OTIMIZAÇÃO NUMÉRICA POR PROGRAMAÇÃO GENÉTICA MULTIGÊNICA

Adriano Soares Koshiyama
PUC-RJ - Rua Marquês de São Vicente, 225, Gávea - Rio de Janeiro/RJ
adriano@ele.puc-rio.br

Douglas Mota Dias
PUC-RJ - Rua Marquês de São Vicente, 225, Gávea - Rio de Janeiro/RJ
douglasmele.puc-rio.br

Marley Maria Bernardes Rebuzzi Vellasco
PUC-RJ - Rua Marquês de São Vicente, 225, Gávea - Rio de Janeiro/RJ
marley@ele.puc-rio.br

Ricardo Tanscheit
PUC-RJ - Rua Marquês de São Vicente, 225, Gávea - Rio de Janeiro/RJ
ricardo@ele.puc-rio.br

Resumo: Esse trabalho apresenta um novo método para problemas de otimização chamado de OGP (Optimization by Genetic Programming). Esse método é baseado em Programação Genética Multigênica. O OGP pode ser visto como uma generalização do PMA (Parameter Mapping Approach) que usa a Programação Genética clássica. São discutidos os aspectos teóricos do método, seus operadores e representação da solução buscada. Após, é efetuado testes com o OGP em oito funções benchmarks, comparando os resultados encontrados com respeito ao Algoritmo Géneticos, Evolução Diferencial, Enxame de Partículas e o PMA. Por fim, são efetuadas análises estatísticas de forma a comprovar os resultados promissores do OGP nas funções objetivos usadas.

Palavras-Chave: Programação Genética Multigênica; Otimização; Problemas Numéricos.

Abstract: This work presents a new method for optimization problems, called by us as OGP. This method is based on Multi-Gene Genetic Programming. The OGP can be seen as an extension of the Parameter Mapping Approach (PMA) that use the canonical Genetic Programming. We discuss the theoretical aspects, the operators and representation of the method, and thus performed comparisons with Genetic Algorithms, Differential Evolution, Particle Swarm and PMA on a set of eight benchmarks functions. We made statistical analyses in order to demonstrate the promising results expressed by OGP on the benchmarks used.

Keywords: Multi-Gene Genetic Programming; Optimization; Numerical Problems.

1. Introdução

Problemas de otimização são encontrados em grande número na literatura acadêmica e industrial. São desde problemas artificiais de otimização numérica e de ordenação (BOYD e VANDENBERGHE, 2004), visando ao aperfeiçoamento dos algoritmos de otimização, ou ainda problemas reais, tais como alocação ótima de poços de petróleo (PACHECO e VELLASCO, 2008), sistemas de distribuição de energia (SOLIMAN e MANTAWY, 2012), entre outros.

Tradicionalmente, estes problemas são solucionados via programação matemática (NOCEDAL e WRIGHT, 2006), que em sua maioria usam a informação do gradiente da função como política de atualização dos parâmetros buscados. Com o advento de meta-heurísticas, como Algoritmos Genéticos (GA) (HOLLAND, 1975), Enxame de Partículas (PSO) (KENNEDY et al., 2001) e Evolução Diferencial (DE) (STORN e PRICE, 1997) foi possível obter, para determinados problemas de otimização, soluções mais céleres, ou superiores à programação matemática.

Estes resultados auferidos pelas meta-heurísticas são possíveis devido a duas principais características:

- Não usam o gradiente da função como informação para atualização dos parâmetros;
- Cada uma possui um conjunto de operadores e conceitos para atualização da solução de forma distinta.

A não utilização do gradiente se torna uma vantagem em situações que seu cálculo é custoso, como também para problemas onde as funções são não convexas. Como cada meta-heurística aplica uma coleção diferente de operações para atualizar sua solução, estas tendem a trafegar por espaços de busca distintos, e, portanto, torna uma mais eficiente para um determinado problema do que outra (MICHALEWICZ, 1994).

Uma outra meta-heurística é a Programação Genética (PG) (KOZA, 1992; LANGDON e POLI, 2002; POLI et al., 2008), cujo uso predominante são em problemas de classificação, previsão, controle, aproximação de funções, ou também em aplicações de desenvolvimento e elaboração de circuitos, artes e música, entre outras. Porém, aplicações na área de otimização são exíguas. Logo, este trabalho propõe o uso da Programação Genética Multigênica (HINCHLIFFE et al., 1996), uma variante da Programação Genética Clássica (KOZA, 1992), para solucionar problemas de otimização que, neste trabalho, são de natureza numérica. Este método, denominado de OGP (Optimization by Genetic Programming) é uma generalização do método precedente proposto por Pujol e Poli (2008) chamado Parameter Mapping Approach (PMA).

O objetivo desde trabalho é, portanto, apresentar e comparar o método OGP com demais meta-heurísticas (GA, PSO, DE) e o PMA, visando a otimização de problemas numéricos. A próxima seção exibe uma revisão bibliográfica sobre a aplicação da Programação Genética em otimização. A terceira seção apresenta a formulação teórica do OGP, seus comparativos e distinções do PMA. A seção 4 descreve os resultados e discussões dos testes propostos para a metodologia, enquanto que a seção 5 exibe as considerações finais.

