Universidade do Minho
Escola de Ciências

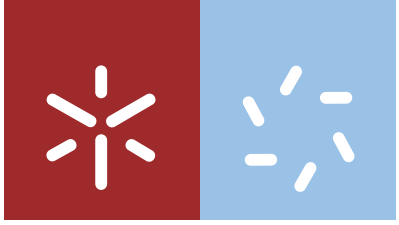
Rogério Brochado Francisco

Um método estocástico para resolver problemas MINLP não suaves

maio de 2017

Um método estocástico para resolver problemas MINLP não suaves
Rogério Brochado Francisco

UMinho | 2017



Universidade do Minho
Escola de Ciências

Rogério Brochado Francisco

Um método estocástico para resolver problemas MINLP não suaves

Tese de Doutoramento em Ciências
Especialidade em Matemática

Trabalho efetuado sob a orientação da
Professora Doutora Maria Fernanda Pires Costa
e da
Professora Doutora Ana Maria Alves Coutinho da Rocha

maio de 2017

DECLARAÇÃO

Nome: Rogério Brochado Francisco

Endereço electrónico: rogerfrancisco.math@gmail.com Telefone: 914346860

Número do Bilhete de Identidade: 11234286–8zz4

Título da tese: Um método estocástico para resolver problemas MINLP não suaves.

Orientadoras:

Professora Doutora Maria Fernanda Pires Costa

Professora Doutora Ana Maria Alves Coutinho da Rocha

Ano de conclusão: 2017

Designação do Ramo de Conhecimento do Doutoramento:

Ciências, especialidade de Matemática

Universidade do Minho, 2 de maio de 2017

Assinatura: Rogério Brochado Francisco

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração da presente dissertação. Confirmo que em todo o trabalho conducente à sua elaboração não recorri à prática de plágio ou a qualquer forma de falsificação de resultados.

Mais declaro que tomei conhecimento integral do Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, 2 de maio de 2017

Nome completo: Rogério Brochado Francisco

Assinatura: Rogério Brochado Francisco

Agradecimentos

Em primeiro lugar, os meus agradecimentos dirigem-se às minhas orientadoras, Professora Doutora Fernanda Costa e Professora Doutora Ana Rocha, pelos enormíssimos contributos prestados na elaboração desta tese e aos conhecimentos que me transmitiram. Sem estes, este trabalho não teria sido possível ser concretizado. Um agradecimento também muito especial pelos incentivos prestados na prossecução dos trabalhos durante uma fase mais difícil da minha vida pessoal. Em segundo lugar, à minha esposa, pelo apoio prestado e pela compreensão pela falta de tempo e de atenção que tanto merecia ao longo de todo este tempo. Por fim, agradeço também à Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto, pelos apoios financeiros disponibilizados para participação nas conferências que me foi possível estar presente.

Resumo

Muitos dos problemas de otimização que surgem com frequência numa vasta gama de aplicações reais, pertencem à área da Otimização Não Linear. Em geral, estes problemas são complexos, pois podem, eventualmente, incluir variáveis contínuas e inteiras, funções não lineares, não contínuas e não diferenciáveis. Nesta área existem duas classes de problemas considerados de difícil resolução aos quais a comunidade científica se tem dedicado: os problemas de programação não linear (NLP) e os problemas de programação não linear inteira-mista (MINLP), não convexos e não suaves. Dada a sua complexidade, em muitos destes problemas não é possível calcular, nem sequer aproximar o cálculo de derivadas, sendo por isso necessário desenvolver outro tipo de abordagem diferente da convencional, para os resolver. Uma das abordagens mais promissoras e utilizadas para resolver este tipo de problemas é o uso de métodos estocásticos. Na literatura existem vários métodos estocásticos, baseados em populações, que têm sido usados com sucesso para resolver estes problemas. Recentemente, surgiu o *Firefly Algorithm* (FA) para resolver problemas de otimização contínuos e com restrições de limites simples, que se tem revelado ser bem sucedido na resolução de problemas práticos e complexos.

Assim, o objetivo central desta tese prende-se com o desenvolvimento de extensões do FA capazes de resolver problemas de NLP e de MINLP, com restrições, não convexos e não suaves. Numa primeira fase, são desenvolvidos algoritmos para a resolução de problemas de NLP com restrições. São propostas duas extensões do FA baseadas em técnicas distintas, para o tratamento das restrições. Uma baseada em esquemas de ordenação dos pontos da população e outra baseada numa nova função de penalidade auto-adaptativa. Esta função

de penalidade auto-adaptativa pode ser implementada em diversos métodos estocásticos de otimização global baseados em populações. O estudo da convergência do algoritmo com a função de penalidade auto-adaptativa, no contexto do FA, é demonstrada em termos probabilísticos.

Numa segunda fase, foram desenvolvidos algoritmos para a resolução de problemas de MINLP. Primeiro desenvolveu-se a extensão do FA para resolver problemas de MINLP com restrições de limites simples e variáveis binárias, baseada em heurísticas. A seguir, foi desenvolvida uma extensão do FA que implementa um algoritmo de penalidade exata, para resolver os problemas de MINLP com restrições de limites simples e variáveis contínuas e inteiras. Neste contexto, foram propostas duas funções de penalidade exata, e o estudo da convergência do algoritmo de penalidade, no contexto do FA, é demonstrada em termos probabilísticos. Por fim, foi desenvolvida uma extensão do FA para resolver problemas de MINLP com restrições genéricas, que combina uma heurística e um esquema de ordenação dos pontos da população, para o tratamento das restrições do problema.

As experiências numéricas realizadas mostraram que as extensões do FA propostas são eficazes, tendo-se obtido soluções de qualidade elevada, quando comparados com outros métodos disponíveis na literatura.

Abstract

Many optimization problems that frequently arise in a wide range of real applications belong to the area of Nonlinear Optimization. In general, these problems are complex, and may eventually involve continuous and integer variables, and nonlinear, non continuous and non differentiable functions. In this area there are two classes of optimization problems considered hard to solve, that have been studied by the scientific community: NonLinear Programming (NLP) problems and Mixed-Integer NonLinear Programming (MINLP) problems, nonconvex and nonsmooth problems. Since in many of these problems it is not possible to calculate or even approximate the derivatives, it is necessary to develop a new kind of approach different from the conventional one. The stochastic methods are the most commonly used to solve these type of problems. In the literature there are several population-based stochastic methods that have been successfully used to solve these problems. Recently, the Firefly Algorithm (FA) has emerged to solve continuous optimization problems with simple bounds that have proven to be successful in solving practical and complex problems.

Thus, the main goal of this thesis is to extend the FA to solve constrained nonconvex and nonsmooth NLP and MINLP problems. In the first phase, algorithms to solve constrained NLP problems are developed. Two extensions of FA using different techniques for handling the constraints are proposed. One is based on ordering schemes of population points and another based on a new self-adaptive penalty function. The self-adaptive penalty function can be implemented in several population-based stochastic methods. The convergence study of the algorithm based on the self-adaptive penalty function, in the FA context, in

probabilistic terms is demonstrated.

In a second phase, algorithms to solve MINLP problems were developed. First, an FA extension was developed to solve MINLP problems with simple bounds and binary variables based on heuristics. Next, an FA extension that implements an exact penalty algorithm to solve the MINLP problems with simple bounds and mixed variables was developed. In this context, two exact penalty functions were proposed and the convergence study of the penalty algorithm, in the context of the FA, in probabilistic terms is proved. Finally, an FA extension to solve constrained MINLP problems, which combines a heuristic and a scheme of ordering the points of the population, for handling the constraints was developed.

Numerical experiments have shown that the proposed FA extensions are effective and produced high quality solutions when compared to other methods available in the literature.

Índice

Lista de Tabelas

Lista de Figuras

Lista de siglas e abreviaturas

NLP	Non Linear Programming
MINLP	Mixed Integer Non Linear Programming
ABC	Artificial Bee Colony
PSO	Particle Swarm Optimization
DE	Differential Evolution
FA	Firefly Algorithm
HS	Harmony Search
BCOP	Bound Constrained Optimization Problem
COP	Constrained Optimization Problem
GA	Genetic Algorithm
SA	Simulated Annealing
CMA-ES	Evolution Strategy with Covariance Matrix Adaptation
OGC	Ordenação global competitiva
RAO	Regras de admissibilidade e otimalidade
EXFA	Extensão do FA usando esquemas de ordenação
EXFA#	Extensão do FA usando as RAO e movimento direto
EXFA1	Extensão do FA usando as RAO
EXFA2	Extensão do FA usando a OGC
FPAA	Função de penalidade auto-adaptativa
HBFA	FA baseado em heurísticas
mEC	Movimento no Espaço Contínuo

mEB . . . . . Movimento no Espaço Binário

pCB . . . . . Probabilidade da Componente Binária

FAbRAO . . . . . FA baseado nas RAO

DUVDE . . . . . Dynamic use of variants diferential evolution

APM . . . . . Adaptive Penalty Method

AMPSO . . . . . Angle Modulated PSO

AMDE . . . . . Angle Modulated DE

binDE . . . . . Algoritmo DE binário

binPSO . . . . . Algoritmo PSO binário

ABHS . . . . . Adaptive Binary Harmony Search

Lista de Algoritmos

Capítulo 1

Introdução

Esta tese integra-se na área da Programação Não Linear (ou Otimização Não Linear). Os problemas de otimização são problemas decorrentes da vida real, e surgem quando se pretende calcular a melhor solução para um dado problema real, a partir da construção de um modelo matemático apropriado para esse problema em estudo. O processo de modelação envolve a identificação das variáveis de decisão, da função objetivo, e das restrições, definidas algebricamente por determinadas funções. Os problemas de otimização surgem nas mais diversas áreas do saber, com particular ênfase nas áreas das Engenharias e Ciências. Em geral, os modelos matemáticos destes problemas são complexos, dado que as funções envolvidas no modelo, função objetivo e funções das restrições, podem ser não lineares, não contínuas e não diferenciáveis, e podem envolver variáveis de decisão de diferentes tipos.

Nesta tese, pretende-se resolver dois tipos de problemas de Programação Não Linear: problemas com restrições e variáveis contínuas; e problemas com restrições e variáveis mistas, isto é, variáveis contínuas e discretas. No processo de modelação, as variáveis discretas que surgem com maior frequência são as variáveis inteiras e binárias. Em ambos os tipos de problemas, não se assume que as diversas funções envolvidas sejam continuamente diferenciáveis e, por isso, são propostos nesta tese métodos de resolução que não envolvem o uso de derivadas analíticas ou numéricas. Além disso, os problemas de otimização abordados

podem ser não convexos, ou seja, a função objetivo pode ser uma função não convexa e a região admissível pode ser um conjunto não convexo. Assim, os problemas podem ter vários minimizantes (ou maximizantes) locais e globais. Em qualquer dos casos, está-se interessado na determinação de um minimizante global dos problemas. Assim, a tese tem como objetivo desenvolver novos algoritmos de otimização eficientes e robustos para resolver cada um destes tipos de problemas. Serão usados os acrónimos NLP e MINLP para designar, respetivamente, Programação Não Linear (do inglês, *Nonlinear Programming*) e Programação Não Linear Inteira Mista (do inglês, *Mixed Integer NonLinear Programming*).

1.1 Motivação

Os problemas de NLP e os problemas de MINLP surgem na modelação de problemas de uma vasta área de aplicações, especialmente na Engenharia (produção industrial, robótica, eletrotecnia, química, mecânica, aeroespacial), nas Ciências de Gestão e Investigação Operacional. Neste contexto, é normal que os problemas envolvam funções não lineares e não convexas, variáveis contínuas ou mistas, podendo umas ser livres e outras com restrições nos seus valores. Adicionalmente, em algumas aplicações não é possível calcular derivadas, ou porque o custo de cálculo é elevado, ou porque as funções envolvidas não são diferenciáveis. Assim, é necessário recorrer a outro tipo de abordagem diferente das tradicionais para a sua resolução.

Uma das abordagens, às quais a comunidade científica se tem dedicado nas últimas décadas, para resolver globalmente problemas de NLP e de MINLP, não convexos e não suaves, são os métodos estocásticos. Em geral, os métodos estocásticos apenas exigem informação relativa à avaliação das funções envolvidas no problema, e apresentam bom desempenho numérico. A sua relevância advém do facto de poderem ser aplicados a uma vasta gama de aplicações práticas nas mais diversas áreas do conhecimento. A título de exemplo, apresentam-se algumas aplicações resolvidas por métodos estocásticos, retirados da literatura. Yao, Yang, Hu, e Yu (2010) utilizaram o algoritmo *Ant Bee Colony* (ABC) para localizar as rotas de um metropolitano que visam maximizar a população coberta

por essas mesmas rotas. Nebti e Boukerram (2010) também usaram o algoritmo ABC no problema de reconhecimento de dígitos manuscritos. O algoritmo *Particle Swarm Otimization* (PSO) tem sido usado na resolução de várias aplicações, nomeadamente na área da medicina, tais como o diagnóstico da doença de Parkinson, a otimização da biomecânica do movimento humano, classificação de neoplasias (Selvan et al., 2006), planeamento de redes de transmissão e reconfigurações e expansão de redes (Kannan, Slochanal, & Padhy, 2005) e também em problemas de despacho económico (do inglês, *Economic Dispatch*) (Zhao, Guo, & Cao, 2005). O algoritmo *Differential Evolution* (DE) foi usado em técnicas de desenho aerodinâmico (Rogalsky & Derksen, 2000), projeções de filtros digitais para aplicações na indústria de papel (Tirronen, Neri, Kärkkäinen, Majava, & Rossi, 2008) e aplicações na otimização da capacidade de redes rodoviárias (Koh, 2009). Nomeiam-se também outros exemplos retirados da literatura, sobre a resolução de problemas de MINLP com restrições de limites simples por métodos estocásticos. Em particular, salienta-se o algoritmo PSO (Pampara, Engelbrecht, & Franken, 2006; Kennedy & Eberhart, 1997; Mirjalili & Lewis, 2013), o algoritmo DE (Engelbrecht & Pampara, 2007), o algoritmo *Harmony Search* (HS) (Padberg, 2012; L. Wang et al., 2013), o algoritmo *Artificial fish swarm* (Azad, Kalam, Rocha, & Fernandes, 2013) e o algoritmo ABC (Pampara & Engelbrecht, 2011; Kashan, Nahavandi, & Kashan, 2012; T. Liu, Zhang, & Zhang, 2013).

Recentemente, surgiu um novo método estocástico, conhecido na literatura por *Firefly Algorithm* (FA), que é um algoritmo bio-inspirado, desenvolvido por X.-S. Yang (2008) para resolver problemas de NLP com restrições de limites simples e variáveis contínuas. Este algoritmo tornou-se muito popular na última década, e revelou-se um sucesso na resolução de problemas práticos e complexos. Desde então, diversas variantes do FA têm sido propostas (Farahani, Abshouri, Nasiri, & Meybodi, 2011; Fister Jr, Yang, Fister, & Brest, 2012; Vahedi Nouri, Fattahi, & Ramezanian, 2013; Lin, Zhong, & Zhang, 2013; X.-S. Yang, 2013; Francisco, Costa, & Rocha, 2014; B. Wang, Li, Jiang, & Liao, 2016), incluindo abordagens híbridas (Farahani, Abshouri, Nasiri, & Meybodi, 2012; Abdullah, Deris, Mohamad, & Hashim, 2012; Hassanzadeh & Meybodi, 2012; Guo, Wang, Wang, & Wang, 2013). Diversas aplicações práticas também têm sido recentemente resolvidas

usando o FA (Hönig, 2010; X.-S. Yang, Hosseini, & Gandomi, 2012; Sayadi, Hafezalkotob, & Naini, 2013; Bacanin & Tuba, 2014).

Nesta tese, tendo como ponto de partida a eficácia demonstrada pelo FA na resolução de problemas complexos, no âmbito de aplicações reais, um dos objetivos prende-se com a extensão do FA para resolver problemas de NLP com restrições genéricas, não convexos e não suaves. Dado que muitas destas aplicações envolvem variáveis contínuas e inteiras, surgem com frequência problemas de MINLP com restrições genéricas, não convexos e não suaves. Interessa realçar que estes problemas de MINLP, não convexos e não suaves, são de difícil resolução. Assim, pretende-se também desenvolver a extensão do FA para resolver problemas de MINLP com restrições de limites simples, não convexos e não suaves e, por fim, fazer a sua extensão para resolver problemas de MINLP com restrições genéricas.

1.2 Descrição do Problema

Um problema de otimização caracteriza-se por ter uma função a otimizar, a *função objetivo*, um conjunto de variáveis ou parâmetros independentes, as *variáveis de decisão*, que afetam o valor da função objetivo, e que geralmente estão sujeitas a algumas condições, as *restrições* do problema. As condições às quais as variáveis de decisão têm de obedecer definem os *pontos admissíveis* do problema. Ao conjunto de todos os pontos admissíveis, que será representado ao longo da tese por Ω, chama-se *região admissível* ou *conjunto admissível* do problema. O objetivo na resolução de um problema de otimização é determinar os valores das variáveis de decisão que satisfazem as restrições do problema e que minimizam (ou maximizam) a função objetivo.

Como os problemas de otimização surgem nas mais diversas áreas do conhecimento, estes podem possuir características muito distintas e, em geral, envolvem restrições nas variáveis de decisão. Contudo, é possível formular os problemas de otimização de uma forma geral. A representação mais simples de um problema de otimização com restrições é a do problema com restrições de limites simples nas variáveis (do inglês, *Bound Constrained Optimization Problem - BCOP*) que pode ser escrito matematicamente da seguinte forma:

$$\begin{aligned} &\text{minimizar} && f(x) \\ &\text{sujeito a} && x \in X \subset \mathbb{R}^n, \end{aligned} \tag{1.2.1}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é a função objetivo e X é um hiper-retângulo de $\mathbb{R}^n$ definido por

$$X = \{x \in \mathbb{R}^n : -\infty < lb \leq x \leq ub < \infty\}. \tag{1.2.2}$$

O vetor x é o vetor das variáveis de decisão, lb e ub são os vetores dos limites inferiores e superiores, respetivamente, das variáveis.

Ao longo desta tese serão apenas abordados problemas de minimização, uma vez que

$$f^* \equiv \min_{x \in X \subset \mathbb{R}^n} f(x) = -\max_{x \in X \subset \mathbb{R}^n}(-f(x)) \quad \text{onde} \quad x^* \equiv \arg\min_{x \in X \subset \mathbb{R}^n} f(x) = \arg\max_{x \in X \subset \mathbb{R}^n}(-f(x)),$$

ou seja, um ponto onde f atinge o seu mínimo é o mesmo onde $-f$ atinge o seu máximo. A região admissível do problema (1.2.1) é $\Omega = X$. Note-se que caso $X = \mathbb{R}^n$, está-se perante um problema de otimização sem restrições.

A representação mais geral de um problema de otimização com restrições (do inglês, *Constrained Optimization Problem - COP*), envolve restrições de desigualdade, igualdade e restrições de limites simples, e pode ser escrito da seguinte forma:

$$\begin{aligned} \text{minimizar} \quad & f(x) \\ \text{sujeito a} \quad & g_i(x) \leq 0, \ i = 1, \ldots, m \\ & h_j(x) = 0, \ j = 1, \ldots, p \\ & x \in X \subset \mathbb{R}^n, \end{aligned} \tag{1.2.3}$$

onde a função objetivo $f : \mathbb{R}^n \to \mathbb{R}$ e as restrições de desigualdade e igualdade $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$ são provavelmente não lineares, para $i = 1, \ldots, m$ e $j = 1, \ldots, p$. O conjunto X é o hiper-retângulo definido em (1.2.2).

A região admissível Ω do problema (1.2.3) é dada por:

$$\Omega = \{x \in X : g_i(x) \leq 0 \ , \ h_j(x) = 0; \quad i = 1, \ldots, m, \quad j = 1, \ldots, p\}. \tag{1.2.4}$$

Sem perda de generalidade, ao longo desta tese considerar-se-á o problema de otimização com restrições genéricas, escrito da seguinte forma:

$$\begin{aligned} \text{minimizar} \quad & f(x) \\ \text{sujeito a} \quad & g_i(x) \leq 0, \;\; i = 1, \ldots, m \\ & x \in X \subset \mathbb{R}^n. \end{aligned} \tag{1.2.5}$$

Note-se que, os problemas com restrições de igualdade $h_j(x) = 0, \; j = 1, \ldots, p$, podem ser reformulados na forma (1.2.5) relaxando as restrições de igualdade para $|h_j(x)| - \epsilon \leq 0$, onde ϵ é uma tolerância positiva suficientemente pequena.

As características das funções envolvidas no problema de otimização, tais como a linearidade, convexidade, continuidade e diferenciabilidade, conduzem a diferentes tipos de classificação dos problemas de otimização. Quando todas as funções do problema são lineares, o problema diz-se de Programação Linear. Caso contrário, se existir pelo menos uma função que seja não linear, então o problema diz-se de NLP. Antes de introduzir outras classificações dos problemas de otimização, é necessário dar as seguintes definições.

Definição 1.2.1. *(Nocedal & Wright, 2006)*

Um conjunto $\Omega \subseteq \mathbb{R}^n$ é convexo se, $\forall \; x, y \in \Omega$,

$$\lambda x + (1 - \lambda) y \; \in \Omega, \quad \forall \lambda \in [0, 1]. \tag{1.2.6}$$

Em termos gráficos, um conjunto é convexo se, para quaisquer pares de pontos do conjunto, o segmento de reta que os une está contido no conjunto.

Definição 1.2.2. *(Nocedal & Wright, 2006)*

Uma função f definida num conjunto convexo Ω diz-se convexa se, $\forall \; x, y \in \Omega$,

$$f(\lambda x + (1 - \lambda) y) \leq \lambda f(x) + (1 - \lambda) f(y) \;\; , \forall \lambda \in [0, 1]. \tag{1.2.7}$$

Em termos gráficos, uma função f é convexa se o segmento de reta que une dois pontos quaisquer do seu gráfico $(x, f(x))$ e $(y, f(y))$, com $x, y \in \Omega$, se encontra acima ou se sobrepõe ao gráfico de f. Se a desigualdade em (1.2.7) é estrita, f diz-se *estritamente*

convexa. Desta definição pode também concluir-se que f é uma função *côncava* num domínio convexo Ω se $-f$ é convexa em Ω.

Atendendo às Definições 1.2.1 e 1.2.2, um problema de otimização de NLP é convexo quando a função objetivo é convexa e a região admissível é um conjunto convexo. Caso contrário, o problema de otimização é não convexo, ou seja, a função objetivo é uma função não convexa ou a região admissível é um conjunto não convexo.

Dependendo se o problema de NLP é convexo ou não convexo, o problema pode ter apenas um único minimizante ou vários minimizantes. Assim, irá definir-se o conceito de minimizante local e minimizante global. Começa-se por introduzir a definição de vizinhança.

Definição 1.2.3. *(Nocedal & Wright, 2006) Seja $x^* \in \Omega$. Chama-se vizinhança B de x^* de raio $\varepsilon > 0$, $B(x^*, \varepsilon)$, ao conjunto de pontos definido por*

$$B = \{x \in \mathbb{R}^n : ||x - x^*|| \leq \varepsilon\}.$$

Definição 1.2.4. *Seja $\Omega \subseteq \mathbb{R}^n$, com $\Omega \neq \emptyset$, a região admissível do problema de otimização. Um ponto $x^* \in \Omega$ é um:*

- *minimizante local forte de f em Ω se existir uma vizinhança $B(x^*, \varepsilon)$ tal que*

$$f(x^*) < f(x), \ \ \forall x \in B(x^*, \varepsilon) \cap \Omega;$$

- *minimizante local fraco de f em Ω se existir uma vizinhança $B(x^*, \varepsilon)$ tal que*

$$f(x^*) \leq f(x), \ \ \forall x \in B(x^*, \varepsilon) \cap \Omega;$$

- *minimizante global fraco de f em Ω se*

$$f(x^*) \leq f(x), \ \ \forall x \in \Omega;$$

- *minimizante global forte de f em Ω se*

$$f(x^*) < f(x), \ \ \forall x \in \Omega.$$

Em qualquer um dos casos, denotar-se-à por $f^* = f(x^*)$ o mínimo local ou global, respetivamente. Note-se que quando um problema de otimização é convexo, um minimizante local é também um minimizante global.

De uma forma geral, é fácil resolver problemas de LP e problemas de NLP convexos, enquanto que os problemas de NLP não convexos são considerados de difícil resolução.

Os problemas de otimização podem também ser classificados quanto à natureza das suas variáveis de decisão. Quando existem apenas variáveis inteiras, o problema diz-se de Programação Inteira. Se o problema envolver simultaneamente variáveis inteiras e contínuas, o problema é designado de problema de Programação Inteira Mista. Caso todas as funções do problema sejam lineares, o problema é dito de Programação Linear Inteira Mista. Se existir pelo menos uma função não linear, então é um problema de MINLP.

Nesta tese, considerar-se-á o problema de MINLP com restrições de limites simples nas variáveis escrito na seguinte forma:

$$\begin{aligned} \text{minimizar} \quad & f(x) \\ \text{sujeito a} \quad & x \in X \subset \mathbb{R}^n \\ & x_i \in \mathbb{R} \text{ para } i \in I_c \subseteq I \equiv \{1, \ldots, n\} \\ & x_j \in \mathbb{Z} \text{ para } j \in I_d \subseteq I \\ & I_c \cap I_d = \emptyset \text{ e } I_c \cup I_d = I, \end{aligned} \tag{1.2.8}$$

onde a função objetivo $f : \mathbb{R}^n \to \mathbb{R}$ é não linear e o conjunto X é dado por (1.2.2). Para $x_j \in \mathbb{Z}$, com $j \in I_d$, a região admissível do problema (1.2.8) é definida por:

$$W = \{x \in X \subset \mathbb{R}^n : x_j \in \mathbb{Z} \text{ para } j \in I_d \subseteq I\}. \tag{1.2.9}$$

Denotar-se-á por $n_c = |I_c|$ o número de variáveis contínuas e $n_d = |I_d|$ o número de variáveis inteiras.

No caso particular em que todas as variáveis inteiras são binárias, o problema (1.2.8) pode ser escrito na forma simplificada:

$$\begin{aligned} \text{minimizar} \quad & f(x) \\ \text{sujeito a} \quad & x \in X \subset \mathbb{R}^n \\ & x_j \in \{0, 1\} \text{ para } j \in I \equiv \{1, \ldots, n\}. \end{aligned} \tag{1.2.10}$$

Para simplificar a escrita, usar-se-á também a seguinte notação: x_z representará o vetor das variáveis inteiras, $[x_j]_{j\in I_d}$, e x_c o vetor das variáveis contínuas, $[x_i]_{i\in I_c}$. Nesta tese considerar-se-á o problema de MINLP com restrições genéricas escrito na forma:

$$\begin{aligned} \text{minimizar} \quad & f(x_c, x_z) \\ \text{sujeito a} \quad & x \in X \subset \mathbb{R}^n \\ & g_i(x_c, x_z) \leq 0,\ i = 1, \ldots, m \\ & h_j(x_c, x_z) = 0,\ j = 1, \ldots, p \\ & x_c \in \mathbb{R}^{n_c},\ x_z \in \mathbb{Z}^{n_d}. \end{aligned} \tag{1.2.11}$$

A sua região admissível Ω é definida por

$$\Omega = \{x \in W : g_i(x_c, x_z) \leq 0, i = 1, \ldots, m,\ \text{e}\ h_j(x_c, x_z) = 0,\ j = 1, \ldots, p\}, \tag{1.2.12}$$

onde W é o conjunto definido em (1.2.9).

Dado que um problema de MINLP pode envolver variáveis reais e variáveis inteiras simultaneamente, para se definir o conceito de mínimo local, é necessário redefinir o conceito de vizinhança.

As definições que se seguem encontram-se em (Liuzzi, Lucidi, & Rinaldi, 2015).

Definição 1.2.5. *Para qualquer ponto* $(\overline{x}_c, \overline{x}_z) \in \mathbb{R}^n$, e $\varepsilon > 0$ tem-se que:

$$B_c((\overline{x}_c, \overline{x}_z), \varepsilon) = \{(x_c, x_z) \in \mathbb{R}^n : x_z = \bar{x}_z,\ ||x_c - \bar{x}_c|| \leq \varepsilon\}$$

e

$$B_z(\overline{x}_c, \overline{x}_z) = \{(x_c, x_z) \in \mathbb{R}^{n_c} \times \mathbb{Z}^{n_d} : x_c = \bar{x}_c,\ ||x_z - \bar{x}_z|| \leq 1\}.$$

O conceito de vizinhança apresentado permite definir o mínimo e minimizante local de um problema de MINLP.

Definição 1.2.6. *Um ponto* $(x_c^*, x_z^*) \in \Omega$ *é um minimizante local do problema (1.2.11) se, para algum* $\varepsilon > 0$,

$$f(x_c^*, x_z^*) \leq f(x_c, x_z),\ \ \forall (x_c, x_z) \in B_c((x_c^*, x_z^*), \varepsilon) \cap \Omega$$

e

$$f(x_c^*, x_z^*) \leq f(x_c, x_z), \ \ \forall (x_c, x_z) \in B_z(x_c^*, x_z^*) \cap \Omega,$$

sendo $f^* = f(x_c^*, x_z^*)$ o mínimo local. Na literatura, os problemas de MINLP são também classificados quanto à sua convexidade. Diz-se que o problema de MINLP é convexo se todas as funções envolvidas no problema são convexas; caso contrário, diz-se não convexo. Na verdade, a existência de variáveis inteiras torna-o sempre num problema não convexo. No entanto, este é classificado de convexo quando a relaxação das funções que definem as suas restrições genéricas são convexas.

De uma forma geral, os problemas de MINLP convexos, ou não convexos, são de difícil resolução, ao passo que os problema de MINLP convexos têm uma resolução mais fácil do que a dos problemas de MINLP não convexos, dado que satisfazem o teorema seguinte:

Teorema 1.2.1. *(Leyffer, 1993) Seja (x_c^*, x_z^*) um minimizante local de um problema MINLP convexo. Então, (x_c^*, x_z^*) é também um minimizante global desse problema.*

Contrariamente a um problema de NLP convexo, onde é garantida a existência e unicidade do minimizante global, num problema de MINLP convexo apenas está garantida a sua existência (Teorema 1.2.1). Quando o problema de MINLP é não convexo, as dificuldades são maiores, dado que o problema poderá ter vários mínimos locais e globais distintos e, não é possível aplicar o Teorema 1.2.1. Se além disto, alguma das funções envolvidas no problema é não suave, está-se perante um problema de MINLP não convexo e não suave, o que dificulta mais a sua resolução.

1.3 Contribuições da Tese

O FA é um dos mais recentes métodos estocásticos baseado em populações de pontos para resolver problemas de otimização. Desde a sua criação, este método tem sido objeto de estudo para resolver problemas nas mais diversas áreas de conhecimento. Porém, a sua versão original permite apenas resolver BCOP com variáveis contínuas.

Os objetivos desta tese centram-se no desenvolvimento de extensões do FA para resolver problemas de NLP e de MINLP, não convexos e não suaves, com restrições de limites simples e restrições genéricas.

Dada a natureza distinta destas duas classes de problemas, esta tese divide-se em duas fases de estudo. Numa primeira fase, são desenvolvidos algoritmos para a resolução de problemas de NLP com restrições genéricas, que se distinguem pela técnica usada para o tratamento das restrições. Neste âmbito, propõem-se duas extensões do FA: o algoritmo EXFA e o algoritmo FPAA. O algoritmo EXFA implementa, para o tratamento das restrições, dois esquemas de ordenação dos pontos da população. Um tem por base a ordenação global competitiva e o outro as regras de admissibilidade e otimalidade. O algoritmo FPAA implementa uma função de penalidade auto-adaptativa. Demonstra-se a equivalência entre o COP e o BCOP reformulado com a função de penalidade auto-adaptativa proposta. É de referir que é analisado o desempenho da função auto-adaptativa quando usada no contexto do FA e de outros métodos estocásticos de otimização global baseados em populações. Por fim, é efetuada a análise de convergência do algoritmo FPAA em termos probabilísticos, a qual tem em consideração a estrutura de um FA dinâmico e as propriedades da função de penalidade auto-adaptativa proposta. Interessa realçar que, ambos os métodos implementam técnicas para o tratamento das restrições que não necessitam da definição de qualquer parâmetro de penalidade.

Numa segunda fase, o estudo é direcionado para a resolução de problemas de MINLP com restrições de limites simples, e posteriormente, para resolver problemas de MINLP com restrições genéricas. Neste contexto, propõe-se a extensão do FA baseado em heurísticas, designado por HBFA, para a resolução problemas de MINLP com restrições de limites simples e variáveis binárias. As regras heurísticas são usadas na discretização das variáveis binárias e são analisados três métodos diferentes de implementação das HBFA no FA dinâmico.

A seguir, propõe-se uma extensão do FA capaz de resolver problemas de MINLP com restrições de limites simples e variáveis contínuas e inteiras. Para tal, optou-se por uma técnica de penalidade exata, que se baseia numa formulação contínua de penalidade exata

do problema de MINLP, que surge da relaxação das restrições de integralidade. Propõem-se dois termos de penalidade, um baseado na função tangente hiperbólica e outro na função inversa do seno hiperbólico. Demonstra-se a equivalência entre o problema de MINLP e a reformulação contínua de penalidade exata do problema. Por fim, é efetuada a análise de convergência estocástica do algoritmo de penalidade, a qual tem em consideração a estrutura de um FA adaptativo e as propriedades das funções de penalidade exata propostas.

Finalmente, é apresentada uma extensão do FA capaz de resolver problemas de MINLP com restrições genéricas e de integralidade, o algoritmo FAbRAO. Dada a existência de variáveis inteiras e contínuas, propõe-se uma nova forma para gerar a população inicial e propõem-se duas equações que descrevem o seu movimento ao longo do processo iterativo, para lidar com os dois tipos de variáveis de decisão do problema. Neste contexto, optou-se pelas regras de admissibilidade e otimalidade para o tratamento das restrições genéricas.

Para concluir, é de salientar que para validar as novas metodologias e algoritmos propostos, é efetuado um conjunto alargado de experiências computacionais, recorrendo a problemas teste disponíveis na literatura científica. Mais ainda, são também efetuados estudos comparativos com outros métodos relatados na literatura, de modo a comparar os desempenhos e a qualidade das soluções obtidas.

No contexto do trabalho desenvolvido nesta tese, publicaram-se os seguintes artigos:

(i) Francisco, R.B., Costa, M.F.P., e Rocha, A.M.A.C. (2014). Experiments with Firefly Algorithm. B. Murgante et al. (Eds.): ICCSA 2014, Part II, LNCS 8580, pp. 227–236. Springer. DOI: 10.1007/978-3-319-09129-7_17

(ii) Costa, M.F.P., Rocha, A.M.A.C., Francisco, R.B., e Fernandes, E.M.G.P. (2014). Heuristic-Based Firefly Algorithm for Bound Constrained Nonlinear Binary Optimization. Advances in Operations Research, Volume 2014, Article ID 215182, 12 pages. DOI: 10.1155/2014/215182

(iii) Francisco, R.B., Costa, M.F.P., e Rocha, A.M.A.C. (2015). A Firefly Dynamic Penalty Approach for Solving Engineering Design Problems. International Conference

of Numerical Analysis and Applied Mathematics 2014, Theodore E. Simos (Eds.), AIP Conference Proceedings, Vol. 1648, 140010. DOI: 10.1063/1.4912430

(iv) Francisco, R.B., Costa, M.F.P., Rocha, A.M.A.C. e Fernandes, E.M.G.P. (2016). Comparison of Penalty Functions on a Penalty Approach to Mixed–Integer Optimization, International Conference of Numerical Analysis and Applied Mathematics 2015, Theodore E. Simos (Eds.), AIP Conference Proceedings, Vol. 1738, 300008. DOI: 10.1063/1.4952100

(v) Francisco, R.B., Costa, M.F.P., e Rocha, A.M.A. (2016). Extensions of Firefly Algorithm for Nonsmooth Nonconvex Constrained Optimization Problems. O. Gervasi et al. (Eds.): ICCSA 2016, Part I, LNCS 9786, pp. 402–417.
DOI: 10.1007/978-3-319-42085-1_31

(vi) Costa, M.F.P., Francisco, R.B., Rocha, A.M.A.C., e Fernandes, E.M.G.P. (2016). Theoretical and Practical Convergence of a Self-adaptive Penalty Algorithm for Constrained Global Optimization. Journal of Optimization Theory and Applications, publicado online em dezembro de 2016. DOI: 10.1007/s10957-016-1042-7

(vii) Costa, M.F.P., Rocha, A.M.A.C., Francisco, R.B., e Fernandes, E.M.G.P. (2016). Firefly penalty-based algorithm for bound constrained mixed-integer nonlinear programming. Optimization, Vol. 65, Issue 5, 1085–1104.
DOI: 10.1080/02331934.2015.1135920

(viii) Costa, M.F.P., Rocha, A.M.A.C., Francisco, R.B., e Fernandes, E.M.G.P. (2017). Extension of the firefly algorithm and preference rules for solving MINLP problems. International Conference of Numerical Analysis and Applied Mathematics 2016, Theodore E. Simos (Eds.), AIP Conference Proceedings. (no prelo)

1.4 Estrutura da Tese

Esta tese está estruturada da seguinte forma. No Capítulo 2, descreve-se com pormenor o FA originalmente proposto por X.-S. Yang (2008), e são apresentadas algumas variantes do algoritmo propostas por diversos autores para resolver BCOP. Neste capítulo são também descritos de uma forma sumária outros métodos estocásticos baseados em populações, bastante populares na literatura científica para resolver globalmente BCOP.

No Capítulo 3, faz-se uma revisão de literatura sobre as técnicas para tratar as restrições em problemas de NLP e em MINLP. Assim, são apresentadas várias técnicas para o tratamento de restrições genéricas em problemas de NLP, e para o tratamento de restrições de integralidade em problemas de MINLP.

No Capítulo 4, apresentam-se duas extensões do FA para resolver problemas de NLP, não convexos e não suaves com restrições genéricas, baseadas em novas técnicas para o tratamento de restrições. As extensões apresentadas são o algoritmo EXFA e o algoritmo FPAA.

No Capítulo 5, apresentam-se três extensões do algoritmo FA para resolver problemas de MINLP, não convexos e não suaves. Primeiro apresentam-se as extensões do FA para resolver problemas de MINLP com restrições de limites simples e variáveis binárias, que recorrem à técnica da discretização, designadas pelas HBFA. A seguir, apresenta-se a extensão do FA para resolver problemas de MINLP com restrições de limites simples e variáveis contínuas e inteiras, recorrendo ao método de penalidade exata para o tratamento das restrições de integralidade. Por fim, mostra-se uma extensão do FA para resolver problemas de MINLP com restrições genéricas e variáveis contínuas e inteiras, o algoritmo FAbRAO.

No Capítulo 6, apresentam-se os testes computacionais realizados com os algoritmos propostos, usando um conjunto de problemas retirados da literatura.

Para finalizar, no Capítulo 7 apresentam-se as conclusões deste trabalho e propostas de investigação para trabalho futuro.

Capítulo 2

Métodos estocásticos de otimização global

Nas últimas décadas foram criados e desenvolvidos vários métodos estocásticos para resolver um alargado conjunto de problemas de otimização. Os algoritmos de inteligência de enxame, por vezes também designados de bio-inspirados, são métodos estocásticos que tentam fazer uma analogia com sistemas biológicos existentes na natureza, para resolver diversos tipos de problemas de otimização complexos e em tempo computacional aceitável. Estes métodos inserem-se na classe de métodos de otimização do tipo probabilístico, e baseiam-se em princípios e mecanismos evolutivos encontrados na natureza, como, por exemplo, a auto-organização, a procura de alimento, a reprodução e o comportamento adaptativo e/ou instintivo de sistemas organizados de espécies animais, tais como insetos, colónias de formigas, entre outros. A grande quantidade de técnicas existentes na literatura é um reflexo da sua ausência de generalidade, acontecendo com frequência que um determinado método produz bons resultados para uma classe de problemas e falha para outro tipo de classe. Assim, persiste a dificuldade em encontrar métodos estocásticos genéricos que produzam bons resultados e que possam ser usados para resolver qualquer tipo de problema de otimização. Em geral, estes algoritmos têm como característica comum basearem-se em populações de indivíduos, onde cada indivíduo representa um ponto do

espaço de procura, que são potenciais soluções para um dado problema de otimização. Normalmente, estes métodos aplicam-se quando os problemas são de difícil resolução e não existem métodos exatos capazes de os resolver ou, quando existem, requerem um elevado esforço computacional. O surgimento de problemas de otimização complexos, e para os quais os métodos estocásticos disponíveis na literatura não foram capazes de os resolver, originou o desenvolvimento de novas metaheurísticas mais versáteis, que pudessem resolver um conjunto mais alargado de problemas. É de referir que as metaheurísticas utilizam heurísticas subordinadas, em que usam uma conjugação entre procura aleatória de soluções em regiões ainda não exploradas (procura global), com procura numa vizinhança de uma solução identificada como promissora (procura local), de forma a encontrarem a solução ótima de um problema. O desempenho de qualquer metaheurística depende da sua capacidade de diversificação (ou exploração) e de intensificação (refinamento), na procura de soluções ótimas. A capacidade de diversificação tem como objetivo fazer uma boa exploração do espaço de procura para encontrar uma grande diversidade de soluções promissoras. Por sua vez, a capacidade de intensificação visa explorar localmente uma região identificada como promissora de modo a encontrar o ótimo global de um problema com uma maior precisão.

De uma forma geral, um método estocástico baseado em populações, começa por gerar uma população inicial, normalmente aleatória, de possíveis soluções do problema, seguindo-se a avaliação da população por uma função, que avalia cada elemento da população e lhe atribui um valor de aptidão. A pesquisa de novas soluções é orientada pela qualidade do valor de aptidão atribuído. Depois, são originadas populações sucessivas, sendo a população seguinte obtida de uma antecedente. Os novos elementos vão substituir os da população inicial fazendo com que os menos "aptos" sejam removidos, permanecendo os mais "aptos". O processo repete-se até ser atingido um determinado critério de paragem.

Neste capítulo, descrevem-se os principais passos do FA (Secção 2.1). Na Secção 2.2 são apresentadas algumas variantes do FA propostas por diversos autores, com o objetivo de melhorar o seu desempenho e velocidade de convergência. Finalmente, na Secção 2.3, são descritos de uma forma sumária outros métodos estocásticos existentes na literatura

utilizados para resolver problemas de otimização complexos.

2.1 O Algoritmo FA

O *Firefly Algorithm*, criado por X.-S. Yang (2008), é um algoritmo estocástico bio-inspirado capaz de encontrar a solução global de um BCOP. Este algoritmo é baseado no comportamento e características próprias dos pirilampos, mais concretamente o brilho que emitem e a forma como estes se atraem. Esta característica muito particular deve-se a um conjunto de reações químicas complexas, que é fundamental para a sobrevivência da sua espécie, pois tem uma função de atração pelos seus pares e uma de rejeição pelos seus predadores. À semelhança da grande maioria dos métodos de otimização estocásticos, o FA é um algoritmo baseado em populações. Todos os candidatos da população inicial são gerados no espaço de procura com o objetivo de serem direcionados para a melhor localização (onde o máximo ou mínimo da função objetivo é atingido). Este método não requer informação sobre a derivada da função a otimizar, e o seu desempenho não depende das propriedades da função objetivo do problema. Apesar do FA ter sido originalmente concebido para resolver BCOP contínuos, este tem sido objeto de muitas modificações por parte de muitos autores por forma a resolver diversos tipos de problemas. Em particular, foram desenvolvidas diversas variantes para resolver problemas práticos e específicos de certas aplicações com uma maior eficiência que outros métodos estocásticos. Segundo Fister, Yang, e Fister (2014), neste momento existem mais de 20 variantes do FA. A título de curiosidade, os autores referem que uma simples busca com a entrada FA no motor de busca Scirus efetuada em julho de 2013 devolveu 158 artigos relacionados, o que demonstra a popularidade deste algoritmo. Apesar da impossibilidade de uma revisão completa sobre os mais recentes avanços do FA, neste artigo encontra-se uma sistematização e classificação de diversas variantes e extensões do FA, bem como da sua aplicação aos diversos tipos de problemas. A título de exemplo, o FA tem sido utilizado em problemas de despacho económico (Horng & Liou, 2011; X.-S. Yang et al., 2012), problemas de otimização com variáveis mistas (A. H. Gandomi, Yang, & Alavi, 2011; Costa, Rocha, Francisco, &

Fernandes, 2014) e problemas de otimização multiobjetivo (X.-S. Yang, 2009, 2013; Costa, Rocha, & Fernandes, 2015). Uma revisão da evolução do algoritmo FA está disponível em (Fister, Yang, & Brest, 2013; X.-S. Yang & He, 2013).

De acordo com (X.-S. Yang, 2009; X. Yang, 2010; Tilahun & Ong, 2012), as três ideias básicas em que se baseia o FA são:

1. Todos os elementos de uma população podem ser atraídos entre si (não há distinção de género);

2. A atração que se estabelece entre dois pirilampos é proporcional ao seu brilho e diminui com o aumento da distância entre eles. O menos brilhante é atraído pelo mais brilhante;

3. A intensidade do brilho emitido por um pirilampo pode ser tomado como o valor da função objetivo.

Existem dois aspetos fundamentais neste algoritmo: a variação da intensidade da luz que os pirilampos emitem e a forma como eles se atraem entre si. A comunicação entre os pirilampos baseia-se na sua capacidade em emitir intensidades de luz variáveis. Além disso, quanto maior for a distância entre dois pirilampos, menor é a intensidade de luz percecionada por ambos. Assim, no seu caso mais simples, a intensidade de luz percecionada por um pirilampo é inversamente proporcional ao quadrado da distância do pirilampo emissor de luz, ou seja, varia de acordo com a equação

$$I(r) = \frac{I_s}{r^2},$$

onde I_s é a intensidade de luz emitida pela fonte e r a distância entre os pirilampos. Dado que o meio ambiente pode influenciar a intensidade de luz recebida, e para evitar uma singularidade em $r = 0$, foi introduzido um coeficiente de absorção de luz γ e, assim, a intensidade de luz passa a ser definida por

$$I(r) = I_0 e^{-\gamma r^2},$$

onde I_0 é a intensidade de luz emitida pela fonte.

A atratividade entre os pirilampos é proporcional à intensidade de luz percecionada por pirilampos adjacentes, pelo que a função que define a atratividade entre dois pirilampos i e j é dada por

$$\beta_{i,j} = \beta_0 e^{-\gamma r_{i,j}^2}, \tag{2.1.1}$$

onde β_0 é a atratividade quando $r = 0$, $r_{i,j} = \|x_i - x_j\| = \sqrt{\sum_{s=1}^{n}(x_{i,s} - x_{j,s})^2}$, onde $\|.\|$ é a norma Euclideana, $x_{i,s}$ e $x_{j,s}$ são os valores das coordenadas s dos vetores que definem as posições dos pirilampos i e j no espaço de procura, e n é a dimensão do problema. A atratividade β depende do valor de γ, que é essencial para a velocidade de convergência do algoritmo e para a forma como este se comporta. Em teoria, γ pode tomar qualquer valor do intervalo $[0, +\infty[$.

Na sua formulação original, a atração entre dois pirilampos é determinada pelo brilho que produzem, que, na prática, e no caso mais simples, pode ser considerado como o valor da função objetivo.

De seguida, apresentam-se as principais etapas do FA na resolução de BCOP (X.-S. Yang, 2008).

Passo 1 (Geração da população inicial): No início, o FA gera uma população aleatória de N pontos em X, coordenada a coordenada s, da seguinte forma:

$$x_{i,s} = lb_s + \epsilon_{i,s}(ub_s - lb_s),\ s = 1, \ldots, n,\ \ i = 1, \ldots, N, \tag{2.1.2}$$

onde $\epsilon_{i,s} \sim U[0, 1]$, n é a dimensão do problema a resolver e lb e ub são os vetores de limites simples inferiores e superiores das variáveis de decisão, respetivamente. Depois de gerada a população inicial, são calculados os valores da função objetivo de todos os pontos x_i desta população inicial e são ordenados por ordem crescente dos valores da função objetivo.

Passo 2 (Movimento da população): Depois, em cada iteração k, o algoritmo compara cada ponto x_i com todos os outros pontos x_j, para $i, j = 1, \ldots, N$. Se o valor da função objetivo no ponto x_i for maior que o valor da função objetivo no ponto x_j (que em

teoria significa que o pirilampo j é mais brilhante que i, num problema de minimização), x_i é movido em direção ao ponto x_j da seguinte forma:

$$x_i^{(k+1)} = x_i^{(k)} + \beta(x_j^{(k)} - x_i^{(k)}) + \alpha\epsilon_i S, \tag{2.1.3}$$

onde $\epsilon_i \sim U[-1,1]$ é um vetor aleatório de dimensão n, α é um parâmetro que controla a aleatoriedade e é definido pelo utilizador, usualmente pertencente a $[0,1]$, e S (designado por parâmetro de escalonamento do problema) é um vetor que depende do conjunto X, e é definido por

$$S = |lb - ub|. \tag{2.1.4}$$

O valor da atratividade β é calculado por (2.1.1). Por fim, após o movimento dos pontos, o algoritmo projeta, se necessário, cada um dos pontos da população no conjunto X da seguinte forma:

$$x_{i,s}^{(k+1)} = \begin{cases} lb_s & \text{se } x_{i,s}^{(k+1)} < lb_s \\ ub_s & \text{se } x_{i,s}^{(k+1)} > ub_s \end{cases}.$$

Passo 3 (Memorização do melhor ponto): Após a geração de uma nova população, estes são novamente ordenados (por ordem crescente da função objetivo) e passam para a iteração seguinte, onde $x_1^{(k)}$ é o melhor ponto da população, pois está no topo da ordenação. O número de iteração k é atualizado para $k+1$.

