ALGORITMO DE BUSCA DISPERSA PARA OTIMIZAÇÃO DE FUNÇÕES CONTÍNUAS MULTIMODAIS COM RESTRIÇÕES

André Mendes Garcia
andre@fai.com.br

Leonardo H. Macedo
leohfmp@ieee.org

Rubén Romero
ruben@dee.feis.unesp.br

Laboratório de Planejamento de Sistemas de Energia Elétrica – LaPSEE
Departamento de Engenharia Elétrica – UNESP – Ilha Solteira
Avenida Brasil, 56 – Centro
15.385-000 ILHA SOLTEIRA, SP, BRASIL

RESUMO

Este artigo apresenta o método da Busca Dispersa aplicado na otimização de funções numéricas contínuas multimodais com restrições. O método da Busca Dispersa é uma meta-heurística que se difere de outros algoritmos evolutivos, pois utiliza estratégias diferentes da aleatoriedade para explorar o espaço de busca do problema. Um algoritmo baseado em sensibilidades da função objetivo e o algoritmo Simplex de Nelder-Mead são utilizados no passo da melhoria local do método de Busca Dispersa. Neste trabalho o método é aplicado em sete problemas testes conhecidos da literatura especializada. Os resultados dos testes comparados com trabalhos de outros autores, que utilizaram outras estratégias e métodos para resolver esses mesmos problemas, mostraram-se semelhantes e em alguns casos discretamente melhores.

PALAVRAS CHAVE. Busca dispersa. Funções multimodais restritas. Meta-heurística. Otimização.

Área Principal: MH – Meta-heurísticas.

ABSTRACT

This paper presents the Scatter Search method applied to the optimization of multimodal continuous constrained numerical functions. The Scatter Search method is a metaheuristic that differs from other evolutionary algorithms because it uses strategies different from the randomization to explore the search space of a problem. An algorithm based on the sensitivities of the objective function and the Nelder-Mead Simplex algorithm are used in the local improvement step of the Scatter Search method. In this work the method is applied to seven test problems available in the specialized literature. The results of the tests, compared with the work of other authors, that used other strategies and methods to solve these same problems, were similar and in some cases discreetly better.

KEYWORDS. Metaheuristic. Optimization. Restricted multimodal functions. Scatter search.

Main Area: MH – Metaheuristics.

https://proceedings.science/p/85380?lang=pt-br

1. Introdução

O algoritmo de Busca Dispersa (BD) é um método evolutivo que tem sido amplamente utilizado em problemas de otimização. Os primeiros conceitos e princípios fundamentais do método foram propostos na década de 1970 e baseados em formulações anteriores à década de 1960, que combinavam regras de decisão a problemas restritos. O método utiliza estratégias de intensificação e diversificação de busca, e tem demonstrado ser eficaz em vários problemas de otimização (LAGUNA E MARTÍ, 2003).

Neste artigo o algoritmo de Busca Dispersa é implementado para otimizar funções contínuas multimodais com restrições, da forma geral apresentada nas equações (1)–(4).

$$\min f(\mathbf{x}) \tag{1}$$

Sujeito a:

$$g_i(\mathbf{x}) \leq 0 \qquad i = 1, 2, \ldots, m \tag{2}$$

$$h_i(\mathbf{x}) = 0 \qquad i = 1, 2, \ldots, l \tag{3}$$

$$a_i \leq x_i \leq b_i \qquad i = 1, 2, \ldots, n \tag{4}$$

Onde $f(\mathbf{x})$, $g_i(\mathbf{x})$, $h_i(\mathbf{x})$ são funções contínuas lineares e/ou não lineares definidas em R^n e $\mathbf{x} \in R^n$ é um vetor contendo as variáveis do problema. Os limites de cada variável são definidos pelos vetores $\mathbf{a}$ e $\mathbf{b}$.

Os problemas multimodais apresentam múltiplas soluções ótimas, sendo algumas, ou apenas uma ótima global. Essa característica desse tipo de problema dificulta a resolução, principalmente por métodos exatos que são altamente dependentes do ponto inicial.

