Uma Nova Versão Aprimorada do Método da Otimização Extrema Generalizada

Roberto Luiz Galski, Fernando Manuel Ramos, Fabiano Luís de Sousa

Instituto Nacional de Pesquisas Espaciais (INPE)

galski@ccs.inpe.br, fernando@lac.inpe.br, fabiano@dem.inpe.br

Resumo

Este artigo apresenta uma nova implementação aprimorada do algoritmo Otimização Extrema Generalizada (Generalized Extremal Optimization - GEO), e sua variante GEO_{var}, e analisa o ganho obtido quando aplicado a um conjunto de funções teste. Em Galski et al., 2003, uma primeira versão aprimorada foi apresentada. O novo aprimoramento, aplicado cumulativamente com o anterior, é mais um dos resultados dos estudos efetuados pelo primeiro autor como parte do seu programa de doutorado. São apresentadas curvas do valor da função objetivo versus o número de avaliações da função tanto para o GEO e GEO_{var} do primeiro aprimoramento quanto para o segundo, sendo os resultados analisados.

***Palavras-chave**: otimização global, algoritmos evolutivos.*

1. Introdução

O algoritmo GEO e sua variante GEO_{var}, são meta-heurísticas de otimização global (Sousa & Ramos, 2002; Sousa et al., 2003a, 2003b) desenvolvidas como uma proposta de generalização do algoritmo τ-EO (Boettcher & Percus, 2001). Assim como os métodos Recozimento Simulado - RS (Kirkpatrick et al., 1983) e os Algoritmos Genéticos - AG (Goldberg, 1989), GEO e GEO_{var} são algoritmos estocásticos cuja inspiração origina-se na analogia de processos observados na natureza.

GEO e GEO_{var} destinam-se, primeiramente, a resolver problemas de otimização complexos, onde pouco ou nada se conhece a respeito do espaço de busca viável e inviável da função a ser otimizada. Podem ser aplicados a praticamente qualquer problema de otimização contínua ou combinatória, não convexo ou disjunto, com ou sem restrições, podendo, inclusive, ser aplicados a problemas com miscelânea de variáveis reais, inteiras ou discretas. Ambos são de fácil implementação e usam apenas valores da função objetivo. Não requerem o fornecimento de um ponto de partida situado no espaço viável. Além disso, possuem apenas um parâmetro de ajuste, τ, o que facilita bastante o ajuste de parâmetros destes dois algoritmos.

Em Galski et al., 2003, uma primeira versão aprimorada de GEO e GEO_{var} foi apresentada. Neste artigo, aquela versão será usada como ponto de partida para um novo aprimoramento. Para evitar confusões, a versão aprimorada apresentada em Galski et al., 2003 será designada por GEO_1 e GEO_{var1} e a nova versão ora apresentada será denominada GEO_2 e GEO_{var2}. O desempenho desse novo aprimoramento é avaliado através de um conjunto de funções teste não lineares. Estas funções foram utilizadas por Potter & De Jong, 1994, para avaliação de AG's. Posteriormente, em Sousa & Ramos, 2002, as mesmas funções foram utilizadas para avaliação comparativa do desempenho do GEO e GEO_{var} originais com os AG's de Potter & De Jong, 1994. As mesmas funções foram utilizadas em Galski et al., 2003, para comparação direta entre o GEO e GEO_{var} originais e GEO_1 e GEO_{var1}.

2. Versão Aprimorada GEO_2 e GEO_{var2}

O segundo aprimoramento sugerido tem em vista o algoritmo GEO especificamente e introduz neste algoritmo a capacidade de auto-reinicialização da configuração corrente durante a busca. Esta funcionalidade permite ao algoritmo monitorar, ao longo da busca, o decaimento do valor da função objetivo (consideram-se aqui problemas de minimização, sem perda de generalidade). Se em algum momento este decaimento estagnar, a auto-reinicialização entra em ação, sorteando um novo ponto no espaço das variáveis de projeto a partir do qual a busca continuará.

Hipoteticamente, ao longo de uma busca o GEO pode, depois de algum tempo, "estacionar", ficar preso no que pode ser chamado de "pontos de estagnação". Tais pontos de estagnação ocorrem quando o algoritmo já se encontra em um ponto do espaço de busca tal que, ao realizar o ordenamento dos bits, todos levam a "pontos piores", ou seja, pontos para os quais o valor da função objetivo é maior. Sob este aspecto, tal ponto pode ser considerado um ponto de mínimo local "relativo". O termo relativo é usado para expressar que, embora não seja necessariamente um mínimo da função objetivo, é um ponto de mínimo local do ponto de vista do GEO, pois de todos os pontos ao qual ele tem acesso numa dada iteração, aquele é o de menor valor da função objetivo. Como a cada iteração o GEO modifica um bit obrigatoriamente, isto faz com que o GEO saia do mínimo local relativo onde se encontra, indo para um dos "pontos piores". Se na iteração seguinte o ordenamento dos bits também levar a "pontos piores", a única exceção será o bit que foi mudado na iteração anterior. Este, ao ser modificado, leva de volta ao ponto de mínimo local relativo aonde o algoritmo estava antes. Sendo assim, é grande a probabilidade do algoritmo modificar novamente o mesmo bit, a fim de voltar para o mínimo local relativo. Se isto ocorrer, o algoritmo volta à exata condição que tinha quando chegou ao mínimo local relativo pela primeira vez, caracterizando assim, um circulo vicioso ou "circulo de estagnação", onde o algoritmo pode repetir indefinidamente os mesmos