Passo 4 (Verificação do critério de paragem): Se o número máximo de iterações permitido (ou outra condição de paragem) é atingido, o algoritmo termina e devolve o melhor ponto da população, x_{best}. Caso contrário, o algoritmo retorna ao Passo 2.

Note-se que o FA é controlado por três parâmetros: o parâmetro aleatório α, a atratividade β e o coeficiente de absorção γ. De acordo com a definição destes parâmetros, o FA distingue dois comportamentos assimptóticos. Se $\gamma \to 0$, a atratividade torna-se constante pois $\beta \simeq \beta_0$. Por outro lado, quando $\gamma \to \infty$ a atratividade é quase nula, o que significa que os pontos se movem de forma aleatória (Arora & Singh, 2013). Uma outra característica importante é que, de acordo com as experiências efetuadas, o desempenho do algoritmo é moderadamente afetado pela dimensão das populações utilizadas, não sendo

assim necessárias populações de grande dimensão para que este convirja para soluções de boa qualidade.

O pseudo-código do FA original para a resolução de problemas do tipo (1.2.1) encontra-se descrito no Algoritmo 1.

Algoritmo 1 Pseudo-código do FA original

Dados: k_{max}, α, β_0, γ, N.

Fazer $k = 1$

Gerar aleatoriamente uma população de N pirilampos, $x_i^{(k)} \in X, i = 1, ..., N$

Calcular $f(x_i^{(k)})$ para $i = 1, ..., N$

Ordenar por ordem crescente os pirilampos, de acordo com os valores da função objetivo

Enquanto $k \leq k_{max}$

- **Para** $i = 1$ **até** N
  - **Para** $j = 1$ **até** N
    - **Se** $f(x_i^{(k)}) > f(x_j^{(k)})$ **então**
      - Calcular β usando (2.1.1)
      - Mover o pirilampo i em direção ao pirilampo j através de (2.1.3)
    - **fim**;
  - **fim**;
- **fim**;
- Projetar $x_i^{(k+1)}$ em X, para $i = 1, ..., N$
- Calcular $f(x_i^{(k+1)})$ para $i = 1, ..., N$
- Ordenar por ordem crescente os pirilampos, de acordo com os valores da função objetivo
- Fazer $k = k + 1$

fim;

Saída: $x_1^{(k)}$

2.2 Algumas variantes do FA

Desde a sua criação, têm sido desenvolvidas muitas variantes e extensões do FA com o objetivo de acelerar a convergência do algoritmo e melhorar a qualidade das soluções obtidas. De seguida, descrevem-se algumas destas variantes e alterações introduzidas por

diversos autores no FA original, na resolução de BCOP. Como já foi referido, os parâmetros de controlo do FA são α, que controla o termo aleatório e, consequentemente, a diversidade das soluções geradas, e os parâmetros γ e β, que definem o grau de atratividade entre os pirilampos. As modificações propostas por diversos autores incidem essencialmente na alteração do cálculo dos parâmetros de controlo (tornando-os variáveis e adaptativos), na alteração da forma como os pontos são movimentados, e no uso de diferentes distribuições estatísticas na geração do vetor de números ϵ.

Em (Arora, Singh, Singh, & Sharma, 2014), os autores introduziram o conceito de mutação para tornar o FA mais eficiente, designando-o de FA mutante. A ideia principal desta variante passa por incorporar nos pirilampos menos brilhantes informações dos pirilampos mais brilhantes, usando as informações destes para acelerar a convergência do algoritmo. Neste método, apenas alguns dos pirilampos mais brilhantes participam na mutação. Assim, um operador de mutação é usado, que seleciona alguns elementos de entre os melhores com probabilidade p_m (probabilidade de mutação) de forma a permitir que os pirilampos menos brilhantes não fiquem retidos em ótimos locais. Para decidir qual o pirilampo que participa na mutação do menos brilhante, é calculada, no final de cada iteração k, a probabilidade $p_m^{(k)}(x_i) = f(x_i^{(k)}) - f(x_i^{(k-1)})$. Os resultados computacionais mostraram que FA mutante tem um melhor desempenho e é mais eficiente que o FA original.

Tilahun e Ong (2012), sugerem que o cálculo do coeficiente β_0 seja feito em função das intensidades do brilho emitido pelos pirilampos. Quando um pirilampo i é movido em direção a um pirilampo j, a proposta é considerar $\beta_0 = e^{I_{0,j} - I_{0,i}}$, onde $I_{0,j}$ e $I_{0,i}$ são as intensidades de luz emitidas pelos pirilampos j e i, respetivamente, quando $r = 0$. Além disso, os autores utilizam uma estratégia diferente na movimentação do melhor ponto da população, quando este não é alterado em várias iterações consecutivas. Esta estratégia passa por movimentar esse ponto segundo um conjunto de vetores gerados aleatoriamente pela distribuição Uniforme. As experiências realizadas mostraram que as estratégias adotadas melhoraram os resultados obtidos.

Shafaati e Mojallali (2012), propuseram uma nova forma de cálculo do parâmetro α, bem como uma nova equação que descreve o movimento dos pontos. Neste proposta, α

decresce da seguinte forma

$$\alpha^{(k)} = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})e^{-k},$$

onde k é o número da iteração corrente, e $\alpha_{\max}$ e $\alpha_{\min}$ são definidos pelo utilizador. Note-se que o parâmetro α diminui exponencialmente, e ao fim de algumas iterações toma um valor muito próximo de $\alpha_{\min}$. Na equação que descreve o movimento dos pontos, foi adicionado um novo termo que envolve o melhor ponto da população:

$$x_i^{(k+1)} = x_i^{(k)} + \beta(x_j^{(k)} - x_i^{(k)}) + \alpha\epsilon_i + \lambda\epsilon_i(x_i^{(k)} - x_{\text{best}}) \ ,$$

onde ϵ_i é um vetor gerado pela distribuição de Gauss ou pela distribuição Uniforme, e λ é um parâmetro similar a α. Neste estudo, os resultados experimentais demonstraram que o FA com estas modificações obtém melhores resultados, ou pelo menos iguais, em termos de convergência, quando comparados com os resultados obtidos com o FA original.

Em (Shakarami & Sedaghati, 2014), para potenciar a capacidade de diversificação e intensificação do FA, os autores propuseram a seguinte fórmula de atualização para α:

$$\alpha^{(k+1)} = \left(\frac{1}{2k_{\max}}\right)^{\frac{1}{k_{\max}}} \alpha^{(k)},$$

onde k é o número da iteração corrente e k_{max} é o número máximo de iterações permitido. Além desta atualização dinâmica de α, para aumentar a diversidade das soluções, os autores aplicaram operadores de mutação e cruzamento a cada elemento da população.

Como a atratividade entre dois pirilampos depende da intensidade do brilho, da distância entre eles, e do coeficiente de absorção de luz do meio ambiente, quando dois pirilampos estão muito próximos ou muito distantes, o termo de atratividade da equação que descreve o movimento dos pontos pode variar drasticamente. Para contornar esta situação, Lin et al. (2013) alteraram a forma como dois pirilampos são atraídos entre si, adotando uma nova fórmula para o cálculo da distância entre dois pirilampos, $r' \in [0, 1]$, dada por

$$r' = \frac{r - r_{\min}}{r_{\max} - r_{\min}} \ ,$$

onde $0 \leq r_{\min} < r_{\max}$, $r_{\max} = \sqrt{\sum_{s=1}^{n} (x_{s,\min} - x_{s,\max})^2}$, $x_{s,\max}$ e $x_{s,\min}$ são os valores máximos e mínimos da coordenada s, respetivamente, de entre todos os pontos da população, e r

é a distância Euclideana entre os pirilampos. O termo de atratividade β proposto neste estudo é definido por

$$\beta = \beta_0\gamma(1 - r').$$

Com o objetivo de eliminar o domínio do termo aleatório sobre o termo da atratividade na equação que descreve o movimento dos pontos, os autores eliminaram o termo aleatório da equação (2.1.3), substituindo-o por um fator estocástico para produzir uma aleatoriedade adaptativa. Assim, a equação (2.1.3) é substituída por

$$x_i = x_i + \beta(x_j - x_i)\alpha\epsilon_i \ ,$$

onde $\epsilon_i \sim U[0, 1]$. Os resultados obtidos num conjunto de problemas teste mostraram que o FA com a implementação destas propostas tem um desempenho significativamente melhor do que o FA original, quer em termos de velocidade de convergência, quer em termos de precisão das soluções obtidas.

Os problemas de agrupamentos de dados (do inglês, *data clustering problems*) surgem com muita frequência em Estatística, cujo objetivo é classificar um conjunto vasto de dados segundo algumas categorias, de acordo com os seus padrões apresentados. O FA também tem sido utilizado com sucesso para resolver um conjunto diversificado de problemas desta categoria. Por exemplo, Hongwei, Liwei, e Dongzheng (2015), no contexto de um problema de agrupamento de dados, propõem uma nova equação que descreve o movimento dos pirilampos, em cada iteração k, dada por

$$x_i^{(k+1)} = x_i^{(k)} + \beta(x_j^{(k)} - x_i^{(k)}) + \beta_0 e^{-\gamma r_{i,x_{\text{best}}}^2}(x_{\text{best}} - x_i^{(k)}) + \alpha\epsilon_i \ ,$$

em que x_{best} é a posição do melhor pirilampo encontrado até então, $r_{i,x_{\text{best}}}^2$ é a distância Euclideana entre x_{best} e x_i, e $\epsilon_i \sim U[0, 1]$. Segundo os autores, o objetivo é aumentar a influência do pirilampo mais brilhante no movimento dos restantes pirilampos da população. Esta influência é dada pelo segundo e terceiros termos do segundo membro da equação do movimento: o segundo termo tem em conta a atratividade existente entre o pirilampo i e o pirilampo j, enquanto que o terceiro termo tem em conta o histórico da população, nomeadamente a posição do pirilampo mais brilhante. Os resultados demonstraram que esta

versão do FA encontra melhores soluções em todos os problemas teste utilizados, quando comparados com resultados obtidos com o FA original.

Uma outra variedade de problemas de otimização são os problemas de despacho económico. Um exemplo de aplicação de um FA modificado para este tipo de problemas é apresentado em (Maidl, de Lucena, & dos Santos Coelho, 2013). Neste estudo, a melhor solução encontrada ao longo das iterações, x_{best}, é armazenada em memória, e a equação que descreve o movimento dos pirilampos é dada por

$$x_i^{(k+1)} = x_i^{(k)} + a\beta(x_{\text{best}} - x_i^{(k)}) + b\alpha(\epsilon - 0.5),$$

onde $a = \frac{f(x_j^{(k)}) - f(x_{best})}{f_{\max} - f_{\min}}$ e $b = x_{i,\max}^{(k)} - x_{i,\min}^{(k)}$, sendo $x_{i,\max}$ e $x_{i,\min}$ a maior e menor coordenada do vetor $x_i^{(k)}$, respetivamente, e $f_{\max}$ e $f_{\min}$ são os valores máximos e mínimos da função objetivo de entre todos os pontos da população. Segundo os autores, o termo a fornece um equilíbrio ao movimento dos pontos da população através da normalização dos respetivos valores da função objetivo, enquanto que o termo b orienta o movimento aleatório, que pode ser útil para evitar a redução do desempenho do algoritmo.

Recentemente surgiram os algoritmos de otimização caótica (do inglês, *chaotic optimization algorithms*), que utilizam sequências caóticas geradas por mapas caóticos para substituir os parâmetros ou variáveis aleatórias do algoritmo. Nestes algoritmos, devido às propriedades de não-repetição e ergodicidade do caos (previsibilidade probabilística), os procedimentos de procura global tornam-se mais eficientes do que os métodos estocásticos que dependem muitas vezes de parâmetros probabilísticos (Santos Coelho & Mariani, 2008). Os resultados têm demonstrado que é uma estratégia poderosa para diversificar a população e melhorar o desempenho na prevenção da convergência prematura para mínimos locais. Estes algoritmos têm sido estendidos para resolver diversos problemas de otimização nas mais diversas áreas de conhecimento, como por exemplo na Física, Engenharia, Economia, Biologia e Ciências da computação. Estudos que incorporam no FA funções de caos podem ser encontrados em (Santos Coelho, de Andrade Bernert, & Mariani, 2011; A. Gandomi, Yang, Talatahari, & Alavi, 2013; Abdel-Raouf, Abdel-Baset, & El-henawy, 2014; Long, Meesad, & Unger, 2015).

Para finalizar, é de referir que em (Francisco et al., 2014) se fez um estudo experimental da implementação do algoritmo FA na resolução de um conjunto de BCOP. O objetivo foi fazer uma primeira implementação do algoritmo para a familiarização com os seus principais fundamentos, analisar a sensibilidade dos parâmetros de controlo e avaliar o seu desempenho. Numa primeira abordagem, foram efetuadas experiências baseadas na substituição da utilização da norma Euclideana (norma-2) pelas norma-1, norma-4, norma-10 e norma-infinita em (2.1.1), com o objetivo de estudar o desempenho e a robustez do FA. Dos resultados computacionais, pode-se concluir que o FA usando (2.1.1) com a norma-1, apresenta melhores resultados, quando comparados com a utilização das outras normas. Numa segunda abordagem, foram consideradas duas novas fórmulas para o cálculo do parâmetro de atratividade, β^1 e β^2. A fórmula de atualização β^1 é dada por

$$\beta_{i,j}^1 = \begin{cases} \beta_0 e^{-\gamma r_{i,j}^2} & \text{se} \quad r_{i,j} \leq a \\ 0.5 & \text{se} \quad r_{i,j} > a \end{cases}. \tag{2.2.1}$$

onde $a = 0.5$. Note-se que β^1 decresce exponencialmente com a distância entre os pirilampos, se a distância for menor ou igual que a. Quando a distância entre os pirilampos for maior que a, β^1 assume o valor constante de 0.5. Esta escolha deve-se ao facto de em (2.1.1) a atratividade entre pirilampos ser muito pequena para certos valores de $r_{i,j}$. A outra fórmula de atualização de atratividade, β^2, tem em consideração a taxa média da distância entre os pirilampos, e é dada por

$$\beta_{i,j}^2 = \begin{cases} \beta_0 e^{-\frac{f(x_i)-f(x_j)}{r_{i,j}}} & \text{se} \quad f(x_i) > f(x_j) \\ 0 & \text{caso contrário} \end{cases}. \tag{2.2.2}$$

Os resultados obtidos usando as fórmulas de atratividade (2.2.1), (2.2.2) e (2.1.1), com a norma-1 no cálculo da distância entre os pontos, permitem concluir que o FA com (2.2.2) revelou-se tão eficiente quanto com (2.1.1). Contudo, o uso de (2.2.1) revelou-se mais robusto. Mais pormenores do estudo e os resultados computacionais obtidos com estas propostas na resolução do referido conjunto de problemas podem ser consultados em (Francisco et al., 2014).

2.3 Outros algoritmos estocásticos

Como já foi referido, os métodos estocásticos são uma classe importante de métodos contemporâneos para resolver problemas de otimização complexos, de que são exemplos os *Genetic Algorithms* (GA) (Holland, 1975), o algoritmo *Ant Colony Optimization* (Dorigo, 1992), o algoritmo *Particle Swarm Optimization* (Kennedy & Eberhart, 1995), o algoritmo *Harmony Searc*h (Geem, Kim, & Loganathan, 2001), o algoritmo *Simulated Annealing* (SA)(Kirkpatrick, 1984), o algoritmo *Differential Evolution* (DE) com controlo de parâmetros adaptativos (Storn, 1996; Storn & Price, 1997), o algoritmo *Evolution Strategy with Covariance Matrix Adaptation* (CMA-ES) (Hansen & Ostermeier, 2001; Hansen, 2006), o algoritmo *Artificial Bee Colony* (Karaboga & Basturk, 2007; Karaboga, Gorkemli, Ozturk, & Karaboga, 2014) e o FA (X.-S. Yang, 2008). Estes métodos, inicialmente concebidos para resolver BCOP, têm sido objeto de estudo da comunidade científica com o objetivo de os tornar cada vez mais robustos e eficazes na resolução de problemas complexos e de grandes dimensões com baixo tempo computacional. Estes algoritmos usam diferentes tipos de exploração do espaço de procura, bem como diferentes mecanismos de seleção dos elementos das populações. De seguida são descritos de uma forma sumária alguns dos métodos mencionados.

O algoritmo PSO é um algoritmo estocástico baseado em populações, cuja tradução literal significa "otimização de enxame de partículas", e é uma metaheurística inspirada na atividade de grupos de animais, como por exemplo, bandos dos pássaros (Kennedy & Eberhart, 1995). Estes indivíduos são considerados como partículas que se encontram numa determinada posição do espaço e que o exploram à procura de uma solução apropriada para um determinado problema de otimização. Neste método, cada partícula tem uma determinada velocidade associada, que será responsável pela exploração do espaço de procura para encontrar a solução ótima. Tanto o movimento entre posições, como a velocidade a que se desloca cada partícula, são dependentes da sua última localização conhecida e da localização da melhor partícula. A população inicial, de dimensão N, é gerada atribuindo a cada partícula/ponto uma localização aleatória no espaço de procura, e uma

velocidade, que é compreendida entre limites definidos pelo utilizador. Assim, em cada iteração k, e para cada partícula $x_i^{(k)}$, (vetor que define a posição da partícula i na iteração k), a velocidade $v_i^{(k+1)}$ é atualizada de acordo com a equação

$$v_i^{(k+1)} = w.v_i^{(k)} + \left(c_1.\xi_1.(x_{i,best}^{(k)} - x_i^{(k)})\right) + c_2.\xi_2.(x_{best}^{(k)} - x_i^{(k)}), \quad i = 1, \ldots, N,$$

onde $v_i^{(k)}$ é a velocidade atual da partícula i, w é a componente inercial, cujo objetivo é ajustar a capacidade de exploração e intensificação do algoritmo, c_1 e c_2 são coeficientes de aceleração positivos definidos pelo utilizador, $\xi_1, \xi_2 \sim \mathrm{U}[0,1]$, $x_{i,best}^{(k)}$ é a melhor posição obtida pela partícula i ao longo das iterações e $x_{best}^{(k)}$ é a posição da melhor partícula do enxame. Note-se que os coeficientes de aceleração c_1 e c_2 afetam o desempenho do algoritmo. Várias formas adaptativas do cálculo destes parâmetros estão disponíveis em (Rezaee Jordehi & Jasni, 2013), onde alguns valores constantes fixos para c_1 e c_2 se mostraram mais apropriados. Assim, em cada iteração, a nova posição da partícula i é definida por

$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)}.$$

Caso as coordenadas dos novos pontos não estejam dentro dos limites lb e ub, estes são projetados em X. A forma de atualização mais comum da componente inercial é feita por um decréscimo linear de um valor máximo, $w_{\max}$, até um valor mínimo, $w_{\min}$, ao longo do processo iterativo. Uma análise de convergência e de sensibilidade dos parâmetros do algoritmo PSO pode ser encontrada em (Jiang, Luo, & Yang, 2007; Trelea, 2003). Desde a sua criação, muitas extensões deste algoritmo têm sido propostas para a resolução de problemas nas mais diversas áreas, inclusive para resolver problemas de MINLP (Suganthan, 1999; Yiqing, Xigang, & Yongjian, 2007; C. C. Coello & Lechuga, 2002; Cagnina, Esquivel, & Coello, 2011).

O algoritmo DE é uma técnica evolutiva baseada em populações que assenta em três princípios para definir os pontos da iteração seguinte (Storn & Price, 1997): mutação, cruzamento e seleção. A população inicial de tamanho N é gerada aleatoriamente no conjunto X, e o tipo de mutação frequentemente utilizada é denominada de DE/rand/1, e

define o ponto mutante i, denotado por v_i, da seguinte forma:

$$v_i = x_{r_1} + F(x_{r_2} - x_{r_3}),$$

onde r_1, r_2, r_3 são índices aleatórios uniformemente escolhidos do conjunto $\{1, 2, \ldots, N\}$, mutuamente diferentes entre si e diferentes do índice i. O operador cruzamento visa aumentar a diversidade das coordenadas $s, s = 1, \ldots, n$, onde n é a dimensão do problema, do ponto mutante v_i, gerando assim um ponto temporário t_i da seguinte forma:

$$t_{i,s} = \begin{cases} v_{i,s} & \text{se } \xi \leq CR \text{ ou } s = j_i \\ x_{i,s} & \text{caso contrário} \end{cases},$$

onde $\xi \sim \mathrm{U}[0,1]$, j_i é um índice aleatoriamente escolhido do conjunto $\{1, \ldots, N\}$ e F e CR são parâmetros de controlo positivos. Na versão jDE (Brest, Greiner, Boskovic, Mernik, & Zumer, 2006) do algoritmo DE, são definidos parâmetros de controlo auto-adaptativos através da geração de valores diferentes de F_i e CR_i para cada ponto i da população. Inicialmente, cada valor de F_i é gerado aleatoriamente no intervalo $[F_l, F_u]$, onde F_l e F_u são os valores dos limites inferiores e superiores de F, respetivamente, e cada valor de CR_i é gerado aleatoriamente no intervalo $[0, 1]$. Assim, em cada iteração,

$$F_i = F_l + \xi_1(F_u - F_l) \text{ se } \xi_2 < \tau_1 \text{ e } CR_i = \xi_3 \text{ se } \xi_4 < \tau_2 ,$$

caso contrário, estes valores são mantidos para a próxima iteração, onde $\xi_j \sim \mathrm{U}[0,1], j = 1, \ldots, 4$ e τ_1, τ_2 representam probabilidades de ajuste dos parâmetros F_i e CR_i, respetivamente. Caso o ponto temporário não pertença ao espaço de procura, é efetuada uma projeção, coordenada a coordenada, do ponto em X. Por fim, é usado um operador de seleção para comparar cada ponto temporário t_i com x_i, e se o valor da função objetivo melhorou, então t_i será considerado o ponto corrente para a iteração seguinte. Caso contrário, x_i será mantido.

O algoritmo *Evolution Strategy (ES)* é o primeiro e mais antigo algoritmo evolutivo a surgir na literatura, e é baseado nos conceitos de adaptação e evolução. O CMA-ES é um algoritmo ES, no sentido de que para um conjunto de 'pais' (pontos), a descendência

é criada por amostragem de distribuições de Gauss, onde o melhor 'descendente' será o progenitor seguinte (Hansen & Ostermeier, 2001). Os parâmetros mais importantes são: $\lambda \geq 2$, que é o tamanho da população (ou tamanho da descendência), $\mu \leq \lambda$ é número de 'pais', ou seja, pontos de pesquisa da população, que é um número estritamente positivo de pesos de recombinação, $\sigma_k > 0$ e $C_k \in \mathbb{R}^{n\times n}$ são o comprimento do passo e a matriz de covariância, respetivamente, na iteração k. A notação $\mathrm{N}(m, C) \sim m + \mathrm{N}(0, C)$ representa a distribuição Normal multivariada com valor médio $m \in \mathbb{R}^n$ e C uma matriz de covariância simétrica e definida positiva. Em cada iteração k, CMA-ES gera uma população de λ pontos (a descendência) por amostragem de uma distribuição Normal multivariada como se segue:

$$x_i^{(k+1)} = m + \sigma\xi, \text{ para } i = 1, \ldots, \lambda, \tag{2.3.1}$$

onde $\xi \sim \mathrm{N}(0, C)$ e σ é o comprimento de passo. À semelhança de outras metaheurísticas, estes pontos são, se necessário, projetados na espaço de procura X. Por razões de simplicidade, o índice relativo ao contador de iterações k não será usado nas fórmulas de atualização. As fórmulas de atualização para m, C e σ para a iteração $k+1$ são apresentadas de seguida. Em (Hansen, 2016), podem ser encontradas descrições mais detalhadas. A média m é uma média ponderada de μ pontos selecionados a partir da população $x_1, x_2, \ldots, x_\lambda$:

$$m = \sum_{i=1}^{\mu} w_i x_{i:\lambda} \text{ onde } \sum_{i=1}^{\mu} w_i = 1,\ w_1 \geq \cdots \geq w_\mu > 0,$$

e $x_{i:\lambda}$ é o i-ésimo melhor ponto de entre os pontos calculados segundo (2.3.1), que no contexto da resolução do problema (1.2.1), significa que o índice '$i : \lambda$' corresponde à i-ésima posição do melhor valor da função objetivo, depois de ordenados todos os seus valores. Quando $\mu > 1$, a equação para m implementa uma recombinação intermédia ponderada, resultando na correspondente equação de atualização para m:

$$m = m + c_m \sum_{i=1}^{\mu} w_i \left(x_{i:\lambda} - m\right),$$

onde $c_m \leq 1$ representa uma taxa de aprendizagem. Em cada iteração, o algoritmo guarda na memória o melhor ponto encontrado até então, e cada vez que um novo m é calculado,

uma seleção (gulosa) entre m e o melhor ponto é aplicada para escolher aquele com melhor valor da função objetivo.

A atualização da ordenação dos valores de μ da matriz de covariância baseada nos valores de λ é dada por

$$C = \left(1 - c_\mu \sum_{i=1}^{\lambda} w_i\right) C + c_\mu \sum_{i=1}^{\lambda} w_i y_{i:\lambda} (y_{i:\lambda})^T,$$

onde $y_{i:\lambda} = (x_{i:\lambda} - m)/\sigma$ e $0 < c_\mu \leq 1$ é a taxa de aprendizagem. A escolha de c_μ é determinante para a convergência do algoritmo, apesar de uma boa definição não depender do problema em questão (Hansen, 2016). A taxa de variação do comprimento do passo σ é controlada por um parâmetro de "amortecimento" $d_\sigma \approx 1$ e $c_\sigma < 1$:

$$\sigma = \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{E[\|\xi\|]} - 1\right)\right),$$

onde $E[\|\xi\|]$ é o valor esperado da norma euclidiana de um vetor $\xi \sim \mathrm{N}(0, I)$ e o vetor p_σ é um caminho de evolução conjugado que é construído por

$$p_\sigma = (1 - c_\sigma) p_\sigma + c_n C^{-1/2} \sigma^{-1} \Delta m,$$

onde Δm é a diferença entre as médias nas iterações $k + 1$ e k, e c_n é uma constante de normalização. Durante a inicialização, $m \sim \mathrm{U}[lb, ub]$, $C = I$, $p_\sigma = 0$ e σ é definido para depender de $[lb, ub]$.

O algoritmo ABC é um algoritmo de otimização baseado no comportamento inteligente de uma colónia de abelhas nas suas atividades diárias de exploracão do terreno próximo da colmeia, para tentar encontrar novas fontes de alimento (Karaboga & Basturk, 2007). A posição das fontes de alimento representam uma possível solução do problema de otimização, e a quantidade de néctar na fonte traduz a qualidade dessa solução. Essas soluções são geradas inicialmente de forma aleatória. De acordo com Karaboga e Akay (2009), neste algoritmo existem três tipos diferentes de abelhas: as abelhas que trabalham na recolha de alimento (*employed bees*), as abelhas que estão à espera na colónia para decidirem qual a fonte de alimento a escolher (*onlooker bees*) e as abelhas que fazem a procura de

novas fontes de alimento (*scout bees*). Estes três tipos de abelhas elaboram as tarefas diárias necessárias à colmeia, até que o número de iterações permitido seja atingido. O número de fontes de alimento é o número de *employed bees*, $N/2$, onde N é o tamanho da colónia de abelhas. Numa fase inicial, o conjunto das posições das fontes de alimento $x_i, i = 1, \ldots, N/2$ é gerado aleatoriamente em X, e a quantidade de néctar é determinada pelo valor da função objetivo f. Durante a fase das *employed bees* é gerado um grupo de soluções mutantes numa área circundante, que são candidatas a novas fontes de néctar, da seguinte forma:

$$v_{i,j} = x_{i,j} + \xi(x_{i,j} - x_{i,s}), \quad j = 1, \ldots, N, \tag{2.3.2}$$

para $i = 1, \ldots, N/2$, s é um índice escolhido aleatoriamente do conjunto $\{1, \ldots, i-1, i+1, \ldots, N/2\}$, j é um número aleatório pertencente a {1,...,N} e $\xi \sim \mathrm{U}[-1, 1]$. Se alguma coordenada da solução mutante estiver fora do intervalo $[lb_s, ub_s]$, esta é projetada para os seus limites. De seguida, a solução mutante v_i é comparada com x_i, e é escolhida aquela que apresentar um melhor valor da função objetivo f. Se a solução corrente x_i não tiver sido melhorada, o contador provisório de iterações é incrementado. Caso contrário, esse contador é reiniciado com o valor 0. Por outro lado, durante a fase das *onlooker bees*, as fontes de alimento são escolhidas de acordo com os valores de probabilidade P_i que dependem da quantidade de néctar da fonte i, sendo calculada por

$$P_i = \frac{\alpha_P F_i}{\max\limits_{j=1,\ldots,N/2} F_j} + \beta_P,$$

onde F_i é o valor da função de avaliação do ponto x_i dado por

$$F_i = \begin{cases} (f(x_i)+1)^{-1} & \text{se } f(x_i) \geq 0 \\ 1 + |f(x_i)| & \text{caso contrário} \end{cases},$$

e α_P e β_P são constantes pré-definidas. A solução mutante é calculada a partir da solução escolhida anteriormente de acordo com a equação (2.3.2) e a seleção é feita entre o ponto x_i e o seu respetivo ponto mutante v_i. Finalmente, apenas uma *scout* é permitida em cada iteração. Se o contador provisório da solução que melhorou menos exceder um valor pré

determinado, chamado de 'limit', essa posição é abandonada e substituída por uma posição gerada aleatoriamente em X.

Capítulo 3

Técnicas de tratamento de restrições

Uma das questões centrais na resolução de um COP é como tratar as restrições não lineares de desigualdade e igualdade do problema. Em geral, a versão original dos algoritmos estocásticos destinam-se à resolução de BCOP, e não possuem mecanismos para lidar com as restrições genéricas do problema. Esta é uma das principais áreas de estudo que está permanentemente em investigação, cuja motivação prende-se com o desenvolvimento de técnicas para o tratamento de restrições com o objetivo de orientar os mecanismos de procura para regiões onde se encontram as soluções admissíveis. Neste capítulo são discutidas técnicas para o tratamento de restrições genéricas em problemas de NLP e técnicas para o tratamento de restrições de integralidade em problemas de MINLP.

3.1 Restrições genéricas

Nesta secção são apresentadas algumas técnicas para o tratamento de restrições genéricas em problemas de NLP existentes na literatura. De uma forma geral, estas técnicas podem ser divididas em dois grandes grupos: técnicas que tratam a função objetivo e as restrições simultaneamente, e técnicas que tratam a função objetivo e as restrições separadamente. Relativamente às tecnicas que tratam simultaneamente a função objetivo e as restrições em problemas de NLP com restrições genéricas destacam-se os métodos de

penalidade.

Os métodos de penalidade foram inicialmente propostos por Courant et al. (1943) nos anos 40, e foram posteriormente expandidos por A. V. Fiacco e McCormick (1966) e A. Fiacco e Jones (1969) para resolver problemas de NLP com restrições genéricas. É de notar que os métodos de penalidade podem ser usados quer com métodos estocásticos baseados em populações quer com métodos estocásticos baseados em ponto-por-ponto. Na descrição que se segue, assume-se que as restrições de limites simples são garantidas pelo método estocástico usado. Considere-se o problema de NLP com restrições genéricas dado por (1.2.3). Neste contexto, os métodos de penalidade substituem o problema (1.2.3) por uma sucessão de problemas com restrições de limites simples (1.2.1), ou apenas num, usando funções de penalidade, os quais podem então ser resolvidos por métodos estocásticos eficientes da NLP, devidamente estudados e documentados. Em geral, as funções de penalidade são definidas através da adição de um termo de penalidade à função objetivo, o qual é multiplicada por algum parâmetro de penalidade positivo. Assim, o tipo de subproblemas a resolver, em cada iteração k, é dado por:

$$\begin{aligned} &\text{minimizar} \quad F(x, c^{(k)}) = f(x) + c^{(k)} p(x) \\ &\text{sujeito a} \quad x \in X \subset \mathbb{R}^n, \end{aligned} \tag{3.1.1}$$

onde X é o conjunto definido em (1.2.2) e $p(x)$ é o termo de penalidade. O *termo de penalidade* p depende da violação das restrições e deve ter as seguintes propriedades:

- $p(x) = 0$ se $x \in \Omega$ dado por (1.2.4);
- $p(x) > 0$ se $x \notin \Omega$.

A função F definida em (3.1.1) designa-se por *função de penalidade*. Ao fazer tender o parâmetro de penalidade para infinito, vai-se penalizando cada vez mais severamente a violação das restrições, forçando o minimizante da função de penalidade a aproximar-se cada vez mais da região admissível do problema (1.2.3) (Nocedal & Wright, 2006). O objetivo dos métodos de penalidade é obter uma sucessão de minimizantes do problema (3.1.1), $x^*(c^{(k)})$, que converge para a solução ótima do problema (1.2.3), quando $c^{(k)} \to \infty$.

Estes métodos de penalidade são designados na literatura de *métodos de penalidade exterior* porque o termo de penalidade é não nulo apenas quando x é não admissível, os quais aproximam-se da região admissível à medida que o parâmetro de penalidade aumenta. Nos chamados *métodos de penalidade interior*, a função de penalidade não está definida para pontos não admissíveis. Neste tipo de métodos todos os pontos têm de ser admissíveis ao longo de todo o processo iterativo. Assim, o método começa com um ponto admissível, e os pontos gerados nas iterações seguintes têm que pertencer à região admissível, o que representa uma complexidade acrescida relativamente aos métodos de penalidade exterior (Nocedal & Wright, 2006).

Os métodos de penalidade podem ainda ser classificados de *exatos* ou *não exatos*. Os *métodos de penalidade exata* tem a seguinte propriedade: para um certo valor finito do parâmetro de penalidade c, a solução do problema (1.2.1) coincide com a solução do problema (1.2.3). Estes métodos são muito eficientes uma vez que qualquer minimizante de (1.2.1) é um minimizante do problema com restrições ((1.2.3), e reciprocamente, para um conjunto de valores finitos do parâmetro de penalidade (Nocedal & Wright, 2006).

O teorema seguinte garante a convergência dos métodos de penalidade, quando a função objetivo e as funções de restrição do problema são função contínuas.

Teorema 3.1.1. *(Theorem 2.1 em (Freund, 2004)) Suponhamos que* $\Omega \neq \emptyset$*,* f*,* $g_i(x), i = 1, \ldots, m, h_j(x), j = 1, \ldots, p,$ *e* $p(x)$ *são funções contínuas. Seja* $\{x_k\}_{k=1}^{\infty}$ *uma sucessão de soluções do problema (3.1.1) com* $c^{(k)} \to \infty$*. Então, qualquer ponto de acumulação* $\bar{x}$ *de* $\{x_k\}_{k=1}^{+\infty}$ *é um minimizante do problema com restrições (1.2.3).*

Existem várias funções de penalidade disponíveis na literaturas, e que diferem entre si de acordo com o termo de penalidade usado. Como por exemplo função de penalidade de: morte súbita (onde o ponto é simplesmente rejeitado caso seja não admissível), exata, estática, dinâmica, e adaptativa (Silva et al., 2011; Tessema & Yen, 2006; Joines & Houck, 1994). Nos métodos baseados em populações, as abordagens mais usadas para resolver problemas de otimização com restrições são os métodos de penalidade exterior. Uma revisão teórica bem como uma comparação prática sobre o uso de funções de penalidade

em algoritmos evolutivos pode ser consultada em (C. A. C. Coello, 2002; Mezura-Montes & Coello, 2011).

De seguida faz-se uma sistematização de alguns tipos de funções de penalidade existentes na literatura, apresentando alguns exemplos clássicos, os seus pressupostos e limitações.

As duas funções de penalidade mais simples, do tipo exterior, tem a seguinte formulação:

$$F(x, c^{(k)}) = f(x) + c^{(k)} \left[\sum_{i=1}^{m} \left[\max \left\{ 0, g_i(x) \right\} \right]^q + \sum_{j=1}^{p} |h_j(x)|^q \right],$$

onde $c^{(k)} > 0$ é o parametro de penalidade e para $q = 1$ é a função de penalidade linear exata l_1; e para $q = 2$ é a função de penalidade quadrática l_2. Estas funções de penalidade são bastante usadas, como se pode ver em (M. Ali & Zhu, 2013; Barbosa & Lemonge, 2008; Mezura-Montes & Coello, 2011; Lemonge, Barbosa, & Bernardino, 2015). A maioria dos parâmetros de penalidade dependem do problema a resolver e requerem *a priori* um conhecimento do grau de violação das restrições do problema. Além disso, na prática, é difícil determinar à partida os valores desses parâmetros, sendo também necessárias regras para ajustar esses parâmetros ao longo processo iterativo. O objetivo dos parâmetros de penalidade é evitar a sobre- e a sub-penalização, e o desempenho dos resultados obtidos é fortemente influenciado pelos parâmetros adotados. Por isso, a definição dos valores iniciais e decorrentes estratégias de atualização são questões críticas num método baseado em funções de penalidade.

As funções de penalidade estáticas têm como principal característica aplicar uma penalidade constante $c > 0$ a todos os pontos que não pertençam à região admissível. Por exemplo, em (Richardson, Palmer, Liepins, & Hilliard, 1989) a função de penalidade é definida em função do número de restrições violadas. Assim, considerando um problema de otimização apenas com restrições de desigualdade, a função de penalidade estática é definida por:

$$F(x, c) = f(x) + c \sum_{i=1}^{m} \delta_i \,,$$

onde

$$\delta_i = \begin{cases} 1 & \text{se a restrição } i \text{ é violada, } g_i(x) > 0 \\ 0 & \text{caso contrário} \end{cases}, i = 1, \ldots, m, \tag{3.1.2}$$

e o parâmetro de penalidade c é aplicado de igual forma a todas as restrições ao longo de todo o processo iterativo. Note-se que esta função de penalidade não tem em consideração o valor da violação da restrição no ponto. Porém, é mais comum e mais eficaz penalizar a função objetivo tendo em consideração uma medida do valor da violação das restrições no ponto. Assim, foram desenvolvidas abordagens de modo a incluir na função de penalidade um termo em função de uma métrica que fizesse depender a penalidade aplicada em função do valor da violação da restrição nos pontos não admissíveis (Richardson et al., 1989). Com este pressuposto, no seu caso geral, a função de penalidade considerada é:

$$F(x, c) = f(x) + c \sum_{i=1}^{m} (\delta_i g_i(x))^{\alpha} \, , i = 1, \ldots, m,$$

onde δ_i é definida por (3.1.2) e α é um parâmetro geralmente definido como sendo 1 ou 2. Alguns autores exploraram variantes desta técnica de penalidade definindo diferentes tipos de métricas (Back & Khuri, 1994; Goldberg, 1989; Huang, Kao, & Horng, 1994; Olsen, 1994).

Homaifar, Qi, e Lai (1994) desenvolveram uma técnica de penalidade exterior estática onde definem vários 'níveis de violação' para cada restrição. Nesta abordagem, para cada nível de violação, é aplicado um único parâmetro de penalidade, cujo valor aumenta, sempre que o nível de violação também aumenta. A função de penalidade proposta é dada por:

$$F(x, R) = f(x) + \sum_{i=1}^{m} R_{s,i} \{max\{0, g_i(x)\}^2, \;\; 1 \leq i \leq m \, ,$$

onde $R_{s,i}$ é o parâmetro de penalidade para o nível de violação s da restrição i, $s = 1, \ldots, l$, e m é o número de restrições de desigualdade. Quanto maior for o nível de violação maiores são os valores dos parâmetros de penalidade. Uma das fragilidades desta abordagem reside na quantidade excessiva de parâmetros de penalidade que têm que ser definidos pelo utilizador, cujos valores afetam fortemente o desempenho e a qualidade das soluções obtidas pelo método (Michalewicz, 1995). Se os parâmetros de penalidade são

pequenos, o algoritmo pode convergir para soluções não admissíveis; se são muito grandes, o método aproxima-se do método de penalidade de morte súbita, pois as soluções não admissíveis são rejeitadas. Deb (2001) e Onwubolu e Babu (2013) também apontaram dificuldades de como selecionar os valores apropriados para os parâmetros $R_{s,i}$, de modo a direcionar a procura para a região admissível. Uma limitação comum a todos os métodos de penalidade estática é a dificuldade em escolher o valor apropriado do seu parâmetro de penalidade.

As funções de penalidade dinâmicas são funções cujos parâmetros de penalidade dependem de alguma forma do número da iteração corrente do processo iterativo. Um exemplo clássico desta abordagem foi efetuada por Joines e Houck (1994) que, considerando apenas problemas com restrições de desigualdade, em cada iteração k, a função de penalidade é definida do seguinte modo:

$$F(x, c^{(k)}) = f(x) + c^{(k)} \sum_{i=1}^{m} (v_i(x))^{\theta(v_i(x))}, \tag{3.1.3}$$

onde $c^{(k)}$ é o parâmetro de penalidade na iteração k, $v_i(x) = \max\{0, g_i(x)\}$ é o valor da violação das restrições, para $i = 1, \ldots, m$, e $\theta(.)$ é uma função constante, que depende da violação de cada restrição, e é dada por $\theta(z) = 1$ se $z \leq 1$, e $\theta(z) = 2$, caso contrário. Notar que, a função de penalidade quadrática poder ser obtida a partir de (3.1.3), considerando θ a função constante $\theta(.) = 2$. Os autores propõem a seguinte fórmula de atualização para o parâmetro de penalidade

$$c^{(k)} = (C \times k)^a,$$

onde k é o número da iteração corrente, e as constantes C e a têm os valores $C = 0.5$ e $a = 1$ ou 2. Outra interessante e eficiente formulação de atualização para o parâmetro de penalidade, proposto por Petalas, Parsopoulos, e Vrahatis (2007), é dada por

$$c^{(k)} = D \times k\sqrt{k},$$

onde a constante D toma o valor 1. Mais exemplos de penalidades dinâmicas podem ser consultadas em (Petalas et al., 2007). Em (Francisco, Costa, & Rocha, 2015), fez-se um

estudo experimental da função de penalidade dinâmica (3.1.3) quando usada no contexto do algoritmo FA, para resolver problemas de NLP com restriçoes genéricas.

Em (Kazarlis & Petridis, 1998), os autores realizaram um estudo detalhado de uma função de penalidade dinâmica mais sofisticada definida do seguinte modo:

$$F(x, c^{(k)}) = f(x) + c^{(k)} \left[A \sum_{i=1}^{m} (\sigma_i \; w_i \; \Phi(d_i(x)) + B \right] \sigma_s \; ,$$

onde A é um 'fator de severidade', m é o número total de restrições (considerando apenas restrições de desigualdade), σ_i é 1 caso a restrição i seja violada e 0 caso contrário, w_i é um 'peso' atribuído à restrição i, $d_i(x)$ é uma medida do grau de violação da restrição i, $\Phi(\cdot)$ é uma função desta medida, B é um limite para o fator de penalidade, σ_s tem o valor de 1 se x é admissível e 0 caso contrário, e $c^{(k)}$ é um parâmetro de penalidade crescente no intervalo $[0, 1]$, onde k é o número da iteração corrente. Neste estudo, os autores concluíram que o método apresenta bom desempenho quando o parâmetro é atualizado usando a seguinte fórmula: $c^{(k)} = \frac{k}{k_{max}}$, onde $k_{\max}$ é o número máximo de iterações permitido.

As funções de penalidade adaptativas podem ser consideradas um caso mais geral das funções de penalidade dinâmicas, e têm sido aplicadas em métodos estocásticos baseados em populações de pontos. Estas funções caracterizam-se pelo facto de os parâmetros de penalidade serem atualizados ao longo do processo iterativo usando as informações produzidas pelo mesmo. Neste contexto, os parâmetros de penalidade são atualizados de acordo com o *feedback* do processo de procura, não sendo necessariamente apenas o número da iteração corrente. Este tipo de funções de penalidade foram introduzidos por Ben Hadj-Alouane e Bean (1997). Neste estudo, para cada ponto da população x_i na iteração k, a função de penalidade é dada por

$$F(x_i, c^{(k)}) = f(x_i) + c^{(k)} \left[\sum_{i=1}^{m} max\{0, g_i(x_i)\}^2 + \sum_{j=1}^{p} |h_j(x_i)| \right] ,$$

onde o parâmetro de penalidade $c^{(k)}$ é atualizado ao longo do processo iterativo da seguinte

forma:

$$c^{(k+1)} = \begin{cases} \frac{1}{\beta_1} c^{(k)} & \text{se Caso 1} \\ \beta_2 c^{(k)} & \text{se Caso 2} \\ c^{(k)} & \text{caso contrário} \end{cases} .$$

O Caso 1 verifica-se quando o melhor ponto das últimas nk iterações anteriores à iteração k da população é admissível. O Caso 2 verifica-se quando o melhor ponto das últimas nk iterações anteriores à iteração k da população nunca é admissível. Nesta formulação $\beta_1 > \beta_2 > 1$. Os valores para os parâmetros β_1, β_2 e nk terão que ser selecionados de modo apropriado para que o método tenha um bom desempenho. Note-se que, de acordo com a fórmula de atualização de $c^{(k)}$, o parâmetro de penalidade da iteração $k+1$ diminui sempre que os melhores elementos das últimas nk iterações são admissíveis, aumenta se os melhores elementos das últimas nk iterações são não admissíveis, e mantém-se constante caso contrário. As desvantagens associadas a este método referidas por Bäck, Fogel, e Michalewicz (2000) devem-se aos factos do método não possuir a capacidade de conduzir o processo iterativo para regiões promissoras ou promover um afastamento de regiões não promissoras, e da fórmula de atualização do parâmetro de penalidade não ter em consideração algum tipo de medida sobre a violação das restrições.

Hamda e Schoenauer (2000) usaram uma função de penalidade semelhante à proposta por Ben Hadj-Alouane e Bean (1997), adotando como critério para a atualização do parâmetro de penalidade a existência de uma proporção mínima de pontos admissíveis na população. A fórmula de atualização do parâmetro de penalidade adaptativo é dado por

$$c^{(k+1)} = \begin{cases} \beta c^{(k)} & \text{se } r_{adm.}^{(k)} < r_{\min}, \\ \frac{1}{\beta} c^{(k)} & \text{se } r_{adm.}^{(k)} < r_{\max} \\ c^{(k)} & \text{caso contrário} \end{cases} ,$$

onde $\beta > 1$, $r_{adm.}^{(k)}$ é a proporção de pontos admissíveis da população na iteração k, $r_{\min}$ e $r_{\max}$ é a proporção mínima e máxima de pontos admissíveis da população definidas pelo utilizador, respetivamente.

Um outro exemplo de uma função de penalidade adaptativa é proposta por M. Ali e

Zhu (2013), a qual usa o valor da função objetivo se o ponto da população é admissível, e combina a soma do valor das restrições violadas com o valor da função objetivo, ou com um limite superior para o mínimo global do COP, se o ponto é não admissível. Considerando que o problema contém apenas restrições de desigualdade, o valor da violação das restrições num dado ponto x_i da população é dada por

$$V_1(x_i) = \sum_{i=1}^{m} \max\{0, g_i(x_i)\}, \tag{3.1.4}$$

e a função de penalidade proposta é dada por

$$F(x_i, c) = \begin{cases} f(x_i) & \text{se} \quad V_1(x_i) = 0 \\ cV_1(x_i) + U_* & \text{se} \quad V_1(x_i) > 0 \text{ e } f(x_i) \leq U_* \\ cV_1(x_i) + f(x_i) & \text{se} \quad V_1(x_i) > 0 \text{ e } f(x_i) > U_* \end{cases},$$

onde $c > 0$ é um parâmetro de penalidade constante e U_* é um limite superior para o valor do mínimo global do problema. Este limite superior é definido inicialmente pelo utilizador, e vai sendo atualizado ao longo das iterações com o valor do melhor ponto admissível da população. Note-se que o valor inicial de U_* não é atualizado até ser encontrada uma solução admissível. M. Ali e Zhu (2013) mostraram nos seus estudos numéricos e teóricos que a função de penalidade proposta é menos sensível para valores baixos do parâmetro de penalidade c, pelo que os motivou a usar $c = 1$. Além disso, os autores demonstraram a equivalência entre um COP e o respetivo BCOP baseado na função de penalidade adaptativa proposta. Apresentaram ainda alguns resultados teóricos de convergência do método baseado na estrutura do algoritmo DE.

