# Evolução Diferencial Aperfeiçoada para otimização contínua restrita

**Wendel A. X. Melo**
Instituto Alberto Luiz Coimbra de Pós-graduação e Pesquisa em Engenharia (COPPE)
Universidade Federal do Rio de Janeiro
wendelmelo@cos.ufrj.br

**Marcia H. C. Fampa**
Instituto de Matemática e COPPE
Universidade Federal do Rio de Janeiro
fampa@cos.ufrj.br

**Fernanda M. P. Raupp**
Departamento de Engenharia Industrial
Pontifícia Universidade Católica do Rio de Janeiro
fraupp@puc-rio.br

**RESUMO**

Este trabalho apresenta uma nova metaheurística baseada em Evolução Diferencial (ED) para o problema geral de programação não linear (PPNL) denominada Evolução Diferencial Aperfeiçoada (EDA). Contando com inovações no uso de operadores de cruzamento, mutação e seleção, EDA trabalha a partir de uma reformulação do problema original em um problema de otimização bi-objetivo em caixa. Ademais, EDA introduz um critério de parada adicional que permite a detecção de convergência da população, o qual evita operações computacionais desnecessárias e, consequentemente, aumenta sua eficiência. A qualidade do método proposto é mostrada a partir de sua aplicação em um conjunto bem conhecido de 13 instâncias de teste padrão de PPNL, e de sua comparação com os resultados de metaheurísticas bem conceituadas.

**PALAVRAS CHAVE:** programação não linear, evolução diferencial, otimização global.
**Área de classificação principal:** Metaheurísticas

**ABSTRACT**

This work presents a new metaheuristic based on Differential Evolution (DE) for the general nonlinear programming problem (NLPP) called Improved Differential Evolution (IDE). From a problem reformulation as a box constrained bi-objective problem, IDE innovates the usage of crossover, mutation and selection operators. Moreover, IDE introduces an additional stopping rule that can detect the population convergence, which avoids unnecessary computational operations and, consequently, increases efficiency. The quality of the proposed method is showed from its application on a well-known set of 13 standard test problems, and its comparison with results from well-respected metaheuristics.

**KEYWORDS:** nonlinear programming, differential evolution, global optimization.
**Main area:** Metaheuristics

# 1 Introdução

Problemas de otimização global aparecem com frequência em diversas aplicações relacionadas às ciências naturais e exatas, engenharias, economia e administração. O problema geral de programação não linear (PPNL), em sua forma de minimização, consiste em encontrar uma solução $x^* \in \mathbb{R}^n$ que resolve:

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

onde $f$, $g_i$, $i = 1, \ldots, p$, e $h_j$, $j = 1, \ldots, q$, são funções reais e $l$ e $u$ são vetores $n$-dimensionais, com $l \leq u$. Denotando a região viável do problema (1) por $\mathcal{F}$, dizemos que $\bar{x} \in \mathcal{F}$ é uma solução ótima local se $f(\bar{x}) \leq f(x)$ para todo $x \in \mathcal{F}$ numa vizinhança de $\bar{x}$. Uma solução $x^* \in \mathcal{F}$ tal que $f(x^*) \leq f(x)$ para todo $x \in \mathcal{F}$ é denominada solução ótima global, e, portanto, é a solução de nosso interesse.

Métodos clássicos ou exatos podem falhar em encontrar a solução ótima global de um PPNL, especialmente se o problema tratado for não-convexo. Nesses casos, pode haver convergência para uma solução ótima local, ou ainda, convergência para uma solução que não é nem ótima local nem ótima global. Adicionalmente, algoritmos dessa classe que resolvem PPNL costumam fazer uso de técnicas de derivação de funções para definir direções de busca, o que em alguns casos pode trazer dificuldades para a convergência caso alguma função não seja diferenciável. Esses algoritmos também podem apresentar dificuldades em convergir caso alguma função tratada seja não contínua ou não suave. Deste modo, a aplicação deste tipo de algoritmo em um PPNL não traz a garantia da obtenção de uma solução ótima global.

Devido às dificuldades expostas no parágrafo anterior, diversos pesquisadores têm se esforçado em desenvolver abordagens metaheurísticas para o PPNL nos últimos anos [1–3, 5, 7, 8, 10–14, 16, 17, 19]. Metaheurísticas são procedimentos que fazem uso de aleatoriedade em diversas estratégias visando encontrar boas soluções para o problema considerado. Este tipo de metodologia, originalmente lançado para a abordagem de problemas de otimização combinatória (o leitor interessado pode consultar [6]), também passou a ser aplicado recentemente em domínios contínuos. Embora não garantam a convergência para uma solução ótima global, as metaheurísticas podem, ao abordar um PPNL, fugir da convergência prematura para pontos de ótimo local muitas vezes apresentada pelos métodos clássicos, sendo então capazes de encontrar um ponto de ótimo global ou uma solução próxima a ele, mesmo em problemas cujas funções são não diferenciáveis, descontínuas ou possuam comportamento bastante irregular.

Metaheurísticas também podem ser combinadas a métodos exatos, formando assim algoritmos híbridos, visando diferentes propósitos, como, por exemplo: identificar regiões promissoras para aplicação de um método de busca exata, ou obter limites para a função objetivo de um determinado problema por meio de soluções viáveis para o mesmo, tal como os métodos de *branch-and-bound*. Nestes casos, o fato de as metaheurísticas poderem apresentar respostas diferentes para execuções independentes sobre um mesmo problema de entrada pode trazer eventuais incoveniências. Entretanto, isto pode ser facilmente evitado através do controle adequado de sementes de geração de números pseudo-aleatórios nos códigos de implementação.