Vários trabalhos disponíveis na literatura resolvem problemas desse tipo, de diversas formas, incluindo métodos exatos, heurísticas e meta-heurísticas. Koziel e Michalewicz (1999) apresentam uma estratégia para mapear o espaço de busca de problemas numéricos e restritos para ser utilizada em algoritmos evolutivos. A proposta é testada em diversos problemas, apresentando bons resultados. Runarsson e Yao (2000) propõem uma nova forma de se incorporar penalidades na função de avaliação deste tipo de problema. Os resultados demonstram que a proposta apresenta um melhor desempenho do que utilizar métodos de penalização tradicionais. Farmani e Wright (2003) apresentam uma proposta de *fitness* auto adaptativo para ser utilizado em algoritmos evolutivos. Venkatraman e Yen (2005) propõem uma estrutura genérica de duas fases para resolver problemas restritos com algoritmos genéricos, com boa performance independente das características do problema.

O principal motivo de se escolher o algoritmo de Busca Dispersa para resolver esse tipo de problemas é que este método obteve excelentes resultados para vários problemas disponíveis na literatura. Pretende-se, portanto, resolver esses problemas gerais e depois adaptar o algoritmo desenvolvido para resolver problemas na área de sistemas de energia elétrica, onde esse tipo de algoritmo ainda não foi aplicado.

2. Busca Dispersa para Otimização de Funções Contínuas Multimodais com Restrições

A Figura 1 ilustra o fluxograma do algoritmo de Busca Dispersa para otimização de funções contínuas multimodais com restrições (BD-FMR) proposto. Como pode ser observado, o algoritmo de Busca Dispersa utilizado neste trabalho possui quatro etapas principais: (i) construção do conjunto P; (ii) preenchimento inicial do conjunto $RefSet$; (iii) geração dos subconjuntos S e $Pool$; e (iv) atualização de $RefSet$.