De modo a evitar a inicialização dos parâmetros de penalidade, ou qualquer outro tipo de parâmetro, surgiu na literatura as chamadas funções de penalidade auto-adaptativas, as quais são usadas em métodos baseados em populações.

Barbosa e Lemonge (2002) criaram uma nova função de penalidade auto-adaptativa designada de APM (do inglês, *Adaptive Penalty Method*), que usa informações da população ao longo do processo iterativo, tais como a média dos valores das soluções obtidas e o nível de violação de cada restrição. Neste método, em cada iteração k, cada ponto x_i da

população é avaliado segundo a função de penalidade definida por

$$F(x_i, c^{(k)}) = \begin{cases} f(x_i) & \text{se } x_i \text{ é admissível} \\ \bar{f}(x_i) + \sum_{i=1}^{m} c_i^{(k)} \max\{0, g_i(x_i)\} + \sum_{j=1}^{p} c_i^{(k)} |h_j(x_i)| & \text{caso contrário} \end{cases},$$

onde

$$\bar{f}(x_i) = \begin{cases} f(x_i) & \text{se } f(x_i) > \overline{f(x)} \\ \overline{f(x)} & \text{caso contrário} \end{cases}.$$

Em cada iteração k, $\overline{f(x)}$ representa a média dos valores da função objetivo da população e cada parâmetro de penalidade $c_i^{(k)}$ é definido por

$$c_i^{(k)} = |\overline{f(x)}| \frac{\overline{V_i(x)}}{\sum_{l=1}^{p+m} \overline{V_l(x)}^2}, i = 1, \ldots, p+m,$$

onde $\overline{V_i(x)}$ representa a média das violações da i-ésima restrição nos pontos da população corrente. Note-se que com esta fórmula de atualização para os parâmetros c_i, as restrições mais difíceis de serem verificadas são mais penalizadas do que as restantes. Desde a proposta inicial, várias variantes do APM foram propostas e aplicadas por diversos investigadores em diversas áreas do conhecimento (Zavislak, 2004; Gallet, Salaun, & Bouchet, 2005; Obadage & Harnpornchai, 2006; Rocha & Fernandes, 2009; Silva et al., 2011; Lemonge, Barbosa, & Bernardino, 2012).

Em Tessema e Yen (2006) e Tessema e Yen (2009), é proposta uma função de penalidade auto-adaptativa, em que o valor da função objetivo e a soma das violações das restrições são normalizadas e combinadas para definir a função de penalidade. Assim, considerando que o problema tem apenas restrições de desigualdade, para cada ponto x_i da população o valor da função objetivo é normalizado fazendo

$$\tilde{f}(x_i) = \frac{f(x_i) - f_{\min}}{f_{\max} - f_{\min}},$$

onde $f_{\min}$ e $f_{\max}$, são o menor e o maior valor da função objetivo de entre os pontos da população, respetivamente. A normalização da soma da violação das restrições em cada

ponto x_i não admissível é dada por

$$v(x_i) = \frac{1}{m}\sum_{i=1}^{m}\frac{g_i(x_i)}{g_{\max,\mathrm{i}}},$$

onde $g_{\max,\mathrm{i}}$ é o maior valor da violação da restrição i de entre todos os pontos da população e m é o número de restrições do problema. Nesta proposta, é também tida em conta a proporção de pontos admissíveis na população, $r_{adm.}$, que é dada por

$$r_{adm.} = \frac{\text{número de pontos admissíveis}}{\text{número de pontos da população}}.$$

Finalmente, o valor da função de penalidade em cada ponto x_i da população é dado por

$$F(x_i) = \begin{cases} v(x_i) & \text{se} \quad r_{adm.} = 0 \\ \tilde{f}(x_i) & \text{se} \quad r_{adm.} > 0 \text{ e } v(x_i) = 0 \\ \sqrt{\tilde{f}(x_i)^2 + v(x_i)^2} + \left[(1 - r_{adm.})v(x_i) + r_{adm.}\tilde{f}(x_i)\right] & \text{se} \quad r_{adm.} > 0 \text{ e } v(x_i) > 0 \end{cases}.$$

É de salientar que Tessema e Yen (2006, 2009) implementaram os algoritmos com a penalidade auto-adaptativa no contexto de um GA.

Como foi referido anteriormente, uma outra técnica de tratamento de restrições consiste em tratar a função objetivo e as respetivas restrições do problema separadamente. Este tipo de técnica é usado, em geral, em métodos baseados em populações. Um exemplo de um método pertencente a este grupo pode ser encontrado em (Hinterding & Michalewicz, 1998), onde os autores dividiram o processo de procura em duas fases. A primeira consiste em encontrar na população apenas soluções admissíveis. Depois de ter encontrado um número apropriado de soluções admissíveis, inicia-se uma segunda fase com o objetivo de encontrar o melhor valor da função objetivo das soluções admissíveis encontradas.

Em (B. Liu, Ma, Zhang, & Zhou, 2007), os autores propõem uma técnica co-evolutiva, na qual duas populações são consideradas. A primeira visa otimizar a função objetivo sem considerar as restrições, enquanto a segunda população visa apenas satisfazer as restrições do problema. Esta técnica tem a particularidade de permitir a migração de um elemento de uma população para outra. Às melhores soluções encontradas é aplicado um procedimento de procura local baseado em mutações gaussianas. No entanto, esta abordagem tem a

desvantagem de requerer valores diferentes para os parâmetros na resolução de alguns problemas, de entre um pequeno conjunto de problemas teste utilizado neste estudo.

Ainda nesta categoria, muitos autores consideram os conceitos da otimização multi-objetivo para resolver um COP. Uma revisão destas abordagens pode ser encontrada em (Mezura-Montes & Coello, 2008), cujos autores consideraram dois tipos de técnicas: *(i)* técnicas que transformam um COP num problema bi-objetivo, onde uma das funções é a função objetivo original e a outra função objetivo é dada pela violação das restrições; *(ii)* técnicas que transformam um COP num problema de otimização multi-objetivo, onde as funções objetivo a minimizar são: a função objetivo original e as funções de restrição.

Os métodos do tipo filtro, introduzidos por Fletcher e Leyffer (2002), têm como objetivo promover a convergência global sem a necessidade de usar uma função de penalidade, através do desenvolvimento de um algoritmo que não necessite de decisões difíceis, por parte do utilizador, no que respeita à escolha dos parâmetros de penalidade. Assim, é introduzido o conceito de um "filtro" que permite que um ponto seja aceite se ele reduz a função objetivo ou a função de violação das restrições. Nesta abordagem de filtro, considera-se o problema como bi-objetivo, onde há dois objetivos em competição: a minimização da função objetivo e a da função de violação das restrições. Um filtro é uma lista de pares (função objetivo, violação das restrições) de modo que nenhum par domina qualquer outro, definindo uma região proibida. Diz-se que um ponto (função objetivo, violação das restrições) é aceite para inclusão no filtro se não for dominado por qualquer ponto no filtro.

Existem outros tipos de abordagens que tratam a função objetivo e as restrições separadamente, na área da otimização global, donde se destacam: a ordenação estocástica (do inglês, *stochastic ranking*) (Runarsson & Yao, 2000), e o método baseado nas regras de admissibilidade e otimalidade, implementado por Deb (2000). Devido à sua simplicidade de implementação e aos bons resultados produzidos, estas técnicas de tratamento de restrições tornaram-se muito populares, onde o seu uso pode ser encontrado numa vasta gama de estudos e aplicações. Uma revisão de literatura sobre a aplicação destes dois métodos pode ser encontrada em (Mezura-Montes & Coello, 2011). Segue-se uma descrição mais detalhada destes dois métodos.

No contexto de um GA, Deb (2000) considerou apenas problemas com restrições de desigualdade, formulando a seguinte função de penalidade, onde cada elemento da população x_i é avaliado de acordo com

$$F(x_i) = \begin{cases} f(x_i) & \text{se } V_1(x_i) \leq 0 \\ f_{\max} + V_1(x_i) & \text{caso contrário} \end{cases},$$

onde a soma dos valores das restrições violadas, $V_1(x)$, é dada por (3.1.4) e $f_{\max}$ é o pior valor da função objetivo obtido de entre todos os pontos admissíveis da população. No caso de não haver pontos admissíveis, considera $f_{\max} = 0$. Depois, através da seleção binária por torneio (do inglês, *binary tournament selection*), Deb (2000) usa as seguintes regras, baseadas nos conceitos de admissibilidade e otimalidade, para ordenar os pontos da população comparando os elementos dois a dois:

(R_1) Qualquer ponto admissível é preferível a um não admissível;

(R_2) Entre dois pontos admissíveis, é escolhido aquele que tiver menor valor da função objetivo (no caso de um problema de minimização);

(R_3) Entre dois pontos não admissíveis, é escolhido aquele que tiver menor valor da soma das violações das restrições.

Nesta abordagem não é necessário nenhum parâmetro de penalidade, uma vez que o procedimento de seleção apenas realiza comparações entre pares, baseadas no valor da função objetivo f e no valor da violação das restrições. É de referir que, o uso da violação das restrições nas comparações tem como objetivo direcionar as soluções não admissíveis para a região admissível. Estas três simples regras, apesar de terem sido propostas há algum tempo, constituem um exemplo de uma técnica de tratamento de restrições que continua a ser bastante utilizada na literatura (Deb & Goel, 1999; Becerra & Coello, 2006; Mezura-Montes, Velázquez-Reyes, & Coello, 2006; Brest, Zumer, & Maucec, 2006; Zielinski, Wang, & Laur, 2008; Barkat Ullah, Sarker, & Cornforth, 2008; Mani & Patvardhan, 2009; Mezura-Montes & Palomeque-Ortiz, 2009; Elsayed, Sarker, & Essam, 2011). A sua popularidade

deve-se ao facto de poder ser usada no contexto de uma grande variedade de algoritmos, e não usar parâmetros. Uma possível desvantagem desta técnica apontada em (Mezura-Montes, Coello, & Tun-Morales, 2004) é a possibilidade de promover uma convergência prematura do algoritmo.

A ordenação estocástica, técnica que trata a função objetivo e as restrições de um dado problema separadamente, foi inicialmente proposta por Runarsson e Yao (2000). Este método visa encontrar um equilíbrio entre os valores da função objetivo em cada ponto da população, e os respetivos valores da violação das restrições. Este método foi construído de forma a evitar a existência de diversos parâmetros, tendo apenas um único parâmetro: a probabilidade P_f. Para cada ponto x_i da população, são atribuídas duas posições relativas: uma que diz respeito à posição do ponto atendendo apenas ao valor da função objetivo, e outra que diz respeito ao valor da violação das restrições:

$$V_2(x_i) = \sum_{k=1}^{m} max\{0, g_k(x_i)\} + \sum_{j=1}^{p} |h_j(x_i)|. \tag{3.1.5}$$

A função de penalidade proposta inicialmente pelos autores depende do parâmetro probabilidade P_f, que traduz esse equilíbrio para a ordenação de um ponto x_i atendendo ao valor da função objetivo e ao valor da violação das restrições. Esta função é definida por

$$\Phi(x, P_f) = P_f \frac{I_f - 1}{N - 1} + (1 - P_f) \frac{I_{V_2} - 1}{N - 1}, \tag{3.1.6}$$

onde I_f e I_{V_2} são as posições do ponto x_i depois de ordenado de forma ascendente, tendo em conta os N pontos da população, baseadas nos valores das funções f e V_2, respetivamente. P_f é a probabilidade da função de penalidade ser calculada baseada na ordenação dos pontos segundo a função objetivo. De acordo com os autores, esta probabilidade deve tomar um valor no intervalo $0 < P_f < 0.5$, de modo a garantir que seja encontrada uma solução admissível. Atendendo à definição da função de penalidade, o melhor ponto da população é o ponto que tem o menor valor nessa função. Uma desvantagem apontada a esta técnica é a necessidade de usar valores diferentes para P_f na resolução de alguns problemas. A ordenação estocástica foi aplicada e modificada em muitos métodos estocásticos de otimização, tais como, por exemplo, em (Fonseca, Capriles, Barbc, & Lemonge, 2007;

Zhang, Luo, & Wang, 2008; Fan, Liu, Sorensen, & Wang, 2009; Geng, Song, Jiao, & Sun, 2009; R. Liu, Li, Zhang, & Jiao, 2009; Rocha, Fernandes, Fernandes, & Martins, 2010).

3.2 Restrições de integralidade

Uma das técnicas usuais para tratar as restrições de integralidade de um problema de MINLP não convexo e não suave, é a técnica de discretização. Neste tipo de abordagem, o problema discreto é transformado num problema contínuo relaxado (Burer & Letchford, 2012). As soluções obtidas a partir do problema contínuo relaxado são então discretizadas para que sejam satisfeitas as restrições de integralidade do problema original. Exemplos de aplicação desta técnica podem ser encontrados em (Bacanin, Brajevic, & Tuba, 2013; Baghlani, Makiabadi, & Sarcheshmehpour, 2014). Na sua forma mais simples, a ideia geral desta técnica passa por resolver o problema assumindo que todas as variáveis são contínuas, e a solução obtida para o problema contínuo relaxado é arredondada para o valor discreto mais próximo inferiormente ou superiormente. Neste processo de discretização, geralmente são usadas regras heurísticas para auxiliar esta decisão, usando operadores que transformam um valor real num valor inteiro. No caso em que as variáveis discretas assumem valores binários, esta conversão recorre geralmente a funções sigmóides (ou funções de transferência) com contradomínio [0,1], isto é, convertem os valores obtidos para as variáveis relaxadas em valores entre 0 e 1. Estas funções são então usadas para definir regras heurísticas de como atribuir os valores binários às respetivas variáveis binárias. Exemplos da aplicação desta técnica de discretização usando algoritmos estocásticos baseados em populações para resolver problemas de otimização do tipo binário podem ser encontrados em (Kennedy & Eberhart, 1997; Mirjalili & Lewis, 2013; Pampara et al., 2006; Engelbrecht & Pampara, 2007; Padberg, 2012; L. Wang et al., 2013; Azad et al., 2013; Kashan et al., 2012; T. Liu et al., 2013; Pampara & Engelbrecht, 2011). Neste tipo de problemas, um outro procedimento bastante simples de discretização do valor de uma coordenada contínua relaxada num valor 0 ou 1, é feito recorrendo à função chão (do inglês, *floor function*), que faz o arredondamento para o número inteiro mais próximo inferiormente, e é descrito

em (Sevkli & Guner, 2006). Neste procedimento, cada coordenada $s \in I_d$ de um ponto x_i, $x_{i,s} \in \mathbb{R}$, é transformado num valor 0 ou 1, designado por $b_{i,s}$, da seguinte forma:

$$b_{i,s} = \lfloor |x_{i,s} \,(\mathrm{mod}\, 2)| \rfloor \text{ , para todo } s \in I_d \text{ ,} \tag{3.2.1}$$

onde $\lfloor z \rfloor$ representa a função chão em z. Neste caso em particular, o valor da coordenada $b_{i,s}$ é o valor absoluto do maior número inteiro menor que o resto da divisão inteira de $x_{i,s}$ por 2.

Em (Crawford et al., 2014) é proposta uma versão binária do FA para resolver problemas de cobertura (do inglês, *covering problems*). No algoritmo proposto, em cada iteração k, é aplicada a equação (2.1.3) para determinar a nova posição $x_i^{(k+1)}$ de cada pirilampo i, $i = 1, \ldots, N$. O uso desta equação para o movimento torna muito provavelmente os valores das coordenadas de $x_i^{(k+1)}$ em valores contínuos. Para passar do espaço de procura contínuo para o espaço de procura binário foram propostas as seguintes três regras heurísticas definidas à custa de funções de transferência T

$$b_{i,s}^{(k+1)} = \begin{cases} 1 & \text{se } rand \leq T(x_{i,s}^{(k+1)}) \\ 0 & \text{caso contrário} \end{cases} \text{ , para todo } \ s \in I_d, \tag{3.2.2}$$

$$b_{i,s}^{(k+1)} = \begin{cases} b_{i,s}^{(k)^{-1}} & \text{se } rand \leq T(x_{i,s}^{(k+1)}) \\ b_{i,s}^{(k)} & \text{caso contrário} \end{cases} \text{ , para todo } \ s \in I_d, \tag{3.2.3}$$

$$b_{i,s}^{(k+1)} = \begin{cases} b_{\mathrm{best},s} & \text{se } rand \leq T(x_{i,s}^{(k+1)}) \\ 0 & \text{caso contrário} \end{cases} \text{ , para todo } s \in I_d, \tag{3.2.4}$$

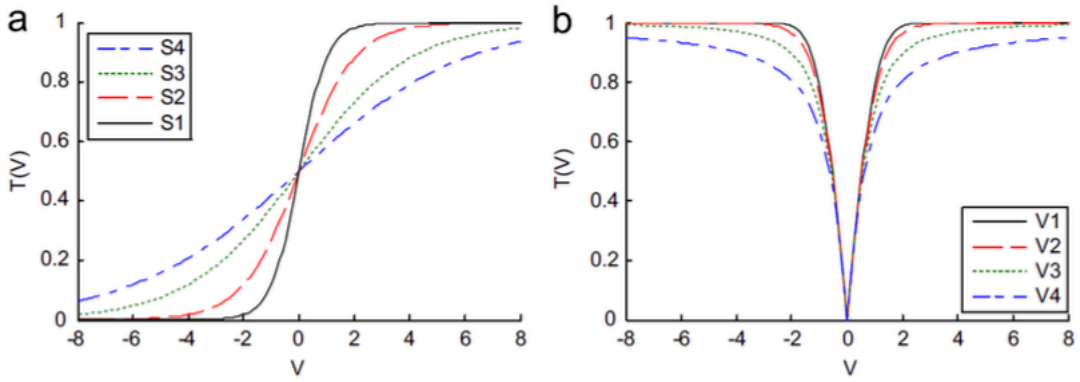
onde $rand \sim U[0,1]$, $b_i^{(k)^{-1}}$ é o complementar de $b_i^{(k)}$ e b_{best} é a posição do melhor pirilampo encontrado até então. As funções de transferência T obrigam os valores das coordenadas dos vetores posição dos pirilampos a moverem-se em espaço binário. De acordo com a forma das funções de transferência, estas são classificadas de funções de transferência de forma-S e funções de transferência de forma-V, definidas na tabela seguinte:

Tabela 3.1: Funções de transferência

forma-S	**forma-V**
S_1: $T(x) = \frac{1}{1+e^{-2x}}$	V_1: $T(x) = \lvert erf(\frac{\sqrt{2}}{\pi}x)\rvert = \lvert\frac{\sqrt{2}}{\pi}\int_0^{\frac{\sqrt{2}}{\pi}} e^{-t^2}dt\rvert$
S_2: $T(x) = \frac{1}{1+e^{-x}}$	V_2: $T(x) = \lvert tanh(x)\rvert$
S_3: $T(x) = \frac{1}{1+e^{-\frac{x}{2}}}$	V_3: $T(x) = \lvert\frac{x}{\sqrt{1+x^2}}\rvert$
S_4: $T(x) = \frac{1}{1+e^{-\frac{x}{3}}}$	V_4: $T(x) = \frac{2}{\pi}arctan(\frac{\pi}{2}x)$

A figura que se segue foi retirada de (Crawford et al., 2014), e mostra os gráficos destas funções.

Figura 3.1: Gráficos das funções de transferência forma-S e forma-V



Os resultados apresentados em (Crawford et al., 2014) demonstraram que as regras heurísticas (3.2.2), (3.2.3) e (3.2.4) convergem mais rapidamente quando aplicadas com as funções de transferência de forma-S.

Em (Palit, Sinha, Molla, Khanra, & Kule, 2011), os autores apresentam uma adaptação binária do FA para resolver um problema de encriptação de mensagens. Falcon, Almeida, e Nayak (2011), propuseram também uma adaptação do FA semelhante ao descrito por Palit et al. (2011) para resolver um problema do tipo binário de identificação de falhas em sistemas paralelos e distribuídos. Nestes trabalhos, a adaptação do FA para resolver o problema binário foi feita recorrendo à regra heurística 3.2.2 baseada na função de transfe-

rência de forma-S, dada por S_2. Um outro exemplo de uma adaptação do FA para resolver um problema de portfólio binário pode ser encontrado em (Bacanin & Tuba, 2014). Neste estudo, após o movimento de um pirilampo $x_i^{(k)}$ em direção a um pirilampo mais brilhante $x_j^{(k)}$, na iteração k, a regra heurística utilizada é dada por

$$x_{i,s}^{(k+1)} = round\left(\frac{1}{1+e^{-x_{i,s}^{(k)}+rand(x_{i,s}^{(k)}-x_{j,s}^{(k)})}} - 0.06\right), \text{para todo } \ s \in I_d,$$

onde $x_{i,s}^{(k)}$ e $x_{j,s}^{(k)}$ são a s-ésima coordenada de $x_i^{(k)}$ e $x_j^{(k)}$, respetivamente, $rand \sim U[0,1]$, e round$(\cdot)$ significa o arredondamento ao número inteiro das respetivas coordenadas s.

Recentemente, outra técnica que tem sido aplicada para o tratamento de restrições de integralidade em problemas de MINLP envolve o uso de funções de penalidades exatas (Lucidi & Rinaldi, 2010; C. Yu, Teo, & Bai, 2013; Lucidi & Rinaldi, 2013; Murray & Ng, 2010; Shandiz & Mahdavi-Amiri, 2011). Num método de penalidade exata, o problema original é transformado num problema contínuo relaxado equivalente. Existem na literatura várias propostas de diversas transformações destas. Em geral, a transformação envolve a relaxação das restrições de integralidade de $x_j \in \mathbb{Z}$, $j \in I_d$, para $x_j \in \mathbb{R}$, para todo $j \in I_d$, e na adição de um termo de penalidade à função objetivo. Giannessi e Tardella (1999), estenderam o método de penalidade exata para resolver problemas de MINLP genéricos. Rinaldi (2009), propôs vários termos de penalidade para resolver problemas de MINLP binários côncavos, que foram generalizados posteriormente por Lucidi e Rinaldi (2010), onde mostraram que uma classe de funções de penalidade, incluindo a que é proposta por Giannessi e Tardella (1999), pode ser usada para resolver problemas de MINLP genéricos. Lucidi e Rinaldi (2010) mostraram que o problema (1.2.10) é equivalente ao problema contínuo relaxado

$$\begin{aligned} \min \quad & f(x) + P(x;\varepsilon) \\ \text{sujeito a} \quad & x \in X \\ & 0 \leq x \leq e, \end{aligned} \tag{3.2.5}$$

onde $\epsilon > 0$, $P(x;\varepsilon)$ é um termo de penalidade escolhido adequadamente, e $e \in \mathbb{R}^n$ é um vetor de uns.

Considerando as seguintes hipóteses da função f e da penalidade $P(.;\epsilon)$, mostra-se que os problemas (1.2.10) e (3.2.5) são equivalentes.

Hipótese 1. *A função f é limitada em X e existe um conjunto aberto $A \supset W$ e números reais positivos p e L, tais que $\forall x, y \in A$, f satisfaz a seguinte condição*

$$|f(x) - f(y)| \leq L\|x - y\|^p. \tag{3.2.6}$$

Hipótese 2. $\forall x, y \in W$ *e* $\forall \varepsilon \in \mathbb{R}^+$,

$$P(x;\varepsilon) = P(y;\varepsilon). \tag{3.2.7}$$

Hipótese 3. *Existe um valor $\bar{\varepsilon}$, e $\forall z \in W$ existe uma vizinhança $S(z)$ tal que*

$$P(x;\varepsilon) - P(z;\varepsilon) \geq \bar{L}\|x - z\|^p, \ \ \forall x \in S(z) \cap (X \backslash W) \ e \ \varepsilon \in]0, \bar{\varepsilon}], \tag{3.2.8}$$

onde $\bar{L} > L$ e p é escolhido como em (3.2.6)*. Além disso, se $S = \bigcup_{z \in W} S(z)$, $\exists \bar{x} \notin S$ tal que*

$$\lim_{\varepsilon \to 0} (P(\bar{x};\varepsilon) - P(z;\varepsilon)) = +\infty \ \ , \ \ \forall z \in W, \tag{3.2.9}$$

e

$$P(x;\varepsilon) \geq P(\bar{x};\varepsilon), \ \ \forall x \in X \backslash S \ \ , \ \ \forall \varepsilon > 0. \tag{3.2.10}$$

Teorema 3.2.1. *(Theorem 2.1 in (Lucidi & Rinaldi, 2010)) Suponha-se que as Hipóteses 1, 2 e 3 são satisfeitas. Sejam W e X ($W \subseteq X \subset \mathbb{R}^n$) conjuntos compactos e $\|\cdot\|$ uma norma apropriada. Então, $\exists \bar{\varepsilon} \in \mathbb{R}^+$ tal que, $\forall \varepsilon \in]0, \bar{\varepsilon}]$, os problemas (1.2.10) e (3.2.5) têm os mesmos minimizantes globais.*

Demonstração: Ver (Lucidi & Rinaldi, 2010). □

Em (Rinaldi, 2009), a equivalência entre os problemas (1.2.10) e (3.2.5) é demonstrada quando o termo de penalidade é dado por

$$P(x;\varepsilon) = \frac{1}{\epsilon} \sum_{i \in I_d} x_i(1 - x_i). \tag{3.2.11}$$

Usando o Teorema 3.2.1, Lucidi e Rinaldi (2010) provaram a equivalência entre os problemas (1.2.10) e (3.2.5) para uma classe mais geral de termos de penalidade, incluindo (3.2.11). Os termos de penalidade considerados são dados por

$$P(x;\varepsilon) = \sum_{i\in I_d}\{\log(x_i+\epsilon) + \log[(1-x_i)+\epsilon]\}, \tag{3.2.12}$$

$$P(x;\varepsilon) = \sum_{i\in I_d}\{-(x_i+\epsilon)^{-p} - [(1-x_i)+\epsilon)]^{-p}\}, \tag{3.2.13}$$

$$P(x;\varepsilon) = \frac{1}{\epsilon}\sum_{i\in I_d}\{[1-\exp(-\alpha.x_i)] + [1-\exp(-\alpha.(1-x_i))]\}, \tag{3.2.14}$$

$$P(x;\varepsilon) = \frac{1}{\epsilon}\sum_{i\in I_d}\{(x_i+\epsilon)^q + [(1-x_i)+\epsilon]^p\}, \tag{3.2.15}$$

$$P(x;\varepsilon) = \frac{1}{\epsilon}\sum_{i\in I_d}\{[1+\exp(-\alpha.x_i)]^{-1} + [1+\exp(-\alpha.(1-x_i))]^{-1}\}, \tag{3.2.16}$$

onde $\epsilon, \alpha, p > 0$ e $0 < q < 1$.

Proposição 1. *Para todo o termo de penalidade (3.2.12)-(3.2.16), existe um valor* $\overline{\epsilon} > 0$ *tal que, para qualquer* $\epsilon \in]0, \overline{\epsilon}]$*, os problemas (1.2.10) e (3.2.5) têm os mesmos minimizantes globais.*

Demonstração: Ver em (Lucidi & Rinaldi, 2010). □

Embora qualquer problema de MINLP com restrições de limites simples e variáveis inteiras, (1.2.8), possa ser reformulado como um problema de programação binário, esta estratégia pode ser problemática e não é recomendada na prática. É preferível usar termos de penalidade específicos para penalizar a violação das restrições de variáveis inteiras.

Em (Lucidi & Rinaldi, 2010) é demonstrada a equivalência entre o problema (1.2.8) e o problema contínuo relaxado

$$\begin{aligned} \min \quad & f(x) + P(x;\varepsilon) \\ \text{sujeito a} \quad & x \in X \subset \mathbb{R}^n, \end{aligned} \tag{3.2.17}$$

onde o termo de penalidade é dado por

$$P(x;\varepsilon) = \sum_{i\in I_d} \min_{\substack{l_i\leq d_j\leq u_i \\ d_j\in\mathbb{Z}}} \{\log[|x_i - d_j| + \epsilon]\}. \tag{3.2.18}$$

Proposição 2. *Para o termo de penalidade (3.2.18), existe um valor $\overline{\epsilon} > 0$ tal que, para qualquer $\epsilon \in]0, \overline{\epsilon}]$, os problemas (1.2.8) e (3.2.17) têm os mesmos minimizantes globais.*

Demonstração: Ver em (Lucidi & Rinaldi, 2010). □

É ainda de referir que em (Lucidi & Rinaldi, 2010, 2013) a equivalência entre os problemas (1.2.8) e (3.2.17) é também estabelecida usando outros termos de penalidade, que podem ser considerados simples adaptações dos descritos para problemas do tipo binário. Por exemplo, o termo de penalidade (3.2.18) pode ser substituído por um dos seguintes termos

$$P(x;\varepsilon) = \frac{1}{\epsilon}\sum_{i\in I_d} \min_{\substack{l_i\leq d_j\leq u_i \\ d_j\in\mathbb{Z}}} \left\{[|x_i - d_j| + \epsilon]^p\right\}, 0 < p < 1 \tag{3.2.19}$$

$$P(x;\varepsilon) = \sum_{i\in I_d} \min_{\substack{l_i\leq d_j\leq u_i \\ d_j\in\mathbb{Z}}} \left\{\min\{-[x_i - d_j + \epsilon]^{-p}, -[d_j - x_i + \epsilon]^{-p}\}\right\}, p > 0 \tag{3.2.20}$$

$$P(x;\epsilon) \ = \sum_{i\in I_d} \min_{\substack{l_i\leq d_j\leq u_i \\ d_j\in\mathbb{Z}}} \left\{-\left[|x_i - d_j| + \epsilon\right]^{-q}\right\}, q > 0 \tag{3.2.21}$$

$$P(x;\epsilon) = \frac{1}{\epsilon}\sum_{i\in I_d} \min_{\substack{l_i\leq d_j\leq u_i \\ d_j\in\mathbb{Z}}} \left\{1 - \exp\left(-\alpha|x_i - d_j|\right)\right\}, \alpha > 0 \tag{3.2.22}$$

$$P(x;\epsilon) = \frac{1}{\epsilon}\sum_{i\in I_d} \min_{\substack{l_i\leq d_j\leq u_i \\ d_j\in\mathbb{Z}}} \left\{\left[1 + \exp\left(-\alpha|x_i - d_j|\right)\right]^{-1}\right\}, \alpha > 0. \tag{3.2.23}$$

Shandiz e Mahdavi-Amiri (2011) também provaram a equivalência entre o problema de MINLP (1.2.8), para o caso particular $\{0,1\}^{n_d}$, e o problema contínuo relaxado (3.2.17). Considere-se a seguinte notação

$$P(x;\epsilon) = rq(x) = r\sum_{i\in I_d} q_i(x_i),$$

com $r = \frac{1}{\epsilon}$. Esta prova é estabelecida assumindo que cada q_i, $i \in I_d$, é diferenciável em $J_\epsilon =]0, \epsilon[\cup]1-\epsilon, 1[$, com $0 < \epsilon < \frac{1}{2}$, $\gamma > 0$ e

$$q_i'(x_i) = \begin{cases} > \gamma & \text{se } x_i < \epsilon \\ < -\gamma & \text{se } x_i > 1-\epsilon \end{cases}, \text{para todo } \ i \in I_d.$$

Apresenta-se em seguida os termos de penalidade sugeridos em (Fu, Fenton, & Cleghorn, 1991; Lucidi & Rinaldi, 2010)), para os quais se provou que verificam a hipótese dada:

$$q_i(x_i) = 4x_i(1-x_i),$$

$$q_i(x_i) = \begin{cases} 2x_i & \text{se } x_i \leq \frac{1}{2} \\ 2(1-x_i) & \text{se } x_i > \frac{1}{2} \end{cases},$$

$$q_i(x_i) = \log(x_i+\theta) + \log(1-x_i+\theta) - \log\theta - \log(1+\theta),$$

$$q_i(x_i) = -(x_i+\theta)^{-p} - [(1-x_i)+\theta]^{-p} + \theta^{-p} + (1+\theta)^{-p},$$

$$q_i(x_i) = 1 - \exp(-\alpha x_i) + 1 - \exp(-\alpha(1-x_i)) - 1 - \exp(-\alpha),$$

$$q_i(x_i) = x_i^q + (1-x_i)^q - 1,$$

$$q_i(x_i) = [1+\exp(-\alpha x_i)]^{-1} + [1+\exp(-\alpha(1-x_i))]^{-1} - \frac{1}{2} - 1 - \exp(-\alpha)^{-1},$$

$$q_i(x_i) = \sin(\pi x_i),$$

$$q_i(x_i) = x_i(1-x_i)(x_i+\tau)(1-x_i+\tau),$$

onde $\alpha, p, \tau > 0$, $0 < \theta < \frac{1}{2}$, $0 < q < 1$ e $x_i \in \{0,1\}$, para todo $i \in I_d$.

Em (Shandiz & Mahdavi-Amiri, 2011) é ainda estabelecida a equivalência entre o problema (1.2.11), para o caso particular $\{0,1\}^{n_d}$, e o problema contínuo relaxado dado por

$$\begin{aligned} \text{minimizar} \quad & f(x) + \mu V_1(x) + rq(x) \\ \text{sujeito a} \quad & x \in X \subset \mathbb{R}^n \\ & x_i \in \mathbb{R} \text{ para } i \in I_c \subseteq I \equiv \{1, \ldots, n\} \\ & x_j \in \{0,1\} \text{ para } j \in I_d \subseteq I \\ & I_c \cap I_d = \emptyset \text{ e } I_c \cup I_d = I, \end{aligned} \tag{3.2.24}$$

onde μ e r são parâmetros de penalidade finitos, e V_1 é o termo de penalidade dado por (3.1.4). Neste estudo é ainda demonstrado que os problemas (1.2.11), para o caso particular $\{0,1\}^{n_d}$, e (3.2.24) são equivalentes.

Para finalizar, é de referir que em (Francisco, Costa, Rocha, & Fernandes, 2016) fez-se um estudo experimental comparativo entre as funções de penalidade definidas pelos termos (3.2.18), (3.2.19), (3.2.21), (3.2.22), (3.2.23), e um novo termo de penalidade baseado na função `erf` sigmóide. A função `erf` sigmóide é baseada na *função erro*. A *função erro* é uma função positiva cujo gráfico é uma curva em forma de sino, e aparece frequentemente no ramo da Probabilidade e Estatística estando relacionada com a distribuição Normal. Quando uma série de medições é descrita por uma distribuição Normal com média 0 e desvio padrão σ, a função `erf`, avaliada em $\frac{x}{\sigma\sqrt{2}}$, para um dado valor positivo x, dá a probabilidade de que o erro de uma única medição esteja no intervalo $[-x, x]$. Esta função `erf` é definida por

$$\texttt{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-x^2)\, dx, \tag{3.2.25}$$

que satisfaz as seguintes propriedades:

$$\texttt{erf}(0) = 0,\ \texttt{erf}(-\infty) = -1,\ \texttt{erf}(+\infty) = 1, \texttt{erf}(-x) = -\texttt{erf}(x).$$

Assim, o termo de penalidade proposto neste estudo é dado por

$$P(x;\epsilon) = \sum_{i \in I_d} \min_{\substack{l_i \leq d_j \leq u_i \\ d_j \in \mathbb{Z}}} \left\{ \texttt{erf}\left(|x_i - d_j| + \epsilon \right) \right\}. \tag{3.2.26}$$

Neste estudo compararativo foi usado um algoritmo de penalidade exata semelhante ao apresentado por Lucidi e Rinaldi (2010), e os subproblemas contínuos (3.2.17) foram resolvidos globalmente usando o FA com (4.2.8). Para mais detalhes sobre o o algoritmo de penalidade exata e comparação das soluções produzidas pelas várias penalidades em estudo, consultar (Francisco, Costa, Rocha, & Fernandes, 2016).

Capítulo 4

Extensão do FA para problemas de NLP com restrições genéricas

Neste capítulo são propostos algoritmos relativos à extensão do FA para resolver problemas de NLP com restrições genéricas, não convexos e não suaves. Estes problemas surgem frequentemente em muitas aplicações nas diversas áreas de conhecimento e, dadas as características do tipo de funções envolvidas, os métodos estocásticos que não utilizam derivadas, tais como o FA, são os mais apropriados para resolvê-los (Tessema & Yen, 2006; Horng & Liou, 2011; Silva et al., 2011; B. Wang et al., 2016). Nas secções que se seguem são apresentados dois estudos onde o FA é estendido e adaptado de modo a incorporar técnicas para o tratamento de restrições dos problemas (ver Secção 3.1).

Na Secção 4.1, apresenta-se a extensão do FA baseado em esquemas de ordenação. Neste contexto é proposta uma ordenação estocástica, com uma nova função de penalidade que trata a função objetivo e as restrições do problema separadamente, e outra definida por regras de admissibilidade e otimalidade. De modo a melhorar a convergência do FA, são usadas novas formas de atualização dos parâmetros α e γ ao longo do processo iterativo. Para encontrar soluções com uma maior precisão, é implementado um procedimento de procura local no final de cada iteração. Este trabalho encontra-se publicado em (Francisco, Costa, & Rocha, 2016).

Na Secção 4.2, é proposta uma função de penalidade auto-adaptativa que não depende de parâmetros definidos pelo utilizador, em que os parâmetros de penalidade são atualizados, em cada iteração, pelas informações recolhidas a partir da função objetivo e da violação das restrições em cada ponto da população. É estabelecida uma equivalência entre um problema de NLP com restrições genéricas e um problema de NLP com restrições de limites simples baseado na função de penalidade proposta. Os subproblemas BCOP definidos pela função de penalidade auto-adaptativa são resolvidos por um FA híbrido. A análise de convergência estocástica do algoritmo de penalidade, baseado na função de penalidade auto-adaptativa, é também estabelecida. Este trabalho encontra-se publicado em (Costa, Francisco, Rocha, & Fernandes, 2016).

4.1 Extensão do FA baseado em esquemas de ordenação

Nesta secção é apresentada uma extensão do FA para resolver problemas de otimização global não suaves e não convexos com restrições genéricas. A formulação matemática dos problemas que se pretendem resolver são do tipo (1.2.3), onde a região admissível Ω é dada por (1.2.4). Para resolver este tipo de problemas, são propostas duas técnicas para o tratamento das restrições, uma baseada na ordenação estocástica e outra nas regras de admissibilidade e otimalidade, que foram apresentadas na Secção 3.1.

4.1.1 Técnicas para o tratamento das restrições propostas

A ordenação estocástica, também conhecida na literatura por ordenação global competitiva (OGC), foi inicialmente proposta por Runarsson e Yao (2000) e apresentada na Secção 3.1. Como referido, uma desvantagem associada a esta técnica é a necessidade de usar valores diferentes para o parâmetro P_f para resolver com sucesso alguns problemas. Para ultrapasar esta desvantagem, e usando um processo de ordenação semelhante, é proposta uma nova função de penalidade que não depende do parâmetro de probabilidade P_f. A função de penalidade proposta tem em consideração o número de restrições violadas

em cada ponto x_i, $nc(x_i)$, para $i = 1, \ldots, N$. Assim, na aplicação da OGC, após calcular $f(x_i)$, $V_2(x_i)$ dado por (3.1.5) e $nc(x_i)$, para todos os pontos x_i da população, estes são ordenados separadamente por ordem crescente de $f(x_i)$ e $V_2(x_i)$, $i = 1, \ldots, N$. Tendo em conta esta ordenação, a função de penalidade proposta para avaliar cada ponto x_i da população, $i = 1, \ldots, N$, é dada por

$$F(x_i) = \frac{I_{i,f} - 1}{N(N-1)} + nc(x_i)\frac{I_{i,V_2} - 1}{N(N-1)}, \tag{4.1.1}$$

onde $I_{i,f}$ e I_{i,V_2}, são as posições do ponto x_i baseadas nos valores de f e de V_2, respetivamente. Por fim, os N pontos da população são ordenados segundo os valores de ordem crescente da função de penalidade F. Note-se que a função penalidade proposta, penaliza os pontos que violem o maior número de restrições. A descrição do esquema de ordenação dos pontos baseado na OGC usando (4.1.1) apresenta-se no Algoritmo 2.

Algoritmo 2 Ordenação da população de acordo com a OGC

Dados: I_f, I_{V_2} e nc

Para $i = 1$ *até* $N - 1$ **fazer**

  Para $j = i + 1$ *até* N **fazer**

    Se $F(x_i) > F(x_j)$ **então**

      Trocar a posição de x_i com x_j

    fim

  fim

fim

A outra técnica proposta para o tratamento das restrições baseia-se nas regras de admissibilidade e otimalidade R_1-R_3 (Deb, 2000), considerando uma nova regra adicional, R_4, que tem em conta o número de restrições violadas, nc:

(R_4) Entre dois pontos não admissíveis, é escolhido aquele que viola o menor número de restrições.

Este conjunto de regras de admissibilidade e otimalidade $R_1 - R_4$ são designadas pelo acrónimo RAO. Assim, quando dois pontos da população são comparados para ver qual

deles é melhor que o outro, são usadas as RAO como critério para a sua ordenação. Estas regras são estabelecidas na seguinte definição:

Definição 4.1.1. *Sejam $x, y \in \Omega$. O ponto y é melhor que x se as seguintes condições são verificadas:*

$$((V_2(x) > V_2(y) \;\; ou \;\; nc(x) > nc(y)) \;\; ou \;\; (V_2(x) = V_2(y) = 0 \;\; e \;\; f(x) > f(y)).$$

Após calcular o valor da função objetivo $f(x_i)$, o valor da violação $V_2(x_i)$ e o número de restrições violadas $nc(x_i)$, para todos os pontos da população x_i, $i = 1, ..., N$, os pontos são ordenados segundo as RAO. Apresenta-se no Algoritmo 3 o esquema de ordenação dos pontos baseado nas RAO.

De seguida, usando as RAO $R_1 - R_4$, os N pontos da população são ordenados através da comparação entre todos os pares de pontos. A forma como é efetuada a ordenação dos pontos, segundo o esquema de ordenação RAO, está descrita no Algoritmo 3.

Algoritmo 3 Ordenação da população de acordo com as RAO

Dados: V_2 e nc

Para $i = 1$ *até* $N - 1$ **fazer**
  Para $j = i + 1$ *até* N **fazer**
    Se x_j é melhor que x_i **então**
      Trocar a posição de x_i com x_j
    fim
  fim
fim

Os esquemas de ordenação baseados na OGC e RAO garantem que as boas soluções admissíveis, bem como as soluções promissoras não admissíveis, sejam colocadas no topo da ordenação dos pontos da população.

4.1.2 O algoritmo EXFA

Nesta secção, é apresentada a extensão do FA, denotada por EXFA, com a implementação da OGC usando (4.1.1) e das RAO usando R_1-R_4. O objetivo da implementação

destas técnicas visa explorar de modo mais eficiente as regiões admissíveis e não admissíveis dos problemas, privilegiando os pontos admissíveis, ou que violem menos restrições, em detrimento dos restantes.

Tal como no FA, a EXFA aqui proposta, usa uma população de pontos/pirilampos para calcular, em cada iteração k, uma solução aproximada $x_{best}^{(k)}$ do problema (1.2.3). O movimento de cada ponto x_i da população, $i = 1, \ldots, N$, em cada iteração k, em direção ao pirilampo mais brilhante x_j (sabendo que x_j é melhor que o ponto x_i), é dado por

$$x_i^{(k+1)} = x_i^{(k)} + \beta(x_j^{(k)} - x_i^{(k)}) + \alpha(\epsilon_i - 0.5)S, \tag{4.1.2}$$

onde o parâmetro da atratividade β é calculado segundo (2.1.1), ϵ_i é um vetor de números aleatórios gerado de uma distribuição Uniforme em $[0, 1]$, α é o parâmetro que controla a aleatoriedade e S é o parâmetro de escalonamento do problema definido em (2.1.4). Após o movimento de x_i, se $x_i \notin X$, então o ponto é projetado em X. Para melhorar a convergência da EXFA, é proposta uma fórmula de atualização dinâmica para calcular o vetor S, de modo a diminuir ao longo das iterações k, dada por

$$S^{(k)} = \frac{|(lb - ub) - (x_N^{(k)} - x_1^{(k)})|}{k}, \tag{4.1.3}$$

onde $x_N^{(k)} - x_1^{(k)}$ é o vetor dado pela diferença entre as posições do pior e do melhor ponto da população. Além disso, para melhorar a velocidade de convergência do FA e a qualidade das soluções, foram propostas atualizações dinâmicas para os parâmetros α e γ de acordo com (5.1.1) e (5.1.2), respetivamente. Estas fórmulas de atualização dependem do número de iteração do algoritmo e permitem reduzir a aleatoriedade dos parâmetros.

É de salientar que, após a movimentação de um pirilampo i em direção a um pirilampo mais brilhante j, o brilho do pirilampo i pode diminuir. Para evitar que isto ocorra, após o movimento de um pirilampo i em direção a um pirilampo j, o pirilampo i toma uma posição temporária denotada por t_i e aplicam-se as regras estabelecidas na Definição 4.1.1. Isto é, se t_i for melhor que $x_i^{(k)}$ (segundo a Definição 4.1.1), t_i será a nova posição do pirilampo i no próximo movimento; caso contrário, a posição de $x_i^{(k)}$ será mantida para o movimento seguinte. Para melhorar o desempenho da EXFA, todos os pirilampos movem-se de acordo

com a equação (4.1.2), exceto o pirilampo menos brilhante, $x_N^{(k)}$. A posição do pirilampo N é substituída pela posição gerada por um movimento aleatório em torno do pirilampo mais brilhante. Por fim, para aumentar a precisão da melhor solução encontrada é aplicado um procedimento estocástico de procura local no fim de cada iteração k. A procura local, como descrita em (Birbil & Fang, 2003), tem como objetivo explorar, coordenada a coordenada, o espaço de procura numa vizinhança do melhor pirilampo, $x_{\text{best}}^{(k)}$, da população. Inicialmente, para um dado parâmetro fixo δ, é calculado o comprimento máximo do passo admissível $\Delta = \delta(\max_{1\leq s\leq n}(ub_s - lb_s))$. Considere-se y a posição do melhor ponto da população. Para cada coordenada s, $s = 1, \ldots, n$, é gerado um número aleatório $\mu \sim U[0, 1]$ para definir o comprimento do passo, e é calculada uma nova posição para y, coordenada a coordenada, ao longo dessa direção, através de $y_s = x_{\text{best,s}}^{(k)} + \mu\Delta$. Se $y \notin \Omega$, então y é rejeitado, e a procura ao longo dessa coordenada termina; se y é melhor que $x_{\text{best}}^{(k)}$, de acordo com a Definição 4.1.1, o melhor ponto $x_{\text{best}}^{(k)}$ é substituído por y, e a procura ao longo dessa coordenada s continua até atingir um número máximo de iterações $LSit_{max}$. A descrição formal deste procedimento de procura local é apresentado no Algoritmo 4.

Os principais passos do algoritmo da EXFA são descritos no Algoritmo 5.

Os resultados obtidos nas experiências computacionais realizadas e a sua comparação com outros métodos estocásticos, igualmente baseados em populações, são apresentados na Secção 6.1.2.

