BUSCA LOCAL INTENSIVA: UMA NOVA METAHEURÍSTICA PARA OTIMIZAÇÃO GLOBAL CONTÍNUA RESTRITA

Wendel A. X. Melo
(Aluno de graduação)
Universidade Federal do Rio de Janeiro, Instituto de Matemática.
wendel@dcc.ufrj.br

Marcia H. C. Fampa
(Professora orientadora)
Universidade Federal do Rio de Janeiro, Instituto de Matemática e COPPE.
fampa@cos.ufrj.br

Fernanda M. P. Raupp
(Professora orientadora)
Pontifícia Universidade Católica do Rio de Janeiro,
Departamento de Engenharia Industrial.
fraupp@puc-rio.br

RESUMO

Este trabalho apresenta uma nova metaheurística sem uso de derivação para otimização global contínua restrita denominada Busca Local Intensiva (BLI). BLI trabalha com a reformulação do problema considerado em um problema de otimização bi-objetivo e procura por boas soluções para o mesmo fazendo uso de estratégias lançadas por metaheurísticas de otimização combinatória, tais como multi-inícios aleatórios, conjuntos elite e procedimento de intensificação de soluções, que dão apoio a ciclos de buscas locais executadas em vizinhanças cada vez menores em torno da solução corrente. A eficiência e a eficácia do método proposto são demonstradas através de sua aplicação sobre um conjunto padrão de 13 instâncias de teste bem conhecidas e da sua comparação com os resultados oriundos da aplicação de metaheurísticas bem conceituadas a esse mesmo conjunto de instâncias de teste.

PALAVRAS CHAVE: Metaheurísticas, Otimização global contínua restrita, Otimização sem derivação.
Área de classificação principal: Metaheurísticas

ABSTRACT

This paper presents a new derivative-free metaheuristic for constrained continuous global optimization called Busca Local Intensiva (Intensive Local Search), or BLI. BLI works on the problem reformulation as a bi-objective optimization problem and search for good solutions for the one making use of strategies from metaheuristics for combinatorial optimization, such as random multi-starts, sets of elite solutions and solution intensification process, that support local search cycles performed at current solution neighbourhood, that are eventually reduced. The effectiveness and efficiency of the proposed method are demonstrated by means of its application on a standard set of 13 well-known test problems and comparison with results from applications of well-respected metaheuristics on this same set of test problems.

KEYWORDS: Metaheuristics, Constrained continuous global optimization, Derivative-free optimization.
Main area: Metaheuristics

1 Introdução

O Problema de Otimização Global Contínua Restrita (POGCR), em sua forma de minimização, consiste em encontrar uma solução x^* que minimiza uma determinada função $f(x)$ em uma região $\mathcal{F}$ do espaço $\mathbb{R}^n$ definida por um conjunto de restrições. Esta região $\mathcal{F}$ é denominada região viável do POGCR. Matematicamente, podemos expressar o POGCR da seguinte forma:

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeito a:} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_j(x) = 0, \quad j = 1, \ldots, q \\ & l \leq x \leq u, \end{array} \tag{1}$$

onde x é vetor em $\mathbb{R}^n$ que contém as variáveis de decisão do problema, f, denominada função objetivo, $g_i,\ i = 1, \ldots, m$ e $h_j,\ j = 1, \ldots, q$, denominadas funções de restrição, são funções de $\mathbb{R}^n$ em $\mathbb{R}$. l e u são vetores em $\mathbb{R}^n$ que contém, respectivamente, limites reais inferiores e superiores para cada uma das variáveis de decisão. Uma solução $\bar{x} \in \mathcal{F}$ tal que $f(\bar{x}) \leq f(x)$ para qualquer solução x na vizinhança de $\bar{x}$ é dita ser uma solução ótima local. Uma solução $x^* \in \mathcal{F}$ tal que $f(x^*) \leq f(x)$ para qualquer solução $x \in \mathcal{F}$ é dita ser uma solução ótima global, e, portanto, a solução ótima do POGCR.

Os algoritmos classificados como exatos muitas vezes falham em encontrar a solução ótima de um POGCR, convergindo para soluções ótimas locais. Os algoritmos dessa classe que resolvem POGCR costumam fazer uso de informações oriundas do processo de derivação das funções, o que em alguns casos pode trazer problemas de convergência caso uma das funções não seja derivável.

