratory Assisted Recombination* (EAR) são baseados no *Assisted Recombination* e, por isso, têm semelhanças como: utiliza os N indivíduos da população do AG no processo de melhora e têm capacidade de assistir a recombinação e identificar indivíduos melhores que o melhor corrente.

Assim como o *Assisted Recombination*, o operador EAR tem o objetivo de melhor aproveitar as informações produzidas pelo AG e agilizar o processo de melhoramento genético da população, sem possíveis perdas de qualidade na solução final.

A maior contribuição desse novo operador, em relação ao *Assisted Recombination*, é sua maior capacidade de fazer "busca global", por meio da característica exploratória acrescentada.

Os operadores citados passam por duas fases. A primeira é de alteração de alguns cromossomos recebidos do AG, intitulada de exploração do espaço de busca. A segunda é de recombinação, fase comum a todos os operadores do AAP, descrita na seção 3.4.

Na fase de exploração do espaço de busca, os N indivíduos da população selecionados são divididos pela metade (N/2) e a última porção sofre alteração, de parte ou de todo o cromossomo.

O operador EAR-P muta parte do cromossomo, para isso, é sorteada uma das partes, definidas pela divisão de material genético. O operador EAR-T muta todas as partes do cromossomo. Já o operador EAR-N gera aleatoriamente novos indivíduos. Após alteração, os indivíduos substituem os *N/2* da população.

A Figura 4 descreve o processo, desde a escolha dos N/2 indivíduos até a nova população alterada (N'), que será utilizada na fase de recombinação (seção 3.4). No exemplo, é estabelecida a Divisão do Material Genético em duas partes: a primeira com 3 genes e a segunda com 2. O número de indivíduos provenientes do AG são 12 (qtdIndv). Assim, os últimos 6 (*N/2)* indivíduos sofrem alterações.

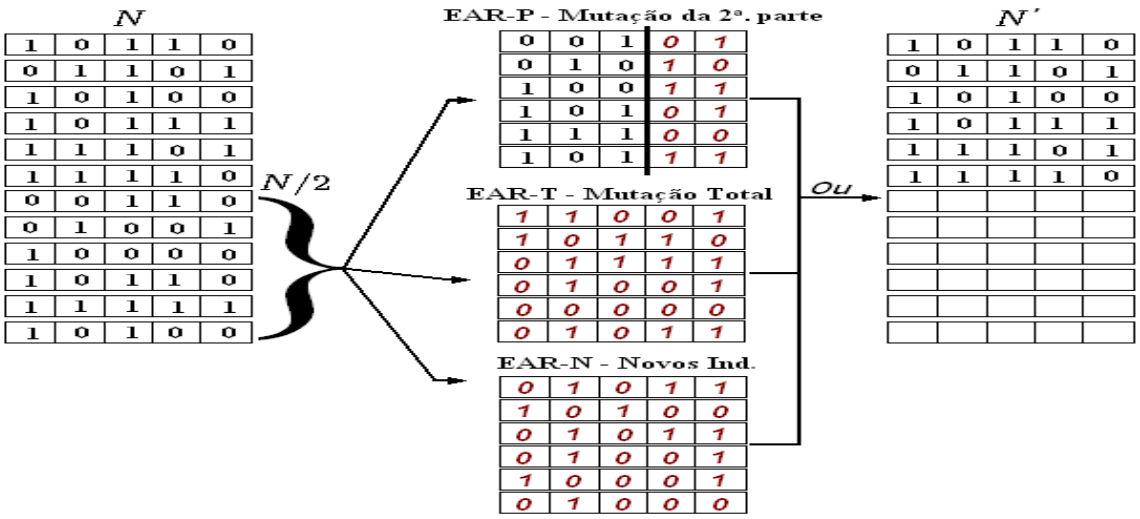


Figure 4: Processo de exploração do espaço de busca do EAR. A mutação de parte é o EAR-P, a mutação de todos os genes é o EAR-T e o EAR-N cria novos indivíduos aleatoriamente.

3.4 A Recombinação

Estabelecidos os grupos de genes (divisão do material genético) e a população dos N' indivíduos, o passo seguinte é a recombinação. Nesse o cromossomo é representado por um vetor. Cada elemento do vetor é um grupo de genes, estabelecido pela divisão do material genético. Assim, estabelecidas 2 partes na divisão, por exemplo, o vetor terá dois elementos.

O melhor indivíduo é reservado como pai. As mães são os indivíduos restantes. A quantidade de mães é limitada pelo parâmetro *qtdIndiv*. Para gerar um filho, a mãe doa um elemento do seu vetor e o pai, com os outros elementos do seu vetor, completa o cromossomo do filho. Esse processo se repete para todas as mães, gerando um grupo de filhos. Outros grupos de filhos são gerados pela troca do elemento doado pela mãe e do complemento do cromossomo pelo pai. Assim, a quantidade de grupos de filhos é igual à quantidade de elementos do vetor, que por sua vez, é igual à quantidade de grupos de genes.

Após a geração de todos os grupos de filhos, o melhor filho é considerado o melhor Super Indivíduo e é comparado ao pai. Caso seja melhor, o filho sobrepõe o pai na população do AAP e recomeça todo o processo de recombinação. Agora o melhor filho é o pai da iteração. Se não for, o algoritmo interrompe o laço de repetição e insere o pai corrente na população final.

Como exemplo de uma iteração da recombinação, a Figura 5 mostra:

- O cromossomo dividido (divisão do material genético) em duas partes. O grupo 1 com 3 genes e o grupo 2 com 2 genes;
- Três mães, estabelecidas pelo parâmetro *qtdIndiv* = 4, sendo, as mães os 3 melhores indivíduos após o pai;
- O primeiro grupo de filhos, gerados pela doação dos primeiros elementos dos vetores das mães e pelo complemento do pai. Por exemplo, o segundo filho do "Filhos do Grupo 1" (01110) é o resultado da união do primeiro elemento da segunda mãe (011) e do complemento, segundo elemento do vetor, do pai (10);
- No segundo grupo de filhos, a mãe doa o segundo elemento do vetor e o pai o completa, com o seu primeiro elemento do vetor. Por exemplo, o terceiro filho do "Filhos do Grupo 2" (00100) é o

resultado da união do segundo elemento da terceira mãe (00) e do complemento do pai (001), primeiro elemento do vetor. Nesse caso, ao contrário do "Filhos do Grupo 1", o filho é formado pela primeira parte do pai e a segunda parte da mãe;

- O melhor indivíduo do grupo de filhos é considerado Super Indivíduo do grupo. O melhor entre os Super Indivíduos é comparado com o pai. Caso seja melhor que o pai, o melhor Super Indivíduo substitui
- o pai na próxima iteração, caso não, o algoritmo é interrompido e retorna o pai corrente.