2. Trabalhos Relacionados

Um dos trabalhos encontrados nesse sentido é o de Pujol e Poli (2008), que apresenta o Parameter Mapping Approach (PMA). Considere como $\mathbf{X}^*$ o vetor de parâmetros que minimiza uma função objetivo $f(.)$. O PMA busca encontrar, a partir de um $\mathbf{X}$ arbitrário

(default) e fixadas inicialmente, uma função $PMA(\mathbf{X})$ tal que $\mathbf{X}^* \simeq PMA(\mathbf{X})$. A figura 1 ilustra um exemplo desse processo.

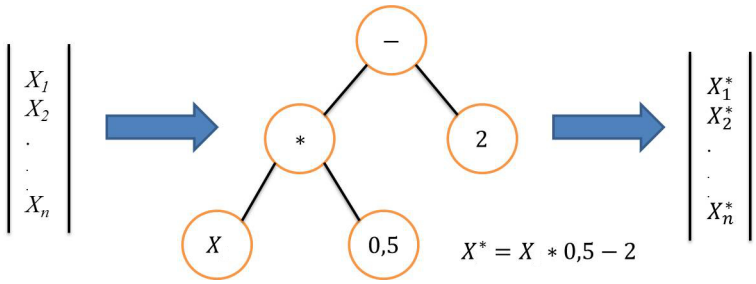

Figura 1: Exemplificação do algoritmo PMA.

Portanto, a PG é utilizada para sintetizar essa função, a partir do uso de operadores de recombinação e mutação. É fácil notar que o método funciona relativamente bem em problemas cujas soluções são triviais, como exibido em Pujol e Poli (2008). Por exemplo, considere $\mathbf{X}^* = 0$, para todo $i = 1, ..., n$, como o vetor de soluções ótimas para uma $f(.)$. Então, basta a PG encontrar uma função $PMA(\mathbf{X}) = \mathbf{X} - \mathbf{X}$, que o problema é concluído com otimalidade.

Entretanto, quando o conjunto de soluções ótimas não é trivial, o PMA tende a enfrentar maiores dificuldades. As figuras 2a e 2b dispõem, graficamente, soluções ótimas quando $\mathbf{X}^* = 1$ (soluções triviais) e $\mathbf{X}^* = 1 + r_i$ (soluções complexas), onde $r_i \sim U(-1, 1)$, respectivamente.

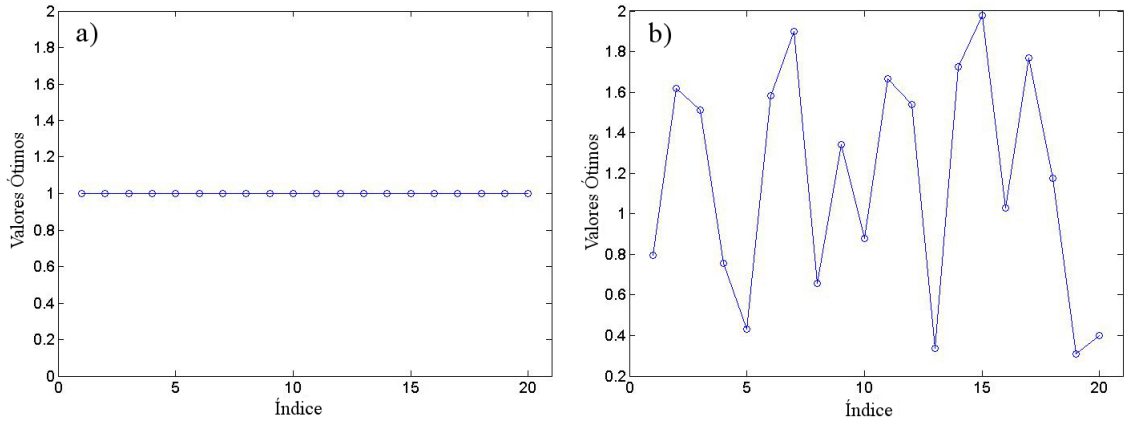

Figura 2: Solução trivial (a) e não trivial (b).

Logo, buscar uma $PMA(\mathbf{X})$ tal que demonstre um comportamento semelhante ao da figura 2b é notoriamente mais complexo do que o da figura 2a. Este caso é relatado em Pujol e Poli (2008), que apresenta uma maior incidência de resultados inferiores.

Uma possível solução é estabelecer um conjunto de funções responsáveis em buscar determinados parâmetros, ou seja, particionar o problema na determinação de um conjunto de funções ao invés de uma única ser responsável por todos os parâmetros. Esta solução, denominada de OGP, é apresentada na próxima seção.

3. Metodologia

3.1 Aspectos do algoritmo OGP

De forma ampla, o método OGP consiste em sintetizar uma quantidade k de funções a partir da Programação Genética Multigênica (PGMG) (HICHLIFFE et al., 1996). Esta variação da PG clássica permite a evolução simultânea de k funções, onde k é um número definido pelo usuário, tal que as funções destes compõem um único indivíduo. Portanto, um indivíduo é um complexo de funções (programas). A PGMG tem sido utilizada com sucesso em problemas de regressão simbólica (SEARSON et al., 2007; MORRISON et al., 2010), porém não foi verificada aplicações desta em problemas de otimização.