Recentemente, alguns trabalhos propuseram metaheurísticas sem uso de derivação para abordar o POGCR (por exemplo, Koziel (1999), Runarsson (2000), Montes (2003), Hedar (2006) e Zhang (2008)). Metaheurísticas são procedimentos não determinísticos que fazem uso combinado de aleatoriedade e critérios gulosos para encontrar boas soluções para o problema considerado. Utilizando estratégias variadas, elas podem, ao abordar um POGCR, fugir da convergência prematura para os pontos de ótimo local muitas vezes apresentada pelos algoritmos exatos, sendo então capazes, mesmo para problemas cujas funções são não-deriváveis, de encontrar o ponto de ótimo global ou ao menos alguma solução próxima a ele. Segundo Hedar e Fukushima (Hedar (2006)),as metaheurísticas podem ser classificadas em duas categorias: métodos baseados em população, como os Algoritmos Evolucionários, e métodos de ponto-a-ponto, como os algoritmos baseados na metodologia Simulated Annealing. A maioria das metaheurísticas propostas para o POGCR são baseadas em população, uma vez que métodos desse tipo tem apresentado resultados de melhor qualidade em relação aos de ponto-a-ponto, embora necessitem de um maior esforço computacional que esses últimos.

Neste trabalho, propomos uma nova metaheurística do tipo ponto-a-ponto sem uso de derivação para resolver o POGCR denominada Busca Local Intensiva (BLI). A seção 2 deste texto apresenta os seus detalhes. Na seção 3, são mostrados resultados numéricos que indicam que esse método é bastante competitivo em comparação com os métodos já existentes com o mesmo propósito, incluindo aqueles baseados em população. A seção 4 traz a conclusão sobre o trabalho desenvolvido.

2 A metodologia BLI

2.1 Reformulação do problema

Seguindo uma idéia semelhante à utilizada pela metaheurística proposta em Hedar (2006), BLI trabalha com uma reformulação do problema (1). Através das seguintes funções de

violação de restrição

$$\begin{array}{lcll} G_i(x) & = & \gamma \ \max\{0,\ g_i(x)\}, & i = 1, \ldots, m, \\ G_{m+j}(x) & = & \gamma \ |h_j(x)|, & j = 1, \ldots, q, \end{array} \tag{2}$$

onde γ é um número real positivo denominado fator de penalização, o problema (1) é reformulado como um problema de otimização bi-objetivo:

$$\begin{array}{ll} \text{minimizar} & (f(x), \quad G(x)) \\ \text{sujeito a:} & l \leq x \leq u, \end{array} \tag{3}$$

onde $G(x) = \sum_{i=1}^{m+q} G_i(x)$. A segunda componente do vetor de função objetivo $G(x)$ é usualmente denominada termo de penalização. Na formulação (3), para uma determinada solução x que satisfaz $l \leq x \leq u$, teremos $G(x) = 0$ se x for uma solução viável, e $G(x) > 0$ caso contrário.

2.2 O algoritmo BLI

A partir da formulação (3), o algoritmo BLI, apresentado como algoritmo 1, se move através da região definida pelos vetores de limites l e u tendo como objetivo principal minimizar o termo de penalização $G(x)$ e como objetivo secundário minimizar a função $f(x)$. Devido à

```
1  |celiteViavel| ← dimCElite; |celiteInviavel| ← dimCElite;
2  celiteViavel ← ∅; celiteInviavel ← ∅;
3  para i = 1, ..., k faça
4      alcance ← 5;
5      γ ← 1;
6      h ← (u − l)/(2alcance);
7      [x, f, G] ← solucaoInicial(l, u, γ);
8      enquanto max(h) ≥ hfinalp faça
9          maxPontosSMel ←min{300 + 25n, (2alcance)^n, 1200} ;
10         [x, f, G] ←buscalocal(x, f, G, n, h, l, u,alcance,maxPontosSMel,γ);
11         se alcance ≤ 2 então
12             se x é uma solução viável então h ← h/3 senão h ← h/10;
13             alcance ← 5;
14         senão alcance ← ⌈alcance/2⌉;
15     se x é uma solução viável então
16         atualizar(cEliteViavel, x, f, G);
17     senão
18         atualizar(cEliteInviavel, x, f, G);
19 [x*, f*, G*] ← intensificar(cEliteViavel, cEliteInviavel, h, hfinali1, hfinali2);
20 retornar [x*, f*, G*];
```

Algoritmo 1: Algoritmo BLI

isso, BLI nunca se move para uma solução cujo termo de penalização $G(x)$ é maior do que o da solução corrente para um mesmo fator de penalização γ. BLI utiliza como referência um vetor h, inicializado na linha 6, que contém os valores correntes para os tamanhos dos passos para cada uma das variáveis x_i. Cada elemento h_i é proporcional ao intervalo definido por $[l_i \quad u_i]$.