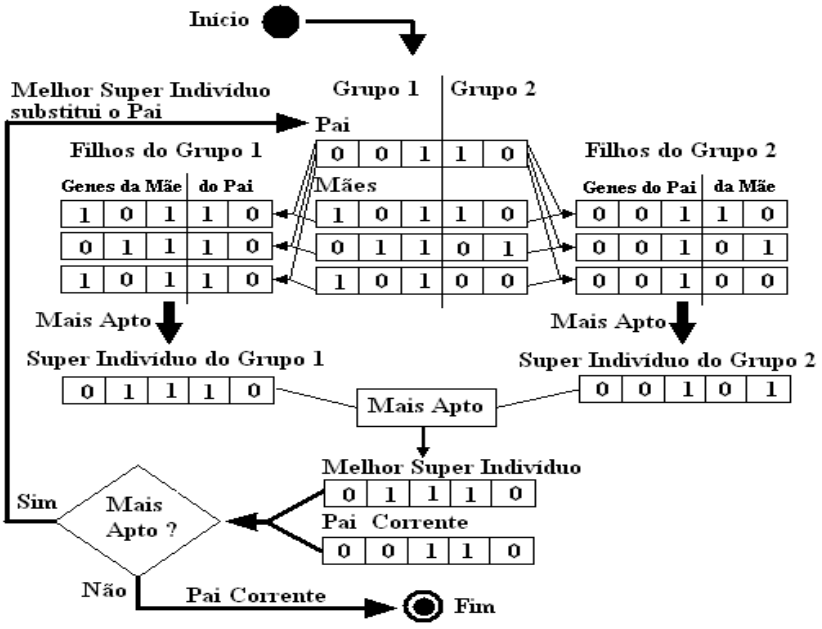


Figure 5: Exemplo de uma recombinação.

4 A Convergência Prematura

Uma das principais características do AAP é a capacidade de melhor aproveitar as informações presentes na população, gerando bons indivíduos que podem agilizar o processo evolutivo dos AGs.

Sempre que se tenta agilizar o processo evolutivo dos AGs uma preocupação fica evidente: a convergência prematura (GOLDBERG, 1989; SULTAN *et al*., 2007) é definida como a perda de diversidade genética na população de forma acelerada e prejudicial à solução final. Apesar de ser considerado um grande problema quando a convergência é para um ótimo local, é desejável se a convergência for para o ótimo global.

A discussão sobre a convergência prematura passa por outros conceitos como *exploration,* exploração global do espaço de busca, e *exploitation,* exploração local do espaço de busca (EIBEN; SCHIPPERS, 1998; SPEARS, 1992). Pode-se definir convergência prematura como pouca *exploration* e prematura *exploitation*. Os algoritmos que usam das duas estratégias, normalmente, no início, usam da busca global para identificar regiões promissoras e, posteriormente, a busca local para encontrar a melhor solução na região.

O AG é um algoritmo que começa sua execução privilegiando a estratégia *exploration,* quando sua diversidade genética é alta, e passa, em um determinado momento, a privilegiar a *exploitation*, quando uma representativa parcela dos indivíduos tem similaridades.

Para evitar a convergência prematura para ótimos locais, é importante fazer uma boa *exploration*. No caso dos AGs, ela é feita pela mutação, com baixa probabilidade, e pela recombinação de soluções geradas pelo início do processo evolutivo, normalmente, com alta diversidade genética. Esse processo, no entanto, é

lento, pois são necessárias várias gerações para se priorizar umas das regiões promissoras.

Fazendo o AG aproveitar melhor seus indivíduos e outros novos, a cada geração pela recombinação, acredita-se que a velocidade pode ser maior sem aumentar as chances de convergência prematura para ótimo local.

5 Experimentos

Para testar o desempenho do algoritmo proposto (AAP) e seus operadores quando acoplados ao AG, optou-se por dois problemas *benchmark*. O primeiro de minimização da função *Rastrigin* e o segundo o da Mochila Multidimensional (Multidimensional *Knapsack Problem*).

Considerando que existem excelentes algoritmos para os problemas citados, o AAP não objetiva ter o melhor desempenho, mas se apresenta como uma proposta de acoplamento benéfica aos Algoritmos Genéticos. Em vista disso, usa-se também como base comparativa um algoritmo híbrido AG/BT, que tem objetivo similar ao AAP - alimentar o AG de bons indivíduos - e estratégia de solução parecidas para o problema, um balanço entre busca local e global.

O híbrido tem a base do AG além do acréscimo do Busca Tabu, que é local. A intervenção do BT no AG se dá a cada 10 iterações do AG canônico. O AG envia o melhor indivíduo corrente e recebe o resultado da busca em vizinhança feita pelo BT.

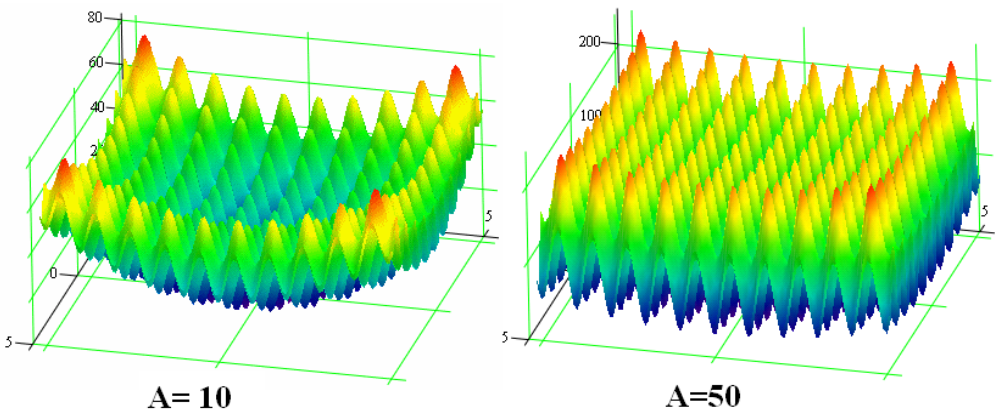
Durante as análises dos resultados, a significância estatística é dada pelo *valor-p* do Teste-T. Compara-se a média do melhor algoritmo com os outros. Se o *valor-p* é menor que 0.05, pode-se afirmar, com 95% de confiança, que existe uma diferença estatística significativa entre as médias.

Os experimentos foram executados em um PC com o processador *Intel Pentium* 4 1.6 GHz e memória de 1 GB.