movimentos no espaço de busca, perdendo eficiência. Os pontos de mínimo local relativos para os quais o fenômeno cíclico recém descrito ocorre definem os pontos de estagnação. A probabilidade de ocorrência dos "círculos de estagnação" depende do parâmetro τ. Quanto maior for τ, maior a probabilidade do GEO ficar preso em "círculos de estagnação", pois com τs elevados diminui a chance do algoritmo deixar de escolher o ponto que leva de volta ao ponto de estagnação, ou seja, ao mínimo local relativo. Além disso, a chance da estagnação acontecer no início da busca é menor, visto que nesse estágio, o espaço de busca ainda está começando a ser explorado e é provável que o algoritmo leve algum tempo antes de encontrar um mínimo local relativo.

É importante ressaltar que apesar de não haver correspondência direta entre mínimos locais reais da função objetivo e "mínimos locais relativos", é provável que pelo menos alguns "mínimos locais relativos" sejam mínimos locais da função objetivo, sendo que quanto mais próximo o mínimo local for do mínimo global, em termos do valor da função objetivo, maior a chance dele ser um ponto de estagnação. O mínimo global é o caso extremo que ajuda a visualizar a situação. Se, durante a busca, o algoritmo encontra-se no ponto de mínimo global da função objetivo, então, ao realizar o ordenamento dos bits, todos, obrigatoriamente, levarão à "pontos piores". Sob esse aspecto, o mínimo global pode ser visto como o "maior" dos pontos de estagnação. Para os mínimos locais cujo valor é próximo ao do mínimo global, a chance de um deles ser um ponto de estagnação também é grande, visto que devem ser pequenas as regiões do espaço de busca que possuem valores menores para a função objetivo. Para os mínimos locais cujos valores são distantes ao do mínimo local, a chance de um deles ser um ponto de estagnação é menor, visto que devem ser grandes as regiões do espaço de busca que possuem valores menores para a função objetivo.

Como GEO e GEO_1 (Galski et al., 2003) são idênticos do ponto de vista dinâmico, ou seja, ambos percorrem os mesmos pontos no espaço de busca, então a hipótese de ocorrência de pontos de estagnação é válida também para GEO_1. Sendo GEO_1 um aprimoramento de GEO, o mesmo foi usado como base para implementar a auto-reinicialização. Apresentam-se a seguir, os passos do algoritmo GEO_1 já com as alterações. Esta variante é denominada GEO_2. As alterações estão realçadas em negrito.

O algoritmo GEO_2 é composto dos seguintes passos:

1. Inicialize aleatoriamente uma seqüência binária C, de comprimento L que codifica N variáveis de projeto. Dentro de C, cada variável é codificada em uma parcela com comprimento l_j, $j \in \{1,2,...,N\}$ e tal que $\sum_j l_j = L$. Calcule o valor da função objetivo V para a configuração C e faça $C_{melhor} = C$ e $V_{melhor} = V$. **Faça E = 0.**
2. **Faça $V_{anterior} = V$.** Faça $V_{melhor}' = V_{melhor}$ e $C_{melhor}' = C_{melhor}$. Para cada bit $i \in \{1,2,...,L\}$ da seqüência C, faça:
   a) Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits C_i. Calcule o valor da função objetivo V_i, para a configuração C_i. Em seguida, se $V_i < V_{melhor}'$ então faça $V_{melhor}' = V_i$ e $C_{melhor}' = C_i$.
   b) Atribua ao bit i um índice de adaptação $\Delta V_i = (V_i - V_{melhor})$, que indica o ganho (ou perda) que se têm ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até o momento.
   c) Retorne o bit i ao seu valor original.
3. Ordene todos os bits $i \in \{1,2,...,L\}$ de acordo com os seus índices de adaptação, de k=1 para o menos adaptado à k=L, para o mais adaptado. Crie uma lista de valores (i,k) relacionando a posição física i do bit em C com seu respectivo número de ordem k, onde $i,k \in \{1,2,...,L\}$. Se $\Delta V_i = \Delta V_j$, $i \neq j$, estabeleça a ordem entre i e j de forma aleatória.
4. Escolha com igual probabilidade um bit candidato i para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme o bit i para ser modificado. Repetir este passo até que um bit seja confirmado. Quando isso acontecer, faça $i_{esc} = i$.
5. Faça $C = C_{iesc}$, calcule V_{iesc} e faça $V = V_{iesc}$.
6. **Se $V \geq V_{anterior}$ e $V_{melhor}' = V_{melhor}$ faça E = E+1 senão, faça E = 0.** Faça $V_{melhor} = V_{melhor}'$ e $C_{melhor} = C_{melhor}'$. **Se $E = E_{lim}$, $E_{lim} \in \{1,2,...\}$, faça E = 0 e reinicialize aleatoriamente a sequência binária C e calcule o respectivo valor de V. Se $V < V_{melhor}$ faça $V_{melhor} = V$.**
7. Repita os passos 2 a 6 até que um dado critério de parada seja satisfeito.
8. Retorne C_{melhor} e V_{melhor}.