Dentre as metaheurísticas de maior sucesso para a abordagem de PPNL está a Evolução Diferencial (ED). ED é um método heurístico baseado em população originalmente proposto

para otimização contínua com restrições de caixa em [18]. Recentemente, diversas abordagens baseadas em ED para PPNL foram apresentadas, como, por exemplo, [2, 9, 13, 15, 19].

Neste trabalho, apresentamos uma variante de ED para PPNL denominada Evolução Diferencial Aperfeiçoada (EDA). EDA introduz um novo mecanismo para aumentar a diversidade no processo de exploração do espaço de busca, juntamente com um aperfeiçoamento no critério de seleção utilizado por [13]. Adicionalmente, EDA introduz um novo critério de parada alternativo para detecção de convergência da população, permitindo ao algoritmo economizar tempo de execução e avaliações de funções. Experimentos computacionais mostraram que EDA é uma metaheurística promissora quando a comparamos com metaheurísticas consideradas eficientes para resolver PPNLs.

Este texto está organizado da seguinte forma: a Seção 2 discute aspectos da Evolução Diferencial. A Seção 3 traz uma reformulação para o PPNL utilizada neste trabalho. Na Seção 4, apresentamos os detalhes do algoritmo EDA juntamente com as suas contribuições. Resultados numéricos são fornecidos na Seção 5 ao passo que a Seção 6 traz as conclusões deste trabalho.

## 2 Evolução Diferencial

Evolução Diferencial (ED) é um método heurístico sem uso de derivadas proposto para resolver problemas de otimização contínua irrestrita por Storn e Price [18]. Como um método baseado em população, ED mantém um conjunto de soluções que evolui ao longo de um processo através da geração de novas soluções e eventual substituição de soluções da população por novas soluções que se mostrem melhores ou mais promissoras. A geração de novas soluções é feita através de um operador de mutação que realiza combinação linear entre três soluções da população e um operador de cruzamento (*crossover*) que mistura coordenadas do vetor gerado pelo operador de mutação e uma quarta solução da população (solução de cruzamento). Esta última solução competirá com a nova solução gerada pela permanência na população. Se a nova solução obtida for melhor que a solução de cruzamento de acordo com algum critério de seleção pré-estabelecido (por exemplo, pelo menor valor apresentado para a função objetivo), então a solução de cruzamento será substituída pela nova solução gerada.

```
1  Gere uma população inicial x^{i,0}, i = 1,...,P e avalie ;
2  Seja F um valor real entre 0 e 1 ;
3  para g = 1, ..., MAXGEN faça
4      para k = 1, ..., P faça
5          para i = 1, ..., M faça
6              Selecione aleatoriamente r_1 ≠ r_2 ≠ r_3 ≠ k ∈ {1...P} ;
7              r_nbr ← randint(1, n) ;
8              para j = 1, ... n faça
9                  se rand(0,1) ≤ CR OU j = r_nbr então
10                     filhos_j^{k,i} ← x_j^{r_3,g} + F(x_j^{r_1,g} − x_j^{r_2,g}) ;
11                 senão
12                     filhos_j^{k,i} ← x_j^{k,g} ;
13         Seja u^k a melhor solução (de acordo com critério pré-estabelecido) dentre x^{k,g} e
           filhos^{k,i}, i = 1,...M ;
14         x^{k,g+1} ← u^k ;
```

**Figura 1**: Algoritmo padrão ED com multi-descendência.

O algoritmo ED é apresentado na Figura 1, onde a função $rand(a, b)$ retorna um número real aleatório entre $a$ e $b$ e a função $randint(a, b)$ retorna um valor inteiro aleatório entre $a$ e $b$. Originalmente, ED realiza, a cada iteração, a geração de uma nova solução para cada solução da população. Entretanto, algumas das abordagens ED mais promissoras para PPNL (por exemplo, [13, 19]) utilizam a multi-geração de soluções descendentes, que nada mais é do que a geração de uma determinada quantidade $M$ de soluções novas para cada solução da população. Uma vez que todas as soluções da população são usadas como solução de cruzamento, esse tipo de abordagem ED gera então $MP$ soluções a cada iteração, onde $P$ é o número de soluções na população.

O total de iterações do algoritmo da Figura 1 é dado por $MAXGEN$. Este tem sido o único critério de parada utilizado por diversas abordagens baseadas em ED. Nestes casos, a execução do algoritmo baseado em ED sempre realiza um total de avaliações de função de $MP(MAXGEN) + P$. Note que se o algoritmo ED convergir para a solução ótima de um problema na iteração $\overline{g}$, tal que $\overline{g} < MAXGEN$, então haverá um desperdício de $MP(MAXGEN - \overline{g})$ avaliações de função. Dentre as contribuições deste trabalho, está um novo e simples critério de parada alternativo para a detecção de convergência da abordagem ED. Este novo critério de parada evita avaliações desnecessárias de função após a convergência, tornando a abordagem ED ainda mais eficiente e competitiva.