5.1 A função *Rastrigin*

A função de Rastrigin (Equação 1) é muito utilizada por ser multimodal e ter um comportamento com vários picos e vales. Caracteriza, portanto, vários ótimos locais. A Figura 6 ilustra a função.

$$f(x) = nA + \sum_{i=1}^{n} \left(x_i^2 - A\cos(2\pi x_i) \right);\ \forall i \in [1..n], x_i \in [-5.12,\ 5.12] \qquad (1)$$



A= 10 **A=50**

Figure 6: Função *Rastrigin* bidimensional com A=10 e A=50
Disponível em: <http://www.cs.rtu.lv/dssg/en/staff/rastrigin/>

Dada a função, o objetivo é minimizar *f(x)*, respeitando as restrições: $-5.12 <= X_i <= 5.12$ e valores com quatro casas decimais. Considerando as restrições, o ponto mínimo da função é atingido quando $X_i = 0$. Nesse ponto, a solução é ótima, gerando um $f(X_i) = 0$.

Para esse problema, apresentam-se 4 experimentos. No primeiro, pretende-se analisar a eficácia do AAP na minimização (cenário 1.1) e o comportamento dele quando alterado o parâmetro A da função (cenário 1.2). No segundo, pretende-se analisar o comportamento do AAP quando alterado o número de indivíduos (diversidade genética) na população. Para isso, define-se o cenário 2.1 com 50 indivíduos, o cenário 2.2 com 10 indivíduos e o cenário 2.3 com 5 indivíduos. No terceiro experimento, analisa-se o impacto do aumento de dimensões (variáveis) no AAP. O cenário 3.1 é executado com 2 variáveis e o cenário 3.2 com 10 variáveis. Por fim, o quarto experimento pretende fazer uma comparação entre o melhor operador do AAP para o problema, identificado pelos cenários anteriores, e um algoritmo híbrido (AG-BT), que mescla as boas características de busca global do AG e de busca local do Busca Tabu.

Para cada cenário, 20 execuções de cada algoritmo são feitas. No experimento 3, são 15. A condição de parada é atingir o ótimo global ou 250 gerações. No experimento 4, a condição de parada é atingir o ótimo global ou 250000 avaliações, por ser uma comparação entre algoritmos de estruturação bem diferente.

Para uma comparação justa, a cada execução, uma nova população inicial é criada aleatoriamente e compartilhada entre os algoritmos. Dessa forma, em cada execução, os algoritmos iniciam com a mesma população inicial. As execuções são independentes e sequenciais. O operador de seleção do AG para os experimentos é a roleta, por se tratar de um método com alta pressão de seleção e, assim, dar ao AG característica similar ao AAP, que tem uma alta pressão de seleção.

As Tabelas 1, 2, 3 e 4 mostram, respectivamente, as configurações dos experimentos 1, 2, 3 e 4. Para melhor visualização, algumas siglas foram usadas: PM para a probabilidade de mutação, PC para probabilidade de cruzamento, QtdG para quantidade de genes no cromossomo, OG para ótimo global, DMG para divisão do material genético e QtdIndiv para quantidade de indivíduos para o AAP. Para todos os experimentos definiu-se a PC=0.65 e PM=0.01, por apresentar melhores resultados. No experimento 4, o BT recebe, a cada 10 gerações do AG, o melhor indivíduo corrente e executa até 100 iterações sem melhoras (BTMax=100). Além disso, a Lista Tabu tem 10 elementos e 30% da vizinhança é analisada.

Tabela 1: Configurações do Experimento 1

Experimento 1									
Cenários	**Parâmetros do AG**					**Parâmetros AAP**		**Parâmetros da Função**	
1.1	**Pop**	**PM**	**PC**	**QtdG**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50 ind.	0.01	0.65	34	OG ou 250G	4	45	2	10
1.2	**Pop**	**PM**	**PC**	**QtdG**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50 ind.	0.01	0.65	34	OG ou 250G	4	45	2	50

Tabela 2: Configurações do Experimento 2

Experimento 2			
Cenários	**Parâmetros do AG**	**Parâmetros AAP**	**Parâmetros da**

								Função	
2.1	**Pop**	**PM**	**PC**	**QtdG**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	30 ind.	0.01	0.65	34	OG ou 250G	2	45	2	10
2.2	**Pop**	**PM**	**PC**	**QtdG**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	10 ind.	0.01	0.65	34	OG ou 250G	2	9	2	10
2.3	**Pop**	**PM**	**PC**	**QtdG**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	5 ind.	0.01	0.65	34	OG ou 250G	2	4	2	10

Tabela 3: Configurações do Experimento 3

Experimento 3									
Cenários	**Parâmetros do AG**					**Parâmetros AAP**		**Parâmetros da Função**	
3.1	**Pop**	**PM**	**PC**	**QtdG**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50 ind.	0.01	0.65	34	OG ou 250G	4	45	2	10
3.2	**Pop**	**PM**	**PC**	**QtdG**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50 ind.	0.01	0.65	170	OG ou 250G	10	45	10	10

Tabela 4: Configurações do Experimento 4

Experimento 4									
Cenários	**Parâmetros do AG**		**Parâmetros do Busca Tabu**			**Parâmetros AAP**		**Parâmetros da Função**	
4.1	**Pop**	**QtdG**	**BTMax**	**Tabu**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50 ind.	34	100	10	OG ou 250G	4	45	2	10
4.2	**Pop**	**QtdG**	**BTMax**	**Tabu**	**Parada**	**DMG**	**QtdIndiv**	**Variáveis**	**A**
	50 ind.	170	100	10	OG ou 250G	10	45	10	10

5.1.1 Resultados

São coletados nos experimentos: o menor valor de função encontrado (MFO), a quantidade de gerações (QG) até atingir a condição de parada, o número de vezes que atinge o ótimo global (QO) e, no caso do experimento 4, a quantidade de avaliações (Aval) até atingir a condição de parada. Além desses elementos, valores estatísticos descritivos e de inferência são apresentados, como o valor máximo (Max), o valor mínimo (Min), e a média (Md). As Tabelas 5, 6, 7 e 8 mostram, respectivamente, os resultados dos experimentos 1, 2, 3 e 4.

Tabela 5: Resultados do Experimento 1

Experimento 1															
Cenário 1.1															
	AG			**AR**			**EAR-T**			**EAR-P**			**EAR-N**		
	MFO	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**
Max	0,9950	250	30,42	0	83	9,01	0	95	10,58	0	87	9,00	**0**	**43**	**6,26**
Min	0	174	12,73	0	12	1,28	0	6	0,72	0	7	0,90	**0**	**8**	**1,14**
Md	0,0645	228,65	18,78	0	36,6	4,41	0	29,6	3,515	0	30	3,53	**0**	**23,3**	**3,03**
QO	**8/20**			**20/20**			**20/20**			**20/20**			**20/20**		
Cenário 1.2															
	AG			**AR**			**EAR-T**			**EAR-P**			**EAR-N**		

	MFO	QG	Ts	MFO	QG	Ts	MFO	QG	Ts	MFO	QG	Ts	MFO	QG	Ts
Max	0	250	23,42	0	167	20,67	0	107	12,40	0	108	13,11	**0**	**88**	**12,01**
Min	0,0259	112	10,03	0	7	0,74	0	11	1,19	0	8	0,88	**0**	**7**	**0,83**
Md	0,0020	220,6	17,70	0	52,65	6,23	0	45,1	5,12	0	45,15	5,36	**0**	**27,9**	**3,48**
QO	**10/20**			**20/20**			**20/20**			**20/20**			**20/20**		