4.2 Extensão do FA baseado baseada numa função de penalidade auto-adaptativa

Nesta secção pretende-se ilustrar o comportamento da técnica de penalidade baseada numa função de penalidade auto-adaptativa para resolver problemas de NLP com restrições genéricas, não convexos e não suaves. Em particular, pretende-se analisar o desempenho do FA quando combinado com a técnica de penalidade auto-adaptativa. A contribuição deste estudo pretende ir além da proposta de uma função de penalidade auto-adaptativa no

Algoritmo 4 Procedimento de procura local

Dados: x_{best} (melhor ponto da população na iteração k), $LSit_{max}$, δ

$\Delta = \delta(\max_{1\leq s\leq n}(ub_s - lb_s))$

Para $s = 1$ **até** n **fazer**

  Fazer it=1

  Enquanto $it < LSit_{max}$ **fazer**

    Fazer $y = x_{\text{best}}$

    $y_s = x_{\text{best},s} + \mu\Delta$

    Se y é melhor que x_{best} e $y \in \Omega$ **então**

      Fazer $x_{\text{best}} = y$

    Senão

      Fazer $it = LSit_{max} - 1$

    fim

    Fazer $it = it + 1$

  fim

fim

contexto do FA. Em primeiro lugar, prova-se que os problemas COP e BCOP, baseados na função de penalidade auto-adaptativa proposta, são equivalentes no sentido de que ambos têm o mesmo minimizante global. Além isso, propõe-se a implementação da função de penalidade auto-adaptativa num conjunto selecionado de metaheurísticas que se encontram descritas na Secção 2.3: o jDE (Brest, Greiner, et al., 2006), o algoritmo PSO (Kennedy & Eberhart, 1995; M. Ali & Kaelo, 2008), o CMA-ES (Hansen & Ostermeier, 2001), o algoritmo ABC (Karaboga & Basturk, 2007) e o FA. É também abordada a questão relacionada com a adequação do cálculo dos parâmetros que são necessários para construir a função de penalidade auto-adaptativa, quando implementada em métodos baseados em populações de pontos. Em segundo lugar, propõe-se um algoritmo híbrido da função de penalidade auto-adaptativa no contexto do FA, que usa um procedimento de intensificação local para explorar o espaço de procura na vizinhança de cada ponto da população. Por fim, é feita a análise de convergência do algoritmo que tem em consideração a estrutura do

Algoritmo 5 Algoritmo EXFA

Dados: $k_{\max}$, $\alpha_{\max}$, $\alpha_{\min}$, $\gamma_{\max}$, $\gamma_{\min}$, N

Fazer $k = 1$

Gerar aleatoriamente uma população de N pirilampos, $x_i^{(k)} \in X, i = 1, ..., N$

Calcular $f(x_i^{(k)})$, $V_2(x_i^{(k)})$ e $nc(x_i^{(k)})$ para $i = 1, ..., N$

Ordenar por ordem crescente os pirilampos pelo Algoritmo 2 (OGC) ou Algoritmo 3 (RAO)

Fazer $x_{\text{best}}^{(k)} = x_1^{(k)}$ e $f_{\text{best}}^{(k)} = f(x_1^{(k)})$

Enquanto (critério de paragem não é satisfeito) **fazer**

  Calcular $\alpha^{(k)}$ usando (5.1.1)

  Calcular $S^{(k)}$ usando (4.1.3)

  Para $i = 2$ **até** $N - 1$ **fazer**

    Para $j = 1$ **até** $i - 1$ **fazer**

      Calcular a atratividade β usando (2.1.1) e (5.1.2)

      Mover o pirilampo i em direção ao pirilampo j, usando (4.1.2), e projetar em X (se necessário), para a nova posição temporária t_i

      Avaliar $f(t_i), V_2(t_i)$ e $nc(t_i)$

      Se t_i é melhor que $x_i^{(k)}$ (de acordo com a Definição 4.1.1) **então**

        Fazer $x_i^{(k+1)} = t_i$

      fim

    fim

  fim

  Fazer $t_N = x_1^{(k+1)} + \epsilon$, com $\epsilon \sim U[0, 1]$, e projetar em X

  Avaliar $f(t_N), V_2(t_N)$ e $nc(t_N)$

  Se t_N é melhor que $x_N^{(k)}$ (de acordo com a Definição 4.1.1) **então**

    Fazer $x_N^{(k+1)} = t_N$

  fim

  Fazer $k = k + 1$

  Avaliar $f(x_i^{(k)})$, $V_2(x_i^{(k)})$ e $nc(x_i^{(k)})$ para $i = 1, ..., N$

  Ordenar por ordem crescente os pirilampos, de acordo com o Algoritmo 2 ou o Algoritmo 3

  Calcular $F(x_i^{(k)})$ para $i = 1, ..., N$

  Fazer $x_{\text{best}}^{(k)} = x_1^{(k)}$ e $F_{\text{best}}^{(k)} = F(x_1^{(k)})$

  Implementar Algoritmo 4 de procura local

fim

Saída: Solução: $x_1^{(k)}$ e $F(x_1^{(k)})$.

FA e as propriedades da função de penalidade auto-adaptativa proposta.

4.2.1 A função de penalidade auto-adaptativa

O problema (1.2.5) pode ser reformulado como um BCOP com uma função de penalidade que combina a função objetivo f e a violação das restrições. O problema (1.2.5) é equivalente a

$$\begin{aligned} \text{minimizar} \quad & \phi(x) \\ \text{sujeito a} \quad & x \in X \subset \mathbb{R}^n, \end{aligned} \tag{4.2.1}$$

no sentido de que têm as mesmas soluções, desde que a função de penalidade ϕ satisfaça algumas propriedades (M. Ali & Zhu, 2013).

Neste estudo, o principal objetivo é definir uma função de penalidade auto-adaptativa, no sentido de que os pesos da violação das restrições, também considerados como valores dos parâmetros de penalidade, não sejam definidos pelo utilizador, mas calculados usando informações recolhidas a partir do valor da violação das restrições em cada ponto da população. Para tal, os valores da função objetivo e da violação das restrições são normalizados, tendo em conta valores de referência da função objetivo e da violação das restrições, encontrados no espaço de procura do problema. Assim, considerando os parâmetros

$$f^{\min} = \min_{x \in X} f(x) \ \text{ e } \ f^{\max} = \max_{x \in X} f(x),$$

o valor da função objetivo f em cada ponto x da população é normalizado segundo a função de avaliação F definida por:

$$F(x) = \frac{f(x) - f^{\min}}{f^{\max} - f^{\min}}. \tag{4.2.2}$$

Para um ponto do espaço de procura $x \in X$, a violação das restrições é dada por $V_1(x)$ definida em (3.1.4), a qual tem o valor 0 quando $x \in \Omega$ e um valor positivo quando $x \notin \Omega$, sendo $\Omega = \{x \in X : g_j(x) \leq 0, \ \ j = 1, \ldots, m\}$. Para que a medida da violação das restrições tenha a mesma ordem de grandeza da função F, a violação de cada restrição é normalizada usando a expressão:

$$v_j(x) = \frac{\max\{g_j(x), 0\}}{g_j^{\max}}, \ \ \text{onde} \ \ g_j^{\max} = \max_{x \in X \setminus \Omega} \{\max\{g_j(x), 0\}\} \tag{4.2.3}$$

é o maior valor da violação da restrição j calculada em todos os pontos $x \in X \setminus \Omega$, sendo o subconjunto $X \setminus \Omega$ o complementar de Ω em X. Assim, propõe-se a seguinte função de penalidade auto-adaptativa:

$$\phi(x) = \begin{cases} F(x) & \text{se } x \in \Omega \\ F(z) + \dfrac{1}{m}\displaystyle\sum_{j=1}^{m} v_j(x) r_j & \text{se } x \in X \setminus \Omega \text{ e } f(x) \leq f(z) \\ F(x) + \dfrac{1}{m}\displaystyle\sum_{j=1}^{m} v_j(x) r_j & \text{se } x \in X \setminus \Omega \text{ e } f(x) > f(z) \end{cases}, \tag{4.2.4}$$

em que $z \in \Omega$ é um ponto fixo tal que $f(z) \geq f^*$, e cada peso r_j é definido pela proporção de pontos do espaço de procura X que violam a restrição g_j:

$$r_j = \frac{|x \in X : g_j(x) > 0|}{|X|}, \ j = 1, \ldots, m. \tag{4.2.5}$$

De seguida prova-se que os problemas (1.2.5) e (4.2.1) são equivalentes.

Teorema 4.2.1. *Seja $x^* \in \Omega$ o minimizante global do problema* (1.2.5) *e seja $z \in \Omega$ tal que $f(z) \geq f(x^*)$. Então, x^* é o minimizante global do problema* (4.2.1)*, onde ϕ é a função de penalidade definida em* (4.2.4)*.*

Demonstração: Seja $x^* \in \Omega$ o minimizante global do problema (1.2.5). Por definição, tem-se que $f(x^*) \leq f(x)$, $\forall x \in \Omega$. Assim, $\forall x \in \Omega$,

$$\phi(x^*) = \frac{f(x^*) - f^{\min}}{f^{\max} - f^{\min}} \leq \frac{f(x) - f^{\min}}{f^{\max} - f^{\min}} = \phi(x).$$

Considere-se o caso em que $x \in X \setminus \Omega$. Tem-se dois casos a considerar:

(a) Assumindo que $f(x) \leq f(z)$, tem-se que:

$$\phi(x^*) = F(x^*) \leq F(z) < F(z) + \frac{1}{m}\sum_{j=1}^{m} v_j(x) r_j = \phi(x),$$

uma vez que v_j e r_j são ambos positivos por definição.

(b) Assumindo que $f(x) > f(z)$ tem-se que:

$$\phi(x^*) = F(x^*) \leq F(z) < F(x) < F(x) + \frac{1}{m}\sum_{j=1}^{m} v_j(x) r_j = \phi(x)$$

e, por conseguinte, $\phi(x^*) \leq \phi(x)$, $\forall x \in X$, i.e., x^* é um minimizante global do problema (4.2.1). □

Lema 4.2.1. *Se x^* é o minimizante global do problema* (4.2.1)*, onde ϕ é a função de penalidade definida em* (4.2.4)*, então x^* é um ponto admissível do problema* (1.2.5)*.*

Demonstração: A demonstração será feita por contradição, assumindo que $x^* \in X \setminus \Omega$. Tem-se dois casos a considerar:
(a) Assumindo que $f(x^*) \leq f(z)$ e $z \in \Omega$ tem-se que, de acordo com a função (4.2.4),

$$\phi(x^*) = F(z) + \frac{1}{m}\sum_{j=1}^{m} v_j(x^*)r_j > F(z) = \phi(z).$$

(b) Assumindo que $f(x^*) > f(z)$ e $z \in \Omega$ tem-se que, de acordo com a função (4.2.4),

$$\phi(x^*) = F(x^*) + \frac{1}{m}\sum_{j=1}^{m} v_j(x^*)r_j > F(x^*) > F(z) = \phi(z),$$

que contradiz a definição de minimizante global do problema (4.2.1). Logo, $x^* \in \Omega$. □

De seguida é estabelecido o recíproco do Teorema 4.2.1.

Teorema 4.2.2. *Seja $x^* \in X$ um minimizante global do problema* (4.2.1)*, onde ϕ é a função de penalidade definida em* (4.2.4)*. Então, x^* é um minimizante global do problema* (1.2.5)*.*

Demonstração: Pelo Lema 4.2.1, $x^* \in \Omega \subset X$. Para todo o $x \in X$, tem-se que

$$F(x^*) = \phi(x^*) \leq \phi(x)$$

e, em particular, para todo o $x \in \Omega$, tem-se que

$$F(x^*) \leq F(x),$$

que implica que $f(x^*) \leq f(x)$. Consequentemente, x^* é um minimizante global do problema (1.2.5). □

A função de penalidade auto-adaptativa tem como objetivo penalizar os pontos que violam as restrições de desigualdade do problema (1.2.5), enquanto que as restrições de limites simples são sempre satisfeitas ao resolver o problema (4.2.1). De acordo com os

Teoremas 4.2.1 e 4.2.2 é suficiente encontrar a solução global do BCOP (4.2.1), ou seja, um minimizante global de $\phi(x)$ em X.

Por fim, é necessário mostrar como adequar o cálculo dos parâmetros $f^{\min}$, $f^{\max}$, $f(z)$, $g_j^{\max}$ e r_j, $j = 1, \ldots, m$, apresentados em (4.2.2), (4.2.3), (4.2.4) e (4.2.5), a um método que usa uma população de pontos em cada iteração.

Seja $X^{(k)} = \{x_1^{(k)}, \ldots, x_N^{(k)}\}$ uma população de N pontos numa dada iteração k. Para normalizar a função de avaliação F, tal como está definida em (4.2.2), em cada ponto x da população, são necessários os seguintes parâmetros:

$$f^{\min} = \min_{x \in X^{(k)}} f(x) \quad \text{e} \quad f^{\max} = \max_{x \in X^{(k)}} f(x),$$

onde $F(x) = 0$ é o ponto com menor valor da função objetivo e $F(x) = 1$ é o ponto com maior valor da função objetivo. Para normalizar a violação da restrição j, o parâmetro $g_j^{\max}$ é definido como sendo o maior valor da violação obtida pela restrição j para todos os pontos da população em $X^{(k)}$:

$$g_j^{\max} = \max_{x \in X^{(k)}} \left\{ \max\{g_j(x), 0\} \right\}.$$

O ponto de referência z é o ponto admissível com o menor valor da função objetivo encontrado até então. Note-se que, se a população não tiver pontos admissíveis, o valor inicial atribuído a $f(z)$ é temporariamente $f^{\max}$, e assim $f(x) \leq f(z)$, para todo o $x \in X^{(k)}$ e $F(z) = 1$ (para definir a função de penalidade tal como em (4.2.4)). O valor de $f(z)$ é atualizado apenas quando é encontrado o primeiro ponto admissível. Em cada iteração k, o conjunto de N pontos temporários gerados é representado por $T^{(k)} := \{t_1^{(k)}, \ldots, t_N^{(k)}\}$. Se a população temporária gerada $T^{(k)}$ contém pontos admissíveis, aquele que apresentar o menor valor da função objetivo, ou seja $f(t_i^{(k)})$, é comparado com $f(z)$ e define-se $f(z) = f(t_i^{(k)})$ se $f(t_i^{(k)}) < f(z)$; caso contrário, $f(z)$ não é atualizado. Da mesma forma, $f(z)$ é mantido para a iteração seguinte se não houver pontos admissíveis na população temporária. Finalmente, cada peso/parâmetro de penalidade r_j é calculado iterativamente como sendo a proporção de pontos da população que violam a restrição g_j:

$$r_j = \frac{\left| x \in X^{(k)} : g_j(x) > 0 \right|}{m}, \; j = 1, \ldots, m.$$

É de salientar que uma restrição que é violada por um conjunto maior de pontos da população do que qualquer outra terá um peso maior.

4.2.2 O algoritmo FPAA

Nesta secção, propõe-se o algoritmo que implementa a função de penalidade auto-adaptativa quando é usada a metaheurística FA para calcular a solução do BCOP (4.2.1). O procedimento de intensificação local implementado é baseado em operadores de mutação evolutivos típicos do DE (Storn & Price, 1997).

O desempenho do FA, tal como o de qualquer outra metaheurística, depende das capacidades de diversificação e intensificação do algoritmo. A diversificação tem como objetivo explorar o espaço de procura de modo a encontrar soluções globais. A intensificação visa explorar localmente uma região do espaço de procura identificada como promissora. O procedimento de intensificação local encontra-se descrito no Algoritmo 6. Primeiro, uma estratégia de mutação é aplicada coordenada a coordenada ao melhor pirilampo, com probabilidade p_m, para criar o melhor ponto mutante, u_1:

$$u_1 = x_1 + F_b \left(x_{i_1} - x_{i_2} \right), \tag{4.2.6}$$

onde i_1 e i_2 são dois índices aleatoriamente selecionados do conjunto $\{2, \ldots, N\}$, e $F_b > 0$ é um parâmetro real constante, cujo objetivo é controlar a dimensão da variação diferencial. Se as coordenadas do melhor ponto mutante não respeitarem as restrições de limites simples, então é projetado em X. O procedimento de seleção entre u_1 e x_1 é levado a cabo para escolher o melhor dos dois pontos. Este procedimento de seleção dá prioridade aos pontos que têm uma redução significativa da violação das restrições e está descrito no Algoritmo 7. Em segundo lugar, a estratégia de mutação do DE, comummente designada de DE/best/1, é aplicada aos restantes pontos da população, coordenada a coordenada, com probabilidade p_m, da seguinte forma:

$$u_i = x_1 + F_o \left(x_{i_1} - x_{i_2} \right), \tag{4.2.7}$$

para $i = 2, \ldots, N$, onde x_1 é a posição do melhor pirilampo encontrado até então, F_o é

o parâmetro de controlo da variação, e i_1 e i_2 são índices aleatoriamente escolhidos do conjunto $\{1, \ldots, i-1, i+1, \ldots, N\}$.

Algoritmo 6 Procedimento de intensificação local

Dados: n, N, $(x_i, f(x_i), V_1(x_i))$ para $i = 1, \ldots, N$, p_m, F_b, F_o, X

Para $s = 1, \ldots, n$ **fazer**

  Se $\xi \sim U[0,1] < p_m$ **então**

    Fazer $u_{1,s} = x_{1,s} + F_b(x_{i_1,s} - x_{i_2,s})$

  Senão

    Fazer $u_{1,s} = x_{1,s}$

  fim

fim;

Fazer $t_1 = u_1$ projetado em X

Avaliar $f(t_1)$, $V_1(t_1)$ e usar procedimento de seleção para identificar x_1 (ver Algoritmo 7)

Para $i = 2, \ldots, N$ **fazer**

  Para $s = 1, \ldots, n$ **fazer**

    Se $\xi \sim U[0,1] < p_m$ **então**

      Fazer $u_{i,s} = x_{1,s} + F_o(x_{i_1,s} - x_{i_2,s})$

    Senão

      Fazer $u_{i,s} = x_{i,s}$

    fim

  fim;

  Fazer $t_i = u_i$ projetado em X

  Avaliar $f(t_i)$, $V_1(t_i)$ e usar procedimento de seleção para identificar x_i (ver Algoritmo 7)

fim;

Saída: $x_i^{k+1} \leftarrow x_i, i = 1, \ldots, N$.

O Algoritmo 8 apresenta os passos do algoritmo FA híbrido com a função de penalidade auto-adaptativa, denotado por FPAA. Inicialmente, a posição de cada pirilampo i é gerada aleatoriamnte em X, $x_s^i = lb_s + \xi(ub_s - lb_s)$ para $s = 1, \ldots, n$, onde $\xi \sim \mathrm{U}[0,1]$. O movimento de um pirilampo i que é atraído para outro pirilampo mais brilhante j é dado por

$$x_i^{(k+1)} = x_i^{(k)} + \beta_0 \exp\left(-\gamma \|x_i^{(k)} - x_j^{(k)}\|^{\nu}\right)(x_j^{(k)} - x_i^{(k)}) + \alpha\, \xi^i, \tag{4.2.8}$$

onde $0 \leq \alpha \leq 1$ é o parâmetro de controlo da aleatoriedade, $\xi_s^i \sim \mathrm{U}[-1,1], s = 1, \ldots, n$, $\nu \geq 1$ e β_0 é o parâmetro de atração quando a distância é nula (X.-S. Yang, 2009; X. Yang,

Algoritmo 7 Procedimento de seleção

Dados: tol_v, $x_i, f(x_i), V_1(x_i)$, $t_i, f(t_i), V_1(t_i)$.

Se $V_1(t_i) \leq tol_v$ **então**

  Se $f(t_i) < f(x_i)$ **ou** $V_1(x_i) > tol_v$ **então**
    Fazer $x_i = t_i$

  fim;

Senão

  Se $V_1(t_i) < V_1(x_i)$ **então**
    Fazer $x_i = t_i$

  fim;

fim

2010; X.-S. Yang & He, 2013). O parâmetro $\gamma \in [0, \infty[$ carateriza a variação da atratividade e é crucial para determinar a velocidade de convergência do algoritmo (Arora & Singh, 2013). Neste estudo são usadas as ideias apresentadas em (Costa, Rocha, Francisco, & Fernandes, 2014), em que os parâmetros α e γ são reduzidos com o número de iteração, k, do algoritmo e dados por (5.1.1) e (5.1.2), respetivamente. O termo de aleatoriedade pode ser estendido para outras distribuições (S. Yu, Yang, & Su, 2013), nomeadamente para a distribuição de Lévy, que é uma distribuição estável e de cauda longa. Em cada iteração k, depois de cada pirilampo i ser movido em direção aos pirilampos mais brilhantes, a posição provisória do pirilampo i será designada por $t_i^{(k)}$. Note-se que se a posição do pirilampo temporário $t_i^{(k)}$ não pertencer ao espaço de procura X, este será projetado em X. No final de cada iteração é aplicado o critério de seleção, i.e., os valores da função de penalidade em $x_i^{(k)}$ e $t_i^{(k)}$ são comparados para determinar a posição corrente do pirilampo i para a iteração seguinte. No contexto do problema (4.2.1), se $\phi(t_i^{(k)}) < \phi(x_i^{(k)})$ então $t_i^{(k)}$ será considerado na iteração $k+1$; caso contrário, a posição $x_i^{(k)}$ será mantida na iteração $k+1$. Note-se que para $t_i^{(k)}$ e $x_i^{(k)}$ serem comparáveis usando a função ϕ, os parâmetros $f^{\min}$, $f^{\max}$, $f(z)$, $g_j^{\max}$ e $r_j, j = 1, \ldots, m$, são calculados tendo em conta a população corrente e a população temporária.

Na Secção 6.1.3 apresentam-se as experiências computacionais realizadas com: a implementação da função de penalidade auto-adaptativa no contexto das metaheurísticas jDE,

Algoritmo 8 Algoritmo FPAA

Dados: $k_{\max}$, ϵ, η, N, f^*

Fazer $k = 1$; Gerar aleatoriamente $x_i^{(k)} \in X$, $i = 1, \ldots, N$;

Baseado em $\{x_1^{(k)}, \ldots, x_N^{(k)}\}$ avaliar $f^{\min}, f^{\max}$, $f(z)$ e $g_j^{\max}, r_j, j = 1, \ldots, m$;

Avaliar $\phi(x_i^{(k)})$ $(i = 1, \ldots, N)$ e ordenar os pirilampos (do menor para o maior valor de ϕ).

Enquanto *(*$(|f^* - f(x_1^{(k)})| > \epsilon$ *ou* $V_1(x_1^{(k)}) > \eta)$ *e* $k \leq k_{\max}$*)* **fazer**

  Para todos $x_i^{(k)}$ **tais que** $i = 1, \ldots, N$ **fazer**

    Faz $x_i = x_i^{(k)}$

      Para todos $x_j^{(k)}$ **tais que** $j = 1, \ldots, i-1$

        Calcular o parâmetro aleatório α usando (5.1.1)

        Calcular a atratividade β usando (2.1.1) e (5.1.2)

        Mover o pirilampo i em direção ao pirilampo j usando (4.2.8)

      fim;

      Fazer $t_i^{(k)} = x_i$ projetado em X

    fim;

  fim;

  Baseado em $\{x_1^{(k)}, \ldots, x_N^{(k)}, t_1^{(k)}, \ldots, t_N^{(k)}\}$ avaliar $f^{\min}, f^{\max}, f(z)$ e $g_j^{\max}, r_j, j = 1, \ldots, m$

  Avaliar $\phi(x_i^{(k)})$, $\phi(t_i^{(k)})$ $(i = 1, \ldots, N)$

  Para $i = 1, \ldots, N$ **fazer**

    Se $\phi(t_i^{(k)}) < \phi(x_i^{(k)})$ **então**

      Fazer $x_i^{(k+1)} = t_i^{(k)}$

    Senão

      Fazer $x_i^{(k+1)} = x_i^{(k)}$

    fim

  fim;

  Baseado em $\{x_1^{(k+1)}, \ldots, x_N^{(k+1)}\}$ avaliar $f^{\min}, f^{\max}, f(z)$ e $g_j^{\max}, r_j, j = 1, \ldots, m$

  Avaliar $\phi(x_i^{(k+1)})$ $(i = 1, \ldots, N)$ e ordenar os pirilampos (do menor para o maior valor de ϕ)

  Invocar o procedimento de intensificação local (ver Algoritmo 6)

  Baseado em $\{x_1^{(k+1)}, \ldots, x_N^{(k+1)}\}$ avaliar $f^{\min}, f^{\max}, f(z)$ e $g_j^{\max}, r_j, j = 1, \ldots, m$

  Avaliar $\phi(x_i^{(k+1)})(i = 1, \ldots, N)$ e ordenar os pirilampos (do menor para o maior valor de ϕ)

  Fazer $k = k + 1$

fim

Saída: Solução: $x_1^{(k)}$ e $F(x_1^{(k)})$.

PSO, CMA-ES, ABC e FA; e os resultados obtidos com a implementação da FPAA na resolução de vários conjuntos de problemas.

4.2.3 Análise de convergência

Para efetuar a análise de convergência do Algoritmo 8, é seguida a metodologia apresentada por M. Ali e Zhu (2013). Atendendo às propriedades do FA, e à forma como a função de penalidade ϕ está definida, é possível estabelecer os seguintes resultados.

Teorema 4.2.3. *Seja $X^{(k)}$ a população corrente com N pontos na iteração k, $T^{(k)} = \{t_1^{(k)}, \ldots, t_N^{(k)}\}$ o conjunto dos pontos temporários na iteração k, e $X^{(k+1)}$ a população dos pontos selecionados para a iteração seguinte $k+1$. Então, $f(z^{(k)}) \geq f(z^{(k+1)})$, onde $z^{(k)}$ é o ponto admissível com o menor valor da função objetivo da população $X^{(k)}$ e $z^{(k+1)}$ é o ponto admissível com o menor valor da função objetivo encontrado em $T^{(k)}$. Além disso, $\phi(z^{(k)}) \leq \phi(t_i^{(k)})$, para todos os pontos não admissíveis $t_i^{(k)} \in T^{(k)}$.*

Demonstração: Seja $z^{(k)}$ a melhor solução admissível de $X^{(k)}$. Obviamente $z^{(k)}$ nunca será substituído por nenhum ponto não admissível de $T^{(k)}$. Assuma-se agora que existe um ponto admissível $t_i^{(k)} \in T^{(k)}$ tal que $\phi(t_i^{(k)}) < \phi(z^{(k)})$. Então,

$$\phi(t_i^{(k)}) = F(t_i^{(k)}) \ < F(z^{(k)}) = \phi(z^{(k)}) \ \Rightarrow \ f(t_i^{(k)}) < f(z^{(k)}),$$

onde $f^{\min}$ e $f^{\max}$ (definidos por F) são selecionados do conjunto $X^{(k)} \cup T^{(k)}$. Concluí-se assim que $f(z^{(k)}) > f(t_i^{(k)}) \geq f(z^{(k+1)})$. No entanto, se o ponto admissível $t_i^{(k)} \in T^{(k)}$ não satisfaz a condição $\phi(t_i^{(k)}) < \phi(z^{(k)})$, então

$$\phi(t_i^{(k)}) = F(t_i^{(k)}) \ \geq F(z^{(k)}) = \phi(z^{(k)}) \ \Rightarrow \ f(t_i^{(k)}) \geq f(z^{(k)})$$

e $f(z^{(k+1)}) = f(z^{(k)})$. Em ambos os casos $f(z^{(k)}) \geq f(z^{(k+1)})$. Considere-se agora o caso em que $t_i^{(k)} \in T^{(k)}$ é não admissível. Analisam-se duas situações: (a) $f(t_i^{(k)}) \leq f(z^{(k)})$ e (b) $f(t_i^{(k)}) > f(z^{(k)})$. No caso (a), assume-se que

$$\phi(t_i^{(k)}) \ < \phi(z^{(k)}) \ \Rightarrow \ F(z^{(k)}) + \frac{1}{m}\sum_{j=1}^{m} v_j(t_i^{(k)}) r_j \ < \ F(z^{(k)}), \tag{4.2.9}$$

desde que $t_i^{(k)}$ seja não admissível e $f(t_i^{(k)}) \leq f(z^{(k)})$ (ver (4.2.4)). No entanto, a última condição em (4.2.9) é uma contradição porque o segundo termo do lado esquerdo da inequação é positivo. No caso (b), assumindo também que $\phi(t_i^{(k)}) < \phi(z^{(k)})$, tem-se que

$$F(t_i^{(k)}) + \frac{1}{m}\sum_{j=1}^{m} v_j(t_i^{(k)})r_j \quad < F(z^{(k)})$$

e, portanto,

$$\frac{1}{m}\sum_{j=1}^{m} v_j(t_i^{(k)})r_j \quad < F(z^{(k)}) - F(t_i^{(k)}) = \frac{f(z^{(k)}) - f(t_i^{(k)})}{f^{\max} - f^{\min}} < 0,$$

que é uma contradição. Assim, tem-se que $\phi(z^{(k)}) \leq \phi(t_i^{(k)})$ para todos os pontos não admissíveis $t_i^{(k)} \in T^{(k)}$. □

No teorema seguinte, prova-se que a sucessão $\{f(z^{(k)})\}$ converge e que o limite é o maior limite inferior, ou ínfimo, f^*.

Teorema 4.2.4. *Seja $z^{(k)}$ o ponto admissível com o menor valor da função objetivo obtido na iteração k. Então,*

$$\lim_{k\to\infty} f(z^{(k)}) = f^*.$$

Demonstração: Pelo Teorema 4.2.3, $\{f(z^{(k)})\}$ é uma sucessão monótona decrescente. Uma vez que f^* é o ínfimo da sucessão, então para todo o $\delta > 0$, $f^* + \delta$ não é um ínfimo da sucessão. Assim, existe $K = K(\delta) \in \mathbb{N}$, tal que

$$f^* - \delta < f^* \leq f(z^{(k)}) \leq f(z^{(K)}) < f^* + \delta \quad \text{para todo o } \; k \geq K,$$

o que significa que $f(z^{(k)}) \to f^*$ quando $k \to \infty$. □

No procedimento de seleção entre o ponto corrente e o ponto temporário, descrito no Algoritmo 8, os valores da função de penalidade $\phi(x_i^{(k)})$ e $\phi(t_i^{(k)})$ são comparados. Quando os pontos $x_i^{(k)}$ e $t_i^{(k)}$ são admissíveis, o ponto com menor valor de f é selecionado, uma vez que o ponto com menor valor de ϕ é o ponto com menor valor de F e, portanto, é o ponto com um menor valor de f (ver 4.2.4 e o facto dos parâmetros $f^{\min}$ e $f^{\max}$ serem calculados com base no conjunto $X^{(k)} \cup T^{(k)}$). Por outro lado, quando ambos os pontos $x_i^{(k)}$ e $t_i^{(k)}$

são não admissíveis, a seleção é determinada pelos valores da violação das restrições e pelo valor de F, combinados na função de penalidade ϕ.

Contudo, quando $x_i^{(k)}$ é admissível e $t_i^{(k)}$ é não admissível, pode ser determinada a probabilidade de que o ponto temporário $t_i^{(k)}$ é selecionado, quando comparado com $x_i^{(k)}$, como ponto corrente para a iteração seguinte $k+1$.

Teorema 4.2.5. *Seja $x_i^{(k)} \in X^{(k)}$, onde $X^{(k)}$ é a população corrente na iteração k, e $t_i^{(k)} \in T^{(k)}$, onde $T^{(k)}$ é o conjunto de pontos temporários na iteração k, tais que $x_i^{(k)}$ é admissível e $t_i^{(k)}$ é não admissível. Assuma-se que existe $0 < \bar{r} \leq 1$ tal que, eventualmente, $r_j \geq \bar{r}$ para $j = 1, \ldots, m$. Então, a probabilidade de selecionar $t_i^{(k)}$ em detrimento de $x_i^{(k)}$ tende para zero, i.e.,*

$$Pr\left[\frac{1}{m}\sum_{j=1}^{m} v_j(t_i^{(k)})r_j < F(x_i^{(k)}) - F(z^{(k)})\right] \to 0 \text{ para } r_j, \;\; j = 1, \ldots, m, \text{ que satisfaz } r_j \geq \bar{r}.$$

Demonstração: Assuma-se que $t_i^{(k)} \in T^{(k)}$ é quase sempre selecionado quando comparado com um ponto admissível $x_i^{(k)} \in X^{(k)}$, i.e., $\phi(t_i^{(k)}) < \phi(x_i^{(k)})$. Assim, tem-se dois casos a considerar:

(a) Se $f(t_i^{(k)}) \leq f(z^{(k)})$, tem-se que

$$\phi(t_i^{(k)}) = F(z^{(k)}) + \Sigma^n(t_i^{(k)}) < \phi(x_i^{(k)}) = F(x_i^{(k)})$$

$$\Rightarrow \; 0 < \Sigma^n(t_i^{(k)}) < F(x_i^{(k)}) - F(z^{(k)}) \leq 1, \tag{4.2.10}$$

onde, por simplicidade de escrita, $\Sigma^n(t_i^{(k)}) = \dfrac{1}{m}\displaystyle\sum_{j=1}^{m} v_j(t_i^{(k)})r_j > 0$.

(b) Se $f(t_i^{(k)}) > f(z^{(k)})$ tem-se que

$$\phi(t_i^{(k)}) = F(t_i^{(k)}) + \Sigma^n(t_i^{(k)}) < \phi(x_i^{(k)}) = F(x_i^{(k)})$$

$$\Rightarrow \; 0 < \Sigma^n(t_i^{(k)}) < F(x_i^{(k)}) - F(t_i^{(k)}) < F(x_i^{(k)}) - F(z^{(k)}) \leq 1. \tag{4.2.11}$$

Note-se que em ambas as condições (4.2.10) e (4.2.11), $f(x_i^{(k)}) - f(z^{(k)}) > 0$, desde que $x_i^{(k)} \neq z^{(k)}$.

Agora veja-se qual é a probabilidade de $\Sigma^n(T_i^{(k)}) < F(x_i^{(k)}) - F(z^{(k)})$ ser verificada. Assume-se que o ponto temporário $T_i^{(k)}$ é uma variável aleatória com realizações $t_i^{(k)}$ e que $\Sigma^n(T_i^{(k)})$ aumenta uniformemente para lá da admissibilidade. Como $F(x_i^{(k)}) - F(z^{(k)})$ é um número fixo no intervalo $]0, 1]$, a probabilidade

$$Pr\left[\Sigma^n(T_i^{(k)}) < F(x_i^{(k)}) - F(z^{(k)})\right] > 0$$

verifica-se para as condições (4.2.10) e (4.2.11). Quanto maior for o valor de $F(x_i^{(k)}) - F(z^{(k)})$, maior é a probabilidade. No entanto, esta probabilidade também depende de $\Sigma^n(T_i^{(k)})$. Por contradição, assume-se que existe $0 < \bar{r} \leq 1$ tal que

$$Pr\left[\frac{1}{m}\sum_{j=1}^{m} v_j(T_i^{(k)})r_j < F(x_i^{(k)}) - F(z^{(k)})\right] > 0$$

onde $r_j, \quad j = 1, \ldots, m$, satisfaz $r_j \geq \bar{r}$. Isto significa que

$$\frac{1}{m}\sum_{j=1}^{m} v_j(t_i^{(k)})r_j < F(x_i^{(k)}) - F(z^{(k)}) \text{ para } r_j \geq \bar{r}, \quad j = 1, \ldots, m. \tag{4.2.12}$$

Porém, existe certamente um valor $T_i^{(k)} = t_i^{(k)}$ tal que, para r_j, $j = 1, \ldots, m$, satisfaz

$$r_j \geq \bar{r} \text{ e } \frac{1}{m}\sum_{j=1}^{m} v_j(t_i^k)r_j \geq F(x_i^{(k)}) - F(z^{(k)}),$$

o que contradiz a condição (4.2.12). □

Considere-se agora o caso em que $x_i^{(k)}$ é não admissível e o ponto temporário $t_i^{(k)}$ é admissível, e analise-se a probabilidade do ponto $x_i^{(k)}$ ser selecionado, em detrimento de $t_i^{(k)}$, como ponto corrente para a população da iteração seguinte $k + 1$.

Teorema 4.2.6. *Seja $x_i^{(k)} \in X^{(k)}$, onde $X^{(k)}$ é a população corrente de N pontos na iteração k, e $t_i^{(k)} \in T^{(k)}$, onde $T^{(k)}$ é o conjunto dos pontos temporários na iteração k, tais que $x_i^{(k)}$ é não admissível e $t_i^{(k)}$ é admissível. Então, existe $0 < \bar{r} \leq 1$ tal que a probabilidade de selecionar $x_i^{(k)}$ em vez de $t_i^{(k)}$ é zero, i.e.,*

$$Pr\left[\frac{1}{m}\sum_{j=1}^{m} v_j(x_i^{(k)})r_j < F(t_i^{(k)}) - F(z^{(k)})\right] = 0 \text{ quando } r_j, \ j = 1, \ldots, m, \quad \text{satisfaz } r_j \geq \bar{r}.$$

Demonstração: Assuma-se que $x_i^{(k)} \in X^{(k)}$ é quase sempre selecionado quando comparado com o ponto admissível $t_i^{(k)} \in T^{(k)}$, ou seja, $\phi(x_i^{(k)}) < \phi(t_i^{(k)})$. Tem-se novamente dois casos a considerar:
(a) Se $f(x_i^{(k)}) \leq f(z^{(k)})$, obtém-se

$$\phi(x_i^{(k)}) = F(z) + \Sigma^n(x_i^{(k)}) \ < \phi(t_i^{(k)}) = F(t_i^{(k)}) \ \Rightarrow \Sigma^n(x_i^{(k)}) < F(t_i^{(k)}) - F(z^{(k)}).$$

(b) Se $f(x_i^{(k)}) > f(z^{(k)})$ obtém-se

$$\phi(x_i^{(k)}) = F(x_i^{(k)}) + \Sigma^n(x_i^{(k)}) < \phi(t_i^{(k)}) = F(t_i^{(k)})$$

$$\Rightarrow \Sigma^n(x_i^{(k)}) < F(t_i^{(k)}) - F(x_i^{(k)}) < F(t_i^{(k)}) - F(z^{(k)}).$$

Assumindo que o ponto temporário $T_i^{(k)}$ e $f(T_i^{(k)})$ são variáveis aleatórias com realizações $t_i^{(k)}$ e $f(t_i^{(k)})$, respetivamente, tem-se que $F(T_i^{(k)}) - F(z^{(k)})$ é limitada (pois $t_i^{(k)}$ é admissível, $f(t_i^{(k)})$ é limitada e $f(z)$ está fixo). Assim, existe um conjunto de valores $T_i^{(k)} = t_i^{(k)}$ tais que $\Sigma^n(x_i^{(k)}) < F(t_i^{(k)}) - F(z^{(k)})$ se verifica, o que significa que $Pr\left[\Sigma^n(x_i^{(k)}) < F(T_i^{(k)}) - F(z^{(k)})\right] > 0$. No entanto, existe certamente $0 < \bar{r} \leq 1$ tal que $\frac{1}{m}\sum_{j=1}^{m} v_j(x_i^{(k)})r_j > F(t_i^{(k)}) - F(z^{(k)})$ se verifica para r_j, $j = 1, \ldots, m$, que satisfaz $r_j \geq \bar{r}$, implicando que

$$Pr\left[\frac{1}{m}\sum_{j=1}^{m} v_j(x_i^{(k)})r_j < F(T_i^{(k)}) - F(z^{(k)})\right] = 0 \ \ para \ \ r_j \geq \bar{r}, \ j = 1, \ldots, m.$$

□

Os resultados obtidos nas diversas experiências computacionais realizadas com o algoritmo FPAA encontram-se descritas na Secção 6.1.3.

Capítulo 5

Extensão do FA para problemas de MINLP

Neste capítulo são propostos algoritmos relativos à extensão do FA para resolver problemas de MINLP não convexos e não suaves. Dado que o algoritmo FA foi originalmente concebido para resolver problemas de otimização contínuos com restrições de limites simples nas variáveis de decisão, é necessário proceder a um conjunto de modificações de modo a incluir o espaço discreto na sua exploração pela procura de uma solução ótima global.

Primeiro propõe-se a extensão do FA para resolver problemas de MINLP com restrições de limites simples nas suas variáveis, cujos valores assumem apenas valores binários, recorrendo à técnica de discretização. Assim, o objetivo da Secção 5.1 é o de descrever a extensão do FA baseado em heurísticas, designado pelo acrónimo HBFA (do inglês, *Heuristic-Based Firefly Algorithm*). Na HBFA é proposto um FA dinâmico, em que os parâmetros α e γ são calculados de uma forma dinâmica ao longo das iterações do algoritmo e um novo termo aleatório na equação que descreve o movimento dos pirilampos. Este trabalho encontra-se publicado em (Costa, Rocha, Francisco, & Fernandes, 2014).

Na Secção 5.2, propõe-se uma extensão do FA capaz de resolver problemas de MINLP com restrições de limites simples e variáveis inteiras. Para tal, é usada a técnica de penalidade apresentada na Secção 3.2, que se baseia numa formulação contínua de penali-

dade exata do problema de MINLP, que surge relaxando as restrições de integralidade e adicionando um termo de penalidade à função objetivo visando penalizar a violação das restrições de integralidade do problema original. São propostos dois termos de penalidade, um baseado na função tangente hiperbólica e outro na função inversa do seno hiperbólico. Demonstrar-se-á que ambas as penalidades são exatas quando usadas para definir o problema contínuo com penalidade, ou seja, ambos são equivalentes ao problema de MINLP original. Para aumentar a capacidade de exploração e diminuir a quantidade de parâmetros de controlo, é proposto um algoritmo de penalidade exata baseado num FA adaptativo para resolver os subproblemas de penalidade contínuos, que inclui uma nova proposta para descrever o movimento dos pirilampos. Este trabalho encontra-se publicado em (Costa, Rocha, Francisco, & Fernandes, 2016).

Finalmente, na Secção 5.3, é apresentada uma extensão do FA capaz de resolver problemas de MINLP com restrições genéricas e de integralidade. Dada a existência de variáveis inteiras e contínuas é proposta uma nova forma para a geração inicial dos pontos da população e para o movimento dos pirilampos. O tratamento das restrições genéricas é efetuado usando as RAO R_1-R_4. Este trabalho encontra-se no prelo para publicação em (Costa, Rocha, Francisco, & Fernandes, 2017).

5.1 Com restrições de limites simples e variáveis binárias

Nesta secção descrevem-se as HBFA desenvolvidas para resolver problemas de MINLP com restrições de limites simples, em que todas as variáveis são discretas e do tipo binário, cuja formulação matemática é a definida em (1.2.10), onde f é uma função contínua, não necessariamente diferenciável e convexa. Para que as HBFA sejam capazes de tratar as variáveis binárias, são analisadas três heurísticas para transformar os valores contínuos das variáveis em valores binários. Neste contexto, é usada a função sigmóide para converter um valor contínuo num valor do intervalo $[0, 1]$. São propostos três métodos distintos de

implementação das três regras heurísticas no âmbito de um FA dinâmico proposto para a resolução deste tipo de problemas.

5.1.1 FA dinâmico

No FA, o parâmetro α controla a aleatoriedade e, consequentemente, a diversidade das soluções obtidas, que permite uma exploração mais global do espaço de procura. O parâmetro γ influencia o grau de atratividade que se estabelece entre os pirilampos. Pequenos valores de α combinados com grandes valores de γ podem aumentar o número de iterações necessárias para que o algoritmo convirja para a solução ótima. Experiências realizadas mostram que o parâmetro α deve tomar valores grandes no início do processo iterativo, de modo a forçar o algoritmo a aumentar a diversidade das soluções, e pequenos valores de α combinados com pequenos valores de γ nas iterações finais aumentam a precisão das soluções, pois a procura da solução ótima está concentrada numa região promissora do espaço de procura (X.-S. Yang, 2008, 2013; X.-S. Yang & He, 2013; X.-S. Yang et al., 2012). Com o objetivo de melhorar a velocidade de convergência e a precisão das soluções obtidas, são propostas atualizações dinâmicas, em função do contador de iterações k, para os parâmetros α e γ. Assim, o parâmetro α diminui, em função de k, de um limite superior $\alpha_{\max}$ até um limite inferior $\alpha_{\min}$ da seguinte forma:

$$\alpha^{(k)} = \alpha_{\max} - k\frac{\alpha_{\max} - \alpha_{\min}}{k_{\max}}. \tag{5.1.1}$$

Para aumentar a atratividade entre os pirilampos, o parâmero γ diminui desde um limite superior $\gamma_{\max}$ até um limite inferior $\gamma_{\min}$ da seguinte forma:

$$\gamma^{(k)} = \gamma_{\max} \exp\left(\frac{k}{k_{\max}} \ln\left(\frac{\gamma_{\min}}{\gamma_{\max}}\right)\right), \tag{5.1.2}$$

onde $k_{\max}$ é o número máximo de iterações permitidas. Note-se que os parâmetros $\alpha_{\max}$, $\alpha_{\min}$, $\gamma_{\max}$ e $\gamma_{\min}$ são definidos pelo utilizador.

Neste FA dinâmico é também proposto um novo termo aleatório na equação que descreve o movimento dos pontos da população baseada na distribuição de Lévy. Assim, o

movimento de um pirilampo i em direção a um pirilampo mais brilhante j é dado por

$$x_i^{(k+1)} = x_i^{(k)} + y_i^{(k)} \quad \text{com} \quad y_i^{(k)} = \beta(x_j^{(k)} - x_i^{(k)}) + \alpha L(x_1,\ '1')\sigma_x^i, \tag{5.1.3}$$

onde $L(x_1,\ '1')$ é um vetor de números aleatórios gerados pela distribuição de Lévy centrada em x_1, que é a posição do pirilampo mais brilhante, $'1'$ é um vetor de uns de dimensão n, e σ_x^i é um vetor que representa uma variação à volta de x_1 dada por,

$$\sigma_x^i = \left(|x_i^1 - x_1^1|, \ldots, |x_i^n - x_1^n|\right)^T.$$

5.1.2 Técnica de discretização para variáveis binárias

No âmbito do FA, quando um pirilampo i se move em direção a um pirilampo j, é muito provável que as coordenadas que definem o vetor posição do pirilampo i passem de números binários para reais.

Para transformar um número real num valor binário pode ser usada a regra heurística (3.2.2) baseada na função logística sigmóide referenciada na Tabela 3.1 por S_2:

$$\text{sig}(x_{i,s}^{(k+1)}) = \frac{1}{1 + \exp\left(-x_{i,s}^{(k)}\right)}, \quad \text{para todo} \ \ s \in I_d. \tag{5.1.4}$$

Assim, com base nesta função logística sigmóide, os valores reais destas coordenadas são transformados em valores no intervalo $[0, 1]$, onde $x_{i,s}^{(k+1)}$, é a coordenada s do vetor posição de $x_i^{(k+1)}$ (do pirilampo i) depois do movimento.

Note-se que a função (5.1.4) interpreta os valores das coordenadas da nova posição de i, $x_i^{(k+1)}$, como um conjunto de probabilidades que é usado para lhes atribuir valores binários apropriados através da regra heurística:

$$b_{i,s}^{(k+1)} = \begin{cases} 1 & \text{se } rand \leq \text{sig}(x_{i,s}^{(k+1)}) \\ 0 & \text{caso contrário} \end{cases}, \text{para todo } s \in I_d, \tag{5.1.5}$$

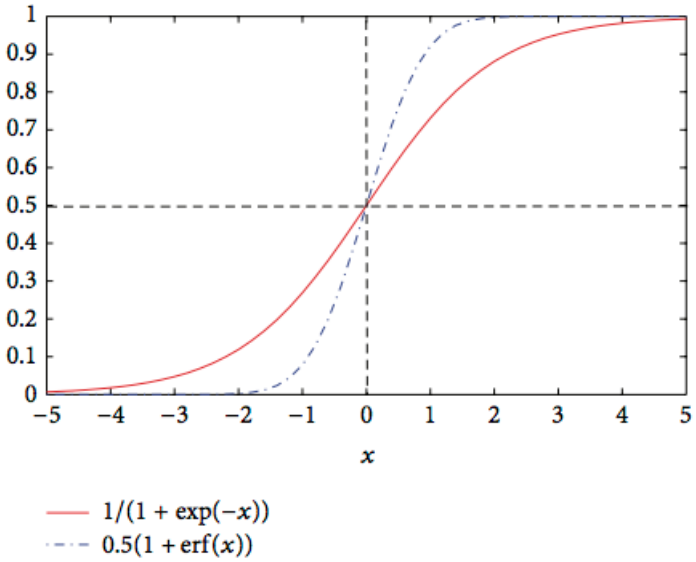
onde $\text{sig}(x_{i,s}^{(k+1)})$ dá a probabilidade da coordenada s da posição $x_i^{(k+1)}$ ser 0 ou 1 e $rand \sim U[0, 1]$. É de salientar que durante o processo iterativo, as posições dos pirilampos, x_i para

$i = 1, \ldots, N$, nunca são movidos para fora do espaço de procura X. Atendendo às boas propriedades da função `erf` (3.2.25), é proposta uma nova função sigmóide, a função `erf` sigmóide:

$$\text{sig}(x_{i,s}) = 0.5(1 + \texttt{erf}(x_{i,s})), \tag{5.1.6}$$

que é uma função diferenciável e limitada, e tem derivada positiva em todos os pontos $x \in \mathbb{R}$. Uma comparação entre as funções (5.1.4) e (5.1.6), é dada graficamente na Figura 5.1. Note-se que o declive na origem da função `erf` sigmóide (5.1.6) é aproximadamente 0.5641895, enquanto que na função logística sigmóide (5.1.4) é 0.25, tendo um crescimento mais rápido entre 0 e 1.

Figura 5.1: Representação gráfica das funções sigmóides (5.1.4) e (5.1.6)



Para além das regras heurísticas baseadas nas funções sigmóides (5.1.4) e 5.1.6 é usada também a regra heurística definida pela função chão dada em (3.2.1).

5.1.3 Implementação das HBFA

De seguida, descrevem-se três métodos diferentes de implementação das HBFA no FA dinâmico para a resolução de problemas de MINLP binários.

Numa primeira implementação, designada por *movimento no espaço contínuo* (mEC), o movimento de cada pirilampo é feito no espaço contínuo e o termo da sua atratividade é atualizado considerando o vetor posição real. O vetor posição real do pirilampo i é discretizado apenas após o movimento de todos os pirilampos em direção ao pirilampo mais brilhante. É de salientar que a avaliação de cada pirilampo, para a sua ordenação, é sempre baseado na sua posição binária. O Algoritmo 9 apresenta HBFA com mEC.