Operadores de mutação e cruzamento são utilizados nas linhas 9-12 do algoritmo da Figura 1. Eles são responsáveis pela eficácia no processo de exploração do espaço de soluções do problema e pela convergência para soluções de boa qualidade. Entretanto, para determinados problemas (especialmente problemas com múltiplas soluções ótimas locais), esses operadores podem não produzir a diversidade necessária para a obtenção de uma solução ótima global. Tentando remediar este fato, propomos neste trabalho uma modificação no operador de cruzamento ED visando adicionar mais diversidade no processo exploratório. Esta modificação também acarreta em um uso diferente do operador de mutação da ED.

Na linha 13 do algoritmo da Figura 1, utiliza-se um critério para selecionar qual solução dentre a solução de cruzamento e as soluções descendentes permanecerá na população. Este critério costuma ser particular nas diferentes abordagens ED; em alguns casos, ele é o que diferencia uma abordagem ED de outra. Para o caso da aplicação em um PPNL, este critério pode estar relacionado com a estratégia adotada para lidar com as restrições do problema. Ainda, neste trabalho, também propomos um aperfeiçoamento do critério de seleção introduzido por [12] e utilizado na abordagem ED por [13]. Nossas contribuições serão discutidas com mais detalhes na Seção 4.

## 3 Reformulação do problema

Utilizando as seguintes funções de violação de restrição:

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

o problema (1) pode ser 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 $G(x) = \sum_{i=1}^{p+q} G_i(x)$. A segunda componente do vetor de função objetivo, $G(x)$, é usualmente denominada como 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 de (1), e $G(x) > 0$ caso contrário.

É importante frisar que, apesar de a formulação (3) apresentar o problema original como um problema de otimização bi-objetivo, não fazemos nenhum uso de técnicas de otimização multi-objetivo neste trabalho. As funções $f(x)$ e $G(x)$ sempre são tratadas de forma separada e independente. Tal estratégia evita toda a dificuldade que os métodos de penalização clássicos apresentam em definir valores adequados para os coeficientes de penalização. Em particular, em nossa abordagem, temos que todos os coeficientes de penalidade têm seu valor definido como 1.

Ainda com respeito as funções $f(x)$ e $G(x)$, Deb apresentou em [4] três critérios para a comparação de soluções em problemas de otimização restrita:

1. Qualquer solução viável prevalece sobre qualquer solução inviável.
2. Entre duas soluções viáveis, a de menor valor da função objetivo prevalece.
3. Entre duas soluções inviáveis, a de menor termo de penalização prevalece.

Note que por estes critérios, a minimização de $G(x)$ tem preferência sobre a minimização de $f(x)$. Este fato está embasado na idéia de que é mais importante fornecer uma solução viável do que fornecer uma boa solução que seja inviável. Entretanto, sob certas condições, ao longo das iterações, trocar uma determinada solução por outra com um maior valor para o termo de penalização pode dar a um algoritmo uma maior habilidade para: (1) explorar os limites da região viável do problema abordado e (2) resolver problemas cuja a região viável possui componentes disjuntos. No caso específico do método ED, ter um critério de seleção que permita este tipo de movimento sem prejudicar o compromisso na obtenção de uma solução viável se mostra fundamental para a construção de uma abordagem de sucesso.

```
1  Gere uma população inicial x^i, i = 1, ..., P e avalie;
2  para g = 1, ..., MAXGEN faça
3      para k = 1, ..., P faça
4          para i = 1, ..., M faça
5              Selecione aleatoriamente r1 ≠ r2 ≠ r3 ≠ k ∈ {1...P};
6              r_nbr ← randint(1, n) ;
7              F ← rand(0,3 , 0,9)
8              se rand(0,1) ≤ α então
9                  para j = 1, ...D faça
10                     se rand(0,1) < CR OU j = r_nbr então
11                         filhos_j^{k,i} ← x_j^{r3} + F(x_j^{r1} − x_j^{r2}) ;
12                     senão
13                         filhos_j^{k,i} ← x_j^k ;
14             senão
15                 filhos^{k,i} ← geraDescDiverso(x^k, x^{r1}, x^{r2}, x^{r3}) ;
16         S_r ← S_r^0 * (1 − g/MAXGEN) ;
17         u^k ← criterioSelecao(S_r, x^k, filhos^k) ;
18         x^k ← u^k ;
       // Critério de parada alternativo
19     se f_max − f_min < ε e todas as soluções da população corrente são viáveis então
20         encerre o processo evolutivo ;
```

**Figura 2**: Algoritmo EDA.

# 4 Evolução Diferencial Aperfeiçoada

Neste trabalho, desenvolvemos uma nova variante de ED denominada Evolução Diferencial Aperfeiçoada (EDA), cujo algoritmo é apresentado na Figura 2. De forma similar a ED original, EDA realiza a geração de descendentes a partir de soluções já existentes na população por meio de seus operadores de mutação e cruzamento. Cada solução de cruzamento competirá com seus descendentes pela permanência na população. Em EDA, entende-se processo evolutivo como o processo de geração de novas soluções e atualização da população corrente. A seguir, descreveremos em detalhes as principais modificações introduzidas no algoritmo EDA.

## 4.1 Aumento da diversidade das soluções descendentes

Com o intuito de aumentar a diversidade nas soluções da população, EDA introduz o procedimento *geraDescDiverso* (linha 15 do algoritmo da Figura 2) para acrescentar soluções na população com uma carga mais elevada de mutações em relação ao padrão ED, o qual é descrito em detalhes na Figura 3.

