UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

FELIPE LUCHI

UM ALGORITMO HÍBRIDO ENTRE EVOLUÇÃO DIFERENCIAL E NELDER-MEAD USANDO ENTROPIA PARA PROBLEMAS DE OTIMIZAÇÃO NÃO-LINEAR INTEIRA MISTA

VITÓRIA
2016

FELIPE LUCHI

UM ALGORITMO HÍBRIDO ENTRE EVOLUÇÃO DIFERENCIAL E NELDER-MEAD USANDO ENTROPIA PARA PROBLEMAS DE OTIMIZAÇÃO NÃO-LINEAR INTEIRA MISTA

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática na área de Inteligência Computacional.

Orientador: Prof. Dr.-Ing. Renato A. Krohling

VITÓRIA
2016

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

Luchi, Felipe, 1984-

L936a Um algoritmo híbrido entre Evolução Diferencial e Nelder-Mead usando entropia para problemas de otimização não-linear inteira mista / Felipe Luchi. – 2016.

95 f. : il.

Orientador: Renato Antônio Krohling.

Dissertação (Mestrado em Informática) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Algoritmos. 2. Entropia. 3. Otimização matemática. 4. Otimização não linear. I. Krohling, Renato Antônio. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

Dissertação sob o título de "UM ALGORITMO HÍBRIDO ENTRE EVOLUÇÃO DIFERENCIAL E NELDER-MEAD USANDO ENTROPIA PARA PROBLEMAS DE OTIMIZAÇÃO NÃO-LINEAR INTEIRA MISTA" defendida por FELIPE LUCHI e aprovada em 2016 em VITÓRIA, Estado do Espírito Santo, pela banca examinadora constituída dos seguintes professores:

Prof. Dr.-Ing. Renato A. Krohling
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Celso Alberto Saibel Santos
Universidade Federal do Espírito Santo
Membro Interno

Prof. Dr. Helio José Corrêa Barbosa
Universidade Federal de Juiz de Fora e
Laboratório Nacional de Computação Científica
Membro externo

Declaração de autoria

Eu declaro que essa dissertação foi escrita por mim. Qualquer ajuda recebida durante o trabalho de pesquisa e na escrita tem sido reconhecida. Além disso, eu certifico que todas as fontes de informação e literatura usadas estão indicadas na dissertação.

_____

Felipe Luchi

Agradecimentos

Ao professor Renato Krohling, por aceitar me orientar nesta jornada.

Aos colegas de laboratório, que de alguma forma contribuíram com meu crescimento acadêmico.

Aos professores Celso Alberto Saibel Santos e Helio José Corrêa Barbosa por aceitarem avaliar este trabalho.

A minha esposa Marcia, pelo companheirismo em todos os momentos.

Aos meus pais, pela inspiração e por sempre me incentivarem nos estudos.

Information is the resolution of uncertainty
(Claude Shannon)

Resumo

Vários problemas na engenharia são formulados como problemas de otimização não-linear inteira mista. Métodos estocásticos vem sendo utilizados devido ao seu desempenho, flexibilidade, adaptabilidade e robustez. Por um lado, Evolução Diferencial pode ser utilizado em funções de qualquer natureza e possui habilidades em busca global. Por outro lado, o algoritmo Nelder-Mead pode ser utilizado para refinar soluções candidatas. Este trabalho propõe uma abordagem híbrida entre os algoritmos Evolução Diferencial e Nelder-Mead para problemas de otimização não-linear inteira mista, onde o chaveamento é realizado através da entropia da população. O algoritmo Nelder-Mead foi estendido para manipular variáveis inteiras. O primeiro protótipo foi desenvolvido para solucionar problemas de otimização não-linear inteira sem restrições. O método *Alfa Constrained* foi incorporado para tratar problemas de otimização não-linear inteira com restrições. Por último, a abordagem foi testada em problemas de otimização não-linear inteira mista com restrições mostrando sua eficácia. A principal vantagem deste método é a habilidade de realizar o chaveamento de acordo com a entropia da população durante a busca.

Palavras-chave: Evolução Diferencial, Nelder-Mead, Entropia, Otimização não-linear inteira mista.

Abstract

Many problems in engineering are formulated as mixed integer non-linear optimization problems. Stochastic methods have been used due to their performance, flexibility, adaptability and robustness. On the one side, Differential Evolution can be used in any kind of functions and has global search ability. On the other side, the Nelder-Mead algorithm can be used to improve candidate solutions. This work proposes a hybrid approach between Differential Evolution and Nelder-Mead for mixed integer non-linear optimization problems, where the local search is activated by means of the population entropy. The Nelder-Mead algorithm was extended to handle integer variables. The first prototype was developed to solve integer non-linear optimization problems without constraint. The Alpha Constrained method was incorporated to deal with constrained integer non-linear optimization problems. Finally, the approach was aplied to constrained mixed integer non-linear optimization problems showing its effectiveness. The main advantage of this method is the ability to switch between global and local by means of the population entropy during the search.

Keywords: Differential Evolution, Nelder-Mead, Entropy, Mixed integer non-linear optimization.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

Um dos objetivos da ciência da computação é propor soluções computacionais para problemas do mundo real. Algoritmos têm sido propostos para encontrar boas soluções, que podem ser obtidas em tempo razoável. Entretanto, devido à sua natureza, ou volume de dados, estes algoritmos não são aplicáveis (SOCHA, 2009).

Entre os algoritmos que garantem a solução ótima, chamados algoritmos determinísticos, está o algoritmo Simplex, proposto por Dantzig em 1947, aplicado a problemas de otimização linear. Os algoritmos *Branch and Bound* e o corte de Gomory também garantem a solução ótima e são aplicados a problemas de otimização linear inteira (ARORA; HUANG; HSIEH, 1994; DANTZIG, 1998; CORNUÉJOLS, 2007). Apesar de garantir a solução ótima, estas abordagens possuem limitações quanto aos recursos computacionais, tais como memória e tempo disponíveis. Além disso, problemas de otimização do mundo real podem ter variáveis, funções objetivos e restrições, de natureza distintas, e os algoritmos citados não podem ser aplicados.

Diante das limitações dos algoritmos determinísticos, os algoritmos evolutivos receberam muita atenção nas últimas décadas, devido à sua flexibilidade e adaptabilidade combinada com seu desempenho robusto e características de busca global (BÄCK; HAMMEL; SCHWEFEL, 1997).

Um algoritmo evolutivo deve incorporar de forma eficaz as buscas global e local. Deve explorar o espaço de busca a fim de identificar regiões promissoras e intensificar a busca nessas regiões. Sendo assim, surgem os seguintes questionamentos (ČREPINŠEK; LIU; MERNIK, 2013):

1. Como os componentes dos algoritmos contribuem para a exploração/intensificação?
2. Quando e como à exploração/intensificação são controladas?
3. Como obter o balanceamento entre a exploração/intensificação?

Apesar dos algoritmos evolutivos possuírem habilidades em identificar regiões promissoras, eles podem não ser capazes de refinar boas soluções. Os algoritmos Busca Tabu e Cozimento Simulado são utilizados como operadores de busca local, ambos modificando uma solução pivô, que, por outro lado, se escolhidas indevidamente, podem levar a mínimos locais (KRASNOGOR; HART; SMITH, 2004).

Diante das características de cada algoritmo, as abordagens híbridas surgem como uma alternativa para contornar estas desvantagens. Um algoritmo híbrido, ou algoritmo memético, é composto por uma meta-heurística baseada em população e um mecanismo de busca local que é ativado com certa frequência em um conjunto de soluções (NERI; COTTA, 2012). Esta definição deixa algumas questões em aberto como:

- Qual solução utilizar como pivô para a busca local?
- Com que frequência realizar a busca local?

A fim de responder os questionamentos relacionados aos algoritmos híbridos, várias abordagens foram propostas. Algumas delas levam em consideração a probabilidade de se aplicar a busca local em um conjunto de soluções (DUAN; LIAO; YI, 2013). Outra mais elaborada, apesar de não ter sido aplicada em um algoritmo híbrido, utilizou a diversidade da população para favorecer os componentes dos algoritmos evolutivos, a fim de explorar ou intensificar a busca (ROSCA, 1995; LIU; MERNIK; BRYANT, 2009).

1.1 Objetivo

O objetivo deste trabalho é desenvolver um abordagem híbrida para solucionar problemas de otimização não-linear inteira mista. Além disso, é proposto um novo mecanismo de chaveamento entre os algoritmos que compõem essa abordagem, a fim de aumentar a eficácia da busca.

1.2 Metodologia

A fim de analisar estatisticamente as variações do algoritmo Evolução Diferencial, os problemas de otimização contínua foram explorados para selecionar a versão mais eficaz, que será utilizada na busca global. O primeiro protótipo do algoritmo híbrido foi desenvolvido focando problemas de otimização não-linear inteira. Um método para tratar restrições foi acoplado ao algoritmo, a fim de tratar problemas de otimização não-linear inteira com restrições. O algoritmo Nelder-Mead para variáveis inteiras foi utilizado como busca local. Para alcançar o objetivo, além dos algoritmos já mencionados, foi proposto um mecanismo de chaveamento entre busca local e global, baseado na entropia da população.

1.3 Estrutura

O restante do trabalho é organizado da seguinte forma: o Capítulo 2 expõe conceitos preliminares em otimização mono-objetiva, a formulação matemática dos problemas de otimização não-linear inteira mista com restrições e uma revisão da literatura dos métodos bio-inspirados. O Capítulo 3 apresenta os algoritmos Evolução Diferencial e Nelder-Mead, utilizados para busca global e local, respectivamente. O Capítulo 4 apresenta o método *Alfa Constrained* utilizado para tratar restrições, a entropia que é utilizada no chaveamento entre as buscas global e local, finalizando com

a descrição do método proposto. O Capítulo 5 apresenta os resultados de simulações aplicados aos problemas contínua, inteira e mista. Por último, as conclusões e direções para trabalhos futuros de otimização são apresentadas no Capítulo 6.

2 Noções preliminares em otimização e métodos biologicamente inspirados

2.1 Conceitos básicos de otimização mono-objetiva

Algoritmos de otimização são métodos de busca cujo objetivo é obter soluções para um problema de otimização. Apesar da simplicidade do conceito, existem algumas noções fundamentais envolvidas em teoria de otimização (ENGELBRECHT, 2007).

A seguir, são definidos os principais elementos de um problema de otimização.

Definição 1. *Variáveis de decisão*
As variáveis de decisão ($\boldsymbol{x}$) são as incógnitas do problema (solução candidata). Podem ser classificadas de acordo com as seguintes características:

- *Número de variáveis de decisão: se o problema possui apenas uma variável ($D = 1$), então é dito univariado, caso contrário, é dito multivariável ($D > 1$).*
- *Tipo de variáveis de decisão: diz-se que o problema é contínuo se todas as variáveis de decisão são reais. O problema é denominado inteiro se todas as variáveis são inteiras. Um problema onde há variáveis inteiras e contínuas, é dito misto.*

Definição 2. *Função objetivo*
É a função matemática (f) que representa o problema a ser otimizado. Ela pode ser uma função de teste (benchmark), utilizada para analisar o desempenho de um algoritmo, ou, ainda, uma formulação matemática de um problema do mundo real. A função objetivo pode ser classificada de acordo com sua linearidade. Uma função objetivo é denominada linear ou não-linear uma vez que possui todas suas variáveis de grau 1 ou ao menos uma variável maior que 1, respectivamente.

Definição 3. *Restrições*
Todo problema é restrito ao espaço de busca ($\mathcal{S}$), definido pelos limites de cada variável, denominados problemas de otimização irrestrito. No entanto, alguns problemas podem ter restrições no espaço de busca $\mathcal{S}$, denominados problemas de otimização com restrições. As inequações (restrições de desigualdade) e equações (restrições de igualdade), g e h, respectivamente, limitam a região de soluções factíveis. O conjunto factível ($\mathcal{F}$) é o conjunto de soluções que atendem a todas as restrições (ENGELBRECHT, 2007; SCHLÜTER, 2012), definido de acordo com

$$\mathcal{F} = \{(\boldsymbol{x}) \in \mathcal{S} \mid g(\boldsymbol{x}) \leq 0 \wedge h(\boldsymbol{x}) = 0\}$$

onde $\mathcal{F} \subseteq \mathcal{S}$.

A finalidade de um algoritmo de otimização é encontrar valores dentro do espaço de busca $\mathcal{S}$, tal que, otimize o valor da função objetivo. Dado um problema restrito,

a solução candidata deve atender às restrições e otimizar o valor da função objetivo (COELLO, 2002; TAKAHAMA; SAKAI, 2005a; ENGELBRECHT, 2007).

As soluções encontradas pelos métodos de otimização são classificadas de acordo com sua qualidade (ENGELBRECHT, 2007). A seguir, são definidos as principais soluções encontradas.

Definição 4. *Mínimo global*
Uma solução $\boldsymbol{x^*} \in \mathcal{F}$ é o mínimo global de uma função objetiva f se

$$f(\boldsymbol{x^*}) < f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{F}$$

O mínimo global é a melhor entre as soluções candidatas (ENGELBRECHT, 2007).

Definição 5. *Mínimo local estrito*
Uma solução $\boldsymbol{x^*_{\mathcal{N}}} \in \mathcal{N} \subseteq \mathcal{F}$ é um mínimo local estrito de f se

$$f(\boldsymbol{x^*_{\mathcal{N}}}) < f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{N}$$

onde $\mathcal{N} \subseteq \mathcal{F}$ é o conjunto de soluções factíveis na vizinhança de $\boldsymbol{x^*_{\mathcal{N}}}$ (ENGELBRECHT, 2007).

Definição 6. *Mínimo local*
Uma solução $\boldsymbol{x^*_{\mathcal{N}}} \in \mathcal{N} \subseteq \mathcal{F}$ é um mínimo local de f se

$$f(\boldsymbol{x^*_{\mathcal{N}}}) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{N}$$

onde $\mathcal{N} \subseteq \mathcal{F}$ é o conjunto de soluções factíveis na vizinhança de $\boldsymbol{x^*_{\mathcal{N}}}$ (ENGELBRECHT, 2007).

Definição 7. *Solução infactível*
Ocorre somente em problemas de otimização com restrições, uma solução $\boldsymbol{x'} \in \mathcal{U}$ é infactível se

$$\left\{\boldsymbol{x'} \in \mathcal{U} \mid g(\boldsymbol{x}) > 0 \vee h(\boldsymbol{x}) \neq 0\right\}$$

onde $\mathcal{U}$ é o complemento de $\mathcal{F}$ em $\mathcal{S}$.

A Figura 1 mostra soluções que podem ser encontradas pelos métodos de otimização. Nota-se que há um ótimo global, um mínimo local e dois mínimos locais estritos.

A Figura 2 mostra um problema de otimização com três restrições lineares e uma função objetivo não-linear. A área hachurada indica a região infactível e todas as soluções nesta área violam a restrição. Na região factível, estão todas as soluções factíveis. Nota-se o mínimo global irrestrito, que, apesar de possuir o menor valor da função objetiva, é infactível, e o mínimo global localizado na fronteira entre as regiões.

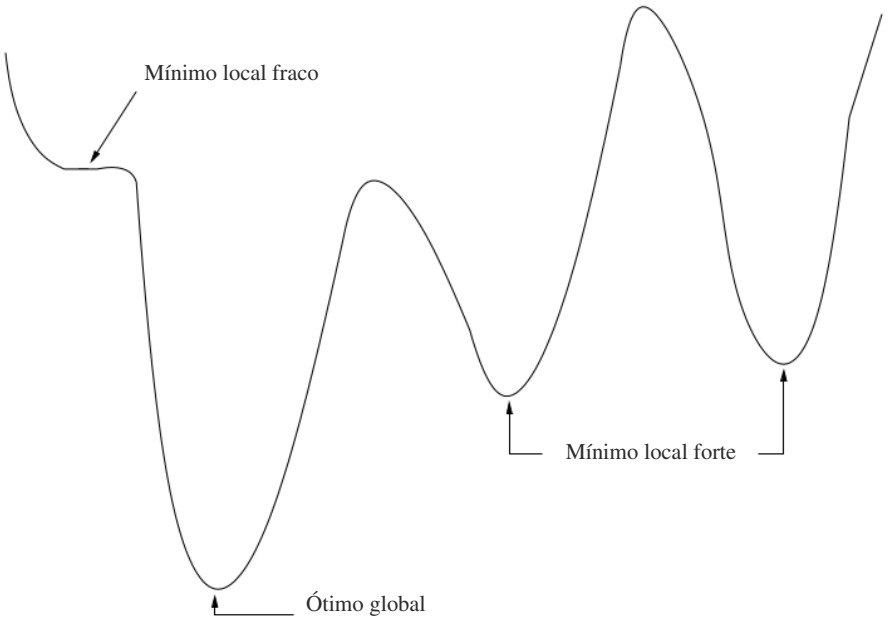


Figura 1 – Tipos de soluções candidatas num problema de otimização sem restrições.

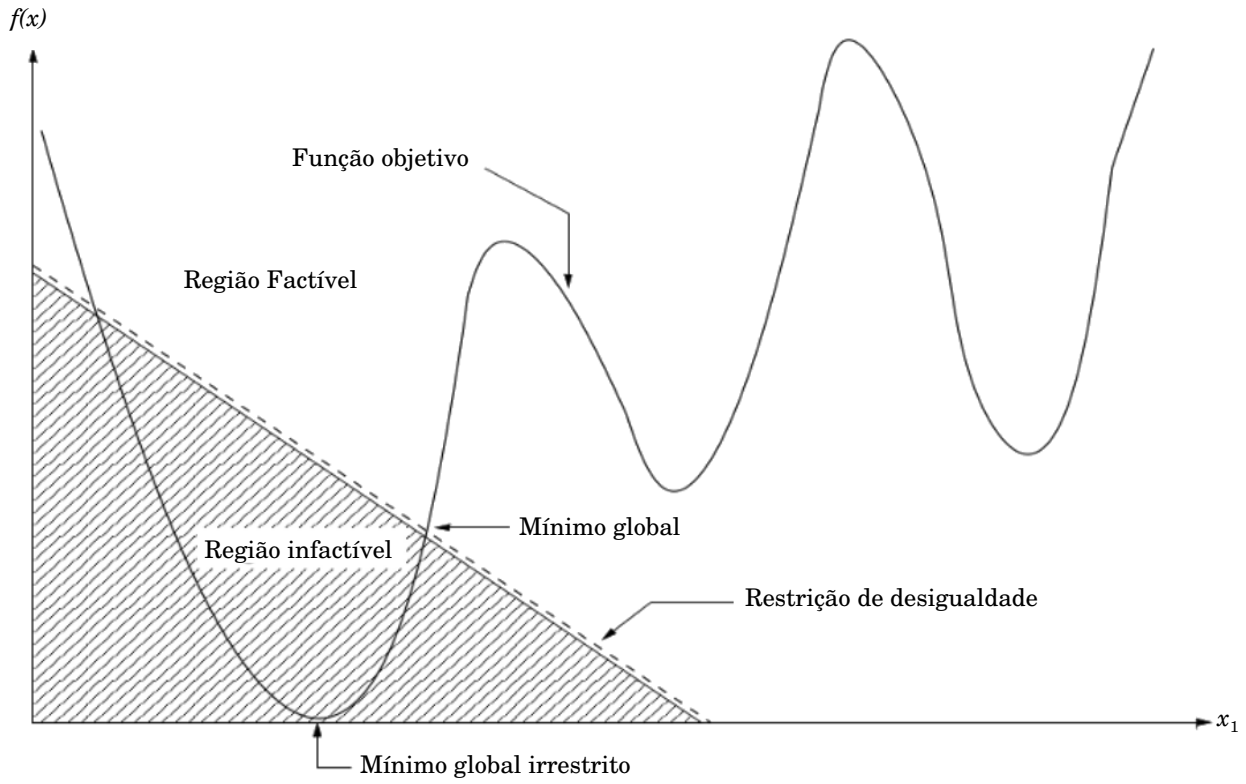


Figura 2 – Ilustração de um problema não-linear com restrição.

2.2 Definição do problema

Problemas de otimização não-linear inteira mista (do inglês *mixed integer non-linear problem*, abreviado por MINLP) é o tipo de problema mais genérico em otimização mono-objetiva. Não há restrições quanto à natureza da função objetivo ou restrições e suas variáveis de decisão podem ser contínuas ou inteiras. Além disso, são difíceis de manusear, pois envolvem dificuldades relacionadas à otimização inteira e contínua (SCHLÜTER, 2012).

Problemas de otimização inteira mista são frequentes nas indústrias e as abordagens para resolução são classificadas em determinísticas e estocásticas (JUN; YUELIN; LINA, 2013). As técnicas estocásticas fazem parte de uma área de pesquisa em desenvolvimento. Por outro lado, as abordagens determinísticas são algoritmos consolidados na literatura, dentre eles o *Branch and Bound* (abreviado por BB), *Outer Approximation* (abreviado por OA) e o método de decomposição de Bender (abreviado por BD) (SCHLÜTER, 2012).

O algoritmo BB foi proposto por Land e Doig (1960) para problemas de otimização combinatórios. Uma vez que as variáveis do problema MINLP podem ser relaxadas, o BB pode ser utilizado. A abordagem gerencia a ramificação das árvores de decisão no espaço de busca, para que algoritmos auxiliares resolvam os subproblemas gerados. No pior caso, o BB poderá enumerar todos os casos do espaço de busca discreto, sendo assim, seu desempenho depende da estrutura do problema e dos algoritmos auxiliares utilizados. Apesar disso, a abordagem garante a solução ótima mesmo para problemas não-convexos (ARORA; HUANG; HSIEH, 1994; SCHLÜTER, 2012).

O algoritmo OA, foi proposto por Duran e Grossmann (1986) para resolver MINLP. A abordagem explora a estrutura do problema, utilizando planos de corte, resolvendo sequências finitas alternativas dos subproblemas e versões relaxadas do problema. Para utilizar o OA, as variáveis discretas e contínuas do problema original devem ser separáveis e todas as funções envolvidas devem ser convexas. Fletcher e Leyffer (1994) propuseram uma extensão do OA em que não é necessário que as variáveis sejam separáveis. Esta abordagem garante a solução ótima somente para problemas convexos (DURAN; GROSSMANN, 1986; FLETCHER; LEYFFER, 1994; SCHLÜTER, 2012).

Geoffrion (1972) propôs o método BD, que é similar ao OA, porém utiliza condições de Karush-Kuhn-Tucker. Ambas abordagens obtêm soluções através de uma série de subproblemas da decomposição do problema original. Assim como o OA, o BD soluciona os subproblemas não-lineares e os subproblemas lineares inteiros sucessivamente. Diferentemente do OA, o BD adiciona somente uma restrição por vez aos problemas inteiros lineares (LIN; WANG; HWANG, 1999; SCHLÜTER, 2012).