https://proceedings.science/p/85380?lang=pt-br

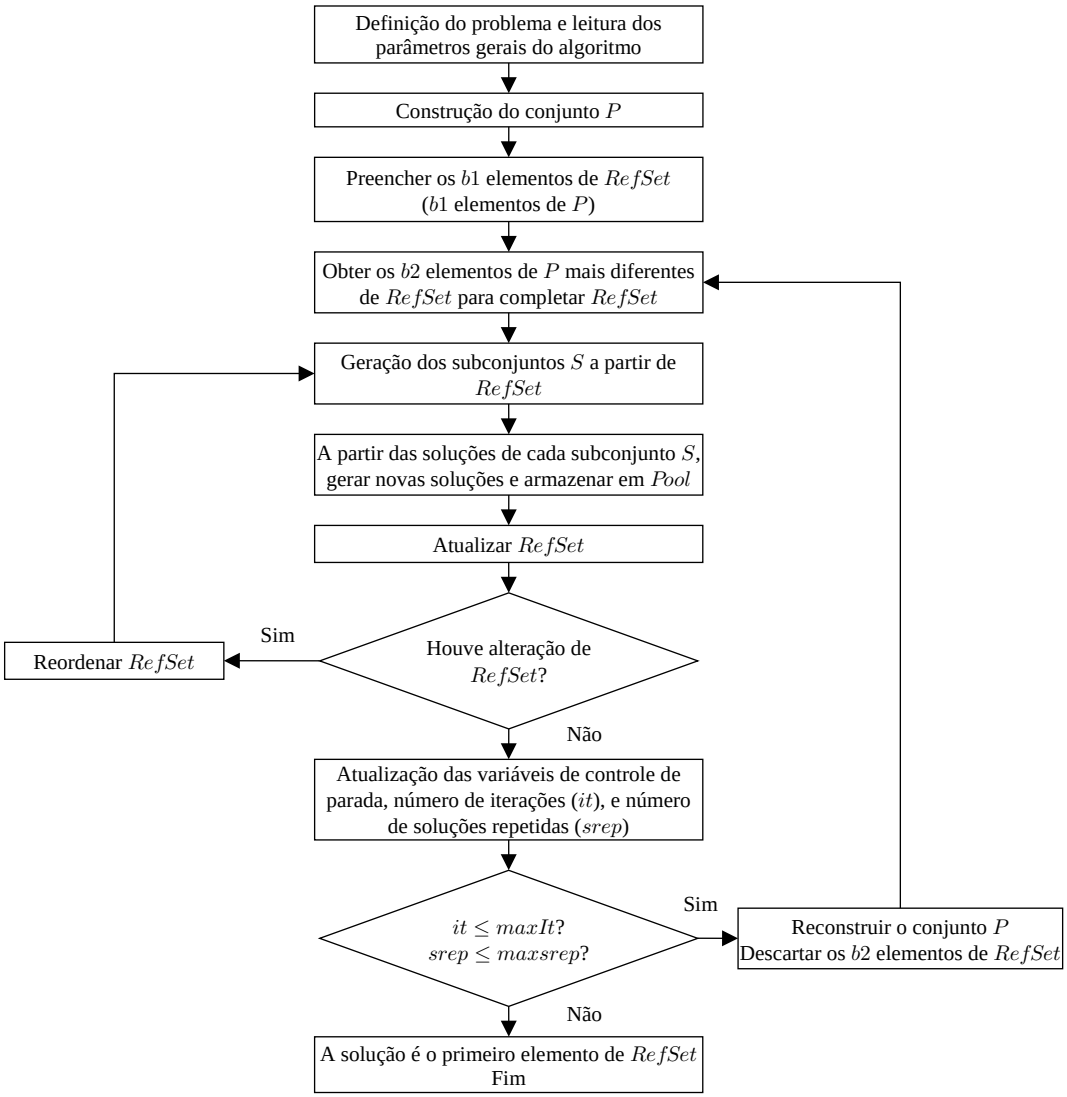


Figura 2: Fluxograma geral do algoritmo BD-FMR.

A primeira etapa do algoritmo consiste na definição da função numérica e suas restrições. Nesta etapa são determinados o número de variáveis da função e restrições, seus limites, a expressão da função objetivo e as expressões das restrições. Também nesta etapa inicial alguns parâmetros importantes para o funcionamento do método são definidos, tais como $PSize$ que define o tamanho do conjunto P de soluções, $b1$ o número de elementos do conjunto $RefSet$ responsáveis por representar soluções de qualidade, $b2$ o número de elementos do conjunto $RefSet$ responsáveis pela diversidade, e $b = b1 + b2$ sendo o tamanho do conjunto $RefSet$ de soluções.

Após a etapa inicial, o conjunto P de soluções é construído, e com base nele o conjunto $RefSet$ inicial também é obtido. Os subconjuntos S_i de soluções são formados a partir dos elementos de $RefSet$ para a montagem do conjunto de soluções $trial$ (testes), chamado de $Pool$. Para cada solução do conjunto $Pool$, um método de melhoria local é aplicado.

O conjunto $RefSet$ é atualizado com as b melhores soluções entre os conjuntos $RefSet$ e $Pool$. Se houver atualização do conjunto $RefSet$, o fluxo do algoritmo é desviado novamente para a construção de um novo subconjunto S. Caso não haja atualização do conjunto $RefSet$, um novo conjunto P de soluções é construído, e depois são geradas novas soluções preservando as $b1$ soluções de $RefSet$ e substituindo as $b2$ soluções usando o novo conjunto P.

https://proceedings.science/p/85380?lang=pt-br

O processo é encerrado a partir de um número máximo de iterações ou a partir do momento em que um determinado número de soluções iguais foi obtido sequencialmente sem melhoria. A solução final, é o primeiro elemento do conjunto $RefSet$, pois quando $RefSet$ sofre alteração, ele é sempre reordenado em ordem da melhor solução para a pior. Nas próximas subseções, as principais etapas do algoritmo são descritas detalhadamente.

2.1 Construção do Conjunto P

Na maioria dos algoritmos de BD o tamanho do conjunto P ($PSize$) de soluções é considerado como o máximo entre 100 e $5 \times b$, onde b é o tamanho do conjunto $RefSet$ (LAGUNA E MARTÍ, 2003). Neste trabalho, o tamanho $PSize$ foi considerado 100 para a maioria dos casos testes, entretanto para um melhor resultado foram escolhidos outros tamanhos em alguns dos testes.

A construção do conjunto P de soluções deve levar em conta a diversidade das soluções e não apenas a qualidade. Para isso, neste trabalho a geração dos elementos do conjunto P emprega randomização controlada utilizando memória baseada em frequência. Nesta técnica, para cada variável do problema são consideradas nr sub-faixas de igual tamanho entre seu limite inferior e superior, onde nr foi considerado igual a quatro neste trabalho. Assim, o valor de cada variável de uma solução do problema é obtido primeiramente selecionando-se uma sub-faixa, e em seguida um valor aleatório dentro desta sub-faixa. A seleção da sub-faixa na primeira solução é totalmente aleatória. A partir da segunda solução, a probabilidade da seleção de uma sub-faixa é inversamente proporcional à frequência em que ela já foi selecionada em soluções anteriores. A equação (5) mostra a expressão para obter esta probabilidade.

$$\operatorname{prob} F_i(k) = \frac{(k-1) - \operatorname{freq} F_i(k-1)}{\sum_{j=1}^{nr=4}[(k-1) - \operatorname{freq} F_j(k-1)]} \tag{5}$$

Onde $\operatorname{prob} F_i(k)$ é a probabilidade da sub-faixa F_i ser selecionada, para $i = 1,2,\ldots,nr = 4$, k consiste no número da solução atual, $\operatorname{freq} F_i$ corresponde a frequência com que a sub-faixa F_i foi utilizada, e nr é número de sub-faixas.