**Entrada**: $x^k, x^{r_1}, x^{r_2}, x^{r_3}$
1 **para** $j = 1, \ldots n$ **faça**
2  $nalet \leftarrow \text{rand}(0,1)$ ;
3  **se** $nalet \leq CR1$ **então**
4  $x_j^d \leftarrow x_j^{r_3} + F(x_j^{r_1} - x_j^{r_2})$ ;
5  **senão se** $nalet \leq CR2 + CR1$ **então**
6  $x_j^d \leftarrow x_j^{r_2} + F(x_j^{r_3} - x_j^{r_1})$ ;
7  **senão se** $nalet \leq CR1 + CR2 + CR3$ **então**
8  $x_j^d \leftarrow x_j^{r_1} + F(x_j^{r_2} - x_j^{r_3})$ ;
9  **senão**
10  $x_j^d \leftarrow x_j^k$ ;
11 **retorna** $x^s$

**Figura 3**: Procedimento *geraDescDiverso*.

Como pode ser visto na Figura 3, as mutações podem ser geradas sobre qualquer uma das soluções $x^{r_1}$, $x^{r_2}$ e $x^{r_3}$, diferentemente da abordagem ED clássica, que sempre gera mutação somente sobre a solução $x^{r_3}$. Uma vez que é feito o cruzamento dessas três perturbações com a solução de cruzamento $x^k$, a tendência é que este tipo de mecanismo introduza soluções descendentes com uma carga maior de diversidade. Se esta carga de diversidade lhes trouxer vantagens competitivas sobre as demais soluções descendentes de $x^k$, então a tendência é que o operador de seleção as escolha para permanecer na população. Uma exploração mais efetiva do espaço de busca, traduzida por um conjunto de soluções de maior diversidade, pode ajudar a evitar a convergência prematura para pontos de ótimo local. Quando o operador de mutação gera um valor para uma coordenada $j$ fora do intervalo dado por $[l_j, u_j]$, um valor aleatório para esta coordenada é sorteado nesse intervalo.

Note que, conforme a linha 8 do algoritmo EDA (Figura 2), este procedimento é chamado com probabilidade $1 - \alpha$ a cada geração de descendente, onde $\alpha$ é um parâmetro ajustado pelo usuário. Valores baixos para $\alpha$ podem comprometer a capacidade de convergência do algoritmo ED, pois acarretam na geração com maior frequência de indivíduos com carga alta de mutação (e consequentemente, com maior diversidade). Para não prejudicar a convergência do algoritmo, recomendamos valores altos para o parâmetro $\alpha$. Neste trabalho, ajustamos esse parâmetro com o valor 0,8.

### 4.2 Operador de seleção aperfeiçoado

Após a geração das $M$ soluções descendentes de uma determinada solução $x^k$ em uma determinada geração $g$, é preciso escolher dentre $x^k$ e as suas $M$ descendentes uma solução para permanecer na população, descartando assim as demais soluções desse grupo. A abordagem ED original escolhe sempre a solução que apresenta o melhor valor para a função objetivo, uma vez que a mesma foi proposta para problemas só com restrições de caixa. As abordagens ED propostas para PPNL costumam também levar em conta questões como viabilidade, aleatoriedade e o fato das soluções se mostrarem promissoras.

Neste trabalho, optamos por utilizar uma estratégia de seleção baseada na apresentada por Mezura-Montes et al. em [12] e utilizada de forma integrada à ED em [13]. No algoritmo EDA (Figura 2), este procedimento é executado na linha 17 e está descrito na Figura 4.

```
   Entrada: S_r, x^k, filhos^k
 1 Escolha u como a melhor solução dentre as M soluções descendentes geradas (filhos^k) com
   base nos três critérios de Deb. ;
 2 se rand(0,1) ≤ S_r então
 3 |   se f(u) ≤ f(x^k) então
 4 |   |   x^s ← u;
 5 |   senão
 6 |   |   x^s ← x^k ;
 7 senão
 8 |   se u é melhor que x^k de acordo com os três critérios de Deb então
 9 |   |   x^s ← u;
10 |   senão
11 |   |   x^s ← x^k;
12 retorna x^s
```

**Figura 4**: Procedimento *criterioSelecao*.

Como pode ser visto na Figura 4, a melhor solução descendente $u$ com base nos critérios de Deb é escolhida para ser comparada com a solução de cruzamento $x^k$. Com probabilidade $S_r$, esta comparação é feita tomando por base apenas o valor da função objetivo nas duas soluções. Dessa forma, ao longo do processo evolutivo, uma solução viável $x^k$ pode eventualmente ser substituída por uma solução inviável $u$. Esse mecanismo permite uma exploração mais eficiente da região viável em problemas onde a mesma possui componentes disjuntos e também permite localizar, de forma eficiente, soluções ótimas que estejam na fronteira da região viável. Observe que, com probabilidade $1 - S_r$, a comparação entre $u$ e $x^k$ é feita também com base nos critérios de Deb.

