NOVO OPERADOR GENÉTICO EM UM FLUXO PARALELO DE EXECUÇÃO PARA MELHORAR O DESEMPENHO DO AG

Celso Camilo*, Keiji Yamanaka†

* *Universidade Federal da Grande Dourados*
Faculdade de Ciências Exatas e Tecnologia
Dourados, MS, Brazil

† *Universidade Federal de Uberlândia*
Faculdade de Engenharia Elétrica
Uberlandia, MG, Brazil

Emails: celsocamilo@gmail.com, keiji@ufu.br

Abstract— Genetic Algorithms (AG) are algorithms used in several applications to optimize solutions, however, the process of solution improvement is, usually, slow and expensive, because involves some genetic operators. Depending of the complexity on the problem, to obtain one good solution is necessary to process days, weeks or months. Thus, this work developed a genetic operator, called "Cruzamento Assistido", that executed in parallel of the selection, mutation and crossing operators build the proposed AG. This, try to improve optimization capacity of the canonical AG. Two groups of experiments were done and the results, compared to canonical AG, show the good performance of proposed AG.

Keywords— Genetic Algorithms, Crossing, Optimization

Resumo— AG são algoritmos usadas em várias aplicações com o intuito de otimizar, no entanto, o processo de melhoramento da solução é, normalmente, lento e custoso computacionalmente, pois envolve alguns operadores genéticos. Dependendo da complexidade do problema, uma solução considerada boa pode demorar dias, semanas ou meses. Por isso, neste trabalho desenvolveu-se um operador genético, chamado Cruzamento Assistido, que executado em um fluxo paralelo aos operadores de seleção, mutação e cruzamento formam o AG proposto, este, objetiva agilizar o melhoramento genético. Aplicou-se 2 grupos de experimentos e os resultados, comparados ao AG canônico, demonstram o bom desempenho do AG proposto.

Palavras-chave— Algoritmo Genético, Cruzamento, Otimização

1 Introdução

Algoritmos Genéticos são métodos de otimização e busca inspirados nos mecanismos de evolução de população de seres vivos (Goldberg, 1989). Os algoritmos baseados nesta técnica seguem o princípio da seleção natural e sobrevivência do mais apto (Charles Darwin).

Uma das vantagens de um algoritmo genético é a simplificação que eles permitem na formulação e solução de problemas de otimização. AG's simples normalmente trabalham com descrições de entrada formadas por cadeias de bits de tamanho fixo. Outros tipos de AG's podem trabalhar com cadeias de bits de tamanho variável, como por exemplo AG's usados para Programação Genética (Rodrigues, 2003; Ferreira, 2001a; Ferreira, 2001b). O AG é indicado para a solução de problemas de otimização complexos, NP-Completos, como o "caixeiro viajante", que envolvem um grande número de variáveis e, conseqüentemente, espaços de soluções de dimensões elevadas. Além disso, em muitos casos onde outras estratégias de otimização falham na busca de uma solução, os AG's normalmente convergem. No entanto, em alguns casos o desempenho, relação tempo/qualidade da solução, não é satisfatório (Gen and Cheng, 2001), por isso, trabalhos (Park et al., 2000; Rong-Long and Kozo, 2005; Rajan et al., 2002; Wu et al., 2004; Ruttkay et al., 1995; Yen et al., 1998; Yang and Douglas, 1998; Musil et al., 1999; Chainate et al., 2007) são desenvolvidos com o intuito de melhorar o desempenho dos AG's.

2 AG canônico

Uma implementação (Whitley, 1994) de um algoritmo genético começa com uma população formada por cromossomos gerados, tipicamente, 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 (*fitness*) que é tipicamente definido com relação à população corrente.

A execução do AG pode ser divida em duas fases. Na primeira e tendo como partida uma população inicial, a seleção é aplicada 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) é referenciada esta implementação básica como *Simple Genetic Algorithm (SGA)*, chamada também neste trabalho de AG canônico. O fluxo de execução

do AG pode ser observado na Figura 1. Nele é possível identificar todas as principais etapas e a arquitetura cíclica do AG.