No passo 6 do algoritmo acima, "E" conta o número de iterações consecutivas do algoritmo em que não houve melhora, nem no valor da função objetivo V atual com relação à $V_{anterior}$, nem no valor V_{melhor}' com relação à V_{melhor}. Se o valor de E for igual ao valor limite E_{lim}, previamente definido, então ocorre a reinicialização da seqüência binária C.

Conforme já mencionado, a versão 2 foi desenvolvida tendo em vista apenas o GEO/GEO_1. O algoritmo GEO_{var}/ GEO_{var1}, pelo fato de realizar a mutação simultânea de N bits ao invés de um, deve possuir uma espécie de "imunidade natural" ao problema da estagnação. De todo modo, movido pela curiosidade, decidiu-se implementar a versão 2 também para o GEO_{var1}, a fim de ver o que de fato acontece.

Os passos para o GEO_{var2} são:

1. Inicialize aleatoriamente uma seqüência binária C, de comprimento L que codifica N variáveis de projeto. Dentro de C, cada variável é codificada em uma parcela com comprimento l_j, $j \in \{1,2,...,N\}$ e tal que $\sum_j l_j = L$. Calcule o valor da função

objetivo V para a configuração C e faça C_{melhor}=C e V_{melhor}=V. **Faça E = 0.**

2. **Faça V_{ant} = V.** Faça $V_{melhor}' = V_{melhor}$ e $C_{melhor}' = C_{melhor}$. Para cada bit $i \in \{1,2,...,L\}$ da seqüência C, faça:
   a) Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits C_i. Calcule o valor da função objetivo V_i, para a configuração C_i. Em seguida, se $V_i < V_{melhor}'$ então faça $V_{melhor}' = V_i$ e $C_{melhor}' = C_i$.
   b) Atribua ao bit i um índice de adaptação $\Delta V_i = (V_i - V_{melhor})$, que indica o ganho (ou perda) obtido ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até o momento.
   c) Retorne o bit i ao seu valor original.
3. Para cada variável $j \in \{1,2,...,N\}$ faça:
   a) Ordene os bits $i \in \{1+s,2+s,...,l_j+s\}$, $s \equiv \sum_{m=0}^{m=j-1} l_m$ e $l_0 \equiv 0$, ou seja, ordene os bits da variável j, de acordo com os seus índices de adaptação ΔV_i , de k=1 para o menor ΔV_i até k= l_j, para o maior ΔV_i, onde l_j é o número de bits da variável j.
   b) Crie uma lista $(i,k)_j$, relacionando a posição física i do bit em C com seu número de ordem k. Se $\Delta V_m = \Delta V_n$, $\forall m \neq n$, estabeleça a ordem entre m e n de forma aleatória.
4. Para cada variável $j \in \{1,2,...,N\}$ faça:
   a) Escolha com igual probabilidade um bit candidato $i \in \{1+s,2+s,...,l_j+s\}$, $s \equiv \sum_{m=0}^{m=j-1} l_m$ para ser modificado.
   b) Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual a $P_i(k)$, confirme o bit para mutar.
   c) Repetir os passos 4.a) e 4.b) até que um bit seja confirmado para ser modificado. Quando isso acontecer, faça $i_{esc_j} = i$.
5. Faça C = C_{esc}, onde C_{esc} é a configuração resultante da mutação em C dos N bits escolhidos em 4.c (bits $i_{esc}j$, $j \in \{1,2,...,N\}$) . Calcule V_{esc} e faça V = V_{esc}. Se V < V_{melhor}' então faça $V_{melhor}' = V$ e $C_{melhor}' = C$.
6. **Se $V \geq V_{ant}$ e $V_{melhor}' = V_{melhor}$ faça E=E+1.** Faça $V_{melhor} = V_{melhor}'$ e $C_{melhor} = C_{melhor}'$. **Se $E > E_{lim}$ faça E = 0, reinicialize aleatoriamente a seqüência binária C e calcule o respectivo valor de V. Se $V < V_{melhor}$ faça $V_{melhor} = V$.**
7. Repita os passos 2 a 6 até que um dado critério de parada seja satisfeito.
8. Retorne C_{melhor} e V_{melhor}.