BLI parte de k soluções iniciais aleatórias na região definida pelos vetores de limites l e u (linha 7). Cada uma dessas k soluções é submetida a um ciclo (linhas 8-14) onde são realizadas buscas locais (linha 10) em vizinhanças cada vez menores em torno da solução corrente. Estas vizinhanças são definidas a partir do parâmetro *alcance* e do vetor h. À medida que este ciclo de buscas locais progride, os valores dos tamanhos dos passos em h vão sendo decrementados através de um fator de decremento (linha 12). Visando acelerar a busca por soluções viáveis, BLI utiliza um fator de decremento maior enquanto se move por fora da região viável em relação ao utilizado enquanto se move por dentro desta. O ciclo de busca local termina quando o maior passo em h é menor que o parâmetro $hfinalp$.

As melhores dessas k soluções encontradas são armazenadas para servirem de entrada a um procedimento de intensificação. Soluções viáveis e inviáveis são mantidas em conjuntos-elite distintos cujo critério de avaliação em ambos é apenas o valor da função objetivo $f(x)$.

2.3 O procedimento de busca local

O procedimento de busca local recebe como entrada uma solução x, os vetores l e u, o valor $maxPontosSMel$, o vetor de passos corrente h e o valor *alcance* para obter, aleatoriamente, soluções em uma vizinhança da solução corrente, denotada por x^b. Esta vizinhança é definida por:

$$\{x \in \mathbb{R}^n : \alpha \leq x \leq \beta\}, \tag{4}$$

onde α e β são vetores em $\mathbb{R}^n$ tais que cada componente α_i e β_i são definidas como:

$$\begin{array}{rcll} \alpha_i & = & max\{x_i^b - (alcance)h_i, & l_i\}, \quad i = 1, \ldots, n \\ \beta_i & = & min\{x_i^b + (alcance)h_i, & u_i\}, \quad i = 1, \ldots, n \end{array} \tag{5}$$

Inicialmente, o procedimento toma x^b como sendo a solução de entrada x. O procedimento também mantém um contador κ que conta o número de soluções obtidas em uma vizinhança sem encontrar uma solução de melhora.

Quando a solução corrente x^b é viável, o procedimento de busca local se move para uma outra solução x^a se, e somente se, $G(x^a) = 0$ e $f(x^a) < f(x^b)$. Para trazer ganhos no que diz respeito à eficiência, inicialmente apenas o valor de $f(x^a)$ é calculado. Se $f(x^a) \geq f(x^b)$, a solução x^a já é descartada sem ter o seu valor $G(x^a)$ calculado. Caso contrário, é verificado se a condição $G(x^a) = 0$ é satisfeita. Em caso afirmativo, o contador κ é zerado e a solução corrente x^b passa a ser a solução x^a, e em caso negativo, o contador κ é incrementado.

Quando a solução corrente x^b é inviável, o procedimento de busca local procura por soluções visando minimizar o termo de penalização $G(x)$ e tentando não perder qualidade com respeito ao valor da função $f(x)$. Para isso, são mantidos um contador ω e uma solução $\widehat{x}$, tal que $G(\widehat{x}) < G(x^b)$ e $f(\widehat{x}) > f(x^b)$. $\widehat{x}$ é definida como sendo a melhor solução, com respeito ao valor de $f(x)$, sorteada na vizinhança de x^b cujo valor do termo de penalização $G(x)$ é menor do que o de x^b. ω é definido como sendo o contador de soluções sorteadas na vizinhança de x^b cujo valor do termo de penalização $G(x)$ é menor do que o de x^b.