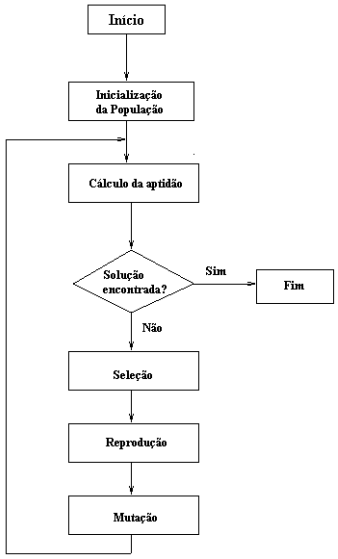

Figura 1: Fluxo de execução do AG canônico.

3 Cruzamento Assistido

Com o objetivo de agilizar o processo de melhoramento genético, este trabalho propõe o uso de um operador genético chamado Cruzamento Assistido. Neste operador os N melhores indivíduos da população corrente serão cruzados com o melhor indivíduo, da mesma população, gerando indivíduos que serão analisados.

O nome assistido é dado pela característica de somente inserir indivíduos na população após analisar a aptidão e detectar que se trata de um indivíduo mais apto que o melhor indivíduo. Caso o indivíduo não seja mais apto, ele é descartado. Este processo de análise do cruzamento feito dá ao operador uma função de tentar melhorar geneticamente a população corrente através da manipulação dos seus melhores indivíduos.

O operador será inserido em um fluxo paralelo ao fluxo dos operadores do SGA, formando o AG proposto (Figura 2). O operador receberá a mesma população que a etapa de seleção e poderá produzir indivíduos melhores que o melhor indivíduo existente. Caso o operador tenha sucesso e o AG esteja usando eletismo, os melhores indivíduos gerados serão inseridos na população no lugar dos eleitos, começando pelo de menor aptidão. Caso não esteja usando eletismo os indivíduos gerados poderão sobrepor qualquer indivíduo da população.

3.1 O algoritmo do Cruzamento Assistido

Após receber a população corrente, o operador executa as seguintes tarefas:

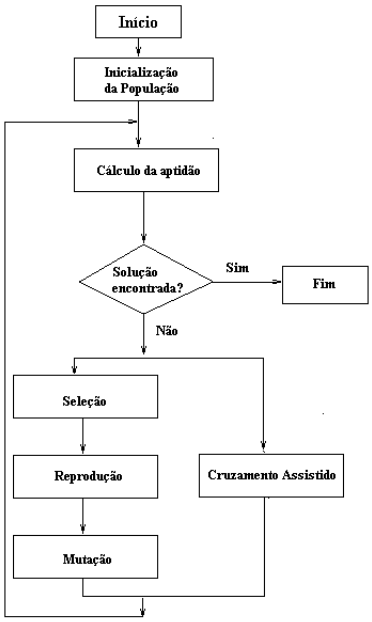

Figura 2: Fluxo de execução do AG proposto com o operador Cruzamento Assistido

1. Ordena, em ordem decrescente, a população, segundo a aptidão. Sendo assim, o primeiro indivíduo, o mais apto, será sempre o Pai. O que faz com que todos os filhos gerados sejam irmãos;

2. Recebe como parâmetro a quantidade de indivíduos (qtdIndiv), que serão manipulados no processo de melhoramento genético, e os indivíduos que poderão ser substituídos;

3. É feito um cruzamento entre o Pai (melhor indivíduo) e os estipulados na variável qtdIndiv. O cruzamento será detalhado na seção 3.1.1;

4. Caso haja bons indivíduos gerados, eles serão inseridos na nova população no lugar dos indivíduos eleitos, escolhidos pelo processo de eletismo. Caso não tenha eletismo, os indivíduos serão sobrepostos aos determinados no ítem 2.