Em [13], o parâmetro $S_r$ é mantido estático ao longo de todo o processo evolutivo. Esta opção apresenta a possibilidade das melhores soluções viáveis da população serem trocadas por soluções inviáveis nas últimas gerações. Tal acontecimento é indesejável devido a possibilidade de perda de uma boa solução (talvez a melhor dentre todas) e ao fato de que inserir uma solução inviável nas últimas gerações, ainda que esta solução inviável seja bastante promissora, não beneficiará o processo evolutivo uma vez que não haverá tempo para o algoritmo valer-se das vantagens da nova solução já que o processo estará perto do fim. Além disso, nas primeiras gerações, pode ser interessante utilizar uma probabilidade $S_r$ mais alta para permitir uma exploração mais eficaz do espaço de busca. Devido a isto, optamos por tornar a probabilidade $S_r$ não-estática. No algoritmo EDA (Figura 2), o valor $S_r$ é atualizado na linha 16 através da seguinte expressão:

$$S_r = S_r^0 \left(1 - g/MAXGEN\right), \tag{4}$$

onde $S_r^0$ é um parâmetro definido pelo usuário, e $g$ é a geração corrente. Repare que pela equação (4), $S_r$ terá um valor próximo a $S_r^0$ nas primeiras gerações, e um valor cada vez mais próximo a zero nas gerações finais (até se anular na última).

É importante ressaltar que implementamos o algoritmo da Figura 4 de uma maneira mais eficiente que a feita em [13]. Note que, diferentemente de como foi sugerido em [13], não é preciso avaliar $f(x)$ e $G(x)$ para todas as soluções descendentes geradas. Na linha 1 do algoritmo da Figura 4, aplicamos os critérios de Deb para escolher a melhor solução descendente. Entretanto, ao avaliar cada solução descendente, se uma solução descendente $x^d$ apresentar:

$$G(x^d) > G(\bar{x}^d),$$

onde $\bar{x}^d$ é a melhor solução descendente conhecida até então no grupo avaliado, então $x^d$ pode ser descartada pelo terceiro critério de Deb sem ter $f(x^d)$ avaliadao. Este fato nos permite poupar consideravelmente o número de avaliações de função objetivo executadas.

Observe que na abordagem EDA, quando uma solução $x^k$ é atualizada na geração corrente, ela imediatamente se torna disponível para o operador de mutação nessa mesma geração, diferentemente da abordagem ED clássica, onde a solução atualizada só pode ser utilizada pelo operador de mutação na geração seguinte.

### 4.3 Um novo critério de parada alternativo

Adicionalmente ao número máximo de gerações, EDA introduz um novo critério de parada alternativo. O objetivo deste novo critério é detectar a convergência do algoritmo e evitar esforço computacional desnecessário. Na linha 19 do algoritmo EDA (Figura 2), temos o seguinte critério de parada adicional:

$$f_{max} - f_{min} < \epsilon, \tag{5}$$

e todas as soluções da população são viáveis. Aqui, $f_{max}$ e $f_{min}$ são, respectivamente, o maior e o menor valores apresentado por alguma solução da população corrente para a função objetivo e $\epsilon$ é um parâmetro definido pelo usuário. A expressão (5) é capaz de detectar convergência, pois se a diferença do valor da função objetivo entre a melhor e a pior solução da população é bem pequena (com todas as soluções da população viáveis), a tendência então é que todas as soluções da população estejam bem próximas umas das outras, caracterizando convergência. Uma vez que todas as coordenadas das novas soluções descendentes geradas dependem das respectivas coordenadas das soluções já existentes, de um modo geral, podemos afirmar que dificilmente as novas soluções geradas se afastarão do ponto de convergência da população corrente. Assim, ainda que esse ponto de convergência não seja uma solução ótima global, podemos, com alto grau de segurança, encerrar o processo evolutivo se a expressão (5) for satisfeita.

## 5 Resultados Numéricos

Nesta seção apresentamos alguns resultados numéricos da aplicação de EDA em 13 instâncias PPNL padrão visando a comparação com algoritmos que resolvem problemas dessa classe. Todos os resultados foram obtidos em uma estação de trabalho Windows XP no ambiente Matlab 7.6.0. A Tabela 1 resume os valores padrão utilizados para os parâmetros de EDA.

Repare que, pelo fato de os parâmetros $MAXGEN$, $P$ e $M$ estarem ajustados, respectivamente, em 1000, 70 e 5, EDA realizará no máximo 350.070 avaliações de função objetivo e de restrições para cada problema considerado. Entretanto, o número efetivo dessas avaliações pode ser significativamente menor em certos casos devido ao critério de parada

alternativo EDA e a economia de avaliações de função objetivo realizada pelo operador de seleção.

| Parâmetro | Valor | Descrição |
|---|---|---|
| $MAXGEN$ | 1000 | Numero máximo de gerações |
| $P$ | 70 | Tamanho da população |
| $M$ | 5 | Número de descendentes gerados para cada solução da população em cada geração |
| $\alpha$ | 0,8 | Probabilidade de mutação apenas sobre $x^{r_3}$. |
| CR | 0,9 | Probabilidade de cada componente das novas soluções geradas ser definida pelo operador de mutação no cruzamento padrão ED |
| CR1 | 0,3 | Probabilidade de cada componente do cruzamento com carga alta de mutação ser definida pelo operador de mutação sobre $x^{r_3}$ |
| CR2 | 0,3 | Probabilidade de cada componente do cruzamento com carga alta de mutação ser definida pelo operador de mutação sobre $x^{r_2}$ |
| CR3 | 0,3 | Probabilidade de cada componente do cruzamento com carga alta de mutação ser definida pelo operador de mutação sobre $x^{r_1}$ |
| $S_r^0$ | 0,7 | Valor inicial de $S_r$ em cada geração |
| $\epsilon$ | $10^{-7}$ | Tolerância para critério de parada alternativo |

