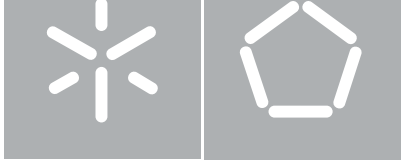
Universidade do Minho
Escola de Engenharia

Alzira Maria Teixeira da Mota

Um método de redução para programação semi-infinita não linear baseado numa técnica de penalidade exacta

Abril de 2010

Um método de redução para programação semi-infinita não linear baseado numa técnica de penalidade exacta
Alzira Mota
UMinho | 2010



Universidade do Minho
Escola de Engenharia

Alzira Maria Teixeira da Mota

Um método de redução para programação semi-infinita não linear baseado numa técnica de penalidade exacta

Tese de Doutoramento
Engenharia Industrial e de Sistemas

Trabalho efectuado sob a orientação do
Professor Doutor António Ismael de Freitas Vaz

Abril de 2010

Universidade do Minho, ___/___/_____

Assinatura: _____

Dedicatória

ao meu irmão Mário

Agradecimentos

A minha profunda gratidão ao Doutor Ismael Vaz pela honra dada em ser meu orientador, de ter partilhado a sua sabedoria e revelado uma atenção e disponibilidade inesgotáveis na concretização deste projecto.

Agradeço ao Instituto Superior de Engenharia do Porto por ter prestado apoio financeiro nos últimos dois anos deste projecto.

Os meus agradecimentos também são dirigidos ao Departamento de Matemática do Instituto Superior de Engenharia do Porto pela dispensa de serviço de alguns meses, na recta final deste trabalho.

Aos colegas Ana Moura, Carla Pinto, Fernando Carvalho e Teresa Costa pela prestabilidade e simpatia manifestadas.

Aos meus pais, ao meu irmão, à minha prima Matilde e ao meu Paolo um "Obrigada".

Resumo

Os problemas de programação semi-infinita (PSI) aparecem nas mais diversas áreas da Engenharia, tais como, no planeamento da trajectória de robôs, no controlo da poluição atmosférica, no planeamento da produção, no desenho óptimo de conjuntos de sinais e desenhos de filtros digitais.

Esta tese é dedicada a problemas de PSI não linear na sua forma mais geral. Os problemas considerados são caracterizados por possuírem um número finito de variáveis e um conjunto infinito de restrições.

Os métodos numéricos existentes para a resolução de problemas de PSI podem ser divididos em três classes principais: métodos de discretização, métodos das trocas e métodos de redução.

Os métodos de redução são os que possuem melhores propriedades teóricas de convergência. São, também, os mais exigentes em termos numéricos uma vez que exigem a resolução de problemas auxiliares, em que se pretende a determinação de todos os óptimos globais e locais (optimização multi-local).

Nas últimas décadas foram apresentados vários algoritmos para problemas de PSI. Contudo há pouco *software* disponível e nenhum fornece uma implementação de um método pertencente à classe de redução.

Neste trabalho é proposto um algoritmo de redução local baseado na técnica de penalidade. A função usada considera uma extensão de uma função de penalidade de norma-∞ aumentada. A escolha desta função de penalidade para propor a extensão às restrições finitas deve-se à obtenção de melhores resultados numéricos para um conjunto de problemas

teste de PSI sem restrições finitas, em comparação com as funções de penalidade baseadas na norma 1, 2 e ∞ da violação das restrições. Fez-se o estudo das propriedades teóricas da função de penalidade estendida.

É feita uma implementação do algoritmo de redução local proposto. O *solver* desenvolvido é designado por SIRedAl (*Semi-Infinite Reduction Algorithm*). Este *solver* foi implementado em MATLAB e é capaz de resolver problemas de PSI na forma mais geral com dimensão infinita máxima de 2. O código do *solver* usa dois algoritmos diferentes na minimização da função de penalidade e dois na resolução dos problemas multi-locais. O *solver* foi testado com 117 problemas teste da base de dados SIPAMPL e os resultados numéricos confirmaram a potencialidade do algoritmo proposto.

Abstract

Semi-infinite programming (SIP) problems arise in several engineering areas such as, for example, robotic trajectory planning, production planning, digital filter design and air pollution control.

This thesis is devoted to SIP problems in the most general form. These problems are characterized to have a finite number of variables and an infinite set of constraints.

The existing numerical methods for solving SIP problems can be divided into three major classes: discretization, exchange and reduction type methods. The reduction type methods are the ones with better theoretical properties, but they are also the most demanding in computation terms, since they require to solve sub-problems to the local and global optimality (multi-local optimization).

In last decades several algorithms were proposed for SIP, but there are not many publicly available software and none provides an implementation of a method belonging to the reduction type class.

In this work we propose a reduction type algorithm based on a penalty technique. The penalty function used is an extension of a penalty function of ∞-norm, allowing the inclusion of finite constraints. In order to define the best penalty function, a numerical study of penalty functions based on the standard 1, 2 and ∞ norms are performed, considering test problems without finite constraints. A theoretical study of the extended penalty function is also performed.

The proposed reduction algorithm is implemented in a solver coined as SIRedAl (Semi-Infinite Reduction Algorithm). The solver has been implemented in MATLAB and is

capable of solving SIP problems in the most general form with a maximum of two infinite variables. The solver code uses two different algorithms in the minimization of the penalty function and also two different algorithms for solving the multi-local problems. The solver has been tested with 117 test problems from the database SIPAMPL and numerical results confirmed the algorithm potential.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Neste capítulo é feita uma introdução aos problemas de programação semi-infinita. É apresentada a sua formulação matemática e as suas principais características. São abordadas as possíveis aplicações de PSI, sendo dado um exemplo prático de aplicação. Faz-se, também, referência ao *software* existente para a programação semi-infinita.

Na Secção 1.1 é descrito o problema de programação semi-infinita (PSI). As áreas de aplicação e a descrição de um problema prático de PSI são apresentados na Secção 1.2. A motivação do trabalho é dada na Secção 1.3. A contribuição do trabalho e a estrutura da tese são descritos nas Secções 1.4 e 1.5, respectivamente.

1.1 Descrição do problema

Um problema de programação semi-infinita pode ser descrito, na sua forma mais geral, da seguinte forma:

$$\begin{aligned} & \min_{x \in R^n} f(x) \\ \text{s.a } & g_i(x,t) \leq 0 \quad i = 1, \ldots, m \\ & h_i(x) = 0 \quad i = 1, \ldots, o \\ & h_i(x) \leq 0 \quad i = o+1, \ldots, q \\ & \forall t \in T \subset R^p, \end{aligned} \tag{1.1.1}$$

onde T é um conjunto infinito, usualmente um produto cartesiano de intervalos com limites finitos do tipo $T = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \cdots \times [\alpha_p, \beta_p]$. A função objectivo é denotada por $f(x)$, as restrições $g_i(x,t) \leq 0$, $i = 1, \ldots, m$, são do tipo infinito (designadas por restrições infinitas) e $h_i(x) = 0$, $i = 1, \ldots, o$, $h_i(x) \leq 0$, $i = o+1, \ldots, q$, são as restrições finitas de igualdade e desigualdade, respectivamente. Um problema é de PSI se tiver pelo menos uma restrição infinita, isto é, $m > 0$.

A designação programação semi-infinita surge quando um problema tem um número finito de variáveis sujeito a um número infinito de restrições ou um número infinito de variáveis sujeito a um número finito de restrições. No problema (1.1.1) o conjunto T é infinito e as restrições infinitas, $g_i(x,t) \leq 0$, $i = 1, \ldots, m$, podem ser vistas como um conjunto infinito de restrições ordinárias indexadas à variável t. Neste trabalho são tratados apenas problemas com um número finito de variáveis e um número infinito de restrições.

Quando o conjunto T é função das variáveis x, $T := T(x)$, considera-se que o problema é de programação semi-infinita generalizada ([60, 81, 91, 92, 100, 102, 103, 134, 139]). Se T não é função das variáveis x então pertence à classe de problemas de PSI padrão. Neste trabalho consideram-se apenas problemas de PSI padrão.

De acordo com as características matemáticas das funções objectivo e das restrições, os problemas de optimização podem ser divididos em Problemas de Optimização Linear (PL) e Problemas de Optimização Não Linear (PNL). Um problema de PSI é linear se ambas as funções objectivo e restrições são lineares em x. Os problemas de PSI não linear admitem pelo menos uma das funções f, g_i, $i = 1, \ldots, m$, ou h_i, $i = 1, \ldots, q$, não lineares em x. Um problema diz-se quadrático se a função objectivo é quadrática e as restrições são lineares.

Este tipo de problemas integra a classe PNL. No entanto, devido às suas características são por vezes tratados como uma classe à parte. O trabalho desenvolvido aplica-se a problemas de PSI não linear. Sempre que necessário faz-se referência aos problemas lineares e quadráticos e também assume-se que as funções que definem o problema são continuamente diferenciáveis em todos os argumentos e limitadas no seu domínio.

1.2 Aplicações da PSI

Os problemas de PSI aparecem nas mais diversas áreas da Engenharia, desde a teoria da aproximação de Chebyshev [35, 36, 93], o planeamento de produção [58, 135], o desenho óptimo de conjuntos de sinais [24, 52, 123], os desenhos de filtros digitais [25, 82, 84, 85, 86, 87, 97], os problemas de controlo da poluição [11, 26, 30, 38, 130] e no planeamento da trajectória de robôs [31, 33, 64, 105, 115, 120, 127]. Um problema característico é a determinação da trajectória de um robô onde são apenas conhecidos alguns pontos da trajectória. O problema de PSI consiste na minimização do tempo de deslocação quando são tomadas em consideração algumas limitações físicas do próprio robô ao longo do tempo.

Para ilustrar a formulação de um problema de PSI, apresenta-se um exemplo de controlo de poluição atmosférica descrito em [130]. Por exemplo, um problema de PSI consiste na optimização da função objectivo (minimizar a altura de uma chaminé) de forma a manter a poluição do ar abaixo de um determinado limiar ao nível do solo, numa dada região.

Um dos possíveis modelos de dispersão atmosférica é denominado por modelo Gaussiano. Para escrever as equações do modelo matemático Gaussiano é considerado um sistema de eixos de coordenadas ortogonal em que a origem representa o nível do solo. Observando a Figura 1.1, a e b são as abcissas e ordenadas, respectivamente, do ponto de emissão dos poluentes. A emissão da poluição na chaminé ocorre numa altura x acima do nível do solo.

Assumindo que a propagação do penacho segue uma distribuição Gaussiana, a concentração, $\mathcal{C}$, do gás ou aerossóis (partículas com menos de 20 microns de diâmetro) na posição

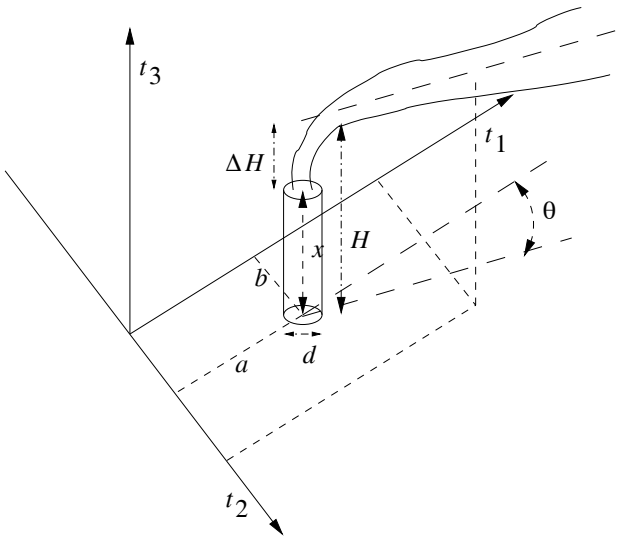


Figura 1.1: Sistema de coordenadas e notação

t_1, t_2 e t_3 de uma fonte contínua com altura de emissão efectiva, $\mathcal{H}$, é dada por:

$$\mathcal{C}(t_1,t_2,t_3,\mathcal{H}) = \frac{\mathcal{Q}}{2\pi\sigma_{t_2}\sigma_{t_3}\mathcal{U}} e^{-\frac{1}{2}\left(\frac{t_2}{\sigma_{t_2}}\right)^2} \left(e^{-\frac{1}{2}\left(\frac{t_3-\mathcal{H}}{\sigma_{t_3}}\right)^2} + e^{-\frac{1}{2}\left(\frac{t_3+\mathcal{H}}{\sigma_{t_3}}\right)^2} \right) \tag{1.2.1}$$

onde $\mathcal{Q}(gs^{-1})$ é a taxa uniforme de emissão, $\mathcal{U}(ms^{-1})$ é velocidade média do vento afectando o penacho, $\sigma_{t_2}(m)$ e $\sigma_{t_3}(m)$ são os desvios padrão do penacho distribuídos pelos planos horizontal e vertical, respectivamente. $\mathcal{Y}$ é dado por:

$$\mathcal{Y} = (t_1 - a)sin(\theta) + (t_2 - b)cos(\theta), \tag{1.2.2}$$

onde $\theta(rad)$ é a direcção média do vento ($0 \leq \theta \leq 2\pi$).