A tarefa 4, acima, explica a forma de interferência do Cruzamento Assistido na população gerada pelo fluxo canônico. A interferência só acontece quando bons indivíduos, melhores que o melhor existente, forem gerados pelo operador proposto.

3.1.1 O cruzamento

Antes do cruzamento (troca de material genético) o cromossomo é divido em grupos de genes ou em gene a gene, a isto chama-se de divisão do material genético a ser trocado. Esta é uma etapa importante do algoritmo, pois acredita-se que o conhecimento do problema ajuda a escolher de que

forma será divido o cromossomo. Uma das opções é que o cromossomo seja divido segundo as variáveis codificadas, por exemplo, se os 3 primeiros genes representam a variável x e os 2 últimos a y, então, o cromossomo será divido em dois grupos: um contendo 3 genes e outro 2 (Figura 3). Dividindo assim, no momento do cruzamento a troca será feita entre valores das variáveis (valores decodificados) e não entre valores codificados.

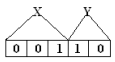

Figura 3: Dois grupos selecionados como material genético de troca.

Estabelecido os grupos de genes (divisão do material genético), o passo seguinte é o cruzamento. No cruzamento o cromossomo é representado por um vetor, sendo que, cada elemento deste é um grupo de genes, estabelecido pela divisão do material genético. O melhor indivíduo é reservado como pai e as mães são os melhores indivíduos restantes, sendo que, 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 outros elementos do seu vetor, completa o cromossomo do filho. Este processo repete para todas as mães, gerando um grupo de filhos. Outros grupos de filhos são gerados pela troca do elemento do vetor doado pela mãe e complemento do cromossomo pelo pai, assim, a quantidade de grupos de filhos é igual a quantidade de elementos do vetor, que por sua vez, é igual a 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 e recomeça todo o processo de cruzamento, com o melhor filho sendo o pai da iteração. Caso não, o algoritmo interrompe o laço de repetição e insere o pai corrente na população final. Como exemplo de uma iteração do Cruzamento Assistido, a Figura 4 mostra:

- O cromossomo dividido (divisão do material genético) em duas partes. Grupo 1 com 3 genes e o grupo 2 com 2 genes;
- Vetor, que representa o cromossomo, com 2 elementos. O primeiro elemento representando o grupo 1 de genes e o segundo elemento representando o grupo 2 de genes.
- Três mães, estabelecidas pelo parâmetro "qtdIndiv" = 3, sendo, as mães os 3 melhores indivíduos após o pai;
- O primeiro grupo de filhos, gerados pela doação dos primeiros elementos do vetor das mães e o complemento do pai. Como 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);
- O segundo grupo de filhos, neste a mãe doa o segundo elemento do vetor e o pai completa, com o seu primeiro elemento do vetor. Como 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, primeiro elemento do vetor, do pai (001). Neste 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.

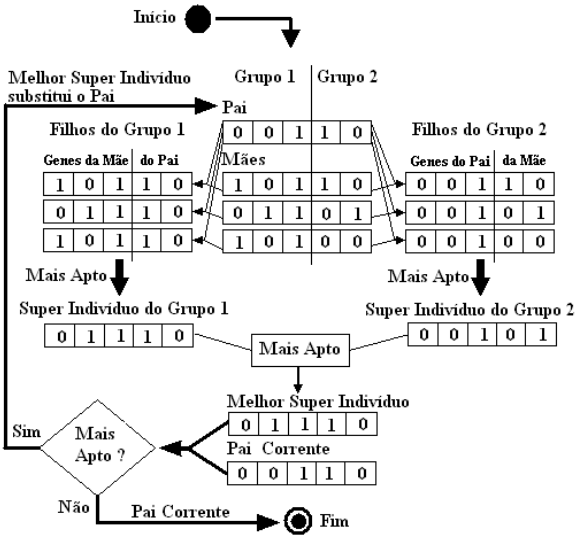

Figura 4: Exemplo de um Cruzamento Assistido.