Tabela 6: Resultados do Experimento 2

Experimento 2															
Cenário 2.1															
	AG			**AR**			**EAR-T**			**EAR-P**			**EAR-N**		
	MFO	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**
Max	0,0114	250	22,83	0	211	14,94	0	194	13,80	0	169	14,13	**0**	**138**	**10,59**
Min	0	93	3,14	0	37	2,86	0	34	2,11	0	25	1,94	**0**	**17**	**1,33**
Md	0,0011	207,2	12,26	0	95,35	7,09	0	89,85	6,08	0	77,7	5,85	**0**	**73,65**	**5,81**
QO	**10/20**			**20/20**			**20/20**			**20/20**			**20/20**		
Cenário 2.2															
	AG			**AR**			**EAR-T**			**EAR-P**			**EAR-N**		
	MFO	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**
Max	5,9252	250	24,42	1,2361	250	28,78	1,9950	250	26,17	2,4723	250	27,62	**0**	**198**	**23,19**
Min	0	59	5,31	0	86	9,28	0	49	5,23	0	69	7,55	**0**	**60**	**7,02**
Md	1,9423	230,8	21,67	0,6142	212,75	23,67	0,7208	204	20,49	0,5209	203,55	21,57	**0**	**112,25**	**13,17**
QO	**4/20**			**8/20**			**8/20**			**9/20**			**20/20**		
Cenário 2.3															
	AG			**AR**			**EAR-T**			**EAR-P**			**EAR-N**		
	MFO	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**
Max	6,1766	250	24,00	8,9036	250	28,14	5,9239	250	25,55	7,9800	250	26,94	**2,5E-05**	**250**	**22,55**
Min	0	234	9,00	0	133	9,64	0	139	8,55	0	104	6,44	**0**	**70**	**7,36**
Md	3,0302	249,2	16,89	2,2651	241,6	20,79	2,2686	233,7	17,74	2,7175	233,6	18,88	**2,3E-06**	**165,1**	**14,53**
QO	**1/20**			**3/20**			**4/20**			**3/20**			**17/20**		

Tabela 7: Resultados do Experimento 3

Experimento 3															
Cenário 3.1															
	AG			**AR**			**EAR-T**			**EAR-P**			**EAR-N**		
	MFO	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**
Max	0,9949	250	16,73	0	125	10,57	0	120	9,72	0	99	8,33	**0**	**93**	**8,53**
Min	0	128	8,06	0	29	2,45	0	34	2,70	0	31	2,55	**0**	**19**	**1,83**
Md	0,0760	237,47	14,94	0	71,47	6,08	0	67,73	5,45	0	73	6,01	**0**	**53,6**	**4,87**
QO	**3/15**			**15/15**			**15/15**			**15/15**			**15/15**		
Cenário 3.2															
	AG			**AR**			**EAR-T**			**EAR-P**			**EAR-N**		
	MFO	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**	**MFO**	**QG**	**Ts**
Max	12,0148	250	37,88	0	213	86,64	0	221	98,45	0	231	140,86	**0**	**159**	**69,05**
Min	38,1304	250	17,33	0	92	44,63	0	89	41,37	0	58	28,53	**0**	**80**	**38,17**
Md	23,9279	250	26,10	0	132,8	58,70	0	127,13	56,41	0	146,8	66,95	**0**	**113,8**	**50,37**
QO	**0/15**			**15/15**			**15/15**			**15/15**			**15/15**		

Tabela 8: Resultados do Experimento 4

Experimento 4							
Cenário 4.1							
	AG/BT			**EAR-N**			
	MFO	**Ts**	**Aval**	**MFO**	**QG**	**Ts**	**Aval**
Max	0	19,31	100737	**0**	**66,00**	**6,17**	**26532**
Min	0	1,56	9906	**0**	**8,00**	**0,68**	**3442**

Md	0	6,85	43031	**0**	**20,50**	**1,86**	**8285**
QO	**20/20**			**20/20**			
Cenário 4.2							
	AG/BT			**EAR-N**			
	MFO	**Ts**	**Aval**	**MFO**	**QG**	**Ts**	**Aval**
Max	31,0435	87,66	288026	**0**	**217,00**	**108,70**	**209730**
Min	8,9449	68,77	250562	**0**	**82,00**	**38,44**	**76510**
Md	18,5629	80,11	263610	**0**	**125,00**	**58,71**	**114842**
QO	**0/20**			**20/20**			

5.1.2 Análise dos Resultados

O objetivo dos experimentos é mensurar no problema abordado, do algoritmo proposto, a eficiência, considerada como capacidade de acelerar o melhoramento genético do AG, e a eficácia, definida como a capacidade de atingir o ótimo global. Neste trabalho, mede-se a eficiência pelo tempo gasto (Ts). Quanto menor o tempo, maior a eficiência. Já a eficácia é analisada pela QO - quanto maior a taxa de sucesso, melhor é a eficácia - e pela média do MFO - quanto menor a média da função, mais próximo do ótimo.

Analisando o Cenário 1.1, Tabela 5, quanto à eficácia, identifica-se ótimo desempenho do AAP e seus operados, pois todos atingiram o ótimo global nas 20 execuções. Já o AG sem o auxílio do AAP não obteve bons resultados: das 20 execuções, encontrou o ótimo global em apenas 4 (20%). No Cenário 1.2, Tabela 5, o parâmetro A da função foi aumentado, o que caracteriza vales mais profundos. Mesmo assim, o AAP e seus operadores continuaram com excelentes resultados, 100% de eficácia. Já o AG apresenta um desempenho mediano: 50% de eficácia.

Analisando a eficiência no experimento 1, no cenário 1.1, o EAR-N tem o melhor resultado, seguido dos EAR-T, EAR-P e AR. *valor-p* = 0,012117 e 4,9025E-17, respectivamente. Já no cenário 1.2, identifica-se uma pequena perda de eficiência do AAP. O AG sem AAP apresenta uma eficiência bem inferior, mesmo analisando somente as execuções em que se atinge o ótimo global. As diferenças apresentadas entre as médias do Ts são estatisticamente significativas entre EAR-N e AR e entre EAR-N e AG para ambos os cenários.