Na equação (1.2.1) a variável t_1 não aparece implicitamente na fórmula, mas os desvios σ_{t_2} e σ_{t_3} dependem da variável $\mathcal{X}$ que é definida como:

$$\mathcal{X} = (t_1 - a)cos(\theta) - (t_2 - b)sin(\theta). \tag{1.2.3}$$

As equações (1.2.2) e (1.2.3) fazem uma alteração das coordenadas do ponto de emissão dos poluentes na direcção média do vento.

Os parâmetros Gaussianos σ_{t_2} e σ_{t_3} são uma função da distância a partir da fonte que tem em conta a turbulência atmosférica.

A altura de emissão efectiva, $\mathcal{H}$, é a soma da altura física da chaminé, $x(m)$, com a elevação do penacho, $\Delta\mathcal{H}(m)$, em que:

$$\Delta\mathcal{H} = \frac{V_o d}{\mathcal{U}}\left(1.5 + 2.68\frac{T_o - T_e}{T_o}d\right),$$

onde $d(m)$ é o diâmetro interno da chaminé, $V_o(ms^{-1})$ é a velocidade de saída do gás na chaminé, $T_o(K)$ é a temperatura de saída do gás e $T_e(K)$ é a temperatura ambiente.

Considerando um cenário com várias fontes (chaminés), a notação usada acima é estendida. Passando a ser $\mathcal{C}_i$, $i = 1, \ldots, n$ a contribuição da fonte i para o total de concentração de poluentes, onde n é o número de chaminés distribuídas numa dada região. A utilização do índice i é, também aplicada aos parâmetros.

Assumindo, ainda, que os poluentes são quimicamente inertes, a sua composição pode ser calculada através da sobreposição das n fontes de poluentes. A concentração de poluentes numa dada região com coordenadas t_1, t_2 e t_3 pode ser calculada somando a contribuição individual de cada fonte $\sum_{i=1}^{n} \mathcal{C}_i\,(t_1, t_2, t_3, \mathcal{H}_i)$.

Numa fase de planeamento, um problema de controlo de poluição em que se pretenda minimizar a altura das chaminé, mantendo o nível de poluição abaixo de um limiar $\mathcal{C}_0$, numa região $\mathcal{R}$, a nível do solo, pode ser formulado pelo seguinte problema de PSI:

$$\begin{aligned}
&\min_{(x_1,\ldots,x_n)\in R^n} \sum_{i=1}^{n} c_i x_i \\
&s.a \quad \sum_{i=1}^{n} \mathcal{C}_i\,(t_1, t_2, 0, \mathcal{H}_i) \leq \mathcal{C}_0 \\
&\forall (t_1, t_2) \in \mathcal{R} \subset R^2,
\end{aligned}$$

onde c_i, $i = 1, \ldots, n$ são custos associados à altura da chaminé.

1.3 Motivação

Apesar de nas últimas décadas serem apresentados vários algoritmos para a resolução de problemas de PSI, não existe muito *software* de domínio público.

O *Feasible Sequential Quadratic Programming* (FSQP) [53, 54, 144] é um algoritmo de optimização com algumas aplicações em engenharia de controlo e matemática. O algoritmo FSQP é baseado no conceito de programação quadrática sequencial admissível. O FSQP é composto por dois pacotes: FFSQP (Fortran) e CFSQP(C). A versão CFSQP [54] inclui um esquema especial para lidar com problemas semi-infinitos discretizados.

A função `fseminf` do Matlab está disponível na *Optimization Toolbox* [68]. O algoritmo `fseminf` consiste num algoritmo de programação quadrática sequencial quasi-Newton que usa uma direcção de procura aplicada a uma função mérito de um problema finito. Este problema finito resulta do problema de PSI em que as restrições infinitas são discretizadas nalguns pontos. A discretização do conjunto T considera uma grelha de pontos igualmente espaçados onde são avaliadas as restrições infinitas. A função `fseminf` não utiliza todos os pontos da grelha, identifica os picos nos dados e estima o máximo das restrições discretizadas através de uma interpolação quadrática ou cúbica. O algoritmo é classificado como um método de discretização (Secção 3.2). A função `fseminf` resolve problemas de PSI com dimensão máxima do espaço infinito de 2 ($p = 2$).

O *Nonlinear Semi-Infinite Programming Solver* (NSIPS) [112, 117, 122, 129] é um *software* de domínio público para problema de PSI. Na versão actual, o NSIPS inclui 4 *solvers*: um *solver* de discretização [116, 120, 123], um de técnica de penalidade [118, 119], um de programação quadrática sequencial (PQS) [121] e um de pontos interiores baseado numa técnica quasi-Newton [125]. O método de discretização usa o *software* comercial NPSOL [23] para resolver os sub-problemas finitos. Todos os métodos (com a excepção do método de discretização) resolvem problemas de PSI contendo apenas restrições infinitas de variáveis infinitas com dimensão 1 ($p = 1$) e nenhum pertence à classe de métodos de redução (Secção 3.3).

A grande aplicabilidade dos problemas de PSI juntamente com a carência de *software* motiva a proposta de um algoritmo capaz de resolver problemas de PSI na forma mais geral usando um algoritmo pertencente à classe de métodos de redução. Entre outras técnicas possíveis para a resolução dos sub-problemas reduzidos optou-se por uma técnica de penalidade.

1.4 Contribuição da tese

Neste projecto é proposto um algoritmo para problemas de PSI não lineares na forma mais geral. O algoritmo pertence à classe dos métodos de redução e é baseado numa técnica de penalidade. Numa fase preliminar fez-se um estudo a várias funções de penalidade e comparou-se o seu comportamento em problemas de PSI. Este estudo induziu a escolha de uma função de penalidade de norma-∞ aumentada e procedeu-se à sua generalização. As propriedades da função de penalidade generalizada são apresentadas neste trabalho bem como o seu estudo teórico.

Fez-se a implementação do algoritmo proposto produzindo um *solver* designado por SIRedAl *(Semi-Infinite Reduction Algorithm)*. A minimização da função de penalidade é efectuada pela função `fminsearch` do MATLAB ou pelo *solver* PSwarm[131]. A resolução dos problemas multi-locais é feita pelo MLOCPSOA (Secção 4.2) ou pelo MLOCGAMO (Secção 4.3). Mostram-se ainda os resultados obtidos com 117 problemas de PSI em que foram utilizados os dois algoritmos na resolução dos problemas multi-locais.

1.5 Estrutura da tese

As condições de optimalidade caracterizam as possíveis soluções de um problema de optimização. Com frequência essas condições são usadas no desenho dos respectivos algoritmos. No próximo capítulo são apresentadas as condições de optimalidade para problemas de PSI, permitindo introduzir notação e conceitos necessários aos capítulos subsequentes.

Uma recolha bibliográfica de trabalhos desenvolvidos em PSI, permite a apresentação no Capítulo 3 das abordagens clássicas mais usadas na resolução dos problemas de PSI.

O algoritmo desenvolvido pertence à classe dos métodos de redução. Estes métodos são caracterizados pela resolução de problemas multi-locais, correspondendo a uma parte significativa do cálculo computacional numa iteração num algoritmo de redução. A resolução de problemas multi-locais têm a dificuldade acrescida, relativamente ao métodos de optimização global, de determinar todos os óptimos globais (e eventualmente alguns

locais) de um problema. No Capítulo 4 são apresentadas as abordagens típicas para este problema, com referência a alguns trabalhos desenvolvidos por diversos autores.

O algoritmo proposto usa uma técnica de penalidade que é apresentada no Capítulo 5 em conjunto com as funções de penalidade mais conhecidas na programação finita e na PSI.

No Capítulo 6 apresenta-se o algoritmo proposto e algumas propriedades.

No Capítulo 7 são apresentados os resultados computacionais obtidos com uma implementação do algoritmo proposto. As experiências computacionais determinantes na escolha da função de penalidade, a análise de sensibilidade realizada aos parâmetros do algoritmo e os resultados para problemas na forma mais geral são também apresentados neste capítulo.

As conclusões e sugestões para trabalho futuro estão descritas no Capítulo 8.

Capítulo 2

Condições de optimalidade

As condições de optimalidade são importantes porque caracterizam as soluções do problema e respondem a questões como a da estabilidade das soluções e a dos tipos de convergência de algumas classes de métodos. A dedução das condições de optimalidade indica também possíveis caminhos para a resolução do problema apresentado.

Este capítulo é dedicado às condições de optimalidade de primeira e segunda ordem para problemas de PSI. Conforme referido anteriormente, um problema de PSI na forma mais geral pode incluir várias restrições infinitas e finitas. Ao longo deste capítulo consideram-se problemas com apenas uma restrição infinita. O uso de uma simplificação do problema advém do grau de complexidade de resolução não ser inferior ao dos problemas na forma mais geral permitindo ter uma notação mais simples das condições de optimalidade, sendo a generalização imediata. A representação do problema de PSI simplificado é a seguinte:

$$\begin{aligned} &\min_{x \in R^n} \ f(x) \\ s.a \ \ &g(x,t) \leq 0 \\ &\forall t \in T \ . \end{aligned} \tag{2.0.1}$$

A função objectivo $f(x)$ é de R^n em R e a função de restrição infinita, $g(x,t)$, é de $R^n \times R^p$ em R. Assume-se que ambas as funções são duas vezes continuamente diferenciáveis em todos os argumentos e $T \subset R^p$ é um conjunto compacto.

O capítulo é composto por três secções. Na primeira secção apresentam-se as condições de optimalidade de primeira ordem. Na segunda secção são introduzidas as condições para que um problema de PSI possa ser reduzido localmente a um problema finito e, com base neste problema finito, são apresentadas na última secção, as condições de optimalidade de segunda ordem.

2.1 Condições de optimalidade de primeira ordem

A seguinte definição caracteriza um ponto estacionário [88, 90, 112] do problema de PSI definido na forma (2.0.1).

Definição 2.1.1 *Seja $x^* \in R^n$ um ponto admissível, ou seja, que satisfaz a restrição:*

$$g(x^*, t) \leq 0, \ \forall t \in T,$$

e suponha que existem $t_1, t_2, \ldots, t_{m^*} \in T$ e números não negativos $\lambda_0^*, \lambda_1^*, \lambda_2^*, \ldots, \lambda_{m^*}^*$ tais que

$$\lambda_0^* \nabla_x f(x^*) + \sum_{i=1}^{m^*} \lambda_i^* \nabla_x g(x^*, t_i) = 0 \tag{2.1.1}$$

com

$$g(x^*, t_i) = 0, \quad i = 1, \ldots, m^*.$$

Então x^* é um ponto estacionário para o problema de PSI (2.0.1).

Hipótese 2.1.1 *Seja válida a seguinte condição de regularidade:*

$$\exists u \in R^n \ \ tal\ que \ \ g(x^*, t) + u^T \nabla_x g(x^*, t) < 0, \forall t \in T.$$

As condições de optimalidade de primeira ordem são dadas pela Definição 2.1.1 e pela Hipótese 2.1.1. Estas condições definem um sistema de equações não lineares, as chamadas

equações Karush-Kunh-Tucker (KKT). Um ponto solução desse sistema, ou que satisfaça a Definição 2.1.1 e a Hipótese 2.1.1, é chamado ponto KKT.

A caracterização da solução depende das diferentes condições de regularidade que são impostas ao sistema de equações KKT. Essas condições dependem também do tipo de restrições existentes no problema. Para um problema com restrições de igualdade, a condição de regularidade teria de ser diferente da Hipótese 2.1.1.