Para ilustrar como o procedimento se move por fora da região viável, considere o seguinte cenário: seja x^b uma solução corrente inviável e x^a a solução na vizinhança de x^b sorteada na iteração corrente. A solução x^a será descartada se $G(x^a) > G(x^b)$, ou se $G(x^a) = G(x^b) \wedge f(x^a) \geq f(x^b)$. Caso contrário, os valores da função objetivo nas duas soluções serão comparados, o contador ω será incrementado e o contador κ será zerado. Se $f(x^a) < f(x^b)$, o procedimento se moverá imediatamente para x^a atribuindo-a à x^b. Caso contrário, os valores da função objetivo em x^a e $\widehat{x}$ serão comparados. Se $f(x^a) < f(\widehat{x})$, ou se $f(x^a) = f(\widehat{x}) \wedge G(x^a) < G(\widehat{x})$, a solução x^a é atribuída à $\widehat{x}$. Quando o contador ω atinge o valor do parâmetro definido pelo usuário $maxInvMel$, o procedimento de busca local se move para

a solução $\widehat{x}$ atribuindo-a à solução corrente x^b. Para trazer ganhos com relação à eficiência, o valor $f(x^a)$ só é calculado se $G(x^a) \leq G(x^b)$.

Ao se mover para uma solução x^a, sendo ela viável ou não, o algoritmo a atribui à solução corrente x^b, zera os contadores κ e ω, atribui o valor $\emptyset$ à $\widehat{x}$ e passa a buscar soluções na região definida por (4) e (5). O procedimento para, retornando a solução x^b, quando o contador κ ficar maior do que o valor $maxPontosSMel$, onde $maxPontosSMel$ é um parâmetro de entrada da busca calculado na linha 9 do algoritmo 1.

Visando uma exploração mais eficiente nas vizinhanças, são utilizadas duas estratégias especiais denominadas teste de extensão do passo e teste de reflexão. O teste de extensão do passo ocorre quando é obtida, na vizinhança da solução corrente x^b, uma solução x^a tal que $f(x^a) < f(x^b) \wedge G(x^a) \leq G(x^b)$. Sob esse cenário, seja p o vetor em $\mathbb{R}^n$ dado por:

$$p = x^a - x^b, \tag{6}$$

e x^e, a solução dada por:

$$x^e = x^a + p.$$

Sob estas condições, x^a passará a ser a solução corrente. Se x^e estiver contida na vizinhança de x^a, ou seja, dentro dos limites definidos por (4) e (5) para $x^b = x^a$, então x^e será a próxima solução avaliada. Note que se $l \leq x^e \leq u$, então x^e estará contida na vizinhança da solução x^a. Repare também que x^e é obtida pela extensão do passo dado de x^b para x^a. Devido à isso, a esta avaliação de x^e damos o nome de teste de extensão do passo. É importante frisar que, se $f(x^e) < f(x^a) \wedge G(x^e) \leq G(x^a)$, então um novo teste de extensão do passo será executado tendo x^e como solução corrente. Em outras palavras, o teste de extensão do passo é executado de forma sucessiva até que suas condições de execução não mais sejam verificadas.

O teste de reflexão ocorre quando é encontrada, na vizinhança da solução corrente x^b, uma solução x^a tal que $G(x^a) > G(x^b)$ ou $G(x^a) = G(x^b) \wedge f(x^a) \geq f(x^b)$ e x^a não tenha sido obtida através dos testes de extensão do passo ou de reflexão. Nesse cenário, x^b continuará sendo a solução corrente. Seja p o vetor em $\mathbb{R}^n$ definido por (6) e x^r a solução dada por:

$$x^r = x^b - p.$$

Para esse caso específico, se o valor do contador κ for menor do que o valor $maxPontosSMel$ e x^r estiver contida na vizinhança de x^b, ou seja, dentro dos limites definidos por (4) e (5), então x^r será a próxima solução a ser avaliada. Note que se $l \leq x^r \leq u$, então x^r estará contida na vizinhança de x^b. Repare também que x^r é obtida pela reflexão de x^a em relação ao ponto x^b. Devido à isso, denominamos a avaliação de x^r como teste de reflexão.