3. Resultados com as Funções Teste

As versões GEO_2 e GEO_{var2} foram aplicadas ao conjunto de funções teste FT_i, $i \in \{1,2,...,5\}$ conforme Tabela a seguir:

Tabela 1 – Funções teste

	Min. F(**X**)
FT_1 (Rosenbrock)	$F(\mathbf{X}) = 100\,(X_1^2 - X_2)^2 + (1 - X_1)^2$ $-2{,}048 \leq X_{i\in(1,..,N\}} \leq 2{,}048$ N=2 $\mathbf{X}^* = [\,1\;\;1\,]$ $F(\mathbf{X}^*) = 0$
FT_2 (Rastringin)	$F(\mathbf{X}) = 3\,N + \sum_{i=1}^{N}\left(X_i^2 - 3\cos(2\pi X_i)\right)$ $-5{,}12 \leq X_{i\in(1,..,N\}} \leq 5{,}12$ N=20 $X_i^* = 0$ $F(\mathbf{X}^*) = 0$
FT_3 (Schwefel)	$F(\mathbf{X}) = 418{,}9829\,N - \sum_{i=1}^{N} X_i \sin\left(\sqrt{\lvert X_i\rvert}\right)$ $-500 \leq X_{i\in(1,..,N\}} \leq 500$ N=10 $X_i^* = 420{,}9687$ $F(\mathbf{X}^*) = 0$
FT_4 (Griewangk)	$F(\mathbf{X}) = 1 + \sum_{i=1}^{N}\frac{X_i^2}{4000} - \prod_{i=1}^{N}\cos\left(\frac{X_i}{\sqrt{i}}\right)$ $-600 \leq X_{i\in(1,..,N\}} \leq 600$ N=10 $X_i^* = 0$ $F(\mathbf{X}^*) = 0$
FT_5 (Ackley)	$F(\mathbf{X}) = 20 + e - 20\,e^{\left(-0{,}2\sqrt{\frac{1}{N}\sum_{i=1}^{N}X_i^2}\right)} - e^{\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi X_i)\right)}$ $-30 \leq X_{i\in(1,..,N\}} \leq 30$ N=30 $X_i^* = 0$ $F(\mathbf{X}^*) = 0$

[1] N = Número de dimensões (variáveis); [2] $\mathbf{X}^*$=Ponto de mínimo global.

Inicialmente, é feita uma busca do tipo varredura a fim de verificar a influência dos parâmetros τ e E_{lim} nas versões GEO_2 e GEO_{var2}. Para τ, a varredura deu-se no intervalo de 0 a 10 com passo 0,25. Para E_{lim}, utilizou-se $E_{lim} \in \{1;2;3;4;20;100;1000\}$. É importante notar que quando $E_{lim} \to \infty$, GEO_2 e $GEO_{var2} \to GEO_1$ e GEO_{var1}, ou seja, a versão 2 torna-se idêntica a versão 1 a partir da qual foi desenvolvida. Para a maioria das implementações práticas, isto ocorre para valores finitos de E_{lim}. De fato, para os testes efetuados, o valor E_{lim}=1000 garante que nenhuma reinicialização seja aplicada, e isto para todas as FT's. Com isto, a menos do desempenho temporal, as buscas efetuadas pela versão 2 com E_{lim}=1000 equivalem a buscas efetuadas pela versão 1. O motivo de utilizar-se E_{lim}=1000 é exatamente o de ter resultados equivalentes ao da versão 1 para utilizar como referência na comparação com a versão 2.

As figuras 1 até 5 apresentam os resultados médios de 50 execuções independentes, em termos do melhor F(**X**) encontrado ao final de cada execução, obtidos com as versões GEO_2 e GEO_{var2}. Nas figuras, cada curva apresenta o resultado para um dado valor de E_{lim}, conforme indicado, e para τ variando de 0 a 10. Visando maior clareza, para $E_{lim} \in \{1,2,3,4\}$ apenas duas das quatro curvas obtidas são apresentadas. O critério de parada usado foi interromper o algoritmo após 10^4 avaliações da função objetivo para FT_1 e após 10^5 avaliações para FT_i, $i \in \{2,3,4,5\}$. Para todas as FT's, 16 bits foram usados para codificar cada uma das variáveis.