A dedução das condições de optimalidade de primeira ordem pode ser apresentada de várias formas. Por exemplo, [88] tem uma dedução baseada em integrais de Lesbegue enquanto que em [29] é baseada em integrais de Riemman.

Na verificação das condições de optimalidade de primeira ordem não é necessário considerar todo o conjunto T, podendo ser usado apenas um subconjunto finito de T. Existem vários métodos, na resolução de problemas de PSI, que substituem o conjunto T por um subconjunto finito de T em que as restrições infinitas são transformadas num conjunto finito de restrições.

Considere-se a seguinte hipótese:

Hipótese 2.1.2 *Para todo o $x \in R^n$ o conjunto dos maximizantes globais ($\Gamma(x)$) de $g(x,t)$ para todo o $t \in T$ é finito.*

As condições de optimalidade (2.1.1) juntamente com a Hipótese 2.1.2 e a não negatividade dos multiplicadores de Lagrange são equivalentes às condições KKT de primeira ordem de um problema de programação não linear (PNL) finito, da forma:

$$\min_{x \in R^n} f(x) \ \text{ s.a } \ g(x,t_i) \leq 0 \quad \forall\, i \in \{1,\ldots,m^*\}\,.$$

Sob determinadas hipóteses específicas, como a Hipótese 2.1.1, o problema PSI é localmente equivalente a um problema de PNL.

2.2 Redução local a um problema finito

Considere-se o problema definido em $\overline{x} \in R^n$

$$\max_{t \in T} g(\overline{x}, t), \tag{2.2.1}$$

em que $T \subset R^p$.

As condições de optimalidade de segunda ordem da PSI exigem que o sistema seja regular. Esta regularidade responde a questões como a estabilidade de soluções, a dependência de soluções do problema (2.2.1) assim como o tipo de convergência de algumas classes de métodos numéricos.

A redução local consiste na substituição das restrições infinitas do problema de PSI por um conjunto finito de restrições, que dependem exclusivamente de x, considerando os maximizantes globais e locais do problema (2.2.1).

Nesta secção começa-se por mostrar um resultado de estabilidade para uma única solução de (2.2.1) [37, 39], de seguida é apresentada a aproximação reduzida de forma a obter as condições de optimalidade de segunda ordem.

Seja $T \subset R^p$ um conjunto compacto definido como $T = \{t | h_j(t) \leq 0, \quad j \in J\}$, em que J é o conjunto de índices e h_j é uma função de $R^p \longrightarrow R$ duas vezes continuamente diferenciáveis.

Definição 2.2.1 *Seja $\overline{t}$ ponto admissível do problema* (2.2.1)*. O conjunto de índices activos define-se como*

$$\mathcal{J}(\overline{t}) = \{j \in J : h_j(\overline{t}) = 0\}.$$

Para um dado $\overline{t}$ do problema (2.2.1), a condição de regularidade (fraca) de *Mangansarian-Fromotivz* (CRMF) verifica-se se existir um vector $\overline{\eta} \in R^n$ tal que:

$$\nabla h_j(\overline{t})\overline{\eta} < 0, \quad j \in \mathcal{J}(\overline{t}).$$

Seja $\overline{t}$ uma solução do problema (2.2.1). Se a condição CRMF é satisfeita então existe um vector de multiplicadores $\overline{\gamma} \in R^{|\mathcal{J}(\overline{t})|}$ tal que:

$$\nabla_t \mathcal{L}^{\bar{t}}(\bar{t}, \overline{\gamma}) = 0, \quad \overline{\gamma} \geq 0 \tag{2.2.2}$$

em que $\mathcal{L}^{\bar{t}}$ representa a função Lagrangeana do problema (2.2.1) com respeito a $\bar{t}$,

$$\mathcal{L}^{\bar{t}}(t, \gamma) = g(\overline{x}, t) - \sum_{j \in \mathcal{J}(\bar{t})} \gamma_j h_j(t). \tag{2.2.3}$$

A condição de complementaridade estrita (CCE), *strict complementary slackness*, verifica-se se o multiplicador $\overline{\gamma}$ em (2.2.2) satisfaz

$$\overline{\gamma}_j > 0, \quad j \in \mathcal{J}(\bar{t}).$$

Introduz-se agora a condição de optimalidade de segunda ordem para o problema (2.2.1):

Hipótese 2.2.1 *Dado $\overline{x} \in R^n$ e seja $\bar{t}$ uma solução do problema* (2.2.1)*. Assume-se que a condição de regularidade de independência linear de $h_j(\bar{t})$ é verificada, isto é, os vectores $\nabla_t h_j(\bar{t})$, $j \in \mathcal{J}(\bar{t})$ são linearmente independentes. A condição de segunda ordem para que $\bar{t}$ seja maximizante local do problema* (2.2.1) *é:*
$\nabla^2_{tt} \mathcal{L}(\bar{t}, \gamma)$ ser definida negativa no espaço tangente, isto é,

$$\eta^T \nabla^2_{tt} \mathcal{L}(\bar{t}, \gamma) \eta < 0, \ \eta \neq 0 \in \mathcal{T}(\bar{x}, \bar{t}),$$

em que

$$\mathcal{T}(\overline{x}, \bar{t}) = \{\eta : \overline{\gamma}_j \nabla_t h_j(\bar{t}) \eta = 0, \ j \in \mathcal{J}(\bar{t})\}.$$

Teorema 2.2.1 *(Teorema 1 em [39]) Dado $\overline{x} \in R^n$ e seja $\bar{t}$ solução do problema* (2.2.1) *em que a Hipótese 2.2.1 é satisfeita. Então existem vizinhanças $U_{\overline{x}}$ de $\overline{x}$, e $V_{\bar{t}}$ de $\bar{t}$, e funções Lipschitz contínuas diferenciáveis direccionais*

$$t : U_{\overline{x}} \to V_{\bar{t}}, \gamma : U_{\overline{x}} \to R^{|\mathcal{J}(\bar{t})|}$$

tais que

- $t(\overline{x}) = \bar{t}$;

- $\gamma(\overline{x}) = \overline{\gamma}$
- *Para todo* $x \in U_{\overline{x}}$, $t(\overline{x})$ *é a única solução local para o problema* (2.2.1) *em* $V_{\overline{t}} \cap T$.

Seja $\overline{x}$ um ponto admissível do problema de PSI. Assumindo que a Hipótese 2.2.1 é satisfeita em todos os pontos $\overline{t}$ solução do problema (2.2.1) (*i.e.*, para todo $\overline{t} \in \Gamma(\overline{x}) = \{\overline{t}_1, \ldots, \overline{t}_r\}$), pela aplicação do Teorema 2.2.1 o problema de PSI pode ser localmente reduzido a um problema equivalente de optimização finita.

Teorema 2.2.2 *(Teorema 3 em [39]) Dado um* $\overline{x} \in R^n$ *tal que a Hipótese 2.2.1 é satisfeita em todas as soluções* $\overline{t} \in \Gamma(\overline{x})$, *então existe uma vizinhança* $U(\overline{x})$ *de* $\overline{x}$ *tal que para todo o* $x \in U_{\overline{x}}$ *existe um ponto admissível* $x \in R^n$ *se e só se:*

$$G_l(x) := g(x, t_l(x)) \leq 0, \;\; l = 1, \ldots, r$$

O problema reduzido é, então, definido da seguinte forma:

$$P_{red}(\overline{x}) \begin{cases} min & f(x) \\ s.a & x \in U(\overline{x}) \\ & G_l(x) \leq 0, \\ & l = 1, \ldots, r. \end{cases}$$

2.3 Condições de optimalidade de segunda ordem

Considere-se um problema de PSI e assume-se que para um $\overline{x} \in R^n$ as condições de regularidade do Teorema 2.2.2 verificam-se. Então numa vizinhança $U(\overline{x})$ de $\overline{x}$ o problema PSI é equivalente a um problema finito $P_{red}(\overline{x})$. Seja K o *cone das direcções críticas* do problema $P_{red}(\overline{x})$ em $\overline{x}$,

$$K = \left\{ \xi \in R^n : \begin{array}{l} \nabla_x f(\overline{x})\xi \leq 0, \\ \nabla_x G_l(\overline{x})\xi \leq 0, \end{array} \quad l = 1, \ldots, r \right\}.$$

As seguintes condições de optimalidade verificam-se.

Teorema 2.3.1 *([39]) Seja $\overline{x}$ um ponto admissível do PSI tal que as hipóteses do Teorema 2.2.2 são satisfeitas.*

(a) (Condição necessária) Seja $\overline{x}$ uma solução local do problema de PSI. Para qualquer $\xi \in K$ o seguinte sistema não tem solução em $d \in R^n$:

$$\begin{cases} \nabla_x g(\overline{x}, \bar{t}_l)d + \xi^\top \nabla^2_{xx} g(\overline{x}, \bar{t}_l)\xi - \nabla^\top_x t(\overline{x})\xi \nabla^2_{tt}\mathcal{L}(\bar{t}_l, \overline{\gamma}_l)\nabla_x t(\overline{x})\xi) < 0, \\ \bar{t}_l \in \Gamma(\overline{x}; \xi), \\ \nabla_x f(\overline{x})d + \xi^\top \nabla^2_{xx} f(\overline{x})\xi < 0, \\ se \;\; \nabla_x f(\overline{x})\xi = 0, \end{cases}$$

onde $\Gamma(\overline{x}, \xi) = \{\bar{t}_l \in \Gamma(\overline{x}) : \nabla_x g(\overline{x}, \bar{t}_l)\xi = 0\}$.

(b) (Condição suficiente) Seja $\overline{x}$ uma solução local do problema de PSI. Para qualquer $\xi \in K$ o seguinte sistema não tem solução.

$$\begin{cases} \nabla_x g(\overline{x}, \bar{t}_l)d + \xi^\top \nabla^2_{xx} g(\overline{x}, \bar{t}_l)\xi - \nabla^\top_x t(\overline{x})\xi \nabla^2_{tt}\mathcal{L}(\bar{t}_l, \overline{\gamma}_l)\nabla_x t(\overline{x})\xi) \leq 0, \\ \bar{t}_l \in \Gamma(\overline{x}), \\ \nabla_x f(\overline{x})d + \xi^\top \nabla^2_{xx} f(\overline{x})\xi \leq 0, \end{cases}$$

Capítulo 3

Abordagens clássicas

A forma intuitiva de resolver os problemas de PSI consiste na substituição do número infinito de restrições por um número finito, sem alterar o valor óptimo. Dos diferentes métodos numéricos existentes realçam-se três classes principais: métodos de discretização, métodos de trocas e métodos baseados em redução local. Neste capítulo faz-se uma abordagem aos principais métodos numéricos para a resolução de problemas de PSI.

Neste capítulo, por uma questão de notação, considera-se o problema de PSI na forma simplificada. O problema é constituído por uma única restrição infinita e tem a seguinte representação:

$$P(T) = \begin{cases} \min & \{f(x) : x \in K\} \\ K = & \{x \in R^n : g(x,t) \leq 0, \forall t \in T\} \end{cases} \tag{3.0.1}$$

em que T é um subconjunto compacto de R^p, f é uma função de $R^n \to R$ e g de $R^n \times R^p \to R$. Assume-se que ambas as funções são contínuas em todos os argumentos. As abordagens que vão ser apresentadas podem ser facilmente estendidas a problemas com mais do que uma restrição. Nesse caso, o conjunto de pontos admissíveis seria dado pela intersecção de vários conjuntos

$$K = \{x \in R^n : g_i(x,t) \leq 0, \text{ para } i = 1, \ldots, m \text{ e } \forall t \in T\},$$

em que, para cada $i = \{1, \ldots, m\}$, T seria um subconjunto compacto de R^p e g_i teriam as mesmas propriedades de g.

Os algoritmos usados em problemas de PSI geram, habitualmente, uma sequência de problemas auxiliares de optimização com restrições finitas que são resolvidos por algoritmos tradicionais de optimização finita. Dependendo da forma como os problemas (auxiliares) finitos são gerados, pode-se agrupá-los em três classes principais distintas: métodos das trocas, métodos de discretização e métodos baseados em redução local. Há ainda outros algoritmos que não se enquadram em nenhuma das classes referidas anteriormente. E, de acordo com a opinião de alguns autores poderia-se acrescentar ainda os algoritmos baseados em transcrições de restrições, os métodos tipo simplex, os métodos descendente e os métodos duais. Sendo que o algoritmo proposto neste trabalho pertence à classe dos métodos de redução.