Algoritmo 9 Algoritmo HBFA com mEC

Dados: $k_{\max}$, f^*, η

Fazer $k = 0$; Gerar aleatoriamente $x_i^{(k)} \in X$, $i = 1, \ldots, N$

Discretizar a posição do pirilampo i: $x_i^{(k)} \to b_i^{(k)}$

Calcular $f(b_i^{(k)}), i = 1, \ldots, N$; Ordenar os pirilampos (do menor para o maior valor de f)

Enquanto $k \leq k_{\max}$ e $\left| f(b_1^{(k)}) - f^* \right| > \eta$ **fazer**

  Para $x_i^{(k)}$, $i = 2, \ldots, N$ **fazer**

    Para $x_j^{(k)}$, $j = 1, \ldots, i-1$ **fazer**

      Calcular o termo aleatório; Calcula a atratividade β

      Mover a posição de $x_i^{(k)}$ do pirilampo i em direção $x_j^{(k)}$ usando (5.1.3)

    fim

  fim

  Discretizar as posições: $x_i^{(k+1)} \to b_i^{(k+1)}$; Calcular $f(b_i^{(k+1)}), i = 1, \ldots, N$

  Ordenar os pirilampos (do menor para o maior valor de f)

  Fazer $k = k + 1$

fim

A segunda implementação, designada de *movimento no espaço binário* (mEB), move a posição binária de cada pirilampo em direção às posições binárias dos pirilampos mais brilhantes, isto é, cada movimento é feito no espaço binário, embora a posição correspondente possa deixar de ter coordenadas com valor 0 ou 1. Neste caso, esta posição deve ser discretizada antes da atualização do termo de atratividade. No mEB, a avaliação da função objetivo é também baseada nas posições binárias dos pirilampos. O Algoritmo 10 descreve HBFA com mEB.

A terceira implementação, designada de *probabilidade para a componente binária* (pCB), usa o conceito do algoritmo PSO binário (Kennedy & Eberhart, 1997; Pampara et al., 2006;

Algoritmo 10 Algoritmo HBFA com mEB

Dados: $k_{\max}$, f^*, η Fazer $k = 0$

Gerar aleatoriamente $x_i^{(k)} \in X$, $i = 1, \ldots, N$

Discretizar a posição do pirilampo i: $x_i^{(k)} \rightarrow b_i^{(k)}$

Calcular $f(b_i^{(k)}), i = 1, \ldots, N$

Ordenar os pirilampos (do menor para o maior valor de f)

Enquanto $k \leq k_{\max}$ e $\left| f(b_1^{(k)}) - f^* \right| > \eta$ **fazer**

  Para $b_i^{(k)}$, $i = 2, \ldots, N$ **fazer**

    Para $b_j^{(k)}$, $j = 1, \ldots, i-1$ **fazer**

      Calcular o termo aleatório

      Calcular a atratividade β baseada na distância $\|b_i^{(k)} - b_j^{(k)}\|^p$

      Mover a posição binária $b_i^{(k)}$ do pirilampo i em direção a $b_j^{(k)}$ usando $x_i^{(k)} = b_i^{(k)} + \beta(b_j^{(k)} - b_i^{(k)}) + \alpha L(b_1^{(k)}, '1')\sigma_i^b$

      Discretizar a posição do pirilampo i: $x_i^{(k+1)} \rightarrow b_i^{(k+1)}$

    fim

  fim

  Calcular $f(b_i^{(k+1)}), i = 1, \ldots, N$

  Ordenar os pirilampos (do menor para o maior valor de f)

  Fazer $k = k + 1$

fim

L. Wang & Singh, 2009), onde cada coordenada do vetor velocidade é usada diretamente para calcular a probabilidade da coordenada correspondente da posição da partícula, $x_{i,s}$, ser 0 ou 1. De modo análogo, no FA, o vetor y_i na equação (5.1.3) não é interpretado como um comprimento de passo, mas sim como um meio de calcular a probabilidade de cada coordenada do vetor posição de pirilampo i ser 0 ou 1. Assim, define-se

$$b_{i,s}^{(k+1)} = \begin{cases} 1, & \text{se } rand \leq \text{sig}(y_{i,s}^{(k+1)}) \\ 0, & \text{caso contrário} \end{cases}, \tag{5.1.7}$$

onde sig(.) representa o valor de uma função sigmóide. O Algoritmo 11 descreve HBFA com pCB.

Algoritmo 11 Algoritmo HBFA com pCB

Dados: $k_{\max}$, f^*, η

Fazer $k = 0$

Gerar aleatoriamente $x_i^{(k)} \in X$, $i = 1, \ldots, N$

Discretizar a posição do pirilampo i: $x_i^{(k)} \rightarrow b_i^{(k)}$

Calcular $f(b_i^{(k)}), i = 1, \ldots, N$

Ordenar os pirilampos (do menor para o maior valor de f)

Enquanto $k \leq k_{\max}$ e $\left| f(b_1^{(k)}) - f^* \right| > \eta$ **fazer**

  Para $b_i^{(k)}$, $i = 2, \ldots, N$ **fazer**

    Para $b_j^{(k)}$, $j = 1, \ldots, i-1$ **fazer**

      Calcular o termo aleatório

      Calcular a atratividade β baseada na distância $\|b_i^{(k)} - b_j^{(k)}\|^p$

      Calcular $y_i^{(k+1)}$ usando as posições binárias segundo a equação (5.1.3)

      Discretizar $y_i^{(k+1)}$ e definir $b_i^{(k+1)}$ usando (5.1.7)

    fim

  fim

  Calcular $f(b_i^{(k+1)}), i = 1, \ldots, N$

  Ordenar os pirilampos (do menor para o maior valor de f)

  Fazer $k = k + 1$

fim

As experiências computacionais com os algoritmos estabelecidos nesta secção, HBFA-Algoritmos 9, 10 e 11, usando as regras heurísticas definidas pelas funções sigmóide (5.1.4) e (5.1.6) e a função (3.2.1), são apresentadas na Secção 6.2.2.

5.2 Com restrições de limites simples e variáveis mistas

Nesta secção descreve-se a extensão do FA para resolver problemas de MINLP com restrições de limites simples e variáveis contínuas e inteiras, onde é usada uma reformulação contínua de penalidade exata do problema (1.2.8).

Considere-se o problema de MINLP com restrições de limites simples definido em (1.2.8), cuja região admissível é dada pelo conjunto W definido em (1.2.9). Considere-se ainda a seguinte reformulação contínua do problema de MINLP (1.2.8), que surge ao

relaxar as restrições de integralidade e adicionando um termo de penalidade específico $P(x;\varepsilon)$ à função objetivo:

$$\begin{aligned} \min \quad & \psi(x;\varepsilon) \equiv f(x) + P(x;\varepsilon) \\ \text{sujeito a} \quad & x \in X \\ & x_i \in \mathbb{R} \text{ para } i = 1,\ldots,n, \end{aligned} \tag{5.2.1}$$

onde $\varepsilon \in]0,\bar{\varepsilon}]$.

Assumindo que a função f satisfaz a Hipótese 1 e que o termo de penalidade $P(.;\varepsilon)$ satisfaz as Hipóteses 2 e 3 (ver Secção 3.2), tem-se que a função $\psi(\cdot;\varepsilon)$ (5.2.1) é 'exata' no sentido que existe $\bar{\varepsilon} > 0$ tal que, para todo $\varepsilon \in]0,\bar{\varepsilon}]$, os problemas (1.2.8) e (5.2.1) são equivalentes, i.e, têm os mesmos minimizantes globais (Lucidi & Rinaldi, 2010, 2013).

Em (Lucidi & Rinaldi, 2010), assumindo que f satisfaz a Hipótese 1, demonstra-se que uma classe específica de termos de penalidade satisfaz as Hipóteses 2 e 3 (ver Secção 3.2). Deste modo, pode aplicar-se o Teorema 3.2.1, e concluir-se que $\exists\bar{\varepsilon} > 0$ tal que, para qualquer $\varepsilon \in]0,\bar{\varepsilon}]$, o problema (1.2.8) e o problema (5.2.1), baseado num termo de penalidade dessa classe, têm os mesmos minimizantes globais. Podem-se definir outros termos de penalidade para o problema 5.2.1, desde que sejam mantidas certas propriedades. No âmbito desta tese, propõem-se dois novos termos de penalidade. Nomeadamente, um termo de penalidade baseado na função *tangente hiperbólica* e outro na função *inversa do seno hiperbólico*. Para os dois termos de penalidade propostos, prova-se que as Hipóteses 2 e 3 são satisfeitas.

5.2.1 A penalidade tangente hiperbólica

O primeiro termo de penalidade proposto, para resolver problemas de MINLP através da equivalência entre (1.2.8) e (5.2.1), baseia-se na função tangente hiperbólica, $\tanh(\cdot)$, que é diferenciável e estritamente crescente em X, dado por:

$$P^t(x;\varepsilon) = \frac{1}{\varepsilon} \sum_{i \in I_d} \min_{\substack{l_i \leq d_j \leq u_i \\ d_j \in \mathbb{Z}}} \tanh\left(|x_i - d_j| + \varepsilon\right) \tag{5.2.2}$$

onde $\varepsilon \in \mathbb{R}^+$ é o parâmetro de penalidade.

Propriedade 1. *Para o termo de penalidade* (5.2.2)*, existe um valor* $\bar{\varepsilon} > 0$ *tal que, para qualquer* $\varepsilon \in]0, \bar{\varepsilon}]$*, os problemas* (1.2.8) *e* (5.2.1) *têm os mesmos pontos mínimos.*

Demonstração: De acordo com o Teorema 3.2.1, para provar que a penalidade P^t pode ser usada no problema (5.2.1), tem-se que demonstrar que o termo de penalidade (5.2.2) satisfaz as Hipóteses 2 e 3. Assume-se que f satisfaz a Hipótese 1. $\forall x, y \in W$, tem-se

$$P^t(x;\varepsilon) = \frac{1}{\varepsilon}\sum_{i\in I_d}\tanh(\varepsilon) = |I_d|\frac{1}{\varepsilon}\tanh(\varepsilon) = P^t(y;\varepsilon)$$

e a Hipótese 2 é satisfeita. Para demonstrar que a Hipótese 3 é satisfeita, estendeu-se o raciocínio usado por Lucidi e Rinaldi (2010) ao termo de penalidade tangente hiperbólica. Assim, estuda-se o comportamento do termo relacionado com x_i (para $i \in I_d$), isto é,

$$P_i^t(x_i;\varepsilon) = \frac{1}{\varepsilon}\min_{\substack{l_i\le d_j\le u_i\\ d_j\in\mathbb{Z}}}\tanh\left(|x_i - d_j| + \varepsilon\right),$$

numa vizinhança de um ponto admissível z_i. Considere-se os três casos seguintes, onde ρ é uma constante positiva suficientemente pequena:

(i) $z_i = d_j$ e $d_j < x_i < d_j + \rho$;

(ii) $z_i = d_j$ e $d_j - \rho < x_i < d_j$;

(iii) $z_i = x_i = d_j$.

Caso (i): Usando o Teorema do valor médio obtém-se

$$P_i^t(x_i;\varepsilon) - P_i^t(z_i;\varepsilon) = \frac{1}{\varepsilon\cosh^2\left((\tilde{x}_i - d_j) + \varepsilon\right)}|x_i - z_i| \ge \frac{1}{\varepsilon\cosh^2\left(\rho + \varepsilon\right)}|x_i - z_i|$$

onde $\tilde{x}_i \in [d_j, x_i]$. Escolhendo ρ e ε tal que

$$\frac{1}{\varepsilon\cosh^2\left(\rho + \varepsilon\right)} \ge \bar{L} \tag{5.2.3}$$

obtém-se

$$P_i^t(x_i;\varepsilon) - P_i^t(z_i;\varepsilon) \ge \bar{L}|x_i - z_i|. \tag{5.2.4}$$

Caso (ii): Usando o Teorema do valor médio obtém-se

$$P_i^t(x_i;\varepsilon) - P_i^t(z_i;\varepsilon) = \frac{1}{\varepsilon \cosh^2\left((d_j - \tilde{x}_i) + \varepsilon\right)}|x_i - z_i| \geq \frac{1}{\varepsilon \cosh^2\left(\rho + \varepsilon\right)}|x_i - z_i|$$

onde $\tilde{x}_i \in [x_i, d_j]$. Portanto, usando o mesmo raciocínio que no Caso (i), tem-se novamente que 5.2.4 é satisfeita quando ρ e ε satisfazem 5.2.3.

Caso (iii): Tem-se que $P_i^t(x_i;\varepsilon) - P_i^t(z_i;\varepsilon) = 0$, pela Hipótese 2. Pode-se concluir que, quando ρ e ε satisfazm (5.2.3), tem-se

$$P^t(x;\varepsilon) - P^t(z;\varepsilon) \geq \bar{L}\sum_{i\in I_d}|x_i - z_i| = \bar{L}\|x - z\|_1 \geq \bar{L}\|x - z\|_\infty$$

para todo $z \in X$ com $z_i \in \mathbb{Z}$ para $i \in I_d$, e $z_j = x_j$ para $j \in I\backslash I_d$, e para todo x tal que $\|x - z\|_\infty < \rho$.

A condição (3.2.8) da Hipótese 3 é satisfeita se S for definido como a união das vizinhanças $S(z) = \{x \in \mathbb{R}^n : \|x - z\|_\infty < \rho\}$ para todo $z \in W$.

Para provar (3.2.9) e (3.2.10), considere-se a sucessão $\{\varepsilon^{(k)}\}$ convergindo para 0 quando $k \to \infty$ (onde k representa o contador de iterações do algoritmo) e seja $\bar{x}$ um ponto dado por

$$\begin{cases} \bar{x}_s = d_s \pm \rho, \text{ onde } l_s \leq d_s \leq u_s, d_s \in \mathbb{Z}, \\ \bar{x}_j = d_j \text{ onde } l_j \leq d_j \leq u_j, d_j \in \mathbb{Z} \text{ para todo } j \neq s, \text{ onde } j, s \in I_d \\ l_j \leq \bar{x}_j \leq u_j, \bar{x}_j \in \mathbb{R} \text{ para } j \in I\backslash I_d \end{cases} \tag{5.2.5}$$

e para qualquer $z \in W$, tem-se

$$\begin{aligned} \lim_{k\to\infty}\left(P^t(\bar{x};\varepsilon^{(k)}) - P^t(z;\varepsilon^{(k)})\right) = & \lim_{k\to\infty}\left(\frac{1}{\varepsilon^{(k)}}\tanh(\rho + \varepsilon^{(k)})\right. \\ & \left. + \frac{1}{\varepsilon^{(k)}}(|I_d| - 1)\tanh(\varepsilon^{(k)}) - \frac{1}{\varepsilon^{(k)}}|I_d|\tanh(\varepsilon^{(k)})\right) \\ = & \lim_{k\to\infty}\frac{1}{\varepsilon^{(k)}}\left(\tanh(\rho + \varepsilon^{(k)}) - \tanh(\varepsilon^{(k)})\right) \\ = & +\infty. \end{aligned}$$

Além disso, para todo $x \in X \backslash S$ e todo $\varepsilon > 0$ tem-se

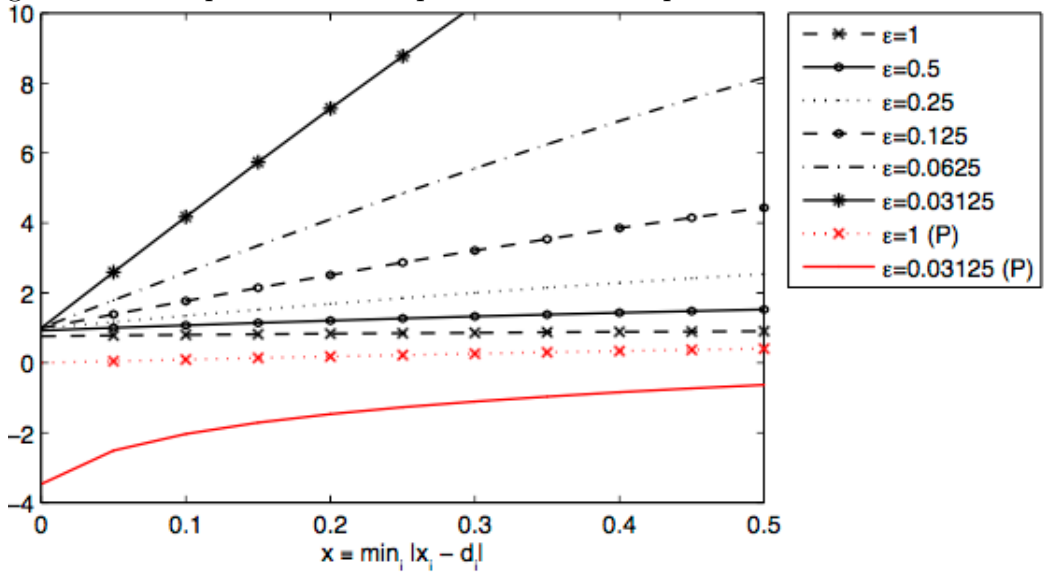
$$\begin{aligned} P^t(x;\varepsilon) - P^t(\bar{x};\varepsilon) = & \frac{1}{\varepsilon}\sum_{i\in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \tanh\left(|x_i - d_j| + \varepsilon\right) - \frac{1}{\varepsilon}\sum_{i\in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \tanh\left(|\bar{x}_i - d_j| + \varepsilon\right) \\ = & \frac{1}{\varepsilon}\sum_{i\in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \tanh\left(|x_i - d_j| + \varepsilon\right) - \frac{1}{\varepsilon}\tanh(\rho+\varepsilon) - \frac{1}{\varepsilon}(|I_d| - 1)\tanh(\varepsilon) \\ = & \frac{1}{\varepsilon}\sum_{i\neq\bar{l}\in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \tanh\left(|x_i - d_j| + \varepsilon\right) - \frac{1}{\varepsilon}(|I_d| - 1)\tanh(\varepsilon) \\ & + \frac{1}{\varepsilon}\tanh\left(|x_{\bar{l}} - \bar{d}| + \varepsilon\right) - \frac{1}{\varepsilon}\tanh(\rho+\varepsilon) \\ \geq & \ 0 \end{aligned}$$

uma vez que a função tangente hiperbólica é estritamente crescente, $|x_{\bar{l}} - \bar{d}| \geq \rho$ e $\bar{d}$ é o valor inteiro admissível mais próximo de $x_{\bar{l}}$.

□

Na Figura 5.2 mostra-se o comportamento do termo de penalidade 'tanh' (5.2.2) à medida que ε diminui ($\varepsilon = 1, 0.5, 0.25, 0.125, 0.0625, 0.03125$). Na figura mostra-se também o comportamento do termo de penalidade 'log' (3.2.18), representado por (P), para $\varepsilon = 1$ e $\varepsilon = 0.03125$. Observa-se que a penalidade 'tanh' cresce mais rapidamente do que a penalidade 'log', à medida que ε diminui e a violação das restrições de integralidade aumenta.

Figura 5.2: Comportamento da penalidade 'tanh' para seis valores diferentes de ε



5.2.2 A penalidade inversa do seno hiperbólico

A inversa do seno hiperbólico, denotada por asinh(.), é uma função diferenciável para todo $x \in \mathbb{R}$, e é também não negativa e estritamente crescente no intervalo $[0, +\infty[$. Assim, o outro termo de penalidade proposto, para o problema 5.2.1, é dado por:

$$P^a(x;\varepsilon) = \sum_{i \in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_i - d_j| + \varepsilon\right) \tag{5.2.6}$$

onde $\varepsilon > 0$ é o parâmetro de penalidade.

Uma vez mais, usando o Teorema 3.2.1, pode demonstrar-se a seguinte propriedade.

Propriedade 2. *Para o termo de penalidade* (5.2.6)*, existe um valor* $\bar{\varepsilon} > 0$ *tal que para qualquer* $\varepsilon \in]0, \bar{\varepsilon}]$*, os problemas* (1.2.8) *e* (5.2.1) *têm os mesmos pontos mínimos.*

Demonstração: De acordo com o Teorema 3.2.1, para provar que a penalidade P^a pode ser usada em (5.2.1), tem-se que demonstrar que o termo de penalidade (5.2.6) satisfaz as Hipóteses 2 e 3. Assume-se que f satisfaz a Hipótese 1. $\forall x, y \in W$,

$$P^a(x;\varepsilon) = \sum_{i \in I_d} \operatorname{asinh}(\varepsilon) = |I_d| \operatorname{asinh}(\varepsilon) = P^a(y;\varepsilon)$$

e a Hipótese 2 é satisfeita.

Para provar que a Hipótese 3 é satisfeita, estuda-se o comportamento de

$$P_i^a(x_i;\varepsilon) = \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_i - d_j| + \varepsilon\right), \text{ para todo } i \in I_d,$$

na vizinhança de um ponto admissível z_j, e considere-se os três casos referidos anteriormente. Seja $\rho > 0$ um valor real suficientemente pequeno.

Caso (i): usando o Teorema do valor médio obtém-se

$$P_i^a(x_i;\varepsilon) - P_i^a(z_i;\varepsilon) = \frac{1}{\varepsilon\sqrt{1 + \left(\frac{1}{\varepsilon}(\tilde{x}_i - d_j) + \varepsilon\right)^2}}|x_i - z_i| \ge \frac{1}{\varepsilon\sqrt{1 + \left(\frac{1}{\varepsilon}\rho + \varepsilon\right)^2}}|x_i - z_i|$$

onde $\tilde{x}_i \in [d_j, x_i]$. Escolhendo ρ e ε tais que

$$\frac{1}{\varepsilon\sqrt{1 + \left(\frac{1}{\varepsilon}\rho + \varepsilon\right)^2}} \ge \bar{L} \tag{5.2.7}$$

obtém-se

$$P_i^a(x_i;\varepsilon) - P_i^a(z_i;\varepsilon) \geq \bar{L}|x_i - z_i|. \tag{5.2.8}$$

Caso (ii): De uma forma análoga tem-se que:

$$\begin{aligned} P_i^a(x_i;\varepsilon) - P_i^a(z_i;\varepsilon) &= \frac{1}{\varepsilon\sqrt{1+\left(\frac{1}{\varepsilon}(d_j - \tilde{x}_i) + \varepsilon\right)^2}}|x_i - z_i| \\ &\geq \frac{1}{\varepsilon\sqrt{1+\left(\frac{1}{\varepsilon}\rho + \varepsilon\right)^2}}|x_i - z_i| \\ &\geq \bar{L}|x_i - z_i| \end{aligned}$$

onde $\tilde{x}_i \in [x_i, d_j]$.

Caso (iii): Neste caso, obtém-se $P_i^a(x_i;\varepsilon) - P_i^a(z_i;\varepsilon) = 0$, e conclui-se que para ρ e ε satisfazendo (5.2.7), tem-se

$$P^a(x;\varepsilon) - P^a(z;\varepsilon) \geq \bar{L}\sum_{i\in I_d}|x_i - z_i| = \bar{L}\|x - z\|_1 \geq \bar{L}\|x - z\|_\infty$$

para todo $z \in X$ (onde $z_i \in \mathbb{Z}$ para $i \in I_d$, e $z_j = x_j$ para $j \in I\backslash I_d$) e para todo x tal que $\|x - z\|_\infty < \rho$.

Se S for definido como a união das vizinhanças $S(z) = \{x \in \mathbb{R}^n : \|x - z\|_\infty < \rho\}$ para todo $z \in W$, então a condição (3.2.8) na Hipótese 3 é satisfeita.

Para provar (3.2.9) e (3.2.10), considere-se a sucessão $\{\varepsilon^{(k)}\}$ convergindo para 0 (para $k \to \infty$), o ponto $\bar{x}$ definido por (5.2.5) e $z \in W$. Então,

$$\begin{aligned} \lim_{k\to\infty}\left(P^a(\bar{x};\varepsilon^{(k)}) - P^a(z;\varepsilon^{(k)})\right) &= \lim_{k\to\infty}\left(\operatorname{asinh}\left(\frac{1}{\varepsilon^{(k)}}\rho + \varepsilon^{(k)}\right)\right. \\ &\quad \left. +(|I_d| - 1)\operatorname{asinh}(\varepsilon^{(k)}) - |I_d|\operatorname{asinh}(\varepsilon^{(k)})\right) \\ &= \lim_{k\to\infty}\left(\operatorname{asinh}\left(\frac{1}{\varepsilon^{(k)}}\rho + \varepsilon^{(k)}\right) - \operatorname{asinh}(\varepsilon^{(k)})\right) \\ &= +\infty. \end{aligned}$$

Além disso, para todo $x \in X \backslash S$ e para todo $\varepsilon > 0$ tem-se

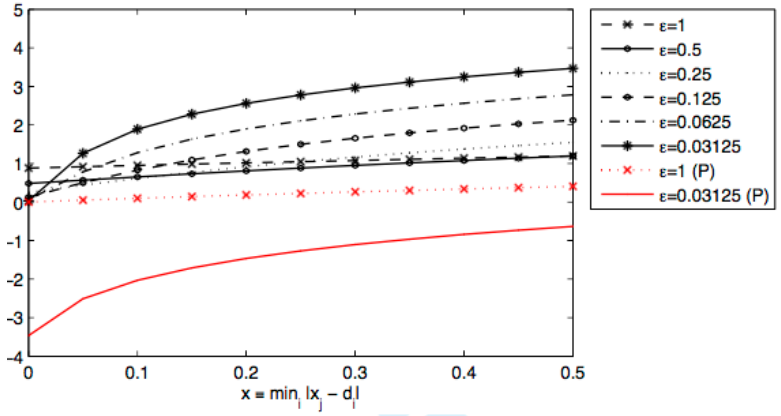
$$\begin{aligned}
P^a(x;\varepsilon) - P^a(\bar{x};\varepsilon) = & \sum_{i \in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_i - d_j| + \varepsilon\right) \\
& - \sum_{i \in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|\bar{x}_i - d_j| + \varepsilon\right) \\
= & \sum_{i \in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_i - d_j| + \varepsilon\right) - \operatorname{asinh}\left(\frac{1}{\varepsilon}\rho + \varepsilon\right) \\
& - (|I_d| - 1)\operatorname{asinh}(\varepsilon) \\
= & \sum_{i \neq \bar{l} \in I_d} \min_{\substack{l_i \le d_j \le u_i \\ d_j \in \mathbb{Z}}} \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_i - d_j| + \varepsilon\right) - (|I_d| - 1)\operatorname{asinh}(\varepsilon) \\
& + \operatorname{asinh}\left(\frac{1}{\varepsilon}|x_{\bar{l}} - \bar{d}| + \varepsilon\right) - \operatorname{asinh}\left(\frac{1}{\varepsilon}\rho + \varepsilon\right) \\
\geq & \ 0
\end{aligned}$$

uma vez que a função inversa do seno hiperbólico é monótona crescente, $|x_{\bar{l}} - \bar{d}| \geq \rho$ e $\bar{d}$ é o valor inteiro admissível mais próximo de $x_{\bar{l}}$.

□

O comportamento da penalidade 'asinh' (5.2.6) para $\varepsilon = 1, 0.5, 0.25, 0.125, 0.0625, 0.03125$ é representado na Figura 5.3. Para comparação, é também representada a penalidade 'log' (P) para $\varepsilon = 1$ e $\varepsilon = 0.03125$. Neste caso, à medida que ε diminui e a violação das restrições de integralidade aumenta, o comportamento da penalidade 'asinh' é um pouco semelhante ao da penalidade 'log'.

Figura 5.3: Comportamento da penalidade 'asinh' para seis valores diferentes de ε.



5.2.3 O algoritmo de penalidade exata

Apresenta-se a seguir uma breve descrição do algoritmo de penalidade exata, Algoritmo 12, que calcula um minimizante global de um problema de MINLP (Lucidi & Rinaldi, 2013). O algoritmo de penalidade exata é implementado com os novos termos de penalidade (5.2.2) e (5.2.6), uma vez que estes satisfazem as Hipóteses 2 e 3, como demonstrado nas Subsecções 5.2.1 e 5.2.2.

Para calcular uma aproximação $\delta^{(k)}$ ao minimizante global do problema (5.2.1) que satisfaz

$$\psi(x^{(k)};\varepsilon^{(k)}) \leq \psi(x;\varepsilon^{(k)}) + \delta^{(k)}, \text{ para todo } x \in X \tag{5.2.9}$$

e para valores fixos $\varepsilon^{(k)}$ e $\delta^{(k)}$, é usado o FA.

O parâmetro de penalidade é atualizado $\varepsilon^{(k+1)} = \sigma_\varepsilon \varepsilon^{(k)}$, $\sigma_\varepsilon \in]0,1[$, quando o minimizante global $x^{(k)}$, da função de penalidade $\psi(x;\varepsilon^{(k)})$, não é admissível para o problema (1.2.8) e a condição

$$\psi(x^{(k)};\varepsilon^{(k)}) - \psi(z^{(k)};\varepsilon^{(k)}) \leq L(\varepsilon^{(k)})\|x^{(k)} - z^{(k)}\|^p \tag{5.2.10}$$

é satisfeita, onde $z^{(k)} \in W$ é o ponto que minimiza a distância entre $x^{(k)}$ e o conjunto $S(z^{(k)}) = \{x \in \mathbb{R}^n : \|x - z^{(k)}\|_\infty < \rho\}$, para um valor positivo suficientemente pequeno ρ, i.e., $z_i^{(k)} \in \mathbb{Z}, i \in I_d$ resulta do arredondamento $x_i^{(k)}$ para o valor inteiro mais próximo, e $z_j^{(k)} = x_j^{(k)}, j \in I_c$ (ver Proposição 2 em (Lucidi & Rinaldi, 2013)). Por outro lado, o parâmetro $\varepsilon^{(k)}$ mantém-se inalterado quando $x^{(k)}$ é admissível ou quando $x^{(k)}$ é não admissível mas a condição (5.2.10) não é satisfeita. Na implementação do Algoritmo de penalidade 12, fez-se a função $L(\varepsilon^{(k)})$ depender do termo de penalidade selecionado ('tanh'(5.2.2) ou 'asinh' (5.2.6)), usando os limites simples superiores definidos para $\bar{L}$ em (5.2.3) ou (5.2.7), respetivamente, e garantindo que $L(\varepsilon^{(k)})$ é positivo para $\varepsilon^{(k)} > 0$ e $L(\varepsilon^{(k)}) \to 0$ quando $\varepsilon^{(k)} \to 0$.

O algoritmo termina quando é encontrada uma solução admissível dentro da tolerância $\delta_{\min}$ da solução ótima global conhecida f^*. É de referir que, se o valor ótimo não é conhecido, o algoritmo pode usar outro critério de paragem, como por exemplo, um baseado no

Algoritmo 12 Algoritmo de penalidade

Dados: f^* (ou $k_{\max}$), $\delta_{\min} \ll 1$, $\varepsilon^{(1)} > 0$, $\delta^{(1)} > 0$, $\sigma_\varepsilon \in]0,1[$, $\sigma_\delta \in]0,1[$, p

Fazer $k = 1$; Gerar aleatoriamente $x^{(0)} \in X$

Enquanto $x^{(k-1)} \notin W$ ou $f(x^{(k-1)}) > f^* + \delta_{\min}$ **fazer**

Dado $x^{(k-1)}$, calcular uma aproximação para o minimizante global $x^{(k)}$ do problema (5.2.1) usando o Algoritmo 13 tal que

$$\psi(x^{(k)}; \varepsilon^{(k)}) \leq \psi(x; \varepsilon^{(k)}) + \delta^{(k)}, \quad \text{para todo } x \in X$$

Se $x^{(k)} \notin W$ *e* $\psi(x^{(k)}; \varepsilon^{(k)}) - \psi(z^{(k)}; \varepsilon^{(k)}) \leq L(\varepsilon^{(k)}) \|x^{(k)} - z^{(k)}\|^p$ **então**

Fazer $\varepsilon^{(k+1)} = \sigma_\varepsilon \varepsilon^{(k)}, \delta^{(k+1)} = \delta^{(k)}$

Senão

Fazer $\varepsilon^{(k+1)} = \varepsilon^{(k)}, \delta^{(k+1)} = \sigma_\delta \delta^{(k)}$

fim

Fazer $k = k + 1$

fim

número de iterações, $k > k_{\max}$, para um dado valor máximo de iterações permitido $k_{\max}$ (ou sobre o número de avaliações da função objetivo, $nf > nf_{\max}$, onde $nf_{\max}$ é o número máximo de avaliações permitido).

Os resultados de convergência apresentados por Lucidi e Rinaldi (2013) para um modelo de algoritmo de penalidade exata, onde um problema de NLP com restrições genéricas é considerado equivalente a uma sequência de problemas de penalidade obtidos adicionando um termo de penalidade específico à função objetivo, foram estendidos ao algoritmo de penalidade exata com o termo de penalidade (3.2.18) para resolver um problema de MINLP através de uma reformulação contínua de NLP. Baseados nos resultados apresentados por Lucidi e Rinaldi (2013), vão estabelecer-se os principais resultados de convergência do Algoritmo 12.

Assume-se que a sequência $\{x^{(k)}\}$ está bem definida, isto é, encontra-se o minimizante global aproximado $x^{(k)}$ da função de penalidade $\psi(\cdot)$ que satisfaz a condição (5.2.9).

Uma vez que é usado o FA para calcular um minimizante global aproximado, não existe garantia teórica de que a solução aproximada possa ser encontrada em tempo finito. Com base na teoria da probabilidade, pode estabelecer-se uma convergência estocástica

para o FA. Este tipo de convergência é diferente do conceito de convergência da Análise clássica. A versão probabilística de convergência pontual da Análise real elementar, é a convergência com probabilidade 1. Isto significa que, no limite, a convergência do FA para um minimizante aproximado da função de penalidade será garantida com probabilidade 1.

Lema 5.2.1. *Seja* $\{x^{(k)}\}$ *uma sucessão de iterandos produzidos pelo Algoritmo 12. Então, verifica-se um dos casos seguintes:*

- $\exists \bar{k}$ *tal que* $\varepsilon^{(k)} = \bar{\varepsilon}$*, para qualquer* $k \geq \bar{k}$*, e qualquer ponto de acumulação* $\bar{x}$ *da sucessão* $\{x^{(k)}\}$ *é admissível (*$\bar{x} \in W$*);*

- $\varepsilon^{(k)} \to 0$ *e qualquer ponto de acumulação* $\bar{x}$ *da subsucessão* $\{x^{(k_i)}\}$ *é admissível, com* k_i *pertencente ao conjunto de índices em que a condição* (5.2.10) *é satisfeita.*

O Lema seguinte garante que o parâmetro ε é atualizado um número finito de vezes.

Lema 5.2.2. *Sejam* $\{x^{(k)}\}$ *e* $\{\varepsilon^{(k)}\}$ *sucessões de pontos produzidos pelo Algoritmo 12. Então,* $\exists \bar{k}$ *tal que* $\varepsilon^{(k)} = \bar{\varepsilon}$*, para qualquer* $k \geq \bar{k}$*.*

O resultado seguinte é uma consequência dos Lemas 5.2.1 e 5.2.2, e de $\delta^{(k)} \to 0$.

Teorema 5.2.1. *Qualquer ponto de acumulação* $\bar{x}$ *da sucessão* $\{x^{(k)}\}$ *produzida pelo Algoritmo 12 é um minimizante global do problema* (1.2.8)*.*

Uma vez que os termos de penalidade propostos satisfazem as Hipóteses 2 e 3, as demonstrações dos Lemas 5.2.1 e 5.2.2, e Teorema 5.2.1 seguem os mesmos argumentos utilizados nas demonstrações dos Lema 1, Lema 2 e Teorema 2 em (Lucidi & Rinaldi, 2013).

As experiências computacionais com o Algoritmo 12, com as penalidades $P^t(.)$ (5.2.2) e $P^a(.)$ (5.2.6), em que usa o FA clássico e o FA adaptativo na resolução dos subproblemas, são apresentados na Secção 6.2.3. As descrições do FA clássico e FA adaptativo são apresentadas na secção seguinte.

5.2.4 FA adaptativo

Tal como referido no Capítulo 2, o desempenho de qualquer metaheurística depende das características de diversificação (ou exploração) e de intensificação (ou refinamento) do algoritmo. A diversificação tem como objetivo explorar uma porção muito maior do espaço de procura com a esperança de calcular outras soluções promissoras, que precisam ainda de ser refinadas. Deste modo promove-se uma procura global, a fim de evitar ficar preso num ótimo local. A capacidade de exploração depende da aleatoriedade. Por sua vez, a intensificação tem como objetivo explorar uma região limitada (mas promissora) do espaço de procura com a esperança de melhorar uma solução promissora identificada. Deste modo, promove-se uma procura local. Para qualquer metaheurística, o equilíbrio entre a procura local e global é determinante para o seu sucesso. No FA, ambos os tipos de procura emergem da equação do movimento do pirilampo (ver equação 4.2.8), em particular, estes dependem de como variam os parâmetros de controlo. Assim, uma escolha apropriada dos parâmetros é fundamental para aumentar a eficiência geral do algoritmo. É proposto um FA adaptativo no sentido em que a atividade de procura local dependa da variação relativa da função de penalidade, entre o pirilampo i e cada um dos outros pirilampos mais brilhantes. Quanto mais brilhante é o pirilampo (relativamente ao pirilampo i), maior é a atração, i.e., quando $\psi^{(k)}(x_j) < \psi^{(k)}(x_l) < \psi^{(k)}(x_i)$, a atratividade é maior quando o pirilampo i move-se em direção ao pirilampo j do que em direção ao pirilampo l, sendo igual a 1 quando o movimento é em direção ao mais brilhante x_1.

No contexto da resolução do problema contínuo de NLP (5.2.1), a equação que descreve o movimento de um pirilampo i em direção a um pirilampo mais brilhante j, na iteração k, é definida por:

$$x_i^{(k+1)} = x_i^{(k)} + \exp(-\alpha)\frac{\psi^{(k)}(x_i) - \psi^{(k)}(x_j)}{\psi^{(k)}(x_i) - \psi^{(k)}(x_1)}(x_j^{(k)} - x_i^{(k)}) + \alpha\mathcal{L}(0,1)\sigma^i/2, \tag{5.2.11}$$

onde a atratividade depende da variação da função de penalidade ψ entre o pirilampo i e o pirilampo mais brilhante j, relativa à diferença entre o pirilampo i e o pirilampo mais brilhante de todos. Assim, o FA adaptativo tem a capacidade de adaptar-se facilmente à forma do problema. O termo aleatório é similar ao de (5.1.3). Note-se que o segundo termo

da equação (5.2.11) é dimensionado pelo parâmetro $\exp(-\alpha)$, cujo objetivo é reforçar a dominância do segundo termo no final do processo iterativo, enquanto o terceiro termo domina nas iterações iniciais (ver a fórmula de atualização (5.1.1). O FA adaptativo possui boas capacidades de exploração e de refinamento. Ao contrário da versão clássica e de outras variantes do FA, o FA adaptativo tem apenas um parâmetro de controlo. O Algoritmo 13 apresenta os passos principais do FA adaptativo. Note-se que os pirilampos são agora ordenados de acordo com o valor da função de penalidade na sua posição, i.e., de acordo com $\psi^k(\cdot) \equiv \psi(\cdot; \varepsilon^{(k)})$, sendo x_1 o pirilampo mais brilhante. Para distinguir as iterações k do Algoritmo 12 com as iterações do FA é usada a notação k^{FA}.

Algoritmo 13 FA adaptativo

Dados: $k^{FA}_{\max}$, f^*, e do Algoritmo 12: $x_1 = x^{(k-1)}$, $\varepsilon^{(k)}$ e $\delta^{(k)}$;

Fazer $k^{FA} = 1$; Gerar aleatoriamente x_i, $i = 2, \ldots, N$; Calcular $\psi^k(x_i) \equiv \psi(x_i; \varepsilon^{(k)}), i = 1, \ldots, N$; Ordenar os pirilampos por ordem crescente do valor de ψ

Enquanto $k^{FA} \leq k^{FA}_{\max}$ **e** $\psi^k(x_1) > f^* + \delta^{(k)}$ **fazer**

  Para $x_i, i = 2, \ldots, N$ **fazer**

    Para x_j, $j < i$ **fazer**

      Se $\psi^k(x_j) < \psi^k(x_i)$ **então**

        Mover o pirilampo i em direção ao pirilampo j usando (5.2.11)

        Calcular $\psi^k(x_i)$

      fim;

    fim;

  fim;

  Calcular $\psi^k(x_i), i = 1, \ldots, N$

  Ordenar os pirilampos por ordem crescente do valor de ψ

  Fazer $k^{FA} = k^{FA} + 1$

fim;

Designar-se-á por 'FA clássico', a versão em que o movimento de um pirilampo i em direção a um pirilampo mais brilhante j é dado pela equação (4.2.8) e os parâmetros α,

γ e β são atualizados ao longo das iterações de acordo com as equações (5.1.1), (5.1.2) e (2.1.1), respetivamente.

As experiências computacionais realizadas para avaliar a velocidade convergência e a qualidade das soluções obtidas com o algoritmo de penalidade quando testado com os termos de penalidade $P^t(\cdot)$ em 5.2.2 e $P^a(\cdot)$ em (5.2.6), são apresentadas na Secção 6.2.3.

5.3 Com restrições genéricas

Nesta secção descreve-se a extensão do FA capaz de resolver problemas de MINLP com restrições genéricas e variáveis mistas. A formulação matemática deste problema de MINLP é dada por (1.2.11). Embora atualmente as funções de penalidade sejam frequentemente usadas para tratar as condições de integralidade e as restrições de igualdade e desigualdade, neste estudo propõe-se a implementação no FA de uma heurística baseada no arredondamento e as regras de admissibilidade e otimalidade RAO apresentadas na Secção 4.1. Esta extensão do FA será designada pelo acrónimo FAbRAO.

5.3.1 Tratamento das restrições pelas RAO

Considere-se o problema de MINLP dado por (1.2.11). Neste problema, as variáveis contínuas são representadas pelo vetor x_c e $\Gamma_{x_c} \subset \mathbb{R}^{n_c}$ é o conjunto definido pelas restrições de limites simples em x_c:

$$\Gamma_{x_c} = \{x_c \in \mathbb{R}^{n_c} : \ lb_c \leq x_c \leq ub_c, \ lb_c, ub_c \in \mathbb{R}^{n_c}\}.$$

As variáveis inteiras são representadas pelo vetor x_z e $\Gamma_{x_z} \subset \mathbb{Z}^{n_d}$ é o conjunto definido pelas restrições de limites simples em x_z:

$$\Gamma_{x_z} = \{x_z \in \mathbb{Z}^{n_d} : \ lb_z \leq x_z \leq ub_z, \ lb_z, ub_z \in \mathbb{Z}^{n_d}\},$$

e $n = n_c + n_d$ representa o número total de variáveis. A região admissível do problema é dada por:

$$\Omega = \{(x_c, x_z) \in \Gamma_{x_c} \times \Gamma_{x_z} : g_i(x_c, x_z) \leq 0, i = 1, \ldots, m, \text{ e } h_j(x_c, x_z) = 0, \ j = 1, \ldots, p\}.$$

Para medir o grau de violação das restrições de igualdade e desigualdade num ponto $(x_c, x_z) \in \Gamma_{x_c} \times \Gamma_{x_z}$, é usada a seguinte função real não negativa definida por:

$$V(x_c, x_z) = \sum_{i=1}^{m} \max\{0, g_i(x_c, x_z)\} + \sum_{j=1}^{p} |h_j(x_c, x_z)|.$$

Assim, um ponto (x_c, x_z) que satisfaça as restrições de igualdade e de desigualdade, i.e., que verifica $V(x_c, x_z) = 0$, é uma solução admissível; se $V(x_c, x_z) > 0$, o ponto (x_c, x_z) é uma solução não admissível.

As dificuldades que surgem quando se resolve este tipo de problema estão relacionadas com as restrições de integralidade das variáveis x_z, e à não linearidade da função objetivo e das funções que definem as restrições do problema. Para lidar com as restrições de integralidade, bem como a violação de restrições, têm sido usadas técnicas baseadas em funções de penalidade (Lucidi & Rinaldi, 2010; Costa, Rocha, et al., 2016). Apesar da popularidade desta técnica, a sua utilização é desafiante, uma vez que requer uma escolha adequada *a priori* para os parâmetros de penalidade. Para ultrapassar essa dificuldade, e devido aos bons resultados obtidos pela implementação das RAO na resolução de problemas de NLP, as restrições de igualdade e desigualdade são tratadas pelas RAO, as quais favorecem as soluções admissíveis em detrimento das não admissívies, mas sem ignorá-las, na ordenação dos pontos da população. Quando as soluções são não admissíveis, a regra R_4 das RAO dá preferência à solução (x_{ci}, x_{zi}) que viola um menor número de restrições quando comparada com qualquer outra (x_{cj}, x_{zj}) que viola um maior número de restrições, independentemente da relação entre $V(x_{ci}, x_{zi})$ e $V(x_{cj}, x_{zj})$.

De seguida, descrevem-se as adaptações do FA que são necessárias para determinar as soluções do problema 1.2.11.

5.3.2 Extensão do FA para variáveis mistas

Para lidar com variáveis inteiras e contínuas simultaneamente foi introduzida no FA uma heurística simples baseada no arredondamento. Enquanto que o movimento das variáveis contínuas no espaço de procura é o usual, o movimento das variáveis inteiras é controlado

por passos de procura que tomam apenas valores inteiros. Isto é efetuado usando um operador que encontra o valor inteiro mais próximo de um dado valor real. Neste contexto, a posição de um pirilampo i é representado pelo ponto $(x_{ci}, x_{zi}) \in \Gamma_{x_c} \times \Gamma_{x_z}$, e um pirilampo j é mais brilhante que o pirilampo i se o ponto (x_{cj}, x_{zj}) é preferido quando comparado com (x_{ci}, x_{zi}), de acordo com as RAO. A geração aleatória da posição dos N pirilampos em $\Gamma_{x_c} \times \Gamma_{x_z}$, cada uma com n_c variáveis contínuas e n_d variáveis inteiras, é feita, coordenada a coordenada, da seguinte forma. Para $i = 1, \ldots, N$

$$\begin{aligned} x_{ci,s} &= lb_{c,s} + \epsilon_s^i (lb_{c,s} - ub_{c,s}) \quad \text{para } s = 1, \ldots, n_c \\ x_{zi,s} &= lb_{z,s} + \tau_s^i \qquad\qquad\qquad\quad \text{para } s = 1, \ldots, n_d \end{aligned} \tag{5.3.1}$$

onde $lb_{c,s}$ representa a coordenada s do vetor lb_c (de forma análoga, para $ub_{c,s}, lb_{z,s}, ub_{z,s}$), ϵ_s^i é um número gerado aleatoriamente do intervalo $[0, 1]$ e τ_s^i é um número aleatório selecionado do conjunto $\{0, 1, \ldots, (ub_{z,s} - lb_{z,s})\}$, $s = 1, \ldots, n_d$.

O movimento dos pirilampos na versão original do FA é feito de acordo com o brilho emitido por cada pirilampo e pelo grau de atratividade que se estabelece entre eles. Neste caso particular, estendendo esta ideia para variáveis inteiras mistas, um pirilampo i ($i = 2, \ldots, N-1$) é movido, na iteração k, em direção aos pirilampos mais brilhantes j, ($j = 1, \ldots, i-1$) da seguinte forma:

$$\begin{aligned} x_{ci}^{(k+1)} &= x_{ci}^{(k)} + \beta(x_{cj}^{(k)} - x_{ci}^{(k)}) + \alpha(\epsilon_i^c - 0.5)X_i \qquad\qquad \text{para } i = 1, \ldots, n_c \\ x_{zi}^{(k+1)} &= x_{zi}^{(k)} + RND\left[\beta(x_{zj}^{(k)} - x_{zi}^{(k)}) + \alpha(\epsilon_i^z - 0.5)Y_j\right] \quad \text{para } i = 1, \ldots, n_d \end{aligned} \tag{5.3.2}$$

onde o operador $RND[z]$ calcula o vetor com coordenadas inteiras mais próximas de z, α é um parâmetro aleatório definido pelo utilizador, usualmente um número do intervalo $[0, 1]$, e X_i e Y_j têm como objetivo escalonar o movimento das variáveis contínuas e inteiras aos conjuntos Γ_{x_c} e Γ_{x_z}, respetivamente. O parâmetro β é dado por

$$\beta = \beta_0 \exp\left(-\gamma \left\|(x_{cj}, x_{zj}) - (x_{ci}, x_{zi})\right\|_2^2\right), \tag{5.3.3}$$

e os parâmetros α e γ variam ao longo do processo iterativo de acordo com as equações (5.1.1) e (5.1.2). Neste algoritmo, é usado um movimento diferente para definir a posição temporária do pior pirilampo (x_{cN}, x_{zN}). A posição temporária do pior pirilampo

é colocada numa vizinhança do melhor pirilampo, através de um movimento aleatório componente a componente em torno de (x_{c1}, x_{z1}). Após todos os movimentos terem sido realizados, a posição temporária de cada pirilampo é verificada, e, se necessário, as suas coordenadas são projetadas em Γ_{x_c} e Γ_{x_z}. A seguir, a posição temporária de cada pirilampo é comparada com a sua posição inicial de acordo com as RAO, e a preferida será mantida para a iteração seguinte. Por fim, os pirilampos são ordenados de acordo com as RAO.

Na fase final do FAbRAO é incluída uma fase de intensificação. Um método que não recorre ao uso de derivadas da classe de procura local padrão, conhecido por *Hooke-and-Jeeves* (Hooke & Jeeves, 1961), é invocado a partir da posição do pirilampo mais brilhante. Este procedimento de procura local realiza $5N$ iterações, e foi estendido para lidar com variáveis contínuas e inteiras simultaneamente, usando o operador $RND[\cdot]$, tal como utilizado em (5.3.2).