4 Experimentos

Para testar o desempenho do operador proposto, usaremos um problema de maximização de função. A função escolhida (equação 1) tem duas variáveis (x, y) e uma superfície com vários picos e vales, caracterizando vários ótimos locais. A Figura 5 ilustra uma projeção suavizada desta superfície.

$$f(x,y) = 21.5 + (x * \sin(4\pi * x)) + (y * \sin(20\pi * y)) \quad (1)$$

Dada a função, o objetivo é maximizar respeitando as restrições: $-3.10 <= x <= 12.10$, $4.10 <= y <= 5.80$ e valores com duas casas decimais. Considerando as restrições, o ponto máximo da função é atingido quando $x = 11.63$

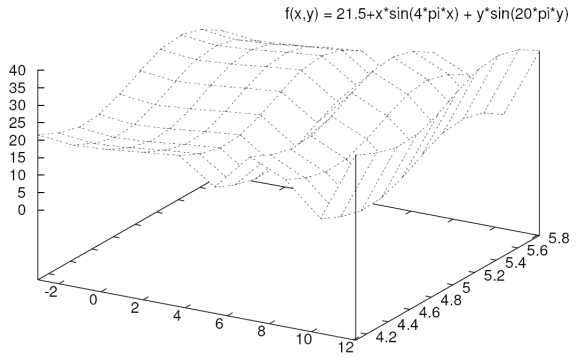

Figura 5: Superfície da função $f(x, y)$.

e $y = 5.73$. Neste ponto a solução é ótima, gerando um $f(x, y) = 38.8096858858971$.

Foi modelado e implementado o problema em ambos os AG's, canônico e proposto, o que propicia uma análise comparativa dos algoritmos.

Para o AG canônico foram estabelecidas as seguintes configurações: 30 indivíduos na população; população inicial aleatória; probabilidade de cruzamento de 0.6, sendo, o cruzamento com 1 ponto de corte, aleatório, e a cada cruzamento gera-se dois filhos; probabilidade de mutação de 0.2, sendo, a mutação feita em 1 ponto, aleatório, pela inversão do valor do bit; seleção por roleta; cromossomo com codificação binária e dividido em 4 partes, sendo, a primeira parte, com 4 bits, representando a parte inteira do x; a segunda parte, com 7 bits, representando a parte decimal do x; a terceira parte, com 3 bits, representando a parte inteira do y e a quarta parte, com 7 bits, representando a parte decimal do y; eletismo de um indivíduo.

Para o AG proposto foram estabelecidas as seguintes configurações: 30 indivíduos na população; população inicial aleatória; probabilidade de cruzamento de 0.6, sendo, o cruzamento com 1 ponto de corte, aleatório, e a cada cruzamento gera-se dois filhos; probabilidade de mutação de 0.2, sendo, a mutação feita em 1 ponto, aleatório, pela inversão do valor do bit; seleção por roleta; cromossomo com codificação binária e dividido em 4 partes, sendo, a primeira parte, com 4 bits, representando a parte inteira do x; a segunda parte, com 7 bits, representando a parte decimal do x; a terceira parte, com 3 bits, representando a parte inteira do y e a quarta parte, com 7 bits, representando a parte decimal do y; eletismo de um indivíduo; Cruzamento Assistido com divisão do material genético em 4 partes, assim, a cada troca de material genético os pais passam características (parte inteira e/ou decimal) das variáveis x e/ou y e o parâmetro "qtdIndiv" = 15, assim, os 15 melhores indivíduos são considerados na formação do Super Indivíduo.