No experimento 2, é possível medir o impacto da redução de indivíduos (diversidade genética) no algoritmo proposto. No cenário 2.1, com 30 indivíduos na população, todos os operadores do AAP apresentam eficácia de 100% para todos os operadores, já o AG apresenta 50% dessa eficácia. No cenário 2.2, com a redução para 10 indivíduos, percebe-se uma grande perda de desempenho por parte do AAP, exceto o operador EAR-N, pois o EAR-T apresenta 45%, o EAR-P e o AR 40%. O EAR-N manteve a eficácia de 100% e o AG baixou para 20%. No cenário 2.3, com apenas 5 indivíduos, comprova-se a tese de que os operadores EAR-P, EAR-T e AR, além do AG, sofrem com a redução de material genético, pois tiveram apenas 20%, 15%, 15% e 5%, respectivamente. O EAR-N também apresenta redução de eficácia, no entanto, obteve um bom resultado, pois, mesmo com pouquíssima diversidade genética, obteve 85% de eficácia. Além da dependência direta da eficácia do AAP ao número de indivíduos, o experimento 2 demonstra a contribuição do AAP para o AG nos cenários com pouco material genético. Com exceção do cenário 2.1, em que o EAR-N apresentou diferença significativa com AG (*valor-p* = 2,63E-07) e não significativa com os demais AAPs, a diferença apresentada entre a média Ts do EAR-N com as demais é estatisticamente significativa nos cenários do experimento 2. Obtém, assim, no cenário 2.2, *valor-p* de 9,35E-07 (entre EAR-

N e AR), 9,37E-05 (entre EAR-N e EAR-P), 0,000618 (entre EAR-N e EAR-T) e 1,08E-05 (entre EAR-N e AG).

No experimento 3, é possível identificar o comportamento dos algoritmos quando se aumenta a dimensão do problema, consequentemente, a complexidade. No cenário 3.1, a função tem 2 dimensões, variáveis X e Y, e o algoritmo AAP apresenta 100% de eficácia. Já o AG sem o AAP apresenta apenas 20% dessa eficácia. Analisando o tempo gasto (Ts), identifica-se uma eficiência maior do EAR-N (4,87), seguido de EAR-T (5,45), EAR-P (6,01) e AR (6,08). No cenário 3.2, a função tem 10 dimensões e o algoritmo AAP mantém os 100% de eficácia, no entanto, identifica-se uma redução da eficiência, pois as médias dos Ts são 50,37 (EAR-N), 56,41 (EAR-T), 58,70 (AR) e 66,95 (EAR-P).

A manutenção da eficácia de 100%, no cenário 3.2, mostra a robustez do AAP para problemas multidimensionais. Nesse cenário, o AG apresenta seu pior desempenho, pois não atinge o ótimo global nenhuma vez. Isso demonstra a dificuldade de ele evoluir em ambientes altamente multimodais e multidimensionais.

Por fim, no experimento 4, são analisados o acréscimo do BT no AG e o desempenho do AAP/EAR-N versus do AG/BT. Percebe-se que o acréscimo da Busca Tabu no AG foi favorável, pois aumentou a eficácia (QO) de 20% (cenário 3.1) para 100% (cenário 4.1), além de diminuir a média do MFO no problema de 10 dimensões, de -23,92, no cenário 3.2, para -18,56, no cenário 4.2.

Quando comparado o desempenho do AG/BT com o EAR-N (melhor AAP para a aplicação), percebe-se que EAR-N foi superior, pois teve no cenário 4.1 média Ts de 1,86 versus 6,85 do AG/BT (*valor-p* = 2,26782E-05), além de ter encontrado o ótimo em média com 8285 avaliações versus 43031 do AG/BT (*valor-p* = 6,79E-07). Analisando o cenário 4.2, percebe-se uma eficácia superior do EAR-N, pois obteve 100% de taxa de sucesso versus 0% do AG/BT, e uma alta sensibilidade do AG/BT quando se aumenta a dimensão do problema.

5.2 Mochila Multidimensional

O Problema da Mochila Multidimensional (PMM) é um *benchmark* largamente estudado na literatura, por ser um problema de otimização combinatorial NP-Difícil e por ocorrer em vários tipos de aplicações reais (PUCHINGER *et al*., 2006). O PMM é uma variante do Problema da Mochila, que é menos complexo, e para este trabalho é formulado conforme as equações 1, 2 e 3.

Maximiza

$$\sum_{j=1}^{n} c_j \cdot x_j \qquad (2)$$

sujeito a

$$\sum_{j=1}^{n} a_{i,j} \cdot x_j \leq b_i, \forall i = 1..m \qquad (3)$$

$$x_j \in \{0,1\}, 1 \leq j \leq n \qquad (4)$$

em que n é o número de objetos, m é o número de dimensões da mochila, cj representa o benefício do objeto j na mochila, xj é uma variável binária que indica se o objeto j está guardado na mochila (xj =1) ou está fora

(xj =0), bi representa a capacidade da dimensão i-th da mochila, e aij representa as entradas da matriz de pesos dos objetos para cada dimensão.

A matriz guarda os valores de peso de cada objeto para cada dimensão. Assim, a matriz tem o tamanho de NxM, sendo n colunas e m linhas.

O objetivo é selecionar um subconjunto de objetos que maximize o benefício total, (ver equação 2); no entanto, os itens escolhidos não podem exceder a capacidade de cada dimensão (ver equação 3).

Esse problema, diferentemente da função *Rastrigin*, é um problema restritivo e, por isso, foi escolhido para verificar o comportamento da proposta nesse tipo de aplicação.

Para os testes, foram coletadas, na literatura, 21 instâncias do PMM (Petersen, 1967), disponíveis na *OR-Library*[1]. Esses problemas consistem de *m*= 5 a 10 e *n*=6 a 100.

Para cada instância, montou-se um cenário e foram feitas 10 execuções, em que a condição de parada é atingir o ótimo global ou 50000 avaliações da função objetivo.

As instâncias do experimento 2 têm um fator de correlação alfa (α) que determina o nível de correlação entre a capacidade de cada dimensão da mochila e o somatório dos pesos dos objetos para essa dimensão.

A codificação é binária. Cada gene representa um objeto e a função a ser maximizada é da equação 1, que não penaliza as soluções infactíveis. O uso exclusivo de soluções factíveis melhora o desempenho do algoritmo para o problema (HOFF *et al.*, 1996). Em vista disso, foi criado um operador de reparação construtivo para as soluções que não atendem a restrição tornarem-se factíveis. O operador é baseado no trabalho de Chu e Beasley (1998), mas menos guloso e menos determinístico. O operador implementado também tem uma fase de retirada e outra de inserção de objetos na solução, no entanto, ao invés de inserir sempre o mais apto e retirar o menos apto, faz um torneio entre os indivíduos para verificar qual será inserido ou retirado. A pressão de seleção do torneio para esse problema é de 15% para a retirada e para a inserção, por apresentar melhor resultado.

A população inicial é, normalmente, fator preponderante no sucesso dos AGs, por isso, foi usada a mesma heurística construtiva de inserir objetos na solução utilizada no operador de reparação. A pressão de seleção do torneio nesse caso é de 20%, por apresentar melhor resultado.