As experiências computacionais realizadas com o FAbRAO são apresentadas na Secção 6.2.4.

Capítulo 6

Resultados computacionais

Neste capítulo apresentam-se os resultados computacionais obtidos relativamente aos estudos efetuados nos Capítulos 4 e 5. Na Secção 6.1 apresentam-se os resultados computacionais dos estudos apresentados no Capítulo 4 relativos à resolução de problemas de NLP e, na Secção 6.2, os resultados computacionais obtidos nos estudos apresentados no Capítulo 5 relativos à resolução de problemas de MINLP.

6.1 Extensão do FA para problemas de NLP

6.1.1 Problemas de NLP

Para avaliar o desempenho e eficácia das propostas apresentadas, foram usados dois conjuntos de problemas teste de referência de otimização global considerados de difícil resolução bem conhecidos na literatura. Apresenta-se a descrição destes problemas nas Tabelas 6.1 e 6.2.

Na Tabela 6.1 apresentam-se treze problemas de otimização global de referência, conhecida como coleção G01–G13[1], cuja dimensão das variáveis varia de 2 a 20, contendo

[1]J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A. Coello Coello, C. Deb, Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. TR, Nanyang T.U., Sept. 18, 2006.

características que são representativas da dificuldade de resolução destes problemas de otimização global. Na Tabela 6.2 apresenta-se um conjunto de 20 problemas de otimização global disponíveis em `http://www.ime.usp.br/~egbirgin/`[2]

Na Tabela 6.1, as duas primeiras colunas apresentam o nome do problema e a respetiva dimensão, seguida da solução ótima global conhecida, f^*, o tipo de função objetivo, o número de restrições de desigualdade lineares (LI) e não lineares (NI), e o número de restrições de igualdade lineares (LE) e não lineares (NE). Na última coluna apresenta-se a proporção ρ, que é uma estimativa da proporção entre a região admissível do problema e o tamanho de todo o espaço de procura. Na prática, ρ representa o grau de dificuldade do problema.

Tabela 6.1: Propriedades dos problemas de NLP da coleção G01–G13

Problema	n	f^*	Tipo de função	LI	NI	LE	NE	$\rho(\%)$
G01	13	-15.000000	Quadrática	9	0	0	0	0.011
G02	20	-0.803619	Não linear	1	1	0	0	99.99
G03	10	-1.000500	Não linear	0	0	0	1	0.002
G04	5	-30665.538672	Quadrática	0	6	0	0	52.123
G05	4	5126.496714	Não linear	2	0	0	3	0.000
G06	2	-6961.813876	Não linear	0	2	0	0	0.006
G07	10	24.306209	Quadrática	3	5	0	0	0.000
G08	2	-0.095825	Não linear	0	2	0	0	0.856
G09	7	680.630057	Não linear	0	4	0	0	0.521
G10	8	7049.248021	Linear	3	3	0	0	0.001
G11	2	0.749900	Quadrática	0	0	0	1	0.000
G12	3	-1.000000	Quadrática	0	9	0	0	4.779
G13	5	0.053942	Não linear	0	0	1	3	0.000

[2]E.G. Birgin, C.A. Floudas, J.M. Martínez, Global minimization using an augmented Lagrangian method with variable lower-level constraints, Technical Report MCDO121206, January 22, 2007.

Tabela 6.2: Outros problemas NLP

Problema	n	f^*	Tipo de função	LI	NI	LE	NE
1	5	0.029213	Polinomial	0	0	0	3
2a	9	-400	Linear	0	2	3	1
2b	9	-600	Linear	0	2	3	1
2c	9	-750	Linear	0	2	3	1
2d	10	-400	Linear	0	2	4	1
3a	6	-0.38880	Linear	0	1	0	4
3b	2	-0.38881	Não linear	0	1	0	0
4	2	-6.6666	Linear	0	1	0	0
5	3	201.16	Linear	0	0	0	3
6	2	376.29	Linear	0	1	0	0
7	2	-2.8284	Linear	2	2	0	0
8	2	-118.70	Polinomial	1	1	0	0
9	6	-13.402	Não linear	3	0	3	0
10	2	0.74178	Linear	0	2	0	0
11	2	-0.5000	Quadrática	0	1	0	0
12	2	-16.739	Quadrática	0	0	0	1
13	3	189.35	Não linear	0	0	1	1
14	4	-4.5142	Não linear	2	0	1	0
15	3	0.0000	Constante	0	0	1	2
16	5	0.70492	Linear	0	0	1	2

6.1.2 Resultados computacionais da EXFA

Nesta secção apresentam-se os resultados computacionais obtidos pela extensão do FA, denotada por EXFA, com a implementação da OGC e das RAO, na resolução dos problemas da coleção G01–G13, e a sua comparação com outros métodos estocásticos baseados em populações existentes na literatura. Para distinguir os esquemas de ordenação dos pontos da população, a extensão do FA que usa as regras de admissibilidade e otimalidade $R_1 - R_4$, RAO, é denotada por EXFA1, e a extensão do FA que usa a ordenação global competitiva,

OGC, por EXFA2. No contexto da implementação da EXFA1, também é proposto um movimento direto (em vez de (4.1.2)), onde todos os pirilampos se movem em direção ao mais brilhante. Esta variante é denotada por EXFA#.

Os algoritmos foram codificados na linguagem de programação Matlab, versão 8.01 (R2013a), e foram feitas 20 execuções para cada um dos problemas. O tamanho da população usada foi de $N = 40$ pontos e o critério de paragem adotado foi:

$$(|f^* - f_{\text{best}}| \leq 10^{-6} \quad \text{e} \quad V_2(f_{\text{best}}) \leq 10^{-6}) \quad \text{ou} \quad k > k_{\max},$$

em que f^* representa a solução ótima global conhecida, f_{best} é o valor da função objetivo do melhor ponto da população, k denota o contador de iteração e $k_{\max}$ é o número máximo de iterações permitidas $k_{\max} = 5000$ iterações.

Os valores iniciais definidos para o cálculo dinâmico dos parâmetros α e β, dados por (5.1.1) e (5.1.2), respetivamente, foram: $\alpha_{\max} = 0.9$, $\alpha_{\min} = 0.01$, $\gamma_{\max} = 100$ e $\gamma_{\min} = 0.001$. As restrições de igualdade $h_j(x) = 0,\ j = 1, \ldots, p$, foram convertidas em restrições de desigualdade usando $|h_j(x)| - \varepsilon \leq 0$, onde $\varepsilon > 0$ é uma tolerância de violação muito pequena. Nos problemas G05, G11 e G13, a tolerância usada na relaxação das restrições de igualdade foi de $\varepsilon = 10^{-4}$, e nos restantes problemas foi utilizado um valor de $\varepsilon = 10^{-6}$. No procedimento de procura local, foi utilizado um comprimento de passo de $\delta = 10^{-5}$, exceto nos problemas assinalados com $*$, onde foi usado $\delta = 10^{-2}$. O número máximo de iterações permitidas neste procedimento foi de $LSit_{\max} = 10$.

Na Tabela 6.3 são apresentados os resultados obtidos pelas extensões EXFA, nomeadamente EXFA1#, EXFA1 e EXFA2, onde f_{best} é a melhor solução obtida, f_{avg}, f_{med} e $St.Dev.$ representam, respetivamente, a média, a mediana e o desvio padrão das soluções obtidas, e f_{worst} designa a pior solução encontrada nas 20 execuções dos algoritmos.

Pela análise da tabela, verifica-se que os algoritmos propostos foram capazes de encontrar soluções admissíveis em todas as execuções efetuadas para todos os problemas. Isto deve-se ao facto de os algoritmos darem preferência às soluções admissíveis. Verifica-se também que as EXFA foram capazes de obter muito bons resultados em quase todos os

problemas. Nos problemas G01, G03, G08, G11 e G12, a EXFA#, EXFA1 e EXFA2 atingiram a solução ótima global conhecida em todas as execuções efetuadas. Consequentemente, os valores da média, da mediana e do pior valor obtido são idênticos a f^*, sendo o desvio padrão igual a 0. Nos problemas G05 e G13, a solução ótima obtida pela EXFA1 é inferior a f^*, uma vez que na relaxação das restrições de igualdade foi usada uma tolerância de 10^{-4}. Para os problemas G04, G06, G07 e G09, as EXFA1 e EXFA2 produziram resultados muito competitivos, uma vez que obtiveram soluções muito próximas das soluções ótimas conhecidas. Por outro lado, os problemas G02 e G10 ficaram aquém do desejável. Em geral, os melhores desempenhos obtiveram-se nas implementações de EXFA1 e EXFA2.

Tabela 6.3: Resultados obtidos por EXFA1#, EXFA1 e EXFA2

Problema	*EXFA*	f_{best}	f_{avg}	f_{med}	*St.Dev*	f_{worst}
	EXFA1#	-15.0000	-15.0000	-15.0000	0.0000	-15.0000
G01	EXFA1	-15.0000	-15.0000	-15.0000	0.0000	-15.0000
	EXFA2	-15.0000	-15.0000	-15.0000	0.0000	-15.0000
	EXFA1#	-0.4373	-0.2968	-0.2841	-0.0557	-0.2405
G02	EXFA1	-0.4048	-0.2941	-0.2911	0.0398	-0.2414
	EXFA2	-0.4799	-0.3458	-0.3330	0.0631	-0.2488
	EXFA1#	-1.0005	-1.0005	-1.0005	0.0000	-1.0005
G03	EXFA1	-1.0005	-1.0005	-1.0005	0.0000	-1.0005
	EXFA2	-1.0005	-1.0005	-1.0005	0.0000	-1.0004
	EXFA1#	-30665.5385	-30538.1527	-30546.2560	96.7170	-30372.8244
G04	EXFA1	-30665.5386	-30663.8601	-30665.5382	7.5011	-30631.9925
	EXFA2	-30665.5385	-30660.8451	-30665.5382	14.4310	-30611.1510
	EXFA1#*	5126.7259	5402.6462	5274.411	5279.0658	5986.442138
G05	EXFA1	5126.3617	5157.5428	5128.489	66.0467	5401.545758
	EXFA2	5126.5175	5128.7370	5128.4245	2.0577	5133.3343

Continua na próxima página

Tabela 6.3 *(Continuação da página anterior)*

Problema	*EXFA*	f_{best}	f_{avg}	f_{med}	*St.Dev*	f_{worst}
	EXFA1#*	-6961.8101	-6961.7905	-6961.7905	0.0130	-6961.7553
G06	EXFA1	-6961.8016	-6961.7747	-6961.7747	0.0172	-6961.7347
	EXFA2*	-6961.5405	-6959.8675	-6960.2069	1.2835	-6956.3551
	EXFA1#*	24.6203	26.0814	26.1657	0.7935	27.1457
G07	EXFA1	24.3571	24.5827	24.5456	0.1967	25.2044
	EXFA2	24.3273	24.3772	24.3663	0.0311	24.4368
	EXFA1#	-0.095825	-0.095825	-0.095825	0.0000	-0.095825
G08	EXFA1	-0.095825	-0.095825	-0.095825	0.0000	-0.095824
	EXFA2	-0.095825	-0.095825	-0.095825	0.0000	-0.095825
	EXFA1#	680.6445	680.7663	680.7867	0.0676	680.8671
G09	EXFA1	680.6348	680.6895	680.6659	0.0468	680.7948
	EXFA2	680.6328	680.6869	680.6821	0.0414	680.7852
	EXFA1#	7069.4263	8183.2192	7785.5967	1053.5456	10276.0255
G010	EXFA1*	7125.2110	7552.1114	7364.5715	486.5234	8872.8741
	EXFA2*	7073.7810	7171.9798	7145.6712	107.9335	7542.8818
	EXFA1#	0.7499	0.7499	0.7499	0.0000	0.749910
G11	EXFA1	0.7499	0.7499	0.7499	0.0000	0.749900
	EXFA2*	0.7499	0.7499	0.7499	0.0000	0.749900
	EXFA1#	-1.0000	-1.0000	-1.0000	0.0000	-1.0000
G12	EXFA1	-1.0000	-1.0000	-1.0000	0.0000	-1.0000
	EXFA2	-1.0000	-1.0000	-1.0000	0.0000	-1.0000
	EXFA1#*	0.054106	0.325382	0.435195	0.2509	0.817849
G13	EXFA1	0.053556	0.283400	0.434445	0.1924	0.444500
	EXFA2	0.053944	0.053960	0.053951	0.0000	0.054017

* Significa o comprimento do passo de $\delta = 10^{-2}$ no procedimento de procura local.

De seguida, foram comparados os resultados de EXFA1 e EXFA2 com os de cinco mé-

todos estocásticos de otimização global baseados em populações, disponíveis na literatura. Apresenta-se uma descrição sucinta destes métodos. No método proposto por Koziel e Michalewicz (1999), é incorporado um mapeamento homomórfico (do inglês, *homomorphous mapping*) entre um cubo n-dimensional e a região admissível do problema. Runarsson e Yao (2000) apresentam os resultados obtidos com o método original da ordenação estocástica no contexto de uma estratégia evolutiva. Em (Farmani & Wright, 2003), é usada uma formulação do problema com uma função de penalidade auto-adaptativa. Os resultados apresentados em (Silva et al., 2011) são obtidos com um método de penalidade adaptativo com variantes dinâmicas do DE, enquanto que Tessema e Yen (2006) usam uma função de penalidade auto-adaptativa no contexto de um GA. A Tabela 6.4 apresenta os resultados obtidos por EXFA1 e EXFA2 e pelos métodos agora referidos.

Tabela 6.4: Resultados obtidos por EXFA1, EXFA2 e outros métodos estocásticos.

Problema	EXFA1	EXFA2	[a]	[b]	[c]	[d]	[e]
G01	-15.0000	-15.0000	-14.7082	-15.0000	-15.0000	-15	-15.000
G02	-0.4048	-0.4799	-0.79671	-0.803515	-0.802970	-0.8036	-0.803202
G03	-1.0005	-1.0005	-0.9989	-1.0000	-1.0000	-1.0	-1.000
G04	-30665.539	-30665.539	-30655.3	-30665.539	-30665.500	-30665.5	-30665.401
G05	5126.3617	5126,5175	n.d.	5126.497	5126.989	5126.4981	5126.907
G06	-6961.8016	-6961,5405	-6342.6	-6961.814	-6961.800	-6961.8	-6961.046
G07	24.3571	24,3273	24.826	24.307	24.480	24.306	24.838
G08	-0.095825	-0.095825	-0.089157	-0.095825	-0.095825	-0.09582	-0.095825
G09	680.6348	680.6328	681.16	680.630	680.640	680.63	680.773
G10	7125.2110	7073.7810	8163.6	7054.316	7061.340	7049.25	7069.981
G11	0.749900	0.749900	0.75	0.750	0.75	0.75	0.749
G12	-1.000000	-1.000000	-0.999135	-1.000000	n.d.	n.d.	-1.000000
G13	0.053556	0.053944	0.557	0.053957	n.d.	n.d.	0.053941

[a] resultados em (Koziel & Michalewicz, 1999), [b] resultados em (Runarsson & Yao, 2000),

[c] resultados em(Farmani & Wright, 2003), [d] resultados em (Silva et al., 2011), [e] resultados em (Tessema & Yen, 2006)

n.d.- Não disponível

Pela análise da Tabela 6.4 verifica-se que o método baseado na ordenação estocástica

apresentado em (Runarsson & Yao, 2000) produziu muito bons resultados. Contudo, para o problema G10, este método só foi capaz de obter soluções admissíveis apenas em 6 das 30 execuções. Em (Farmani & Wright, 2003), o algoritmo foi executado 20 vezes para cada problema, e apenas 17 das 20 soluções encontradas são admissíveis para o problema G10, e 9 para o problema G05. Por outro lado, a EXFA1 e EXFA2 obtiveram soluções admissíveis para todos os problemas e em todas as execuções dos algoritmos. Assim, em geral, os resultados computacionais demonstram que a EXFA1 e EXFA2 são bastante competitivas quando comparadas com outros métodos estocásticos de otimização global.

6.1.3 Resultados computacionais com o FPAA

Nesta secção apresentam-se as experiências computacionais realizadas com o FPAA, que implementa um algoritmo híbrido da função de penalidade auto-adaptativa, com um procedimento de intensificação local, Algoritmo 8, para calcular a solução de problemas de otimização global com restrições genéricas. Este método foi codificado na linguagem de programação Matlab, Versão 8.1.0.604 (R2013a). Para testar a convergência do algoritmo, foram realizadas algumas experiências preliminares para analisar o efeito dos valores de alguns parâmetros.

Na primeira fase, é analisado o desempenho da função auto-adaptativa quando é usado o FA e as metaheurísticas descritas na Secção 2.3, para calcular o minimizante global da função de penalidade $\phi(x)$ em X, na resolução dos problemas G01–G13, descritos na Tabela 6.1.

As metaheurísticas FA, jDE, PSO, CMA-ES e ABC são testadas, usando os valores dos parâmetros como sugerido nos artigos (Costa, Rocha, Francisco, & Fernandes, 2014; Brest, Greiner, et al., 2006; M. Ali & Kaelo, 2008; Hansen & Ostermeier, 2001; Karaboga & Basturk, 2007). As restrições de igualdade $h_i(x) = 0,\ i = 1, \ldots, p$, foram convertidas em restrições de desigualdade usando $|h_j(x)| - \varepsilon \leq 0$, com $\varepsilon = 10^{-4}$.

Experiências preliminares com o FA mostraram que são obtidas soluções de boa qualidade no movimento dos pontos, através de (4.2.8), quando $\beta_0 = 1$ e $\nu = 2$, cuja norma

utilizada no cálculo de β é a norma-2. Os parâmetros iniciais para o cálculo iterativo de α e γ definidos em (5.1.2) e (5.1.2), respetivamente, são os seguintes: $\gamma_{\min} = 0.1$, $\gamma_{\max} = 10$, $\alpha_{\min} = 0.01$ e $\alpha_{\max} = 0.5$. As distribuições Uniforme e de Lévy foram também testadas no termo aleatório da equação que descreve o movimento dos pirilampos. Caso nada seja dito em contrário, o tamanho da população é de $N = 50$. No contexto da definição do ponto de referência z e do valor da função de penalidade em (4.2.4), um ponto x é considerado admissível se $V_1(x) \leq 10^{-8}$.

Os parâmetros considerados para o algoritmo jDE, tal como sugerido por Brest, Greiner, et al. (2006), foram: $F_l = 0.1, F_u = 0.9$ e $\tau_1 = \tau_2 = 0.1$. No algoritmo PSO, foram usados os parâmetros usualmente adotados na literatura: $c_1 = 1.5, c_2 = 2, w_{\max} = 0.95$ e $w_{\min} = 0.4$. O parâmetro $v_{\max}$ foi tomado consoante o tamanho de X da seguinte forma: $v_{s,max} = (ub_s - lb_s), s = 1, \ldots, n$.

No algoritmo CMA-ES, alguns parâmetros importantes adotados foram: $\lambda = m, \mu = \frac{\lambda}{n_\mu}$ (onde n_μ é igual a 2 ou m, dependendo do problema) e os pesos w_i são os valores normalizados de $w_i = \log(\mu+0.5) - \log(i), i = 1, \ldots, \mu$. Na implementação deste algoritmo, os valores de alguns outros parâmetros para a atualização da covariância e atualização do tamanho do passo seguem as estratégias propostas em (Hansen, 2016).

No algoritmo ABC, os parâmetros utilizados foram $\alpha_P = 0.9, \beta_P = 0.1$ e limit = 100. Nesta comparação, o critério de paragem dos algoritmos foi de 200000 avaliações da função.

Os resultados obtidos encontram-se nas Tabelas 6.5 e 6.6, onde f^* é a melhor solução conhecida, 'dist.' é a distribuição utilizada no termo aleatório da equação do movimento do FA, f_{best}, f_{avg} e St.dev representam a melhor, a média e o desvio padrão das soluções obtidas pelos algoritmos após 20 execuções de cada problema. A Tabela 6.5 apresenta a melhor solução obtida e a Tabela 6.6 apresenta os valores da média e do desvio padrão. Os melhores valores obtidos (que são designados por vitórias) em ambas as tabelas são apresentados a negrito e os empates em itálico.

Pela análise das tabelas, verifica-se que o FA apresenta um grande número de vitórias em comparação com as outras metaheurísticas, quando é usado o critério f_{avg} e St.dev.. Relativamente ao critério f_{best}, o FA tem duas vitórias e dois empates contra três vitórias

Tabela 6.5: Melhores soluções obtidas pelas metaheurísticas FA, jDE, PSO, CMA-ES e ABC

P	f^*	FA		jDE	PSO	CMA-ES	ABC
		dist.	f_{best}	f_{best}	f_{best}	f_{best}	f_{best}
G01	-15.00000	Unif.	-14.999998	-14.999999	-14.984722	-14.999999	**-15.000000**
G02	-0.803619	Lévy	-0.563374	-0.211660	-0.454244	**-0.676941**	-0.199999
G03	-1.00000	Lévy	-1.000408	-0.983895	-0.539194	**-1.000443**	-0.563430
G04	-30665.539	Unif.	-30665.511	-30665.535	-30665.533	-30665.536	**-30665.539**
G05	5126.4981	Lévy	**5126.3023**	5126.7775	5127.8771	5517.3689	5127.2853
G06	-6961.8139	Unif.	-6961.7392	-6961.8119	-6958.5305	**-6961.8152**	-6961.8151
G07	24.306209	Lévy	26.675466	**24.331042**	28.233673	24.639231	25.253567
G08	-0.095825	Lévy	*-0.095825*	*-0.095825*	*-0.095825*	*-0.095825*	*-0.095825*
G09	680.63006	Unif	680.64725	**680.63008**	680.94396	680.80981	681.77911
G10	7049.3307	Unif.	7089.9239	**7063.7817**	7675.2213	7139.4722	7353.5582
G11	0.750000	Unif.	0.749901	*0.749900*	0.749906	0.749902	*0.749900*
G12	-1.000000	Unif.	*-1.000000*	-0.999944	*-1.000000*	*-1.000000*	-0.999993
G13	0.0539498	Lévy	**0.074739**	0.447568	0.088210	0.229387	0.998928

e dois empates em comparação com o jDE, dois empates com o PSO, três vitórias e dois empates com o CMA-ES, e duas vitórias e dois empates com o ABC. Assim, em geral, a técnica de penalidade auto-adaptativa, simples e fácil de codificar, quando implementada em metaheurísticas para resolver BCOP, é eficaz na procura de soluções ótimas globais de problemas COP contínuos.

Numa segunda fase, é analisado o comportamento prático do algoritmo FPAA, Algoritmo 8, quando comparado com outras funções de penalidade adaptativas recentemente combinadas com métodos de otimização global estocásticos baseados em populações de pontos (M. Ali & Zhu, 2013; Silva et al., 2011; Barbosa & Lemonge, 2008; Tessema & Yen, 2006, 2009) e com o algoritmo ABC modificado baseado nas regras de admissibilidade e otimalidade para o tratamento das restrições (Karaboga & Akay, 2011).

Tabela 6.6: Valores de f_{avg} e St.dev. obtidos pelas metaheurísticas FA, jDE, PSO, CMA-ES e ABC

P	FA		jDE		PSO		CMA-ES		ABC	
	f_{avg}	St.dev.	f_{avg}	St.dev.	f_{avg}	St.dev.	f_{avag}	St.dev.	f_{avg}	St.dev.
G01	-14.999941	5.6e-05	-14.997003	5.6e-03	-14.884585	4.5e-02	**-14.999987**	**1.1e-05**	-14.999712	3.1e-04
G02	**-0.507532**	2.6e-02	-0.169305	1.5e-02	-0.403995	3.1e-02	-0.472315	1.4e-01	-0.170567	**1.3e-02**
G03	**-0.996849**	**4.7e-03**	-0.128970	2.9e-01	-0.311851	1.3e-01	-0.881072	1.9e-01	-0.189673	1.6e-01
G04	-30623.060	3.6e+01	-30662.870	3.8e+00	-30665.272	*2.0e-01*	**-30665.387**	*2.0e-01*	-30614.971	1.1e+02
G05	5176.6804	7.6e+01	5198.1389	1.3e+02	5323.8485	2.7e+02	5564.5457	4.5e+02	**5144.9478**	**1.8e+01**
G06	**-6961.4581**	**1.5e-01**	-6444.5577	3.4e+02	-6483.5724	3.3e+02	-6422.5097	4.9e+02	-6871.5246	1.6e+02
G07	32.101364	3.1e+00	37.379619	6.7e+00	36.541811	4.4e+00	**24.813485**	**3.7e-01**	29.761456	1.2e+01
G08	*-0.095825*	*2.8e-17*	-0.095003	1.4e-03	*-0.095825*	*2.8e-17*	*-0.095825*	*2.8e-17*	-0.095680	3.2e-04
G09	**680.69347**	**1.9e-02**	798.55773	7.5e+01	686.49823	4.8e+00	681.03125	1.0e-01	848.86188	9.9e+01
G10	**7119.2548**	**1.7e+01**	7184.6003	2.2e+02	8542.6951	3.6e+02	7670.4043	5.3e+02	7594.8362	2.3e+02
G11	**0.749904**	**1.9e-06**	0.956143	9.2e-02	0.830885	7.4e-02	0.749948	9.2e-02	0.749905	4.0e-06
G12	*-1.000000*	8.1e-11	-0.998962	2.2e-03	-0.999854	1.5e-04	*-1.000000*	**0.0e+00**	-0.999748	1.7e-04
G13	**0.610415**	1.6e-01	0.842288	2.0e-01	0.653038	2.4e-01	0.753785	2.1e-01	0.999883	**2.5e-04**

Assim, primeiro comparou-se o Algoritmo 8, FPAA, com (M. Ali & Zhu, 2013), onde é proposta uma função de penalidade auto-adaptativa baseada no algoritmo DE, sem necessidade de definição de parâmetros de penalidade. Nesta comparação foram utilizados os 13 problemas da coleção-G descritos na Tabela 6.1. Durante o processo de invocação do procedimento de intensificação local, alguns parâmetros foram escolhidos em função do problema a resolver, nomeadamente, p_m, F_b e F_o, com o objetivo de obter melhores desempenhos. No Algoritmo 8 considerou-se $k_{\max} = 600$, $\epsilon = 10^{-5}$, $\eta = 10^{-6}$ e, no Algoritmo 7, $tol_v = 10^{-8}$ e uma população de $N = 50$ pontos. Além disso, considerou-se como critério de paragem 50000 avaliações da função, como utilizado em (M. Ali & Zhu, 2013). Os resultados obtidos nestes 13 problemas encontram-se na Tabela 6.7, onde 'best_E', 'worst_E' e 'St.dev._E' representam o valor do melhor erro, $|f_{\text{best}} - f^*|$, o pior erro e o desvio padrão dos valores dos erros obtidos em 100 execuções dos algoritmos, respetivamente. Nestas experiências, foram testados diferentes conjuntos de valores para os parâmetros F_b e F_o, assim como as distribuições Uniforme e de Lévy no termo aleatório da equação que descreve o movimento dos pontos. Os valores encontrados na tabela correspondem ao conjunto de resoluções que produziram melhores resultados.

Apesar dos resultados produzidos pelo FPAA, Algoritmo 8, ao resolver este conjunto de problemas serem satisfatórios, estes não são superiores aos reportados em (M. Ali & Zhu, 2013), exceto nos problemas G01 e G13, e no problema G12, onde se verificam resultados idênticos. Pela análise da tabela verifica-se que um maior número de avaliações da função é certamente necessária para alguns problemas. No entanto, verificou-se que o procedimento de intensificação local permitiu encontrar soluções de boa qualidade, mas fez aumentar o esforço computacional. Isto foi confirmado durante outras experiências realizadas no presente estudo, que a seguir se relatam.

De seguida, é apresentada a comparação dos resultados obtidos por FPAA, Algoritmo 8, com os resultados de Silva et al. (2011), designado por DUVDE+APM. Neste método, o método de penalidade adaptativo APM (do inglês, *Adaptive Penalty Method*), proposto em (Barbosa & Lemonge, 2008), é estendido e aplicado com o uso dinâmico de variantes do DE (do inglês, *Dynamic use of diferential evolution variants*). Barbosa e Lemonge (2008)

Tabela 6.7: Resultados obtidos pelo FPAA e os apresentados em (M. Ali & Zhu, 2013)

		resultados obtidos pelo FPAA						resultados em (M. Ali & Zhu, 2013)		
P	f^*	dist.	F_b	F_o	best$_E$	worst$_E$	St.dev.$_E$	best$_E$	worst$_E$	St.dev.$_E$
G01	-15.0000000	Unif.	1	1	3.000e-11	5.100e-10	1.255e-10	1.358e-06	9.166e-06	2.178e-06
G02	-0.803619	Lévy	2.5	0.8	1.080e-03	4.285e-02	1.021e-02	3.836e-05	3.909e-02	1.117e-02
G03	-1.000000	Lévy	0.1	1.5	2.822e-04	1.000e+00	1.970e-01	4.354e-09	7.854e-01	1.537e-01
G04	-30665.539	Unif.	2.5	0.8	3.285e-04	3.491e-04	5.875e-06	1.035e-08	3.250e-06	7.105e-07
G05	5126.49810	Lévy	1.5	0.8	1.352e-02	6.832e+01	2.045e+01	1.018e-10	3.468e+02	9.842e+01
G06	-6961.81388	Unif.	1.2	0.8	5.688e-05	6.493e-04	1.727e-04	1.373e-10	1.291e-10	3.129e-12
G07	24.306209	Lévy	2.5	0.8	3.260e-02	2.521e-01	6.163e-02	1.846e-05	1.467e-04	3.029e-05
G08	-0.095825	Lévy	1	1	4.142e-08	4.142e-08	0	5.008e-11	5.008e-11	0
G09	680.630057	Lévy	0.01	0.8	5.657e-03	4.292e-02	9.889e-03	2.16e-12	2.16e-12	0
G10	7049.33070	Unif.	2.5	0.8	5.449e+01	6.484e+02	1.770e+02	7.900e+00	3.731e+01	8.385e+00
G11	0.7500000	Unif.	1.5	0.8	7.642e-05	7.642e-05	4.850e-06	0	0	0
G12	-1.0000000	Unif.	1	1	0	0	0	0	0	0
G13	0.0539498	Lévy	0.01	0.8	3.947e-02	8.575e-01	1.796e-01	3.851e-01	9.107e-01	1.433e-01

Para este conjunto de problemas foi considerado $p_m = 1$, exceto no problema G13, onde $p_m = 0.4$.

usam informações da população, como a média dos valores da função objetivo e o nível de violação de cada restrição, em cada iteração, para definir o parâmetro de penalidade. Para a comparação de resultados, foram usados os problemas G01-G11. Os resultados obtidos encontram-se na Tabela 6.8, onde são apresentados a melhor solução obtida, f_{best}, a média, f_{avg}, e o desvio padrão, St.dev., das soluções obtidas em 20 execuções do algoritmo. O tamanho da população é de $N = 50$, e o critério de paragem utilizado foi de 350000 avaliações da função. Releva-se que os resultados do DUVDE+APM são retirados de (Silva et al., 2011). Da análise dos resultados na Tabela 6.8, conclui-se que o FPAA é capaz de produzir soluções comparáveis e de alta qualidade, quando é permitido um número elevado de avaliações da função.

A Tabela 6.9 mostra a comparação dos resultados obtidos por FPAA e pelos apresentados em (Tessema & Yen, 2006), onde foi usada uma função de penalidade auto-adaptativa

Tabela 6.8: Resultados obtidos pelo FPAA e pelos apresentados em (Silva et al., 2011)

	resultados obtidos pelo FPAA			resultados em (Silva et al., 2011)		
P	f_{best}	f_{avg}	St.dev.	f_{best}	f_{avg}	St.dev.
G01	-15.000000	-15.000000	0.000e+00	-15	-12.5	2.37254e+00
G02	-0.803603	-0.787892	1.379e-02	-0.8036	-0.7688	3.568e-02
G03	-0.980341	-0.962513	8.508e-03	-1.0	-0.2015	3.4508e-01
G04	-30665.538672	-30665.538672	1.866e-11	-30665.5	-30665.5	0
G05	5125.273729	5125.105038	6.083e-02	5126.4981	5126.4981	0
G06	-6961.813876	-6961.813876	9.331e-13	-6961.8	-6961.8	0
G07	24.312256	24.376587	5.044e-02	24.306	30.404	2.156839e+01
G08	-0.095825	-0.095825	2.848e-17	-0.09582	-0.09582	0
G09	680.630123	680.630848	4.258e-04	680.63	680.63	3e-05
G10	7103.509964	7279.735151	1.375e+02	7049.25	7351.17	5.2562430e+02
G11	0.749900	0.749900	8.050e-13	0.75	0.98749	5.590e-02

No problema G03 foi usada a distribuição Uniforme, $p_m = 0.1$, $F_b = 0.01$ e $F_o = 0.8$. Para os outros problemas, os valores usados foram os previamente definidos.

no contexto de um GA. Em (Tessema & Yen, 2006, 2009), o valor normalizado da função objetivo e o valor da soma das violações de restrições normalizadas são combinados para definir uma nova função de avaliação. Nos dois trabalhos, é usado um GA no método de penalidade adaptativo. Para comparação dos resultados, foram usados os problemas G01-G13, cujo tamanho da população utilizado foi de $N = 100$, o critério de paragem foi de 500000 avaliações da função, e a tolerância nas restrições de igualdade foi de $\delta = 10^{-4}$ (tal como em Tessema e Yen (2006)). A tabela apresenta a melhor solução obtida, f_{best}, a média, f_{avg}, e o desvio padrão, St.dev., das soluções obtidas em 50 execuções dos algoritmos para cada problema. Os resultados obtidos neste estudo são, em geral, superiores aos apresentados por Tessema e Yen (2006), donde se reiteram as conclusões anteriores.

A seguir, é feita a comparação entre os resultados obtidos por FPAA, Algoritmo 8, com o algoritmo ABC modificado baseado nas regras de admissibilidade e otimalidade para o tratamento de restrições, regras R_1-R_3 descritas na Secção 3.1, e um esquema de seleção

Tabela 6.9: Resultados obtidos pelo FPAA e os apresentados em (Tessema & Yen, 2006)

	resultados obtidos pelo FPAA			resultados em (Tessema & Yen, 2006)		
P	f_{best}	f_{avg}	St.dev.	f_{best}	f_{avg}	St.dev.
G01	-15.000000	-15.000000	0.000e+00	-15.000	-14.552	7.0e-01
G02	-0.803585	-0.797191	4.850e-03	-0.803202	-0.755798	1.33210e-01
G03	-0.976735	-0.940689	1.024e-02	-1.000	-0.964	3.01e-01
G04	-30665.538672	-30665.538672	7.371e-12	-30665.401	-30659.221	2.043e+00
G05	5125.031908	5125.103381	3.639e-02	5126.907	5214.232	2.47476e+02
G06	-6961.813876	-6961.813876	9.214e-13	-6961.046	-6953.061	5.876e+00
G07	24.309466	24.347169	1.478e-02	24.838	27.328	2.172e+00
G08	-0.095825	-0.095825	5.624e-17	-0.095825	-0.095635	1.055e-03
G09	680.630196	680.631087	3.438e-04	680.773	681.246	3.22e-01
G10	7050.095847	7149.949024	4.839e+01	7069.981	7238.964	1.37773e+02
G11	0.749900	0.749900	1.125e-16	0.749	0.751	2e-03
G12	-1.000000	-1.000000	0.000e+00	-1.000000	-0.999940	1.41e-04
G13	0.353983	0.628807	1.136e-01	0.053941	0.286270	2.75463e-01

Estes resultados foram obtidos com os valores de p_m, F_b e F_o definidos na Tabela 6.7.

probabilístico baseado nos valores da função de avaliação se as soluções são admissíveis, e nos valores da violação se as soluções são não admissíveis (Karaboga & Akay, 2011). Nesta comparação foram consideradas as seguintes condições: $\delta = 10^{-4}$, população de $N = 40$ pontos, 30 execuções do algoritmo e um máximo de 240000 avaliações da função como critério de paragem (tal como em (Karaboga & Akay, 2011)). A partir dos resultados da Tabela 6.10 é possível concluir que o FPAA tem um desempenho semelhante ao do algoritmo ABC modificado em nove problemas, é melhor nos problemas G06 e G13, e é pior nos problemas G05 e G10.

Na terceira fase, com o objetivo de comparar o algoritmo proposto FPAA, que é um algoritmo híbrido da função de penalidade auto-adaptativa com um procedimento de intensificação local, com outros métodos baseados em funções de penalidade, foram usados

Tabela 6.10: Resultados do FPAA e os apresentados em (Karaboga & Akay, 2011)

	resultados obtidos pelo FPAA			resultados em (Karaboga & Akay, 2011)†		
P	f_{best}	f_{avg}	St.dev.	f_{best}	f_{avg}	St.dev.
G01	-15.000000	-15.000000	0.00e+00	-15.000	-15.000	0.000
G02	-0.803470	-0.778942	1.348e-02	-0.803598	-0.792412	1.2e-02
G03	-1.000278	-0.999522	3.645e-04	-1.000	-1.000	0.000
G04	-30665.538673	-30665.538672	2.220e-11	-30665.539	-30665.539	0.000
G05	5153.670975	5451.215691	2.490e+02	5126.484	5185.714	7.5358e+01
G06	-6961.813876	-6961.813876	1.850e-12	-6961.814	-6961.813	2e-03
G07	24.320519	24.757232	5.157e-01	24.330	24.473	1.86e-01
G08	-0.095825	-0.095825	2.823e-17	0.095825	0.095825	0.000
G09	680.631787	680.641211	1.142e-02	680.634	680.640	4e-03
G10	7072.574892	7221.442900	9.565e+01	7053.904	7224.407	1.33870e+02
G11	0.749900	0.749900	1.129e-16	0.750	0.750	0.000
G12	-1.000000	-1.000000	0.000e+00	1.000	1.000	0.000
G13	0.056841	0.659425	1.764e-01	0.760	0.968	5.5e-02

† O número de dígitos utilizados na apresentação dos resultados são os apresentados no artigo citado.

20 problemas disponíveis em `http://www.ime.usp.br/~egbirgin/`[3].

A comparação envolve os resultados apresentados por Costa, Rocha, e Fernandes (2014), denotado por HAL-iAFS, onde um método baseado numa função Lagrangeana aumentada foi combinado com um algoritmo estocástico baseado em populações de pontos, designado de algoritmo *Artificial Fish Swarm Algorithm*, e os reportados por Di Pillo et al. (2012), onde é implementado o algoritmo determinístico DIRECT com uma função de penalidade exata não diferenciável para resolver globalmente os subproblemas de BCOP, denotado por EPGO (do inglês, *Exact Penalty Global Optimization*).

Na Tabela 6.11, apresentam-se os resultados, para 30 execuções do algoritmo para cada

[3]E.G. Birgin, C.A. Floudas, J.M. Martínez, Global minimization using an augmented Lagrangian method with variable lower-level constraints, Technical Report MCDO121206, January 22, 2007.

problema (no caso de algoritmos estocásticos), onde: f_{best} é a melhor solução encontrada, f_{med} é a mediana entre as 30 soluções, e 'n.f.e.(b)' é o número de avaliações da função necessárias para encontrar a melhor solução, f_{best}. A solução, 'sol.', o número de avaliações da função, 'n.f.e.', reportados em (Di Pillo et al., 2012), e a melhor solução conhecida na literatura, f^*, são também apresentados na tabela. Nesta experiência foi usada uma população de $N = \min\{5n, 50\}$, onde n é a dimensão do problema. O critério de paragem é o apresentado no Algoritmo 8 com $\epsilon = 10^{-5}$, $\eta = 10^{-6}$ e $k_{\max} = 600$ (à semelhança dos valores utilizados em (Costa, Rocha, & Fernandes, 2014)). Na resolução deste conjunto de problemas considerou-se $p_m = 0.5$, $F_b = 1$ e $F_o = 1$, exceto para o Problema 1, onde $p_m = 1$, $F_b = 2.5$ e $F_o = 0.8$ e o problema 13, onde $p_m = 1$, $F_b = 0.2$ e $F_o = 0.8$.

Quando se comparam os resultados obtidos pelo FPAA com os de HAL-iAFS apresentados em (Costa, Rocha, & Fernandes, 2014)), conclui-se que a qualidade das soluções obtidas são semelhantes, apesar de ser necessário um grande número de avaliações da função para obter tais soluções. Por outro lado, a qualidade das soluções obtidas com o FPAA é superior às do EPGO apresentadas em (Di Pillo et al., 2012).

Neste conjunto de experiências verificou-se que o Algoritmo FPAA proposto, que usa uma função de penalidade auto-adaptativa no contexto do FA, é bastante eficaz na resolução de problemas NLP com restrições genéricas e bastante competitivo quando comparado com outros métodos existentes na literatura.

6.2 Extensão do FA para problemas de MINLP

6.2.1 Problemas de MINLP

De seguida apresentam-se os conjuntos de problemas de MINLP usados nas experiências computacionais realizadas relativos aos métodos propostos apresentados no Capítulo 5.

A Tabela 6.12 apresenta o conjunto de problemas teste utilizados nas experiências computacionais relativas ao estudo apresentado na Secção 5.1, na resolução de problemas de MINLP com restrições de limites simples e variáveis binárias. Embora estes problemas

Tabela 6.11: Resultados obtidos pela FPAA e os apresentados em (Costa, Rocha, & Fernandes, 2014) e (Di Pillo et al., 2012)

		FPAA				HAL-iAFS			EPGO	
P	f^*	dist.	f_{best}	f_{med}	n.f.e.(b)	f_{best}	f_{med}	n.f.e.(b)	sol.	n.f.e.
1	0.0293	Lévy	0.0690	12.0945	59405	0.0342	0.1204	9608	0.0625	39575
2a	-400.000	Unif.	-400.000	-380.7241	59420	-380.674	-369.111	15813	-134.1127	115107
2b	-600.000	Unif.	-400.000	-366.330	59411	-385.051	-360.786	15808	-768.4569	120057
2c	-750.000	Unif.	-749.999	-749.770	52238	-743.416	-693.743	15612	-82.9774	102015
2d	-400.000	Lévy	-400.000	-399.980	26005	-399.910	-399.492	15394	-385.1704	229773
3a	-0.3888	Lévy	-0.3882	-0.3837	62306	-0.3880	-0.3849	18928	-0.3861	48647
3b	-0.3888	Unif.	-0.3888	-0.3881	2741	-0.3888	-0.3888	2589	-0.3888	3449
4	-6.6666	Lévy	-6.6667	-6.6667	20825	-6.6667	-6.6667	2242	-6.6666	3547
5	201.1600	Lévy	201.1593	201.1593	20824	201.159	201.159	2926	201.1593	14087
6	376.2919	Unif.	376.2921	376.2939	20874	376.292	376.293	5617	0.4701	1523
7	-2.8284	Unif.	-2.8284	-2.8283	20836	-2.8284	-2.8284	3434	-2.8058	13187
8	-118.7000	Lévy	-118.7049	-118.7048	20791	-118.705	-118.705	2884	-118.7044	7621
9	-13.4020	Lévy	-13.4019	-13.4019	31068	-13.4018	-13.4017	5732	-13.4026	68177
10	0.74178	Lévy	0.74179	0.74181	19551	0.7418	0.7418	6342	0.7420	6739
11	-0.5000	Lévy	-0.5000	-0.5000	6141	-0.5000	-0.5000	3313	-0.5000	3579
12	-16.7390	Unif.	-16.7393	-16.6103	20765	-16.7389	-16.7389	98	-16.7389	3499
13	189.350	Lévy	189.347	226.017	23514	189.345	189.347	9230	195.9553	8085
14	-4.5142	Lévy	-4.5142	-4.5142	27267	-4.5142	-4.5142	6344	-4.3460	19685
15	0.0000	Lévy	0.0000	0.0000	5696	0.0000	0.0000	2546	0.0000	1645
16	0.70492	Lévy	0.7049	0.7049	1017	0.7049	0.7049	1850	0.7181	22593

sejam bastantes utilizados no contexto da otimização contínua, pretende-se que convirjam para uma solução ótima binária. Os problemas teste apresentados na Tabela 6.13 foram usados nas experiências computacionais com o algoritmo de penalidade, Algoritmo 12, na Secção 5.2. Estes problemas são também geralmente utilizados em otimização contínua e, para este efeito, foram adequadamente modificados de modo a que pelo menos uma das variáveis assuma pelo menos um valor inteiro (M. M. Ali, Khompatraporn, & Zabinsky, 2005; Liuzzi, Lucidi, & Rinaldi, 2012). Em ambas as tabelas, na primeira coluna apresenta-se o

Tabela 6.12: Problemas de BCOP usados com as HBFA

Problema	n	x^*	f^*
Ackley	n	(0,...,0)	0
Foxholes	2	(0,0)	≈ 13
Griewank	n	(0,...,0)	0
Quartic	n	(0,...,0)	0
Rastrigin	n	(0,...,0)	0
Rosenbrock2	2	(1,1)	0
Rosenbrock	n	(1,...,1)	0
Schaffer	2	(0,0)	0
Spherical	3	(0,0,0)	0
Step	5	(0,0,0,0,0)	30

Tabela 6.13: Problemas de BCOP usados com o algoritmo de penalidade

Problema	n	x^*	f^*
Ackley(ACK_n)	n	(0,...,0)	0
Aluffi-Pentini (AP)	2	(-1.0465,0)	≈ -0.3523
Beale (Bea)	2	(3,0.5)	0
Becker-Lago(BL)	2	$(\mp 5, \mp 5)$	0
Bohachevsky 1 (BF1)	2	(0,0)	0
Bukin (Buk)	2	(-10,1)	0
Dekkers-Aarts (DA)	2	$(0, \mp 15)$	-24777
Dixon-Price (DP_n)	n	(0,...,0)	0
Himmelblau (Him)	2	(3,2)	0
Levy-Montalvo 2 (LM2_n)	n	(1,1,...,1)	0
Neumaier 2 (NF2)	4	(1,2,2,3)	0
Rastrigin (RG_n)	n	(0,...,0)	0
Shekel 10 (S10)	4	(4,4,4,4)	≈ -10.5319
Sum Squares (SS_n)	n	$(0, \ldots, 0)$	0

Tabela 6.14: Problemas de MINLP usados com o FAbRAO

Problema	n_c/n_d	x^*	f^*	m_{eq}	m
ex 12.2.1*	2/3	(1.118035,1.310370,0,1,1)	7.66718	2	5
ex 12.2.2*	2/1	(0.94194,-2.1,1)	1.07654	0	3
ex 12.2.3*	3/4	(0.19999892, 0.7999976, 1.907877, 1, 1, 0, 1)	4.57958	0	9
ex 12.2.6*	1/1	(4,1)	-17.0000	0	3
st_e13**	1/1	(0.5,1)	2.0000	0	2
f2***	2/1	(1.375,0.375,1)	2.124	1	2
f10***	1/1	n.d.	-2.444	0	2
f11***	1/2	n.d	3.2361	0	3

*Problema teste incluídos em (Floudas et al., 2013)

**Problema teste incluído em (Grossmann & Kravanja, 1995)

***Problema teste incluído em (Hedar & Fahim, 2011); n.d. significa não disponível

nome do problema e o respetivo acrónimo, na segunda coluna a respetiva dimensão, e nas terceiras e quartas colunas são apresentados os minimizantes e os mínimos globais de cada um dos problemas. A Tabela 6.14 apresenta os problemas de MINLP usados nas experiências computacionais realizadas no âmbito do FAbRAO proposto na Secção 5.3, onde, além dos itens referidos anteriormente, também são apresentados o número de variáveis contínuas e inteiras de cada problema, n_c/n_d, o número de restrições de igualdade, m_{eq}, e o número total de restrições, m.

6.2.2 Resultados computacionais das HBFA

Nesta secção apresentam-se os resultados computacionais obtidos com as HBFA (Algoritmos 9, 10 e 11), usando as regras heurísticas definidas pelas funções sigmódes (5.1.4), (5.1.6) e a função (3.2.1), com o objetivo de averiguar os seus desempenhos na resolução de problemas de MINLP com restrições de limites simples e variáveis binárias. São também usados dois problemas da mochila 0-1, de pequena dimensão, para averiguar o desempenho dos algoritmos em problemas lineares com variáveis binárias. Os algoritmos foram codificados na linguagem de programação Matlab, versão 8.0.0.783 (R2012b). Nas experiências

computacionais realizadas foram usadas as seguintes configurações. Os códigos foram executados 30 vezes para cada um dos problemas teste apresentados na tabela 6.12 e para dois problemas da mochila 0-1. O tamanho da população depende da dimensão do problema, e é dado por $N = \min\{40, 2^n\}$.