Os experimentos são divididos em 2 grupos, sendo que, no primeiro o critério de parada do AG é atingir 10 gerações e no segundo 30 gerações. Cada grupo é composto de 10 experimentos, sendo que, para cada experimento foi executado independentemente o algoritmo canônico e o proposto. A população inicial de ambos, canônico e proposto, é a mesma para cada experimento, sendo assim, pode-se fazer uma análise comparativa da evolução gerada.

Durante os experimentos serão analisados: a avaliação (função objetivo) do melhor indivíduo (MAF) após 10 gerações, grupo 1 (ver 4.1), e após 30 gerações, grupo 2 (ver 4.2); a porcentagem de acréscimo na avaliação (% Acréscimo), determinando a proporção de crescimento referente a população inicial.

Para facilitar a diagramação das tabelas, alguns nomes são representados por siglas: Ex = Experimento; MAI = maior avaliação (melhor indivíduo) da População Inicial; AG/C = Algoritmo Genético Canônico; AG/P = Algoritmo Genético Proposto; Md = Média.

4.1 Resultado do grupo 1 de experimentos

Observa-se, como descrito na seção acima, que cada experimento deste grupo é finalizado após 10 gerações do AG/C e outras 10 do AG/P. A Tabela 1 demonstra os resultados obtidos (MAF, % Acréscimo) e a média após 10 experimentos.

Tabela 1: Resultado do grupo 1 de experimentos.

Ex	MAI	MAF		% Acréscimo	
		AG/C	AG/P	AG/C	AG/P
1	34,25	34,25	37,70	0,00	10,07
2	30,63	30,63	38,40	0,00	25,37
3	30,68	30,68	37,53	0,00	22,33
4	36,40	36,40	38,69	0,00	6,29
5	31,49	31,49	38,60	0,00	22,58
6	33,04	33,04	37,64	0,00	13,92
7	32,23	36,63	37,50	13,65	16,35
8	33,20	33,20	37,85	0,00	14,01
9	29,95	33,30	37,88	11,19	26,48
10	35,59	35,59	38,39	0,00	7,87
Md		33,52	38,02	2,48	16,53

4.2 Resultado do grupo de experimentos 2

Cada experimento deste grupo é finalizado após 30 gerações do AG/C e outras 30 do AG/P. A Tabela 2 demonstra os resultados obtidos (MAF, % Acréscimo) e a média após 10 experimentos.

4.3 O tempo de execução

Todo acréscimo de funcionalidade acarreta em aumento de tempo na execução, no entanto, é necessário analisar se o custo (Tempo) compensa o

Tabela 2: Resultado do grupo 2 de experimentos.

Ex	MAI	MAF		% Acréscimo	
		AG/C	AG/P	AG/C	AG/P
1	29,95	32,76	38,29	9,38	27,85
2	34,17	34,17	38,59	0,00	12,94
3	35,13	35,13	38,20	0,00	8,74
4	31,49	35,48	38,70	12,67	22,90
5	28,03	31,91	36,90	13,84	31,64
6	29,66	31,14	38,80	4,99	30,82
7	29,39	29,74	38,74	1,19	31,81
8	30,55	30,55	37,00	0,00	21,11
9	34,37	35,46	37,44	3,17	8,93
10	35,16	35,16	38,69	0,00	10,04
Md		33,15	38,14	4,52	20,68

benefício da função. As seções 4.1 e 4.2 mostraram os resultados quanto aos benefícios, ou seja, capacidade de melhorar a solução a cada geração. Nesta seção serão mostrados os tempos gastos, em novas execuções dos AG's, seguindo a mesma estrutura de experimentos definida para as seções anteriores.

A Tabela 3 mostra os tempos, em segundos, obtidos da execução de 10 experimentos, sendo, cada experimento interrompido após 10 gerações do AG canônico e outras 10 do AG proposto. A Tabela 4 mostra os tempos, em segundos, obtidos da execução de 10 experimentos, sendo, cada experimento interrompido após 30 gerações do AG canônico e outras 30 do AG proposto. Os experimentos foram executados em um PC com o processador AMD Turion 1.6 GHz e memória de 1 GB.