2.2 Conjunto $RefSet$ Inicial

Todos os elementos de $RefSet$ são extraídos do conjunto P, sendo que a porção $b1$ é formada pelos $b1$ elementos de melhor qualidade do conjunto P, e a porção $b2$ é formada pelos elementos do conjunto P mais diversos em relação aos já alocados no conjunto $RefSet$.

Para obter os elementos $b2$ com diversidade, é utilizada a distância euclidiana entre os elementos já constantes do conjunto $RefSet$ e os elementos do conjunto P. Desta forma, para cada elemento constante no conjunto P, é encontrada a menor distância euclidiana, utilizando (6), entre os elementos constantes no conjunto $RefSet$. Seleciona-se, então o elemento de P que possuir a maior distância entre as mínimas.

$$\operatorname{dist}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{6}$$

2.3 Definição dos Subconjuntos S e $Pool$

Os conjuntos S são constituídos de elementos de $RefSet$, os quais serão combinados e as novas soluções geradas são armazenadas em $Pool$. Neste trabalho, os elementos de $RefSet$ são combinados de dois a dois, formando 45 subconjuntos de dois elementos cada, pois $RefSet$ possui

https://proceedings.science/p/85380?lang=pt-br

dez elementos. Para generalizar, o número de elementos de $Pool$ é dado por $C_{b,2} = b(b-1)/2$, onde b é o número de elementos de $RefSet$.

Nesta etapa do algoritmo, várias técnicas e tipos de combinação podem ser utilizadas, tais como Path Relinking, dentre outros. O método de combinação utilizado aqui é a combinação linear simples de dois elementos de S ($\mathbf{x}'$ e $\mathbf{x}''$) e que gera três soluções testes (trial), $\mathbf{c}_1 = \mathbf{x}' - \mathbf{d}$, $\mathbf{c}_2 = \mathbf{x}' + \mathbf{d}$ e $\mathbf{c}_3 = \mathbf{x}'' + \mathbf{d}$, onde $\mathbf{d} = r\frac{\mathbf{x}''-\mathbf{x}'}{2}$ corresponde a um incremento e r é um número aleatório entre zero e um.

Caso algum valor de c_i fique abaixo do limite inferior da variável, o limite inferior é assumido, e caso fique acima do limite superior da variável, o limite superior é assumido. Após obter três soluções testes, o método de melhoria local é aplicado em cada uma das soluções e essas incorporadas ao conjunto $Pool$.

2.4 Atualização do Conjunto $RefSet$

Após concluir a montagem do conjunto $Pool$, o conjunto $RefSet$ é atualizado com as b melhores soluções da união dos dois conjuntos: $Pool \cup RefSet$.

2.5 Método de Melhoria Local

O método de melhoria local é utilizado, na maioria dos algoritmos de Busca Dispersa, em duas fases do método: na construção dos conjuntos P e $Pool$. Dependendo do número de variáveis e consequentemente visando um desempenho melhor do método, pode-se utilizar a melhoria local com menos frequência, ou seja, apenas na construção do conjunto P, ou apenas na construção do conjunto $Pool$, ou em alguma etapa do método que seja executada poucas vezes, como por exemplo, aplicando o método de melhoria local no conjunto $RefSet$ após sua atualização.

Existe uma grande variedade de opções de técnicas de melhoria local, e claro, com resultados dependentes do problema em questão. Pode-se utilizar nesta etapa, por exemplo, outras metaheurísticas, tais como Otimização por Exame de Partículas (KENNEDY E EBERHART, 1995), Busca em Vizinhança Variável (MLADENOVIĆ E HANSEN, 1997), dentre outras.

Neste trabalho, duas técnicas de melhoria local são consideradas. A primeira técnica é baseada na sensibilidade da função objetivo. Considerando uma determinada solução, este método de melhoria local consiste primeiramente em acrescentar um pequeno valor d à primeira variável da função, caso a qualidade da função objetivo melhore, a variável é incrementada até seu limite superior ou até o momento em que a função objetivo não melhore mais. Caso a função objetivo não melhore na primeira tentativa, o procedimento é feito no sentido inverso, o pequeno valor d é subtraído da variável e caso a função objetivo melhore, a variável é decrementada até seu limite inferior ou até que a função objetivo não melhore mais. Este procedimento é aplicado em todas as variáveis do problema. A escolha do valor de d é feita realizando testes em cada função a ser otimizada pelo método. Geralmente o valor inicial é $d = 0{,}01$.