Além das abordagens como BB, OA e BD, existem várias outras abordagens que podem ser aplicadas a MINLP. O Plano de corte estendido (do inglês *Extended Cutting Plane*, abreviado por ECP) foi proposto por Westerlund e Pettersson (1995), que considera somente os subproblemas inteiros mistos não lineares, solucionando-os através de cortes no plano iterativamente. Para problemas cujas funções possuem alto grau, esta abordagem necessita de mais iterações comparadas aos demais métodos. Exler e Schittkowski (2007) propuseram o algoritmo Programação Quadrática Sequencial (do inglês *Sequential Quadratic Programming*, abreviado por SQP). Esta abordagem não necessita de relaxar as variáveis discretas do problema original. Além disso, o algoritmo é estabilizado se combinado com BB ou OA. Abramson et al. (2009) propuseram o algoritmo *Mesh Adaptive Direct Search* (abreviado por MADS). Esta abordagem também não necessita de relaxar as variáveis discretas do problema, porém, é preciso conhecer o problema para definir a vizinhança de todas as variáveis discretas (SCHLÜTER, 2012).

Um problema de otimização não-linear inteira mista é definido de acordo com

$$
\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x}, \boldsymbol{y}) \\
\text{Sujeito a} \quad & g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \quad (j = 1, \ldots, q) \\
& h_j(\boldsymbol{x}, \boldsymbol{y}) = 0, \quad (j = q+1, \ldots, m) \\
& \boldsymbol{x} \in \mathbb{R}^{n_{con}}, \boldsymbol{y} \in \mathbb{Z}^{n_{int}}, n_{con}, n_{int}, \in \mathbb{N} \\
& \boldsymbol{x_l} \leq \boldsymbol{x} \leq \boldsymbol{x_u} \quad (\boldsymbol{x_l}, \boldsymbol{x_u} \in \mathbb{R}^{n_{con}}) \\
& \boldsymbol{y_l} \leq \boldsymbol{y} \leq \boldsymbol{y_u} \quad (\boldsymbol{y_l}, \boldsymbol{y_u} \in \mathbb{Z}^{n_{int}})
\end{aligned}
\tag{2.1}
$$

onde $\boldsymbol{x}$ e $\boldsymbol{y}$ são as variáveis contínuas e inteiras do problema, respectivamente, $f(\boldsymbol{x}, \boldsymbol{y})$ é a função objetivo, $g_j(\boldsymbol{x}, \boldsymbol{y})$ representa as q restrições de desigualdade e $h_j(\boldsymbol{x}, \boldsymbol{y})$ representa as $m - q$ restrições de igualdade. Os vetores $\boldsymbol{x_l}, \boldsymbol{y_l}$ e $\boldsymbol{x_u}, \boldsymbol{y_u}$ indicam os limites inferiores e superiores, respectivamente (SCHLÜTER; EGEA; BANGA, 2009).

2.3 Métodos bio-inspirados

O desenvolvimento de algoritmos é impulsionado pelos complexos problemas existentes no mundo real. Inteligência Computacional (abreviada por IC) estuda mecanismos adaptativos que habilitam e/ou facilitam o comportamento inteligente em ambientes dinâmicos. Ela cobre disciplinas que abordam sistemas adaptativos e inteligentes tais como: redes neurais artificiais, computação evolutiva, inteligência coletiva, sistemas imunes artificiais, e sistemas *fuzzy* (ENGELBRECHT, 2007; BROWNLEE, 2011).

As principais técnicas utilizadas para otimização são a computação evolutiva e a inteligência coletiva, ambas baseada em população. Computação evolutiva é inspirada na teoria da evolução em que os indivíduos são gerados através da recombinação entre membros da população. A inteligência coletiva produz inteligência computacional através da interação social entre membros de uma população(BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

2.3.1 Computação evolutiva

Desde a década de 1950, pesquisadores se inspiraram na natureza para propor algoritmos que solucionem problemas computacionais. Uma das inspirações é a evolução das espécies que se adaptam ao ambiente em que vivem, denominada computação evolutiva. Os algoritmos evolutivos são baseados nos princípios da seleção natural e da genética, propostos por Darwin e Mendel, respectivamente, e são utilizados em buscas e otimização (BÄCK; HAMMEL; SCHWEFEL, 1997; CASTRO, 2007).

Através de uma população, os indivíduos são guiados por meio de mecanismos evolutivos para alcançar um determinado objetivo. Dessa forma, um algoritmo evolutivo é composto dos seguintes elementos (CASTRO, 2007; DAS; SUGANTHAN, 2011):

- População: cada indivíduo representa uma possível solução de um dado problema. Eles se reproduzem, gerando novos indivíduos que herdam características dos pais;
- Variação genética: os novos indivíduos tendem a criar novas características através do cruzamento (*crossover*) e da mutação, que por sua vez, geram novas características. Essa modificação do material genético levam os novos indivíduos a explorarem o espaço de busca de um problema;
- Seleção natural: cada indivíduo no ambiente é avaliado, resultando no valor de sua aptidão. Através dessa medida, eles competem entre si para sobreviver e se reproduzir.

Cada iteração corresponde a uma geração e os indivíduos representam as soluções candidatas de um problema de otimização. Os indivíduos se reproduzem através do *crossover*, que é a recombinação entre dois ou mais indivíduos para produzir um ou mais novos indivíduos. A mutação permite o surgimento de novas características nos indivíduos para promover a diversidade. A aptidão (qualidade de uma solução ou *fitness*) é utilizada para a seleção, que determina quais indivíduos serão mantidos na população para as próximas gerações. Como critério de parada é utilizado o número de iterações ou de avaliações da função objetivo (BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

Os principais algoritmos que compõem a computação evolutiva são os Algoritmos Genéticos (do inglês *Genetic Algorithms*, abreviado por GA), Programação Evolutiva (do inglês *Evolutionary Programming*, abreviado por EP) e Estratégia Evolutiva (do inglês *Evolution Strategies*, abreviado por ES). Além disso, os algoritmos Programação Genética (do inglês *Genetic Programming*) e Evolução Diferencial (do inglês *Differential Evolution*, abreviado por DE) são variações dos algoritmos evolutivos (BÄCK; HAMMEL; SCHWEFEL, 1997; DAS et al., 2009).

Holland (1975) apresentou um modelo geral do GA, sendo que uma das aplicações eram problemas de otimização. O desenvolvimento prosseguiu com De Jong (1975) que projetou e formalizou o GA. Goldberg (1985) demostrou a efetividade do GA em sistema dinâmico de controle. Além disso, os problemas de otimização de parâmetros e sistemas de classificadores via aprendizagem foram considerados. Em otimização de parâmetros, o GA obteve resultados muito próximos do mínimo global. Em sistemas de classificadores via aprendizagem (do inglês *Learning classifier systems*, abreviada por LCS) o GA obteve resultados satisfatórios (BÄCK; HAMMEL; SCHWEFEL, 1997).

Fogel (1962) propôs o EP para desenvolver uma máquina de estados finitos, com o objetivo de prever eventos baseados em observações anteriores. Burgin (1969) estendeu o EP para desenvolver uma máquina de estados finitos aplicado ao jogo de soma zero. No início da década de 1990, EP foi reformulado para manipular problemas como aprendizado de máquinas e otimização numérica por Fogel (1991) e Fogel (2006), respectivamente. A única operação realizada pelo algoritmo é a mutação através da adição de um número aleatório extraído de uma distribuição de probabilidade. Fogel, Fogel e Atmar (1991) propuseram o meta-ES, que utiliza uma distribuição de probabilidade normal para efetuar a mutação. Yao e Liu (1996) propuseram o FEP ao substituir a distribuição de probabilidade normal pela Cauchy. Yao, Liu e Lin (1999) propuseram o IFEP que utiliza uma distribuição de probabilidade de Lévy. Apesar das pesquisas realizadas para melhorar o desempenho do EP, este algoritmo é pouco utilizado devido a sua similaridade com o ES (BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

Outra abordagem inspirada nos princípios da seleção natural para resolver problemas de otimização é o ES, que foi introduzido por Rechenberg (1965) e Schwefel (1965) para experimentos no campo de otimização de parâmetros. A primeira versão do ES é baseada em um simples esquema mutação e seleção, onde um indivíduo produz um único descendente. A seleção é utilizada para que os melhores indivíduos sejam mantidos na próxima geração. Schwefel (1981) propôs duas versões do ES. Na primeira versão, denominada $(\mu + \lambda)$-ES, são selecionados μ pais para gerar λ novos indivíduos através da mutação e *crossover*. A segunda versão, denominada (μ, λ)-ES, a seleção ocorre entre λ descendentes, sendo que seus pais são descartados, independente de sua aptidão. A mutação no ES é realizada através de uma distribuição de probabilidade normal com média zero e desvio padrão σ. Uma vez que esses parâmetros influenciam diretamente no processo de busca, várias abordagens foram propostas. A mais simples consistem em manter o desvio padrão constante. Outras abordagens propõem que o σ se ajuste dinamicamente dependendo do número de iterações ou das respostas do processo de busca. As principais melhorias na mutação do ES, foram desenvolvidas por Rechenberg (1973), Herdy (1992), Schwefel e Rudolph (1995) e Sebag, Schoenauer e Ravisé (1997). Hansen, Ostermeier e Gawelczyk (1995) propuseram Adaptação da Matriz de Covariância da Estratégia Evolutiva (do inglês *Covariance Matrix Adaptation Evolution Strategy*, abreviado por CMA-ES) que se tornou um dos algoritmos mais utilizados para otimização global (BÄCK; HAMMEL; SCHWEFEL, 1997; BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

O GP é uma extensão do GA proposto por Koza (1992) para manipular estruturas de dados como grafos e árvores. Os indivíduos não possuem tamanho fixo e suas expressões são representadas através de árvores, que podem ter altura fixa ou variáveis, onde os nós terminais armazenam as constantes e variáveis enquanto os nós internos contém as operações aritméticas. A recombinação é feita utilizando sub-árvores que podem gerar sub-árvores de tamanhos diferentes em posições distintas. A mutação é realizada no GP e a mais comum é a mutação de uma sub-árvore, que é selecionada, aleatoriamente, e substituída por outro segmento. Esta abordagem é utilizada em aplicações como aprendizagem de máquinas, predição e classificação (BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

O DE foi proposto por Storn e Price (1995) como uma ferramenta para resolver o problema polinomial de Chebyshev e se tornou um dos algoritmos mais utilizados para problemas de otimização contínua. Na primeira conferência internacional em computação evolutiva (*International Conference on Evolutionary Computation* (ICEO)) em 1996, apesar de obter o terceiro lugar, tornou-se um dos melhores algoritmos para otimização contínua. Price (1997) apresentou o DE na segunda ICEO, onde se tornou um dos melhores entre os algoritmos concorrentes. Em 2005, DE conquistou a segunda posição, seguido de uma variação auto adaptativa (do inglês *Self-adaptive*

Differential Evolution, abreviada por SaDE), proposto por Qin, Huang e Suganthan (2009). Embora o CMA-ES tenha superado o DE e SaDE em 2005, outras variações foram propostas (DAS; SUGANTHAN, 2011). Dentre elas o jDE (BREST et al., 2006), ODE (RAHNAMAYAN; TIZHOOSH; SALAMA, 2008), DEGL (DAS et al., 2009) e JADE (ZHANG; SANDERSON, 2009).

2.3.2 Inteligência coletiva

Inteligência coletiva é o estudo de sistemas computacionais que emergem através da cooperação entre agentes homogêneos no ambiente, tais como cardume de peixes, colônia de formigas, bando de pássaros, entre outros. Nestes grupos, a inteligência é descentralizada, organizada e distribuída, ou seja, os indivíduos interagem trocando informações localmente, que se propaga para o grupo inteiro para solucionar um objetivo global (ENGELBRECHT, 2007; BROWNLEE, 2011).

Formalmente, um enxame pode ser definido como um grupo de agentes que se comunicam entre si, de acordo com seu ambiente local. Inteligência coletiva refere-se ao comportamento para resolução de problema que emerge da interação entre estes agentes, ao passo que, algoritmos baseados em enxame referem-se as abordagens provenientes de tal comportamento (ENGELBRECHT, 2007).

Diante do comportamento emergente dos grupos sociais, vários algoritmos foram propostos. Os principais são Otimização via Enxame de Partículas proposto por Kennedy e Eberhart (1995) (do inglês *Particle Swarm Optimization*, abreviado por PSO) e Otimização via Colônia de Formigas proposto por Dorigo (1992) (do inglês *Ant Colony Optimization*, abreviado por ACO). Além disso, foram propostos algoritmos baseados no comportamento de abelhas e bactérias (CASTRO, 2007; BROWNLEE, 2011).

Kennedy e Eberhart (1995) se inspiraram num modelo social simplificado para propor o algoritmo PSO. Para testar a abordagem, foram utilizados problemas de otimização não-linear e treinamento de redes neurais artificiais. Li (2010) propôs uma variação do PSO que utiliza uma topologia em anel. Para testar o desempenho do algoritmo, foram utilizados problemas de otimização não-linear, e o PSO obteve resultados competitivos comparado com outros algoritmos evolutivos (POLI; KENNEDY; BLACKWELL, 2007).

A fim de controlar efetivamente o comportamento do PSO, Shi e Eberhart (1998) acrescentaram um termo de inércia à equação da velocidade do PSO. Quando este parâmetro assume valores altos, 0.9 por exemplo, o algoritmo favorece a exploração, ao passo que, reduzindo até valores baixos, 0.4 por exemplo, a intensificação é favorecida. Várias estratégias foram propostas para a variação do termo de inércia. Porém, sabe-se que a escolha apropriada dos parâmetros influencia no desempenho do algoritmo

(POLI; KENNEDY; BLACKWELL, 2007).

Analisando o comportamento do PSO, foi notado que as partículas atingem níveis inaceitáveis com poucas iterações quando não há controle de velocidade. Kennedy (1998) notou regularidade das partículas quando a soma das acelerações está entre 0 e 4. Em consequência, Clerc e Kennedy (2002) propuseram um parâmetro de constrição que restringe a velocidade das partículas do enxame, a fim de prevenir a explosão. Na literatura do PSO, esta versão é denominada a versão canônica.

Em todas as versões citadas, uma partícula sofre influência das melhores posições encontradas previamente por ela mesma (p_{best}) e pela população (g_{best}), ou seja, o p_{best} dos demais indivíduos não são utilizadas. Kennedy e Mendes (2002) analisaram a comunicação entre os indivíduos da população e propuseram o *Fully informed Particle Swarm* (abreviado por FIPS), em que uma partícula é afetada por seus vizinhos. O desempenho do FIPS, com os parâmetros sugeridos na literatura, é mais eficaz que o PSO canônico, uma vez que encontra melhores soluções em menos iterações. Entretanto, esta versão depende da topologia da população (POLI; KENNEDY; BLACKWELL, 2007).

Kennedy (2003) propôs o *Bare-bones* PSO em que a velocidade é eliminada e as partículas são amostradas de uma distribuição de probabilidade normal. Os primeiros experimentos utilizaram a distribuição de probabilidade normal e o desempenho do algoritmo foi equivalente em alguns problemas, porém, menos efetivo em outros. Richer e Blackwell (2006) substituíram a distribuição de probabilidade normal pela de Lévy, que possui o parâmetro α que, ora se aproxima da distribuição de Cauchy, a medida que se aproxima de 1, ora se aproxima da distribuição normal, a medida que se aproxima de 2. Os autores concluíram que ao se utilizar $\alpha = 1.4$, o *Bare-bones* PSO possui o desempenho semelhante ao PSO canônico (POLI; KENNEDY; BLACKWELL, 2007).

Outro grupo social que serviu de inspiração para o desenvolvimento de uma abordagem baseada em inteligência coletiva, são as colônias de formigas. Na busca de alimentos, as formigas se comunicam através de trilhas de feromônios que são formadas entre a fonte de alimento e seu ninho. Dorigo (1992) explorou este comportamento para propôr o algoritmo Otimização via Colônia de Formigas (do inglês *Ant Colony Optimization*, abreviado por ACO) para solucionar problemas de otimização combinatória.

A primeira versão do ACO, denominada *Ant System* (abreviada por AS), é aplicada ao Problema do Caixeiro Viajante (do inglês *Traveling Salesman Problem*, abreviado por TSP), porém, mesmo tendo obtido resultados promissores, não superou as principais abordagens aplicadas a este problema. Apesar dos resultados, a inspiração nas colônias de formigas encorajou o desenvolvimento de várias extensões que melhoraram o desempenho do algoritmo, entre elas Elitista AS, *Ant-Q*, Sistema de

Colônia de Formigas, Max-Min AS, AS baseado em *ranking*, ANTS e *Hyper-cube* AS. Todas essas versões mantém as características do ACO, tais como procedimentos de construção de solução e evaporação. No entanto, diferem na atualização do feromônio e gerenciamento da evaporação do feromônio nas trilhas (DORIGO; STÜTZLE, 2004).

Uma vez que o ACO foi desenvolvido para problemas de otimização combinatória, ele não se aplica a problemas contínuos. Sendo assim, Socha e Dorigo (2008) desenvolveram uma versão do ACO para manipular variáveis contínuas nativamente, denominado $\text{ACO}_{\mathbb{R}}$. Entre as principais mudanças está a substituição da distribuição de probabilidade discreta para contínua e a representação do feromônio. Os algoritmos utilizados para comparação foram ES, CMA-ES, o algoritmo de estimação de densidade iterativa (do inglês *Iterated Density Estimation*, abreviado por IDEA) e o algoritmo misto Bayesiano (do inglês *Mixed Bayesian*, abreviado por MBOA). $\text{ACO}_{\mathbb{R}}$ apresentou um desempenho competitivo e superou os algoritmos utilizados para comparação em 4 dos 10 problemas. Leguizamón e Coello (2010) propuseram o $\text{DACO}_{\mathbb{R}}$ que favorece a exploração no espaço de busca em problemas contínuos com muitas variáveis. Liao et al. (2011) propuseram o $\text{IACO}_{\mathbb{R}}$-LS que aumenta o tamanho da população para favorecer a exploração, ao passo que, uma busca local é aplicada para favorecer a intensificação. Liao et al. (2014) propuseram o $\text{UACO}_{\mathbb{R}}$ que utiliza os componentes do $\text{ACO}_{\mathbb{R}}$, $\text{DACO}_{\mathbb{R}}$ e $\text{IACO}_{\mathbb{R}}$-LS. Ele foi testado utilizando problemas de otimização contínuos não-lineares e se mostrou competitivo com os principais algoritmos para otimização contínua.

A primeira extensão do ACO para manipular problemas de otimização inteira mista foi proposta por Socha (2004), que consiste em uma combinação entre o tradicional ACO e o $\text{ACO}_{\mathbb{R}}$. Schlüter, Egea e Banga (2009) propuseram o ACO_{mi}, que utiliza uma função densidade de probabilidade contínua e discreta para manipular variáveis contínuas e inteiras, respectivamente. O desempenho do algoritmo foi avaliado utilizando *benchmarks* da literatura e um problema do mundo real, e os resultados obtidos refletem o sucesso da abordagem.

3 Métodos de busca global e local

3.1 Algoritmo Evolução Diferencial

Evolução Diferencial é um algoritmo promissor para otimização de problemas multimodais com variáveis contínuas. Ele opera com a mesma estrutura de outros algoritmos evolutivos. Vários problemas de otimização de diversas áreas passaram a ser resolvidos pelo DE. Algumas características favorecem seu uso, tais como (DAS; SUGANTHAN, 2011):

1. Simples implementação comparado aos demais algoritmos evolutivos e de inteligência coletiva;
2. Apesar de sua simplicidade, o DE possui o desempenho melhor que outros algoritmos como PCX, MA-S2, ALEP, CPSO-H (DAS; SUGANTHAN, 2011);
3. O algoritmo possui poucos parâmetros para ajustar;
4. O DE possui baixa complexidade comparado a algoritmos mais competitivos como CMA-ES.

O algoritmo inicia com uma população de NP indivíduos e D variáveis que são gerados aleatoriamente utilizando uma distribuição de probabilidade uniforme, que representam soluções candidatas. A população do DE pode ser representada de acordo com

$$X_i^G = \left[x_{1,i}^G, x_{2,i}^G, \ldots, x_{D,i}^G\right] \tag{3.1}$$

onde $G = 1, 2, \ldots, G_{max}$ indicam as gerações (iterações) e $i = 1, 2, \ldots, NP$ indicam os indivíduos da população (STORN; PRICE, 1995).

Cada variável do problema possui limites inferiores e superiores. A população em $G = 1$ é inicializada aleatoriamente e deve cobrir todo o espaço de busca utilizando $X_{min} = \{x_{1,min}, x_{2,min}, \ldots, x_{D,min}\}$ e $X_{max} = \{x_{1,max}, x_{2,max}, \ldots, x_{D,max}\}$, que representam os limites inferiores e superiores, respectivamente. A inicialização da população é feita de acordo com

$$x_{j,i} = x_{j,min} + rand_{i,j}[0,1] \cdot (x_{j,max} - x_{j,min}) \tag{3.2}$$

onde $rand[0,1]$ é um número aleatório gerado a partir de uma distribuição uniforme $U(0,1)$ (STORN; PRICE, 1995; DAS; SUGANTHAN, 2011).

3.1.1 Algoritmo Evolução Diferencial padrão

Diferente de outros algoritmos evolutivos, DE aplica uma diferença entre os vetores de variáveis para explorar o espaço de busca. A mutação representa uma mudança, ou perturbação, entre elementos aleatórios, e é a principal operação realizada

pelo DE. O vetor de variáveis atual é chamado *target*, o vetor obtido através da mutação é chamado *donor* e, por último, o vetor obtido através da combinação entre o *target* e *donor* é chamado *trial* (DAS; SUGANTHAN, 2011). A mutação é definida de acordo com

$$V_i = X_{r_1} + F(X_{r_2} - X_{r_3}) \tag{3.3}$$

onde r_1, r_2 e r_3 são indivíduos escolhidos aleatoriamente na população de forma que $r_1 \neq r_2 \neq r_3$, F é chamado fator de mutação assumindo valores entre 0.4 e 1, que controla a ampliação da diferença entre X_{r_2} e X_{r_3} (STORN; PRICE, 1995).

A Figura 3 mostra a representação gráfica da mutação. Os indivíduos X_2, X_6 e X_9 foram selecionadas aleatoriamente. A diferença entre X_6 e X_9 é multiplicada pelo fator de mutação F, e somado a X_2 para formar o vetor *donor*.