Para uma comparação justa, a cada execução, uma nova população inicial é criada aleatoriamente e compartilhada entre os algoritmos. Dessa forma, em cada execução, os algoritmos iniciam com a mesma população inicial. As execuções são independentes e sequenciais.

O operador de seleção do AG para os experimentos é o Torneio, por se tratar de um método largamente utilizado na literatura. Para todos os experimentos definiu-se a PC=0.8, PM=1/N e Pop= 5*N, sendo N o número de objetos (HOFF *et al.*, 1996). O BT é executado a cada 10 iterações do AG. As configurações dos experimentos estão nas Tabela 9 e 10 abaixo:

Tabela 9: Configurações dos algoritmos para o experimento 1

Experimento 1				
Cenários	**Parâmetros do AG**	**Parâmetros do Busca Tabu**	**Parâmetros AAP**	**Parâmetros do Problema**

[1]As instâncias estão disponíveis em: <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html>

	Pop	QtdG	BTMax	Tabu	DMG	QtdIndiv	N	M	Melhor
1	5*N	N	100	10	2	20%Pop	6	10	3800
2	5*N	N	100	10	2	20%Pop	10	10	8706,1
3	5*N	N	100	10	2	20%Pop	10	15	4015
4	5*N	N	100	10	2	20%Pop	10	20	6120
5	5*N	N	100	10	2	20%Pop	10	28	12400
6	5*N	N	100	10	2	20%Pop	5	39	10618
7	5*N	N	100	10	2	20%Pop	5	50	16537

Tabela 10: Configurações dos algoritmos para o experimento 2

Experimento 2									
Cenários	**Parâmetros do Problema**			**Parâmetros do AG**		**Parâmetros do Busca Tabu**		**Parâmetros AAP**	
	N	**M**	**α**	**Pop**	**QtdG**	**BTMax**	**Tabu**	**DMG**	**QtdIndiv**
2.1 a 2.10	5	100	0,25	50	N	50	2	2	20%Pop
2.11 e 2.12	5	100	0,50	50	N	50	2	2	20%Pop
2.13 e 2.14	5	100	0,75	50	N	50	2	2	20%Pop

5.2.1 Resultados

O maior valor de função encontrado (MFO), o número de vezes que atinge o ótimo global (QO), o tempo gasto em segundos (Ts) e a quantidade de avaliações (Aval) até atingir a condição de parada são dados coletados nos experimentos. No experimento 2, é calculado o *Gap%*, que é a diferença entre o maior valor encontrado na literatura e o alcançado pelos algoritmos.

A Tabela 11 apresenta os resultados dos algoritmos AG, AAP/AR (AR), AAP/EAR-P (EAR-P), AAP/EAR-T (EAR-T) para o experimento 1. A Tabela 12 apresenta os resultados dos algoritmos AAP/EAR-N (EAR-N) e AG/BT para o experimento 1. A Tabela 13 apresenta os resultados dos algoritmos para o experimento 2.

Tabela 11: Resultado do experimento 1 para os algoritmos AG, EAR-T, EAR-P e AR

		Experimento 1											
		AG			EAR-T			EAR-P			AR		
Cenário		**MFO**	**Ts**	**Aval**	**MFO**	**Ts**	**Aval**	**MFO**	**Ts**	**Aval**	**MFO**	**Ts**	**Aval**
1	**Md**	3800	0,05	42	3800	0,05	47	**3800**	**0,05**	**47**	3800	0,05	51
	QO	10/10			10/10			**10/10**			10/10		
2	**Md**	8706,1	0,32	325	8706,1	0,26	326,8	**8706,1**	**0,15**	**199**	8706,1	0,20	248,4
	QO	10/10			10/10			**10/10**			10/10		
3	**Md**	4015	0,84	735	4015	0,57	614,3	**4015**	**0,46**	**511,3**	4015	0,44	442
	QO	10/10			10/10			**10/10**			10/10		
4	**Md**	6120	0,50	400	6120	1,69	1570,4	**6120**	**0,97**	**894,2**	6120	0,13	131,4
	QO	10/10			10/10			**10/10**			10/10		
5	**Md**	12400	14,86	10402	12400	7,05	5524,8	**12400**	**3,05**	**2583,6**	12400	3,81	3169,8
	QO	10/10			10/10			**10/10**			10/10		
6	**Md**	10589,6	59,05	50115	10592,6	49,98	50111,8	**10599**	**51,77**	**50128,3**	10596	54,83	50133,6
	QO	0/10			0/10			**0/10**			0/10		
7	**Md**	16473,4	66,17	50000	16494	61,41	50159,8	**16516,4**	**58,39**	**48520,2**	16500,6	60,83	49205,4
	QO	0/10			0/10			**2/10**			1/10		

Tabela 12: Resultado do experimento 1 para os algoritmos EAR-N e AG/BT

		Experimento 1 continuação					
		EAR-N			AG/BT		
Cenário		MFO	Ts	Aval	MFO	Ts	Aval
1	Md	3800	0,05	51	3800	0,05	42
	QO	10/10			10/10		
2	Md	8706,1	0,17	214,4	8706,1	0,29	325,1
	QO	10/10			10/10		
3	Md	4015	0,40	392,4	4015	0,44	452,6
	QO	10/10			10/10		
4	Md	6120	0,18	159	6120	1,01	1170,5
	QO	10/10			10/10		
5	Md	12400	8,46	6173,6	12399	10,92	13031,5
	QO	10/10			9/10		
6	Md	10595,7	68,38	50165	10585,6	38,56	50125,9
	QO	0/10			0/10		
7	Md	16510,9	77,07	47055,2	16492,8	42,57	50985,2
	QO	2/10			1/10		