A segunda técnica de melhoria local empregada é o algoritmo Simplex de Nelder-Mead (LAGARIAS *et al.*, 1998), onde inicialmente quatro parâmetros são especificados: $\alpha > 0$, $\chi > 1$, $\chi > \alpha$, $0 < \gamma < 1$ e $0 < \sigma < 1$. Na maioria dos casos, os valores escolhidos são: $\alpha = 1$, $\chi = 2$, $\gamma = 1/2$ e $\sigma = 1/2$. O passo inicial do método é gerar $n+1$ soluções (vértices) para o problema que formam um simplex não degenerado.

Os passos do algoritmo, em cada iteração, são os seguintes (LAGARIAS *et al.*, 1998):

1. Ordene os $n+1$ vértices para satisfazer $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_{n+1})$;
2. Calcular o ponto refletido $\mathbf{x}_r$ usando (7):

$$\mathbf{x}_r = \overline{\mathbf{x}} + \alpha(\overline{\mathbf{x}} - \mathbf{x}_{n+1}) \tag{7}$$

https://proceedings.science/p/85380?lang=pt-br

onde $\overline{\mathbf{x}} = \sum_{i=1}^{n} \mathbf{x}_i / n$ é o centroide dos n melhores pontos. Calcular $f(\mathbf{x}_r)$. Se $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) \leq f(\mathbf{x}_n)$, aceite o ponto refletido $\mathbf{x}_r$ (substitua $\mathbf{x}_{n+1}$ por $\mathbf{x}_r$) e termine a iteração (voltar ao passo 1).

3. Se $f(\mathbf{x}_r) < f(\mathbf{x}_1)$, calcule o ponto expandido $\mathbf{x}_e$ usando (8):

$$\mathbf{x}_e = \overline{\mathbf{x}} + \chi(\mathbf{x}_r - \overline{\mathbf{x}}) \tag{8}$$

Calcular $f(\mathbf{x}_e)$. Se $f(\mathbf{x}_e) < f(\mathbf{x}_r)$, aceite $\mathbf{x}_e$ e termine a iteração; caso contrário, aceite $\mathbf{x}_r$ e termine a iteração.

4. Se $f(\mathbf{x}_r) \geq f(\mathbf{x}_n)$, faça uma contração entre $\overline{\mathbf{x}}$ e o melhor entre $\mathbf{x}_{n+1}$ e $\mathbf{x}_r$.
   a. Se $f(\mathbf{x}_n) \leq f(\mathbf{x}_r) < f(\mathbf{x}_{n+1})$, calcule o ponto contraído para fora $\mathbf{x}_c$ usando (9):

$$\mathbf{x}_c = \overline{\mathbf{x}} + \gamma(\mathbf{x}_r - \overline{\mathbf{x}}) \tag{9}$$

   e calcule $f(\mathbf{x}_c)$. Se $f(\mathbf{x}_c) \leq f(\mathbf{x}_r)$, aceite $\mathbf{x}_c$ e termine a iteração; caso contrário, vá ao passo 5.
   b. Se $f(\mathbf{x}_r) \geq f(\mathbf{x}_{n+1})$, calcule o ponto contraído para dentro $\mathbf{x}_{cc}$ usando (10):

$$\mathbf{x}_{cc} = \overline{\mathbf{x}} - \gamma(\overline{\mathbf{x}} - \mathbf{x}_{n+1}) \tag{10}$$

   e calcule $f(\mathbf{x}_{cc})$. Se $f(\mathbf{x}_{cc}) \leq f(\mathbf{x}_{n+1})$, aceite $\mathbf{x}_{cc}$ e termine a iteração; caso contrário, vá ao passo 5.
5. Calcule f nos n pontos $\mathbf{v}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_i - \mathbf{x}_1)$, para $i = 2, \ldots, n+1$. Os vértices do simplex na próxima iteração são $\mathbf{x}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n+1}$.

Como o algoritmo Simplex de Nelder-Mead foi projetado para encontrar a solução de problemas irrestritos, neste trabalho, caso uma variável ultrapasse algum limite, o seu valor é fixado nesse limite. Além disso, a solução em que se deseja realizar a busca local é incluída no simplex inicial.