Tabela 3: Tempos de execuções dos AG's, após 10 gerações.

Ex	Tempo gasto(s)		Tempo acrescido(s)
	AG/C	AG/P	AG/P - AG/C
1	0,39	0,42	0,03
2	0,37	0,43	0,06
3	0,38	0,42	0,04
4	0,38	0,44	0,06
5	0,39	0,44	0,05
6	0,37	0,42	0,05
7	0,39	0,46	0,07
8	0,39	0,43	0,04
9	0,37	0,42	0,05
10	0,39	0,42	0,03
Md	0,382	0,43	0,048

4.4 Resultados gerais

O objetivo dos grupos de experimentos é medir o desempenho, capacidade de acelerar o melhoramento genético da solução, do algoritmo proposto.

Tabela 4: Tempos de execuções dos AG's, após 30 gerações.

Ex	Tempo gasto(s)		Tempo acrescido(s)
	AG/C	AG/P	AG/P - AG/C
1	1,23	1,39	0,16
2	1,22	1,41	0,19
3	1,23	1,41	0,18
4	1,22	1,42	0,2
5	1,24	1,4	0,16
6	1,24	1,39	0,15
7	1,25	1,41	0,16
8	1,24	1,4	0,16
9	1,22	1,39	0,17
10	1,21	1,36	0,15
Md	1,23	1,398	0,168

Por isso, criou-se dois pontos de aferição do desempenho, sendo, o primeiro após 10 gerações (grupo 1) e o segundo após 30 gerações (grupo 2). Com os resultados do primeiro ponto é possível analisar o desempenho do algoritmo proposto para os momentos iniciais de execução, já no segundo ponto, é possível analisar um período de execução maior.

Analisando a Tabela 1 identifica-se uma diferença de 14,05% entre a média da % Acréscimo do algoritmo proposto (16,53%) e do canônico (2,48%), caracterizando um desempenho superior do algoritmo proposto após 10 gerações. Esta diferença de desempenho, identificada na média, está presente em todos os experimentos, sendo que, a maior diferença foi de 25,37% (0% do canônico e 25,37% do proposto) no experimento 2. Observa-se, também, que o algoritmo canônico melhorou a solução em apenas dois experimentos (7 e 9), o que demonstra pouca participação, já o proposto, melhorou a população em todos os experimentos.

Analisando a Tabela 2 identifica-se uma diferença de 16,16% entre a média da % Acréscimo do algoritmo proposto (20,68%) e do canônico (4,52%), caracterizando um desempenho superior do algoritmo proposto após 30 gerações. Esta diferença de desempenho, identificada na média, está presente em todos os experimentos, sendo que, a maior diferença foi de 30,62% (1,19% do canônico e 31,81% do proposto) no experimento 7. Observa-se, também, que o algoritmo canônico melhorou a solução em seis experimentos (1,4,5,6,7 e 9), já o proposto, melhorou a população em todos os experimentos.

As Tabelas 1 e 2 demonstram uma melhora significativa da solução final, dada a aplicação do algoritmo proposto, e uma dificuldade do algoritmo canônico em contribuir para o melhoramento, principalmente durante as 10 primeiras gerações (Tabela 1). Apesar do algoritmo canônico ter crescido, entre o grupo 1 e 2, na participação de melhoria da solução, foi bem inferior ao algoritmo proposto, que teve participação em todos os

experimentos.

A seção 4 descreve o problema e determina o ponto ótimo (MAF) de 38,80. A proximidade do ótimo pelo AG proposto, em média 38,02 (grupo 1) e 38,14 (grupo 2), é outro fator que determina um melhor desempenho, em relação ao AG canônico, que obteve em média MAF de 33,52 (grupo 1) e 33,15 (grupo 2). Analisando os experimentos individualmente, todos MAF's do AG proposto, exceto do experimento 5 no grupo 2 (36,90), foram maior ou igual a 37,00, sendo que no experimento 6 do grupo 2 o ótimo foi alcançado. Já no AG canônico, o maior MAF foi de 36,40. Assim, o menor MAF do AG proposto (36,90) é maior do que o maior MAF (36,40) do AG canônico.