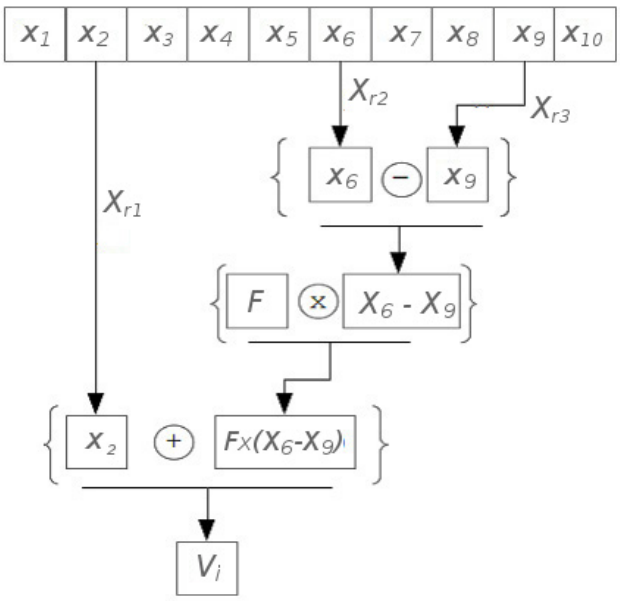


Figura 3 – Representação gráfica da mutação no DE.

Para aumentar a diversidade entre os indivíduos, o *crossover* é aplicado após a mutação, através da troca de variáveis entre o *target* e *donor*. Esta operação é definida de acordo com

$$U_{i,j} = \begin{cases} V_{i,j}, & \text{se } U(0,1) \leq C_r, \text{ ou } j = j_{rand} \\ X_{i,j}, & \text{caso contrário} \end{cases} \tag{3.4}$$

onde C_r é a taxa de *crossover*, j_{rand} é um número aleatório entre $[1, D]$, para garantir que pelo menos uma variável de V_i faça parte de U_i (STORN; PRICE, 1995; DAS; SUGANTHAN, 2011).

Considere um problema de otimização bidimensional, o vetor *trial* pode ser formado quando ambas variáveis de V_i são transmitidas adiante, ou apenas uma delas. A Figura 4 mostra o *crossover* sobre o indivíduo $\bar{X}_i$, U_i representa quando

ambas variáveis são transmitidas para a próxima geração, $U_i^{'}$ e $U_i^{''}$ representam quando apenas uma das variáveis são passadas adiante.

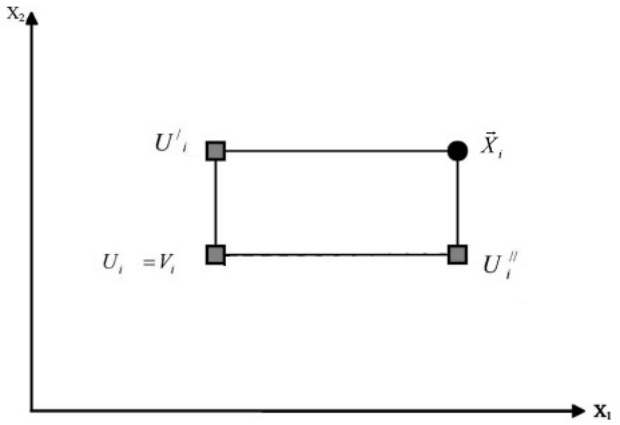


Figura 4 – Possíveis vetores *trials* em um espaço bi-dimensional.

Após a mutação e *crossover*, a seleção é realizada para garantir que somente os melhores indivíduos prossigam para a próxima geração e é definida de acordo com

$$X_i = \begin{cases} U_i, & \text{se } f(U_i) < f(X_i) \\ X_i, & \text{caso contrário} \end{cases} \tag{3.5}$$

onde U_i e X_i são os vetores *trial* e *target*, respectivamente (STORN; PRICE, 1995).

3.1.2 Algoritmo Evolução Diferencial *local-to-best*

Storn e Price (1995) sugeriram algumas variações na mutação do DE, que são generalizadas por DE/p/q, onde *p* define o vetor base a ser mutado e *q* determina o número de diferenças realizadas. A mutação definida na Equação 3.3 é utilizada no DE/*rand*/1. As demais operações são conduzidas normalmente.

A mutação da versão DE/*local-to-best*/1 é definida de acordo com

$$V_i = X_i + F(X_{best} - X_i) + F(X_{r_2} - X_{r_3}) \tag{3.6}$$

onde X_i é o i-ézimo indivíduo da população, X_{best} é o melhor indivíduo da população e X_{r_2} e X_{r_3} são indivíduos selecionados aleatoriamente de forma que $i \neq r_2 \neq r_3$ (STORN; PRICE, 1995).

3.1.3 Algoritmo Evolução Diferencial com topologia global-local

Evolução diferencial com topologia global local (DEGL) utiliza uma topologia em anel, proposta por Li (2010). A mutação é realizada através de uma combinação linear entre a vizinhança local (vetor local) e a vizinhança global (vetor global), que equilibra a

capacidade de exploração/intensificação do DE (DAS et al., 2009; DAS; SUGANTHAN, 2011)

A Figura 5 mostra as topologias em anel e a vizinhança global utilizadas pelo DEGL.

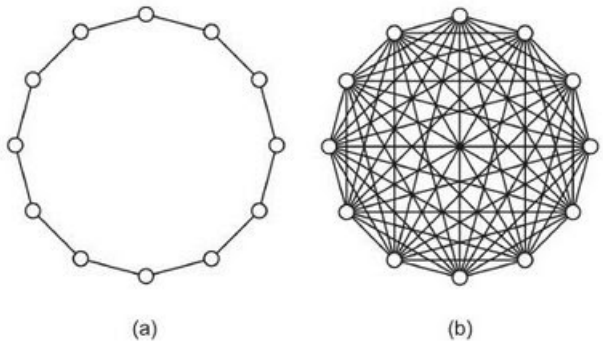


Figura 5 – Topologias local e global.

O vetor local utiliza uma vizinhança de tamanho k, ou seja, o indivíduo i possui a vizinhança $[i-k, i+k]$, e é calculado de acordo com

$$L_i = X_i + F_\alpha(X_{best_i} - X_i) + F_\beta(X_{r_1} - X_{r_2}) \tag{3.7}$$

onde X_{best_i} é o melhor indivíduo da vizinhança, r_1 e r_2 são indivíduos escolhidos aleatoriamente na vizinhança de forma que $i \neq r_1 \neq r_2$, F_α e F_β são os fatores de mutação (DAS et al., 2009).

O vetor global é calculado de acordo com

$$G_i = X_i + F_\alpha(X_{gbest} - X_i) + F_\beta(X_{q_1} - X_{q_2}) \tag{3.8}$$

onde X_{gbest} é o melhor indivíduo da população, q_1 e q_2 são escolhidos aleatoriamente de forma que $i \neq q_1 \neq q_2$, F_α e F_β são os fatores de mutação (DAS et al., 2009).

O vetor mutante é calculado através da combinação linear entre L_i e G_i de acordo com

$$V_i = wG_i + (1-w)L_i \tag{3.9}$$

onde w pondera as componentes L_i e G_i e é calculado de acordo com

$$w = \frac{\text{número da iteração}}{\text{número total de iterações}} \tag{3.10}$$

O fator peso w é adicionado para balancear a exploração/intensificação, portanto, quanto menor o valor de w, maior é a capacidade de exploração, e vice versa.

O pseudo-código do DE com topologia global-local é apresentado no algoritmo 1.

Algoritmo 1: Evolução diferencial com topologia global-local

```
início
    inicialização da população de acordo com a equação (3.2)
    repita
        para i = 1 to NP faça
            sortear r1, r2 tal que i ≠ r1 ≠ r2
            calcular o vetor local com (3.7)
            sortear q1, q2 tal que i ≠ q1 ≠ q2
            calcular o vetor global com (3.8)
            mutação de acordo com (3.9)
            crossover de acordo com (3.4)
            seleção de acordo com (3.5)
        fim
    até critério de parada seja satisfeito;
fim
```

3.1.4 Algoritmo Evolução Diferencial para variáveis binárias

A mutação efetuada pelo DE assume valores contínuos como resultado, considerando que o fator de mutação F, ou as variáveis, são valores decimais. Dessa forma, Datta e Figueira (2013) adaptaram a mutação para manipular variáveis binárias.

Dado um problema de otimização binário, a Tabela 1 lista os possíveis valores de X_{r_1}, X_{r_2} e X_{r_3} nas colunas (1), (2) e (3), respectivamente, e os resultados são listados na coluna (4). Nota-se que as linhas (1)-(4) assumem valores binários, enquanto as demais linhas resultam valores diferentes de 0 ou 1. As linhas (5) e (8) são iguais a F e $1 - F$, respectivamente, que pertencem ao intervalo entre 0 e 1, portanto, assumem valor 1, como é mostrado na coluna (6). Por outro lado, as linhas (6) e (7) são iguais a $-F$ e $1 + F$, respectivamente, que estão fora do intervalo entre 0 e 1, desta forma, atribui-se o valor 0, como é mostrado na coluna (6) (DATTA; FIGUEIRA, 2013). Em resumo, para o resultado da mutação com valores maiores que 0 e menor ou igual a 1, resulta 1. Por outro lado, para o resultado da mutação com valor menor ou igual 0 e maior que 1, resulta 0.

Apesar das suposições acima, as variáveis binárias podem perder a diversidade e cair em mínimos locais. Para contornar este problema, é adicionado um fator aleatório à mutação, como é mostrado na coluna (7) da Tabela 1. O resultado da mutação é negado, de 0 para 1, ou vice-versa, caso um número gerado de uma distribuição uniforme seja menor que a probabilidade de mutação (p_m), a uma taxa de 15%. Além disso, se todos os indivíduos da população convergirem para uma única solução, ele é negado a uma probabilidade de 5%. Ambas as taxas foram sugeridas por Datta e Figueira (2013).

Uma das principais características dos algoritmos evolutivos é sua capacidade

Tabela 1 – Combinações possíveis para a mutação em problemas binários.

	X_{r_1}	X_{r_2}	X_{r_3}				V_i
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
(1)	0	0	0	0	0	0	0, 1 se $U(0,1) < p_m$
(2)	0	1	1	0	0	0	0, 1 se $U(0,1) < p_m$
(3)	1	0	0	1	1	1	1, 0 se $U(0,1) < p_m$
(4)	1	1	1	1	1	1	1, 0 se $U(0,1) < p_m$
(5)	0	1	0	F	(0,1)	1	1, 0 se $U(0,1) < p_m$
(6)	0	0	1	$-F$	< 0	0	0, 1 se $U(0,1) < p_m$
(7)	1	1	0	1+F	> 1	0	0, 1 se $U(0,1) < p_m$
(8)	1	0	1	1-F	(0,1)	1	1, 0 se $U(0,1) < p_m$

de realizar busca global. Apesar das melhorias propostas para balancear a exploração/intensificação dos algoritmos evolutivos, eles não utilizam uma solução pivô, assim como os mecanismos de busca local. A busca local examina um conjunto de indivíduos na vizinhança de uma solução candidata e substitui por uma outra melhor, se houver. Portanto, uma alternativa para encontrar soluções melhores é aplicar um mecanismo com habilidades em busca local (BÄCK; HAMMEL; SCHWEFEL, 1997; KRASNOGOR; HART; SMITH, 2004).

3.2 Algoritmo Nelder-Mead

O algoritmo Nelder-Mead (NM) surgiu do interesse em solucionar problemas de otimização não-linear do mundo real. Eles são difíceis de resolver devido a multimodalidade de um problema e à complexidade de se obter a primeira derivada de uma função f (WRIGHT, 2012).

O NM começou a ser vislumbrado por John Nelder ao assistir uma palestra de William Spendley sobre um método que se adapta a superfície, denominado *simplex*, proposto por Spendley, Hext e Himsworth (1962). Apesar do termo *simplex* não fazer referência ao algoritmo utilizado para problemas lineares, proposto por Dantzig, é apropriado pois a abordagem utiliza $n+1$ vértices num espaço n-dimensional a fim de explorar a superfície da função objetivo (WRIGHT, 2012).

Nelder-Mead é um algoritmo capaz de alongar a área de busca, a fim de escapar de mínimos locais, e contrair-se para convergir para o ótimo global. Implementações e resultados computacionais mostraram que o NM superou o método de *Powell* (POWEL, 1964; NELDER; MEAD, 1965; WRIGHT, 2012).

3.2.1 Algoritmo Nelder-Mead para variáveis contínuas

Nelder-Mead é um algoritmo utilizado para minimização de funções não-lineares. Dada uma função de D variáveis, NM utiliza $D+1$ pontos e substitui o pior entre eles. Seja y_i o valor da função objetivo no ponto P_i, y_h e y_l é a maior e menor aptidão, respectivamente (NELDER; MEAD, 1965).

Inicialmente calcula-se o centróide de acordo com

$$\bar{P} = \frac{1}{D}\sum_{i=1}^{D} P_i \qquad (i \neq h) \tag{3.11}$$

A seguir, NM procura substituir P_h através das operações de reflexão, contração, expansão e encolhimento. A primeira operação executada é a reflexão e é calculada de acordo com

$$P^* = (1+\alpha)\bar{P} - \alpha P_h \tag{3.12}$$

onde α é o coeficiente de reflexão, $\bar{P}$ é o centróide e P_h é o ponto que possui o maior valor da função objetivo. Em outras palavras, P^* é o segmento que une $\bar{P}$ e P_h, do lado oposto a $\bar{P}$, e, portanto, $[P^*\bar{P}] = \alpha[P_h\ \bar{P}]$ (NELDER; MEAD, 1965). A Figura 6 mostra P^* gerado através da reflexão.

Se um novo mínimo local for encontrado ao refletir P_h, tenta-se melhorar P^* através da operação de expansão, que é calculada de acordo com

$$P^{**} = \gamma P^* + (1-\gamma)\bar{P} \tag{3.13}$$

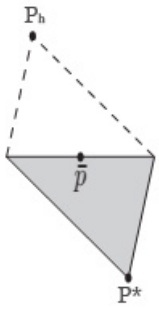


Figura 6 – Representação gráfica da operação de reflexão.

onde γ é o coeficiente de expansão, P^* é o ponto refletido e $\bar{P}$ é o centróide. A Figura 7 mostra o vértice P^{**} gerado através da expansão.

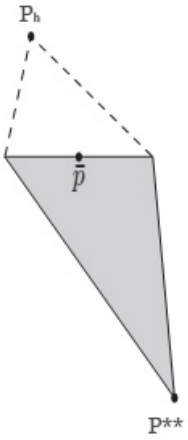


Figura 7 – Representação gráfica da operação de expansão.

Uma vez que a operação de reflexão ou expansão gerou um novo mínimo local, o vértice P_h é substituído por P^* ou P^{**}, respectivamente. Caso contrário, a operação de contração é executada de acordo com

$$P^{**} = \beta P_h + (1 - \beta)\bar{P} \tag{3.14}$$

onde β é o coeficiente de contração, P_h é o vértice que possui a maior aptidão e $\bar{P}$ é o centróide. As Figuras 8 e 9 mostram a operação de contração sobre o ponto P_h e P^*, respectivamente.

A última operação que o algoritmo NM poderá executar é o encolhimento em direção a P_l. Esta operação é definida de acordo com

$$P_i = \frac{(P_i + P_l)}{2} \tag{3.15}$$

onde P_i é o vértice i e P_l é o vértice com menor aptidão.

A Figura 10 mostra a operação de encolhimento em direção a P_l.

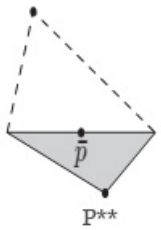


Figura 8 – Representação gráfica da operação de contração sobre o ponto P^*.

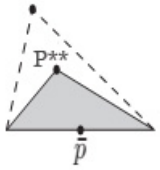


Figura 9 – Representação gráfica da operação de contração sobre o ponto P_h.

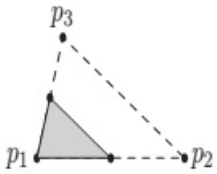


Figura 10 – Representação gráfica do encolhimento em direção a P_l.

O pseudo-código do MN para variáveis contínuas é apresentado no algoritmo 2.

O algoritmo NM é utilizado para minimização de funções contínuas não-lineares, sendo assim, é necessário estender suas operações para tratar variáveis inteiras.

3.2.2 Algoritmo Nelder-Mead para variáveis inteiras

Brea (2013) propôs uma variação do NM, denominada *Integer Mixed Simplex Algorithm*, para problemas de otimização mista não-linear, que é composto por funções no espaço D-dimensional com variáveis reais e inteiras. As operações propostas por Nelder e Mead (1965) são efetuadas sobre pontos reais, sendo assim, um novo grupo de operações é aplicado para pontos discretos.

Seja y_i o valor da função objetivo no ponto P_i, y_h e y_l é o maior e menor valor da função objetivo, respectivamente. O centróide é calculado de acordo com a equação 3.11.

A reflexão é definida de acordo com

$$P^* = P_h + \alpha \cdot \mu \cdot sgnd(\bar{P} - P_h) \tag{3.16}$$

Algoritmo 2: Algoritmo Nelder-Mead para variáveis contínuas

início
  repita
    calcular o centróide dos $D_i(i \neq h)$ com (3.11)
    calcular reflexão (P^*) com (3.12)
    se $y^* < y_l$ **então**
      // ótimo local encontrado
      calcular expansão (P^{**}) com (3.13)
      se $y^{**} < y_l$ **então**
        substituir P_h por P^{**}
      senão
        substituir P_h por P^*
      fim
    senão
      // ponto refletido entre $P_i, i \neq l, i \neq h$
      se $y^* > y_i, i \neq h$ **então**
        // contração do melhor entre P_h e P^*
        se $y^* > y_h$ **então**
          calcular P^{**} com (3.14)
        senão
          substituir P_h por P^*
          calcular P^{**} com (3.14)
        fim
        se $y^{**} > y_h$ **então**
          encolhimento com (3.15)
        senão
          substituir P_h por P^{**}
        fim
      senão
        substituir P_h por P^*
      fim
    fim
  até critério de parada seja satisfeito;
fim

onde α é o coeficiente de reflexão, $\mu = \lceil |\bar{P} - P_h| \rceil$ é a distância entre o centróide e o pior vértice, arredondado para o próximo inteiro, e a função sinal $sgnd(k)$ é definida de acordo com

$$sgnd(k) = \begin{cases} 1, & \text{se } k > 0 \\ 0, & \text{se } k = 0 \\ -1, & \text{se } k < 0 \end{cases} \tag{3.17}$$

A expansão é definida de acordo com

$$P^{**} = P_h + \beta \cdot \mu \cdot sgnd(\bar{P} - P_h) \tag{3.18}$$

onde β é o coeficiente de expansão.

A contração é definida de acordo com

$$P^{**} = P_h + \gamma \cdot \mu \cdot sgnd(\bar{P} - P_h) \tag{3.19}$$

onde γ é o coeficiente de contração.

A operação de encolhimento é similar à aquela executada pelo NM para variáveis contínuas, porém, ela pode assumir valores contínuos como resultado. Portanto, o encolhimento utilizado pelo NM para variáveis inteiras é definido de acordo com

$$P_i = \left[\frac{(P_i + P_l)}{2}\right] \tag{3.20}$$

onde $[\cdot]$ representa o operador arredondamento para o inteiro mais próximo (BREA, 2013).

O pseudo-código do MN para variáveis inteiras é apresentado no algoritmo 3.

O algoritmo NM possui habilidades em busca local, ou seja, ele analisa a vizinhança de uma solução pivô. Para isto, são necessários um conjunto de vértices, que, se escolhidos inadequadamente, pode levar a busca a mínimos locais. Uma vez que as buscas global e local podem ser realizadas utilizando os algoritmos DE e NM, respectivamente, é possível uni-los em um algoritmo híbrido, a fim de se obter soluções melhores.

Algoritmo 3: Algoritmo Nelder-Mead para variáveis inteiras

início
  repita
    calcular o centróide dos $D_i(i \neq h)$ com (3.11)
    calcular a reflexão (P^*) com (3.16)
    se $y^* < y_l$ **então**
      // ótimo local encontrado
      calcular expansão (P^{**}) com (3.18)
      se $y^{**} < y_l$ **então**
        substituir P_h por P^{**}
      senão
        substituir P_h por P^*
      fim
    senão
      // ponto refletido entre $P_i, i \neq l, i \neq h$
      se $y^* > y_i, i \neq h$ **então**
        // contração do melhor entre P_h e P^*
        se $y^* > y_h$ **então**
          calcular P^{**} com (3.19)
        senão
          substituir P_h por P^*
          calcular P^{**} com (3.19)
        fim
        se $y^{**} > y_h$ **então**
          encolhimento com (3.20)
        senão
          substituir P_h por P^{**}
        fim
      senão
        substituir P_h por P^*
      fim
    fim
  até critério de parada seja satisfeito;
fim

4 Método proposto para solucionar problemas de otimização não-linear inteira mista

4.1 Método para tratar restrições

A computação evolutiva recebeu muita atenção devido a sua eficácia em resolver problemas de otimização. No entanto, não existia uma linha diretiva para tratar soluções infactíveis em problemas de otimização com restrições (MICHALEWICZ, 1995).

Os algoritmos utilizados para resolver problemas de otimização com restrições não tinham habilidades suficientes para resolver problemas multimodais e ficavam presos a mínimos locais. Em problemas com restrições de igualdade, as soluções obtidas eram inadequadas, uma vez que estas restrições eram convertidas em restrições de desigualdades e retornavam soluções inconsistentes. Por último, os algoritmos utilizados não eram estáveis nem eficazes, porque, em alguns casos, não eram capazes de superar a aleatoriedade (TAKAHAMA; SAKAI, 2006).

O espaço de busca $\mathcal{S}$ de um problema de otimização com restrições é formado por subconjuntos disjuntos de soluções factíveis e infactíveis, $\mathcal{F}$ e $\mathcal{U}$, respectivamente. A Figura 11 mostra as soluções infactíveis e, f e b, as soluções factíveis c, a, d e x, no espaço de busca $\mathcal{S}$ (MICHALEWICZ, 1995).

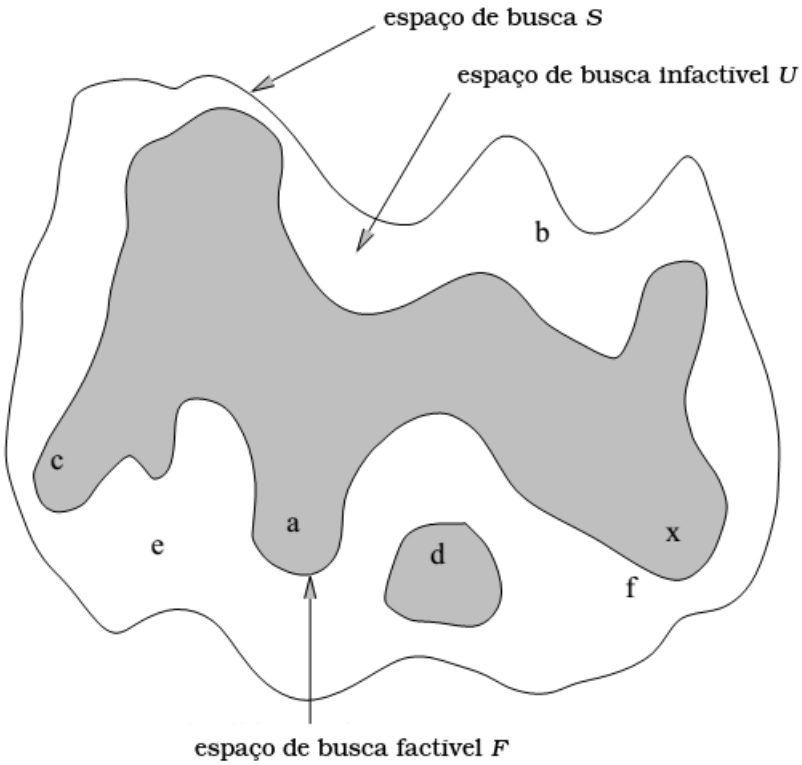


Figura 11 – Espaço de busca em um problema de otimização com restrições.