Para descrever os métodos de trocas e discretização supõe-se que o conjunto K é compacto e considera-se os seguintes problemas auxiliares com restrições finitas:

$$P(T^k) = \begin{cases} \min & \{f(x) : x \in K^k\} \\ K^k = & \{x \in Z_0 : g(x,t) \leq 0, \forall t \in T^k\} \end{cases}$$

em que k representa o número da iteração e Z_0 é um conjunto compacto e convexo, $Z_0 \supset K$, que possibilita a adição de mais restrições. Por vezes Z_0 é introduzido artificialmente para tornar os problemas finitos solúveis. Se $K \neq \emptyset$, o problema aproximado $P(T^k)$ tem uma solução para todo o conjunto finito $T^k \subset T$ [37].

Este capítulo inclui quatro secções que descrevem as várias abordagens existentes para a resolução de problemas PSI. A Secção 3.1 aborda os métodos das trocas. Os métodos de discretização são abordados na Secção 3.2. Na Secção 3.3 são descritos os métodos baseados em redução local e na última secção são apresentados métodos que não se integram em nenhuma das classes anteriores.

3.1 Métodos de trocas

O método de trocas [37] é um método iterativo caracterizado pelo facto de, em cada iteração k, resolver um problema finito $P(T^k)$, em que o conjunto T^{k+1} é obtido a partir do conjunto T^k, adicionando e eventualmente removendo algum ponto, o que é equivalente à adição de pelo menos uma restrição e à possível remoção de outras.

Apresenta-se, agora, o algoritmo conceptual do método de trocas.

Algoritmo 3.1.1 *(Algoritmo conceptual do método de trocas [38])*

- *Passo(k): Dado $T^k \subset T$. Determinar uma solução x^k de $P(T^k)$ e alguns (ou todos) maximizantes $t_1^k, \ldots, t_{q^k}^k$ do sub-problema*

$$\max_{t \in T} \; g(x^k, t) \tag{3.1.1}$$

  Se $g(x^k, t_j^k) \leq 0$, $j = 1, \ldots, q^k$ parar. Caso contrário, escolher T^{k+1} verificando as seguintes condições:

$$T^{k+1} \subset T^k \bigcup \{t_1^k, \ldots, t_{q^k}^k\},$$
$$\max_{t \in T^{k+1}} \; g(x^k, t) > 0.$$

Uma condição necessária para existir convergência é:

$$\max_{j=1,\ldots,q^k} \; g(x^k, t_j^k) = \max_{t \in T} \; g(x^k, t),$$

isto é, em cada passo, ou no mínimo numa sequência de passos, é necessário determinar a solução global de um problema de optimização não-convexo (3.1.1), podendo ser bastante custoso para problemas cuja dimensão de T seja superior a dois. Os métodos das trocas têm sido aplicados quase exclusivamente à PSI linear. Estes métodos aplicados a este tipo de problemas normalmente geram soluções (aproximadas) com pouca precisão, mas são eficientes. A diversidade dos algoritmos baseia-se principalmente na escolha do conjunto T^{k+1}.

Em 1979, P.R. Gribik [27] propõe um algoritmo de planos de corte (*cutting-plane*) para problemas de PSI linear baseado num algoritmo de planos de corte central. O método gera uma sequência de pontos (admissíveis ou não) do problema. Num ponto não admissível o método pode gerar um corte a partir de uma restrição violada qualquer, sem afectar a convergência ou a taxa de convergência do algoritmo.

Em [6], os autores apresentam dois algoritmos de planos de corte: um na forma geral e outro, que designam por algoritmo relaxado, que difere do primeiro apenas na forma como gera o conjunto T^{k+1}. Este conjunto é dado por:

$$T^{k+1} = T^k \cup \{t^{k+1}\} \setminus Z^k, \text{ onde}$$

$$Z^k = \{t \in T^k : g(x^k, t) < 0\}.$$

K. Roleff, em [99], apresenta um algoritmo de trocas múltiplas estável para PSI linear. Um passo do algoritmo de trocas múltiplas corresponde a algumas etapas do método de simplex aplicado a um problema reduzido. Os problemas auxiliares do problema primal são obtidos depois da resolução do problema dual, em que sempre que um vector entra na base, são usados métodos de modificação estável para começar uma nova partição da matriz da base alterada.

R.L. Streit [104] apresenta um algoritmo para sistemas de equações lineares complexas com norma-∞. O objectivo é encontrar valores complexos de incógnitas de modo que a magnitude máxima residual do sistema seja minimizada. As incógnitas satisfazem determinadas restrições convexas, onde lhe são impostos limites sobre a sua magnitude. O problema original é substituído por um problema discretizado, que consiste num programa linear gerado de forma a que o erro relativo possa ser estimado. O problema primal é resolvido através do seu dual, usando um método de simplex revisto.

M.D. Asic et al., em [1], apresentam um método para PSI com restrições convexas. O método gera uma sequência de pontos admissíveis e não requer a maximização global da função restrição g. O método usa informação analítica da função restrição para obter uma discretização selectiva do conjunto T^k, em que os sucessivos refinamentos da grelha são usados para testar a admissibilidade dos pontos.

R. Reemtsen, em [94], apresenta um método de planos de corte para resolver problemas minimax num plano complexo. O autor transforma os problemas minimax complexos em problemas PSI real convexa e mostra como os métodos de discretização e planos de corte podem ser aplicados neste tipo de problemas. O algoritmo apresentado em [83] está relacionado com o de [94]. Enquanto que em [94] o método requer a restrição mais violada em cada iteração de um subconjunto finito de restrições infinitas, o de [83] usa a restrição mais violada na globalidade.

Em [51] os autores apresentam um algoritmo de optimização global para PSI convexa não linear baseado num algoritmo de planos de corte central para PSI linear. O método é de planos de corte de pontos interiores e preserva algumas características dos algoritmos de planos de corte central. Para a adição das restrições é usada informação dos gradientes da função objectivo e das restrições. O algoritmo tem um esquema de gestão de grelha para gerar cortes.

Em 1993, B. Fischer e J. Modersitzki, em [19], descrevem um método para determinar a melhor aproximação Chebyshev linear para funções definidas no plano complexo. O algoritmo é baseado numa reformulação do problema aproximado como um problema de programação semi-infinita linear, em que os módulos complexos são substituídos por restrições infinitas. O algoritmo descrito utiliza a ligação entre a formulação do problema primal e dual. Esta abordagem permite obter problemas de pequena dimensão e não depende do número de pontos extremos.

R. Reemtsen, em [95], apresenta um algoritmo para problemas gerais de optimização semi-infinita convexa, onde em cada iteração são resolvidos sub-problemas quadráticos.

S.C-Fang et al, em [17], propõem um algoritmo para programação semi-infinita quadrática convexa, em que usam uma perturbação entrópica. Os autores adicionam à função objectivo a função entrópica

$$h(x) = x \cdot lnx, \text{definida em } \{x \in R^n : x > 0\}.$$

Em cada iteração, a restrição adicionada corresponde ao máximo global do problema (3.1.1). Para determinar o máximo global, começam por dividir o intervalo $[0, 1]$ em 100

intervalos e usam uma sub-rotina IMSL [72] para encontrar o maximizante de cada intervalo. Não havendo garantia de que os maximizantes encontrados sejam globais, seleccionam como maximizante global aproximado aquele com maior valor da função objectivo.

A. Potchinkov et al., em [86], apresentam um método considerado como um desenvolvimento do método de planos de corte de Kelley-Cheney-Goldstein para programação convexa finita. A aproximação foi aplicada a um projecto de desenho de filtros digitais (filtros FIR) no plano complexo.

Em [44], os autores desenvolveram um algoritmo de planos de corte para problemas de PSI convexa. Os cortes gerados próximos da região admissível são cortes lineares. O algoritmo foi implementado nas versões sequencial e paralela.

H. Hu em [41], apresenta um método de convergência global para PSI linear baseado no método do sub-gradiente (descida máxima). O autor define o problema da seguinte forma:

$$\max_{x \in R^n} \; c^T x$$
$$\text{s.a} \;\; a(t)^T x - b(t) \leq 0 \;\forall t \in T.$$

em que $c \in R^n$, $T \subset R^p$, $a : T \rightarrow R^n$ e $b : T \rightarrow R$. Para um dado $\epsilon^k > 0$, $\lambda^k > 0$ e x^k, o algoritmo determina a ϵ^k-solução do problema não linear

$$sup\{a(t)^T(x^k + \lambda^k c) - b(t) : t \in T\},$$

ou seja, determina um t^k que satisfaz

$$a^T(t^k)(x^k + \lambda^k c) - b(t^k) \geq sup\{a(t)^T(x^k + \lambda^k c) - b(t) : t \in T\} - \epsilon^k.$$

Se $x^k + \lambda^k c$ satisfaz a restrição então está perto da região admissível e faz $x^{k+1} = x^k + \lambda^k c$. Caso contrário, calcula x^{k+1} como a projecção de $x^k + \lambda^k c$ em $a(t^k)^T x = b(t^k)$.

A.W. Potchinkov, em [82], propõe um método para desenho de diversos filtros FIR de fase-linear óptima. O método resolve problemas de PSI convexa. No código do programa são utilizadas rotinas da IMSL MATH/Library [72] para resolver os problemas de optimização (em particular, a IMSL-DUVMIF).

Y.V. Volkov et al., em [133], apresentam um método de aproximações exteriores estocásticas que incorpora mecanismos de pesquisa activa para as restrições relevantes e

remoção das restrições irrelevantes. O método proposto é um desenvolvimento do método de Eaves-Zangwill. Em cada iteração são aplicadas técnicas multi-partidas na pesquisa de parâmetros das restrições relevantes.

M.-H. Wang and Y.-E. Kuo apresentam em [136] um algoritmo para resolver problemas de PSI linear, combinando o método de adição de restrições para a PSI linear com o método de perturbação, em que é utilizada uma função barreira na resolução de problemas de programação linear finita. A restrição que é adicionada em cada iteração corresponde à solução do problema (3.1.1).

S.-Y. Wu et al. em [142] propõem um método de planos de corte relaxado para problemas de PSI convexos definidos da seguinte forma:

$$\begin{aligned} &\min_{x \in R^n} f(x) \\ \text{s.a } &a^T(t)x \geq b(t) \quad \forall t \in T \\ &x \geq 0 \end{aligned}$$

em que $f(\cdot)$ é convexa, $a(t)$ e $b(t)$ são funções contínuas em T. O esquema relaxado vai gerando novos cortes, seleccionando, em cada iteração, um ponto que verifique a condição:

$$a^T(t^{k+1})x^k - b(t^{k+1}) < -\delta,$$

sendo δ um valor pequeno.

B. Bretò [5] propõe um algoritmo para problemas de PSI linear baseado no plano de corte central com um procedimento para acelerar a convergência. O método tenta o corte mais profundo até que os sub-problemas se tornem não admissíveis ou um ponto admissível seja encontrado. O autor também mostrou que este algoritmo pode ser aplicado a problemas de PSI convexa.

Em [140] o autor mostra que um problema de cónicas robusto pode ser representado por um problema de PSI linear e propõe um ajuste a um algoritmo de trocas, tornando-o uma generalização do algoritmo tipo cascata para problemas de cónicas robusto.

3.2 Métodos de discretização

O método de discretização [37] é um método iterativo que calcula a solução do problema (3.0.1) através da resolução de uma sequência de problemas $P(T^k)$, onde T^k é uma grelha-h^k em T, isto é, um subconjunto finito $T^k \subset T$ tal que $\sup_{t\in T} dist(t, T^k) \leq h^k$, em que $dist$ é a distância de um ponto a um conjunto. Num k-ésimo passo, a grelha T^k (habitualmente uniforme) obtém-se por $h^k = \gamma^k h^{k-1}$, com $\gamma^k \in (0, 1)$. Em alguns algoritmos verifica-se a seguinte igualdade ($\kappa^k \in N$):

$$\gamma^k = \frac{1}{\kappa^{k-1}} \text{ com } \kappa^{k-1} \geq 2,$$

implicando $T^{k-1} \subset T^k, \forall k$.

O algoritmo conceptual do método da discretização pode ser descrito da seguinte forma.