Tabela 1: Parâmetros de EDA usados nos experimentos numéricos

### 5.1 Comparação com outras metaheurísticas

Aplicamos EDA em 13 instâncias de teste PPNL padrão. A descrição dessas instâncias pode ser encontrada em [13, 17, 19]. Comparamos os resultados de EDA com os resultados das seguintes abordagens que representam o estado-da-arte com respeito a metaheurísticas competitivas para PPNL:

- Busca Local Intensiva (BLI), de Melo et al. [11].
- Improved Stochastic Ranking (ISR), de Runarsson e Yao [17].
- Diversity-Differential Evolution (DDE), de Mezura-Montes et al. [13].
- Dynamic Stochastic Selection-Multimember Differential Evolution, segundo experimento (DSS-MDE), de Zhang et al. [19].

O método BLI foi executado 30 vezes para cada instância de teste, ao passo que EDA e os demais métodos foram executados 100 vezes para cada instância. A Tabela 2 mostra os resultados da aplicação dos métodos acima mencionados nas 13 instâncias de teste. Os resultados obtidos pelas abordagens citadas foram retirados de suas respectivas referências. A primeira coluna da Tabela 2 (Prob) identifica o problema teste, o tipo da função objetivo (minimização ou maximização), e seu melhor valor conhecido. A segunda coluna (Abord) identifica o procedimento que forneceu os resultados. As três colunas seguintes apresentam, respectivamente, o melhor valor (Melhor), o valor médio (Média), e o pior valor (Pior) da função objetivo obtido com os procedimentos em todas as execuções. O desvio padrão (DP), a média de avaliações da função objetivo (F-avals) e a média de avaliações das funções de restrições (R-avals) são exibidos nas últimas três colunas.

Como pode ser verificado na Tabela 2, EDA foi capaz de obter a melhor solução conhecida em todas as suas execuções para 12 das 13 instâncias de teste (todas as instâncias exceto G2). Este é o melhor resultado dentre as metaheurísticas consideradas para esse conjunto de instâncias de teste, uma vez que BLI só foi capaz de sempre obter a melhor solução em