Tabela 13: Resultado do experimento 2 para os algoritmos

	Experimento 2											
	AG		EAR-T		EAR-P		AR		EAR-N		AG/BT	
Cenário	*Gap%*	Ts	*Gap%*	Ts	*Gap%*	Ts	*Gap%*	Ts	*Gap%*	Ts	*Gap%*	Ts
1	0,558	189,969	0,656	241,36	0,213	216,657	**0,455**	**193,015**	0,394	233,625	1,046	44,625
2	0,101	158,437	0,202	228,674	0,091	207,51	**0,202**	**180,86**	0,101	230,32	0,610	67,83
3	0,234	190,94	0,242	240,82	0,301	215,22	**0,119**	**194,66**	0,119	234,54	0,119	69,17
4	0,954	192,54	0,988	245,26	1,126	219,795	**0,699**	**194,965**	1,037	231,665	1,557	67,28
5	0,277	191,395	0,288	241,735	0,217	216,505	**0,217**	**194,475**	0,369	236,385	0,329	67,56
6	0,154	192,16	0,049	246,46	0,154	220,05	**0,049**	**195,69**	0,049	237,17	0,154	69,21
7	0,584	194,305	0,334	243,36	0,477	220,02	**0,557**	**198,515**	0,539	242,17	0,807	69,065
8	0,000	193,375	0,171	244,31	0,000	219,055	**0,000**	**195,79**	0,000	238,49	0,092	68,945
9	0,516	193,5	0,570	243,39	0,743	219,62	**0,516**	**195,33**	0,273	238,06	0,834	68,56
10	0,283	192,78	0,553	240,84	0,283	217,84	**0,000**	**193,43**	0,283	234,64	0,291	69,53
Md	**0,366**		**0,405**		**0,361**		**0,281**		**0,316**		**0,584**	
11	0,122	193,835	0,122	191,225	0,122	197,91	**0,122**	**205,605**	0,122	280,085	0,122	71,545
12	0,235	187,35	0,221	180,13	0,209	183,65	**0,235**	**186,14**	0,235	256,85	0,613	68,22
Md	**0,178**		**0,171**		**0,165**		**0,178**		**0,178**		**0,368**	
13	0,000	171,6	0,000	163,65	0,000	178,81	**0,000**	**200,35**	0,000	325,34	0,301	72,05
14	0,269	185,22	0,272	162,44	0,351	183,12	**0,179**	**197,97**	0,280	330,38	0,288	72,35
Md	**0,135**		**0,136**		**0,176**		**0,089**		**0,140**		**0,295**	
Md Total	**0,306**		**0,333**		**0,306**		**0,239**		**0,271**		**0,512**	

5.2.2 Análise dos Resultados

Analisando os cenários 1 e 2, identifica-se um comportamento similar entre os algoritmos, pois todos obtiveram 100% de eficácia e excelente eficiência. Apesar do pequeno acréscimo de Ts nos algoritmos AG e AG/BT, a diferença é mínima e desconsiderável. No cenário 3, todos os algoritmos mantiveram os 100% de eficácia e boa eficiência, apesar de apresentarem um acréscimo de Ts em relação aos cenários 1 e 2. Destaca-se a diferença do Ts do AG (0,84s) em relação aos demais algoritmos (0,46s em média). No cenário 4, manteve-se a eficácia de 100% de todos os algoritmos, mas a eficiência teve variações. Houve destaque positivo para o AR e o EAR-N, que gastaram, respectivamente, 0,13s e 0,18s de Ts, além de 131,4 e 159 de Aval. No cenário 5, o AG/BT teve 90% de eficácia e os demais algoritmos 100%. Quanto à eficiência, destaca-se positivamente o EAR-P, que obteve os Ts e Aval mais baixos.

No cenário 6, nenhum dos algoritmos alcançou o ótimo global nas 10 execuções, provavelmente, por ser um problema mais complexo e pela condição de parada de 50000 Aval. Como o objetivo é fazer uma análise de impacto do AAP no AG, o cenário 6 demonstra, pela média da MFO, que os AAP tiveram os melhores resultados, com destaque para o EAR-P com média de 10599 Aval (*Gap%* = 0,17%). Destaca-se negativamente o desempenho do AG/BT, que teve, nesse cenário, resultado pior que o AG, provavelmente, causado pela convergência prematura para mínimos locais oriunda do acréscimo da busca local. No cenário 7, quatro algoritmos alcançaram o ótimo global em uma das dez execuções, o EAR-P com 20% de eficácia, o AR com 10%, o EAR-N com 20% e o AG/BT com 10%. Analisando a média da MFO, os AAPs tiveram os melhores resultados: o EAR-P foi o melhor novamente (*Gap* = 0,12%), seguido de EAR-N (*Gap* = 0,16%), AR (*Gap* = 0,22%) e EAR-T (*Gap* = 0,26%). O AG/BT foi melhor que o AG, mas foi pior que os AAPs.

No experimento 2, o EAR-P foi melhor nas instâncias com $\alpha = 0{,}50$, no entanto, o AR foi o melhor na maioria das outras e, por isso, obteve a melhor média do experimento. Nas instâncias com coeficiente mais alto ($\alpha = 0{,}75$), o AR teve o melhor desempenho entre os cenários.

Os resultados apresentados do AG, neste trabalho, foram melhores que os apresentados no trabalho de Khuri *et al*. (1994). Acredita-se que a configuração atribuída para o AG, neste trabalho, apesar de similar, favoreceu o desempenho, com destaque para as sugestões de configurações feitas pelo trabalho de Hoff *et al.* (1996).

Os resultados apresentados pelo AAP não são os melhores da literatura, como mostram Fleszar *et al.* (2009), no entanto, atenderam o objetivo de mostrar que o acoplamento do AAP no AG traz benefícios, que são tão bons, ou melhores, do que o algoritmo de estratégia similar AG/BT.

6 Considerações Finais

Alguns trabalhos na literatura (SINGH; DEB, 2006; RAJAN *et al*., 2002; YANG; DOUGLAS, 1998) propõem novos algoritmos que, de forma independente, têm melhores resultados que o AG em alguns *benchmarks*. Este trabalho, no entanto, propõe um algoritmo de acoplamento (AAP), que possibilita um melhor aproveitamento das informações geradas pelo AG ou pelo próprio AAP, por meio de seus operadores.

Algumas instâncias de problemas *benchmark* foram aplicadas para medir o impacto do AAP no AG. Apesar de existirem algoritmos, normalmente com aplicabilidade mais restrita, com desempenho melhor do que o AAP para os problemas abordados (*No-Free-Lunch Theorem,* em Wolpert (1995)), os testes demonstraram que o acoplamento do AAP pode ser benéfico para os Algoritmos Genéticos e tem aplicabilidade mais ampla, por não usar conhecimento do domínio para guiar a busca.

Entre os operadores do AAP, o EAR-N mostrou-se mais eficaz e eficiente que os demais no problema *Rastrigin*. Já no problema PMM, o EAR-P foi melhor para as instâncias menores (experimento 1) e o AR o melhor para as instâncias maiores (experimento 2).

Apesar dos resultados significativos, o algoritmo proposto não foi testado em todos os tipos de problemas abordados com o AG. Em vista disso, em trabalhos futuros sugere-se: a aplicação do AAP a outros tipos de problemas e o desenvolvimento de outros operadores que possam melhorá-lo, já que o AAP permite combinações (hibridismo) e/ou melhorias em seus módulos. Outra sugestão é o acoplamento do AAP a outros Algoritmos Genéticos, além do AG Canônico. Por ser uma concepção modular pode ser facilmente acoplado em outros AGs.

Referências Bibliográficas

(Beasley, 1990) BEASLEY, J. E. OR-Library: Distributing Test Problems by Electronic Mail.*Journal of the Operational Research Society*, v. 41, 1069-1072, 1990.

(Chainate et al., 2007) CHAINATE, W.; THAPATSUWAN, P.; PONGCHAROEN, P. A new heuristic for improving the performance of genetic algorithm. *World Academy of Science, Engineering and Technology*, v. 21, n.1, 217-220, 2007.