Algoritmo 3.2.1 *(Algoritmo conceptual do método de discretização [37])*
Passo (k)*: Dados* h^k*, o último conjunto* $\overline{T}^{k-1} \subset T^{k-1}$ *e uma solução* x^{k-1} *do problema* $P(\overline{T}^{k-1})$*;*

i1) *Escolher* $h^k = \gamma^k h^{k-1}$ *e gerar* T^k*;*

i2) *Seleccionar* $\overline{T}^k \subset T^k$*;*

i3) *Calcular a solução* $\bar{x}$ *de* $P(\overline{T}^k)$*.*

> **Se** $\bar{x}$ *é admissível em* $P(T^k)$ *para uma dada precisão, então* $x^k := \bar{x}$*.*
> *Se a sequência* $\{\gamma^k\}$ *não estiver pré-definida então definir* γ^{k+1}*.*
> *Continuar com o passo* $(k + 1)$*.*
>
> **Caso contrário** *repetir o passo* $i2)$ *para escolher um novo conjunto* $\overline{T}^k$*.*

Uma vantagem dos métodos de discretização relativa a outros métodos de PSI é que estes métodos trabalham exclusivamente com subconjuntos finitos de T e não abordam os problemas multi-locais. Em particular, a admissibilidade de um ponto $x \in R^n$ pode ser verificada nos problemas finitos $P(T^k)$, excepto para o próprio problema de PSI $P(T)$

[96]. Uma desvantagem destes métodos é o aumento drástico do número de restrições do problema de PSI discretizado provocado pelo aumento de precisão. Uma forma de tornar o algoritmo mais eficiente é usar a máxima informação possível das grelhas anteriores na resolução de $P(T^k)$ [37, 38]. A principal diferença entre este tipo de algoritmos está na escolha de $\overline{T}^k$ que deve ser tal que a solução $P(\overline{T}^k)$ resolva aproximadamente também $P(T^k)$. Uma forma de seleccionar $\overline{T}^k$ é a seguinte:

$$\overline{T}^k \supset T^k_\alpha = \{t \in T^k : g(\bar{x}, t) \geq -\alpha\},$$

sendo $\alpha > 0$ algum valor escolhido e $\bar{x}$ o iterado anterior. A escolha de α é de extrema importância porque para α's muito grandes leva-nos a muitas restrições em $P(\overline{T}^k)$ e α's muito pequenos leva-nos a um grande número de passos $i2)$ e $i3)$ para um dado T^k fixo.

Em [35] o autor apresenta um método de discretização para problemas de PSI linear. A grelha inicial é refinada sucessivamente de forma a que as soluções das grelhas precedentes possam ser usadas como pontos iniciais das grelhas subsequentes e reduzam consideravelmente o número de restrições nos problemas seguintes. Para resolver os sub-problemas de programação linear é utilizado um algoritmo simplex numericamente estável.

R. Hettich e G. Gramlich em [36] descrevem um algoritmo implementado em FORTRAN para problemas semi-infinitos quadráticos convexos. O algoritmo apresentado é uma extensão do trabalho em [35].

Em [93] é proposto um algoritmo para problemas lineares que requer a solução de um problema de programação quadrática em cada iteração.

Em [95] o autor prova a convergência de um algoritmo de discretização aplicado a problemas gerais de PSI convexa.

Em [108] o autor utiliza uma estratégia de selecção das restrições $\epsilon-$activas evitando o aumento das restrições ao longo do método (das iterações).

3.3 Métodos baseados na redução local

Os métodos de redução [37] substituem as várias restrições infinitas do problema original por restrições finitas que, localmente e sob determinadas hipóteses, são suficientes para descrever K. Assume-se que $f \in C^2(R^n)$ e $g \in C^2(R^n \times T)$. Seja $\bar{x} \in R^n$ um ponto dado. Seja $\bar{t}_1, \ldots, \bar{t}_{q(\bar{x})}$, com $q(\bar{x}) < \infty$ todas as soluções locais de:

$$\max_{t \in T} g(\bar{x}, t) \tag{3.3.1}$$

para um dado $\bar{x}$. O ponto $\bar{x} \in K$ se e só se $g(\bar{x}, t_j) \leq 0, j = 1, \ldots, q(\bar{x})$. Assumindo que existe uma vizinhança $\overline{U}$ de $\bar{x}$ tal que existem funções t_j, $j = 1, \ldots, q(\bar{x})$ duas vezes continuamente diferenciáveis, em que:

$$t_j : \overline{U} \mapsto T, t_j(\bar{x}) = \bar{t}_j, j = 1, \ldots, q(\bar{x})$$

para todo o ponto $x \in \overline{U}$, os $t_j(\bar{x})$ são todos soluções locais de (3.3.1). Então, com

$$G_j(x) := g(x, t_j(x)), j = 1, \ldots, q(\bar{x}),$$

tem-se $G_j \in C^2(\overline{U})$ e

$$K \cap \overline{U} = \{x \in \overline{U} : G_j(x) \leq 0, j = 1, \ldots, q(\bar{x})\}$$

isto é, em $\overline{U}$ o problema $P(T)$ foi substituído pelo problema finito

$$P_{\bar{x}}(T) \;\; min\{f(x) : G_j(x) \leq 0, j = 1, \ldots, q(\bar{x})\}.$$

Os métodos de redução local são caracterizados pela necessidade de calcular soluções globais dos problemas auxiliares e geralmente são descritos da seguinte forma:

Algoritmo 3.3.1 *(método de redução conceptual [37])*

- *Passo k: Para um dado x^{k-1} (não necessariamente admissível).*

  1. *Determinar todos os maximizantes locais $t_1^{k-1}, \ldots, t_{q^{k-1}}^{k-1}$ de* (3.3.1)

2. *Usando alguns passos de um algoritmo finito, resolver o problema finito reduzido*

$$(P_{red}(x^{k-1})) \quad min\{f(x)|G_j(x) \leq 0, j = 1, \ldots, q^{k-1}\} \tag{3.3.2}$$

com $G_j(x) = g(x, t_j(x))$ *e* $t_j(\cdot)$ *definido numa vizinhança de* x^{k-1}.

Seja x^{k-1} *a solução do problema* (3.3.2).

3. *Ir para passo* $(k + 1)$.

No primeiro sub-passo está implícito que o número de maximizantes dos problemas (3.3.1) é finito. Se não for o caso, a hipótese básica para a redução falha e deve ser utilizado outro método. Este sub-passo é dispendioso porque requer uma pesquisa multi-local dos máximos em (3.3.1). Para o segundo sub-passo é conveniente usar métodos de Programação Não Linear com uma taxa de convergência super-linear para garantir um número baixo de execuções do primeiro sub-passo. Inicialmente os métodos de programação quadrática sequencial (SQP) eram usados quase exclusivamente neste contexto. As vantagens dos métodos de redução são a boa taxa de convergência (com boa precisão) e, normalmente, o problema reduzido finito possui um número pequeno de restrições.

E.J. Anderson e A.S. Lewis [2] apresentam um algoritmo de redução para resolver problemas de PSI linear. Os autores usam uma estratégia baseada no método de Simplex e uma estratégia de purificação. Neste trabalho não é feita referência ao método para determinar os óptimos locais (e globais) do problema (3.3.1).

Em 1985, I.D. Coope e G.A. Watson [10] propuseram um algoritmo para PSI baseado na Lagrangeana projectada. A função Lagrangeana para problemas de PSI reduzidos, com apenas uma restrição infinita, é definida da seguinte forma:

$$L(x, \lambda) = f(x) + \sum_{j=1}^{\bar{q}} \lambda_j g(x, t_j),$$

onde λ_j são multiplicadores de Lagrange não negativos e $g(x, t_j)$, para $j = 1, \ldots, \bar{q}$, são os máximos locais e globais de $g(x, t)$, para um dado x. Para a obtenção dos maximizantes do problema (3.3.1) (procura multi-local), é usada uma grelha uniforme do conjunto T e

os máximos são determinados com aplicação de um método do tipo Newton. A direcção de descida d é obtida através da resolução do problema quadrático

$$\begin{aligned} &\min_d \ d^T \nabla f(x) + \frac{1}{2} d^T H d \\ &s.a \ \ \nabla g(x, t_j) d + g(x, t_j) = 0, \ j = 1, \ldots, \bar{q}, \end{aligned} \tag{3.3.3}$$

em que H é uma matriz simétrica. A função mérito utilizada é a seguinte função de penalidade exacta:

$$P(x) = f(x) + \mu \sum_{j=1}^{\bar{q}} [g(x, t_j)]_+, \tag{3.3.4}$$

em que $[z]_+ = \max\{0, z\}$. Para $\mu > ||\lambda||_\infty$ a direcção de descida de (3.3.3) é direcção de descida da função (3.3.4). Quando $d = 0$ é a solução do problema (3.3.3), x é um ponto estacionário do problema de PSI. Para manter a matriz Hessiana do problema (3.3.3) definida positiva, usa a relação $H = \nabla^2 L(x, \lambda) + \mu I$, em que μ é o mesmo de (3.3.4) e I a matriz identidade. A diminuição do valor da função de penalidade P é garantida através do cálculo do comprimento de passo γ para a direcção d. γ é o maior membro da sequência $\{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$ que satisfaz a condição $T(x, \gamma) \geq \rho$, em que

$$T(x, \gamma) = \frac{P(x + \gamma d) - P(x)}{\gamma P'(x; d)},$$

$P'(x; d) < 0$ é a derivada direccional da função de penalidade (3.3.4) em x na direcção d e ρ é um valor pequeno.

O algoritmo de redução apresentado por C.J. Price at al., em [88, 89, 90], usa apenas derivadas de primeira ordem. Os maximizantes globais (e alguns locais) do problema (3.3.1) são determinados através da procura numa grelha, em que nos refinamentos é usado um método quasi-Newton. Na resolução do problema reduzido são usadas técnicas de programação quadrática sequencial juntamente com a função de penalidade exacta de norma infinita:

$$\phi_P(x, \mu, \nu) = f(x) + \mu \theta_\infty(x) + \frac{1}{2} \nu \theta_\infty^2(x), \text{onde } \theta = \max_{t \in T} [g(x, t)]_+$$

sendo $\mu > 0$ e $\nu \geq 0$ os parâmetros de penalidade.

A direcção de procura d é calculada com base numa aproximação quadrática ψ definida por:

$$\min_{d\in R^n} \psi(x_0, A_0; \mu, \nu; d) = f(x_0) + d^T \nabla f(x_0) + \frac{1}{2} d^T H d + \mu\zeta(d) + \frac{1}{2}\nu\zeta^2(d)$$

$$\text{onde } \zeta(d) = \max_{t\in A_0} [g(x_0, t) + d^T \nabla_x g(x_0, t)]_+ \text{ e } A_0 \subset T,$$

sendo A_0 um conjunto finito. Como o problema de minimização ψ pode ser reescrito na forma:

$$\begin{aligned} \min_{d\in R^n, \zeta\in R} \; & d^T \nabla f + \frac{1}{2} d^T H d + \mu\zeta + \frac{1}{2}\nu\zeta^2 \\ s.a \;\; & g(x^k, t) + d^T \nabla_x g(x^k, t) - \zeta \leq 0 \\ & \zeta \geq 0 \\ & \forall t \in A_0, \end{aligned}$$

os multiplicadores de Lagrange deste problema são usados como estimativas de multiplicadores de Lagrange óptimos e na actualização de H, μ e ν.

A nova aproximação é obtida através de uma procura unidimensional baseada no critério de Armijo e no arco definido por

$$x_{k+1} = x_k + \alpha d + \alpha^2 c.$$

O vector c é um corrector para evitar o efeito de Maratos e consequentemente garantir a convergência superlinear.

O algoritmo de optimização multi-local proposto em [55] usa uma estratégia passiva numa partição de T. O método baseia-se no cálculo da derivada de g para um número de pontos, igualmente espaçados, de um subconjunto T. Os intervalos que não contêm minimizantes locais são excluídos (descartados), enquanto que nos restantes é usada uma linha unidimensional não exacta.

Em [56] os autores propuseram dois algoritmos para PSI linear: um combina uma estratégia de pivotagem tipo simplex com um esquema de direcção admissível; o outro executa um método de direcção admissível juntamente com um procedimento de purificação. O

algoritmo de purificação permite obter um ponto extremo a partir de um ponto admissível, sem piorar o valor da função objectivo. Este algoritmo é uma extensão do algoritmo apresentado em [57].