Várias pesquisas relacionam tratamento de restrições em problemas de otimização utilizando algoritmos evolutivos, entre eles Michalewicz (1995), Michalewicz e Schoenauer (1996) e Coello (2002). As abordagens são classificadas de acordo com

a forma como as restrições são tratadas e são listadas a seguir (TAKAHAMA; SAKAI, 2005a).

1. As restrições são utilizadas para verificar se uma solução candidata é factível ou não. Durante o processo de otimização, quando uma solução é infactível, ela é ajustada ou descartada.

2. A soma das violações é combinada com a função objetivo. Abordagens que utilizam este método são chamadas funções de penalidade (do inglês *penalty functions*).

3. A violação das restrições da função objetivo são utilizadas separadamente através de uma ordem lexicográfica, onde as restrições precedem a função objetivo.

4. As restrições e a função objetivo são otimizadas por métodos de otimização multi-objetivos.

Além da classificação proposta por Michalewicz (1995), os métodos de tratamento de restrições são classificados em dois grupos. Os genéricos não exploram a estrutura do problema, podem ser utilizados em problemas de qualquer natureza sem alterar a estrutura do método e podem ser acoplados a qualquer algoritmo. Os específicos que só podem ser aplicados a um tipo de restrição, ou seja, problemas convexos ou com poucas variáveis (DEB, 2000).

Várias abordagens para tratamento de restrições são encontrados na literatura e aquelas baseada em função penalidade são as mais comuns. Este método aplica uma penalidade em soluções infactíveis, onde a distância da região factível é utilizada como fator de penalidade. Entretanto, estes métodos possuem muitos parâmetros para serem ajustados, o que levou ao desenvolvimento de métodos como o multinível, proposto por Homaifar, Lai e Qi (1994); a função de penalidade dinâmica, proposto por Joines e Houck (1994), Michalewicz e Attia (1994) e Kazarlis e Petridis (1998). Deb (2000) propôs uma abordagem em que não há parâmetros para ajustar, porém, é necessário utilizar um método de nicho e um operador de mutação controlado para manter a diversidade da população. Barbosa e Lemonge (2002) desenvolveram um método que utiliza informações da população, como a aptidão ou o nível de violação de cada restrição. Tessema e Yen (2006) propuseram uma função penalidade auto adaptativa que define o coeficiente de penalidade através da quantidade de indivíduos factíveis. A função objetivo é calculada através da soma entre a distância, que é a média normalizada de todas as violações, e a função objetivo. De acordo com Michalewicz, estas abordagens pertencem a segunda e terceira classificações (MICHALEWICZ, 1995; DEB, 2000; HO; SHIMIZU, 2007; RODRIGUES; LIMA; GUIMARÃES, 2016).

Uma das dificuldades de utilizar a função penalidade é balancear os pesos entre as funções objetivo e penalidade. Runarsson e Yao (2000) propuseram o método *ranking* estocástico que faz o balanceamento estocasticamente. Uma probabilidade P_f é utilizada com *ranking* para comparar indivíduos infactíveis. Embora este método tenha obtido bons resultados, a probabilidade P_f deve ser ajustada. Fonseca, Capriles e Barbosa (2007) utilizaram o ACO e *ranking* estocástico para projetos de estruturas. Ho e Shimizu (2007) propuseram uma abordagem baseada em múltiplos *rankings*, onde são levados em consideração a função objetivo, a violação das restrições e a quantidade de restrições que foram violadas. Rodrigues, Lima e Guimarães (2016) propuseram o método de *ranking* balanceado, que mantém duas filas, de indivíduos factíveis e infactíveis, e depois as combina.

O método *Alfa Constrained* proposto por Takahama e Sakai (1999) utiliza uma ordem lexicográfica para comparar dois indivíduos. Este método foi utilizado efetivamente em problemas de otimização. Eles utilizaram o método *Alfa Constrained* combinado com o método de *Powell*. Takahama e Sakai (2000) desenvolveram o *Alfa Constrained Simplex*, onde o método foi utilizado para tratamento de restrições junto com o algoritmo Nelder-Mead. Takahama e Sakai (2005a) propuseram o algoritmo *Alfa Constrained* GA que utiliza seleção com *ranking* linear. O αGA e αPSO foram desenvolvidos por Takahama e Sakai (2004) e Takahama e Sakai (2005b), respectivamente. Apesar do bom desempenho dos algoritmos utilizando o *Alfa Constrained*, o método possui parâmetros a serem ajustados. Sendo assim, Takahama, Sakai e Iwane (2005) propuseram o método *Epsilon Constrained*, através de um algoritmo híbrido entre GA e PSO, livre de parâmetros. Em seguida, Takahama e Sakai (2005c) combinaram o *Epsilon Constrained* com o PSO. Várias abordagens foram propostas utilizando DE e o *Epsilon Constrained*, entre elas, Takahama e Sakai (2006), Takahama e Sakai (2015), Takahama e Sakai (2009b), Takahama e Sakai (2009a), Takahama e Sakai (2010a), Takahama e Sakai (2010b) e Takahama, Sakai e Iwane (2006).

Dentre os métodos desenvolvidos com *Alfa Constrained*, o αGA com seleção através de *ranking* linear possui o melhor desempenho, porém, em problemas multimodais, frequentemente fica preso a mínimos locais. Já o algoritmo εDE possui bom desempenho em problemas multimodais (TAKAHAMA; SAKAI, 2006).

Para utilizar o método *Alfa Constrained*, o nível de satisfação é definido para indicar o quanto uma solução candidata satisfaz o conjunto de restrições. Além disso, ele também é definido a fim de comparar soluções candidatas (TAKAHAMA; SAKAI, 2005a).

O nível de satisfação $\mu(\boldsymbol{x})$ é responsável por medir o quanto uma solução viola

o conjunto de restrições e é definido de acordo com

$$\begin{cases} \mu(\boldsymbol{x}) = 1, & \text{se } g_i(\boldsymbol{x}) \leq 0, h_j(\boldsymbol{x}) = 0, \forall i,j \\ 0 \leq \mu(\boldsymbol{x}) < 1, & \text{caso contrário} \end{cases} \tag{4.1}$$

Para defini-lo, é necessário medir a violação de uma solução em cada restrição individualmente. O nível de violação para restrições de desigualdade e igualdade são calculadas de acordo com

$$\mu_{gi} = \begin{cases} 1, & \text{se } g_i(\boldsymbol{x}) \leq 0 \\ 1 - \frac{g_i(\boldsymbol{x})}{b_i}, & \text{se } 0 \leq g_i(\boldsymbol{x}) \leq b_i \\ 0, & \text{caso contrário} \end{cases} \tag{4.2}$$

$$\mu_{hj} = \begin{cases} 1 - \frac{|h_j(\boldsymbol{x})|}{b_j}, & \text{se } |h_j(\boldsymbol{x})| \leq b_j \\ 0, & \text{caso contrário} \end{cases} \tag{4.3}$$

onde b_i e b_j são parâmetros fixos, definidos de acordo com o número de iterações do DE.

Uma vez que todas as violações são medidas individualmente, elas devem ser combinadas. A combinação é feita utilizando o operador *min* de acordo com

$$\mu(\boldsymbol{x}) = \min_{i,j} \{\mu_{gi}(\boldsymbol{x}), \mu_{hj}(\boldsymbol{x})\} \tag{4.4}$$

A comparação dos indivíduos é feita utilizando o nível de satisfação através de uma ordem lexicográfica entre duas soluções candidatas. Uma vez que é mais importante obter soluções factíveis, $\mu(\boldsymbol{x})$ precede $f(\boldsymbol{x})$. Dados $f_1(f_2)$ e $\mu_1(\mu_2)$, sendo a aptidão e o nível de satisfação das soluções x_1 e x_2, respectivamente, então a comparação $<_\alpha$ e $\leq_\alpha$ entre (f_1, μ_1) e (f_2, μ_2), para qualquer α, é utilizada, respectivamente, por

$$(f_1, \mu_1) <_\alpha (f_2, \mu_2) \Leftrightarrow \begin{cases} f_1 < f_2, & \text{se } \mu_1, \mu_2 \geq \alpha \\ f_1 < f_2, & \text{se } \mu_1 = \mu_2 \\ \mu_1 > \mu_2, & \text{caso contrário} \end{cases} \tag{4.5}$$

$$(f_1, \mu_1) \leq_\alpha (f_2, \mu_2) \Leftrightarrow \begin{cases} f_1 \leq f_2, & \text{se } \mu_1, \mu_2 \geq \alpha \\ f_1 \leq f_2, & \text{se } \mu_1 = \mu_2 \\ \mu_1 > \mu_2, & \text{caso contrário} \end{cases} \tag{4.6}$$

4.2 Entropia

O conceito de entropia foi proposto pelo físico e matemático alemão Rudolf Clausius. Inicialmente foi aplicada à termodinâmica para representar a mudança de estados quando ocorre um aumento de energia durante um processo reversível. Claude Shannon utilizou a mesma função no contexto da teoria da informação para expressar o grau de desordem das informações em um sistema (ROSCA, 1995).

Inicialmente, a informação era vista como uma grandeza pobremente definida. Entretanto, em 1948, Claude Shannon publicou *A mathematical theory of comunication* e a definiu como uma grandeza quantitativa. Além disso, Shannon descreveu como a entropia pode ser conectada entre diferentes elementos de qualquer sistema e natureza (STONE, 2015).

Rosca (1995) desenvolveu um algoritmo composto por GA/GP que permite o controle da diversidade e formas de adaptação para que este sistema convirja para a solução ótima rapidamente. O algoritmo foi aplicado à regressão booleana e ao controle de agentes em ambientes dinâmicos. Para isto, foi utilizada a entropia como medida de diversidade da população para permitir a adaptação inteligente na busca.

Através da hipótese "Explorar ou Intensificar" (do inglês "*To explore or to exploit*"), Liu, Mernik e Bryant (2009) utilizaram 5 formas diferentes de calcular a entropia para medir a diversidade da população para aumentar ou diminuir a taxa de mutação. O algoritmo foi aplicado a problemas de otimização contínua. Os experimentos mostraram que o método proposto obteve sucesso na exploração/intensificação no processo de busca além de obter melhores valores da aptidão e taxas de convergências.

Ni e Deng (2014) combinaram o PSO com três medidas de diversidade da população para analisar sua evolução: desvio padrão da aptidão, da posição dos indivíduos no espaço de busca e entropia. Quatro *benchmarks* de otimização contínua foram utilizados e foi concluído que qualquer medida é capaz de prover a diversidade da população. Porém, o cálculo do desvio padrão da aptidão e da posição dos indivíduos no espaço de busca possui complexidade $O(n^2)$.

Črepinšek, Liu e Mernik (2013) afirmam que a diversidade da população pode ser medida através dos indivíduos (nível genotípico), da aptidão (nível fenotípico) ou ainda uma combinação entre elas. Eles abordaram os principais problemas encontrados nos algoritmos evolutivos para solucionar problemas sem restrições, porém não mencionaram os problemas de otimização com restrições.

Campos e Krohling (2016) desenvolveram o BBPSO utilizando entropia para problemas de otimização em ambientes dinâmicos. A entropia é utilizada para medir a diversidade da população e atua como um fator de balanceamento nas direções entre os melhores global e local. A alta diversidade guia a busca através da melhor partícula

da população a fim de favorecer a intensificação, ao passo que, a baixa diversidade guia a busca através da melhor solução local para favorecer a exploração. Os experimentos mostraram a eficácia da abordagem proposta em encontrar boas soluções quando aplicado aos *benchmarks* da literatura e problemas do mundo real.

Como visto na Seção 4.1, ao gerar uma nova solução, deve-se calcular à aptidão e a violação das restrições, ou seja, $f(\boldsymbol{x})$ e $\mu(\boldsymbol{x})$. Sendo assim, este trabalho propõe que a informação para o cálculo da entropia seja calculada através do produto $f(\boldsymbol{x}) \cdot \mu(\boldsymbol{x})$.

Uma vez que a informação foi redefinida, a entropia é calculada de acordo com

$$Entropia(G) = -\sum_{i=1}^{NP}(p_i)\log_2(p_i) \tag{4.7}$$

onde G é o número da iteração, NP indica o número de indivíduos e p_i é a probabilidade de a informação corresponder à categoria i. A probabilidade p_i é calculada como segue:

1. Normaliza-se o conjunto de informações e divide-se em NP partes, denominada $n_i \quad (i = 1, 2, \ldots, NP)$;
2. Conta-se o número de elementos diferentes de zero e calcula-se $p_i = \dfrac{n_i}{NP}$;

Neste trabalho, utiliza-se a entropia normalizada[1] de acordo com

$$Entropia(G) = -\sum_{i=1}^{NP}\frac{(p_i)\log_2(p_i)}{\log_2 NP} \tag{4.8}$$

[1] http://math.stackexchange.com/questions/395121/how-entropy-scales-with-sample-size

4.3 Algoritmo híbrido entre evolução diferencial e Nelder-Mead

Apesar da eficácia dos algoritmos DE e NM, ambos possuem suas desvantagens. O DE foi desenvolvido originalmente para otimização contínua. Por outro lado, Brea (2013) desenvolveu uma versão do NM para problemas de otimização inteira mista, porém, ainda, é necessário fornecer um conjunto de soluções iniciais, que, se escolhidas indevidamente, pode levar o processo de otimização a mínimos locais.

A fim de contornar estas desvantagens, os algoritmos híbridos são uma alternativa para esses problemas. Um algoritmo híbrido, ou algoritmo memético, é composto por um algoritmo evolutivo, que atua como o mecanismo de busca global, e um componente de busca local, que é acionado com certa frequência (NERI; COTTA, 2012).

Duan, Liao e Yi (2013) fizeram um estudo comparativo utilizando DE, *Harmony Search* e *Hooke and Jeeves*, como busca local, que foi implementado de 18 formas diferentes. A abordagem proposta foi aplicada em 19 problemas de otimização inteira mista com restrições e os resultados apontaram três estratégias principais. A primeira delas é aplicar a busca local a cada indivíduo da população com uma probabilidade específica (10%). As demais consistem em aplicar a busca local a cada indivíduo melhorado e a cada melhor solução gerada, além de aplicar o primeiro método.

Ponsich e Coello (2013) desenvolveram uma abordagem híbrida entre DE e Busca Tabu (TS) para problemas de escalonamento *job-shop*, em que a busca local é executada a cada 10 iterações, após as operações do DE. A maioria dos *benchmarks* testados foram de tamanho médio, e a abordagem proposta foi capaz de retornar a solução ótima. Além disso, em instâncias maiores e mais complexas, DE+TS teve desempenho competitivo aos demais algoritmos.

Domınguez-Isidro, Mezura-Montes e Leguizamón (2014) analisaram o desempenho do algoritmo híbrido DE em conjunto com 3 operadores de busca local, utilizados separadamente, que são *Hooke and Jeeves* (HJ), NM e *Hill Climber* (HC). Foram utilizados 36 *benchmarks* da sessão especial "Otimização mono-objetiva contínua com restrições" (CEC 2010) para analisar o comportamento da busca local e a eficácia do método em encontrar o ótimo global. O mecanismo de busca local é ativado se a melhor solução encontrada não melhorar após T iterações, onde T é o número de dimensões do problema (LIU et al., 2007). Além disso, é utilizada uma probabilidade ψ que é definida de acordo com

$$\psi = 1 - \left| \frac{J_{avg} - J_{best}}{J_{worst} - J_{best}} \right|$$

onde J_{avg}, J_{best} e J_{worst} é a média, a melhor e a pior aptidão da população, respectivamente (NERI et al., 2006). Os dados obtidos para análise da busca local mostram que

o HC obteve a melhor média de desempenho, porém, teve resultado similar comparado ao NM e HJ. Além disto, o algoritmo híbrido encontrou a solução ótima em todos os *benchmarks* utilizados.

García, García-Ródenas e Gómez (2014) desenvolveram um método híbrido utilizando GA, PSO e Cozimento Simulado (do inglês *Simulated Annealing*, abreviado por SA) como método de busca global e NM para realizar a busca local. A busca local é executada a cada 5 iterações. O método proposto foi aplicado a agrupamento de dados (do inglês *data clustering*) com restrições no domínio do tempo e problemas de reconhecimento de padrões. Os autores concluíram que o SA+NM superou os algoritmos GA+NM e PSO+NM.

Wu et al. (2014) utilizaram um híbrido entre PSO com solução guiada superiormente (SSG-PSO) com duas técnicas de busca local baseadas em gradiente, que são *Broyden Fletcher Goldfarb Shanno* (BFGS) e *Davidon Fletcher Powell* (DFP), além de duas técnicas livres de derivadas, que são Busca por Padrões e NM. Para alternar entre busca local e global, o método proposto utiliza a comparação $fitcount \geq \beta \cdot MaxEF$, onde $fitcont$ é inicializado com 40 e incrementado com o número de avaliações de funções utilizada pela busca local, $\beta = 0.8$ e $MaxEF = 3 \cdot 10^5$. Este algoritmo foi aplicado à problemas de otimização sem restrições. Os autores concluíram que, para problemas unimodais, SSG-PSO combinado com as buscas locais baseadas em gradiente tiveram bom desempenho, porém, em problemas multimodais, SSG-PSO combinado com as buscas locais livre de derivadas obtiveram melhores resultados.

Schneider e Krohling (2014) propuseram um algoritmo híbrido utilizando DE, TS e técnica para avaliar o desempenho de alternativas através de similaridade com a solução ideal (do inglês *Technique for Order Preference by Similarity to Ideal Solution*, abreviado por TOPSIS), aplicado a dois tipos de problemas. O mecanismo de busca local é ativado a cada 10 iterações. Em problemas de otimização inteira mono-objetivo, todas as versões do DE estudadas obtiveram desempenho similar, uma vez que convergem facilmente ao ótimo global. Por outro lado, em problemas de otimização inteira com múltiplas soluções, DEGL supera as demais variações do DE, pois ambas sofrem convergência prematura, enquanto DEGL possui a capacidade de manter a diversidade na população.

Luchi e Krohling (2015b) propuseram um algoritmo híbrido utilizando DE e NM (abreviado por DE+NM) para variáveis inteiras aplicado a problemas de otimização sem restrições. Em seguida, Luchi e Krohling (2015a) estenderam o DE+NM para problemas de otimização não-linear inteira, utilizando o método *Alfa Constrained*. Em ambas abordagens, o NM é ativado a cada 10 iterações, após as operações realizadas pelo DE. Para problemas sem restrições, o desempenho do algoritmo foi equivalente ao D&BMS, que foi utilizado para comparação. Nos problemas de otimização com

restrições, o DE+NM obteve 100% de taxa de sucesso, que é o mesmo desempenho dos algoritmos DE+TOPSIS, (μ, λ)-ES e MI-LXPM. Além disso, superou o algoritmo MI-LXPM, que obteve 71% de taxa de sucesso em um dos problemas testados.

Bonyadi, Li e Michalewicz (2014) propuseram uma abordagem híbrida entre PSO e dois métodos de busca local, Mutação Gradiente modificada (MG) e Programação Quadrática Sequencial (SQP). A busca local é executada se um número sorteado através de uma distribuição de probabilidade uniforme $U(0, 1)$ for menor que $P_L = 0.01$. Além disso, os autores propuseram uma vizinhança variável a fim de atuar em problemas com regiões factíveis disjuntas. Para testar o algoritmo, foram utilizados os *benchmarks* da CEC2010 e os resultados mostraram que o método encontrou as soluções ótimas.

Diante da eficácia dos algoritmos apresentados, este trabalho propõe um método híbrido entre DEGL e NM para problemas de otimização não-linear inteira mista. Além disso, utiliza-se a entropia para medir a diversidade da população e atuar efetivamente no algoritmo, favorecendo a exploração/intensificação e ativando o mecanismo de busca local, denominada algoritmo híbrido entre Evolução Diferencial e Nelder-Mead com chaveamento baseado em entropia (abreviado por DE-E-NM) (KROHLING, 2015). O fluxograma do DE-E-NM é mostrado na Figura 12.

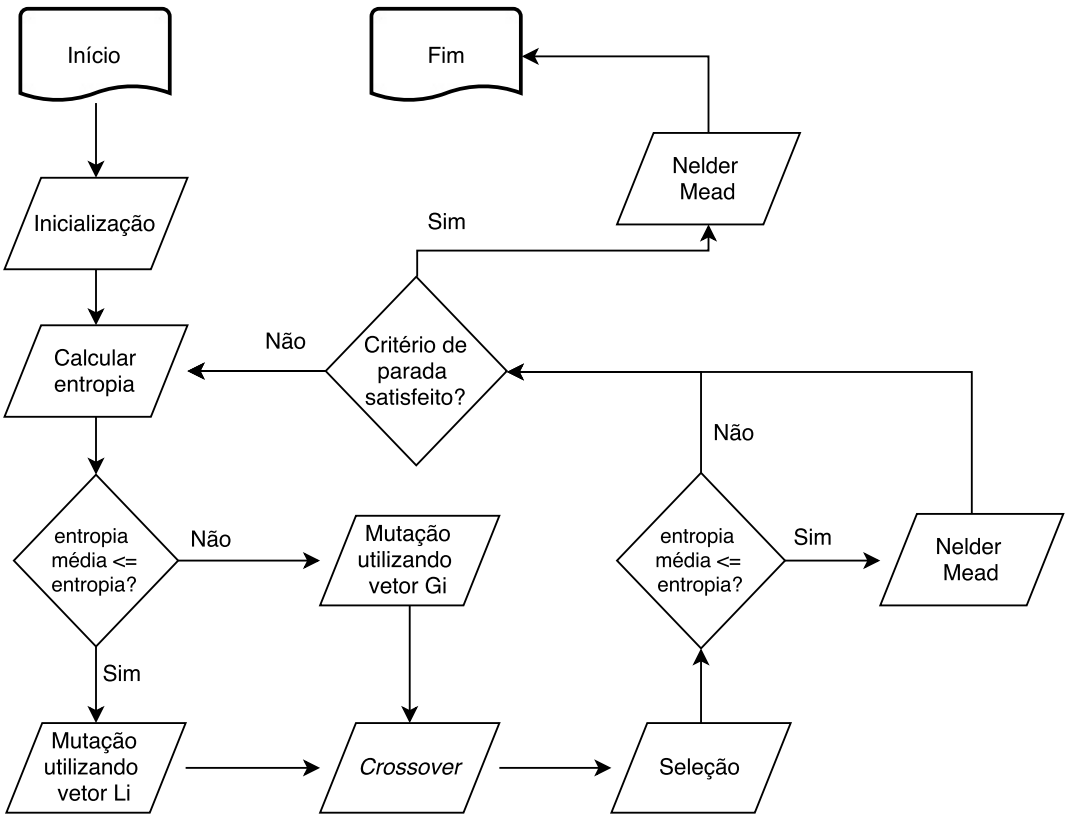


Figura 12 – Fluxograma do algoritmo DE-E-NM.