2.6 Manipulação das Restrições

Existem vários métodos para se tratar problemas de otimização com restrições, e estes podem ser categorizados em métodos baseados em funções de penalização, métodos baseados na preferência de soluções factíveis sobre as infactíveis e métodos baseados em otimização multiobjetivo (COELLO, 2002).

Na implementação do algoritmo de Busca Dispersa, optou-se por utilizar um método baseado em funções de penalização. O valor de cada restrição não satisfeita é somado ao valor da função objetivo, penalizando assim a solução infactível com este valor. A expressão (11) ilustra a função objetivo penalizada.

$$f'(\mathbf{x}) = f(\mathbf{x}) + \rho \left[\sum_{i=1}^{m} \max(g_i^2(\mathbf{x}), 0) + \sum_{i=1}^{l} h_i^2(\mathbf{x}) \right] \tag{11}$$

Onde $f'(\mathbf{x})$ é o valor da função de avaliação da solução $\mathbf{x}$ e $\rho = 10^7$ é o coeficiente de penalização.

https://proceedings.science/p/85380?lang=pt-br

3. Resultados

O algoritmo de Busca Dispersa deste trabalho foi implementado em MATLAB® e executado em um computador com processador Intel® Core™ i7-7700 de 3,6 GHz e 16 GB de RAM. Testes foram realizados considerando sete problemas que podem ser encontrados em (KOZIEL E MICHALEWICZ, 1999).

Tabela 1: Descrição dos problemas resolvidos (KOZIEL E MICHALEWICZ, 1999).

Problema	n	Tipo de Função Objetivo	Relação Região Factível/Espaço de busca (%)	Desigualdades Lineares	Igualdades Não Lineares	Desigualdades Não Lineares	Número de Restrições Ativas na Solução
G_1	13	Quadrática	0,0111	9	0	0	6
G_3	20	Polinomial	0,0000	0	1	0	1
G_6	2	Cúbica	0,0066	0	0	2	2
G_7	10	Quadrática	0,0003	3	0	5	6
G_8	2	Não linear	0,8560	0	0	2	0
G_{10}	8	Linear	0,0010	3	0	3	6
G_{11}	2	Quadrática	0,0000	0	1	0	1

Os problemas e as soluções obtidas são descritos abaixo e comparados com outros resultados disponíveis na literatura na Tabela 2.

3.1 Problema G1

O problema G1 é apresentado em (12)–(24). Trata-se de um problema de minimização de uma função objetivo quadrática com restrições lineares.

$$\min G_1(\mathbf{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{12}$$

Sujeito a:

$$2x_1 + 2x_2 + x_{10} + x_{11} \leq 10 \tag{13}$$
$$-8x_1 + x_{10} \leq 0 \tag{14}$$
$$-2x_4 - x_5 + x_{10} \leq 0 \tag{15}$$
$$2x_1 + 2x_3 + x_{10} + x_{12} \leq 10 \tag{16}$$
$$-8x_2 + x_{11} \leq 0 \tag{17}$$
$$-2x_6 - x_7 + x_{11} \leq 0 \tag{18}$$
$$2x_2 + 2x_3 + x_{11} + x_{12} \leq 10 \tag{19}$$
$$-8x_3 + x_{12} \leq 0 \tag{20}$$
$$-2x_8 - x_9 + x_{12} \leq 0 \tag{21}$$
$$0 \leq x_i \leq 1 \qquad i = 1, \ldots, 9 \tag{22}$$
$$0 \leq x_i \leq 100 \qquad i = 10, 11, 12 \tag{23}$$
$$0 \leq x_{13} \leq 1 \tag{24}$$

A solução obtida é $\mathbf{x}^* = (1; 1; 1; 1; 1; 1; 1; 1; 1; 3; 3; 3; 1)$, com $G_1(\mathbf{x}^*) = -15$.

3.2 Problema G3

O problema G3 é apresentado em (25)–(27). Trata-se de um problema de maximização

de uma função objetivo polinomial com uma restrição de igualdade não linear.

$$\max G_3(\mathbf{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i \tag{25}$$

Sujeito a:

$$\sum_{i=1}^{n} x_i^2 = 1 \tag{26}$$

$$0 \leq x_i \leq 1 \qquad 1 \leq i \leq n \tag{27}$$

Para $n = 20$, a solução obtida é $\mathbf{x}^* = 1/\sqrt{n} = 0{,}223606$, $i = 1, \ldots, 20$, com $G_3(\mathbf{x}^*) = 1$.