Em [143] é apresentado um método iterativo que resolve um sistema KKT aplicado a problemas PSI. Em cada iteração, k, é resolvido um sistema de programação não linear (PNL) com restrições finitas e é determinado um novo t^k. Na resolução do PNL é usado um método de Newton semi-suave.

A.I.P.N. Pereira e E.M.G.P. Fernandes em [75, 76, 77] propõem um algoritmo de redução local para resolver problemas de PSI. Na resolução dos problemas multi-locais utilizam uma variante do algoritmo estocástico arrefecimento simulado (*simulated annealing*) que usa uma estratégia de redefinição dos parâmetros de controlo e uma técnica que conserva a admissibilidade. A este algoritmo foi-lhe incorporado a técnica de *stretching* evitando desta forma a convergência para soluções já detectadas. Na optimização do problema reduzido é utilizada uma função de penalidade exponencial e uma técnica quasi-Newton.

3.4 Outros métodos

C.-J. Lin et al, em [59], usam um método de transcrição para resolver um problema de PSI linear. O problema da forma:

$$\begin{aligned} &max \;\; c^T x \\ &s.a \quad a(t)^T x \leq b(t), \quad \forall t \in T \end{aligned}$$

é resolvido através de um problema de programação convexa sem restrições:

$$\begin{aligned} &max \;\; c^T x - \mu \int_T e^{\frac{a(t)^T x - b(t)}{\mu}} d\lambda(t) \\ &s.a \;\; x \in R^n \end{aligned}$$

em que $\lambda(t)$ representa uma medida de Lesbegue. Na integração numérica é utilizado o método de Simpson.

Em [106], K. Suyama et al. representam um problema de aproximação Chebyshev complexa através de um problema de PSI linear usando uma técnica baseada no método de Simplex. O algoritmo é composto por três fases. Na primeira, T é dividido num conjunto de pontos, o problema semi-infinito transforma-se num finito e são seleccionados candidatos a óptimos. Nesta fase utilizam um método convencional de simplex. A segunda fase consiste em eliminar as restrições redundantes. Na última fase é seleccionada a solução óptima, com a verificação das condições KKT.

Em [61], os autores resolvem problemas de PSI quadrática, com uma restrição apenas, através do respectivo problema finito dual parametrizado. O algoritmo proposto determina a solução global do problema dual e usa um esquema para o refinamento na parametrização de T juntamente com um procura local. O algoritmo apresentado em [63] é uma extensão do algoritmo de [61]. O método proposto resolve problemas de programação estritamente convexa com mais do que uma restrição infinita.

Os autores em [107] reformulam o problema PSI (3.0.1) como um problema de optimização estocástico descrito da seguinte forma:

$$\begin{aligned} &\min f(x) \\ &s.a \quad \int h(g(x,t))p(t)dt \end{aligned}$$

onde p é uma função contínua e estritamente positiva e h define-se como:

$$\begin{aligned} h(y) = 0, \quad &\forall t \in (-\infty, 0] \text{ e} \\ h(y) > 0, \quad &\forall t \in (0, \infty) \end{aligned}$$

Os autores apresentam algoritmos aleatórios (baseados em aproximações estocásticas) para resolver este tipo de problema.

A.I.F. Vaz et al., em [124], usam um esquema de transcrição de restrições para a PSI (o problema de PSI é transformado num problema finito) em que utilizam uma técnica de penalidade clássica baseada na função exponencial. O problema (3.0.1) é transcrito da

seguinte forma:

$$\begin{aligned} &\min_{x \in R^n} \ f(x) \\ &s.a \ \ G_\varepsilon(x) \equiv \int_T g_\varepsilon(x,t)dt \leq \tau \end{aligned} \tag{3.4.1}$$

com

$$g_\varepsilon(x,t) = \begin{cases} 0 & se \quad g(x,t) < -\varepsilon \\ \frac{(g(x,t)+\varepsilon)^2}{4\varepsilon} & se \quad -\varepsilon \leq g(x,t) \leq \varepsilon \\ g(x,t) & se \quad g(x,t) > \varepsilon, \end{cases} \tag{3.4.2}$$

em que ε é factor de suavização e τ o de relaxamento. A função de penalidade exponencial proposta para o problema (3.4.1), considerando apenas uma restrição de desigualdade, é uma vez continuamente diferenciável, depende também dos multiplicadores de Lagrange e é definida da seguinte forma:

$$\phi(x,\lambda,\mu) = f(x) + \frac{1}{\mu}\lambda\left(e^{\mu\left(\int_T g_\varepsilon(x,t)dt - \tau\right)} - 1\right).$$

O λ é o multiplicador de Lagrange e $\mu > 0$ o parâmetro de penalidade. No algoritmo, a minimização do problema:

$$\min_{x \in R^n} \phi(x,\lambda,\mu)$$

para um dado valor de μ, é feita usando um algoritmo quasi-Newton baseado na fórmula de BFGS para calcular uma aproximação à inversa da Hessiana da função de penalidade. Para um dado valor de λ, ε e τ a solução determinada $x^*(\mu)$ é dada como aproximação da solução do problema (3.4.2). Se a evolução é significativa nas duas soluções consecutivas do problema (3.4.2) então o algoritmo actualiza λ, ε e τ, e continua com uma nova iteração, caso contrário pára.

Em [126], os autores propõem um algoritmo que usa uma aproximação quadrática sequencial para problemas de PSI não linear. O algoritmo começa por construir um problema de PSI quadrático (PSIQ) a partir do problema inicial e depois usa o método proposto por Liu et al. [62] para resolver o problema de PSIQ. O problema de PSI quadrático é resolvido através de uma parametrização dual.

Capítulo 4

Optimização multi-local

Uma parte significativa do cálculo computacional de uma iteração num algoritmo de PSI, que utiliza um método de redução, é dedicada à determinação de todos os maximizantes globais (e locais) das funções de restrição infinita. A determinação de todos os óptimos globais e locais caracteriza a optimização multi-local, a qual poderá ser entendida como uma extensão da optimização global. Neste capítulo é feita uma breve descrição das técnicas propostas na literatura, bem como as duas abordagens usadas na implementação do algoritmo de PSI proposto.

O problema de optimização multi-local consiste na determinação de todos os óptimos globais e locais do problema seguinte, para um dado valor fixo $\bar{x} \in R^n$,

$$\max_{t \in T} g(\bar{x}, t) \equiv \bar{g}(t) \tag{4.0.1}$$

onde o conjunto T é compacto, g supõe-se uma função não linear nas variáveis t, continuamente diferenciável. A optimização multi-local pode ser considerada uma extensão da optimização global. Enquanto que na optimização global pretende-se determinar apenas um maximizante $t^* \in T$ que verifique a condição:

$$\bar{g}(t^*) \geq \bar{g}(t), \quad \forall t \in T$$

na optimização multi-local são requeridos todos os óptimos globais e locais.

Nos problemas de PSI, a optimização multi-local pode surgir na verificação da admissibilidade de um ponto ou na utilização de um método de redução na resolução do problema de PSI. A verificação da admissibilidade de um ponto requer a resolução de um problema do tipo (4.0.1), para cada restrição infinita. Recorde-se que, na função de restrição infinita, um ponto $\bar{x}$ é admissível se se verifica a desigualdade $g(\bar{x}, t) \leq 0$ para todo t no conjunto T. Esta verificação pode-se fazer determinando todos os óptimos globais do problema (4.0.1). Como foi visto anteriormente, os métodos de redução substituem as restrições infinitas por restrições finitas nos pontos óptimos do problema (4.0.1). Esta determinação dos óptimos locais e globais representa uma parte significativa do cálculo computacional de uma iteração no algoritmo de redução para PSI.

Este capítulo está dividido em três secções. Na primeira secção são explanadas as técnicas tradicionais usadas na optimização multi-local, com referência a alguns trabalhos desenvolvidos nesta área. A segunda secção descreve o algoritmo do *solver* MLOCPSOA e a última secção a do *solver* MLOCGAMO. Estes *solvers* foram implementados com o intuito de serem aplicados nos métodos de redução para PSI e são usados na implementação do algoritmo descrito no capítulo 6.

4.1 Métodos para a optimização multi-local

Os métodos de optimização global determinam apenas um óptimo global e a repetição sucessiva destes métodos para detectar outros óptimos poderá não ser suficiente em alguns problemas multi-modais. Com o objectivo de ultrapassar esta dificuldade têm surgido algoritmos que combinam métodos de optimização global com outras técnicas para 'forçar' a convergência para todos os óptimos globais (e locais).

Os métodos de optimização global podem ser divididos em determinísticos [3, 4, 40, 79] e estocásticos [45]. Nos métodos determinísticos a garantia de que um óptimo global foi encontrado é dada depois de fazer uma procura exaustiva no conjunto de soluções,

enquanto que na maioria dos métodos estocásticos é possível provar que um óptimo global foi alcançado com probabilidade um, desde que se verifique um conjunto de condições. Embora estes últimos algoritmos sejam simples de implementar, os custos computacionais tendem a ser elevados. Nomeadamente, o tempo computacional de resolução poderá ser muito longo para satisfazer as condições que garantam o óptimo global. Habitualmente opta-se por condições menos restritas, mas sem a garantia de encontrar o óptimo global. Nos métodos determinísticos a informação dos pontos é usada numa sequência de passos, bem definida, gerada pelo algoritmo, enquanto que os métodos estocásticos modelam a função objectivo através de uma variável aleatória que usa a informação de pontos (bons) já avaliados e os respectivos valores da função. Tipicamente, estes métodos geram pontos de T baseados na informação de outros pontos gerados no passado.

Algumas técnicas existentes para a optimização multi-local são baseadas nos algoritmos estocásticos, nomeadamente algoritmos que combinam técnicas *multi-start* com as de *clustering*, arrefecimento simulado, colónia (também conhecido por enxame) de partículas e algoritmos evolutivos.

Algoritmos baseados nas técnicas *multi-start* juntamente com as técnicas *clustering* podem ser encontrados nos trabalhos [45, 46, 88, 110, 111]. Em [88], o algoritmo apresentado é designado por *Multi-Local Optimisation Subalgorithm* (MOS) e tem uma estrutura semelhante ao de [45, 46]. Inicialmente o algoritmo gera um conjunto de pontos, também chamados por pontos teste. Na geração destes pontos são usadas sequências de pontos pseudo-aleatórios de Halton [88]. Segue-se uma fase exploratória, onde é feito um estudo topográfico da função g, em que é calculado o valor de g para cada ponto teste. Usando esta informação os pontos teste são agrupados em *clusters*. Cada *cluster* é formado por pontos teste numa região de atracção para algum maximizante. A região de atracção é formada por pontos de T, com um caminho ascendente (num caminho ascendente t_0 está ligado a t_1 se o valor de $g(t_1)$ for superior ao de $g(t_0)$) de um ponto qualquer para um maximizante local. O critério de paragem é formado pelas avaliações à fiabilidade dos caminhos. A fiabilidade é determinada através de um processo estocástico que modela g ao longo de cada caminho. O processo estocástico é um processo de movimento *Brownian*

generalizado que permite formar uma estimativa de probabilidade relativa à monotonia estrita da função g ao longo de cada caminho.

Uma abordagem híbrida baseada no método de arrefecimento simulado (*simulated annealing*) para encontrar todos os minimizantes de uma função é apresentada em [101]. Nesta abordagem, o autor combina a técnica do arrefecimento simulado, com uma pesquisa tabú e um método descendente.