2.4 O procedimento de intensificação

O procedimento de intensificação consiste, primeiramente, na execução de um novo ciclo de buscas locais semelhante ao definido nas linhas 8-14 do algoritmo 1 para a solução sob intensificação corrente, definida aqui como x^i. Esse novo ciclo de buscas locais inicia com os valores em h fornecidos como entrada e termina quando a maior componente em h for menor que o valor $hfinali1$. Posteriormente, x^i é submetida à uma execução do algoritmo de Nelder-Mead proposto em Nelder (1965). Esse algoritmo, que também foi utilizado para intensificar soluções em Hedar (2006), recebe como entrada um conjunto de $n+1$ soluções denominado conjunto simplex e, através de movimentos simples, busca por soluções melhores sem utilizar informação de derivada. Para compor o conjunto Simplex, são utilizadas n soluções equidistantes de x^i, que podem ser viáveis ou não. BLI utiliza como função objetivo para o algoritmo de Nelder-Mead a função $f_p(x)$ definida pela equação:

$$f_p(x) = f(x) + \lambda(\, G(x) + max\{0,\ l - x\} + max\{0,\ x - u\}\,)$$

onde λ é número real grande que evite movimentos para fora da região viável (por exemplo, $1.0e8$). Neste trabalho, utilizamos a implementação do método de Nelder-Mead disponível em Mathews (1992). Após a aplicação do algoritmo de Nelder-Mead, se a solução sob intensificação x^i for inviável, um novo ciclo de buscas locais é executado, agora tendo como critério de parada a maior componente do vetor h ser menor que o valor $hfinali2$, e uma nova execução do algoritmo de Nelder-Mead é feita após esse novo ciclo.

O procedimento de intensificação é executado primeiramente sobre a melhor solução disponível. Após essa execução, um número de $numIntensifExt$ execuções são realizadas sobre outras soluções dos conjuntos elites tomadas de modo aleatório. Para a escolha da solução a ser intensificada, é dada preferência às soluções viáveis em detrimento das inviáveis. A melhor dessas $numItensifExt+1$ soluções intensificadas é retornada pelo procedimento.

3 Resultados Numéricos

O algoritmo BLI foi aplicado sobre um conjunto de 13 instâncias de testes bem conhecidas na literatura. Para uma lista completa dessas instâncias de teste, consulte Montes (2003). A tabela 1 apresenta os valores dos parâmetros de BLI utilizados para a execução dos testes.

Parâmetro	Valor	Descrição
k	10	Número de vezes em que uma solução aleatória é gerada e submetida ao ciclo de buscas locais.
$hfinalp$	10^{-2}	Define o critério de parada do ciclo de buscas locais no núcleo de BLI.
$hfinali1$	10^{-3}	Define o critério de parada do primeiro ciclo de buscas locais na etapa de intensificação.
$hfinali2$	10^{-5}	Define o critério de parada do segundo ciclo de buscas locais, caso haja, na etapa de intensificação.
$dimCElite$	3	Número máximo de soluções incluídas em cada conjunto-elite.
$numIntensifExt$	1	Número de soluções intensificadas além da melhor solução.
$maxInvMel$	5	Número máximo de soluções obtidas em uma determinada vizinhança cujo termo de penalização possui valor menor do que o da solução corrente no procedimento de busca local.

Tabela 1: Valores dos parâmetros de BLI utilizados nesse trabalho

BLI foi comparada com as seguintes metaheurísticas bem conceituadas para o POGCR:

- Simple Multimembered Evolution Strategy (SMES), Montes (2003)
- Filter Simulated Annealing (FSA), Hedar (2006)
- Dynamic Stochastic Selection-Multimember Differential Evolution, segundo experimento (DSS-MDE), Zhang (2008).

O método FSA, assim como BLI, é de ponto-a-ponto, ao passo que SMES e DSS-MDE são baseados em população. Para BLI, SMES e FSA, foram feitas 30 execuções independentes para cada instância de teste, ao passo que para a DSS-MDE foram realizadas 100 execuções independentes. Para lidar com restrições de igualdade, BLI adotou, assim como todos os demais métodos, uma tolerância da ordem de 10^{-4} para a não-satisfação destas restrições.