Através da Figura 12, é possível identificar os procedimentos do método proposto. A inicialização da população é realizada através de uma distribuição de probabilidade uniforme limitada pelo espaço de busca. Em seguida, calcula-se a entropia da população ($entr$) e a entropia média ($meanEntr$) das últimas 100 iterações do DE, que é definido através da taxa de 10% do número de iterações. Através da $meanEntr$, serão tomadas as decisões de qual vetor será utilizado para fazer a mutação (vetor local ou global) e se ocorrerá a busca local na iteração atual. A mutação é realizada utilizando, exclusivamente, o vetor local (L_i), a fim de favorecer a exploração, ou o vetor global (G_i), a fim de favorecer a intensificação. O *crossover* e a seleção são conduzidas normalmente. Em seguida, a execução da busca local também dependerá da entropia, sendo assim, a segunda decisão baseada na entropia. Por último, caso o critério de parada seja satisfeito, o algoritmo termina e a última busca local é executada.

As soluções candidatas utilizadas no NM são geradas a partir de uma distribuição de probabilidade normal $X \sim N(X_{gbest}, 1)$, onde X_{gbest} é a melhor solução encontrada pelo algoritmo. Para problemas de otimização inteira, as soluções candidatas são arredondadas para o inteiro mais próximo.

O tamanho do desvio padrão afeta consideravelmente na escolha da vizinhança que é utilizada no NM. De acordo com a regra empírica, ou regra dos 3-*sigmas*, tem-se 68%, 95% e 99.7% de probabilidade de se sortear valores dentro do intervalo $\pm 1\sigma$, $\pm 2\sigma$ e $\pm 3\sigma$, respectivamente. Em outras palavras, utilizando uma distribuição normal $X \sim N(X_{gbest}, 1)$, tem-se 68%, 95% e 99.7% de probabilidade de se sortear valores a uma distância de 1, 2 e 3 unidades, respectivamente, de X_{gbest}. Uma vez que o NM é utilizado para busca local, é adequado utilizar o desvio padrão indicado.

O pseudo-código do DE-E-NM é apresentado no Algoritmo 4.

Algoritmo 4: Algoritmo híbrido DE-E-NM

início
- inicialização da população
- **repita**
  - calcular a entropia ($entr$) de acordo com (4.8)
  - calcular a entropia média ($meanEntr$)
  - **para** i de 1 até NP **faça**
    - **se** $meanEntr \leq entr$ **então**
      - sortear r_1, r_2 tal que $i \neq r_1 \neq r_2$
      - calcular o vetor local com (3.7)
    - **senão**
      - sortear q_1 e q_2 tal que $i \neq q_1 \neq q_2$
      - calcular o vetor global com (3.8)
    - **fim**
    - *crossover* de acordo com (3.4)
    - seleção de acordo com (3.5)
  - **fim**
  - **se** $meanEntr \leq entr$ **então**
    - atribuir o melhor indivíduo encontrado a P_0
    - gerar D candidatos através de uma distribuição de probabilidade normal em torno de P_0
    - executar a busca local com o algoritmo Nelder-Mead
  - **fim**
- **até** critério de parada seja satisfeito;
- atribuir o melhor indivíduo encontrado a P_0
- gerar D candidatos através de uma distribuição de probabilidade normal em torno de P_0
- executar a busca local com o algoritmo Nelder-Mead

fim

5 Resultados de simulações

5.1 Problemas de otimização contínua irrestrita

Nesta seção são considerados problemas de otimização não-linear contínua irrestrita. Eles podem ser considerados como uma variação do problema 2.1, onde não há restrições de igualdade e desigualdade ($q = m = 0$) e o vetor de variáveis de decisão inteiro é vazio ($\boldsymbol{y} = \varnothing$). São definidos de acordo com

$$\text{Minimizar} \qquad f(\boldsymbol{x})$$

onde $\boldsymbol{x}$ representa o vetor de variáveis contínuas, $f(\boldsymbol{x})$ representa a função objetivo (KITAYAMAA; ARAKAWAB; YAMAZAKI, 2011).

A fim de avaliar o desempenho das variações do DE apresentadas na Seção 3.1, cinco *benchmarks* foram utilizados (KITAYAMAA; ARAKAWAB; YAMAZAKI, 2011).

Problema 1 (2^n minima):

$$\text{Min.} \qquad f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i^4 - 16x_i^2 + 5x_i\right)$$

Os limites inferiores e superiores são $-5 \leq x_i \leq 5$ $(i = 1, 2, 3, \ldots, 10)$. A solução ótima é $f(\boldsymbol{x}^*) = -391.661000$.

Problema 2 (*Griewank*):

$$\text{Min.} \qquad f(\boldsymbol{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$$

Os limites inferiores e superiores são $-10 \leq x_i \leq 10$ $(i = 1, 2, \ldots, 10)$. A solução ótima é $f(\boldsymbol{x}^*) = 0$.

Problema 3 (*Ackley*):

$$\text{Min.} \qquad f(\boldsymbol{x}) = 22.71828 - 20\exp\left(-0.2\left[\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right]\right) - \exp\left(\left[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right]\right)$$

Os limites inferiores e superiores são $-30 \leq x_i \leq 30$ $(i = 1, 2, \ldots, 10)$. A solução ótima é $f(\boldsymbol{x}^*) = 0$.

Problema 4 (*Rastrigin*):

$$\text{Min.} \qquad f(\boldsymbol{x}) = 10 + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Os limites inferiores e superiores são $-5.12 \leq x_i \leq 5.12$ $(i = 1, 2, \ldots, 6)$. A solução ótima é $f(\boldsymbol{x}^*) = 0$.

Problema 5 (*Michalewicz*):

$$\text{Min.} \qquad f(\boldsymbol{x}) = -\sum_{i}^{n}\sin(x_i)\cdot\left\{\sin\left(\frac{ix_i^2}{\pi}\right)\right\}^{20}$$

Os limites inferiores e superiores são $0 \leq x_i \leq \pi$ $(i = 1, 2, \ldots, 5)$. A solução ótima é $f(\boldsymbol{x}^*) = -4.687600$.

A Tabela 2 mostra os parâmetros utilizados no DE. As variações utilizadas são DE/*rand*/1, DE/*local-to-best*/1 e DEGL. O tamanho da população, taxa de *crossover* e fator de mutação são sugeridos por Storn e Price (1995). Os parâmetros F_α, F_β e k (tamanho da vizinhança) são sugeridos por Das et al. (2009).

Tabela 2 – Parâmetros utilizados no DE em problemas de otimização contínua.

Tamanho da população	40
Critério de parada (iterações)	500
Taxa de *crossover*	0.9
Fator de mutação F	0.8
Parâmetros F_α e F_β	0.8
k (tamanho da vizinhança)	2

Tabela 3 – Resultados encontrados nos problemas de otimização contínua.

Problema	Variação	Ótimo global	Melhor	Média	D. padrão
2^n minima	DE/*rand*/1	-391.661000	-356.419400	-330.580000	14.020000
	DE/*local-to-best*/1		-356.856500	-333.000000	13.260000
	DEGL		$\mathbf{-391.661700}$	$\mathbf{-387.130000}$	$\mathbf{7.790000}$
Griewank	DE/*rand*/1	0	0.342800	0.500000	0.070000
	DE/*local-to-best*/1		0.206900	0.480000	0.090000
	DEGL		$\mathbf{1.1102E-16}$	$\mathbf{0.020000}$	$\mathbf{0.030000}$
Ackley	DE/*rand*/1	0	6.442200	8.290000	1.010000
	DE/*local-to-best*/1		5.697100	8.750000	1.230000
	DEGL		$\mathbf{-1.8285E-6}$	$\mathbf{0.040000}$	$\mathbf{0.220000}$
Rastrigin	DE/*rand*/1	0	8.482700	17.660000	4.490000
	DE/*local-to-best*/1		7.090600	17.650000	3.930000
	DEGL		$\mathbf{0}$	$\mathbf{1.950000}$	$\mathbf{2.110000}$
Michalewicz	DE/*rand*/1	-4.687600	-4.602200	-4.310000	0.180000
	DE/*local-to-best*/1		-4.614500	-4.280000	0.290000
	DEGL		$\mathbf{-4.687700}$	$\mathbf{-4.680000}$	$\mathbf{0.010000}$

A fim de comparar o desempenho do DEGL, são considerados os resultados obtidos pelo DE/*rand*/1 e DE/*local-to-best*/1. A Tabela 3 mostra o mínimo global, o melhor resultado encontrado, a média e desvio padrão, respectivamente, considerando 30 execuções de cada *benchmark*. Os números em negrito indicam os melhores resultados.

Nota-se pela Tabela 3 que o algoritmo DEGL encontrou o mínimo global em todos os problemas, enquanto as variações DE/*rand*/1 e DE/*local-to-best*/1 falharam

nos problemas 2^n minima, *Ackley* e *Rastrigin*. Além disso, DEGL obteve os menores desvios padrão e as melhores média em todos os problemas testados. Portanto, nota-se que o DEGL possui desempenho superior às demais variações utilizadas devido a sua capacidade de manter a diversidade da população durante o processo de otimização.

Os experimentos utilizando problemas de otimização contínua foram utilizados a fim de comprovar a eficácia do DEGL sobre as versões DE/*rand*/1 e DE/*local-to-best*/1. Nota-se que a variação DE/*local-to-best*/1 sofre influência do melhor indivíduo da população, desta forma, a informação se propaga rapidamente aumentando as chances de ficar preso a mínimos locais. No caso do DEGL a informação se propaga lentamente, uma vez que um indivíduo está conectado apenas a seus vizinhos, diminuindo as chances de ficarem presos a mínimos locais (DAS; SUGANTHAN, 2011).

5.2 Problemas de otimização inteira irrestrita

Nesta seção são considerados problemas de otimização não-linear inteira irrestrita. Eles podem ser considerados como uma variação do problema 2.1, onde não há restrições de igualdade e desigualdade ($q = m = 0$) e o vetor de variáveis de decisão contínua é vazio ($\boldsymbol{x} = \varnothing$). São definidos de acordo com

$$\begin{aligned} \text{Minimizar} \quad & f(\boldsymbol{y}) \\ & \boldsymbol{y} \in \mathbb{Z} \end{aligned}$$

onde $\boldsymbol{y}$ representa o vetor de variáveis inteiras do problema, $f(\boldsymbol{y})$ representa a função objetivo (TIAN; MA; ZHANG, 1998).

A fim de avaliar o desempenho do DE-E-NM, três *benchmarks* foram utilizados e os resultados foram comparados com o algoritmo *Darwin and Boltzmann mixed strategy* proposto por Tian, Ma e Zhang (1998).

Problema 1 (*Shekel*):

$$\text{Min.} \quad f(\boldsymbol{y}) = \sum_{j=1}^{m} \frac{(-1)}{|(\boldsymbol{y} - A_j)^T(\boldsymbol{y} - A_j) + c_j|}$$

onde A_j e c_j são mostrados na Tabela 4. A solução ótima é $\boldsymbol{y}^* = (4, 4, 4, 4)$, as variações utilizadas são $m = 5$ (SQRN5), $m = 7$ (SQRN7) e $m = 10$ (SQRN10) e os mínimos globais são listados na Tabela 5. Os limites inferiores e superiores utilizados são $0 \leq y_i \leq 10, (i = 1, 2, 3, 4)$.

Tabela 4 – Matrizes A_j e c_j utilizadas em *Shekel*.

A_j				c_j
4.0	4.0	4.0	4.0	0.1
1.0	1.0	1.0	0.1	0.2
8.0	8.0	8.0	8.0	0.2
6.0	6.0	6.0	6.0	0.4
3.0	7.0	3.0	7.0	0.6
2.0	9.0	2.0	9.0	0.6
5.0	5.0	3.0	3.0	0.3
8.0	1.0	8.0	1.0	0.7
6.0	2.0	6.0	2.0	0.5
7.0	3.6	7.0	3.6	0.5

Problema 2:

$$\begin{aligned} \text{Min.} \quad f(\boldsymbol{y}) = & \left[(y_1 - 3)^2 \cdot \cos(\pi \cdot y_1)\right] + \left[(y_2 - 6)\sin(\frac{\pi}{4})\right] + \\ & \left[(y_3 - 2.5)^2 \cdot (y_2 + 2)^{-1}\right] + \left[(y_3 + 2)^3 \cdot e^{-y_4}\right] \end{aligned}$$

Tabela 5 – Mínimos globais da função *Shekel*.

Instância	Função	$f(\boldsymbol{y}^*)$
p1-I	SQRN5	-10.152700
p1-II	SQRN7	-10.402300
p1-III	SQRN10	-10.535800

Tabela 6 – Limites inferiores e superiores, pontos ótimos e mínimos globais do problema 2.

Instância	Limites	$\boldsymbol{y}^*$	$f(\boldsymbol{y}^*)$
p2-I	$[0, 60]$	$(59, 54, 2, 34)$	-3183.995000
p2-II	$[0, 80]$	$(79, 78, 2, 33)$	-5847.996900
p2-III	$[0, 100]$	$(99, 94, 2, 32)$	-9303.997400

Os limites inferiores e superiores, pontos ótimos e valores das funções objetivas são mostrados na Tabela 6.

Problema 3:

$$\text{Min.} \quad f(\boldsymbol{y}) = \Big[(y_1 - 2.5)^2 \cdot (y_2 + 12.6)^2 \cdot (y_3 + 25.4)\Big] + \Big[(y_3 - 4.5)^2 \cdot (y_4 + 18.4)^{-1} \cdot e^{y_2 - 6.5}\Big] + \left[y_4^3 \cdot (y_5 + 10.8)^2 \cdot \sin\left(\frac{\pi}{10} \cdot (y_6 + 1) \cdot y_5\right)\right]$$

Os limites inferiores e superiores, soluções ótimas e mínimos globais são apresentados na Tabela 7.

Tabela 7 – Limites inferiores e superiores, pontos ótimos e mínimos globais do problema 3.

Instância	Limites	$\boldsymbol{y}^*$	$f(\boldsymbol{y}^*)$
p3-I	$[-5, 5]$	$(2, -5, -5, 5, 5, -2)$	-30910.424000
p3-II	$[-10, 10]$	$(2, -10, -10, 10, 9, -6)$	-392013.974000
p3-III	$[10, 30]$	$(10, 10, 10, 30, 29, 26)$	-41752008.452800
p3-IV	$[-30, -10]$	$(-30, -30, -30, -30, -29, -26)$	-10414515.149900

Os parâmetros utilizados no DEGL são listados na Tabela 8. O tamanho da população, a taxa de *crossover* são sugeridos por Storn e Price (1995). Os parâmetros F_α, F_β e k (tamanho da vizinhança), são sugeridos por Das et al. (2009).

Tabela 8 – Parâmetros utilizados no DEGL.

Tamanho da população	40
Critério de parada (iterações)	1000
Taxa de *crossover*	0.9
Parâmetros F_α e F_β	0.8
k (tamanho da vizinhança)	2

Os parâmetros utilizados no NM são os mesmos sugeridos por Nelder e Mead (1965) e são mostrados na Tabela 9.

Tabela 9 – Parâmetros utilizados no NM.

Número de iterações	10
α (coeficiente de reflexão)	1
β (coeficiente de expansão)	2
γ (coeficiente de contração)	0.5

A fim de analisar estatisticamente o desempenho do DE-E-NM, a Tabela 10 mostra o melhor, o pior resultado encontrado, a média e o desvio padrão, respectivamente, para 100 execuções de cada *benchmark*, seguindo os experimentos de Tian, Ma e Zhang (1998).

Tabela 10 – Resultados obtidos pelo DE-E-NM para problemas de otimização não-linear inteira irrestrita.

Problema	Melhor	Pior	Média	Desvio padrão
p1-I	-10.152700	-1.797000	-9.168400	2.404700
p1-II	-10.402300	-2.751300	-10.167600	1.174000
p1-III	-10.535800	-2.420700	-10.270700	1.329700
p2-I	-3183.995000	-3175.994700	-3183.915500	0.800000
p2-II	-5847.996000	-5831.996000	-5847.676800	1.943110
p2-III	-9303.997000	-9219.937500	-9301.556700	9.550800
p3-I	-30910.424000	-29691.562100	-30845.293600	210.7703
p3-II	-392013.974000	-369836.410000	-388439.089800	7527.4698
p3-III	-41752008.452000	-39658740.682700	-41383777.078000	766433.9987
p3-IV	-10414515.149000	-85386.000800	-10186163.324400	462323.0099

Além disso, para comparar o desempenho do DE-E-NM, a taxa de sucesso obtida pelo D&BMS é considerada e é mostrada na Tabela 11.

Tabela 11 – Comparação entre DE-E-NM e D&BMS.

Problema	DE-E-NM(%)	D&BMS(%)
p1-I	**85**	25
p1-II	**96**	29
p1-III	**96**	24
p2-I	**99**	**99**
p2-II	97	**100**
p2-III	85	**100**
p3-I	35	**80**
p3-II	20	**42**
p3-III	**71**	39
p3-IV	**72**	48

Nota-se pela Tabela 11 que o algoritmo DE-E-NM obteve altas taxas de sucesso em p1 e p2, superando o D&BMS em todas as instâncias de p1, porém foi superado

em p2-II e p2-III. Além disso, mesmo obtendo baixas taxas em p3, o DE-E-NM supera o D&BMS nas instâncias p3-III e p3-IV, sendo superado nas demais instâncias.

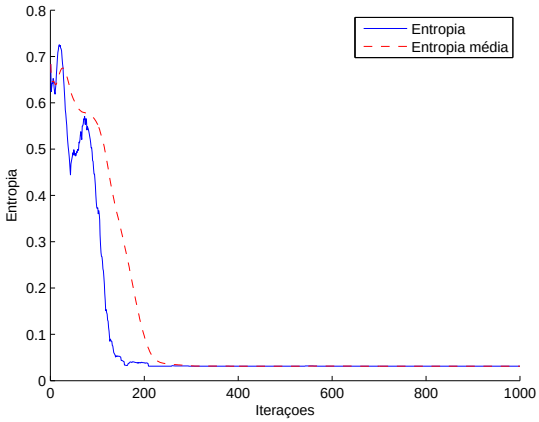
O algoritmo foi implementado em Matlab®(versão R2008b) e os experimentos foram feitos em uma máquina com processador AMD Phenon II B53 (64bits) e memória RAM de 4GB. A fim de analisar a eficiência do algoritmo DE-E-NM, o tempo computacional foi medido e é mostrado, em segundos, na Tabela 12.

Tabela 12 – Tempos médios de execução para problemas de otimização não-linear inteira irrestrita.

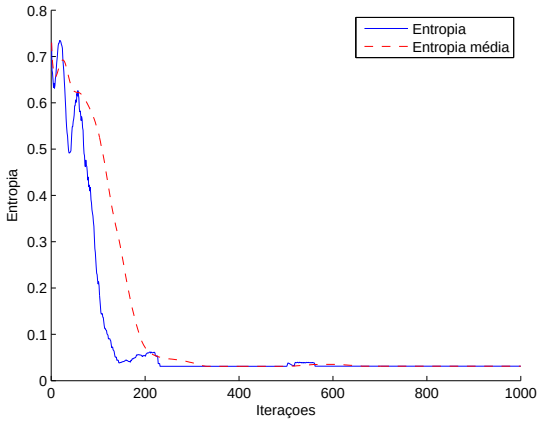
Problema	Tempo computacional
p1-I	12.405282
p1-II	12.529715
p1-III	12.213684
p2-I	12.161976
p2-II	12.178883
p2-III	11.757271
p3-I	12.734946
p3-II	10.887581
p3-III	13.077665
p3-IV	8.397506

As Figuras 13, 14 e 15 mostram a evolução da entropia durante as iterações do DE nos problemas 1, 2 e 3, respectivamente. A linha sólida azul indica a entropia ($entr$) da população. A linha pontilhada vermelha indica a entropia média ($meanEntr$) sobre as 100 últimas iterações.

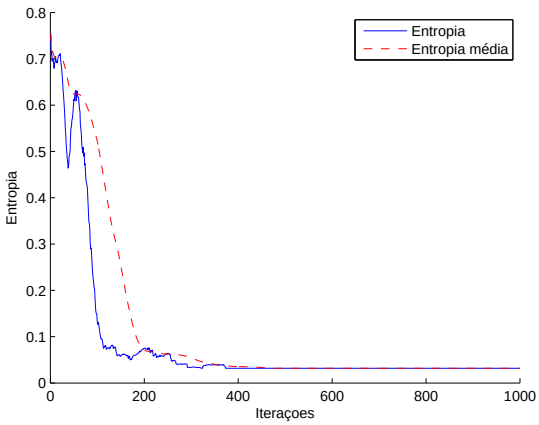
Nota-se pelas Figuras 13 e 14 que a entropia possui um comportamento similar, e, apesar de o algoritmo DEGL ter habilidades em manter a diversidade da população, ela cai lentamente até chegar próximo a zero em, aproximadamente, 200 iterações. É possível notar também que a entropia entra na eminência de aumentar, por volta da iteração 100, porém volta a decrescer em seguida. A evolução da entropia no problema 3 é mostrada na Figura 15 e nota-se que a entropia cai menos rapidamente do que para os demais problemas, chegando próximo a zero em, aproximadamente, 600 iterações.



(a) Instância I do problema 1.

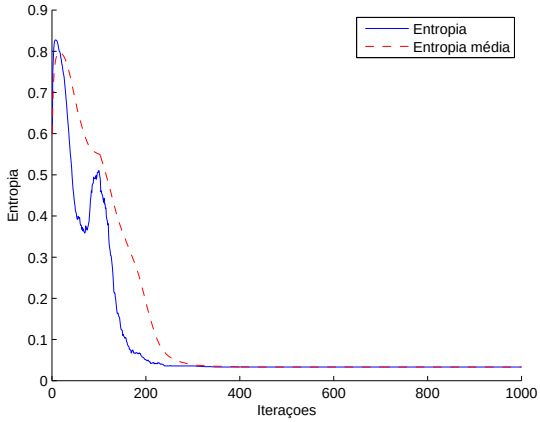


(b) Instância II do problema 1.