Previamente realizaram-se algumas experiências computacionais com o objetivo de ajustar certos parâmetros dos algoritmos. Na HBFA propostas, são usadas atualizações dinâmicas de α e γ, dadas respetivamente por (5.1.1) e (5.1.2), com os seguintes parâmetros: $\beta_0 = 1$, $p = 1$, $\alpha_{\max} = 0.5$, $\alpha_{\min} = 0.01$, $\gamma_{\max} = 10$, $\gamma_{\min} = 0.1$. Na condição de paragem dos algoritmos 9 (mEC), 10 (mEB) e 11 (pCB), foram usados os seguintes valores: $k_{\max} = 500$ (número máximo de iterações permitido) e $\eta = 10^{-6}$ (tolerância exigida para encontrar uma solução de boa qualidade). Para os resultados obtidos nas diversas experiências serão usados os seguintes acrónimos: f_{avg} para a média das soluções obtidas, nfe para o número médio de avaliações da função f, nit para o número médio de iterações efetuadas, e $St.Dev$ para o desvio padrão das soluções obtidas.

Numa primeira experiência, foram utilizados os problemas descritos na Tabela6.12 que têm diferentes características e dimensões. Por exemplo, 5 funções são unimodais e as restantes são multi-modais (M. M. Ali et al., 2005; Engelbrecht & Pampara, 2007; Pampara et al., 2006; Pampara & Engelbrecht, 2011). Nesta experiência considerou-se $n = 30$. O objetivo é comparar os resultados obtidos pelas HBAF- Algoritmos 9, 10 e 11- usando a regra heurística 5.1.5 definida pela função logística sigmóide 5.1.4, com os apresentados em (Engelbrecht & Pampara, 2007; Pampara et al., 2006; Pampara & Engelbrecht, 2011). Devido aos fracos resultados obtidos com o algoritmo ABC, Pampara e Engelbrecht (2011) não aconselham o seu uso para resolver problemas do tipo binário. As outras metaheurísticas lá implementadas foram as seguintes:

- Ângulo modelado PSO (do inglês, *angle modulated PSO:AMPSO*) e ângulo modelado DE (do inglês, *angle modulated DE:AMDE*), que incorporam uma função trigonométrica como um gerador de sequências de 0 e 1 nos algoritmos clássicos PSO e DE, respetivamente;

- PSO e DE binários, baseados na regra heurística 5.1.5 definida pela função logística sigmóide 5.1.4, denotados por binPSO e binDE, respetivamente.

Na Tabela 6.15 são comparados os items f_{avg}, nfe e $St.Dev.$ (entre parêntesis) obtidos em 30 execuções dos algoritmos, com os resultados obtidos pelos métodos AMPSO, binPSO, binDE e AMDE.

Da tabela observa-se que a proposta da HBFA produz soluções de elevada qualidade e supera as versões binárias binPSO e binDE, bem como as versões AMPSO e AMDE. Observa-se também que, em termos de nfe, o mEC é melhor em 6 problemas, o mEB é melhor em 3 problemas (um é um empate com o mEC), e o pCB em 2 problemas, sendo este último menos eficiente em problemas de grande dimensão.

Para analisar a significância estatística dos resultados, foi realizado o teste estatístico de Friedman. Este é um teste estatístico não paramétrico usado para testar se existem diferenças significativas em média para uma variável independente com dois ou mais níveis, também designados por tratamentos, e uma variável dependente (ou grupos correspondentes, neste caso dado pelos problemas). A hipótese nula neste teste é que "a média dos *ranks* atribuídos aos tratamentos são iguais". Uma vez que as 3 implementações foram capazes de obter a solução ótima dentro da tolerância de erro η em 9 dos 10 problemas, a análise estatística foi baseada no item nfe. Neste teste de hipótese, existem 3 tratamentos e 10 grupos. A estatística de teste de χ^2 de Friedman tem um valor de 2.737 (com um p-valor de 0.255). Para a distribuição χ^2, com dois graus de liberdade, o valor crítico para um nível de significância de 5% é de 5.99. Portanto, uma vez que $2.737 \leq 5.99$, a hipótese nula não é rejeitada e conclui-se que não existe evidência estatística de que os valores das médias dos 3 *ranks* têm diferenças significativas.

Tabela 6.15: Comparação com AMPSO, binPSO, binDE, AMDE

	mEC baseada em (5.1.4)		mEB baseada em (5.1.4)		pCB baseada em (5.1.4)		AMPSO	binPSO	binDE	AMDE
Prob.	f_{avg}	nfe	f_{avg}	nfe	f_{avg}	nfe	f_{avg}	f_{avg}	f_{avg}	f_{avg}
Ackley	8.88e-16	80	8.88e-16	1156	8.88e-16	2168	1.97e01	2.01e01	1.73e01	1.64e01
	(0.00e00)		(0.00e00)		(0.00e00)		(0.57e-01)	(0.49e-01)	(0.31e01)	(0.76e00)
Foxholes	1.27e01	6.1	1.27e01	7.2	1.27e01	6.3	0.53e-10	0.53e-10	1.29e01	5.0e02
	(9.03e-15)		(9.03e-15)		(9.03e-15)		(0.00e00)	(0.97e-14)	(0.86e00)	(0.0e00)
Griewank	0.00e00	80	0.00e00	1332	0.00e00	2300	1.06e02	6.79e01	2.63e02	2.06e02
	(0.00e00)		(0.00e00)		(0.00e00)		(0.44e01)	(0.98e01)	(0.10e02)	(0.39e01)
Quartic	4.55e-01	2012	5.37e-01	1771	4.88e-01	2951	4.15e01	2.09e01	1.49e00	3.55e00
	(3.01e-01)		(3.03e-01)		(2.71e-01)		(0.19e01)	(0.19e01)	(0.66e00)	(0.81e00)
Rastrigin	0.00e00	80	0.00e00	1282.7	0.00e00	2406.7	2.25e02	3.08e02	2.14e02	9.05e01
	(0.00e00)		(0.00e00)		(0.00e00)		(0.35e02)	(0.28e01)	(0.45e02)	(0.31e02)
Rosenbrock2	0.00e00	6.5	0.00e00	5.7	0.00e00	6.1	0.49e-04	0.14e-03	0.20e-05	0.55e-04
	(0.00e00)		(0.00e00)		(0.00e00)		(0.11e-03)	(0.88e-04)	(0.15e-04)	(0.17e-04)
Rosenbrock	0.00e00	180	0.00e00	2190.7	0.00e00	2088	2.20e03	2.24e03	1.81e03	9.14e01
	(0.00e00)		(0.00e00)		(0.00e00)		(0.86e02)	(0.77e02)	(0.16e02)	(0.42e02)
Schaffer	0.00e00	6.1	0.00e00	8	0.00e00	4.9	0.24e-01	0.73e-01	-0.995e00	-1.0e00
	(0.00e00)		(0.00e00)		(0.00e00)		(0.42e-02)	(0.11e-01)	(0.27e-06)	(0.0e00)
Spherical	0.00e00	12	0.00e00	12	0.00e00	11.7	0.30e-03	0.30e-03	0.15e-03	0.20e-04
	(0.00e00)		(0.00e00)		(0.00e00)		(0.00e00)	(0.0e00)	(0.41e-04)	(0.15e-04)
Step	3.00e01	42.7	3.00e01	42.7	3.00e01	48	0.00e00	0.17e-04	0.15e-01	0.25e00
	(0.00e00)		(0.00e00)		(0.00e00)		(0.00e00)	(0.15e-04)	(0.0e00)	(0.60e-01)

Para comparar as heurísticas definidas pelas funções sigmóides com a heurística definida pela função chão, apresenta-se na Tabela 6.16 os resultados obtidos com a função `erf` sigmóide (5.1.6) aplicada em (5.1.5), e a regra heurística definida à custa da função chão (3.2.1). Para tal, apenas foram testadas as implementações mEC e mEB. Desta tabela conclui-se que a implementação mEC (Algoritmo 9) produz bons resultados com ambas as regras heurísticas (5.1.6) definidas pelas funções (5.1.5) e (3.2.1). A taxa de sucesso é de 100%, o que significa que os algoritmos terminam porque o valor de f na posição do pirilampo mais brilhante está dentro da tolerância η da solução ótima f^* em todas as execuções. Conclui-se ainda que o mEB (Algoritmo 10) produz melhores resultados quando a discretização das variáveis é feita usando a função chão (3.2.1).

Tabela 6.16: Comparação mEC *vs.* mEB e (5.1.6) *vs.* (3.2.1)

	mEC baseado em (5.1.6)			mEB baseado em (5.1.6)			mEC baseado em (3.2.1)			mEB baseado em (3.2.1)		
Prob.	f_{avg}	nfe	nit	f_{avg}	nfe	nit	f_{avg}	nfe	nit	f_{avg}	nfe	nit
Ackley	8.88e-16	80	1	8.88e-16	678.7	16	8.88e-16	80	1	8.88e-16	82.7	1.1
Foxholes	1.27e01	5.7	0.4	1.27e01	8	1	1.27e01	6.1	0.5	1.29e01	404.8	100.2
Griewank	0.00e00	80	1	0.00e00	717.3	16.9	0.00e00	80	1	0.00e00	82.7	1.1
Quartic	2.38e-01	81.3	1.03	4.66e-01	794.7	18.9	9.48e-02	80	1	3.93e-01	80	1
Rastrigin	0.00e00	80	1	0.00e00	702.7	16.6	0.00e00	80	1	0.00e00	80	1
Rosenbrock2	0.00e00	6.4	0.6	0.00e00	5.3	0.3	0.00e00	6.3	0.6	2.33e-01	470.8	116.7
Rosenbrock	0.00e00	80	1	0.00e00	105.3	1.6	0.00e00	80	1	2.90e01	20040	500
Schaffer	0.00e00	6.8	0.7	0.00e00	12.9	2.2	0.00e00	5.1	0.3	7.08e-02	205.1	50.3
Spherical	0.00e00	9.9	0.2	0.00e00	22.7	1.8	0.00e00	10.1	0.3	0.00e00	11.5	0.4
Step	3.00e01	45.9	0.4	3.00e01	62.9	1	3.00e01	40.5	0.3	3.00e01	40.5	0.3

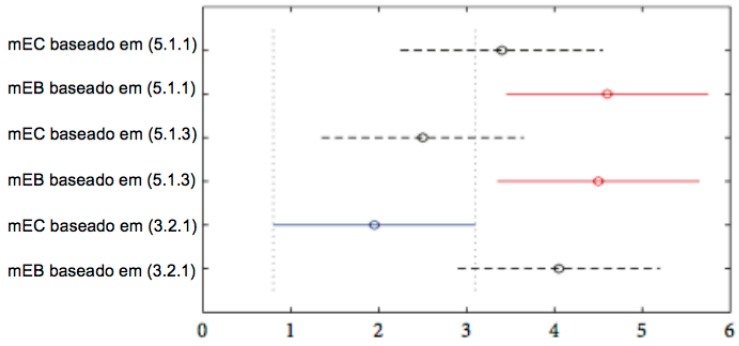
No geral, o mEC baseado em (5.1.6) produz melhores resultados em 6 problemas, o mEC baseado em (3.2.1) produz melhores resultados em 7 problemas (incluindo 4 empates com o caso anterior), o mEB baseado em (5.1.6) é melhor apenas num problema, e o mEB baseado em (3.2.1) é melhor em 3 problemas (todos são empates com o mEC baseado em (3.2.1)). Mais ainda, realizando um teste de Friedman para as 4 distribuições dos valores

de nfe, o valor da estatística de teste é de 13.747 (e o p-valor é 0.0033). Da tabela da distribuição χ^2, com 3 graus de liberdade, o valor crítico para um nível de significância de 5%, é de 7.81. Uma vez que $13.747 > 7.81$, a hipótese nula é rejeitada, concluindo-se assim que existem diferenças estatisticamente significativas entre as quatro distribuições dos valores de nfe.

A seguir, é realizada a análise estatística dos resultados apresentados nas Tabelas 6.15 e 6.16, relativamente às implementações mEC e mEB. Assim, esta análise envolve a comparação de 6 distribuições de valores de nfe. O valor da estatística de teste de χ^2 de Friedman é de 18.175 (com p-valor de 0.0027). O valor crítico da distribuição de χ^2, para um nível de significância de 5% e 5 graus de liberdade é de 11.07. Assim, a hipótese nula de que "não existem diferenças significativas nas médias dos *ranks* de nfe" é rejeitada. Portanto, existe evidência estatística que as 6 distribuições têm diferenças significativas.

Realizando-se múltiplas comparações entre pares, pode-se determinar quais as distribuições dos valores de nfe são significativamente diferentes. As estimativas dos intervalos de confiança de 95% são apresentadas na 6.1.

Figura 6.1: Intervalos de confiança para a média dos *ranks* de nfe



Quando comparadas, duas distribuições de nfe são significativamente diferentes se os seus intervalos forem disjuntos, e não são significativamente diferentes se os seus intervalos se intersetam. Assim, para os 6 casos considerados, conclui-se que a distribuição de nfe dada pelo mEC baseada em (3.2.1) é significativamente diferente das distribuições de nfe dada pelo mEB baseado em (5.1.4) e mEB baseada em (5.1.6), respetivamente. Para os

restantes pares em comparação não existem diferenças significativas nas distribuições das médias dos *ranks* de nfe. Para fins comparativos, são apresentados na Tabela 6.17 os resultados obtidos pela implementação do mEC usando a distribuição de Lévy (L) e a distribuição Uniforme (U), no termo aleatório nas equações de movimento (5.1.3) e (2.1.3), respetivamente. Nestas experiências considerou-se a implementação do mEC (descrito no Algoritmo 9) com a regra heurística 5.1.7, definida pela função '`erf`' sigmóide 5.1.6, e a regra heurística 5.1.7 definida pela função chão 3.2.1. Pela análise da tabela observa-se que o desempenho da HBFA-mEC, com a distribuição Uniforme, é muito sensível à dimensão do problema, uma vez que a eficiência diminui à medida que n aumenta. Portanto, daqui conclui-se que o uso da distribuição de Lévy é uma boa escolha.

Tabela 6.17: Comparação entre a distribuição de Lévy e a distribuição Uniforme

	mEC baseado em (5.1.6) + **L**			mEC baseado em (5.1.6) + **U**			mEC baseado em (3.2.1) + **L**			mEC baseado em (3.2.1) + **U**		
Prob.	f_{avg}	nfe	nit	f_{avg}	nfe	nit	f_{avg}	nfe	nit	f_{avg}	nfe	nit
Ackley	8.88e-16	80	1	8.88e-16	1096	26.4	8.88e-16	80	1	1.41e00	20040	500
	(0.00e00)			(0.00e00)			(0.00e00)			(1.26e-02)		
Foxholes	1.27e01	5.7	0.4	1.27e01	8	1	1.27e01	6.1	0.5	1.27e01	6.8	0.7
	(9.03e-15)			(9.03e-15)			(9.03e-15)			(9.03e-15)		
Griewank	0.00e00	80	1	0.00e00	2436	59.9	0.00e00	80	1	1.27e-01	20040	500
	(0.00e00)			(0.00e00)			(0.00e00)			(2.76e-02)		
Quartic	2.38e-01	81.3	1.03	6.90e00	14692	366.2	9.48e-02	80	1	5.10e-01	10581	263.5
	(1.70e-01)			(1.16e01)			(1.03e-01)			(2.90e-01)		
Rastrigin	0.00e00	80	1	0.00e00	1157	27.9	0.00e00	80	1	1.00e-01	18104	451.6
	(0.00e00)			(0.00e00)			(0.00e00)			(4.03e-01)		
Rosenbrock2	0.00e00	6.4	0.6	0.00e00	36.8	8.2	0.00e00	6.3	0.6	0.00e00	5.9	0.5
	(0.00e00)			(0.00e00)			(0.00e00)			(0.00e00)		
Rosenbrock	0.00e00	80	1	8.22e01	14136	352.4	0.00e00	80	1	7.42e01	17698	441.2
	(0.00e00)			(1.02e02)			(0.00e00)			(8.43e01)		
Schaffer	0.00e00	6.8	0.7	0.00e00	18.5	3.6	0.00e00	5.1	0.3	0.00e00	5.6	0.4
	(0.00e00)			(0.00e00)			(0.00e00)			(0.00e00)		
Spherical	0.00e00	9.9	0.2	0.00e00	13.3	0.7	0.00e00	10.1	0.3	0.00e00	14.7	0.8
	(0.00e00)			(0.00e00)			(0.00e00)			(0.00e00)		
Step	3.00e01	45.9	0.4	3.00e01	46.9	0.5	3.00e01	40.5	0.3	3.00e01	52.3	0.6
	(0.00e00)			(0.00e00)			(0.00e00)			(0.00e00)		

De seguida, pretende-se comparar os resultados obtidos pela HBFA-mEC com os reportados por L. Wang et al. (2013), onde é usado o método ABHS (do inglês, *Adaptive Binary Harmony Search*). Para tal, foram considerados apenas problemas de dimensão $n = 30$. Assim, além dos apresentados na Tabela 6.12- Ackley, Griewank, Rastrigin, Rosenbrock,

Spherical- foram acrescentadas três outras funções: Schwefel 2.22, Schwefel 2.26 e Sum of Different Power. A função Schwefel 2.22 é unimodal e, para $X = [-10, 10]^{30}$, a solução binária é $(0, \ldots, 0)$ com $f^* = 0$; a Schwefel 2.26 é multimodal e, para $X = [-500, 500]^{30}$, a solução binária é $(1, 1, \ldots, 1)$ com $f^* = -25.244129544$; a Sum of Different Power é unimodal e, para $X = [-1, 1]^{30}$, o mínimo é 0 em $(0, \ldots, 0)$.

Na Tabela 6.18 são apresentados os resultados obtidos pela HBFA-mEC usando a regra heurística 5.1.5 definida pela função '`erf`' sigmóide 5.1.6, e a regra heurística 5.1.5 definida pela função chão 3.2.1. A tabela apresenta os itens f_{avg}, nfe, e a taxa de sucesso (TS). Para comparar com os resultados apresentados por L. Wang et al. (2013), a HBFA-mEC foi executada 50 vezes por problema, e o número máximo de avaliações de f permitido foi de 90000.

Tabela 6.18: Comparação entre HBFA-mEC e ABHS em (L. Wang et al., 2013)

	mEC baseado em (5.1.6)			mEC baseado em (3.2.1)			ABHS em (L. Wang et al., 2013)		
Prob.	f_{avg}	nfe	TS (%)	f_{avg}	nfe	TS (%)	f_{avg}	nfe	TS (%)
Ackley	8.88e-16	80	100	8.88e-16	80	100	1.56e-01	62350	90
Griewank	0.00e00	80	100	0.00e00	80	100	3.30e-02	79758	38
Rastrigin	0.00e00	80	100	0.00e00	80	100	1.32e01	90000	0
Rosenbrock	0.00e00	80	100	0.00e00	80	100	6.80e02	90000	0
Schwefel 2.22	0.00e00	80	100	0.00e00	80	100	0.00e00	59870	100
Schwefel 2.26	-2.52e01	80	100	-2.47e01	10867	87	-1.195e04	90000	0
Spherical	0.00e00	80	100	0.00e00	80	100	0.00e00	62234	100
Sum Different Power	0.00e00	91	100	0.00e00	168	100	0.00e00	80371	100

Da Tabela 6.18 conclui-se que a HBFA-mEC tem melhor desempenho do que o método ABHS apresentado por L. Wang et al. (2013).

Para analisar o efeito da dimensão do problema no desempenho das HBFA, são usados seis problemas da Tabela 6.12 com diferentes dimensões. As três dimensões testadas são: $n = 50, n = 100$ e $n = 200$. Os parâmetros dos algoritmos são os definido anteriormente. Nestas experiências, o tamanho da população para todos os problemas teste e dimensões é de $N = 40$. A Tabela 6.19 apresenta os resultados para comparação baseados nos itens f_{avg}, nfe, nit e $St.Dev.$. Uma vez que a implementação do mEC tem um melhor desempenho

e mostra resultados mais consistentes do que o mEB e pCB, testou-se apenas o mEC com a regra heurística 5.1.5 definida na função '`erf`' sigmóide 5.1.6, e na regra heurística 5.1.5 definida pela função chão 3.2.1.

Tabela 6.19: Resultados obtidos para diferentes dimensões do problema

.

			mEC baseado em (5.1.6)				mEC baseado em (3.2.1)			
	f^*	n	f_{avg}	$St.Dev.$	nfe	nit	f_{avg}	$St.Dev.$	nfe	nit
Ackley	0.00e+00	50	8.88e-16	0.00e+00	80	1	8.88e-16	0.00e+00	80	1
	0.00e+00	100	8.88e-16	0.00e+00	80	1	8.88e-16	0.00e+00	80	1
	0.00e+00	200	8.88e-16	0.00e+00	80	1	8.88e-16	0.00e+00	80	1
Griewank	0.00e+00	50	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
	0.00e+00	100	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
	0.00e+00	200	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
Quartic	0.00e+00+noise	50	2.24e-01	1.95e-01	82.7	1.1	1.43e-01	1.59e-01	80	1
	0.00e+00+noise	100	4.32e-01	2.73e-01	146.7	2.7	1.73e-01	1.10e-01	80	1
	0.00e+00+noise	200	5.23e-01	2.94e-01	1738.7	42.5	1.96e-01	2.13e-01	81.3	1.03
Rosenbrock	0.00e+00	50	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
	0.00e+00	100	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
	0.00e+00	200	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
Spherical	0.00e+00	50	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
	0.00e+00	100	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
	0.00e+00	200	0.00e+00	0.00e+00	80	1	0.00e+00	0.00e+00	80	1
Step	3.00e+02	50	3.00e+02	0.00e+00	80	1	3.00e+02	0.00e+00	80	1
	6.00e+02	100	6.00e+02	0.00e+00	80	1	6.00e+02	0.00e+00	80	1
	1.20e+03	200	1.20e+03	0.00e+00	80	1	1.20e+03	0.00e+00	80	1

Além de testar se existem diferenças significativas nos *ranks* das distribuições de nfe obtidas pelos dois tratamentos, mEC baseada na função '`erf`' sigmóide 5.1.6, e mEC baseada na regra heurística definida pela função chão 3.2.1, pretende-se também testar se a diferença nos *ranks* das distribuições de nfe obtidas pela dimensão do problema -50, 100 e 200- são estatísticamente significativas a um nível de significância de 5%. Portanto, pretende-se analisar o efeito de dois fatores 'A' e 'B', onde 'A' é a implementação das HBFA (com dois níveis), e 'B' é a dimensão do problema (com três níveis). Assim, os resultados obtidos para os seis problemas para cada combinação dos níveis de 'A' e 'B' são considerados como réplicas. Ao realizar o teste χ^2 de Friedman para o fator 'A', o

valor da estatística de teste é de 1.225 (com p-valor 0.2685), com 1 grau de liberdade. O valor crítico para um nível de significância de 5%, com 1 grau de liberdade, na tabela da distribuição χ^2 é de 3.84, concluindo-se assim que não existe evidência estatística de diferenças significativas nos *ranks* das distribuições de nfe obtidas pelos tratamentos mEC baseada na função '`erf`' sigmóide 5.1.6, e mEC baseada na regra heurística definida pela função chão 3.2.1. A partir do teste χ^2 de Friedman para o fator 'B', também conclui-se que não existe evidência estatística de diferenças significativas, pois o valor estatístico é de 0.746 (com p-valor de 0.6886) com dois graus de liberdade, e o valor crítico da distribuição χ^2, para um nível de significância de 5% e 2 graus de liberdade, é de 5.99.

Finalmente, pretende-se analisar o comportamento da melhor das HBFA na resolução de problemas de otimização lineares e binários bem conhecidos da literatura. Para estas experiências, selecionaram-se dois problemas da mochila de pequena dimensão. Este problema surge diretamente, ou então como um subproblema em diversas aplicações: planeamento de produção, modelação financeira, amostragem estratificada, planeamento em capacidade na fabricação, cuidados de saúde e redes de computadores (Bretthauer & Shetty, 2002). O problema da mochila é um problema de otimização do tipo combinatório, cujo nome se deve à modelação da seguinte situação: encher uma mochila com objetos de diferentes pesos e valores. O objetivo é encher a mochila com o maior valor possível, não ultrapassando a sua capacidade máxima. Este problema possui muitas variantes, sendo a mais comum a do problema da mochila 0-1. O problema da mochila 0-1 pode ser descrito da seguinte forma. Seja n o número de items a partir dos quais se tem que selecionar alguns para serem transportados numa mochila. Sejam w_l e v_l o peso e o valor do item l, respetivamente, e W a capacidade da mochila. Em geral, é assumido que todos os pesos e valores são não negativos. O objetivo é maximizar o valor total da mochila sujeita à restrição da sua capacidade:

$$\begin{aligned} \max_x \quad & V(x) \equiv \sum_{l=1}^{n} v_l x_l \\ \text{s.a.} \quad & \sum_{l=1}^{n} w_l x_l \leq W,\ x_l \in \{0,1\},\ \ l = 1, \ldots, n. \end{aligned}$$

Se o item l for selecionado, $x_l = 1$; caso contrário, $x_l = 0$. Usando uma função de

penalidade, o problema pode ser transformado em:

$$\min_x -\sum_{l=1}^{n} v_l x_l + \mu \max\{0, \sum_{l=1}^{n} w_l x_l - W\}$$

onde μ é o parâmetro de penalidade, cujo valor foi de 100 nesta experiência.

Caso 1: Exemplo de um problema da mochila 0-1 com 4 itens. A capacidade da mochila é $W = 6$ e os vetores dos valores e dos pesos são $v = (40, 15, 20, 10)$ e $w = (4, 2, 3, 1)$. Baseados nos parâmetros referidos anteriormente, a HBFA com mEC baseada na função `erf` sigmóide (5.1.6) foi executada 30 vezes. Com uma taxa de sucesso de 100%, os itens 1 e 2 são incluídos na mochila, e os itens 3 e 4 são excluídos, com um valor máximo de 55 (com um desvio padrão igual a 0). Em média, as execuções do algoritmo necessitaram de 0.8 iterações e 29.3 avaliações da função. Com uma taxa de sucesso de 23%, a HBFA com mEC, baseada na regra heurística definida pela função chão (3.2.1), teve $f_{\text{avg}} = 49$ (com um desvio padrão igual a 4), após 6161.1 avaliações da função e 384.1 iterações, em termos médios.

Caso 2: Exemplo de um problema da mochila 0-1 com 8 itens. A capacidade máxima da mochila é 8 e os vetores dos valores e pesos são $v = (83, 14, 54, 79, 72, 52, 48, 62)$ e $w = (3, 2, 3, 2, 1, 2, 2, 3)$. Os resultados que se seguem são os valores médios obtidos ao fim de 30 execuções do algoritmo. Após 8.7 iterações e 386.7 avaliações da função, o valor máximo obtido pela HBFA com mEC baseada na função `erf` sigmóide (5.1.6) foi de 286 (com um desvio padrão igual a 0), com uma taxa de sucesso de 100%. Os itens 1, 4, 5, 6 são incluídos na mochila, e os restantes são excluídos. A HBFA com mEC, baseado na regra heurística definida pela função chão (3.2.1), não alcançou a solução ótima. Todas as execuções necessitaram de 500 iterações e 20040 avaliações da função, e a média dos valores da função foi de $f_{\text{avg}} = 227$, com um desvio padrão igual 31.4.

6.2.3 Resultados computacionais do algoritmo de penalidade exata

Foram realizados vários conjuntos de experiências com o objetivo de comparar a velocidade de convergência e a qualidade das soluções produzidas pelo Algoritmo 12, quando

testado com os termos de penalidade $P^t(\cdot)$ definido em (5.2.2) e $P^a(\cdot)$ definido em (5.2.6). A velocidade de convergência é medida em termos do número de avaliações da função objetivo necessárias para atingir o valor ótimo da função objetivo disponível na literatura, com uma pequena tolerância positiva. A qualidade da solução é medida em termos da diferença entre o resultado obtido e o valor ótimo conhecido, f^*. Os algoritmos foram codificados em Matlab, versão 8.1 (R2013a). Para testar os algoritmos foram usadas 22 instâncias dos 14 problemas de MINLP apresentados na Tabela 6.13, em que o problema Dixon-Price foi testado com $n = 2$ e $n = 4$, o problema Sum Squares foi testado com $n = 5$ e $n = 10$, e os problemas Ackley, Levy e Rastrigin foram testados com $n = 5$, $n = 10$ e $n = 20$.

Os parâmetros usados no Algoritmo 12 foram os seguintes: $\varepsilon^{(1)} = 10$, $\delta^{(1)} = 1$, $\delta_{\min} = 10^{-4}$, $\sigma_\varepsilon = 0.1$, $\sigma_\delta = 0.1$, $p = 1$ e $k_{\max} = 20$. No Algoritmo 13, o número máximo de iterações permitido foi $k_{\max}^{FA} = 100$, e para as atualizações dinâmicas de α e γ, os parâmetros usados foram: $\alpha_{\max} = 0.5$, $\alpha_{\min} = 0.001$, $\gamma_{\max} = 10$ e $\gamma_{\min} = 0.001$. O valor de β_0 foi 1. Na maioria das experiências, considerou-se o tamanho da população no FA clássico e no FA adaptativo dependente da dimensão do problema e dado por $N = \min\{5n, 50\}$.

Primeiro, é apresentada a comparação entre o FA clássico e o FA adaptativo baseada na qualidade da solução. O Algoritmo 12 termina quando o número de iterações $k_{\max}$ é atingido. A Tabela 6.20 mostra a diferença entre o melhor valor da função objetivo obtido em 10 execuções do Algoritmo 12 e a solução ótima global conhecida, $|f_{\text{best}} - f^*|$. Ambos os termos de penalidade (5.2.2) e (5.2.6) foram testados. Os resultados correspondem às instâncias dos 14 problemas com pequenas dimensões apresentados na Tabela 6.13. Pela análise dos resultados, conclui-se que o FA adaptativo tem um desempenho ligeiramente melhor do que o FA clássico, e o Algoritmo 12 com o termo de penalidade (5.2.2) produz soluções de qualidade superior em 8 casos enquanto que com o termo de penalidade (5.2.6) produz soluções de qualidade superior em 6 (entre os 30 resultados de ambas as variantes, onde 16 são empates).

Na experiência seguinte, pretende-se avaliar o efeito do tamanho da população de pirilampos nos resultados obtidos pelo Algoritmo 12 com o FA adaptativo. Foram usadas quatro instâncias de problemas com $n = 5$, e testou-se três valores de N: $N = 2n, N =$

Tabela 6.20: Comparação entre o FA clássico e o FA adaptativo.

		usando o termo de penalidade (5.2.2)		usando o termo de penalidade (5.2.6)	
Prob.	f^*	'FA clássico'	'FA adaptado'	FA clássico	FA adaptativo
ACK_5	0	8.882e-16	8.882e-16	8.882e-16	8.882e-16
AP	-0.3523	8.603e-05	8.603e-05	8.604e-05	8.607e-05
Bea	0	1.573e-18	2.444e-21	1.034e-18	6.296e-20
BL	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
BF1	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
Buk	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
DA	-24777	4.831e-01	4.817e-01	4.817e-01	4.817e-01
DP_2	0	1.583e-20	1.680e-19	6.073e-19	3.931e-18
DP_4	0	5.064e-17	3.242e-17	2.371e-18	5.089e-19
Him	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
LM2_5	0	1.146e-31	1.500e-32	1.701e-27	2.122e-27
NF2	0	2.038e-04	0.000e+00	0.000e+00	0.000e+00
RG_5	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00
S10	-10.5319	4.384e-03	4.384e-03	4.384e-03	4.384e-03
SS_5	0	2.581e-17	8.969e-31	0.000e+00	2.782e-26

$5n$ e $N = 10n$. A Tabela 6.21 apresenta os valores de $|f_{\text{best}} - f^*|$ obtidos usando os termos de penalidades (5.2.2) e (5.2.6). Baseados nestes resultados, pode-se concluir que a qualidade das soluções é superior quando foram usadas populações com tamanhos maiores. Porém, a diferença entre o uso de $N = 5n$ e $N = 10n$ é quase inexistente.

Tabela 6.21: Efeito do tamanho da população usando o FA adaptativo.

		$\|f_{\text{best}} - f^*\|$ com a penalidade (5.2.2)			$\|f_{\text{best}} - f^*\|$ com a penalidade (5.2.6)		
Prob.	f^*	$N = 10$	$N = 25$	$N = 50$	$N = 10$	$N = 25$	$N = 50$
ACK_5	0	2.850e-09	8.882e-16	8.882e-16	1.712e-10	8.882e-16	4.441e-15
LM2_5	0	2.935e-17	1.500e-32	1.500e-32	1.796e-17	2.122e-27	2.000e-27
RG_5	0	2.842e-13	0.000e+00	0.000e+00	4.320e-12	0.000e+00	0.000e+00
SS_5	0	7.436e-19	8.969e-31	1.409e-17	9.307e-17	2.782e-26	4.084e-27

Finalmente, usam-se todas as instâncias dos problemas mencionados acima e analisa-se o desempenho do algoritmo de penalidade em termos de qualidade das soluções obtidas. A Tabela 6.22 mostra a diferença $|f_{\text{best}} - f^*|$ obtida após 10 execuções do algoritmo por cada instância, e com cada um dos termos de penalidade testados: (5.2.2), (5.2.6) e (3.2.18). Nes-

tas experiências, o algoritmo de penalidade termina somente quando a condição $k_{\max} = 20$ é atingida. Considerou-se que o tamanho da população é dado por $N = \min\{5n, 50\}$. Pretende-se também comparar o desempenho do Algoritmo 12 para diferentes valores iniciais de $\varepsilon^{(1)}$: 10 e 100. Dado que agora são incluídas instâncias de grande dimensão, considerou-se no FA adaptativo o número máximo de iterações $k_{\max}^{FA} = 50$. Com exceção de quatro instâncias, BL, BF1, Buk e Him, que desempenham de forma uniforme e extremamente bem com os três termos de penalidade testados, observa-se que a qualidade das soluções obtidas é superior com a penalidade (5.2.2) em 42% dos casos (considerando o total de 26 conjuntos de execuções com diferentes valores de $|f_{\text{best}} - f^*|$, dos 44 conjuntos de execuções) do que com as penalidades (5.2.6) (em 35% dos casos), e (3.2.18) (em apenas 23% dos casos).

Observa-se também que a qualidade das soluções é superior quando é usado $\varepsilon^{(1)} = 10$ com a penalidade (5.2.6) (71% contra 29%, em 14 instâncias com valores diferentes de $|f_{\text{best}} - f^*|$), e com a penalidade (3.2.18) (59 % contra 41% em 17 instâncias) e há um empate quando é usada a penalidade (5.2.2). Por outro lado, a Tabela 6.23 mostra a média do número de avaliações da função, nf_{avg}, e o respetivo desvio padrão, $St.Dev.$, necessário para obter uma solução admissível dentro de uma tolerância $\delta_{\min}$ da solução ótima conhecida, isto é, o algoritmo termina quando as condições $x^{(k)} \in W$ e $f(x^{(k)}) \leq f^* + \delta_{\min}$ (para $\delta_{\min} = 10^{-4}$) são verificadas. Nestas experiências, para verificar se $x^{(k)} \in W$, onde $z^{(k)}$ está definido em (5.2.10), é usado $\|x^{(k)} - z^{(k)}\|_\infty \leq \sigma_{\min}$, onde a tolerância do erro é $\sigma_{\min} = 10^{-5}$.

O objetivo agora é analisar a velocidade de convergência do Algoritmo 12, usando os três termos de penalidade e $\varepsilon^{(1)} = 100$. Nestas experiências considerou-se também $k_{\max}^{FA} = 50$. É de referir que o número médio de avaliações da função é calculado apenas nas execuções que terminam com as condições $x^{(k)} \in W$ e $f(x^{(k)}) \leq f^* + \delta_{\min}$, sendo estas consideradas execuções bem sucedidas. A tabela apresenta a percentagem de execuções bem sucedidas, denotada de SR (%), em 10 execuções do algoritmo por problema. O caracter "–" significa que apenas uma execução em 10 foi bem sucedida, sendo, nesse caso, nf_{avg} o número

Tabela 6.22: Resultados da qualidade da solução $|f_{\text{best}} - f^*|$, usando $k_{\max} = 20$.

		penalidade (5.2.2)		penalidade (5.2.6)		penalidade (3.2.18)	
Prob.	f^*	$\varepsilon^{(1)} = 10$	$\varepsilon^{(1)} = 100$	$\varepsilon^{(1)} = 10$	$\varepsilon^{(1)} = 100$	$\varepsilon^{(1)} = 10$	$\varepsilon^{(1)} = 100$
ACK_5	0	8.882e-16	8.882e-16	8.882e-16	8.882e-16	7.994e-15	4.441e-15
ACK_10	0	1.155e-14	2.220e-14	3.286e-14	4.707e-14	1.998e-13	2.212e-13
ACK_20	0	2.049e+00	2.027e+00	1.884e+00	1.425e+00	2.639e+00	2.026e+00
AP	-0.3523	8.606e-05	8.603e-05	8.605e-05	8.601e-05	8.603e-05	8.607e-05
Bea	0	4.933e-18	3.896e-20	7.654e-19	1.057e-17	2.776e-19	2.415e-18
BL	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
BF1	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
Buk	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
DA	-24777	4.817e-01	4.817e-01	4.817e-01	4.817e-01	4.817e-01	4.817e-01
DP_2	0	1.392e-19	2.939e-19	1.019e-19	9.919e-18	3.001e-18	7.392e-18
DP_4	0	2.159e-18	1.029e-18	2.778e-21	5.908e-19	1.762e-16	4.226e-17
Him	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
LM2_5	0	1.500e-32	4.134e-31	1.500e-32	3.215e-27	1.184e-28	2.815e-26
LM2_10	0	7.228e-29	3.757e-26	2.412e-25	6.729e-25	4.323e-28	1.141e-26
LM_20	0	1.164e-12	9.289e-14	7.292e-13	9.477e-14	2.051e-12	3.806e-16
NF2	0	1.803e-03	0.000e+00	7.557e-02	0.000e+00	8.109e-02	0.000e+00
RG_5	0	0.000e+00	0.000e+00	0.000e+00	0.000e+00	9.468e-09	0.000e+00
RG_10	0	2.985e+00	4.033e+00	3.980e+00	4.975e+00	4.975e+00	5.586e+00
RG_20	0	8.958e+00	7.961e+00	8.959e+00	1.095e+01	7.790e+00	1.112e+01
S10	-10.5319	4.384e-03	4.384e-03	4.384e-03	4.384e-03	4.423e-03	4.384e-03
SS_5	0	2.821e-30	1.440e-18	1.077e-26	1.963e-26	2.246e-28	1.200e-25
SS_10	0	5.327e-27	4.490e-18	5.393e-25	4.904e-20	1.110e-25	1.951e-24

de avaliações da função objetivo dessa execução, não fazendo assim sentido calcular o desvio padrão, $St.Dev.$. As instâncias onde o algoritmo não foi capaz de encontrar uma solução admissível, dentro da tolerância de erro especificada, em nenhuma das execuções, são identificadas com '0' em SR. Os valores dentro de parêntesis correspondem ao valor da função objetivo da melhor execução (na coluna do número médio de avaliações da função) e a medida da proximidade da melhor solução obtida $x^{(k)}$ à solução ótima global conhecida x^* (na coluna do $St.Dev.$).

Dos resultados obtidos pode concluir-se que as penalidades (5.2.2) e (5.2.6) têm um comportamento de convergência comparável. A penalidade (5.2.2) resolve 59% das instâncias testadas, com 90-100% de execuções bem sucedidas (e 73% das instâncias com

80-100% de execuções bem sucedidas), e tem 14% das instâncias com apenas execuções mal sucedidas. A penalidade (5.2.6) resolve com sucesso 68% das instâncias, com 90-100% de execuções bem sucedidas (e 73% das instâncias com 80- 100% de execuções bem sucedidas), mas tem 18% das instâncias com apenas execuções mal sucedidas. Por outro lado, a penalidade (3.2.18) usada por Lucidi e Rinaldi (2010, 2013) é bem sucedida em 90-100% das execuções ao resolver 64% das instâncias (e 68% das instâncias com 80-100% de execuções bem sucedidas), e produz apenas execuções mal sucedidas em 23% das instâncias. É de salientar que o Algoritmo 12 associado ao termo de penalidade (5.2.2) apresenta um desempenho ligeiramente melhor do que com os outros dois termos de penalidade, com uma percentagem média total de execuções bem sucedidas de 75% contra 74% com a penalidade (5.2.6), e 72% com a penalidade (3.2.18).

Na experiência seguinte, foram usadas duas técnicas de otimização global bem conhecidas, disponíveis na literatura, para resolver os problemas contínuos (5.2.1) no contexto da penalidade exata: o procedimento de procura global determinístico e exato DIRECT (Jones, 2001); e o algoritmo global estocástico ponto-por-ponto SA. Para este último foi usada a função *simulannealbnd* da Global Optimization Toolbox$^{\text{TM}}$ do Matlab. Os resultados para comparação apresentam-se na Tabela 6.24. As condições de paragem do Algoritmo de penalidade 12 são

$$x^{(k)} \in W \ \text{ e } \ f(x^{(k)}) \leq f^* + 10^{-3} \ , \tag{6.2.1}$$

onde a tolerância do erro é $\sigma_{\min} = 10^{-3}$ e $f(x^{(k)})$ é a solução obtida pelo DIRECT e pelo SA, e é a melhor solução obtida (entre m) pelo FA adaptativo. Para estes testes, o valor inicial para $\varepsilon^{(1)}$ é 100. É de referir que, enquanto se usa nf_{avg} e $St.Dev.$ como critério para analisar a velocidade de convergência do FA adaptativo e SA, apenas é necessário o número de avaliações, nf, como critério de convergência para o DIRECT, desde que as soluções obtidas satisfaçam as condições (6.2.1), identificando-se nas colunas Suc com o valor 1. Por outro lado, se uma das condições definida em (6.2.1) não é satisfeita, o algoritmo de penalidade termina ao fim de 50 iterações, a execução é identificada com 0 (execução mal

sucedida), e a solução obtida é apresentada entre parêntesis.

Na Tabela 6.24 também se apresenta o tempo CPU, colunas identificadas com T, em segundos, necessário para obter a solução apresentada. Para o FA adaptativo e para o SA, os tempos correspondem ao valor médio dos tempos registados nas execuções bem sucedidas, T_{avg}, (entre as 10 execuções), ou o tempo da melhor execução quando todas as execuções são mal sucedidas. Todas as outras estatísticas relativas ao FA adaptativo têm o mesmo significado que as da tabela anterior. Para uma comparação equitativa entre o FA adaptativo, DIRECT e o SA, quando invocados no contexto do Algoritmo 12, usou-se o mesmo critério de paragem. Assim, cada algoritmo termina quando o número máximo de avaliações da função de penalidade excede um valor pré-definido $npenf_{\max}$, dado por 5000. Foram efetuados dois conjuntos de experiências com o SA: no primeiro conjunto, o SA termina quando são atingidos os 5000 cálculos da função; no segundo, usou-se o critério de paragem e os parâmetros definidos por defeito pelo Matlab. Observa-se que o Algoritmo 12 baseado no FA adaptativo é capaz de resolver (isto é, as condições definidas em (6.2.1) são satisfeitas em pelo menos uma execução) 82% das instâncias, e tem 90-100% de execuções bem sucedidas em 73% das instâncias. Em geral, as execuções mal sucedidas ocorrem quando se resolvem problemas de grande dimensão. O tempo total necessário para resolver todo o conjunto de problemas testados foi de 633 segundos.

Quando é usado o DIRECT para resolver os problemas do tipo (5.2.1), obtém-se velocidades de convergência diferentes. A versão baseada no critério de paragem $npenf_{\max}$=5000 é capaz de resolver com sucesso 55% das instâncias (Suc=1 na tabela), enquanto que a versão com os valores dos parâmetros definidos por defeito resolve apenas 32%.

O tempo total necessário pelo DIRECT com o critério $npenf_{\max}$=5000 para obter as soluções apresentadas para todos os problemas é de 426 segundos. A versão do DIRECT com os parâmetros definidos por defeito leva muito menos tempo (4.57 segundos), uma vez que a execução do algoritmo termina antes que uma boa solução seja atingida, devido aos valores dos parâmetros definidos por defeito ($npenf_{\max} = 20$ e o número máximo de iterações é 10).

Quando o Algoritmo 12 invoca o SA, conclui-se que as duas versões testadas, uma ba-

seada no critério $npenf_{\max}$ =5000 e outra baseada nos parâmetros definidos por defeito, resolve com sucesso (com 90-100% de execuções bem sucedidas) uma pequena percentagem de instâncias, 27% e 32%, respetivamente. No geral, a primeira versão produz uma solução que satisfaz as condições (6.2.1) em pelo menos uma execução para 45% das intâncias, contra 55% da segunda versão. Os tempos totais necessários para obter as soluções reportadas são da mesma grandeza: 1730 segundos para o SA com $npenf_{\max}$ =5000, e 3920 segundos para o SA com o critério de paragem com os valores definidos por defeito.

A partir da comparação entre o FA adaptativo e o DIRECT com o critério de paragem $npenf_{\max}$ =5000, conclui-se que o FA adaptativo tem melhor desempenho em termos de robustez (73% de execuções bem sucedidas contra 55%), embora exija muitas mais avaliações da função. Em termos do tempo total para obter as soluções reportadas de todas as instâncias, o FA adaptativo usa apenas 1,5 vezes mais segundos do que o DIRECT.

Quando é feita uma comparação entre algoritmos que convergem com sucesso para as soluções, o número de avaliações da função exigido pelo FA adaptativo é, em geral, um pouco maior do que o do SA (com o critério $npenf_{\max}$ =5000). No entanto, os tempos médios com o SA são geralmente superiores aos do FA adaptativo. Além disso, o FA adaptativo resolve com sucesso quase três vezes mais instâncias do que o SA.

Concluindo, as experiências numéricas realizadas para comparar a qualidade das soluções produzidas, bem como a velocidade de convergência do algoritmo de penalidade, mostram que a penalidade tangente hiperbólica produz resultados ligeiramente melhores do que as outras duas penalidades em comparação, em termos da qualidade da solução e percentagem média de execuções bem sucedidas. Além disso, o algoritmo de penalidade-FA adaptativo proposto mostra-se competitivo quando comparado o algoritmo de penalidade com o solver determinístico DIRECT ou com o solver SA.

Tabela 6.23: Resultados da velocidade de convergência, usando $\delta_{\min} = 10^{-4}$ e $\sigma_{\min} = 10^{-5}$

	penalidade (5.2.2)			penalidade (5.2.6)			penalidade 3.2.18		
Prob.	nf_{avg}	*St.Dev.*	SR	nf_{avg}	*St.Dev.*	SR	nf_{avg}	*St.Dev.*	SR
ACK_5	3.585e+04	6.860e+03	100	4.417e+04	1.360e+04	90	3.740e+04	3.785e+03	90
ACK_10	2.429e+05	1.449e+04	60	2.537e+05	4.038e+04	70	2.471e+05	5.254e+04	50
ACK_20	6.808e+05	–	10	(*1.9e+00*)	(*1.0e+00*)	0	(*1.8e+00*)	(*9.1e-01*)	0
AP	7.769e+03	1.596e+04	80	3.624e+03	2.881e+04	90	4.922e+03	6.051e+03	100
Bea	4.668e+03	4.316e+03	100	3.040e+03	1.972e+03	100	2.742e+03	1.533e+03	100
BL	3.958e+03	2.232e+03	100	4.368e+03	2.107e+03	100	3.762e+03	1.295e+03	100
BF1	3.585e+03	1.741e+03	100	4.476e+03	1.659e+03	100	3.777e+03	1.253e+03	100
Buk	1.202e+04	1.682e+04	100	1.305e+04	1.127e+04	100	6.092e+03	6.206e+03	100
DA	(*-2.5e+04*)	(*0.0e+00*)	0	(*-2.5e+04*)	(*0.0e+00*)	0	(*-2.5e+04*)	(*0.0e+00*)	0
DP_2	1.044e+04	1.854e+04	100	3.055e+03	2.264e+04	90	2.594e+03	1.108e+03	100
DP_4	1.226e+04	3.906e+03	100	1.410e+04	3.060e+03	90	1.059e+04	1.595e+04	70
Him	6.280e+03	5.827e+03	100	6.233e+03	2.557e+03	100	4.284e+03	1.574e+03	100
LM2_5	3.483e+04	9.766e+03	100	3.265e+04	4.510e+03	100	2.759e+04	5.135e+03	100
LM2_10	2.077e+05	2.556e+04	100	1.947e+05	2.139e+04	100	2.274e+05	2.818e+04	100
LM_20	5.856e+05	5.003e+04	80	5.670e+05	9.963e+04	90	8.435e+05	2.145e+04	80
NF2	9.558e+04	1.162e+05	100	5.870e+04	1.128e+05	90	2.832e+04	2.765e+04	100
RG_5	2.026e+05	2.147e+05	50	1.333e+05	1.467e+05	40	(*9.9e-01*)	(*9.9e-01*)	0
RG_10	(*2.0e+00*)	(*1.0e+00*)	0	(*1.0e+00*)	(*1.0e+00*)	0	(*4.0e+00*)	(*9.9e-01*)	0
RG_20	(*1.1e+01*)	(*1.0e+00*)	0	(*8.3e+00*)	(*1.0e+00*)	0	(*8.0e+00*)	(*1.0e+00*)	0
S10	3.954e+04	1.632e+05	80	4.345e+04	2.888e+04	80	2.447e+04	9.547e+03	100
SS_5	2.743e+04	5.375e+03	100	3.234e+04	6.598e+03	100	2.960e+04	6.394e+03	100
SS_10	1.962e+05	2.550e+04	100	2.155e+05	3.197e+04	100	2.025e+05	2.288e+04	100

Tabela 6.24: Comparação de nf_{avg}, nf, SR, Suc, T_{avg} e T, usando $\delta_{\min} = 10^{-3}$ e $\sigma_{\min} = 10^{-.3}$, e a penalidade (5.2.2).