Para exemplificar a distinção entre o PMA e o OGP, considere a figura 3 que demonstra os parâmetros ótimos (pontos no gráfico) de um problema de otimização hipotético. A figura 3a apresenta a função que deve ser encontrada pelo PMA para solucionar o problema de otimização, enquanto que a figura 3b exibe a solução encontrada pelo OGP. Neste caso, $k = 3$, e como $\mathbf{X} = \{X_1, X_2, ..., X_{14}\}$ tem-se $\mathbf{X^{(1)}} = \{X_1, ..., X_4\}$, $\mathbf{X^{(2)}} = \{X_5, ..., X_9\}$ e $\mathbf{X^{(3)}} = \{X_9, ..., X_{14}\}$, e respectivamente $OGP_1(\mathbf{X^{(1)}})$, $OGP_2(\mathbf{X^{(1)}})$ e $OGP_3(\mathbf{X^{(1)}})$. Logo o método OGP particiona em três o número de parâmetros que são buscados, dispondo para cada função a responsabilidade de otimizar em torno da terça parte do problema. No limite, quando $k = n$, sendo n o número de parâmetros a serem otimizados, o OGP elabora, para cada parâmetro, uma função que recebe um valor arbitrário e retorna a transformação deste valor pela função decodificada no indivíduo.

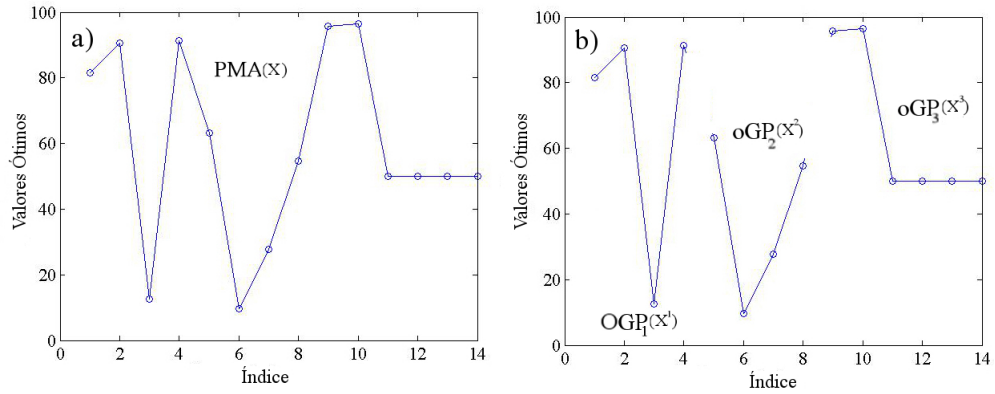

Figura 3: Exemplo da distinção das soluções encontradas por PMA e OGP.

Da mesma forma, quando $k = 1$, o OGP se resume ao PMA. Cabe ressaltar que no caso limite ($k = n$), não é necessário incluir valores arbitrários para os parâmetros (o vetor $\mathbf{X}$, como no exemplo) como entrada do OGP, incluindo somente as constantes usualmente inseridas em uma rotina de PG. Como cada função retorna somente um valor para um dos parâmetros da função objetivo, este pode ser representado por um número que é resultado de combinações aritméticas de constantes.

De forma geral, o pseudo-código do método OGP, para uma função objetivo qualquer, é descrito abaixo:

O OGP tem início a partir da especificação dos parâmetros populacionais (tamanho da população, número de gerações, etc.), dos terminais e operações matemáticas e operadores genéticos (taxa de cruzamento, mutação, etc.). Em seguida o usuário estabelece o valor de k, que é o número de funções incorporadas em um indivíduo.

A partir dessas especificações, uma população de indivíduos com k funções é gerada aleatoriamente. As soluções expressas por cada indivíduo são avaliadas na função objetivo do problema de otimização explorado. Após toda a população ter sido avaliada, verifica-se se o critério de parada foi atendido; em caso positivo, retorna a população atual; caso contrário o algoritmo entra em um laço que só é interrompido caso o critério de parada seja alcançado.

```
begin
    Definir parâmetros da OGP;
    Definir k;
    t = 0;
    Iniciar População(t);
    Fit = Avaliar(População(t));
    while Fit ≠ Critério de Parada do
        t=t+1;
        População(t) = Seleciona(População(t-1));
        População(t) = AplicaOperadores(População(t));
        Fit = Avaliar(População(t));
    end
    Retorna População(t) ;
end
```

Durante o laço são efetuadas três operações: seleção, aplicação de operadores e avaliação. A primeira consiste em escolher, a partir de alguns métodos heurísticos (roleta, torneio, etc.) (POLI et al., 2008), os entes da próxima população. A esta nova população são aplicados operadores de mutação e cruzamento de baixo e alto nível.

Como exemplo, considere uma execução do OGP com $k = n = 5$. A figura 4 exibe o cruzamento de baixo nível, enquanto que a figura 5 o procedimento de mutação, ambos semelhante à forma canônica da PG.