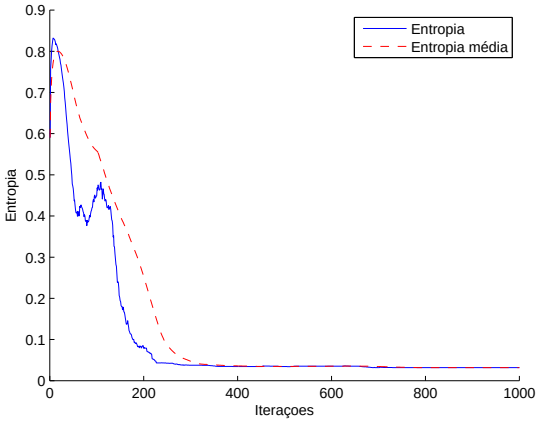


(c) Instância III do problema 1.

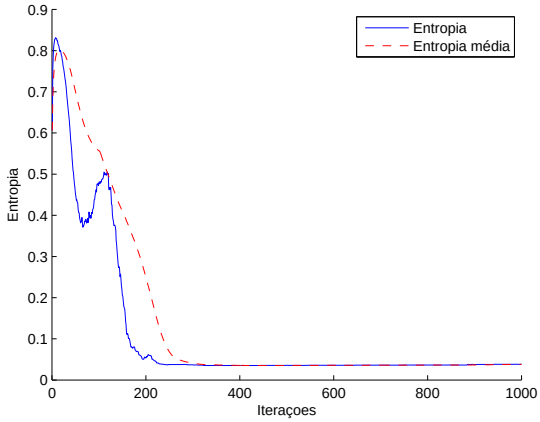
Figura 13 – Entropia e entropia média no problema 1, nas instâncias I, II e III.



(a) Instância I do problema 2.

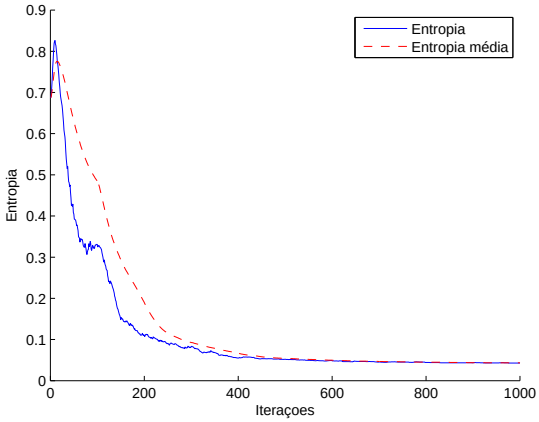


(b) Instância II do problema 2.

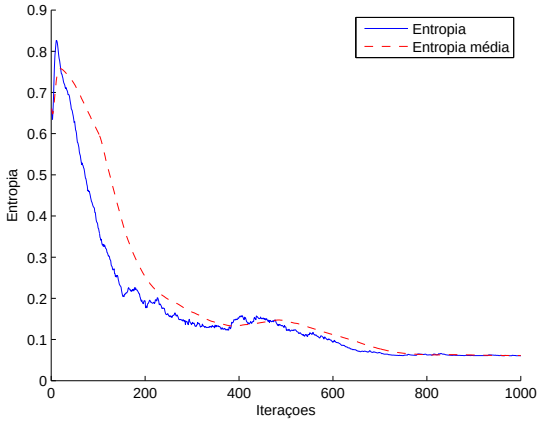


(c) Instância III do problema 2.

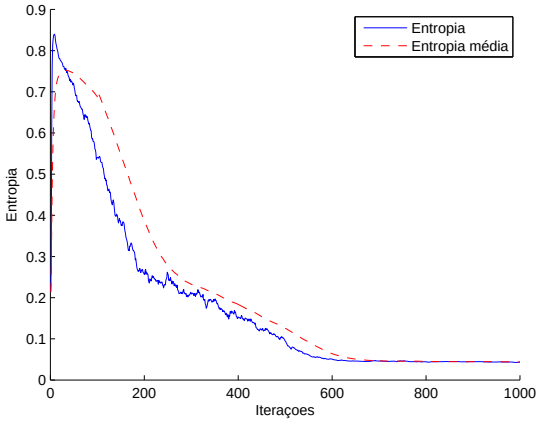
Figura 14 – Entropia e entropia média no problema 2, nas instâncias I, II e III.



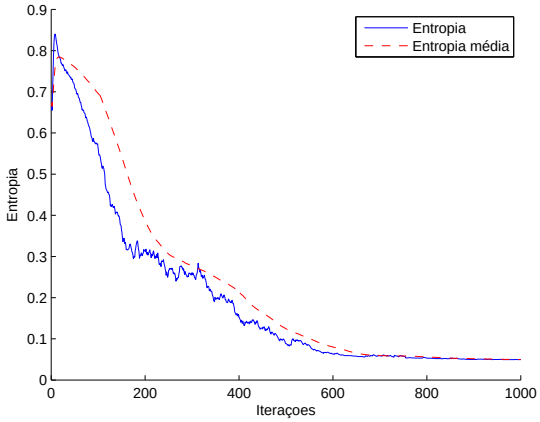
(a) Instância I do problema 3.



(b) Instância II do problema 3.



(c) Instância III do problema 3.



(d) Instância IV do problema 3.

Figura 15 – Entropia e entropia média no problema 3, nas instâncias I, II, III e IV.

5.3 Problemas de otimização inteira com restrições

Nesta seção são considerados problemas de otimização não-linear inteira com restrições. Eles podem ser considerados como uma variação do problema 2.1, onde o vetor de variáveis de decisão contínuo é vazio ($\boldsymbol{x} = \varnothing$). São definidos de acordo com

$$\begin{aligned} \text{Minimizar} \quad & f(\boldsymbol{y}) \\ \text{Sujeito a} \quad & g_j(\boldsymbol{y}) \quad (j = 1, 2, \ldots, q) \\ & h_j(\boldsymbol{y}) \quad (j = q+1, \ldots, m) \\ & l_i \leq y_i \leq u_i \quad (i = 1, 2, \ldots, n) \end{aligned}$$

onde $\boldsymbol{y}$ representa o vetor de variáveis inteiras do problema, $f(\boldsymbol{y})$ representa a função objetivo, g_j e h_j representam as q restrições de desigualdade e $m - q$ restrições de igualdade, respectivamente. Os valores de l_i e u_i representam os limites inferiores e superiores de y_i, respectivamente, e definem o espaço de busca do problema ($\mathcal{S}$). As restrições definem a região factível do problema $\mathcal{F}$, sendo que, $\mathcal{F} \subseteq \mathcal{S}$ (TAKAHAMA; SAKAI, 2005a).

A fim de analisar o desempenho do algoritmo DE-E-NM, oito *benchmarks* foram utilizados e os resultados foram comparados ao DE+TOPSIS proposto por Schneider e Krohling (2014), *Evolution Strategy with stochastic ranking* ((μ, λ)-ES) proposto por Runarsson e Yao (2000) e MI-LXPM proposto por Deep et al. (2009).

Problema 1 (MOHAN; NGUYEN, 1999):

$$\begin{aligned} \text{Min.} \quad & f(\boldsymbol{y}) = y_1^2 + y_2^2 + 3y_3^2 + 4y_4^2 + 2y_5^2 \\ & \qquad -8y_1 - 2y_2 - 3y_3 - y_4 - 2y_5 \\ \text{Sujeito a} \quad & y_1 + 2y_2 + 2y_3 + y_4 + 6y_5 \leq 800 \\ & 2y_1 + y_2 + 6y_3 \leq 200 \\ & y_3 + y_4 + 5y_5 \leq 200 \\ & y_1 + y_2 + y_3 + y_4 \geq 48 \\ & y_2 + y_4 + y_5 \geq 34 \\ & 6y_1 + 7y_5 \geq 104 \\ & 55 \leq y_1 + y_2 + y_3 + y_4 + y_5 \leq 400 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq y_i \leq 99$ $(i = 1, \ldots, 5)$. A solução ótima é $\boldsymbol{y}^* = (16, 22, 5, 5, 7)$ e $f(\boldsymbol{y}^*) = 807$.

Problema 2 (CHERN; JAN, 1986):

$$\begin{aligned} \text{Max.} \quad & f(\boldsymbol{y}) = (1-(1-0.98)^{y_1}(1-0.92)^{y_2}) \\ \text{Sujeito a} \quad & 11y_1 + 5y_2 \leq 23 \\ & 4y_1 + 6y_2 \leq 12 \\ & y_1 + y_2 \geq 1 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq y_i \leq 10$ $(i = 1, 2)$. A solução ótima é $\boldsymbol{y}^* = (2, 0)$ e $f(\boldsymbol{y}^*) = 0.9996$.

Problema 3 (SRIVASTAVA; FAHIM, 2001):

$$\begin{aligned} \text{Min.} \quad & f(\boldsymbol{y}) = 13y_1 - 5y_2^2 + 30.2y_2 - y_1^2 + 10y_3 + 2.5y_3^2 \\ \text{Sujeito a} \quad & 2y_1 + 4y_2 + 5y_3 \leq 10 \\ & y_1 + y_2 + y_3 \leq 5 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq y_i \leq 30$ $(i = 1, 2, 3)$. A solução ótima é $\boldsymbol{y}^* = (3, 1, 0)$ e $f(\boldsymbol{y}^*) = 55.2$

Problema 4 (SRIVASTAVA; FAHIM, 2001):

$$\begin{aligned} \text{Min.} \quad & f(\boldsymbol{y}) = y_1^2 + y_2^2 + 2y_3^2 + y_4^2 - 5y_1 - 21y_3 + 7y_4 \\ \text{Sujeito a} \quad & \sum_{i=1}^{4} y_i^2 + \sum_{i=0}^{3} (-1)^i y_{i+1} \leq 8 \\ & y_1 - 1 + 2y_2^2 + y_3^2 + y_4(2y_4 - 1) \leq 10 \\ & 2y_1(y_1 + 1) + y_2(y_2 - 1) + y_3^2 - y_4 \leq 5 \end{aligned}$$

Os limites inferiores e superiores são $-5 \leq y_i \leq 5$ $(i = 1, 2, 3, 4)$. A solução ótima é $\boldsymbol{y}^* = (0, 1, 2, -1)$ e $f(\boldsymbol{y}^*) = -44$.

Problema 5 (RUNARSSON; YAO, 2000):

$$\begin{aligned} \text{Min.} \quad & f(\boldsymbol{y}) = 5\sum_{i=1}^{4} y_i - 5\sum_{i=1}^{4} y_i^2 - \sum_{i=5}^{13} y_i \\ \text{Sujeito a} \quad & 2y_1 + 2y_2 + y_{10} + y_{11} - 10 \leq 0 \\ & 2y_1 + 2y_3 + y_{10} + y_{12} - 10 \leq 0 \\ & 2y_2 + 2y_3 + y_{11} + y_{12} - 10 \leq 0 \\ & -2y_4 - y_5 + y_{10} \leq 0 \\ & -2y_6 - y_7 + y_{11} \leq 0 \\ & -2y_8 - y_9 + y_{12} \leq 0 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq y_i \leq 1$ $(i = 1, \ldots, 9)$, $0 \leq y_i \leq 100$ $(i = 10, 11, 12)$ e $0 \leq y_{13} \leq 1$. A solução ótima é $\boldsymbol{y}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ e $f(\boldsymbol{y}^*) = -15$.

Problema 6 (DEEP et al., 2009):

$$\begin{aligned} \text{Min.} \quad & f(\boldsymbol{y}) = y_1^2 + y_1 y_2 2y_2^2 - 6y_1 - 2y_2 - 12y_3 \\ \text{Sujeito a} \quad & 2y_1^2 + y_2^2 \leq 15 \\ & -y_1 + 2y_2 + y_3 \leq 3 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq y_i \leq 10$ $(i = 1, 2, 3)$. A solução ótima é $\boldsymbol{y}^* = (2, 0, 5)$ e $f(\boldsymbol{y}^*) = -68$.

Problema 7 (DEEP et al., 2009):

$$\begin{aligned} \text{Min.} \quad & f(\boldsymbol{y}) = y_1^2 + y_2^2 + y_3^2 + y_4^2 + y_5^2 \\ \text{Sujeito a} \quad & y_1 + 2y_2 + y_4 \geq 4 \\ & y_2 + 2y_3 \geq 3 \\ & y_1 + 2y_5 \geq 5 \\ & y_1 + 2y_2 + 2y_3 \leq 6 \\ & 2y_1 + y_3 \leq 4 \\ & y_1 + 4y_5 \leq 13 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq y_i \leq 3 \quad (i = 1, 2, \ldots, 5)$. A solução ótima é $\boldsymbol{y}^* = (1, 1, 1, 1, 2)$ e $f(\boldsymbol{y}^*) = 8$.

Problema 8 (DEEP et al., 2009):

$$\begin{aligned} \text{Min.} \quad & f(\boldsymbol{y}) = y_1 y_7 + 3y_2 y_6 + y_3 y_5 + 7y_4 \\ \text{Sujeito a} \quad & y_1 + y_2 + y_3 \geq 6 \\ & y_4 + y_5 + 6y_6 \geq 8 \\ & y_1 y_6 + y_2 + 3y_5 \geq 7 \\ & 4y_2 y_7 + 3y_4 y_5 \geq 25 \\ & 3y_1 + 2y_3 + y_5 \geq 7 \\ & 3y_1 y_3 + 6y_4 + 4y_5 \leq 20 \\ & 4y_1 + 2y_3 + y_6 y_7 \leq 15 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq y_i \leq 4$ $(i = 1, 2, 3)$, $0 \leq y_i \leq 2$ $(i = 4, 5, 6)$ e $0 \leq y_7 \leq 6$. A solução ótima é $\boldsymbol{y}^* = (0, 2, 4, 0, 2, 1, 4)$ e $f(\boldsymbol{y}^*) = 14$.

Os parâmetros utilizados no DEGL e no NM são listados nas Tabelas 8 e 9, respectivamente.

A fim de analisar estatisticamente o desempenho do algoritmo DE-E-NM, a Tabela 13 mostra o ótimo global, melhor solução encontrada, entre parenteses o nível de satisfação, a média e desvio padrão, respectivamente. Os problemas 1 a 4 foram

executados 20 vezes, o problema 5 foi executado 100 vezes e os problemas 6 a 8 foram executados 30 vezes, seguindo os experimentos de Schneider e Krohling (2014), Runarsson e Yao (2000) e Deep et al. (2009), respectivamente.

Tabela 13 – Resultados obtidos pelo DE-E-NM para problemas de otimização não-linear inteira com restrições.

Problema	$f(\boldsymbol{y}^*)$	Melhor(α)	Média	Desvio padrão
1	807	807(1)	807	0
2	0.9996	0.9996(1)	0.9996	0
3	55.2	55.2(1)	55.2	0
4	-44	-44(1)	-44	0
5	-15	-15(1)	-15	0
6	-68	-68(1)	-68	0
7	8	8(1)	8	0
8	14	14(1)	14	0

A fim de comparar o desempenho do DE-E-NM, são consideradas as taxas de sucesso obtidas por DE+TOPSIS, ((μ, λ)-ES) e MI-LXPM. A Tabela 14 mostra a taxa de sucesso obtida pelos algoritmos. Os números em negrito indicam os melhores resultados.

Tabela 14 – Comparação entre DE-E-NM, DE+TOPSIS, (μ, λ)-ES e MI-LXPM.

Problema	DE-E-NM	DE+TOPSIS	(μ, λ)-ES	MI-LXPM
1	**100%**	**100%**	–	–
2	**100%**	**100%**	–	–
3	**100%**	**100%**	–	–
4	**100%**	**100%**	–	–
5	**100%**	–	**100%**	–
6	**100%**	–	–	**100%**
7	**100%**	–	–	**100%**
8	**100%**	–	–	71%

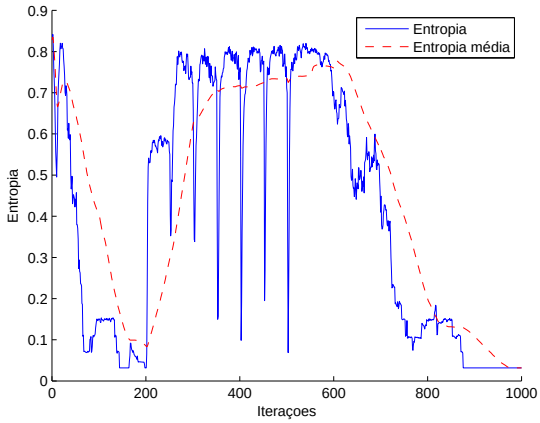
Nota-se pela Tabela 14 que o DE-E-NM obteve 100% de sucesso em todos os *benchmarks* testados, igualando o desempenho dos demais algoritmos. Além disso, o DE-E-NM superou o MI-LXPM no problema 8, com uma taxa de sucesso de 71%.

A fim de analisar a eficiência do algoritmo DE-E-NM, o tempo computacional foi medido e é mostrado, em segundos, na Tabela 15.

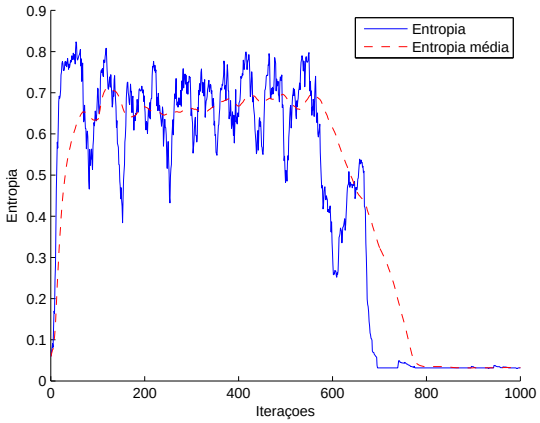
As Figuras 16 e 17 mostram a evolução da entropia durante as iterações do DE nos problemas 1 a 4 e 5 a 8, respectivamente. A linha sólida azul indica a variação da entropia ($entr$) da população. A linha pontilhada vermelha indica a média da entropia ($meanEntr$) sobre as 100 iterações.

Tabela 15 – Tempos médios de execução para problemas de otimização não-linear inteira com restrições.

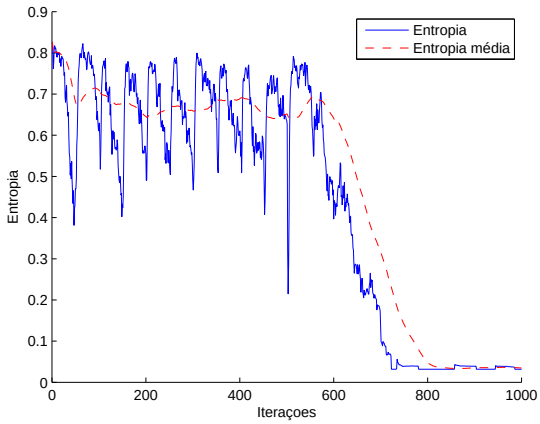
Problema	Tempo computacional
1	28.703084
2	21.098318
3	23.638317
4	25.935122
5	42.556333
6	26.866047
7	23.556234
8	32.068678



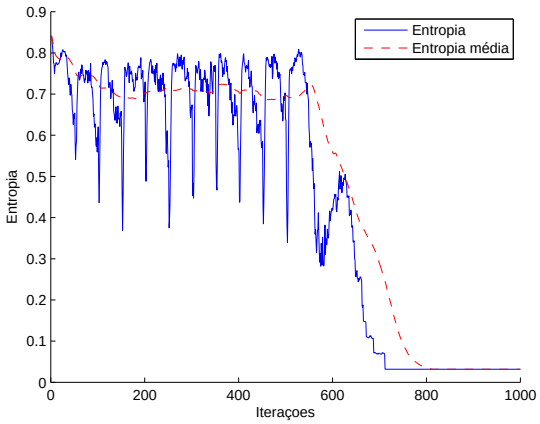
(a) Entropia e entropia média no problema 1.



(b) Entropia e entropia média no problema 2.



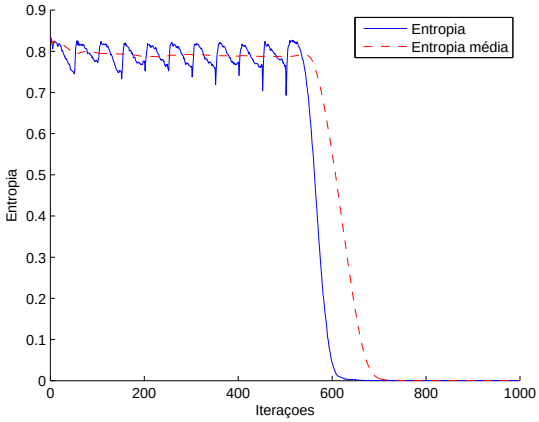
(c) Entropia e entropia média no problema 3.



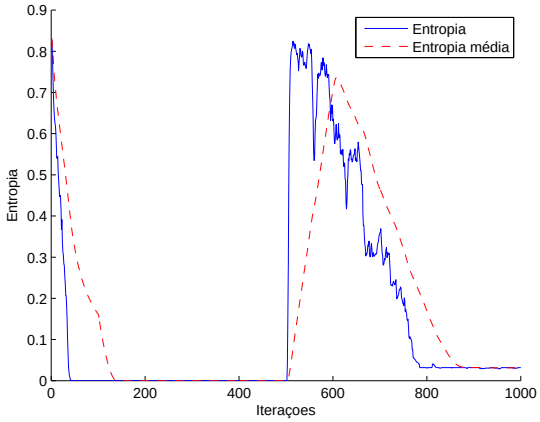
(d) Entropia e entropia média no problema 4.

Figura 16 – Entropia e entropia média nos problemas 1 a 4.

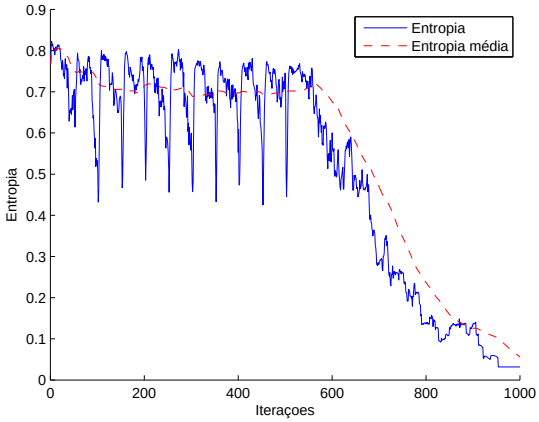
Nota-se pelas Figuras 16 e 17 que a entropia varia com inúmeras quedas súbitas, devido à relaxação do método *Alfa Constrained*. Ao final da busca, a entropia cai próximo a zero, indicando que o método aceita somente soluções factíveis. Além disso, observa-se um comportamento particular nos problemas 5 e 6 devido a uniformidade na variação da entropia e ao forte mínimo local entre as iterações 150 a 500, respectivamente.