Na tabela 2, são apresentados os resultados numéricos fornecidos pelas abordagens para as instâncias de teste. Os resultados referentes a SMES, FSA e DSS-MDE foram retirados de suas respectivas referências, enquanto os resultados referentes ao método BLI foram obtidos sobre uma implementação feita no ambiente MATLAB 7.5 em uma estação Windows XP. A primeira coluna traz o identificador da instância, o tipo da função objetivo e seu respectivo valor ótimo conhecido. Na segunda coluna é discriminada a abordagem, seguida dos valores

da função objetivo para a melhor solução, solução média e pior solução, do desvio padrão, da média de avaliações da função objetivo e da média de avaliações das funções de restrição considerando todas as execuções feitas em cada abordagem para cada instância.

Problema	Abordagem	Melhor	Media	Pior	DP	F-avals	R-avals
G1	SMES	-15	-15	-15	0,00000	240.000	240.000
(min)	FSA	-14,999105	-14,993316	-14,979977	0,00481	205.748	87.701
-15	DSS-MDE	-15	-15	-15	0,00000	350.000	350.000
	BLI	-14,993115	-14,966125	-14,861771	0,02554	167.870	37.431
G2	SMES	0,803601	0,785238	0,751322	0,01676	240.000	240.000
(max)	FSA	0,754913	0,371708	0,271311	0,09802	227.832	101.903
0,803619	DSS-MDE	0,803619	0,788011	0,744690	0,01500	350.000	350.000
	BLI	0,786136	0,624451	0,470020	0,07409	267.811	23.283
G3*	SMES	1,00104	1,00099	1,00058	0,00021	240.000	240.000
(max)	FSA	1,00000	0,99919	0,99152	0,00165	314.938	118.404
1	DSS-MDE	1,00050	1,00050	1,00050	0,00000	350.000	350.000
	BLI	1,00049	0,99927	0,99688	0,00081	89.144	176.853
G4	SMES	-30665,539	-30665,539	-30665,539	0,00000	240.000	240.000
(min)	FSA	-30665,538	-30665,467	-30664,688	0,17322	86.154	37.000
30665,539	DSS-MDE	-30665,539	-30665,539	-30665,539	0,00000	350.000	350.000
	BLI	-30665,469	-30665,400	-30665,322	0,03776	119.972	53.605
G5*	SMES	5126,5996	5174,4923	5304,1670	50,0579	240.000	240.000
(min)	FSA	5126,4981	5126,4981	5126,4981	0,00000	47.661	17.757
5126,4981	DSS-MDE	5126,4970	5126,4970	5126,4970	0,00000	350.000	350.000
	BLI	5126,4963	5126,7328	5127,8049	0,36410	3.798	210.768
G6	SMES	-6961,814	-6961,284	-6952,482	1,86114	240.000	240.000
(min)	FSA	-6961,814	-6961,814	-6961,814	0,00000	44.538	15.817
-6961,814	DSS-MDE	-6961,814	-6961,814	-6961,814	0,00000	350.000	350.000
	BLI	-6961,814	-6961,814	-6961,814	0,00000	8.620	13.585
G7	SMES	24,32672	24,47493	24,84283	0,13239	240.000	240.000
(min)	FSA	24,31057	24,37953	24,64440	0,07164	404.501	171.299
24,3062091	DSS-MDE	24,306	24,306	24,306	0,00000	350.000	350.000
	BLI	24,31013	24,50311	24,75874	0,10153	124.687	95.338
G8	SMES	0,095826	0,095826	0,095826	0,00000	240.000	240.000
(max)	FSA	0,095825	0,095825	0,095825	0,00000	56.476	23.219
0,095825	DSS-MDE	0,095825	0,095825	0,095825	0,00000	350.000	350.000
	BLI	0,095825	0,095825	0,095825	0,00000	10.731	2.467
G9	SMES	680,63159	680,64341	680,71930	0,01553	240.000	240.000
(min)	FSA	680,63008	680,63642	680,69832	0,01452	324.569	147.035
680,630057	DSS-MDE	680,630	680,630	680,630	0,00000	350.000	350.000
	BLI	680,63016	680,65581	680,76712	0,03223	128.022	49.890
G10	SMES	7051,9028	7253,0470	7638,3662	136,024	240.000	240.000
(min)	FSA	7059,8635	7509,3210	9398,6492	542,342	243.520	93.667
7049,3307	DSS-MDE	7049,248	7049,248	7049,249	0,00031	350.000	350.000
	BLI	7057,0090	7131,9114	7453,7413	91,8994	173.709	234.256
G11*	SMES	0,7491	0,7494	0,7498	0,00015	240.000	240.000
(min)	FSA	0,7500	0,7500	0,7500	0,00000	23.722	8.485
0,75	DSS-MDE	0,7499	0,7499	0,7499	0,00000	350.000	350.000
	BLI	0,7499	0,7500	0,7539	0,00073	1.070	7.604
G12	SMES	-1,0000	-1,0000	-1,0000	0,00000	240.000	240.000
(min)	FSA	-1,0000	-1,0000	-1,0000	0,00000	59.355	25.818
-1	DSS-MDE	-1,0000	-1,0000	-1,0000	0,00000	350.000	350.000
	BLI	-1,0000	-1,0000	-1,0000	0,00000	51.680	1.790
G13*	SMES	0,0540	0,1664	0,4683	0,17669	240.000	240.000
(min)	FSA	0,0539	0,2977	0,4389	0,18865	120.268	42.268
0,0539498	DSS-MDE	0,0539	0,0539	0,0539	0,00000	350.000	350.000
	BLI	0,0539	0,0566	0,0768	0,00479	1.487	67.548