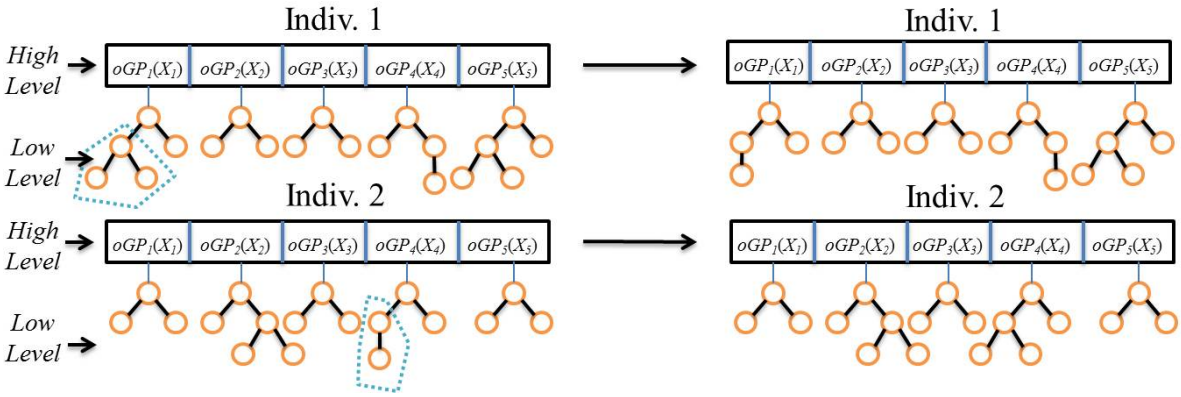

Figura 4: Exemplo de operação de cruzamento de baixo nível.

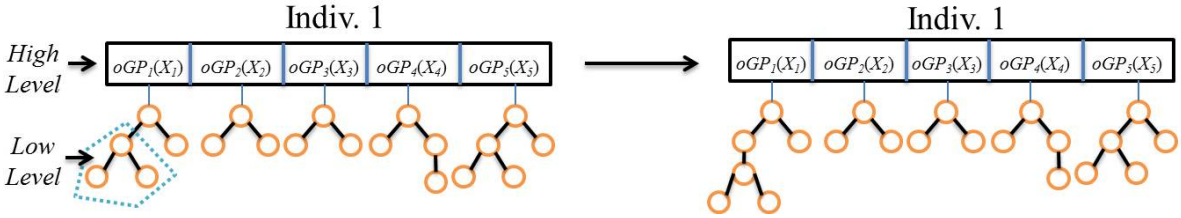

Figura 5: Exemplo de operação de mutação.

As figuras 4 e 5 apresentam os dois níveis de um indivíduo: baixo e alto nível. O baixo nível se configura pela presença de 5 funções, dispostas em estruturas de árvore. O alto nível representa as saídas de cada função, denotadas por um vetor semelhante a um cromossoma. Então, pelas figuras 4 e 5 verifica-se que os resultados alcançados

pelas operações são bastante semelhantes, apesar das operações serem distintas em sua concepção. A figura 6 dispõe um cruzamento de alto nível.

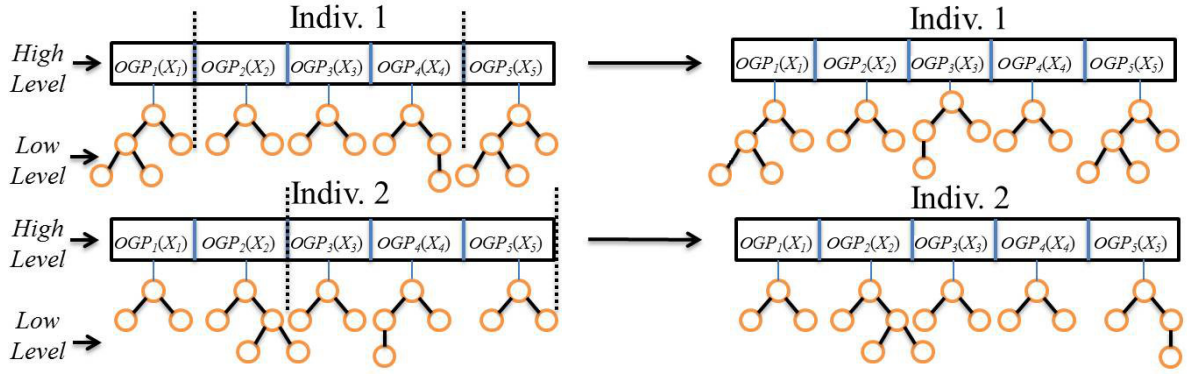

Figura 6: Exemplo de operação de cruzamento de alto nível.

O cruzamento de alto nível consiste na aplicação de um corte de dois pontos, tal que o número de funções permutadas de um indivíduo a outro é idêntico. Em comparação com o de baixo nível, torna-se notório que o cruzamento de alto nível implica em alterações em nível macro do indivíduo, enquanto que o de baixo nível altera em níveis micro. Portanto, o de alto nível tende a afetar mais substancialmente a avaliação de um indivíduo, do que o de baixo nível.