3.3 Problema G6

O problema G6 é apresentado em (28)–(32). Trata-se de um problema de minimização de uma função objetivo cúbica com duas restrições não lineares.

$$\min G_6(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{28}$$

Sujeito a:

$$-(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \tag{29}$$

$$(x_1 - 6)^2 + (x_2 - 5)^2 - 82{,}81 \leq 0 \tag{30}$$

$$13 \leq x_1 \leq 100 \tag{31}$$

$$0 \leq x_2 \leq 100 \tag{32}$$

A solução obtida é $\mathbf{x}^* =$ (14,09494204572340700; 0,84284058214040081), com $G_6(\mathbf{x}^*) = -6961{,}88176$.

3.4 Problema G7

O problema G7 é apresentado em (33)–(42). Trata-se de um problema de minimização de uma função objetivo quadrática com restrições lineares e não lineares.

$$\begin{aligned}\min G_7(\mathbf{x}) \\ &= x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ &\quad + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45\end{aligned} \tag{33}$$

Sujeito a:

$$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0 \tag{34}$$

$$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0 \tag{35}$$

$$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0 \tag{36}$$

$$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0 \tag{37}$$

$$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0 \tag{38}$$

$$-x_1^2 - 2(x_2 - 2)^2 + 2x_1 x_2 - 14x_5 + 6x_6 \geq 0 \tag{39}$$

$$-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0 \tag{40}$$

$$-0{,}5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0 \tag{41}$$

$$-10 \leq x_i \leq 10 \qquad i = 1, \ldots, 10 \tag{42}$$

https://proceedings.science/p/85380?lang=pt-br

A solução obtida é $\mathbf{x}^* =$ (2,1723061660132563; 2,3631022264309318; 8,7741794614522544; 5,0977329509349412; 0,9904776972039584; 1,4304215174580341; 1,3221410176021420; 9,8290763757147204; 8,2804780802833786; 8,3750729290372057), com $G_7(\mathbf{x}^*) = 24{,}306$.

3.5 Problema G8

O problema G8 é apresentado em (43)–(47). Trata-se de um problema de maximização de uma função objetivo não linear com restrições não lineares.

$$\max G_8(\mathbf{x}) = \frac{\text{sen}^3(2\pi x_1)\,\text{sen}(2\pi x_2)}{x_1^3(x_1 + x_2)} \tag{43}$$

Sujeito a:

$$x_1^2 - x_2 + 1 \leq 0 \tag{44}$$

$$1 - x_1 + (x_2 - 4)^2 \leq 0 \tag{45}$$

$$0 \leq x_1 \leq 10 \tag{46}$$

$$0 \leq x_2 \leq 10 \tag{47}$$

A solução obtida é $\mathbf{x}^* =$ (1,2279661965175832; 4,2453742217070642), com $G_8(\mathbf{x}^*) = 0{,}0958250$.

3.6 Problema G10

O problema G10 é apresentado em (48)–(57). Trata-se de um problema de minimização de uma função objetivo linear com restrições lineares e não lineares.

$$\min G_{10}(\mathbf{x}) = x_1 + x_2 + x_3 \tag{48}$$

Sujeito a:

$$-1 + 0{,}0025(x_4 + x_6) \leq 0 \tag{49}$$

$$-1 + 0{,}0025(x_5 + x_7 - x_4) \leq 0 \tag{50}$$

$$-1 + 0{,}01(x_8 - x_5) \leq 0 \tag{51}$$

$$-x_1x_6 + 833{,}33252x_4 + 100x_1 - 83333{,}333 \leq 0 \tag{52}$$

$$-x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0 \tag{53}$$

$$-x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0 \tag{54}$$

$$100 \leq x_1 \leq 10000 \tag{55}$$

$$1000 \leq x_i \leq 10000 \qquad i = 2, 3 \tag{56}$$

$$10 \leq x_i \leq 1000 \qquad i = 4, \ldots, 8 \tag{57}$$

A solução obtida é $\mathbf{x}^* =$ (579,19802294366627; 1360,24891286860470; 5106,95228232167760; 182,02468628833080; 295,66989105459993; 218,01448720916255; 286,45888551082550; 395,69535519135650), com $G_{10}(\mathbf{x}^*) = 7047{,}8207201713$.