(Chang et al., 2008) CHANG, P. C.; CHEN, S. H.; FAN, C. Y. Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems. *Applied Soft Computing*, v. 8, n.1, 767-777, 2008.

(Chun e Beasley, 1998) CHUN, P. C.; BEASLEY, J. E. A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, v. 4, n.1, 63-86, 1998.

(Eiben e Schippers, 1998) EIBEN, A. E.; SCHIPPERS C. A. On evolutionary exploration and exploitation. *Fundamenta Informaticae*, n. 35, p. 35-50, 1998.

(Ferreira, 2001a) FERREIRA, C. Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems*, v.13, n.2, p. 87–129, 2001.

(Ferreira, 2001b) FERREIRA, C. Gene expression programming in problem solving. *In WSC6 tutorial*, 2001. Disponível em: <http:// www.propesq.ufpe.br/anais/anais.htm>. Acesso em: 17 jun. 2008.

(Fleszar et al., 2009) FLESZAR, K; HINDI, K. S. Fast, effective heuristics for the 0-1 multi-dimensional knapsack problem. *Computers & Operations Research*, v.36, n.5, p. 1602–1607, 2009.

(Gen e Cheng, 1997) GEN, M.; CHENG, R. *Genetic Algorithms and Engineering Design*. Wiley-Interscience, 1997.

(Goldberg, 1989) GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley, 1989.

(Hoff et al., 1996) HOFF, A.; LOKKETANGEN, A.; MITTET, I. Genetic Algorithms for 0/1 Multidimensional Knapsack Problems. *In Proceedings Norsk Informatikk Konferanse*. 1996.

(Khuri et al., 1994) KHURI, S.; BACK, T.; HEITKOTTER, J. The zero/one multiple knapsack problem and genetic algorithms. *In Proceedings of the 1994 ACM Symposium on Applied Computing*. ACM Press, p.188–193, 1994.

(Petersen, 1967) PETERSEN, C. C. Computational Experience with Variants of the Balas Algorithm Applied to the Slection of R&D Projects. *Management Science*, v.13, p. 736–750, 1967.

(Puchinger et al., 2006) PUCHINGER, J.; RAID, G. R.; PFERSCHY, U. The Core Concept for the Multidimensional Knapsack Problem. *In Evolutionary Computation in Combinatorial Optimization*. Springer, p.195–208, 2006.

(Mahfoud, 1992) S. MAHFOUD. Crowding and preselection revisited. *Parallel Problem Solving from Nature*, p. 27-37, 1992.

(Mahfoud, 1995) S. MAHFOUD. *Niching Methods for Genetic Algorithms*. Tese de Doutorado, Department of General Engineering, University of Illinois at Urbana-Champaign, 1995.

(Mathias e Whitley, 1992) MATHIAS, K.; WHITLEY, D. Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem. *In Parallel Problem Solving from Nature*, p. 219–228, 1992.

(Michalewicz, 1996) MICHALEWICZ, Z. *Genetic algorithms + data structures = evolution programs*. 3 ed. Berlin: Springer-Verlag, 1996.

(Mitchell e Forrest, 1994) MITCHELL, M.; FORREST, S. Genetic algorithms and artificial life. *Artificial Life*, vol. 1, p. 267–289, 1994.

(Musil et al., 1999) MUSIL, M.; WILMUT, M. J.; CHAPMAN, R. A hybrid simplex genetic algorithm for estimating geoacoustic parameters using matched-field inversion. *IEEE Journal of Oceanic Engineering*, vol. 24, n. 3, p. 358–369, 1999.

(Park et al., 2000) PARK, J.-B.; PARK, Y.-M.; WON, J.-R.; LEE, K. An improved genetic algorithm for generation expansion planning. *Power Systems*, vol. 15, p.916–922, 2000.

(Rajan et al., 2002) RAJAN, C.; MOHAN, M.; MANIVANNAN, K. Improved genetic algorithm solution to unit commitment problem. *In Transmission and Distribution Conference and Exhibition 2002: Asia Pacific*. IEEE/PES, vol. 1, p.255–260, 2002.

(Rodrigues, 2003) RODRIGUES, E. Programação genética na otimização de circuitos digitais. *In IV Encontro Nacional de Inteligência Artificial*. Campinas, SP, Brasil, 2003.

(Rong-Long e Kozo, 2005) RONG-LONG, W.; KOZO, O. Solving facility layout problem using an improved genetic algorithm. *Fundamentals of Electronics, Communications and Computer Sciences*. IEICE, v. E88-A, n. 2, p. 606–610, 2005.

(Ruttkay et al., 1995) RUTTKAY, Z.; EIBEN, A.; RAUE, P. Improving the performances of GAs on a GA-hard CSP. *In CP95 Workshop on Studying and Solving Really Hard Problems*. Cassis, França, p. 157–171, 1995.

(Shimodaira, 2002) SHIMODAIRA H. **An Empirical Performance Comparison of Niching Methods for Genetic Algorithms**. *IEICE Trans Inf Syst*. v. E85-D, n.11, p. 1872-1880, 2002.

(Singh e Deb, 2006) SINGH G.; DEB K. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. *In 8th annual conference on Genetic and evolutionary computation*. Seattle, USA, p. 1305-1312, 2006.

(Spears, 1992) SPEARS, W. M. Crossover or Mutation? *In Foundations of Genetic Algorithms Workshop*. p. 221-237, 1992.

(Sultan et al., 2007) SULTAN, A. B. M.; MAHMUD, R.; SULAIMAN M. S.. Reducing Premature Convergence Problem through Numbers Structuring in Genetic Algorithm. *International Journal of Computer Science and Network Security*, vol. 7, n. 4, p. 215–217, 2007.

(Tackett, W.A. e Carmi, A., 1994) TACKETT, W.A.; CARMI, A. The unique implications of brood selection for genetic programming. *In IEEE World Congress on Computational Intelligence*. Orlando, FL, USA, vol. 1, p. 160-165, 1994.

(Whitley, 1994) WHITLEY, D. A genetic algorithm tutorial. *Statistics and Computing*, vol. 4, p. 65–85, 1994.

(Wu et al., 2004) WU, A. S.; YU, H.; JIN, S.; LIN, K.-C.; SCHIAVONE, G. An incremental genetic algorithm approach to multiprocessor scheduling. *Parallel and Distributed Systems*, vol. 15, n. 9, p. 824–834, 2004.

(Yang e Douglas, 1998) YANG, R.; DOUGLAS, I. Simple genetic algorithm with local tuning: Eficient global optimizing technique. *J. Optim. Theory Appl.*, v. 98, n. 2, p. 449–465, 1998.

(Yen et al., 1998) YEN, J.; LIAO, J. C.; LEE, B.; RANDOLPH, D. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Trans. on Syst., Man, and Cybern.*, v. 28, n.2, p. 173–191, 1998.