Esta nova população é reavaliada e novamente é verificado se o critério de parada do algoritmo foi alcançado; se for atendido a população atual é retornada, caso contrário este processo é retomado quantas vezes for necessário. A próxima seção apresenta os experimentos realizados com o algoritmo OGP.

3.2 Experimentos Propostos

Em linhas gerais, os experimentos realizados seguiram os de Pujol e Poli (2008), visando à comparação com o PMA. A seguir são detalhados os demais métodos que foram usados para comparação, suas parametrizações, forma de avaliação e a implementação computacional do OGP e as análises estatísticas efetuadas.

3.2.1 Comparação com demais métodos

O desempenho do método OGP foi comparado com relação a quatros diferentes metaheurísticas de otimização: Evolução Diferencial (DE), Enxame de Partículas (PSO), Algoritmo Genético (AG) com codificação real e o PMA (MICHALEWICZ, 1994; STORN e PRICE, 1997; KENNEDY et al., 2001; PUJOL e POLI, 2008). Os resultados alcançados por esses quatro métodos foram obtidos em Pujol and Poli (2008).

3.2.2 Funções de teste

Para explorar o potencial do método OGP, este foi aplicado ao mesmo conjunto de oito funções benchmarks que o PMA foi submetido. As cinco primeiras (f_1, f_2, f_3, f_4, f_5) pertencem ao conjunto de teste de De Jong (MICHALEWICZ, 1994). Outras três funções foram selecionadas: a parábola de Corona (f_6), a função de Griewangk (f_7) e a função de Schwefel (f_8) (INGBER e ROSEN, 1992). A tabela 1 exibe as principais características de cada função objetivo. A função de Schwefel foi executada com diferentes números de

parâmetros, visando a análise da suscetibilidade do método segundo a dimensionalidade do problema.

Tabela 1: Principais características das funções benchmarks.

Função	Nome da Função	Número de Parâmetros	Mínimo Global
f_1	De Jong 1	3	0
f_2	De Jong 2	2	0
f_3	De Jong 3	5	0
f_4	De Jong 4	30	$0+\bar{v}$, $\bar{v} \sim U(0,1)$
f_5	De Jong 5	25	0.998004
f_6	Parábola de Corona	4	0
f_7	Função de Griewangk	10	0
f_8	Função de Schwefel	[10, 30, 50, 70, 100]	$\approx$ 420,97

Para cada uma destas funções, as parametrizações em termos de operadores e demais detalhes de cada algoritmo de otimização são fornecidos nas tabelas 2 e 3. Todas as informações dispostas nas tabelas 2 e 3 foram as disponíveis em Pujol e Poli (2008), excetuando às referentes ao OGP. As parametrizações básicas e a disposição das informações do método OGP seguiram as do método PMA, visando a plena comparação entre estes. A única distinção ocorre que no OGP não foram incluídos valores arbitrários para as variáveis, mas somente constantes no intervalo $[-10, 10]$, e portanto, o número de funções embutidas em cada indivíduo foi igual ao número de parâmetros a serem otimizados ($k = n$). O conjunto de operações matemáticas e outros parâmetros de uso genérico nos testes efetuados sao apresentados na tabela 4.

Tabela 2: Parâmetros usados na otimização pelo DE, PSO and AG. $*$

Fitness	DE			PSO		AG			
function	POP	SF	CP	POP	MSP	POP	CP	MP	TS
f_1	10	0.50	0.30	10	0.01	500	0.90	0.10	20
f_2	6	0.95	0.50	10	0.01	500	0.90	0.10	0.10
f_3	10	0.80	0.30	20	10	200	0.50	0.10	100
f_4	10	0.75	0.50	5	0.10	100	0.30	1.00	5
f_5	15	0.90	0.30	150	1.00	1	0.90	0.10	20
f_6	10	0.40	0.20	200	0.10	500	0.50	0.10	20
f_7	30	1.00	0.30	200	0.01	2	0.80	0.20	2
f_8	20	0.50	0.50	50	1.00	2	0.50	0.10	100

$*$ POP - População, SF - Auto-Adaptação, CP - Probabilidade de Cruzamento, MP - Probabilidade de Mutação, MSP - Termo de momento do enxame, TS - Tamanho do Torneio.

Para o primeiro conjunto de experimentos, foram realizadas 100 execuções independentes de cada algoritmo para cada função objetivo. O número máximo de avaliações é de 2×10^6. Para o segundo conjunto de experimentos, as funções f_1 a f_7 foram novamente utilizadas, porém o valor dos parâmetros ótimos sofrem uma perturbação r_i, tal que $r_i \sim U(-1, 1)$. Por exemplo, enquanto que no primeiro conjunto de experimento é $f_1(X_1, X_2, X_3)$, no segundo é $f_1(X_1 + r_1, X_2 + r_2, X_3 + r_3)$. No caso da f_8 foi aplicada uma matriz de rotação, cujos elementos foram selecionados a partir de uma distribuição uniforme $U(-1, 1)$.