3.7 Problema G11

O problema G1 é apresentado em (58)–(60). Trata-se de um problema de minimização de uma função objetivo quadrática com uma restrição de desigualdade não linear.

$$\min G_{11}(\mathbf{x}) = x_1^2 + (x_2 - 1)^2 \tag{58}$$

Sujeito a:

$$x_2 - x_1^2 = 0 \tag{59}$$

$$-1 \leq x_i \leq 1 \qquad i = 1, 2 \tag{60}$$

A solução obtida é $\mathbf{x}^*$ = (-0,70710678118; 0,5), com $G_{11}(\mathbf{x}^*) = 0{,}75$.

3.8 Comparação dos Resultados

A Tabela 2 apresenta os resultados obtidos com o método BD-FMR proposto em comparação com os melhores resultados de outros autores que utilizaram Algoritmos Genéticos.

Tabela 2: Comparação dos melhores resultados.

Problema	Melhor Solução Conhecida: Koziel e Michalewicz (1999)	Koziel e Michalewicz (1999)	Runarsson e Yao (2000)	Farmani e Wright (2003)	Venkatraman e Yen (2005)	Busca Dispersa
$\min G_1$	-15	-14,7864	-15,000	-15,0000	-14,9999	-15
$\max G_3$	1,0	0,9997	1,000	1,0000	1,0000	1,0000
$\min G_6$	-6961,8	-6952,1	-6961,814	-6961,800	-6961,17856	-6961,88176
$\min G_7$	24,306	24,620	24,307	24,480	24,410977	24,306
$\max G_8$	0,095825	0,0958250	0,095825	0,095825	0,095825	0,0958250
$\min G_{10}$	7049,33	7147,9	7054,316	7061,34	7060,5528	7047,8207201713
$\min G_{11}$	0,75	0,75	0,750	0,75	0,75	0,75

Os tempos de execução de cada teste em segundos foram respectivamente (0,434; 653,075; 0,605; 570,255; 0,213; 730,295; 1,364).

Observou-se também, que o número $b1$ de elementos representantes das soluções de qualidade do conjunto $RefSet$ influenciou bastante nos casos G_3, G_7 e G_{10} onde foi necessário aumentar o número de elementos de diversidade. Sendo assim, para esses casos foram utilizados os valores $b1 = 1$ e $b2 = 9$. Para o problema G_6 utilizou-se $b1 = 2$ e $b2 = 8$. Para os demais problemas foram utilizados os valores $b1 = 5$ e $b2 = 5$.

4. Conclusões

O objetivo deste trabalho foi apresentar o algoritmo de Busca Dispersa para resolução de problemas de otimização contínuos multimodais e restritos. O método apresentado utiliza um algoritmo baseado em sensibilidades da função objetivo e o algoritmo Simplex de Nelder-Mead para realizar a melhoria local nas soluções.

O método proposto foi testado em sete problemas multimodais restritos. Em todos eles o algoritmo foi capaz de encontrar as melhores soluções descritas na literatura. Desta forma, o algoritmo de Busca Dispersa apresentado se mostra promissor para ser aplicado em problemas de diversas áreas.

Como trabalhos futuros, o método apresentado será aplicado para resolver problemas de otimização de sistemas de energia elétrica, como o problema de fluxo de potência ótimo.

5. Agradecimentos

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processos nº 2015/21972-6 e 2014/23741-9, pelo apoio financeiro a este projeto de pesquisa.

https://proceedings.science/p/85380?lang=pt-br

Referências

Coello, C. A. C. (2002), Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11–12, 1245–1287.
Farmani, R.; Wright, J. A. (2003), Self-adaptive fitness formulation for constrained optimization, *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 5, pp. 445–455.
Kennedy, J.; Eberhart, R. (1995), Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, Perth, WA, vol.4, pp. 1942–1948.
Koziel, S.; Michalewicz Z. (1999), Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, *Evolutionary Computation*, Cambridge, vol. 7, no. 1, pp. 19–44.
Lagarias, J. C.; Reeds, J. A.; Wright, M. H.; Wright, P. E. (1998), Convergence properties of the Nelder-Mead Simplex method in low dimensions, *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 112–147.
Laguna, M.; Martí, R., Scatter search: methodology and implementations in C, *Springer US*, 2003.
Mladenović, N; Hansen P. (1997), Variable neighborhood search, *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100.
Runarsson, T. P.; Yao, X. (2000), Stochastic ranking for constrained evolutionary optimization, *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294.
Venkatraman, S.; Yen, G. G. (2005), A generic framework for constrained optimization using genetic algorithms, *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 424–435.

https://proceedings.science/p/85380?lang=pt-br