| Prob | Abord | Melhor | Média | Pior | DP | F-avals | R-avals |
|---|---|---|---|---|---|---|---|
| | EDA | -15.000 | -15.000 | -15.000 | 4.41E-08 | 71,504 | 135,254 |
| G1 | ISR | -15.000 | -15.000 | -15.000 | 5.80E-14 | 350,000 | 350,000 |
| (min) | DDE | -15.000 | -15.000 | -15.000 | 1.00E-09 | 225,000 | 225,000 |
| -15 | DSS-MDE | -15.000 | -15.000 | -15.000 | 0.00E+00 | 350,000 | 350,000 |
| | BLI | -14.993 | -14.966 | -14.862 | 0.02554 | 167,870 | 37,431 |
| | EDA | 0.803619 | 0.796934 | 0.751228 | 9.77E-03 | 169,294 | 231,588 |
| G2 | ISR | 0.803619 | 0.782715 | 0.723591 | 2.20E-02 | 350,000 | 350,000 |
| (max) | DDE | 0.803619 | 0.798079 | 0.751742 | 1.01E-02 | 225,000 | 225,000 |
| 0.803619 | DSS-MDE | 0.803619 | 0.788011 | 0.744690 | 1.50E-02 | 350,000 | 350,000 |
| | BLI | 0.786136 | 0.624451 | 0.470020 | 0.07409 | 267,811 | 23,283 |
| | EDA | 1.0005 | 1.0005 | 1.0005 | 8.64E-08 | 67,892 | 137,610 |
| G3* | ISR | 1.0010 | 1.0010 | 1.0010 | 8.20E-09 | 350,000 | 350,000 |
| (max) | DDE | 1.0000 | 1.0000 | 1.0000 | 0.00E+00 | 225,000 | 225,000 |
| 1 | DSS-MDE | 1.0005 | 1.0005 | 1.0005 | 2.70E-09 | 350,000 | 350,000 |
| | BLI | 1.0005 | 0.9993 | 0.9969 | 0.00081 | 89,144 | 176,853 |
| | EDA | -30665.539 | -30665.539 | -30665.539 | 1.97E-08 | 33,275 | 57,148 |
| G4 | ISR | -30665.539 | -30665.539 | -30665.539 | 1.10E-11 | 350,000 | 350,000 |
| (min) | DDE | -30665.539 | -30665.539 | -30665.539 | 0.00E+00 | 225,000 | 225,000 |
| -30665.539 | DSS-MDE | -30665.539 | -30665.539 | -30665.539 | 2.70E-11 | 350,000 | 350,000 |
| | BLI | -30665.469 | -30665.400 | -30665.322 | 0.03776 | 119,972 | 53,605 |
| | EDA | 5126.4961 | 5126.4961 | 5126.4961 | 7.32E-09 | 46,615 | 95,613 |
| G5* | ISR | 5126.497 | 5126.497 | 5126.497 | 7.20E-13 | 350,000 | 350,000 |
| (min) | DDE | 5126.497 | 5126.497 | 5126.497 | 0.00E+00 | 225,000 | 225,000 |
| 5126.4981 | DSS-MDE | 5126.497 | 5126.497 | 5126.497 | 0.00E+00 | 350,000 | 350,000 |
| | BLI | 5126.4963 | 5126.7328 | 5127.8049 | 0.36410 | 3,798 | 210,768 |
| | EDA | -6961.814 | -6961.814 | -6961.814 | 3.43E-09 | 11,414 | 18,225 |
| G6 | ISR | -6961.814 | -6961.814 | -6961.814 | 1.90E-12 | 350,000 | 350,000 |
| (min) | DDE | -6961.814 | -6961.814 | -6961.814 | 0.00E+00 | 225,000 | 225,000 |
| -6961.814 | DSS-MDE | -6961.814 | -6961.814 | -6961.814 | 0.00E+00 | 350,000 | 350,000 |
| | BLI | -6961.814 | -6961.814 | -6961.814 | 0.00000 | 8,620 | 13,585 |
| | EDA | 24.306 | 24.306 | 24.306 | 2.82E-07 | 101,865 | 201,366 |
| G7 | ISR | 24.306 | 24.306 | 24.306 | 6.30E-05 | 350,000 | 350,000 |
| (min) | DDE | 24.306 | 24.306 | 24.306 | 8.22E-09 | 225,000 | 225,000 |
| 24.306 | DSS-MDE | 24.306 | 24.306 | 24.306 | 7.00E-08 | 350,000 | 350,000 |
| | BLI | 24.310 | 24.503 | 24.759 | 0.10153 | 124,687 | 95,338 |
| | EDA | 0.095825 | 0.095825 | 0.095825 | 7.64E-11 | 4,197 | 5,436 |
| G8 | ISR | 0.095825 | 0.095825 | 0.095825 | 2.70E-17 | 350,000 | 350,000 |
| (max) | DDE | 0.095825 | 0.095825 | 0.095825 | 0.00E+00 | 225,000 | 225,000 |
| 0.095825 | DSS-MDE | 0.095825 | 0.095825 | 0.095825 | 3.90E-17 | 350,000 | 350,000 |
| | BLI | 0.095825 | 0.095825 | 0.095825 | 0.00000 | 10,731 | 2,467 |
| | EDA | 680.630 | 680.630 | 680.630 | 1.52E-08 | 33,136 | 54,089 |
| G9 | ISR | 680.630 | 680.630 | 680.630 | 3.20E-13 | 350,000 | 350,000 |
| (min) | DDE | 680.630 | 680.630 | 680.630 | 0.00E+00 | 225,000 | 225,000 |
| 680.630 | DSS-MDE | 680.630 | 680.630 | 680.630 | 2.50E-13 | 350,000 | 350,000 |
| | BLI | 680.630 | 680.656 | 680.767 | 0.03223 | 128,022 | 49,890 |
| | EDA | 7049.248 | 7049.248 | 7049.248 | 1.15E-07 | 143,263 | 301,270 |
| G10 | ISR | 7049.248 | 7049.250 | 7049.270 | 3.20E-03 | 350,000 | 350,000 |
| (min) | DDE | 7049.248 | 7049.266 | 7049.617 | 4.45E-02 | 225,000 | 225,000 |
| 7049.248 | DSS-MDE | 7049.248 | 7049.248 | 7049.249 | 3.10E-04 | 350,000 | 350,000 |
| | BLI | 7057.009 | 7131.911 | 7453.741 | 91.8994 | 173,709 | 234,256 |
| | EDA | 0.7499 | 0.7499 | 0.7499 | 7.95E-10 | 8,556 | 16,300 |
| G11* | ISR | 0.7500 | 0.7500 | 0.7500 | 1.10E-16 | 350,000 | 350,000 |
| (min) | DDE | 0.7500 | 0.7500 | 0.7500 | 0.00E+00 | 225,000 | 225,000 |
| 0.75 | DSS-MDE | 0.7499 | 0.7499 | 0.7499 | 0.00E+00 | 350,000 | 350,000 |
| | BLI | 0.7499 | 0.7500 | 0.7539 | 0.00073 | 1,070 | 7,604 |
| | EDA | -1.0000 | -1.0000 | -1.0000 | 3.93E-10 | 4,794 | 7,441 |
| G12 | ISR | -1.0000 | -1.0000 | -1.0000 | 1.20E-09 | 350,000 | 350,000 |
| (min) | DDE | -1.0000 | -1.0000 | -1.0000 | -1.00E+00 | 225,000 | 225,000 |
| -1 | DSS-MDE | -1.0000 | -1.0000 | -1.0000 | 0.00E+00 | 350,000 | 350,000 |
| | BLI | -1.0000 | -1.0000 | -1.0000 | 0.00000 | 51,680 | 1,790 |

| Prob | Abord | Melhor | Média | Pior | DP | F-avals | R-avals |
|---|---|---|---|---|---|---|---|
| | EDA | 0.0539 | 0.0539 | 0.0539 | 3.68E-09 | 46,241 | 96,443 |
| G13* | ISR | 0.0539 | 0.0668 | 0.4388 | 7.00E-02 | 350,000 | 350,000 |
| (min) | DDE | 0.0539 | 0.0693 | 0.4388 | 7.58E-02 | 225,000 | 225,000 |
| 0.0539 | DSS-MDE | 0.0539 | 0.0539 | 0.0539 | 8.30E-17 | 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

5 dessas 13 instâncias (G3, G6, G8, G11 e G12), ISR e DDE realizaram este feito para 10 das 13 instâncias (G1, G3, G4, G5, G6, G7, G8, G9, G11 e G12) e DSS-MDE para 11 das 13 instâncias (todas as instâncias exceto G2 e G10).