A seção 4.3 apresenta os tempos, em segundos, da execução dos AG's analisados. Como esperado, o AG proposto aumentou o tempo de execução, no entanto, este acréscimo é muito pequeno. Para o primeiro grupo de experimentos (Tabela tabResTempo1) o acréscimo é, em média, de 0,048 segundos, já no segundo grupo (Tabela tabResTempo2), o acréscimo é, em média, de 0,168 segundos. Assim, a relação custo/benefício é favorável para a aplicação do AG proposto, pois, os benefícios compensam o pequeno custo de tempo.

5 Conclusão

Tendo como base o AG canônico, o AG proposto mostrou-se eficiente e eficaz, cumprindo assim, o objetivo de agilizar o melhoramento da solução. Os resultados apresentados na seção 4.4 demonstram o ganho obtido com a inserção do operador genético Cruzamento Assistido, já a seção 4.3, relata o baixo custo (tempo acrescido) da solução.

Apesar dos significativos resultados, o AG proposto foi testado somente no cenário descrito acima, por isso, tem uma limitação de aplicação. Para minimizar esta limitação, está em estudo a aplicação do mesmo no problema do caixeiro viajante.

Como trabalhos futuros sugere-se o uso do AG proposto em outras aplicações e um estudo das diferentes configurações para os parâmetros inseridos, "qtdIndiv" e divisão do material genético.

Referências

Chainate, W., Thapatsuwan, P. and Pongcharoen, P. (2007). A new heuristic for improving the performance of genetic algorithm, *Transactions on Engineering, Computing and Technology* **19**.

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

Ferreira, C. (2001b). Gene expression programming in problem solving, *WSC6 tutorial*.

Gen, M. and Cheng, R. (2001). *Genetic Algorithms and Engineering Design*, 1 edn, Wiley-Interscience.

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

Musil, M., Wilmut, M. J. and Chapman, R. (1999). A hybrid simplex genetic algorithm for estimating geoacoustic parameters using matched-field inversion, *IEEE J. Oceanic Eng.* (24): 358–369.

Park, J.-B., Park, Y.-M., Won, J.-R. and Lee, K. (2000). An improved genetic algorithm for generation expansion planning, *Power Systems, IEEE Transactions on* **15**: 916–922.

Rajan, C., Mohan, M. and Manivannan, K. (2002). Improved genetic algorithm solution to unit commitment problem, *Transmission and Distribution Conference and Exhibition 2002: Asia Pacific. IEEE/PES* **1**: 255–260.

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

Rong-Long, W. and Kozo, O. (2005). Solving facility layout problem using an improved genetic algorithm, *Fundamentals of Electronics, Communications and Computer Sciences, IEICE Transactions on* **E88-A**: 606–610.

Ruttkay, Z., Eiben, A. and Raue, P. (1995). Improving the performances of gas on a ga-hard csp, *CP95 Workshop on Studying and Solving Really Hard Problems*, pp. 157–171.

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

Wu, A. S., Yu, H., Jin, S., Lin, K.-C. and Schiavone, G. (2004). An incremental genetic algorithm approach to multiprocessor scheduling, *Parallel and Distributed Systems, IEEE Transactions on* **15**(9): 824–834.

Yang, R. and Douglas, I. (1998). Simple genetic algorithm with local tuning: Efficient global optimizing technique, *J. Optim. Theory Appl.* (28): 449–465.

Yen, J., Liao, J. C., Lee, B. and Randolph, D. (1998). A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method, *IEEE Trans. on Syst., Man, and Cybern.* **B**(28): 173–191.