Tabela 3: Parâmetros usados na otimização pelo PMA e OGP. ∗

Fitness	PMA				OGP		
function	POP	FC	TS	NV	POP	FC	TS
f_1	100	0.90	100	100	100	1.00	100
f_2	50	0.90	50	100	50	1.00	50
f_3	50	0.90	50	100	50	1.00	50
f_4	100	0.90	100	100	100	1.00	100
f_5	50	0.90	50	100	50	1.00	50
f_6	200	0.90	100	100	200	1.00	100
f_7	100	0.90	100	100	100	1.00	100
f_8	100	0.90	100	100	100	1.00	100

∗ FC - Taxa de geração de constantes, NV - Número de Variáveis.

Tabela 4: Configurações genéricas do OGP durante os testes.

Parâmetros	Valores
Taxa de Cruzamento de Alto-Nível	65%
Taxa de Cruzamento de Baixo-Nível	65%
Taxa de Mutação	35%
Taxa de Elitismo	1%
Pressão Lexicográfica (LUKE e PANAIT, 2002)	Yes
Terminais	Constantes no intervalo [-10, 10]
Operações Matemáticas	$\{+, -, \times, \div, \sin, \cos, \exp\}$

3.2.3 Formas de avaliação

A forma de avaliação de um indivíduo da população foi definida como $|f(OGP(\mathbf{X}))$ $-f(\mathbf{X}^*)|$, onde $f(OGP(\mathbf{X}))$ é o valor computado para a função, e $f(\mathbf{X}*)$ é o mínimo global conhecido dentro do domínio especificado para cada função. Esta busca pelo mínimo global foi interrompida quando a avaliação do melhor indivíduo foi maior ou igual à 10^-6. Nos experimentos com a função Schwefel, devido à crescente dificuldade em alcançar uma solução quando o número de variáveis aumenta, o critério de parada em relação ao mínimo global foi reduzido para uma diferença menor ou igual a 10^2.

3.2.4 Implementação computacional e análise estatísticas

O algoritmo de otimização OGP foi implementado em MATLAB R2010a (MATLAB, 2010). Análises estatísticas foram aplicadas com o objetivo de verificar qual algoritmo aproximou-se do mínimo global da função com menor número de avaliações. Para esta abordagem, seguiu-se as recomendações de Derrac et al. (2011) que sugerem um conjunto de testes estatísticos não paramétricos para a validação e escolha do melhor algoritmo. A partir disso, empregou-se o teste de Friedman para verificar se a hipótese de diferença significativa entre os algoritmos é valida, e, em seguida, o algoritmo melhor posicionado, segundo a abordagem do teste de Friedman, foi submetido ao teste de Holm para verificar em relação a quais algoritmos este é significativamente superior. Estes testes foram efetuados no software KEEL (ÁLCALA-FDEZ et al., 2009), com um nível de significância adotado de 5%.

Tabela 5: Resultados do primeiro ($f_n|n \in [1..8]$) e segundo conjunto de experimentos ($f_n^*|n \in [1..8]$).