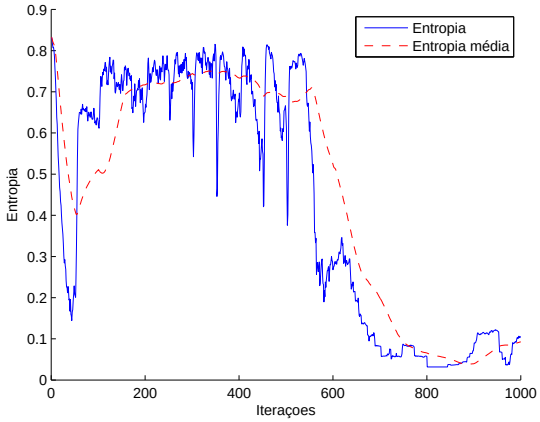
(a) Entropia e entropia média no problema 5.



(b) Entropia e entropia média no problema 6.



(c) Entropia e entropia média no problema 7.



(d) Entropia e entropia média no problema 8.

Figura 17 – Entropia e entropia média nos problemas 5 a 8.

5.4 Problemas de otimização mista

Nesta seção são considerados problemas de otimização não-linear inteira mista que são os problemas mais genéricos em otimização mono-objetivo, que podem ter variáveis contínuas e inteiras. Uma vez que a função objetivo e restrições podem ser não-lineares, estes problemas são considerados desafiadores e são definidos na Seção 2.2 (SCHLÜTER; EGEA; BANGA, 2009).

A fim de avaliar o desempenho do algoritmo DE-E-NM, 14 *benchmarks* foram utilizados e os resultados foram comparados ao *Real-integer-discrete-coded Differential Evolution* (ridDE), proposto por Datta e Figueira (2013) e *Real-integer-discrete-coded Particle Swarm Optimization* (ridPSO), proposto por Datta e Figueira (2011).

Os problemas 1–9, 10–12, são formulações de problemas do mundo real como síntese de processos químicos e sistemas de confiabilidade, respectivamente. Os problemas 13 e 14 são formulações de problemas do mundo real como projetos de recipientes de pressão (do inglês *pressure vessel*) e otimização de processos de trituração (do inglês *grinding process optimization*), respectivamente.

Problema 1:

$$\begin{aligned} \text{Minimizar} \quad & f(\boldsymbol{x},\boldsymbol{y}) = 7.5y_1 + 6.4x_1 + 5.5y_2 + 6.0x_2 \\ \text{Sujeito a} \quad & 0.8x_1 + 0.67x_2 = 10 \\ & x_1 - 20y_1 \leq 0 \\ & x_2 - 20y_1 \leq 0 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq x_i \leq 20$ $(i = 1, 2)$, $0 \leq y_j \leq 1$ $(j = 1, 2)$. A solução ótima é $\boldsymbol{x}^* = (12.5, 0)$, $\boldsymbol{y}^* = (1, 0)$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 87.5$.

Problema 2:

$$\begin{aligned} \text{Minimizar} \quad & f(\boldsymbol{x},\boldsymbol{y}) = 2x_1 + 3x_2 + 1.5y_1 + 2y_2 - 0.5y_3 \\ \text{Sujeito a} \quad & x_1^2 + y_1 = 1.25 \\ & x_2^{1.5} + 1.5y_2 = 3 \\ & x_1 + y_1 \leq 1.6 \\ & 1.333x_2 + y_2 \leq 3 \\ & -y_1 - y_2 + y_3 \leq 0 \end{aligned}$$

Os limites inferiores e superiores são $0 \leq x_i \leq 2$ $(i = 1, 2)$, $0 \leq y_j \leq 1$ $(j = 1, 2, 3)$. A solução ótima é $\boldsymbol{x}^* = (1.117984, 1.310199)$, $\boldsymbol{y}^* = (0, 1, 1)$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 7.666566$.

Problema 3:

$$\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x},\boldsymbol{y}) = (y_1-1)^2 + (y_2-2)^2 + (y_3-1)^2 - ln(y_4+1) \\
& +(x_1-1)^2 + (x_2-2)^2 + (x_3-3)^2 \\
\text{Sujeito a} \quad & y_1+y_2+y_3+x_1+x_2+x_3 \leq 5 \\
& y_3^2+x_1^2+x_2^2+x_3^2 \leq 5.5 \\
& y_1+x_1 \leq 1.2 \\
& y_2+x_2 \leq 1.8 \\
& y_3+x_3 \leq 2.5 \\
& y_4+x_1 \leq 1.2 \\
& y_2^2+x_2^2 \leq 1.64 \\
& y_3^2+x_3^2 \leq 4.25 \\
& y_2^2+x_3^2 \leq 4.64
\end{aligned}$$

Os limites inferiores e superiores são $0 \leq x_1 \leq 1.2$, $0 \leq x_2 \leq 1.281$, $0 \leq x_3 \leq 2.062$ e $0 \leq y_j \leq 1$ $(j = 1, \ldots, 4)$. A solução ótima é $\boldsymbol{x}^* = (0.2, 0.8, 1.9078)$ e $\boldsymbol{y}^* = (1, 1, 0, 1)$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 4.579582$.

Problema 4:

$$\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x},\boldsymbol{y}) = 2x_1 + y_1 \\
\text{Sujeito a} \quad & 1.25 - x_1^2 - y_1 \leq 0 \\
& x_1 + y_1 \leq 1.6
\end{aligned}$$

Os limites inferiores e superiores são $0 \leq x_1 \leq 1.6$, $0 \leq y_1 \leq 1$. A solução ótima é $\boldsymbol{x}^* = 0.5$, $\boldsymbol{y}^* = 1$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 2$.

Problema 5:

$$\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x},\boldsymbol{y}) = -y_1 + 2x_1 - ln(x_1/2) \\
\text{Sujeito a} \quad & -x_1 - ln(x_1/2) + y_1 \leq 0
\end{aligned}$$

Os limites inferiores e superiores são $0.5 \leq x_1 \leq 1.4$ e $0 \leq y_1 \leq 1$. A solução ótima é $\boldsymbol{x}^* = 1.374825$, $\boldsymbol{y}^* = 1$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 2.124470$.

Problema 6:

$$\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x},\boldsymbol{y}) = -0.7y_1 + 5(x_1 - 0.5)^2 + 0.8 \\
\text{Sujeito a} \quad & -e(x_1 - 0.2) - x_2 \leq 0 \\
& x_2 + 1.1y_1 + 1.0 \leq 0 \\
& x_1 - y_1 \leq 0.2
\end{aligned}$$

Os limites inferiores e superiores são $0.2 \leq x_1 \leq 1$, $-2.22554 \leq x_2 \leq -1$ e $0 \leq y_1 \leq 1$. A solução ótima é $\boldsymbol{x}^* = (0.94194, -2.1)$, $\boldsymbol{y}^* = 1$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 1.076555$.

Problema 7:

$$\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x}, \boldsymbol{y}) = 7.5y_1 + 5.5(1 - y_1) + 7x_1 + 6x_2 + \frac{50(1 - y_1)}{0.8(1 - e^{-0.4x_2})} + \frac{50y_1}{0.9(1 - e^{-0.5x_1})} \\
\text{Sujeito a} \quad & 0.9(1 - e^{-0.5x_1}) - 2y_1 \leq 0 \\
& 0.8(1 - e^{-0.4x_2}) - 2(1 - y_1) \leq 0 \\
& x_1 \leq 10y_1 \\
& x_2 \leq 10(1 - y_1)
\end{aligned}$$

Os limites inferiores e superiores são $0 \leq x_i \leq 10$ $(i = 1, 2)$ e $0 \leq y_1 \leq 1$. A solução ótima é $\boldsymbol{x}^* = (3.514, 0)$, $\boldsymbol{y}^* = 1$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 99.239635$.

Problema 8:

$$\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x}, \boldsymbol{y}) = (y_1 - 1)^2 + (y_2 - 1)^2 + (y_3 - 1)^2 - \ln(y_4 + 1) \\
& +(x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 \\
\text{Sujeito a} \quad & y_1 + y_2 + y_3 + x_1 + x_2 + x_3 \leq 5 \\
& y_3^2 + x_1^2 + x_2^2 + x_3^2 \leq 5.5 \\
& y_1 + x_1 \leq 1.2 \\
& y_2 + x_2 \leq 1.8 \\
& y_3 + x_3 \leq 2.5 \\
& y_4 + x_1 \leq 1.2 \\
& y_2^2 + x_2^2 \leq 1.64 \\
& y_3^2 + x_3^2 \leq 4.25 \\
& y_2^2 + x_3^2 \leq 4.64
\end{aligned}$$

Os limites inferiores e superiores são $0 \leq x_1 \leq 1.2$, $0 \leq x_2 \leq 1.8$, $0 \leq x_3 \leq 2.5$ e $0 \leq y_j \leq 1$ $(j = 1, \ldots, 4)$. A solução ótima é $\boldsymbol{x}^* = (0.199998, 1.280614, 1.954489)$, $\boldsymbol{y}^* = (1, 0, 0, 1)$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 3.557464$.

Problema 9:

$$\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x}, \boldsymbol{y}) = 5.357854x_1^2 + 0.835689y_1x_3 + 37.29329y_1 - 40792.141 \\
\text{Sujeito a} \quad & 0.0056858y_2x_3 + 0.0006262y_1x_2 - 0.0022053x_1x_3 \leq 6.665593 \\
& 0.0071317y_2x_3 + 0.0029955y_1y_2 + 0.0021813x_1^2 \leq 29.48751 \\
& 0.0047026x_1x_3 + 0.0012547y_1x_1 + 0.0019085x_1x_2 \leq 15.699039
\end{aligned}$$

Os limites inferiores e superiores são $27 \leq x_i \leq 45$ $(i = 1, 2, 3)$, $78 \leq y_1 \leq 45$ e $33 \leq y_2 \leq 45$. A solução ótima é $\boldsymbol{x}^* = (27, x', 27)$, $\boldsymbol{y}^* = (78, 37)$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = -32217.427780$, onde x' é qualquer valor entre 27 e 45.

Problema 10:

$$
\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x},\boldsymbol{y}) = -\prod_{j=1}^{10}[1-(1-p_j)^{y_j}] \\
\text{Sujeito a} \quad & \prod_{j=1}^{10}(a_{ij}y_j^2 + c_{ij}y_j) \leq b_i \\
\text{onde} \quad & p_j = (0.81, 0.93, 0.92, 0.96, 0.99, 0.89, 0.85, 0.83, 0.94, 0.92) \\
& a_{ij} = \begin{bmatrix} 2 & 7 & 3 & 0 & 5 & 6 & 9 & 4 & 8 & 1 \\ 4 & 9 & 2 & 7 & 1 & 0 & 8 & 3 & 5 & 6 \\ 5 & 1 & 7 & 4 & 3 & 6 & 0 & 9 & 8 & 2 \\ 8 & 3 & 5 & 6 & 9 & 7 & 2 & 4 & 0 & 1 \end{bmatrix} \\
& c_{ij} = \begin{bmatrix} 7 & 1 & 4 & 6 & 8 & 2 & 5 & 9 & 3 & 3 \\ 4 & 6 & 5 & 7 & 2 & 6 & 9 & 8 & 1 & 0 \\ 1 & 10 & 3 & 5 & 4 & 7 & 8 & 9 & 4 & 6 \\ 2 & 3 & 2 & 5 & 7 & 8 & 6 & 10 & 9 & 1 \end{bmatrix} \\
& b = [2.000.315.700.93] \cdot 10^{13}
\end{aligned}
$$

Os limites inferiores e superiores são $0 \leq y_j \leq 10$ $(j = 1, 2, \ldots, 10)$. A solução ótima é $\boldsymbol{y}^* = (2, 2, 2, 1, 1, 2, 2, 2, 1, 2)$ e $f(\boldsymbol{y}^*) = -0.793323$.

Problema 11:

$$
\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x},\boldsymbol{y}) = -\prod_{j=1}^{4} R_j \\
\text{Sujeito a} \quad & \sum_{j=1}^{4} d_{1j}y_j^2 \leq 100 \\
& \sum_{j=1}^{4} d_{2j}\left[y_j + e^{\frac{y_j}{4}}\right] \leq 150 \\
& \sum_{j=1}^{4} d_{3j}y_j e^{\frac{y_j}{4}} \leq 160 \\
\text{onde} \quad & R_1 = 1 - q_1[(1-\beta_1)q_1 + \beta_1]^{y_1 - 1} \\
& R_2 = 1 - \frac{\beta_2 q_2 + p_2 q_2^{y_2}(1-\beta_2)^{y_2}}{p_2 + \beta_2 q_2} \\
& R_3 = 1 - q_3^{y_3} \\
& R_4 = 1 - q_4[(1-\beta_4)q_4 + \beta_4]^{y_4 - 1} \\
& p_j = (0.93, 0.92, 0.94, 0.91) \\
& q_j = (0.07, 0.08, 0.06, 0.09) \\
& \beta_j = (0.2, 0.06, 0.0, 0.3) \\
& d_{ij} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 7 & 7 & 5 & 7 \\ 7 & 8 & 8 & 6 \end{bmatrix}
\end{aligned}
$$

Os limites inferiores e superiores são $1 \leq y_i \leq 6$ $(i = 1, 2, 4)$ e $1 \leq y_3 \leq 5$. A solução ótima é $\boldsymbol{y}^* = (3, 3, 2, 3)$ e $f(\boldsymbol{y}^*) = -0.974565$.

Problema 12:

$$
\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x}, \boldsymbol{y}) = -\prod_{j=1}^{4}[1 - (1 - x_j)^{y_j}] \\
\text{Sujeito a} \quad & \sum_{j=1}^{4} v_j y_j^2 \leq 250 \\
& \sum_{j=1}^{4} \alpha_j \left[\frac{-1000}{\ln(x_j)}\right]^{\beta_j} [y_j + e^{\frac{y_j}{4}}] \leq 400 \\
& \sum_{j=1}^{4} w_j y_j e^{\frac{y_j}{4}} \leq 500 \\
\text{onde} \quad & v_j = (1, 2, 3, 2) \\
& w_j = (6, 6, 8, 7) \\
& \alpha_j = (1.0, 2.3, 0.3, 2.3) \cdot 10^{-5} \\
& \beta_j = (1.5, 1.5, 1.5, 1.5)
\end{aligned}
$$

Os limites inferiores são $0.5 \leq x_i \leq 1 - 10^6$ $(i = 1, 2, 3, 4)$ e $1 \leq y_j \leq 10$ $(j = 1, 2, 3, 4)$. A solução ótima é $\boldsymbol{x}^* = (0.898590, 0.847115, 0.953296, 0.882140)$, $\boldsymbol{y}^* = (5, 6, 4, 5)$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = -0.999949$.

Problema 13:

$$
\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x}, \boldsymbol{y}) = 0.6224 x_1 x_2 y_1 + 1.7781 x_1^2 y_2 + 3.1661 x_2 y_1^2 + 19.84 x_1 y_1^2 \\
\text{Sujeito a} \quad & 0.0193 x_1 - y_1 \leq 0 \\
& 0.00954 x_1 - y_2 \leq 0 \\
& x_2 - 240 \leq 0 \\
& 96000 - \frac{4}{3}\pi x_1^3 - \pi x_1^2 x_2 \leq 0
\end{aligned}
$$

Os limites inferiores e superiores são $25 \leq x_1 \leq 150$, $25 \leq x_2 \leq 240$ e $y_j = \{0.0625, 0.125, \ldots, 1.1875, 1.25\}$ $(j = 1, 2)$. A solução ótima é $\boldsymbol{x}^* = (38.860099, 221.365548)$, $\boldsymbol{y}^* = (0.750000, 0.375000)$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = 5850.383760$.

Problema 14:

$$
\begin{aligned}
\text{Minimizar} \quad & f(\boldsymbol{x}, \boldsymbol{y}) = -x_1 x_2 \\
\text{Sujeito a} \quad & 0.145 x_1^{0.1939} x_2^{0.7071} y_1^{-0.2343} \leq 0.3 \\
& 29.67 x_1^{0.4167} x_2^{-0.8333} \leq 7
\end{aligned}
$$

Os limites inferiores e superiores são $5 \leq x_1 \leq 30$, $8.6 \leq x_2 \leq 13.4$ e $y_1 = \{120, 140, 170, 200, 230, 270, 325, 400, 500\}$. A solução ótima é $\boldsymbol{x}^* = (5.607028, 13.400000)$, $\boldsymbol{y}^* = 500$ e $f(\boldsymbol{x}^*, \boldsymbol{y}^*) = -75.134168$.

Os parâmetros utilizado DEGL e no NM são listados nas Tabelas 8 e 9, respectivamente.

A fim de quantificar estatisticamente o desempenho do DE-E-NM, são considerados os resultados obtidos pelo ridDE e ridPSO. A Tabela 16 mostra a média e desvio padrão dos algoritmos após 30 execuções de cada *benchmark*. Os números em negrito indicam os melhores resultados.

Tabela 16 – Resultados obtidos pelo DE-E-NM para problemas de otimização mista.

	DE-E-NM		ridDE		ridPSO	
Problema	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
1	**87.500000**	**0.000000**	**87.500000**	**0.000000**	**87.5000000**	**0.000000**
2	7.904719	0.080533	**7.666747**	**0.011884**	7.667003	0.011899
3	4.614687	0.192277	**4.579582**	**0.000000**	**4.579582**	**0.000000**
4	**2.000000**	**0.000000**	**2.000000**	**0.000000**	**2.000000**	**0.000000**
5	2.557817	0.000000	**2.124519**	**0.005323**	2.124532	0.007573
6	1.151708	0.087423	1.076771	0.010590	**1.078348**	**0.021125**
7	**99.239635**	**0.000000**	**99.239635**	**0.000000**	**99.239635**	**0.000000**
8	4.103927	0.683057	**3.557513**	**0.007497**	3.557947	0.008003
9	**-32217.427780**	**0.000000**	**-32217.427780**	**0.000000**	**-32217.427780**	**0.000000**
10	**-0.793323**	**0.000000**	**-0.793323**	**0.000000**	**-0.793323**	**0.000000**
11	**-0.974565**	**0.000000**	**-0.974565**	**0.000000**	**-0.974565**	**0.000000**
12	**-0.999947**	**0.000001**	-0.999914	0.003569	-0.999889	0.000764
13	5882.330951	32.112845	5872.103874	3.369841	**5850.514287**	**0.571243**
14	**-75.134173**	**0.000000**	-75.134149	0.002892	-75.134142	0.000794

Nota-se pela Tabela 16 que o DE-E-NM obteve as melhores médias nos problemas 12 e 14 e igualou o desempenho do ridDE nos problemas 1, 4, 7, 9, 10 e 11.

A fim de analisar a eficiência do algoritmo DE-E-NM, o tempo computacional foi medido e é mostrado, em segundos, na Tabela 17.

Tabela 17 – Tempos médios de execução para problemas de otimização não-linear inteira mista.

Problema	Tempo computacional
1	26.291871
2	25.989680
3	26.128718
4	20.004615
5	20.239953
6	18.980955
7	23.546651
8	20.484518
9	29.894912
10	21.896874
11	26.337523
12	22.449492
13	16.959390
14	20.088015

A fim de comparar as melhores soluções encontradas, a Tabela 18 mostra a melhor aptidão encontrada, o autor e a solução ótima, respectivamente.

Tabela 18 – Soluções ótimas conhecidas para os *benchmarks* dos problemas de otimização não-linear inteira mista.

Problema	Melhor solução	Fonte	Solução ótima
1	87.500000	(LIAO, 2010)	$\boldsymbol{x} = [12.500000, 0]$ e $\boldsymbol{y} = [1, 0]$
2	7.666566	(LIAO, 2010)	$\boldsymbol{x} = [1.117984, 1.310199]$ e $\boldsymbol{y} = [0, 1, 1]$
3	4.579582	(LIAO, 2010)	$\boldsymbol{x} = [0.200000, 0.800000, 1.907878]$ e $\boldsymbol{y} = [1, 1, 0, 1]$
4	2.000000	(LIAO, 2010)	$\boldsymbol{x} = 0.500000$ e $\boldsymbol{y} = 1$
5	2.124468	(DATTA; FIGUEIRA, 2013)	$\boldsymbol{x} = 1.374825$ e $\boldsymbol{y} = 1$
6	1.076543	Este trabalho	$\boldsymbol{x} = [0.941937, -2.1]$ e $\boldsymbol{y} = 1$
7	99.239635	(DATTA; FIGUEIRA, 2013)	$\boldsymbol{x} = [3.514000, 0]$ e $\boldsymbol{y} = 1$
8	3.557461	Este trabalho	$\boldsymbol{x} = [0.200000, 1.280624, 1.954482]$ e $\boldsymbol{y} = [1, 0, 0, 1]$
9	-32217.427780	(LIAO, 2010)	$\boldsymbol{x} = [27, qualquer, 27]$ e $\boldsymbol{y} = [78, 37]$
10	-0.793323	(DATTA; FIGUEIRA, 2013)	$\boldsymbol{y} = [2, 2, 2, 1, 1, 2, 2, 2, 1, 2]$
11	-0.974565	(LIAO, 2010)	$\boldsymbol{y} = [3, 3, 2, 3]$
12	-0.999952	Este trabalho	$\boldsymbol{x} = [0.908087, 0.850502, 0.949143, 0.880943]$ e $\boldsymbol{y} = [5, 6, 4, 5]$
13	5850.383760	(DATTA; FIGUEIRA, 2011)	$\boldsymbol{x} = [38.860099, 221.365548]$ e $\boldsymbol{y} = [0.75, 0.375]$
14	-75.134168	(DATTA; FIGUEIRA, 2013)	$\boldsymbol{x} = [5.607028, 13.4]$ e $\boldsymbol{y} = 500$

O algoritmo DE-E-NM encontrou o mínimo global em cada um dos *benchmarks* testados em pelo menos uma das 30 execuções. Além disso, melhorou as soluções dos problemas 6, reportada em Liao (2010), 8 e 12, reportada em Datta e Figueira (2013). Sendo assim, a abordagem proposta mostra eficácia para MINLP igualando às abordagens utilizadas para comparação e, em alguns casos, superando os melhores resultados encontrados na literatura.

6 Conclusão

O balanceamento entre exploração e intensificação é um dos principais pontos em algoritmos evolutivos, e foi o principal foco deste trabalho. Foi desenvolvido um algoritmo híbrido onde a busca global é realizada pelo algoritmo Evolução Diferencial com topologia global-local e a busca local pelo algoritmo Nelder-Mead. Além disso, foi proposto um novo mecanismo de chaveamento entre os algoritmos, baseado na entropia da população.