É possível também notar que, em termos de avaliações de funções objetivo e de restrições, EDA apresentou um menor esforço computacional que ISR e DSS-MDE sobre todas as instâncias de teste. Com relação a DDE, EDA apresentou menor esforço computacional de forma incontestável sobre todas as instâncias de teste, exceto por G10. Nessa última instância, EDA apresentou um número maior de avaliações de restrições e um número menor de avaliações de função objetivo. Em comparação com BLI, podemos dizer com segurança que EDA apresentou menor esforço computacional sobre as instâncias G3, G4 e G9. Entretanto, para as instâncias G1, G5, G8 e G12 não podemos afirmar com certeza qual das duas abordagens foi mais eficiente nesses termos. Podemos atribuir os excelentes resultados apresentados por EDA com relação ao esforço computacional ao bom mecanismo de convergência do núcleo da metodologia ED, a implementação eficiente do operador de seleção, que economizou em média 43% de avaliações de função objetivo, e ao critério de parada alternativo proposto neste trabalho, que foi capaz de economizar em até 98% o número de avaliações de função executadas (instância G12).

## 6 Conclusões

Apresentamos nesse trabalho, uma nova metaheurística baseada em Evolução Diferencial (ED) para o Problema geral de Programação Não Linear (PPNL) denominada Evolução Diferencial Aperfeiçoada (EDA). EDA introduz inovações nos operadores de cruzamento e mutação, um aperfeiçoamento no critério de seleção utilizado em [13] capaz de aumentar sua eficácia e diminuir seu esforço computacional, além de introduzir um critério de parada alternativo de simples implementação, que pode encerrar o processo evolutivo quando a convergência das soluções da população é detectada, evitando assim operações desnecessárias.

Aplicamos o algoritmo EDA em um conjunto padrão composto de 13 instâncias de teste PPNL bem conhecidas. A comparação de EDA com metaheurísticas bem conceituadas para este tipo de problema revela EDA como uma metodologia bastante competitiva e promissora para a abordagem de PPNL, uma vez que, considerando o conjunto de instâncias como um todo, EDA forneceu os resultados de melhor qualidade dentre as abordagens tomadas a um custo computacional inferior na maioria delas, evidenciando assim o sucesso das inovações aqui propostas.

## Referências

[1] S. Akhtar, K. Tai e T. Ray, A socio-behavioural simulation model for engineering design optimization, Eng. Optim. 34(4) 2002, pp. 341-354.

[2] R. L. Becerra e C. A. Coello Coello, Cultured differential evolution for constrained optimization, Comput. Methods Appl. Mech. Engig. 195, 2006, pp. 4303-4322.

[3] S. P. Chandra e L. Ozdamar, Investigating a hybrid simulated annealing and local search algorithm for constrained optimization, European Journal of Operational Research 185 2008, pp. 1230-1245.

[4] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2-4), 2000, pp. 311-338.

[5] S. B. Hamida e M. Schoenauer, ASCHEA: New rsults using adaptive segregational constraint handling; n Proceedings of the Congress on Evolutionary Computation(CEC2002); Piscataway, New Jersey, IEEE Service Center 2002, pp. 884-889.

[6] F. Glover e G. A. Kochenberger (editors), Handbook of Metaheuristics, Kluwer Academic Publishers, New York, 2003.

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

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

[9] J. Lampinen, A constraint handling approach for the differential evolution algorithm, in: Proc. 2002 IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, May 2002, pp. 1468-1473.

[10] H. Liu, Z. Cai e Y. Wand, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Applied Soft Computing 10, 2010, pp. 629-640.

[11] W. A. X. Melo, M. H. C. Fampa e F. M. P. Raupp, Busca Local Intensiva: Uma nova metaheurística para otimização global contínua restrita, Anais do XLI Simpósio Brasileiro de Pesquisa Operacional, SBPO, 2009.

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

[13] E. Mezura-Montes, J. Velázquez-Reyes e C. A. Coello Coello, Promising Infeasibility and Multiple Offspring Incorporated to Differential Evolution for Constrained Optimization; Genetic and Evolutionary Computation Conference (GECCO 2005); pp 225-232.

[14] E. Mezura-Montes, C.A. Coello Coello e J.V. Reyes, Increasing Successful Offspring and Diversity in Differential Evolution for Engineering Design; Proceedings of the Seventh International Conference on Adaptive Computing in Design and Manufacture (ACDM 2006); pp. 131-139.

[15] E. Mezura-Montes, C.A. Coello Coello e E.I. Tun-Morales. Simple feasibility rules and differential evolution for constrained optimization. IMICAI'2004, LNAI 2972, pp. 707-716.

[16] T. Ray e K.M. Liew. Society and Civilization: An Optimization Algorithm Based on the Simulation of Social Behavior; IEEE Trans. Evol. Comput. 7 (4) 2003; pp. 386-396.

[17] T. P. Runarsson e X. Yao, Search Biases in Constrained Evolutionary Optimization. IEEE Transactions on Systems, Man, and Cybernetics-part C 35 (2) 2005; pp. 233-243.

[18] R. Storn e K. Price. Differential evolution: a simple and efficient heuristic for global optimization over continuous space. Journal of Global Optimization, 11 (4), 1997, pp. 341-359.

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