A.I.P.N. Pereira, em [75], apresenta um algoritmo designado por arrefecimento simulado *streched* (ASS). Este algoritmo usa uma variante do método de arrefecimento simulado ([42]) modificada e combinada com a técnica da função *stretching*, [73]. O algoritmo de arrefecimento simulado determina os pontos óptimos e a técnica da função *stretching* é usada, ao longo do processo iterativo, com o intuito de prevenir a convergência para maximizantes calculados nas iterações anteriores. O método ASS gera uma sucessão de problemas de optimização global formulados da seguinte forma:

$$\max_{t \in T} \Phi_k(t)$$

em que k indica a iteração e a função Φ é definida por:

$$\Phi_k(t) = \begin{cases} \bar{g}(t), & \text{se } k = 1 \\ w(t), & \text{se } k > 1. \end{cases}$$

A função $w(t)$ resulta da aplicação da técnica da função *stretching* numa vizinhança dos maximizantes que vão sendo encontrados ao longo das iterações, modificando a função objectivo da seguinte forma:

$$w(t) = \begin{cases} \bar{\bar{g}}(t) - \frac{\delta_2}{2} \frac{sign(\bar{g}(\bar{t}^i) - \bar{g}(t)) + 1}{tanh(\mu(\bar{\bar{g}}(\bar{t}^i) - \bar{\bar{g}}(t)))}, & \text{se } t \in V_\epsilon(\bar{t}^i) \\ & \text{para } i = 1, \ldots, \bar{N} \\ \bar{g}(t) & \text{caso contrário} \end{cases}$$

com

$$sign(z) = \begin{cases} -1, & se \quad z < 0 \\ 0, & se \quad z = 0, \\ 1, & se \quad z > 0 \end{cases}$$

e onde $\overline{t}^i$, para $i = 1, \ldots, \bar{N}$ representam os maximizantes (globais e locais) já encontrados pelo processo iterativo, $\bar{N}$ é o número de maximizantes já determinados, δ_2 e μ são constantes positivas e $V_\epsilon(\overline{t}^i)$ representa uma vizinhança em torno de $\overline{t}^i$ de raio ϵ. A função $\bar{\bar{g}}(t)$ corresponde à primeira transformação que é feita na função objectivo, provocando um decréscimo dos valores da função objectivo numa determinada vizinhança $\delta_1||t-\bar{t}||$ do ponto óptimo $\bar{t}$

$$\bar{\bar{g}}(t) = \bar{g}(t) - \frac{\delta_1}{2}||t-\bar{t}||(sign(\bar{g}(\bar{t}) - \bar{g}(t)) + 1),$$

em que δ_1, representa uma constante positiva.

Em [16, 47] é proposto um algoritmo de optimização de colónia de partículas (OCP). Este algoritmo é baseado numa população simples que é motivada a partir da simulação do comportamento social. Apesar deste algoritmo estar preparado para o cálculo de soluções globais é vulnerável a problemas em que a função objectivo é multi-modal, podendo oscilar entre as soluções. Na tentativa de encontrar múltiplas soluções locais do problema (4.0.1), Parsopoulos e Vrahatis em [74] propuseram uma modificação no algoritmo de OCP. Os autores introduziram no algoritmo a técnica da função *stretching* para isolar algumas partículas do enxame e gerar pequenas populações em torno destas. Nas populações mais pequenas é feita uma pesquisa mais refinada enquanto que no enxame 'grande' é feita uma pesquisa para as melhores soluções. Uma outra variação do algoritmo OCP tradicional, que localiza nichos através do crescimento de sub-enxames a partir de um enxame inicial foi proposto em [7]. É usado um algoritmo de optimização de colónia de partículas de convergência garantida para treinar as partículas dos sub-enxames. Mengs et al. em [69] combinam o algoritmo de colónias de partículas com estratégias de estimação de forma a que este seja capaz de identificar potenciais lideres durante a evolução e manter a diversidade destes no espaço de pesquisa. A.I.F. Vaz e E.M.G.P. Fernandes em [114] propuseram uma modificação ao algoritmo OCP introduzindo uma estratégia determinística que usa a informação do gradiente ou uma aproximação de uma direcção ascendente. Esta alteração previne que todas as partículas se concentrem na vizinhança de algum maximizante global e foi implementado no *solver* MLOCPSOA [113]. Na próxima secção é feita uma descrição

mais detalhada deste algoritmo.

Como algoritmo evolutivo tem-se um algoritmo genético para optimização multi-local proposto por L. Costa e A.I.F. Vaz. O algoritmo usa duas aproximações: a primeira é baseada numa estratégia multi-objectivo e a segunda usa uma estratégia uni-objectivo. Em ambas as aproximações é utilizada a informação do gradiente para melhorar a convergência local. Este algoritmo está implementado no *solver* MLOCGAMO [12] e é descrito com mais detalhe na última secção deste capítulo.

Para além dos algoritmos estocásticos têm sido propostos métodos que usam técnicas determinísticas na resolução de problemas de optimização multi-local. León et al. em [55] propuseram um algoritmo que combina uma técnica de partições com uma análise intervalar. Floudas em [21] combina o método *branch and bound* com a técnica de partições na região admissível. Em [48] os autores usam uma técnica que reduz o problema inicial não diferenciável a um problema de partições do conjunto óptimo.

4.2 MLOCPSOA

Nesta secção é descrito um algoritmo baseado em OCP. Este algoritmo foi proposto em [114] e está implementado no *solver* MLOCPSOA [113].

O *solver* MLOCPSOA está preparado para resolver problemas na forma:

$$\max_{t \in T} \ \bar{g}(t) \tag{4.2.1}$$

onde $\bar{g}(t)$ é uma função objectivo multi-modal e T é um conjunto compacto definido por $T = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \cdots \times [\alpha_p, \beta_p]$. Considera-se que o número de óptimos do problema (4.2.1) é finito e que a função $\bar{g}(t)$ é continuamente diferenciável.

O algoritmo de OCP simula o comportamento social de pássaros para calcular os óptimos globais do problema (4.2.1). Este algoritmo usa uma população (colónia ou enxame) de indivíduos (partículas). Para cada partícula i, no instante de tempo l (ou iteração), é atribuída uma posição $t^i(l) \in T$ e uma velocidade $v^i(l)$ que fornece informação sobre a trajectória da partícula. Também é mantida a informação sobre a melhor posição $y^i(l)$

visitada pela partícula $t^i(l)$. A velocidade no instante de tempo $l+1$ é actualizada da seguinte forma:

$$v_j^i(l+1) = \iota(l)v_j^i(l) + \mu\omega_{1j}(l)(y_j^i(l) - t_j^i(l)) + \nu\omega_{2j}(l)(\hat{y}_j(l) - t_j^i(l)), \quad j = 1, \ldots, p \quad (4.2.2)$$

onde $\iota(l)$ é o parâmetro inércia, μ é um parâmetro cognitivo, ν é um parâmetro social, $\omega_{1j}(l)$ e $\omega_{2j}(l)$ são números aleatórios gerados uniformemente a partir de $(0, 1)$ e $\hat{y}$ representa a melhor partícula até ao momento da colónia. Então, $y_j^i(l) - t_j^i(l)$ é a direcção cognitiva, *i.e.*, a direcção da partícula para a sua melhor posição e $\hat{y}_j(l) - t_j^i(l)$ é a direcção social, *i.e.*, a direcção para a melhor posição de sempre da colónia. A definição da nova posição da partícula faz-se através de:

$$t_j^i(l+1) = t_j^i(l) + v_j^i(l+1), \quad j = 1, \ldots, p. \quad (4.2.3)$$

Desta forma, uma partícula usa a sua melhor posição encontrada e toda a colónia direcciona-se para a solução óptima.

O algoritmo de optimização de colónia de partículas multi-local (OCPML) implementado no *solver* MLOCPSOA é capaz de localizar múltiplas soluções do problema (4.2.1). Para evitar que todas as partículas se concentrem em torno da melhor posição da colónia, a direcção da melhor posição de sempre da colónia, na equação da velocidade (4.2.2), foi substituída por uma direcção ascendente determinada na posição da melhor partícula de sempre. A diversidade é mantida porque a informação relacionada com um óptimo global não é propagada socialmente. A experiência individual de cada partícula juntamente com a informação da direcção ascendente leva-a a aproximar-se de um maximizante. O algoritmo OCPML difere do OCP original na forma como a equação de velocidade é actualizada, onde a direcção para a melhor colónia de sempre é substituído pela direcção de subida (gradiente). A nova equação de velocidade é definida então como:

$$v_j^i(l+1) = \iota(l)v_j^i(l) + \mu\omega_{1j}(l)(y_j^i(l) - t_j^i(l)) + \nu\omega_{2j}(l)(\nabla_j g(t^i(l))), \quad j = 1, \ldots, p \quad (4.2.4)$$

O último termo nesta fórmula pode ser visto como uma visão local da partícula. A partícula tem a 'capacidade' de seguir, com alguma probabilidade, a direcção de subida dada pelo

gradiente e melhorar a sua posição localmente. O algoritmo pode usar, em substituição da direcção do gradiente, um método heurístico para avaliar uma direcção aproximada de subida na melhor posição de cada partícula.

O algoritmo OCPML implementado no *solver* MLOCPSOA, é o seguinte:

Algoritmo 4.2.1 *MLOCPSOA*

1. *Dados prog e maxiter, o número máximo de progressos e o número máximo de iterações, respectivamente.*

2. *Seja s o tamanho da população.*

3. *Inicializar a colónia através de uma distribuição uniforme ($t_j^i(0) \sim U(\alpha_j, \beta_j), \quad i = 1, \ldots, s, \quad j = 1, \ldots, p$) ou por uma sequência Halton pseudo-aleatória de pontos. Inicializar a velocidade a zero e a melhor valor da função objectivo a $-\infty$.*

4. *Seja $pr = 0$ o número consecutivo de iterações sem progresso, $l = 0$ o número de iterações e $gbest = 0$ o índice da melhor posição de sempre de toda a colónia.*

5. *Enquanto $pr < prog$ e $l < maxiter$ fazer:*

   (a) *Seja pr=pr+1.*

   (b) *Calcular o parâmetro $\iota(l)$.*

   (c) *Para cada partícula ($i = 1, \ldots, s$) fazer:*

      i. *Calcular o valor da função objectivo ($g(t^i)$) e o respectivo gradiente ($\nabla g(t^i)$).*

      ii. *Se ($g(y^i) < g(t^i)$) então seja $pr = 0$ e $y^i = t^i$.*

      iii. *Se ($g(y^{gbest}) < g(y^i)$) então seja $gbest = i$.*

      iv. *Actualizar a velocidade da partícula ($v^i(l+1)$) usando a equação (4.2.4) e verificar os limites.*

      v. *Actualizar a partícula ($t^i(l+1)$) usando a equação (4.2.3).*

   (d) *Fazer $l = l + 1$.*

4.3 MLOCGAMO

O algoritmo no *solver* MLOCGAMO tem duas versões implementadas. A primeira baseia-se numa estratégia multi-objectivo enquanto que a segunda usa uma estratégia uni-objectivo. Ambas as versões usam uma direcção ascendente local para melhorar a convergência. O algoritmo de optimização foi desenvolvido para determinar todos os óptimos locais (incluindo os globais) de um problema de optimização não linear na forma (4.2.1), sob as mesmas condições anteriormente impostas.

O algoritmo do MLOCGAMO é um algoritmo genético com adaptações no sentido de determinar todos os maximizantes globais e alguns locais em problemas do tipo (4.2.1) e descreve-se da seguinte forma:

Algoritmo 4.3.1 *O algoritmo MLOCGAMO*

1. *Seja* $l = 0$ *o contador de iterações. Inicializar aleatoriamente* $\mathcal{E}^0$.

2. *Enquanto o número de iterações e número de avaliações da função objectivo for inferior a um determinado valor:*

   - *Calcular o conjunto de progenitores* $\mathcal{P}^l$ *a partir da população de elite* $\mathcal{E}^l$ *usando uma função fitness.*
   - *Calcular o conjunto dos Offsprings* $\mathcal{O}^l$ *a partir da população progenitora* $\mathcal{P}^l$ *usando os operadores de crossover e mutação.*
   - *Aplicar uma iteração do método quasi-Newton a todos os pontos no conjunto de elite* $\mathcal{E}^l$.
   - *Calcular o novo conjunto de elite* $\mathcal{E}^{l+1}$ *através da selecção de pontos do conjunto* $\mathcal{E}^l \cup \mathcal{O}^l$ *com o melhor valor fitness. Para os pontos que tenham o mesmo valor fitness é seleccionado aquele com o maior valor da função objectivo.*
   - $l = l + 1$.

Na próxima subsecção descreve-se a função *fitness* e nas restantes subsecções a selecção dos progenitores e dos operadores de *crossover* e mutação.

4.3.1 A função *fitness*

A maior diferença entre um algoritmo genético uni-objectivo regular e o algoritmo proposto pelos autores está na função *fitness* usada. Com o propósito de obter todos os óptimos locais a função *fitness* proposta tem o seguinte algoritmo:

Algoritmo 4.3.2 *A função fitness*

1. *Calcular a distância entre todos os pontos no conjunto* $\mathcal{U} = \mathcal{E} \cup \mathcal{O}$. *Se a distância entre dois pontos estiver dentro de uma determinada tolerância, o ponto com menor valor da função objectivo é removido. Seja* $\mathcal{R}$ *o conjunto de pontos removidos.*

2. *Seja* $\mathcal{P} = \mathcal{U} \setminus \mathcal{R}$ *o conjunto de pontos sem valor atribuído.*

3. *Se a versão considerada é a uni-objectivo então o valor fitness corresponde ao valor da função objectivo dos pontos do conjunto* $\mathcal{P}$.