Para avaliar o desempenho desta abordagem, três tipos de problemas foram testados. Os primeiros foram os problemas de otimização não-linear inteira sem restrições. Três *benchmarks*, divididos em dez instâncias foram utilizados. A taxa de sucesso dos algoritmos foi considerada para compará-los e a abordagem proposta superou o algoritmo D&BMS em cinco das 10 instâncias utilizadas. Em seguida, os problemas de otimização não-linear inteira com restrições foram utilizados e oito *benchmarks* foram testados. O método utilizado para tratamento de restrições foi o *Alpha Constrained*. A fim de comparar o desempenho do algoritmo DE-E-NM, foram consideradas as taxas de sucesso dos algoritmo DE+TOPSIS, (μ, λ)-ES e MI-LXPM. O método proposto igualou o desempenho de 100% dos algoritmos DE+TOPSIS e (μ, λ)-ES nos problemas 1-7 e superou o MI-LXPM no problema 8, que obteve 71% de sucesso.

A seguir, o algoritmo híbrido entre Evolução Diferencial e Nelder-Mead foi aplicado a problemas de otimização não-linear inteira mista. A fim de quantificar estatisticamente o desempenho do método proposto, foram considerados os resultados obtidos pelos algoritmos ridDE e ridPSO. O algoritmo DE-E-NM igualou o desempenho de 8 dos 14 *benchmarks* utilizados. Além disso, superou os melhores resultados reportados na literatura em 3 instâncias e mostrou sua eficácia para esse tipo de problema.

O método de chaveamento proposto neste trabalho tem o objetivo de balancear as buscas global e local em algoritmos evolutivos ou de inteligência coletiva. A busca local é ativada e favorecida de acordo com a diversidade da população, que é medida através da entropia. Apesar das justificativas de utilizar os algoritmos DE e NM, a abordagem proposta pode ser utilizada em qualquer algoritmo evolutivo ou de inteligência coletiva, e método para tratamento de restrições. Além disso, o mecanismo se adapta às características do problema e da evolução da busca, logo, não é necessário ter conhecimentos específicos do problema.

Para dar seguimento a esta pesquisa, sugere-se utilizar outros algoritmos bio-inspirados, mecanismos de busca local e outros métodos de tratamento de restrições, aplicados a problemas de otimização em dois níveis e com múltiplas soluções.

Referências

ABRAMSON, M. A.; AUDET, C.; CHRISSIS, J. W.; WALSTON, J. G. Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, v. 3, n. 1, p. 35–47, 2009.

ARORA, J. S.; HUANG, M. W.; HSIEH, C. C. Methods for optimization of nonlinear problems with discrete variables: a review. *Structural Optimization*, v. 8, n. 2–3, p. 69–85, 1994.

BÄCK, T.; HAMMEL, U.; SCHWEFEL, H.-P. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary computation*, v. 1, n. 1, p. 3–17, 1997.

BARBOSA, H. J.; LEMONGE, A. C. An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In: *Genetic and Evolutionary Computation Conference*. Filadélfia, Estados Unidos: 2002. v. 2, p. 287–294.

BONYADI, M. R.; LI, X.; MICHALEWICZ, Z. A hybrid particle swarm with a time-adaptive topology for constrained optimization. *Swarm and Evolutionary Computation*, n. 18, p. 22–37, 2014.

BOUSSAÏD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. *Information Sciences*, v. 237, p. 82–117, 2013.

BREA, E. Una extensión del método de Nelder Mead a problemas de optimización no lineales enteros mixtos. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, v. 29, n. 3, p. 163–174, 2013.

BREST, J.; GREUNER, S.; BOŠKOVIĆ, B.; MERNIK, M.; ZUMER, V. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, v. 10, n. 6, p. 646–657, 2006.

BROWNLEE, J. *Clever algorithms: nature-inspired programming recipes*. Austrália: Lulu, 2011.

BURGIN, G. On playing two-person zero-sum games against nonminimax players. *IEEE Transactions on Systems Science and Cybernetics*, v. 5, n. 4, p. 369–370, 1969.

CAMPOS, M.; KROHLING, R. A. Entropy-based bare bones particle swarm for dynamic constrained optimization. *Knowledge-Based Systems*, v. 97, p. 203–223, 2016.

CASTRO, L. N. de. Fundamentals of natural computing: an overview. *Physics of Life Reviews*, v. 4, p. 1–36, 2007.

CHERN, M.-S.; JAN, R.-H. Reliability optimization problems with multiple constraints. *IEEE Transactions on Reliability*, v. 35, n. 4, p. 431–436, 1986.

CLERC, M.; KENNEDY, J. The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Transaction on Evolutionary Computation*, v. 6, p. 58–73, 2002.

COELLO, C. A. C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. v. 191, n. 11–12, p. 1245–2187, 2002.

CORNUÉJOLS, G. Revival of the Gomory cuts in the 1990's. *Annals of Operations Research*, v. 149, n. 1, p. 63–66, 2007.

ČREPINŠEK, M.; LIU, S.-H.; MERNIK, M. Exploration and exploitation in evolutionary algorithms: a survey. *ACM Computing Surveys*, v. 45, n. 3, p. 35, 2013.

DANTZIG, B. G. *Linear programming and extensions*. Princeton, New Jersey: Princeton university press, 1998.

DAS, S.; ABRAHAM, A.; CHAKRABORTY, U. K.; KONAR, A. Differential evolution using a neighborhood-based mutation operator. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 3, p. 526–553, 2009.

DAS, S.; SUGANTHAN, P. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, v. 15, n. 1, p. 4–31, 2011.

DATTA, D.; FIGUEIRA, J. R. A real–integer–discrete-coded particle swarm optimization for design problems. *Applied Soft Computing*, n. 11, p. 3625–3633, 2011.

DATTA, D.; FIGUEIRA, J. R. A real–integer–discrete-coded differential evolution. *Applied Soft Computing*, n. 13, p. 3884–3893, 2013.

De Jong, K. A. *Analysis of the behavior of a class of genetic adaptive systems*. Tese (Doutorado) — Universidade de Michigan, Estados Unidos, 1975.

DEB, K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, v. 186, n. 2–4, p. 311–338, 2000.

DEEP, K.; SINGH, K. P.; KANSAL, M. L.; MOHAN, C. A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Applied Mathematics and Computation*, v. 212, n. 2, p. 505–518, 2009.

DOMINGUEZ-ISIDRO, S.; MEZURA-MONTES, E.; LEGUIZAMÓN, G. Performance comparison of local search operators in differential evolution for constrained numerical optimization problems. In: IEEE *Symposium on Differential Evolution (SDE)*. Orlando, Estados Unidos, 2014. p. 1–8.

DORIGO, M. *Optimization, learning and natural algorithms*. Tese (Doutorado) — Politecnico de Milão, Itália, 1992.

DORIGO, M.; STÜTZLE, T. *Ant Colony Optimization*. Massachusetts, Estados Unidos: Bradford Company, 2004.

DUAN, Q.; LIAO, T. W.; YI, H. Z. A comparative study of different local search application strategies in hybrid metaheuristics. *Applied Soft Computing*, n. 13, p. 1464–1477, 2013.

DURAN, M. A.; GROSSMANN, I. E. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming*, v. 36, n. 3, p. 307–339, 1986.

ENGELBRECHT, A. P. *Computational intelligence: an introduction*. South Africa: John Wiley & Sons, 2007.

EXLER, O.; SCHITTKOWSKI, K. A trust region SQP algorithm for mixed-integer nonlinear programming. *Optimization Letters*, v. 1, n. 3, p. 269–280, 2007.

FLETCHER, R.; LEYFFER, S. Solving mixed integer nonlinear programs by outer approximation. *Mathematical programming*, v. 66, n. 1-3, p. 327–349, 1994.

FOGEL, D. B. *System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling*. New Jersey: Ginn Press, 1991.

FOGEL, D. B. *Evolutionary computation: toward a new philosophy of machine intelligence*. New Jersey: John Wiley & Sons, 2006.

FOGEL, D. B.; FOGEL, L. J.; ATMAR, J. W. Meta-evolutionary programming. *IEEE Conference record of the twenty-fifth asilomar conference on Signals, systems and computers*, Pacific Grove, CA, p. 540–545, 1991.

FOGEL, L. J. Autonomous automata. *Industrial Research*, v. 4, n. 2, p. 14–19, 1962.

FONSECA, L.; CAPRILES, P.; BARBOSA, H. J. A stochastic rank-based ant system for discrete structural optimization. In: IEEE *Swarm Intelligence Symposium*. Honolulu, HI, 2007. p. 68–75.

GARCÍA, L. L.; GARCÍA-RÓDENAS, R.; GÓMEZ, G. A. Hybrid meta-heuristic optimization algorithms for time-domain-constrained data clustering. *Applied Soft Computing*, v. 23, p. 319–332, 2014.

GEOFFRION, A. M. Generalized Benders decomposition. *Journal of optimization theory and applications*, v. 10, n. 4, p. 237–260, 1972.

GOLDBERG, D. E. Genetic algorithms and rule learning in dynamic system control. In: *Proceedings of the 1st International Conference on Genetic Algorithms*. Pensilvânia, Estados Unidos: 1985. v. 876, p. 8–15.

HANSEN, N.; OSTERMEIER, A.; GAWELCZYK, A. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In: *International Conference on Genetic Algorithms*. Pensilvânia, Estados Unidos: 1995. p. 57–64.

HERDY, M. Reproductive isolation as strategy parameter in hierarichally organized evolution strategies. In: *International Conference on Parallel Problem Solving from Nature*. Bruxelas, Bélgica: 1992. p. 209.

HO, P. Y.; SHIMIZU, K. Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. *Information Sciences*, v. 177, n. 14, p. 2985–3004, 2007.

HOLLAND, J. H. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. Michigan Press, Oxford, Inglaterra, 1975.

HOMAIFAR, A.; LAI, S. H. V.; QI, X. Constrained optimization via genetic algorithms. *Simulation*, v. 4, n. 62, p. 242–254, 1994.

JOINES, J.; HOUCK, C. R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In: IEEE *World Congress on Computational Intelligence*. Orlando, FL, 1994. p. 579–584.

JUN, W.; YUELIN, G.; LINA, Y. An improved differential evolution algorithm for mixed integer programming problems. In: IEEE *9th International Conference on Computational Intelligence and Security*. Leshan, China, 2013. p. 31–35.

KAZARLIS, S.; PETRIDIS, V. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. *Parallel Problem Solving from Nature*, p. 211–220, 1998.

KENNEDY, J. The behavior of particles. In: *Proceedings of the 7-th Annual Conference on Evolutionary Programming*. Califórnia, Estados Unidos: 1998. v. 1447, p. 581–589.

KENNEDY, J. Bare bones particle swarms. In: IEEE *Swarm Intelligence Symposium*. 2003. p. 80–87.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE *International Conference on Neural Networks*. 1995. v. 4, p. 1942–1948.

KENNEDY, J.; MENDES, R. Population structure and particle swarm performance. In: IEEE *Congress on Evolutionary Computation*. 2002. p. 1671–1676.

KITAYAMAA, S.; ARAKAWAB, M.; YAMAZAKI, K. Differential evolution as the global optimization technique and its application to structural optimization. *Applied Soft Computing*, v. 11, p. 3792–3803, 2011.

KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. Londres, Inglaterra: MIT press, 1992.

KRASNOGOR, N.; HART, W.; SMITH, J. Recent advances in memetic algorithms. *Studies in Fuzziness and Soft Computing Series Springer*, 2004.

KROHLING, R. A. Comunicado interno. PPGI/Ufes, 2015.

LAND, A. H.; DOIG, A. G. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, JSTOR, p. 497–520, 1960.

LEGUIZAMÓN, G.; COELLO, C. A. C. An alternative ACOr algorithm for continuous optimization problems. *Swarm Intelligence*, p. 48–59, 2010.

LI, X. Niching without niching parameters: particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation*, v. 14, n. 1, p. 150–169, 2010.

LIAO, T.; DE OCA, M. A. M.; AYDIN, D.; STÜTZLE, T.; DORIGO, M. An incremental ant colony algorithm with local search for continuous optimization. In: *Proceedings of the 13th annual Conference on Genetic and evolutionary computation*. New York, NY: ACM, 2011. p. 125–132.

LIAO, T.; STÜTZLE, T.; DE OCA, M. A. M.; DORIGO, M. A unified ant colony optimization algorithm for continuous optimization. *European Journal of Operational Research*, v. 234, n. 3, p. 597–609, 2014.

LIAO, T. W. Two hybrid differential evolution algorithms for engineering design optimization. *Applied Soft Computing*, n. 10, p. 1188–1199, 2010.

LIN, Y.-C.; WANG, F.-S.; HWANG, K.-S. A hybrid method of evolutionary algorithms for mixed-integer nonlinear optimization problems. In: IEEE *Congress on Evolutionary Computation*. Washington, Estados Unidos, 1999. p. 2159–2166.

LIU, B.; MA, H.; ZHANG, X.; ZHOU, Y. A memetic co-evolutionary differential evolution algorithm for constrained optimization. In: IEEE *Congress on Evolutionary Computation*. Singapore, 2007. p. 2996–3002.

LIU, S.-H.; MERNIK, M.; BRYANT, B. R. To explore or to exploit: An entropy-driven approach for evolutionary algorithms. *International Journal of Knowledge-based and Intelligent Engineering Systems*, p. 185–206, 2009.

LUCHI, F.; KROHLING, R. A. Differential evolution and Nelder-Mead for constrained non-linear integer optimization problems. *Procedia Computer Science*, v. 55, p. 668–677, 2015.

LUCHI, F.; KROHLING, R. A. Um algoritmo híbrido entre evolução diferencial e Nelder-Mead inteiro para problemas de otimização. *XLVII Simpósio Brasileiro de Pesquisa Operacional*, Porto de Galinhas, Pernambuco, 2015.

MICHALEWICZ, Z. A survey of constraint handling techniques in evolutionary computation methods. *Evolutionary Programming*, v. 4, p. 135–155, 1995.

MICHALEWICZ, Z.; ATTIA, N. Evolutionary optimization of constrained problems. In: *Proceedings of the 3rd Annual Conference on Evolutionary Programming*. 1994. p. 98–108.

MICHALEWICZ, Z.; SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, v. 4, n. 1, p. 1–32, 1996.

MOHAN, C.; NGUYEN, H. A controlled random search technique incorporating the simulated annealing concept for solving integer and mixed integer global optimization problems. *Computational Optimization and Applications*, v. 14, n. 1, p. 103–132, 1999.

NELDER, J. A.; MEAD, R. A simplex method for function minimization. *The Computer Journal*, v. 7, n. 4, p. 308–313, 1965.

NERI, F.; COTTA, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, v. 2, p. 1–14, 2012.

NERI, F.; TOIVANEN, J.; CASCELLA, L.; YEW-SOON O. An adaptive multimeme algorithm for designing HIV multidrug therapies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 4, p. 1–41, 2006.

NI, Q.; DENG, J. Analysis of population diversity of dynamic probabilistic particle swarm optimization algorithms. *Mathematical Problems in Engineering*, v. 2014, p. 1–9, 2014.

POLI, R.; KENNEDY, J.; BLACKWELL, T. Particle swarm optimization: an overview. *Swarm Intelligence*, v. 1, p. 33–57, 2007.

PONSICH, A.; COELLO, C. A hybrid differential evolution-tabu search algorithm for the solution of job-shop scheduling problems. *Applied Soft Computing*, v. 13, n. 1, p. 462–474, 2013.

POWEL, M. J. D. An efficient method for finding minimum of a function of several variables without calculating derivates. *The Computer Journal*, v. 7, p. 155–162, 1964.

PRICE, K. V. Differential evolution vs. the functions of the 2nd ICEO. In: IEEE *International Conference on Evolutionary Computation*. Indianapolis, IN, 1997. p. 153–157.

QIN, A. K.; HUANG, V. L.; SUGANTHAN, P. N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 2, p. 398–417, 2009.

RAHNAMAYAN, S.; TIZHOOSH, H. R.; SALAMA, M. Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, v. 12, n. 1, p. 64–79, 2008.

RECHENBERG, I. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment*, Farnborough p. Library Translation 1122, 1965.

RECHENBERG, I. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Tese (Doutorado) — Technical University of Berlin, Departament of Process Engineering, 1973.

RICHER, T.; BLACKWELL, T. M. The Lévy particle swarm. In: IEEE *Congress on evolutionary computation*. Vancouver, Canadá, 2006. p. 3150–3157.

RODRIGUES, M. de C.; LIMA, B. S. L. P. de; GUIMARÃES, S. Balanced ranking method for constrained optimization problems using evolutionary algorithms. *Information Sciences*, n. 327, p. 71–90, 2016.

ROSCA, J. P. Entropy-driven adaptive representation. *Proceedings of the workshop on genetic programming: From theory to real-world applications*, v. 9, p. 23–32, 1995.

RUNARSSON, T. P.; YAO, X. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, v. 4, n. 3, p. 284–294, 2000.

SCHLÜTER, M. *Nonlinear mixed integer based optimization technique for space applications*. Tese (Doutorado) — Universidade de Birmingham, 2012.

SCHLÜTER, M.; EGEA, J. A.; BANGA, J. R. Extended ant colony optimization for non-convex mixed integer nonlinear programming. *Computers & Operations Research*, n. 36, p. 2217–2229, 2009.

SCHNEIDER, E. R. F. A.; KROHLING, R. A. A hybrid approach using TOPSIS, differential evolution, and tabu search to find multiple solutions of constrained non-linear integer optimization problems. *Knowledge-Based Systems*, v. 62, p. 47–56, 2014.

SCHWEFEL, H.-P. *Kybernetische Evolution als Strategie der exprimentellen Forschung in der Strömungstechnik*. Dissertação (Mestrado) — Technical University of Berlin, Berlin, Alemanha, 1965.

SCHWEFEL, H.-P. *Numerical optimization of computer models*. New York, NY: John Wiley & Sons, Inc., 1981.

SCHWEFEL, H.-P.; RUDOLPH, G. Contemporary evolution strategies. In: *Advances in Artificial Life*. Granada, Espanha: 1995. v. 929, p. 891–907.

SEBAG, M.; SCHOENAUER, M.; RAVISÉ, C. Inductive learning of mutation step-size in evolutionary parameter optimization. *Evolutionary Programming VI*, p. 247–261, 1997.

SHI, Y.; EBERHART, R. C. A modified particle swarm optimizer. In: IEEE *International Conference on Evolutionary Computation*. Anchorage, Alasca, 1998. v. 9, p. 69–73.

SOCHA, K. ACO for continuous and mixed-variable optimization. In: *Ant colony optimization and swarm intelligence*. Bruxelas, Bélgica: Springer, 2004. p. 25–36.

SOCHA, K. *Ant Colony Optimisation for Continuous and Mixed-Variable Domains*. Saarbrücken: VDM Publishing, 2009.

SOCHA, K.; DORIGO, M. Ant colony optimization for continuous domain. *European Journal of Operational Research*, v. 185, p. 1155–1173, 2008.

SPENDLEY, W.; HEXT, G. R.; HIMSWORTH, F. R. Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics*, v. 1962, p. 441–461, 1962.

SRIVASTAVA, V.; FAHIM, A. A two-phase optimization procedure for integer programming problems. *Computers & Mathematics with Applications*, v. 42, n. 12, p. 1585–1595, 2001.

STONE, J. V. *Information Theory: A Tutorial Introduction*. 1st. ed. Inglaterra: Sebtel Press, 2015. 260 p.

STORN, R.; PRICE, K. *Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Technical Report,*. Berkeley, CA, 1995.

TAKAHAMA, T.; SAKAI, S. Learning fuzzy control rules by α constrained Powell's method. In: IEEE *Proceedings of International Fuzzy System Conference*. Seoul, Korea, 1999. v. 2, p. 650–655.

TAKAHAMA, T.; SAKAI, S. Tuning fuzzy control rules by the α constrained method which solves constrained nonlinear optimization problems. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, Wiley Online Library, v. 83, n. 9, p. 1–12, 2000.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by α constrained genetic algorithm (αGA). *Systems and Computers in Japan*, n. 5, p. 11–22, 2004.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations. *IEEE Transactions on Evolutionary Computation*, v. 9, n. 5, p. 437–451, 2005.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by the α constrained particle swarm optimizer. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, n. 3, p. 282–289, 2005.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by ε constrained particle swarm optimizer with ε-level control. *Soft Computing as Transdisciplinary Science and Technology*, p. 1019–1029, 2005.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In: IEEE *Congress on Evolutionary Computation*. Vancouver, Canada, 2006. p. 1–8.

TAKAHAMA, T.; SAKAI, S. Fast and stable constrained optimization by the ε constrained differential evolution. *Pacific Journal of optimization*, v. 5, n. 2, p. 261–282, 2009.

TAKAHAMA, T.; SAKAI, S. Solving difficult constrained optimization problems by the ε constrained differential evolution with gradient-based mutation. In: Constraint-Handling in Evolutionary Optimization. Berlin, Alemanha, 2009. v. 198, p. 51–72.

TAKAHAMA, T.; SAKAI, S. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In: IEEE *Congress on Evolutionary Computation*. Barcelona, Espanha, 2010. p. 1–9.

TAKAHAMA, T.; SAKAI, S. Efficient constrained optimization by the ε constrained adaptive differential evolution. In: IEEE *Congress on Evolutionary Computation*. Barcelona, Espanha, 2010. p. 1–8.

TAKAHAMA, T.; SAKAI, S. Optimization by the ε constrained differential evolution with rough approximation. *Evolutionary Constrained Optimization*, p. 157–180, 2015.

TAKAHAMA, T.; SAKAI, S.; IWANE, N. Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm. In: *Advances in Artificial Intelligence*. Sydney, Austrália: Springer, 2005. p. 389–400.

TAKAHAMA, T.; SAKAI, S.; IWANE, N. Solving nonlinear constrained optimization problems by the ε constrained differential evolution. In: IEEE *International Conference on Systems, Man and Cybernetics*. Taipei, 2006. v. 3, p. 2322–2327.

TESSEMA, B.; YEN, G. G. A self adaptive penalty function based algorithm for constrained optimization. In: IEEE *Congress on Evolutionary Computation*. Vancouver, Canadá, 2006. p. 246–253.

TIAN, P.; MA, J.; ZHANG, D. Non-linear integer programming by Darwin and Boltzmann mixed strategy. *European Journal of Operational Research*, p. 224–235, 1998.

WESTERLUND, T.; PETTERSSON, F. An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, v. 19, p. 131–136, 1995.

WRIGHT, M. H. Nelder, Mead, and the other simplex method. *Documenta Mathematica*, v. 2012, p. 271–276, 2012.

WU, G.; QIU, D.; YU, Y.; PEDRYCZ, W.; MA, M.; LI, H. Superior solution guided particle swarm optimization combined with local search techniques. *Expert Systems with Applications*, v. 41, n. 16, p. 7536–7548, 2014.

YAO, X.; LIU, Y. Fast evolutionary programming. *Evolutionary Programming*, v. 3, p. 451–460, 1996.

YAO, X.; LIU, Y.; LIN, G. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, v. 3, n. 2, p. 82–102, 1999.

ZHANG, J.; SANDERSON, A. C. JADE: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 5, p. 945–958, 2009.