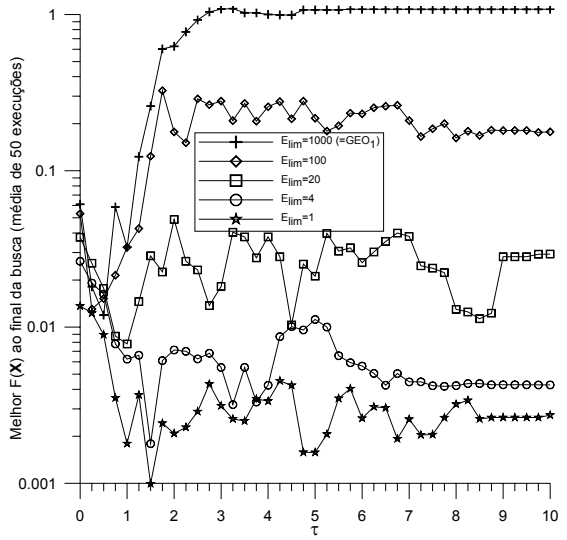

Figura 1a - FT_1 x τ x E_{lim} para GEO_2

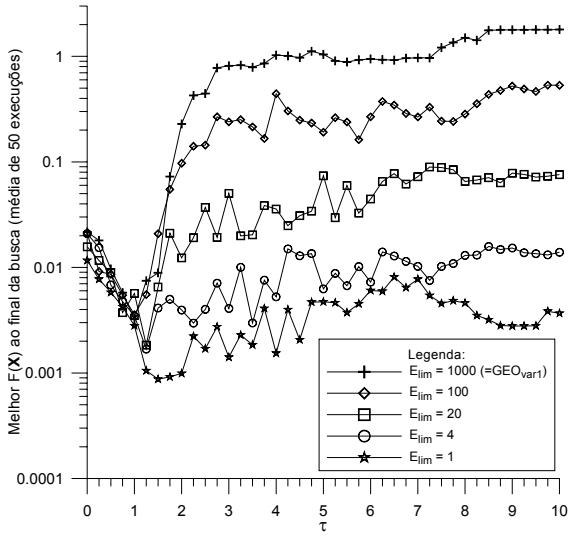

Figura 1b - FT_1 x τ x E_{lim} para GEO_{var2}

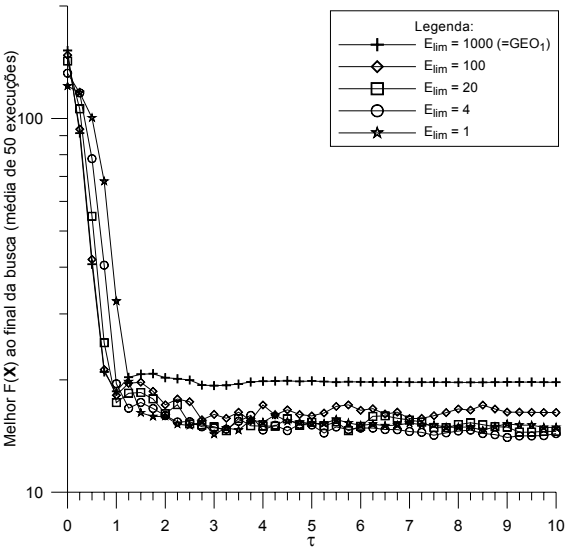

Figura 2a - FT_2 x τ x E_{lim} para GEO_2

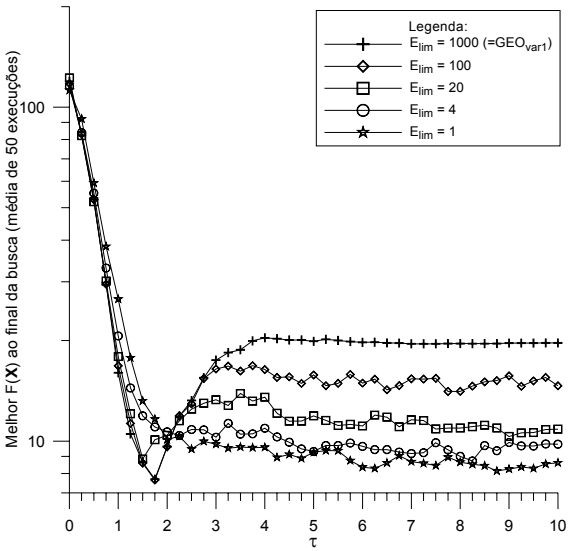

Figura 2b - FT_2 x τ x E_{lim} para GEO_{var2}

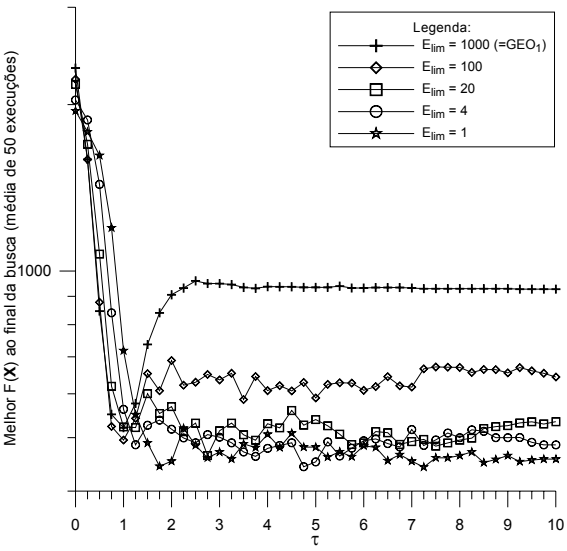

Figura 3a - FT_3 x τ x E_{lim} para GEO_2

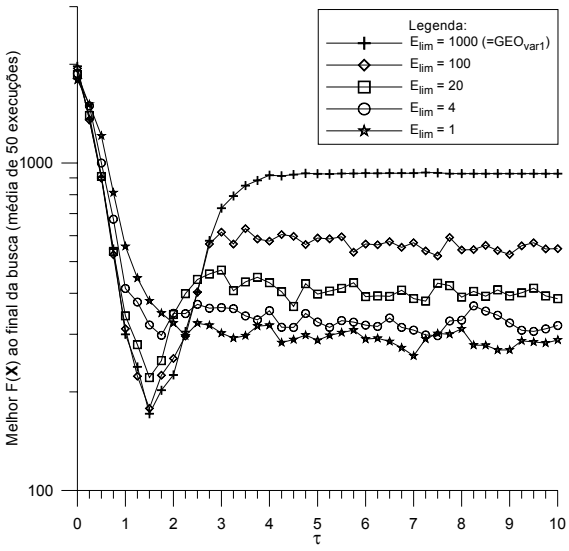

Figura 3b - FT_3 x τ x E_{lim} para GEO_{var2}

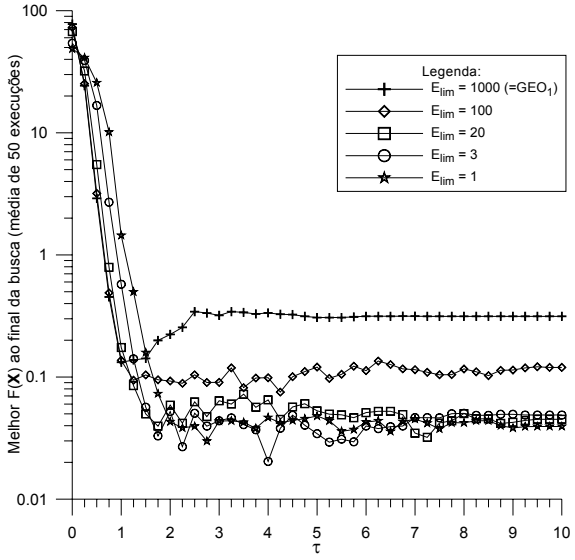

Figura 3.12a - FT_4 x τ x E_{lim} para GEO_2

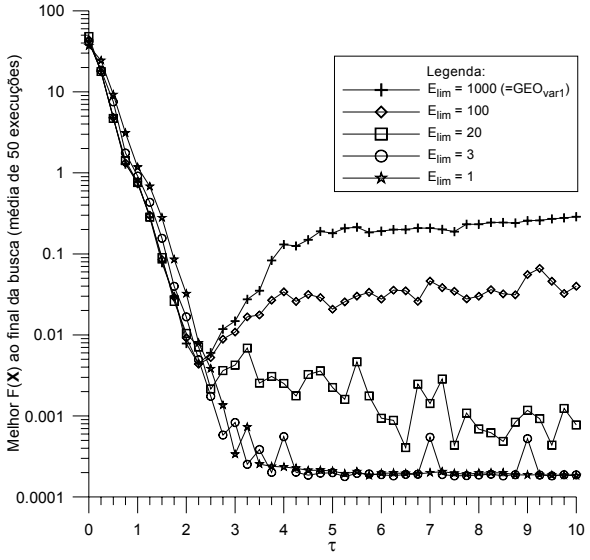

Figura 3.12b - FT_4 x τ x E_{lim} para GEO_{var2}

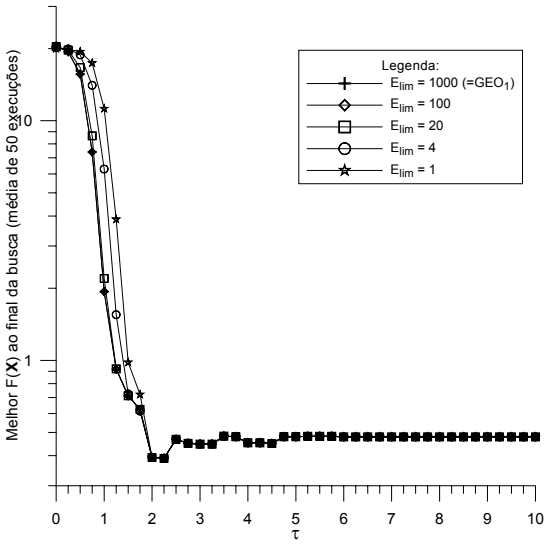

Figura 3.13a - FT_5 x τ x E_{lim} para GEO_2

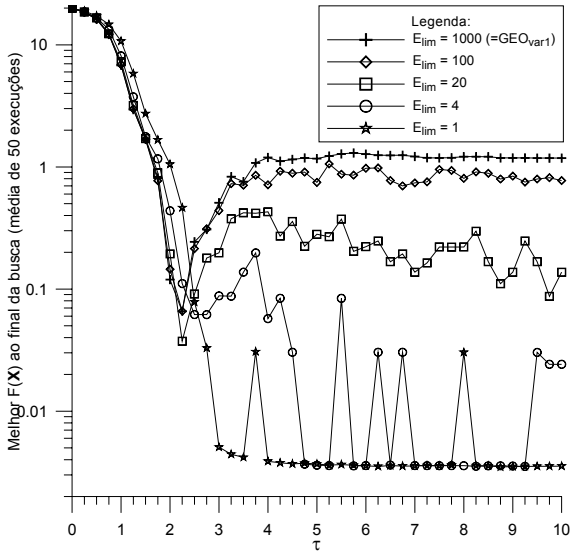

Figura 3.13b - FT_5 x τ x E_{lim} para GEO_{var2}

Analisando primeiramente os resultados obtidos para GEO_2 (figuras 1a a 5a), em termos do melhor resultado de todos, observa-se que há melhora do desempenho, relativamente ao GEO_1, e isto para todas as funções teste menos FT_5, onde ambas as versões empatam. Em termos gerais, o efeito mais evidente é a melhora do desempenho à medida que E_{lim} diminui. Outro efeito