	FA adaptativo$^{(a)}$				DIRECT$^{(a)}$			DIRECT (default†)			SA$^{(a)}$				SA (default‡)			
Prob.	nf_{avg}	St.D.	$\text{T}_{\text{avg}}{}^{\sharp}$	SR	nf	T§	Suc	nf	T§	Suc	nf_{avg}	St.D.	$\text{T}_{\text{avg}}{}^{\sharp}$	SR	nf_{avg}	St.D.	$\text{T}_{\text{avg}}{}^{\sharp}$	SR
ACK_5	1.865e+04	2.511e+03	3.6e+00	90	4.849e+03	1.1e+00	1	1.300e+01	2.1e-02	1	*(5.6e-02)*	*(1.7e-02)*	7.6e+01	0	*(1.7e+00)*	*(9.6e-01)*	7.6e+01	0
ACK_10	1.088e+05	1.721e+04	3.3e+01	80	5.147e+03	1.7e+00	1	2.300e+01	8.1e-03	1	*(7.9e+00)*	*(3.0e+00)*	1.6e+02	0	*(8.2e+00)*	*(3.0e+00)*	1.8e+02	0
ACK_20	*(3.6e+00)*	*(1.5e-03)*	1.4e+02	0	5.775e+03	3.0e+00	1	4.300e+01	2.2e-02	1	*(7.6e+01)*	*(8.3e+00)*	2.1e+02	0	*(1.5e+01)*	*(8.4e+00)*	5.0e+02	0
AP	8.541e+03	5.341e+03	7.8e-01	100	6.810e+02	9.9e-02	1	*(0.0e+00)*	1.8e-01	0	7.227e+03	4.831e+03	3.5e+00	100	1.094e+04	9.296e+03	5.3e+00	100
Bea	9.552e+03	4.998e+03	8.7e-01	100	1.029e+03	1.7e-01	1	*(6.8e-01)*	1.9e-01	0	2.253e+04	2.491e+04	1.1e+01	100	1.086e+04	7.553e+03	5.3e+00	100
BL	8.041e+03	4.232e+03	9.2e-01	100	2.663e+03	3.9e-01	1	*(3.1e+00)*	2.7e-01	0	5.433e+03	2.870e+03	2.7e+00	100	7.651e+03	4.375e+03	3.8e+00	100
BF1	9.544e+03	5.525e+03	1.1e+00	100	1.247e+03	2.0e-01	1	1.500e+01	4.0e-03	1	3.300e+03	2.232e+03	1.7e+00	100	2.364e+03	1.019e+03	1.2e+00	100
Buk	4.121e+04	5.279e+04	4.8e+00	100	*(1.0e+02)*	2.0e+00	0	*(1.0e+02)*	2.2e-01	0	2.359e+04	1.386e+04	1.2e+01	30	2.049e+04	1.014e+04	1.0e+01	80
DA	*(-2.5e+04)*	*(0.0e+00)*	3.0e+01	0	*(-2.5e+04)*	2.2e+00	0	*(-2.1e+04)*	2.0e-01	0	0.000e+00	0.000e+00	0.0e+00	0	1.307e+04	2.507e+03	1.3e+00	100
DP_2	1.106e+04	6.187e+03	1.1e+00	100	6.630e+02	9.6e-02	1	*(1.0e+00)*	1.7e-01	0	6.348e+03	5.685e+03	3.1e+00	90	5.253e+03	2.275e+03	2.5e+00	100
DP_4	2.919e+04	4.331e+04	2.9e+00	90	*(3.8e-03)*	6.7e+00	0	*(1.0e+00)*	2.2e-01	0	7.510e+04	1.906e+04	3.6e+01	40	5.335e+04	2.000e+04	2.6e+01	30
Him	2.713e+04	2.553e+04	3.1e+00	100	2.450e+02	4.9e-02	1	*(1.3e+01)*	1.7e-01	0	4.113e+03	1.479e+03	2.0e+00	100	4.254e+03	2.854e+03	2.1e+00	100
LM2_5	1.527e+04	2.445e+03	2.9e+00	100	*(5.1e-06)*	1.1e+01	0	*(9.9e-01)*	2.0e-01	0	*(2.4e-03)*	*(1.8e-01)*	7.6e+01	0	*(2.8e-03)*	*(9.2e-02)*	7.4e+01	0
LM2_10	6.428e+04	1.230e+04	2.0e+01	100	*(8.3e-03)*	3.5e+01	0	*(1.4e+00)*	3.9e-01	0	*(6.3e-01)*	*(1.3e+00)*	1.7e+02	0	*(6.0e-01)*	*(1.2e+00)*	1.8e+02	0
LM_20	2.134e+05	2.740e+04	1.2e+02	90	*(3.1e-01)*	1.5e+02	0	*(2.4e+00)*	1.1e+00	0	*(7.4e+00)*	*(2.9e+00)*	2.2e+02	0	*(4.6e+00)*	*(2.7e+00)*	8.9e+02	0
NF2	9.375e+04	8.267e+04	1.5e+01	90	*(4.3e-01)*	1.0e+01	0	*(2.6e+03)*	2.9e-01	0	8.213e+04	5.961e+04	4.5e+01	30	6.352e+04	2.893e+04	3.5e+01	40
RG_5	4.773e+04	2.952e+04	8.7e+00	60	5.597e+03	1.2e+00	1	1.300e+01	4.0e-03	1	*(1.6e-02)*	*(9.1e-03)*	7.4e+01	0	1.130e+04	–	6.5e+00	10
RG_10	*(2.7e+00)*	*(9.4e-01)*	7.9e+01	0	5.135e+03	1.7e+00	1	2.300e+01	7.8e-03	1	*(5.0e+00)*	*(1.0e+00)*	1.6e+02	0	*(5.0e+00)*	*(1.0e+00)*	1.8e+02	0
RG_20	*(1.1e+01)*	*(9.8e-01)*	1.4e+02	0	7.097e+03	3.7e+00	1	4.300e+01	2.3e-02	1	*(2.8e+01)*	*(3.0e+00)*	2.1e+02	0	*(3.4e+01)*	*(2.0e+00)*	1.5e+03	0
S10	3.805e+04	4.809e+04	6.8e+00	100	*(-1.1e+01)*	2.7e+00	0	*(-9.3e-01)*	2.8e-01	0	3.746e+04	2.020e+04	2.1e+01	70	4.729e+04	–	2.6e+01	10
SS_5	1.429e+04	4.050e+03	2.6e+00	100	*(1.2e+00)*	5.1e+01	0	*(9.4e+01)*	2.2e-01	0	*(1.3e-04)*	*(6.5e-03)*	7.8e+01	0	*(4.4e-05)*	*(3.2e-03)*	7.6e+01	0
SS_10	7.321e+04	1.583e+04	2.2e+01	100	*(2.9e+02)*	1.4e+02	0	*(3.4e+02)*	3.9e-01	0	*(3.7e-02)*	*(6.3e-02)*	1.6e+02	0	*(1.7e-02)*	*(3.9e-02)*	1.8e+02	0
Tempo total			6.33e+02			4.26e+02			4.57e+00				1.73e+03				3.92e+03	

$^{(a)}$ corresponde a $npenf_{\max}$ =5000;

† $npenf_{\max} = 20$ e o número máximo de iterações foi de 10;

‡ $npenf_{\max} = 3000n$ e outro critério tais como a variação média do valor da função, o tempo limite da resolução, limite da função objetivo (de entre seis critérios);

$^{\sharp}$ tempo médio, em segundos, nas resoluções bem sucedidas, ou o tempo utilizado na obtenção da solução se nenhuma resolução foi bem sucedida;

§ tempo de execução, em segundos, para encontrar a solução se Suc='1', ou o tempo de execução da obtenção da solução se Suc='0'.

6.2.4 Resultados computacionais do FAbRAO

Com o objetivo de analisar o desempenho da extensão do FA com base nas RAO na resolução de problemas de MINLP com restrições genéricas e variáveis mistas, foram realizadas experiências computacionais preliminares com os problemas apresentados na Tabela 6.14. O algoritmo foi codificado na linguagem de programação Matlab-versão 8.1 (R2013a). O FAbRAO foi executado 30 vezes para cada um dos problema teste e o tamanho da população considerado foi de $N = 30$. Os resultados obtidos encontram-se na Tabela 6.25, que apresenta a melhor solução obtida, f_{best}, e a sua correspondente violação, V_{best}, (das 30 execuções), a média de f dos 30 resultados, f_{avg}, e o número médio de avaliações da função objetivo, nfe, necessárias para encontrar, na iteração k, a solução $(x_{c1}^{(k)}, x_{z1}^{(k)})$ que satisfaz as condições do critério de paragem:

$$\frac{\left| f\left(x_{c1}^{(k)}, x_{z1}^{(k)}\right) - f^* \right|}{|f^*|} \leq 10^{-4} \quad \text{e} \quad V\left(x_{c1}^{(k)}, x_{z1}^{(k)}\right) \leq 10^{-3},$$

onde f^* é o mínimo global conhecido. Se estas condições não forem satisfeitas ao fim de $10N$ iterações, a execução do algoritmo termina e devolve a melhor solução encontrada até então. Compararam-se os resultados obtidos com os resultados de Filter-based GA (do inglês, *A filter-based genetic algorithm*) apresentados em (Hedar & Fahim, 2011), e com os resultados Extended ACO (do inglês, *An extended version of the ant colony optimization*) de Schlüter et al. (2009). Nestes estudos, os resultados foram obtidos usando $N = 20$, 30 execuções dos algoritmos, e o critério de paragem utilizado foi: 10 000 cálculos da função objetivo ou quando é obtida uma solução com um erro inferior a 10^{-4}.

Tabela 6.25: Comparação entre os resultados obtidos pelo FAbRAO e os apresentados em (Hedar & Fahim, 2011) e (Schlüter et al., 2009)

			Nosso estudo				Filter-based GA		Extended ACO	
Prob.	n_c/n_i	f^*	f_{best}	V_{best}	f_{avg}	nfe	f_{avg}	nfe	f_{avg}	nfe
ex 12.2.1*	2/3	7.66718	7.6671	1.87E-04	8.0695	8622	7.7406	4720	7.6672	363
ex 12.2.2*	2/1	1.07654	1.0766	0.00E+00	1.0767	5178	1.0767	8074	1.1459	4250
ex 12.2.3*	3/4	4.57958	4.5796	0.00E+00	4.7758	12157	5.1322	8125	4.5796	731
ex 12.2.6*	1/1	-17.0000	-17.0000	5.44E-07	-16.9998	3243	-17	999	-17	307
st_e13**	1/1	2.0000	2.0000	0.00E+00	2.0000	3409	2.0000	4530	–	–
f_2***	2/1	2.124	2.1320	2.75E-07	2.7149	5253	2.1852	3799	–	–
f_{10}***	1/1	-2.444	-2.4380	0.00E+00	-2.4380	3501	-2.4444	230	-2.4444	270
f_{11}***	2/1	3.2361	3.2361	0.00E+00	3.2361	4405	3.4208	5616	23.475	1180

*Problema teste incluídos em (Floudas et al., 2013); **Problema teste incluído em (Grossmann & Kravanja, 1995)

***Problema teste incluído em (Hedar & Fahim, 2011); – significa não disponível

Como se pode verificar pela análise da tabela, a integração das RAO na extensão do FA para a resolução de problemas de MINLP com restrições genéricas e variáveis mistas produz uma metodologia com bom desempenho. É de notar que não existe parâmetro de penalidade para inicializar e atualizar. Os resultados obtidos validam a metodologia proposta, uma vez que são bastante competitivos quando comparados com outras técnicas estocásticas disponíveis na literatura para resolver problemas de MINLP.

Capítulo 7

Conclusões e trabalho futuro

Na área da Otimização Não Linear, existem duas classes de problemas de otimização considerados de difícil resolução: os problemas de NLP e os problemas de MINLP, não convexos e não suaves. Uma classe de métodos habitualmente usada para resolver estes problemas é a dos métodos estocásticos. Recentemente, surgiu o FA que é um método estocástico baseado em populações, originalmente concebido para resolver BCOP. Devido ao seu sucesso na resolução de problemas complexos, que surgem no âmbito de aplicações reais, nas mais diversas áreas do conhecimento, este tem sido alvo de estudo pela comunidade científica.

Neste tese, o objetivo central prende-se com o desenvolvimento de extensões do FA para resolver problemas de NLP e de MINLP, com restrições, não convexos e não suaves. Quando os problemas de otimização envolvem restrições, quer de integralidade quer restrições genéricas, existem várias técnicas para o tratamento das restrições, donde se destacam as técnicas que usam funções de penalidade.

Para resolver os problemas de NLP com restrições genéricas, foram propostas duas extensões do FA: o algoritmo EXFA e o algoritmo FPAA. É de salientar que, nas extensões propostas, as técnicas usadas para o tratamento das restrições, são baseadas em informações produzidas ao longo do processo iterativo, e não necessitam da definição de qualquer parâmetro de penalidade. No contexto do desenvolvimento da EXFA foram apresentadas

duas novas técnicas para o tratamento das restrições: a ordenação estocástica e as regras de admissibilidade e otimalidade. A EXFA implementa um FA dinâmico, para melhorar capacidade de diversificação e de intensificação do FA. Para melhorar a precisão das soluções obtidas ao longo do processo iterativo, é implementado um procedimento de procura local no final de cada iteração. Os resultados mostraram que o algoritmo EXFA foi bastante eficaz na resolução de problemas NLP com restrições genéricas e bastante competitivo quando comparado com outras metaheurísticas existentes na literatura. No desenvolvimento do algoritmo FPAA, foi proposta uma função de penalidade auto-adaptativa, para o tratamento das restrições em problemas de NLP. Foi apresentado o estudo da equivalência entre os COP e BCOP reformulado com a função de penalidade auto-adaptativa proposta. Foi analisado o desempenho da função de penalidade auto-adaptativa quando implementada no FA e num conjunto de metaheurísticas. Verificou-se que, em geral, a função de penalidade auto-adaptativa, simples e fácil de codificar, quando implementada no contexto de outras metaheurísticas para resolver BCOP, é eficaz na procura de soluções ótimas globais de problemas COP contínuos. Para potenciar ainda mais a capacidade de intensificação do FA, implementou-se um procedimento de intensificação local baseado num estratégia de mutação no algoritmo FPAA. Por fim, apresentou-se análise de convergência do algoritmo, que tem em consideração a estrutura do algoritmo FPAA e as propriedades da função de penalidade auto-adaptativa proposta. Verificou-se que o procedimento de intensificação local permitiu encontrar soluções de boa qualidade, mas fez aumentar o esforço computacional. No entanto, o FPAA foi capaz de produzir soluções comparáveis e de alta qualidade, quando é permitido um número elevado de avaliações da função.

Para a resolução dos problemas de MINLP, não convexos e não suaves, foram propostas várias extensões do FA para resolver este tipo de problema. Começou-se por desenvolver a extensão do FA para resolver problemas de MINLP com restrições de limites simples e variáveis binárias, usando a técnica da discretização, designada por HBFA. Neste contexto, foram desenvolvidos três métodos diferentes de implementação das HBFA no âmbito de um FA dinâmico. Na técnica de discretização, foram analisadas três heurísticas, e foi apresentada uma nova função sigmóide, a função `erf` sigmóide. As três implementações HBFA

foram denotadas por HBFA-mEC, 'movimento no espaço contínuo', HBFA-mEB, 'movimento no espaço binário' e HBFA-pBC, 'Probabilidade para a componente binária'. Os resultados experimentais mostram que a implementação denotada por HBFA-mEC quando combinada quer com a nova função `erf` sigmóide quer com o esquema de arredondamento baseado na função chão, é bastante eficiente e superior em relação aos outros métodos em comparação. Além disso, as HBFA produzem soluções de elevada qualidade e superam as versões binárias de algumas metaheurísticas relatadas na literatura. Em particular, os resultados computacionais mostraram que a HBFA-mEC com a função `erf` sigmóide é a melhor, tanto em termos de custo computacional como de precisão.

A seguir, apresentou-se a extensão do FA para resolver problemas de MINLP com restrições de limites simples e variáveis contínuas e inteiras, usando o método de penalidade exata para o tratamento das restrições de integralidade. Neste contexto, foram propostas duas funções de penalidade exata, uma baseada na função tangente hiperbólica e outra baseada na função inversa do seno hiperbólico. Foi apresentado o estudo da equivalência entre os COP e BCOP reformulado com as funções de penalidade exata propostas. Apresentou-se o algoritmo de penalidade exata e a sua análise de convergência, a qual tem em conta a estrutura de um FA adaptativo e as propriedades das funções de penalidade exata propostas. Das experiências computacionais realizadas, concluiu-se que as funções de penalidade propostas têm um comportamento de convergência comparável. O algoritmo de penalidade com a função de penalidade baseada na função tangente hiperbólica apresenta um desempenho ligeiramente melhor, em termos da qualidade das soluções produzidas e da velocidade de convergência do algoritmo. Além disso, verificou-se que o algoritmo de penalidade com o FA adaptativo tem desempenho semelhante quando o algoritmo de penalidade invoca outros *solvers*.

Por fim, apresentou-se a extensão do FA para resolver problemas de MINLP com restrições genéricas e variáveis contínuas e inteiras, o algoritmo FAbRAO. Dada a existência de variáveis inteiras e contínuas foram propostas uma nova forma da geração inicial dos pontos da população e novas equações do movimento, de acordo com a natureza das variáveis de decisão envolvidas. Para o tratamento de restrições genéricas usaram-se as regras de

admissibilidade e otimalidade, enquanto que as restrições de integralidade foram tratadas por uma heurística de arredondamento. Os resultados obtidos validam a metodologia proposta, uma vez que mostraram ser bastante competitivos quando comparados com outras técnicas estocásticas disponíveis na literatura.

Como se pode constatar, nesta tese, as extensões propostas mostraram-se bastante competitivas quando comparadas com outros métodos de otimização global, estocásticos e determinísticos, para resolver problemas complexos de NLP e de MINLP. Face aos resultados produzidos, os algoritmos desenvolvidos revelaram-se promissores. No entanto, nalguns casos apresentam um número elevado de avaliações da função objetivo. Relativamente a este aspeto, pretende-se num futuro próximo desenvolver estratégias para a sua redução, através do uso de técnicas de geração de populações iniciais mais diversificadas para cobrir o espaço de procura de uma forma mais abrangente, de modo a direcionar a procura da solução ótima de uma forma mais célere. Outra questão em aberto, é o uso de uma população dinâmica de pontos, por redução e ampliação, através da eliminação de pontos da população que não sejam promissores, e inclusão de outros promissores, de forma a reduzir o custo computacional do algoritmo e o tempo de CPU, tornando-o mais eficiente. Por fim, a resolução de problemas complexos e de maior dimensão serão também foco para trabalho futuro.

Bibliografia

Abdel-Raouf, O., Abdel-Baset, M., & El-henawy, I. (2014). Chaotic firefly algorithm for solving definite integral. *International Journal of Information Technology and Computer Science (IJITCS)*, *6*(6), 19.

Abdullah, A., Deris, S., Mohamad, M. S., & Hashim, S. Z. M. (2012). A new hybrid firefly algorithm for complex and nonlinear problem. In *Distributed computing and artificial intelligence* (pp. 673–680). Springer.

Ali, M., & Kaelo, P. (2008). Improved particle swarm algorithms for global optimization. *Applied Mathematics and Computation*, *196*(2), 578 - 593.

Ali, M., & Zhu, W. (2013). A penalty function-based differential evolution algorithm for constrained global optimization. *Computational Optimization and Applications*, *54*(3), 707–739.

Ali, M. M., Khompatraporn, C., & Zabinsky, Z. B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, *31*(4), 635–672.

Arora, S., & Singh, S. (2013). The firefly optimization algorithm: convergence analysis and parameter selection. *International Journal of Computer Applications*, *69*(3).

Arora, S., Singh, S., Singh, S., & Sharma, B. (2014). Mutated firefly algorithm. In *2014 International conference on parallel, distributed and grid computing* (pp. 33–38).

Azad, M., Kalam, A., Rocha, A. M. A., & Fernandes, E. M. d. G. (2013). A simplified binary artificial fish swarm algorithm for uncapacitated facility location problems. In *World Congress on Engineering 2013, WCE 2013* (Vol. 1, pp. 31–36).

Bacanin, N., Brajevic, I., & Tuba, M. (2013). Firefly algorithm applied to integer programming problems. *Recent Adv Math*, 143–148.

Bacanin, N., & Tuba, M. (2014). Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint. *The Scientific World Journal*, *2014*.

Bäck, T., Fogel, D. B., & Michalewicz, Z. (2000). *Evolutionary computation 1: basic algorithms and operators* (Vol. 1). CRC Press.

Back, T., & Khuri, S. (1994). An evolutionary heuristic for the maximum independent set problem. In *Evolutionary computation, 1994. IEEE world congress on computational intelligence., Proceedings of the first IEEE conference* (pp. 531–535).

Baghlani, A., Makiabadi, M., & Sarcheshmehpour, M. (2014). Discrete optimum design of truss structures by an improved firefly algorithm. *Advances in Structural Engineering*, *17*(10), 1517–1530.

Barbosa, H. J., & Lemonge, A. C. (2002). An adaptive penalty scheme in genetic algorithms for constrained optimiazation problems. In *Gecco* (Vol. 2, pp. 287–294).

Barbosa, H. J., & Lemonge, A. C. (2008). *An adaptive penalty method for genetic algorithms in constrained optimization problems*. INTECH Open Access Publisher.

Barkat Ullah, A. S., Sarker, R., & Cornforth, D. (2008). Search space reduction technique for constrained optimization with tiny feasible space. In *Proceedings of the 10th annual conference on genetic and evolutionary computation* (pp. 881–888).

Becerra, R. L., & Coello, C. A. C. (2006). Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering*, *195*(33), 4303–4322.

Ben Hadj-Alouane, A., & Bean, J. C. (1997). A genetic algorithm for the multiple-choice integer program. *Operations research*, *45*(1), 92–101.

Birbil, Ş. İ., & Fang, S.-C. (2003). An electromagnetism-like mechanism for global optimization. *Journal of global optimization*, *25*(3), 263–282.

Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark

problems. *IEEE transactions on evolutionary computation*, *10*(6), 646–657.

Brest, J., Zumer, V., & Maucec, M. S. (2006). Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *2006 IEEE International conference on evolutionary computation* (pp. 215–222).

Bretthauer, K. M., & Shetty, B. (2002). The nonlinear knapsack problem âĂŞ algorithms and applications. *European Journal of Operational Research*, *138*(3), 459 - 472.

Burer, S., & Letchford, A. N. (2012). Non-convex mixed-integer nonlinear programming: a survey. *Surveys in Operations Research and Management Science*, *17*(2), 97–106.

Cagnina, L. C., Esquivel, S. C., & Coello, C. A. C. (2011). Solving constrained optimization problems with a hybrid particle swarm optimization algorithm. *Engineering Optimization*, *43*(8), 843–866.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, *191*(11), 1245–1287.

Coello, C. C., & Lechuga, M. S. (2002). Mopso: A proposal for multiple objective particle swarm optimization. In *Evolutionary computation, 2002. cec'02. proceedings of the 2002 congress on* (Vol. 2, pp. 1051–1056).

Costa, M. F. P., Francisco, R. B., Rocha, A. M. A., & Fernandes, E. M. (2016). Theoretical and practical convergence of a self-adaptive penalty algorithm for constrained global optimization. *Journal of Optimization Theory and Applications*. doi: 10.1007/s10957 -016-1042-7

Costa, M. F. P., Rocha, A. M. A., & Fernandes, E. M. (2014). An artificial fish swarm algorithm based hyperbolic augmented lagrangian method. *Journal of Computational and Applied Mathematics*, *259*, 868–876.

Costa, M. F. P., Rocha, A. M. A., Francisco, R. B., & Fernandes, E. M. (2014). Heuristic-based firefly algorithm for bound constrained nonlinear binary optimization. *Advances in Operations Research*, *2014*, Article ID 215182, 12 pages.

Costa, M. F. P., Rocha, A. M. A., Francisco, R. B., & Fernandes, E. M. (2016). Firefly penalty-based algorithm for bound constrained mixed-integer nonlinear program-

ming. *Optimization*, *65*(5), 1085–1104.

Costa, M. F. P., Rocha, A. M. A. C., & Fernandes, E. M. G. P. (2015). Combining non-dominance, objective-order and spread metric to extend firefly algorithm to multi-objective optimization. In A. Gaspar-Cunha, C. Henggeler Antunes, & C. C. Coello (Eds.), *Evolutionary multi-criterion optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, march 29 –april 1, 2015. Proceedings, Part I* (pp. 292–306). Cham: Springer International Publishing.

Costa, M. F. P., Rocha, A. M. A. C., Francisco, R., & Fernandes, E. M. G. P. (2017). Extension of the firefly algorithm and preference rules for solving minlp problems. In *International conference of numerical analysis and applied mathematics 2016 (ICNAAM 2016), Theodore E. Simos (Eds.).* AIP Conference Proceedings (no prelo).

Courant, R., et al. (1943). Variational methods for the solution of problems of equilibrium and vibrations. *Bull. Amer. Math. Soc*, *49*(1), 1–23.

Crawford, B., Soto, R., Olivares-Suarez, M., Palma, W., Paredes, F., Olguin, E., & Norero, E. (2014). A binary coded firefly algorithm that solves the set covering problem. *Rom. J. Inf. Sci. Technol*, *17*, 252–264.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, *186*(2), 311–338.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms* (Vol. 16). John Wiley & Sons.

Deb, K., & Goel, T. (1999). Evolutionary multi-criterion optimization. *Evolutionary algorithms in engineering and computer science*, 135–162.

Di Pillo, G., Lucidi, S., & Rinaldi, F. (2012). An approach to constrained global optimization based on exact penalty functions. *Journal of Global Optimization*, *54*(2), 251–260.

Dorigo, M. (1992). *Doctoral thesis: Optimization, learning and natural algorithms (in Italian).* Dipartimento di Elettronica, Politecnico di Milano: Italy.

Elsayed, S. M., Sarker, R. A., & Essam, D. L. (2011). GA with a new multi-parent crossover for constrained optimization. In *2011 IEEE Congress of Evolutionary Computation*

(CEC) (pp. 857–864).

Engelbrecht, A. P., & Pampara, G. (2007). Binary differential evolution strategies. In *2007 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1942–1947).

Falcon, R., Almeida, M., & Nayak, A. (2011). Fault identification with binary adaptive fireflies in parallel and distributed systems. In *2011 IEEE Congress of Evolutionary Computation (CEC)* (pp. 1359–1366).

Fan, Z., Liu, J., Sorensen, T., & Wang, P. (2009). Improved differential evolution based on stochastic ranking for robust layout synthesis of mems components. *IEEE Transactions on Industrial Electronics*, *56*(4), 937–948.

Farahani, S. M., Abshouri, A., Nasiri, B., & Meybodi, M. (2011). An improved firefly algorithm with directed movement. In *Proceedings of 4th IEEE International conference on computer science and information technology, Chengdu* (pp. 248–251).

Farahani, S. M., Abshouri, A., Nasiri, B., & Meybodi, M. (2012). Some hybrid models to improve firefly algorithm performance. *International Journal of Artificial Intelligence*, *8*(S12), 97–117.

Farmani, R., & Wright, J. A. (2003). Self-adaptive fitness formulation for constrained optimization. *IEEE Transactions on Evolutionary Computation*, *7*(5), 445–455.

Fiacco, A., & Jones, A. (1969). Generalized penalty methods in topological spaces. *SIAM Journal on Applied Mathematics*, *17*(5), 996-1000.

Fiacco, A. V., & McCormick, G. P. (1966). Extensions of sumt for nonlinear programming: equality constraints and extrapolation. *Management Science*, *12*(11), 816–828.

Fister, I., Yang, X.-S., & Brest, J. (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, *13*, 34–46.

Fister, I., Yang, X.-S., & Fister, I., Dušan e Fister Jr. (2014). Firefly algorithm: a brief review of the expanding literature. In *Cuckoo search and firefly algorithm* (pp. 347–360). Springer.

Fister Jr, I., Yang, X.-S., Fister, I., & Brest, J. (2012). Memetic firefly algorithm for combinatorial optimization. *arXiv preprint arXiv:1204.5165*.

Fletcher, R., & Leyffer, S. (2002). Nonlinear programming without a penalty function.

Mathematical Programming, *91*(2), 239–269.

Floudas, C. A., Pardalos, P. M., Adjiman, C., Esposito, W. R., Gümüs, Z. H., Harding, S. T., ... Schweiger, C. A. (2013). *Handbook of test problems in local and global optimization* (Vol. 33). Springer Science & Business Media.

Fonseca, L., Capriles, P. V., Barbc, H., & Lemonge, A. C. (2007). A stochastic rank-based ant system for discrete structural optimization. In *2007 IEEE Swarm Intelligence Symposium* (pp. 68–75).

Francisco, R. B., Costa, M. F. P., & Rocha, A. M. A. (2014). Experiments with firefly algorithm. In *International conference on computational science and its applications.* Part II, LNCS 8580, pp. 227–236. Springer.

Francisco, R. B., Costa, M. F. P., & Rocha, A. M. A. (2015). A firefly dynamic penalty approach for solving engineering design problems. In *International conference of numerical analysis and applied mathematics 2014 (ICNAAM 2014), Theodore E. Simos (Eds.).* Vol. 1648, p. 140010, AIP Conference Proceedings.

Francisco, R. B., Costa, M. F. P., & Rocha, A. M. A. (2016). Extensions of firefly algorithm for nonsmooth nonconvex constrained optimization problems. In *International conference on computational science and its applications.* Part I, LNCS 9786, pp. 402–417.

Francisco, R. B., Costa, M. F. P., Rocha, A. M. A., & Fernandes, E. M. (2016). Comparison of penalty functions on a penalty approach to mixed-integer optimization. In *International conference of numerical analysis and applied mathematics 2015 (ICNAAM 2015), Theodore E. Simos (Eds.).* Vol. 1738, p. 300008, AIP Conference Proceedings.

Freund, R. M. (2004). Penalty and barrier methods for constrained optimization. *Massachusetts Institute of Technology*.

Fu, J.-F., Fenton, R. G., & Cleghorn, W. L. (1991). A mixed integer-discrete-continuous programming method and its application to engineering design optimization. *Engineering Optimization*, *17*(4), 263–280.

Gallet, C., Salaun, M., & Bouchet, E. (2005). An example of global structural optimisation

with genetic algorithms in the aerospace field. In *VIII International Conference on Computational Plasticity.*

Gandomi, A., Yang, X.-S., Talatahari, S., & Alavi, A. (2013). Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, *18*(1), 89–98.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers and Structures*, *89*(23), 2325–2336.

Geem, Z. W., Kim, J. H., & Loganathan, G. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, *76*(2), 60–68.

Geng, H.-T., Song, Q.-X., Jiao, F., & Sun, Y.-J. (2009). An evolution strategy with stochastic ranking for solving reactive power optimization. In *Power electronics and intelligent transportation system (PEITS), 2nd International Conference* (Vol. 1, pp. 14–17).

Giannessi, F., & Tardella, F. (1999). Connections between nonlinear programming and discrete optimization. In D.-Z. Du & P. M. Pardalos (Eds.), *Handbook of combinatorial optimization: Volume1–3* (pp. 149–188). Boston, MA: Springer US.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning* (1st ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Grossmann, I. E., & Kravanja, Z. (1995). Mixed-integer nonlinear programming techniques for process systems engineering. *Computers and Chemical Engineering*, *19*, 189–204.

Guo, L., Wang, G.-G., Wang, H., & Wang, D. (2013). An effective hybrid firefly algorithm with harmony search for global numerical optimization. *The Scientific World Journal*, *2013*.

Hamda, H., & Schoenauer, M. (2000). Adaptive techniques for evolutionary topological optimum design. In *Evolutionary design and manufacture* (pp. 123–136). Springer.

Hansen, N. (2006). The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation* (pp. 75–102). Springer.

Hansen, N. (2016). The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolu-

tion strategies. *Evolutionary computation*, *9*(2), 159–195.

Hassanzadeh, T., & Meybodi, M. R. (2012). A new hybrid algorithm based on firefly algorithm and cellular learning automata. In *Electrical engineering (ICEE), 20th Iranian Conference* (pp. 628–633).

Hedar, A., & Fahim, A. (2011). Filter-based genetic algorithm for mixed variable programming. *Numer. Algebra Control Optim.*, *1*(1), 99–116.

Hinterding, R., & Michalewicz, Z. (1998). Your brains and my beauty: parent matching for constrained optimisation. In *Evolutionary computation proceedings. IEEE World congress on computational intelligence. IEEE International Conference* (p. 810-815).

Holland, J. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press. Retrieved from `https://books.google.pt/books?id=JE5RAAAAMAAJ`

Homaifar, A., Qi, C. X., & Lai, S. H. (1994). Constrained optimization via genetic algorithms. *Simulation*, *62*(4), 242–253.

Hongwei, Z., Liwei, T., & Dongzheng, W. (2015). Research on improved firefly optimization algorithm based on cooperative for clustering. *International Journal of Smart Home*, *9*(3), 205–214.

Hönig, U. (2010). A firefly algorithm-based approach for scheduling task graphs in homogeneous systems. *Proceeding Informatics*, 2010–724.

Hooke, R., & Jeeves, T. A. (1961). "Direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)*, *8*(2), 212–229.

Horng, M.-H., & Liou, R.-J. (2011). Multilevel minimum cross entropy threshold selection based on the firefly algorithm. *Expert Systems with Applications*, *38*(12), 14805–14811.

Huang, W.-C., Kao, C.-Y., & Horng, J.-T. (1994). A genetic algorithm approach for set covering problems. In *Evolutionary computation, 1994. IEEE World congress on computational intelligence. Proceedings of the first IEEE Conference on* (pp. 569–574).

Jiang, M., Luo, Y. P., & Yang, S. Y. (2007). Stochastic convergence analysis and para-

meter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, *102*(1), 8–16.

Joines, J. A., & Houck, C. R. (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In *Evolutionary computation, 1994. IEEE World congress on computational intelligence., Proceedings of the first IEEE Conference on* (pp. 579–584).

Jones, D. R. (2001). Direct global optimization algorithmdirect global optimization algorithm. In *Encyclopedia of optimization* (pp. 431–440). Springer.

Kannan, S., Slochanal, S. M. R., & Padhy, N. P. (2005). Application and comparison of metaheuristic techniques to generation expansion planning problem. *IEEE Transactions on Power Systems*, *20*(1), 466–475.

Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied mathematics and computation*, *214*(1), 108–132.

Karaboga, D., & Akay, B. (2011). A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing*, *11*(3), 3021–3031.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, *39*(3), 459–471.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21–57.

Kashan, M. H., Nahavandi, N., & Kashan, A. H. (2012). Disabc: a new artificial bee colony algorithm for binary optimization. *Applied Soft Computing*, *12*(1), 342–352.

Kazarlis, S., & Petridis, V. (1998). Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In *International conference on parallel problem solving from nature* (pp. 211–220).

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *IEEE International conference on neural networks (Perth, Australia)* (pp. 1942–1948). IEEE Service Center, Piscataway, NJ.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Systems, man, and cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on* (Vol. 5, pp. 4104–4108).

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, *34*(5-6), 975–986.

Koh, A. (2009). An adaptive differential evolution algorithm applied to highway network capacity optimization. In *Applications of soft computing* (pp. 211–220). Springer.

Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evol. Comput.*, *7*(1), 19–44.

Lemonge, A. C., Barbosa, H. J., & Bernardino, H. S. (2012). A family of adaptive penalty schemes for steady-state genetic algorithms. In *2012 IEEE Congress on Evolutionary Computation* (pp. 1–8).

Lemonge, A. C., Barbosa, H. J., & Bernardino, H. S. (2015). Variants of an adaptive penalty scheme for steady-state genetic algorithms in engineering optimization. *Engineering Computations*, *32*(8), 2182–2215.

Leyffer, S. (1993). *Deterministic methods for mixed integer nonlinear programming* (PhD thesis). Department of Mathematics & Computer Science, University of Dundee.

Lin, X., Zhong, Y., & Zhang, H. (2013). An enhanced firefly algorithm for function optimisation problems. *International Journal of Modelling, Identification and Control*, *18*(2), 166–173.

Liu, B., Ma, H., Zhang, X., & Zhou, Y. (2007). A memetic co-evolutionary differential evolution algorithm for constrained optimization. In *2007 IEEE Congress on Evolutionary Computation* (pp. 2996–3002).

Liu, R., Li, Y., Zhang, W., & Jiao, L. (2009). Stochastic ranking based differential evolution algorithm for constrained optimization problem. In *Proceedings of the first acm/sigevo summit on genetic and evolutionary computation* (pp. 887–890).

Liu, T., Zhang, L., & Zhang, J. (2013). Study of binary artificial bee colony algorithm based on particle swarm optimization. *Journal of Computational Information Systems*, *9*(16), 6459–6466.

Liuzzi, G., Lucidi, S., & Rinaldi, F. (2012). Derivative-free methods for bound constrained mixed-integer optimization. *Computational Optimization and Applications*, *53*(2), 505–526.

Liuzzi, G., Lucidi, S., & Rinaldi, F. (2015). Derivative-free methods for mixed-integer constrained optimization problems. *Journal of Optimization Theory and Applications*, *164*(3), 933–965.

Long, N. C., Meesad, P., & Unger, H. (2015). A highly accurate firefly based algorithm for heart disease prediction. *Expert Systems with Applications*, *42*(21), 8221–8231.

Lucidi, S., & Rinaldi, F. (2010). Exact penalty functions for nonlinear integer programming problems. *Journal of optimization theory and applications*, *145*(3), 479–488.

Lucidi, S., & Rinaldi, F. (2013). An exact penalty global optimization approach for mixed-integer programming problems. *Optimization Letters*, *7*(2), 297–307.

Maidl, G., de Lucena, D. S., & dos Santos Coelho, L. (2013). Economic dispatch optimization of thermal units based on a modified firefly algorithm. In *22st International Congress of Mechanical Engineering (COBEM 2013)*.

Mani, A., & Patvardhan, C. (2009). A novel hybrid constraint handling technique for evolutionary optimization. In *2009 IEEE Congress on Evolutionary Computation* (pp. 2577–2583).

Mezura-Montes, E., & Coello, C. A. C. (2008). Constrained optimization via multiobjective evolutionary algorithms. In *Multiobjective problem solving from nature* (pp. 53–75). Springer.

Mezura-Montes, E., & Coello, C. A. C. (2011). Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, *1*(4), 173–194.

Mezura-Montes, E., Coello, C. A. C., & Tun-Morales, E. I. (2004). Simple feasibility rules and differential evolution for constrained optimization. In *Mexican International Conference on Artificial Intelligence* (pp. 707–716).

Mezura-Montes, E., & Palomeque-Ortiz, A. G. (2009). Parameter control in differential evolution for constrained optimization. In *2009 IEEE Congress on Evolutionary*

Computation (pp. 1375–1382).

Mezura-Montes, E., Velázquez-Reyes, J., & Coello, C. C. (2006). Modified differential evolution for constrained optimization. In *2006 IEEE International Conference on Evolutionary Computation* (pp. 25–32).

Michalewicz, Z. (1995). Genetic algorithms, numerical optimization, and constraints. In *Proceedings of the sixth international conference on genetic algorithms* (Vol. 195, pp. 151–158).

Mirjalili, S., & Lewis, A. (2013). S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation*, *9*, 1–14.

Murray, W., & Ng, K.-M. (2010). An algorithm for nonlinear optimization problems with binary variables. *Computational Optimization and Applications*, *47*(2), 257–288.

Nebti, S., & Boukerram, A. (2010). Handwritten digits recognition based on swarm optimization methods. In *International conference on networked digital technologies* (pp. 45–54).

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Obadage, A., & Harnpornchai, N. (2006). Determination of point of maximum likelihood in failure domain using genetic algorithms. *International Journal of Pressure Vessels and Piping*, *83*(4), 276–282.

Olsen, A. L. (1994). Penalty functions and the knapsack problem. In *Evolutionary computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the first IEEE Conference on* (pp. 554–558).

Onwubolu, G. C., & Babu, B. (2013). *New optimization techniques in engineering* (Vol. 141). Springer.

Padberg, M. (2012). Harmony search algorithms for binary optimization problems. In *Operations research proceedings 2011* (pp. 343–348). Springer.

Palit, S., Sinha, S. N., Molla, M. A., Khanra, A., & Kule, M. (2011). A cryptanalytic attack on the knapsack cryptosystem using binary firefly algorithm. In *Int. conf. on computer and communication technology (ICCCT)* (Vol. 2, pp. 428–432).

Pampara, G., & Engelbrecht, A. P. (2011). Binary artificial bee colony optimization. In *Swarm Intelligence (SIS), 2011 IEEE symposium on* (pp. 1–8).

Pampara, G., Engelbrecht, A. P., & Franken, N. (2006). Binary differential evolution. In *2006 IEEE International Conference on Evolutionary Computation (CEC)* (pp. 1873–1879).

Petalas, Y. G., Parsopoulos, K. E., & Vrahatis, M. N. (2007). Memetic particle swarm optimization. *Annals of Operations Research*, *156*(1), 99–127.

Rezaee Jordehi, A., & Jasni, J. (2013). Parameter selection in particle swarm optimisation: a survey. *Journal of Experimental & Theoretical Artificial Intelligence*, *25*(4), 527–542.

Richardson, J. T., Palmer, M. R., Liepins, G. E., & Hilliard, M. (1989). Some guidelines for genetic algorithms with penalty functions. In *Proceedings of the third international conference on genetic algorithms* (pp. 191–197). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rinaldi, F. (2009). New results on the equivalence between zero-one programming and continuous concave programming. *Optimization Letters*, *3*(3), 377–386.

Rocha, A. M. A., & Fernandes, E. M. d. G. (2009). Self-adaptive penalties in the electromagnetism-like algorithm for constrained global optimization problems. In *8th World congress on structural and multidisciplinary optimization. june 15, 2009. Lisbon, Portugal.*

Rocha, A. M. A., Fernandes, E. M. d. G., Fernandes, J. P., & Martins, T. F. (2010). Embedding a competitive ranking method in the artificial fish swarm algorithm for global optimization. In *2nd International Conference on Engineering Optimization-EngOpt 2010* (pp. 1271–1280).

Rogalsky, T., & Derksen, R. (2000). Hybridization of differential evolution for aerodynamic design. In *Proceedings of the 8th Annual Conference of the Computational Fluid Dynamics Society of Canada* (pp. 729–736).

Runarsson, T. P., & Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, *4*(3), 284–294.

Santos Coelho, L., de Andrade Bernert, D. L., & Mariani, V. C. (2011). A chaotic firefly algorithm applied to reliability-redundancy optimization. In *2011 IEEE Congress of Evolutionary Computation (CEC)* (pp. 517–521).

Santos Coelho, L., & Mariani, V. C. (2008). Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. *Expert Systems with Applications*, *34*(3), 1905–1913.

Sayadi, M. K., Hafezalkotob, A., & Naini, S. G. J. (2013). Firefly-inspired algorithm for discrete optimization problems: an application to manufacturing cell formation. *Journal of Manufacturing Systems*, *32*(1), 78–84.

Schlüter, M., Egea, J. A., & Banga, J. R. (2009). Extended ant colony optimization for non-convex mixed integer nonlinear programming. *Computers and Operations Research*, *36*(7), 2217–2229.

Selvan, S. E., Xavier, C. C., Karssemeijer, N., Sequeira, J., Cherian, R. A., & Dhala, B. Y. (2006). Parameter estimation in stochastic mammogram model by heuristic optimization techniques. *IEEE Transactions on Information Technology in Biomedicine*, *10*(4), 685–695.

Sevkli, M., & Guner, A. R. (2006). A continuous particle swarm optimization algorithm for uncapacitated facility location problem. In *International Workshop on Ant Colony Optimization and Swarm Intelligence* (pp. 316–323).

Shafaati, M., & Mojallali, H. (2012). Modified firefly optimization for iir system identification. *Journal of Control Engineering and Applied Informatics*, *14*(4), 59–69.

Shakarami, M. R., & Sedaghati, R. (2014). A new approach for network reconfiguration problem in order to deviation bus voltage minimization with regard to probabilistic load model and dgs. *World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, *8*(2), 430–435.

Shandiz, R. A., & Mahdavi-Amiri, N. (2011). An exact penalty approach for mixed integer nonlinear programming problems. *American Journal of Operations Research*, *1*(03), 185.

Silva, E. K., Barbosa, H. J., & Lemonge, A. C. (2011). An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization. *Optimization and Engineering*, *12*(1-2), 31–54.

Storn, R. (1996). On the usage of differential evolution for function optimization. In *Fuzzy information processing society, 1996. NAFIPS., 1996 Biennial Conference of the North American* (pp. 519–523).

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341–359.

Suganthan, P. N. (1999). Particle swarm optimiser with neighbourhood operator. In *Evolutionary Computation. CEC 99. Proceedings of the 1999 Congress on* (Vol. 3).

Tessema, B., & Yen, G. G. (2006). A self adaptive penalty function based algorithm for constrained optimization. In (pp. 246–253). IEEE.

Tessema, B., & Yen, G. G. (2009). An adaptive penalty formulation for constrained evolutionary optimization. In (Vol. 39, pp. 565–578). IEEE.

Tilahun, S. L., & Ong, H. C. (2012). Modified firefly algorithm. *Journal of Applied Mathematics*, *vol. 2012, Article ID 467631*, 12 pages.

Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K., & Rossi, T. (2008). An enhanced memetic differential evolution in filter design for defect detection in paper production. *Evolutionary Computation*, *16*(4), 529–555.

Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, *85*(6), 317–325.

Vahedi Nouri, B., Fattahi, P., & Ramezanian, R. (2013). Hybrid firefly-simulated annealing algorithm for the flow shop problem with learning effects and flexible maintenance activities. *International Journal of Production Research*, *51*(12), 3501–3515.

Wang, B., Li, D.-X., Jiang, J.-P., & Liao, Y.-H. (2016). A modified firefly algorithm based on light intensity difference. *J. Comb. Optim.*, *31*(3), 1045–1060.

Wang, L., & Singh, C. (2009). Unit commitment considering generator outages through a mixed-integer particle swarm optimization algorithm. *Applied soft computing*, *9*(3),

947–953.

Wang, L., Yang, R., Xu, Y., Niu, Q., Pardalos, P. M., & Fei, M. (2013). An improved adaptive binary harmony search algorithm. *Information Sciences*, *232*, 58–87.

Yang, X. (2010). Firefly algorithm, stochastic test functions and design optimisation. *Int. J. Bio-Inspired Comput.*, *2*(2), 78–84.

Yang, X.-S. (2008). *Nature-inspired metaheuristic algorithms (First edition).* Luniver Press.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *International Symposium on Stochastic Algorithms* (pp. 169–178).

Yang, X.-S. (2013). Multiobjective firefly algorithm for continuous optimization. *Engineering with Computers*, *29*(2), 175–184.

Yang, X.-S., & He, X. (2013). Firefly algorithm: recent advances and applications. *International Journal of Swarm Intelligence*, *1*(1), 36–50.

Yang, X.-S., Hosseini, S. S. S., & Gandomi, A. H. (2012). Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. *Applied soft computing*, *12*(3), 1180–1186.

Yao, B., Yang, C., Hu, J., & Yu, B. (2010). The optimization of urban subway routes based on artificial bee colony algorithm. In F. Chen, L. Gao, & Y. Bai (Eds.), *Key technologies of railway engineeringhigh speed railway, heavy haul railway and urban rail transit* (pp. 747–751). Beijing Jiaotong University, Beijing.

Yiqing, L., Xigang, Y., & Yongjian, L. (2007). An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints. *Computers & chemical engineering*, *31*(3), 153–162.

Yu, C., Teo, K. L., & Bai, Y. (2013). An exact penalty function method for nonlinear mixed discrete programming problems. *Optimization Letters*, *7*(1), 23–38.

Yu, S., Yang, S., & Su, S. (2013). Self-adaptive step firefly algorithm. *Journal of Applied Mathematics*, *2013*.

Zavislak, M. J. (2004). Constraint handling in groundwater remediation design with genetic algorithms. *University of Illinois at Urbana-Champaign*.

Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, *178*(15), 3043–3074.

Zhao, B., Guo, C., & Cao, Y. (2005). A multiagent-based particle swarm optimization approach for optimal reactive power dispatch. *IEEE Transactions on Power Systems*, *20*(2), 1070–1078.

Zielinski, K., Wang, X., & Laur, R. (2008). Comparison of adaptive approaches for differential evolution. In *International Conference on Parallel Problem Solving from Nature* (pp. 641–650).