	DE		PSO		AG		PMA		OGP	
$f(.)$; n	NMA (σ)	Hits	NMA (σ)	Hits	NMA (σ)	Hits	NMA (σ)	Hits	NMA (σ)	Hits
f_1; 3	487 (4.9e1)	98	4,164 (8.5e2)	100	1401 (3.6e2)	100	905 (1.2e3)	100	257 (3.3e2)	100
f_2; 2	1100 (1.4e3)	94	4,503 (1.8e3)	100	7189 (4.2e3)	100	3,653 (6.5e3)	100	620 (6.3e2)	100
f_3; 5	156 (4.8e1)	100	52 (1.1e1)	100	509 (2.1e1)	100	30 (3.5e1)	100	103 (6.0e1)	100
f_4; 30	2688 (5.3e2)	100	461 (1.6e2)	100	2918 (1.1e3)	100	92 (5.6e1)	100	656 (2.5e2)	100
f_5; 25	706 (1.3e2)	100	3,262 (1.2e3)	100	4174 (1.7e3)	100	413 (4.0e2)	100	2698 (2.6e3)	100
f_6; 4	1235 (1.0e2)	88	1.8e6 (1.2e5)	66	1.0e4 (9.9e3)	100	9,639 (1.2e4)	100	495 (5.4e2)	100
f_7; 10	3.6e4 (1.5e3)	88	2.0e6 (0.0e0)	0	1.0e5 (1.2e4)	100	3654 (7.4e3)	100	6328 (6.2e3)	100
f_8; 10	4.8e5 (3.9e5)	98	2.0e6 (0.0e0)	0	1.7e4 (1.0e4)	93	1657 (3.2e3)	100	5100 (8.2e3)	100
f_8; 30	7.0e5 (4.8e5)	75	2.0e6 (0.0e0)	0	3.4e4 (1.6e4)	99	1827 (3.0e3)	100	1435 (1.7e3)	100
f_8; 50	8.9e5 (5.6e5)	58	2.0e6 (0.0e0)	0	4.8e4 (2.3e4)	98	1764 (2.3e3)	100	1278 (8.6e2)	100
f_8; 70	8.8e5 (6.0e5)	50	2.0e6 (0.0e0)	0	6.4e4 (2.4e4)	99	2378 (5.0e3)	100	1292 (5.3e2)	100
f_8; 100	8.9e5 (5.6e5)	35	2.0e6 (0.0e0)	0	1.6e5 (2.2e5)	50	1404 (1.1e3)	100	1477 (5.5e2)	100
f_1^*; 3	498 (5.3e1)	98	4504 (8.6e2)	100	1382 (3.1e2)	100	2817 (1.5e3)	100	264 (2.0e2)	100
f_2^*; 2	1221 (1.6e3)	95	4338 (1.7e3)	100	8890 (1.2e4)	100	5250 (4.6e3)	100	3,353 (4.8e3)	100
f_3^*; 5	134 (3.9e1)	100	51 (3.0e1)	100	337 (1.1e2)	100	42 (3.9e1)	100	122 (4.0e1)	100
f_4^*; 30	3821 (7.9e2)	100	761 (1.8e2)	100	4473 (1.5e3)	100	1.0e5 (5.0e4)	100	922 (2.2e2)	100
f_5^*; 25	697 (1.3e2)	96	3278 (1.1e3)	100	4959 (4.4e3)	100	885 (8.3e3)	100	1,223 (6.4e3)	100
f_6^*; 4	1233 (9.7e1)	91	1.8e5 (1.2e5)	60	9755 (2.2e4)	100	2.6e4 (1.6e4)	100	822 (4.0e2)	100
f_7^*; 10	3.1e4 (1.4e3)	100	2.0e6 (0.0e0)	0	1.0e5 (1.6e4)	100	1.7e4 (8.2e3)	100	1.2e4 (7.6e3)	100
f_8^*; 10	3.2e5 (3.7e5)	98	1.0e5 (1.0e5)	99	3.7e4 (2.8e4)	100	1131 (1.2e3)	100	2782 (1.8e3)	100
f_8^*; 30	7.7e5 (5.7e5)	72	3.7e5 (3.2e5)	99	1.1e5 (8.7e4)	100	3146 (1.3e4)	100	2904 (5.2e3)	100
f_8^*; 50	7.8e5 (5.8e5)	55	7.2e5 (5.0e5)	95	4.2e5 (3.5e5)	97	5320 (2.3e4)	100	3685 (6.4e3)	100
f_8^*; 70	8.9e5 (5.8e5)	39	6.0e5 (4.6e5)	81	6.1e5 (4.8e5)	79	3557 (8.9e3)	99	5240 (7.6e3)	100
f_8^*; 100	8.6e5 (5.8e5)	45	7.6e5 (5.3e5)	71	7.8e5 (5.4e5)	46	2621 (4.1e3)	100	6540 (8.5e3)	100

4. Resultados e Discussões

4.1 Apresentação dos Resultados

A tabela 5 apresenta os resultados para o primeiro ($f_j(.)$) e segundo conjunto ($f_j^*(.)$) de experimentos. Estes são descritos em termos de número médio de avaliações (NMA) para alcançar o critério de parada, desvio padrão das avaliações (σ) e o número de vezes em que o algoritmo alcançou a tolerância em relação ao mínimo global da função. Lembrando que para o segundo conjunto de experimentos foram usadas as mesmas funções f_1 a f_7, porém com a adição de uma perturbação r_i com distribuição $U(-1, 1)$ nos valores ótimos dos parâmetros. A f_8 teve seus parâmetros rotacionados, e estes variaram entre 10, 30, 50, 70 e 100.

4.2 Discussões e Análises Estatísticas

No primeiro conjunto de experimentos é notório a alternância nos melhores resultados (menor NMA) entre os algoritmos OGP e o PMA, enquanto que no segundo conjunto os resultados apresentados pelo PMA não foram tão expressivos quanto no primeiro conjunto, excetuando-se na f_8 rotacionada.

O algoritmo OGP auferiu resultados expressivos em ambos os experimentos, sendo 11 vezes o melhor algoritmo em 24 testes efetuados (12 no primeiro e 12 no segundo conjunto de experimentos). O posto médio (que mensura a posição média do algoritmo nos diferentes testes) do OGP foi de 1,79, do PMA de 2,26, do DE de 2,89, do GA de 4 e PSO de 4,05. O teste de Friedman, que a partir do posto de cada algoritmo verifica se existiu um ou mais algoritmos que obtiveram um posto significativamente menor do que os demais, obteve um p-valor menor que $0,01$. Portanto, foi efetuado o teste de Holm para comparar o OGP com os demais algoritmos (tabela 6).

Verifica-se que o OGP obteve um posto médio significativamente menor do que o PSO, DE e AG. No caso do PMA, os resultados não foram estatisticamente significativos. Tais

Tabela 6: Resultados do teste de Holm.

i	Método	$Z = (Rank_{OGP} – Rank_i)/SE$	p-valor
1	PSO	4,41	0,0001
2	AG	4,31	0,0002
3	DE	2,15	0,0312
4	PMA	0,92	0,3558

resultados eram aguardados, pois como o OGP é uma generalização do PMA, este deveria apresentar resultados tão substanciais ou melhores do que o PMA. Como neste caso um indivíduo da população do OGP codifica um número de funções equivalente ao número de parâmetros otimizados em cada função ($k = n$), e o PMA se situa em outro extremo ($k = 1$), deve existir um número p natural, tal que $1 \leq p \leq n$ e que quando $k = p$, a OGP com p funções produza resultados superiores aos encontrados.