muito importante é a relativa "estabilidade" de desempenho que GEO_2 apresenta para $\tau>2$. Este fato fortalece a hipótese da ocorrência de pontos de estagnação para o GEO, principalmente para τs elevados, conforme havia sido antecipado.

Analisando agora os resultados obtidos para GEO_{var2} (figuras 1b a 5b), em termos do melhor resultado de todos, observa-se que também para esta versão há melhora de desempenho relativamente ao GEO_{var1}. Isto ocorre para as funções teste FT_1, FT_4 e FT_5. Além disso, é possivel observar que GEO_{var2} também apresenta certa estabilidade de desempenho para $\tau>4,0$.

Para FT_2 o melhor resultado ocorre com E_{lim}=100, mas é muito próximo ao obtido com E_{lim}=1000 (diferença de 0,6%). Para FT_3, o melhor resultado ocorre para E_{lim}=1000. Para estas duas funções teste, portanto, pode-se dizer que GEO_{var1} obtém vantagem. Para GEO_{var2}, o efeito geral mais evidente é também a melhora do desempenho à medida que E_{lim} diminui, mas com exceções. As exceções no caso, são FT_2 e FT_3. Nestas duas funções teste, para valores de τ até ~2,5 ocorre piora no desempenho à medida que E_{lim} diminui. Somente para valores de τ maiores que 2,5 é que a situação se inverte, ocorrendo melhora do