* Problemas que possuem restrições de igualdade

Tabela 2: Resultados numéricos de SMES, FSA, DSS-MDE e BLI

Considerando a solução média fornecida pelas metodologias em termos da função objetivo e da quantidade média de avaliações de função executas pelas mesmas, podemos apontar que BLI apresentou uma abordagem melhor do que SMES sobre 7 das 13 instâncias (G5, G6, G8, G10, G11, G12, G13). Em comparação com FSA, sob os mesmos critérios, BLI apresentou melhor abordagem sobre 8 das 13 instâncias (G2, G3, G6, G8, G10, G11, G12, G13). Em comparação com DSS-MDE, BLI apresentou melhor abordagem sobre apenas 4 das 13 intâncias (G6, G8, G11, G12). No entanto, BLI apresentou um custo computacional bastante inferior a DSS-MDE uma vez que a quantidade de avaliações das funções objetivo e de restrição realizada é consideravelmente menor para todas as instâncias do conjunto de teste. Sob apenas essa ótica do custo computacional, BLI forneceu resultados melhores do que SMES para todas as instâncias de teste, e resultados melhores do que FSA para 10 das 13 instâncias de teste (G1, G2, G3, G6, G7, G8, G9, G11, G12, G13).

4 Conclusão

Uma nova metaheurística do tipo ponto-a-ponto sem uso de derivação denominada Busca Local Intensiva (BLI) foi proposta para otimização global contínua restrita neste trabalho. BLI faz uso de estratégias lançadas por metaheurísticas de otimização combinatória, tais como multi-inícios aleatórios, conjuntos elite e procedimento de intensificação de soluções, que dão apoio a ciclos de buscas locais executadas em vizinhanças cada vez menores em torno da solução corrente. BLI demonstrou, através da aplicação sobre um conjunto de 13 instâncias de teste bem conhecidas, que é um método bastante competitivo em comparação com as metaheurísticas bem conceituadas existentes com esse mesmo propósito, fornecendo resultados de boa qualidade a um custo computacional médio consideravelmente inferior aos das demais, fortalecendo assim a classe de metaheurísticas do tipo ponto-a-ponto frente à classe daquelas baseadas em população.

Referências

A. Hedar e M. Fukushima (2006), Derivative-free filter simulated annealing method for constrained continuous global optimization, Journal of Global Optimization 35, pp. 521-549.

S. Koziel e Z. Michalewicz (1999), Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evolutionary Computation 7(1), pp. 19-44.

J. H. Mathews (1992), Numerical Methods for Mathematics, Science and Engineering, 2ª Ed.

E. M. Montes e C. A. Coello Coello (2003), A simple multimembered evolution strategy to solve constrained optimization problems, Technical Report EVOCINV-04-2003, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México D.F., México.

J. A. Nelder e R. Mead (1965), A simplex method for function minimization, The Computer Journal 7, pp. 308-313.

T. P. Runarsson e X. Yao (2000), Stochastic Ranking for Constrained Evolutionary Optimization, IEEE Transactions on Evolutionary Computation 4(3), pp. 284-294.

M. Zhang, W. Luo, X. Wang (2008), Differential evolution with dynamic stochastic selection for constrained optimization, Information Sciences 178, pp. 3043-3074.