5. Conclusões

Este trabalho apresentou o método de otimização OGP (Optimization by Genetic Programming), uma generalização do método PMA (Parameter Mapping Approach) de Pujol e Poli (2008), que torna-se uma nova metodologia para resolução de problemas de otimização. Os resultados auferidos pelo OGP mostraram-se promissores, alcançando substancialmente melhores métricas de avaliação do que meta-heurísticas já estabelecidas na literatura.

Um caminho promissor é permitir que o próprio algoritmo defina o número de funções a serem incorporadas em um indivíduo (ou seja, o valor de k). Assim, este pode transitar entre diferentes representações e tipos de solução, permitindo que o método em si se adeque ao problema, e não o contrário. Ainda, outros problemas de otimização, tanto de otimização de funções, ou de ordenação, com comparação com demais meta-heurísticas devem ser efetuadas para melhores investigações e aprimoramentos do OGP.

Além disso, trabalhos futuros devem focar em investigações a respeito da redução de dimensionalidade do problema, que pode ser proporcionada pelo uso da OGP. Portanto, como apresentado na figura 3, este método possibilita encontrar um número n de parâmetros de uma função objetivo, usando uma ou mais funções, ou seja, $k \leq n$. Por exemplo, é possível resolver um problema com 1000 variáveis sintetizando uma única função que ao receber valores fixos, retorne 1000 valores ótimos para cada variável, transformando um problema com um espaço de busca imenso, em um outro possivelmente menor e mais fácil de se resolver. Logo, trabalhos futuros devem contemplar essa linha de resolução de problemas com alta dimensionalidade.

Referências Bibliográficas

ALCALÁ-FDEZ, J. et al. KEEL: a software tool to assess evolutionary algorithms for data mining problems. **Soft Computing**, v. 13, n. 3, p. 307-318, 2009.

BOYD, S.; VANDENBERGHE, L. **Convex optimization**. Cambridge: Cambridge university press, 2004.

DERRAC, J.; GARCÍA, S.; MOLINA, D.; HERRERA, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, v. 1, n. 1,

p. 3-18, 2011.

HINCHLIFFE, M.P.; WILLIS, M.J.; HIDEN, H.; THAM, M.T.; MCKAY, B.; BARTON, G.W. Modelling Chemical Process Systems Using a Multi-Gene. **Late Breaking Papers at the Genetic Programming 1996**, p. 56-65, 1996.

HOLLAND, J. **Adaptation in Natural and Artificial Systems**. Cambridge: MIT Press, 1975.

INGBER, L.; ROSEN, B. Genetic algorithms and very fast simulated reannealing: A comparison. **Mathematical and Computer Modelling**, v. 16, n. 11, p. 87-100, 1992.

KENNEDY, J.F.; KENNEDY, J.; EBERHART, R.C. **Swarm intelligence**. New Jersey: Morgan Kaufmann Pub, 2001.

KOZA, J.R. **Genetic Programming**: On the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.

LANGDON, W.B.; POLI, R. **Foundations of Genetic Programming**. Berlin: Springer-Verlag, 2002.

LUKE, S.; PANAIT, L. Lexicographic parsimony pressure. In: **Proceedings of the Genetic and Evolutionary Computation Conference**. p. 829-836, 2002.

MATLAB. Versão 7.10.0 (R2010a). Natick: The MathWorks Inc., 2010.

MICHALEWICZ, Z. **Genetic Algorithms + Data Structures = Evolution Programs**. 2 ed. Berlin: Springer-Verlag, 1994.

MORRISON, G.; SEARSON, D.; WILLIS, M. Using genetic programming to evolve a team of data classifiers. **Journal of Chemometrics**, v. 2, p.592-603, 2007.

NOCEDAL, J.; WRIGHT, S.J. **Numerical optimization**. Berlin: Springer, 2006.

PACHECO, M.A.C.; VELLASCO, M.M.B.R. **Intelligent Systems in Oil Field Development under Uncertainty**. New York: Springer, 2009.

POLI, R.; LANGDON, W.B.; MCPHEE, N.F. **A Field Guide to Genetic Programming**. Raleigh: Lulu.com, 2008.

PUJOL, J.C.F.; POLI, R. Parameter Mapping: A genetic programming approach to function optimization. **International Journal of Knowledge-based and Intelligent Engineering Systems**, v.12, n.1, p.29-45, 2008.

SEARSON, D.; WILLIS, M.; MONTAGUE, G. Co-evolution of non-linear pls model components. **Journal of Chemometrics**, v. 1, p.492-503, 2007.

SOLIMAN, S.A.H; MANTAWY, A.A.H. **Modern Optimization Techniques with Applications in Electric Power Systems**. New York: Springer, 2012.

STORN, R.; PRICE, K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization**, v. 11, n. 4, p. 341-359, 1997.