desempenho à medida que E_{lim} diminui. Como o valor ótimo de τ para estas duas funções teste ocorre para $\tau<2,5$ então o uso de GEO_{var2} nestas duas funções ocasiona perda de desempenho. De todo modo, não deixa de ser surpreendente que também para GEO_{var1}, que se pensava "imune" à estagnação, GEO_{var2} tenha propiciado ganho de desempenho, ainda que restrito a 3 das 5 funções teste.

A Tabela 2 a seguir, apresenta os valores ótimos de τ e E_{lim} para GEO_2 e GEO_{var2}, ou seja, aqueles com os quais obteve-se o melhor resultado para cada função teste. Por melhor resultado entenda-se o menor valor médio da respectiva função teste ao final de 50 execuções independentes. Como referência, os valores ótimos de τ para GEO_1 e GEO_{var1} também são mostrados. A Tabela 2 informa também a média R do número de reinicializações ocorrido nas 50 execuções independentes. Para FT_5 com GEO_2, houve empate, tendo sido o mesmo e melhor resultado obtido para $E_{lim} \in \{3;4;20;100;1000\}$ e todos com τ=2,25 conforme apresentado na tabela. Todavia, para $E_{lim} \in \{1;2\}$, os resultados foram piores que os demais por uma diferença ínfima (0,013%). Para FT_2 e FT_3 com GEO_{var2}, os melhores resultados foram obtidos com $E_{lim} \geq 100$.

Tabela 2.1 – Valores ótimos de τ e E_{lim} e R

FT	GEO_1	GEO_2		
	τ	τ	E_{lim}	R
FT_1	0,5	1,5	1	42,1
FT_2	1,0	5,5	3	2,00
FT_3	1,0	7,25	1	10,0
FT_4	1,0	4,0	3	9,24
FT_5	2,25	2,25	{3;4;20; 10^2;10^3}	{0,26;0,2; 0,02;0;0}

[1] Convenção: {.}= o melhor resultado foi obtido para mais de um valor do parâmetro

Tabela 2.2 – Valores ótimos de τ e E_{lim} e R

FT	GEO_{var1}	GEO_{var2}		
	τ	τ	E_{lim}	R
FT_1	1,0	1,5	1	61,2
FT_2	1,75	1,75	100	0,78
FT_3	1,5	1,5	1000	0,0
FT_4	2,25	5,25	3	53,8
FT_5	2,25	8,75	1	16,2

A análise dos dados da Tabela 2a para GEO_2 corrobora a análise feita com base nas Figuras 1a a 5a, qual seja, a de que GEO_2 obteve desempenho superior (4 em 5) ou equivalente (1 em 5) a GEO_1 para todas as funções teste. Além disso, a tabela mostra que para a maioria das funções teste (4 em 5), o τ ótimo de GEO_2 ocorre para τ>2, ou seja, dentro da região de estabilidade onde o desempenho de GEO_2 não varia significativamente com o parâmetro τ. Isto é importante, pois sugere que é possivel utilizar-se *a priori* um valor elevado de τ e efetuar a varredura para descobrir o valor ótimo de E_{lim} apenas. O fato do E_{lim} ótimo de todas as funções teste para GEO_2 ter ocorrido para $E_{lim} \in \{1,2,3\}$ sugere ainda que também para este parâmetro a busca pode ser efetuada para um conjunto pequeno de valores. Já para GEO_{var2} os dados da Tabela 2b mostram que a situação é mais complexa, pois além de inesperada, não há nos resultados nenhuma indicação do porque houve melhora no desempenho em 3 das 5 funções teste (FT_1,FT_4 e FT_5), e nem tampouco porque houve piora no desempenho em 2 das 5 funções teste (FT_2 e FT_3). É possível, entretanto, observar que para as funções teste onde GEO_{var2} ocasionou melhora no desempenho, o valor ótimo para E_{lim} está restrito a um conjunto bastante pequeno ($E_{lim} \in \{1,2,3\}$). Isto, aliado à informação da relativa "estabilidade" de desempenho de GEO_{var2} observada nas Figuras 1b a 5b para τ>4,0, permite aventar como possível estratégia para o ajuste dos parâmetros τ e E_{lim} de GEO_{var2}, usar τ>4,0 fixo e fazer varredura para $E_{lim} \in \{1;2;3\}$ e, adicionalmente, utilizar GEO_{var1} (ou GEO_{var2} com $E_{lim} \to \infty$) como referência quanto ao desempenho, fazendo neste último varredura para $\tau \in [0;10]$.

4. Conclusões

Neste artigo, um novo aprimoramento dos algoritmos estocásticos de otimização global GEO e GEO_{var} foi apresentado e os resultados obtidos da sua aplicação a um conjunto de funções teste foram analisados. GEO_2 foi superior a GEO_1 em todas as funções teste à exceção de uma, onde houve empate. Este fato corrobora a hipótese de ocorrência de pontos de estagnação durante a busca com GEO/GEO_1. Surpreendentemente, GEO_{var2} também obteve resultados superiores a GEO_{var1} em mais da metade das funções teste. A auto-reinicialização demonstrou ser uma funcionalidade de valor para melhorar o desempenho de GEO_1 e GEO_{var1}.

5. Referências

Boettcher, S.; Percus, A. "Optimization With Extremal Dynamics". Physical Review Letters, v. 86, pp. 5211-5214, 2001.

Galski, R. L.; Sousa, F. L.; Ramos, F. M.; Muraoka, I. "Um Estudo para o Aprimoramento do Método da Otimização Extrema Generalizada". Terceiro Workshop dos cursos de Computação Aplicada do INPE (III WORCAP), São José dos Campos, Novembro, 2003.

Goldberg, D.E., Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Publishing, New York, 1989.

Kirkpatrick, S.; Gellat Jr., C.D.; Vecchi, M.P. "Optimizing by simulated annealing". Science, v. 220, n. 4598, pp. 671-680, 1983.

Potter, M.; De Jong, K.A., "A Cooperative Coevolutionary Approach to Function Optimization", Anais do Third Parallel Problem Solving From Nature, Springer-Verlag, pp. 249-257, 1994.

Sousa, F. L.; Ramos, F. M., "Function Optimization Using Extremal Dynamics", Anais do 4th International Conference on Inverse Problems in Engineering, Rio de Janeiro, Brasil, 2002.

Sousa, F.L., Ramos, F.M., P. Paglione, R.M. Girardi, "New Stochastic Algorithm for Design Optimization", AIAA Journal, Vol. 41, Number 9, pp. 1808-1818, 2003.

Sousa, F.L. Vlassov, V. e Ramos, F.M. "Generalized Extremal Optimization for Solving Complex Optimal Design Problems". Lecture Notes on Computer Science, 2723:375-376, 2003.