4. *Caso contrário,considera-se a versão multi-objectivo (versão bi-objectivo)*

   (a) *Seja* $rank = 1$.

   (b) *Calcular o conjunto de pontos vizinhos não dominados* $\mathcal{ND}$.

   (c) *Atribuir rank a todos os pontos em* $\mathcal{ND}$.

   (d) *Seja* $\mathcal{P} = \mathcal{P} \setminus \mathcal{ND}$. *Fazer* $rank = rank + 1$ *e ir para* $4(b)$.

No caso da função uni-objectivo, o valor *fitness* é igual ao valor da função objectivo. A excepção consiste nos pontos vizinhos, pois o algoritmo aplica um procedimento local no conjunto de elite, $\mathcal{E}^k$.

Na versão multi-objectivo, o algoritmo considera dois objectivos a minimizar (bi-objectivo): o simétrico do valor da função objectivo e a norma do gradiente da Lagrangeana do problema (4.2.1). Para o posicionamento dos pontos é usado um conceito de dominância [13]. Sejam $\bar{g}_1$ e $\bar{g}_2$ as funções objectivo. Um ponto t' domina um ponto t'' ($t' \prec t''$):

$$\forall i \in \{1,2\} : \ \bar{g}_i(t') \leq \bar{g}_i(t'') \ \wedge \ \exists j \in \{1,2\} : \ \bar{g}_j(t') < \bar{g}_j(t''), \tag{4.3.1}$$

em que $\bar{g}_1$ é a função $-g$ e $\bar{g}_2$ representa a norma do gradiente da função Lagrangeana definida em (2.2.3). Se as condições em (4.3.1) não se verificam, diz-se que t' não domina t'' ($t' \not\prec t''$). A este conceito de dominância padrão foi acrescentado uma vizinhança. Um ponto t' só domina t'' se satisfaz a condições em (4.3.1) e está dentro de uma determinada distância euclidiana. O *ranking* é calculado nos passos $4(b) - 4(d)$, onde os *ranks* com menor valor vão para os pontos mais não dominados, *i.e.*, pontos t' tais que $t' \not\prec t, \forall t \in \mathcal{P}$. Removendo do conjunto $\mathcal{P}$ os pontos não dominados (o conjunto $\mathcal{ND}$), as iterações prosseguem determinando o próximo conjunto de pontos não dominados.

4.3.2 Selecção dos progenitores e os operadores *crossover* e mutação

A partir do conjunto de elite $\mathcal{E}$ são seleccionados vários pares aleatoriamente. Baseado no valor da função *fitness* são seleccionados dois progenitores, onde lhes é aplicado o operador *Simulated Binary Crossover* (SBX) [14] gerando dois descendentes (*offsprings*) que poderão ainda sofrer mutações.

4.3.3 Iterações quasi-Newton

A iteração quasi-Newton é executada usando a fórmula de actualização BFGS na aproximação da inversa da Hessiana Lagrangeana.

A diferença entre o método usado e o método quasi-Newton padrão deve-se ao facto do conjunto de elite $\mathcal{E}$ não se manter inalterado ao longo das iterações (gerações), havendo pontos que poderão não ser seleccionados para a geração seguinte e outros novos que entrarão para o conjunto. O algoritmo quasi-Newton usa, como estratégia de globalização, uma procura unidimensional combinada com a regra de Armijo [70].

Capítulo 5

Técnicas de penalidade

As técnicas de penalidades são uma estratégia possível para a resolução de problemas de optimização com restrições. A simplicidade e adaptabilidade a problemas com muitas restrições são duas características importantes destas técnicas. As técnicas de penalidade classificam-se em métodos de pontos interiores e métodos de pontos exteriores. Neste capítulo são abordadas as técnicas de penalidade.

Os métodos clássicos admissíveis (métodos cujo iterado é admissível ao longo do processo iterativo) para problemas de optimização com restrições de desigualdade são chamados os métodos de restrições activas (ou conjunto activo). Os métodos de conjunto activo resolvem uma sequência de sub-problemas, alterando o conjunto activo até que uma solução óptima seja encontrada. Estes métodos tratam as restrições de desigualdade como se fossem restrições de igualdade, baseiam-se na função Lagrangeana e tendem a ser uma generalização do método de Simplex. Uma desvantagem destes métodos está relacionada com o aumento do número de restrições. Quando tal acontece o número potencial de sub-problemas aumenta exponencialmente. Outra desvantagem advém de manter as restrições exactamente satisfeitas, o que no caso linear é fácil, mas para problemas com restrições não lineares poderá ser muito difícil [70].

As técnicas de penalidade eliminam algumas dessas dificuldades. Estes métodos resolvem um problema de optimização com restrições através da resolução de uma sequência de problemas de optimização sem restrições. Sob determinadas condições, as soluções dos problemas sem restrições convergem para a solução do problema com restrições. Os problemas sem restrições envolvem uma função auxiliar (função de penalidade ϕ) que incorpora a função objectivo (ou a função Lagrangeana) juntamente com termos de 'penalidade' que medem a violação das restrições. A função auxiliar também inclui um ou mais parâmetros (designados por parâmetros de penalidade) que determinam a importância relativa das restrições. Alterando de uma forma adequada estes parâmetros, é gerada uma sequência de problemas onde o efeito do termo de penalidade se torna acentuado. Ao contrário dos métodos que usam o conjunto activo, a função auxiliar considera todas as restrições, mesmo quando há desigualdades.

Considere-se um problema de PNL finito:

$$\begin{aligned} &\min_{x \in R^n} f(x) \\ &s.a\ h_i(x) = 0 \quad i = 1, \ldots, o; \\ &h_i(x) \leq 0 \quad i = o+1, \ldots, q. \end{aligned} \tag{5.0.1}$$

Nos métodos de penalidade a função penalidade com um parâmetro pode-se definir da seguinte forma:

$$\phi(x, \mu) = f(x) + \mu\psi(x) \tag{5.0.2}$$

em que ϕ é uma função $R^n \times R_+ :\longrightarrow R$ e μ (o parâmetro de penalidade) é um escalar positivo. Os métodos de penalidade resolvem uma sequência de problema de minimização sem restrições da forma:

$$\min_x \phi(x, \mu^k). \tag{5.0.3}$$

Este problemas são parametrizados por μ^k, em que k representa a iteração. As técnicas de penalidade incluem dois tipos de métodos: os métodos barreira (ou de pontos interiores) e os métodos de penalidade externa (ou de pontos exteriores). Os primeiros são mais utilizados em problemas com desigualdades, enquanto que os métodos dos pontos exteriores são usados em problemas que contenham, também, restrições de igualdade.

Este capítulo está dividido em quatro secções. Nas primeira e segunda secção são descritos os métodos de penalidade clássicos: o método barreira e o dos pontos exteriores, respectivamente. Nestas secções o objectivo é a apresentação dos métodos de penalidade e por esta razão não são impostas hipóteses sobre as funções penalidade. Os fundamentos teóricos destes métodos estão disponíveis, por exemplo, em [20, 65, 71]. Nos métodos de penalidade o aumento/diminuição (dependendo do método) do parâmetro de penalidade leva muitas vezes a um mau condicionamento do problema sem restrições. Esta dificuldade pode ser evitada se a solução do problema for determinada por um valor finito do parâmetro de penalidade. As funções que determinam a solução com um valor do parâmetro de penalidade finito, chamam-se funções de penalidade exacta. A terceira secção é dedicada a este tipo de funções. Na última secção mostram-se algumas funções de penalidade usadas na PSI.

5.1 Métodos barreira ou de pontos interiores

Os métodos barreira são métodos iterativos estritamente admissíveis, isto é, são métodos que durante o processo iterativo, o iterado situa-se sempre no interior da região admissível. Existe a formação de uma espécie de barreira que mantém o iterado afastado da fronteira da região admissível. Habitualmente, estes métodos são usados em problemas com restrições de desigualdade descritos da seguinte forma:

$$\begin{aligned} &min \;\; f(x) \\ &s.a \;\; h_i(x) \leq 0, \;\; i = 1, \ldots, q. \end{aligned} \tag{5.1.1}$$

em que $f : R^n \longrightarrow R$ e $h_i : R^n \longrightarrow R$ para $i = 1, \ldots, q$.

As funções penalidade relativas ao problema (5.1.1) definem-se como (5.0.2), em que μ representa o parâmetro barreira e $\mu\psi(x)$ é o termo barreira. Existem dois tipos de funções barreira muito conhecidas na literatura [70], a função barreira logarítmica e a função barreira inversa, as quais são definidas respectivamente por:

$$\phi(x,\mu) = f(x) + \mu \sum_{i=1}^{q} log(-h_i(x))$$

$$\phi(x,\mu) = f(x) - \mu \sum_{i=1}^{q} \frac{1}{h_i(x)}.$$

O algoritmo conceptual para os métodos barreira é o seguinte:

Algoritmo 5.1.1 *Métodos barreira*

1. *Dados $\mu^0 > 0$, uma aproximação inicial x^0 e seja o contador de iterações $k = 0$.*
2. *Enquanto o critério de paragem não for satisfeito, fazer:*
   (a) $x^{k+1} = argmin\ \phi(x^k, \mu^k)$
   (b) *Actualizar (diminuir)* μ^k
   (c) $k \leftarrow k + 1$.

O passo do Algoritmo 5.1.1 com maior trabalho computacional é o passo $2(a)$, em que se minimiza a função barreira ϕ. Neste passo faz-se a minimização do problema sem restrições, o qual pode ser resolvido por métodos desenvolvidos para problemas de programação não linear sem restrições. Ao longo das iterações o parâmetro barreira μ^k vai diminuindo gradualmente para zero e, como o termo barreira é infinito no limite da região admissível, força a permanência dos iterados no seu interior.

A convergência do método só é conseguida se o parâmetro barreira tender para zero (passo $2(b)$) [65, 71]. No entanto, a diminuição excessiva deste parâmetro pode tornar a matriz Hessiana do problema (5.0.3) mal condicionada, dificultando a sua resolução.

5.2 Métodos de penalidade exterior ou de pontos exteriores

Os métodos de penalidade exterior são utilizados tanto na resolução de problemas com restrições de desigualdade como em problemas com restrições de igualdade. Consequente-

mente, podem ser usados em problemas na forma mais geral, ou seja, que contenham os dois tipos de restrições. Considere-se o problema de PNL na forma mais geral:

$$\begin{aligned} & min \;\; f(x) \\ & s.a \;\; h_i(x) = 0 \;\; i = 1, \ldots, o, \\ & \qquad h_i(x) \leq 0 \;\; i = o+1, \ldots, q. \end{aligned} \tag{5.2.1}$$

As funções f e h_i, para $i = 1, \ldots, q$ são funções de $R^n \longrightarrow R$. Os métodos de penalidade exterior usam funções de penalidade definidas como em (5.0.2), sendo as seguintes funções de penalidade, para resolver problemas do tipo (5.2.1), as mais conhecidas na literatura [49]:

- A função de penalidade de norma-1 ou l_1

$$\phi_1(x, \mu) = f(x) + \mu \left(\sum_{i=1}^{o} |h_i(x)| + \sum_{i=o+1}^{q} [h_i(x)]_+ \right), \text{ onde } [z]_+ = max\{0, z\}.$$

- A função penalidade de norma-2 ou l_2

$$\phi(x, \mu) = f(x) + \mu \left(\|h_{eq}(x)\|_2 + \|[h_{ineq}(x)]_+\|_2 \right).$$

  onde h_{eq} é o vector das restrições de igualdade e h_{ineq} o vector das restrições de desigualdade.

- A função penalidade l_2^2

$$\phi(x, \mu) = f(x) + \mu \left(\|h_{eq}(x)\|_2^2 + \|[h_{ineq}(x)]_+\|_2^2 \right).$$

- A função de norma infinita ou l_∞

$$\phi(x, \mu) = f(x) + \mu \left(\max_{i \in \{1, \ldots, o\}} |h_i(x)| + \max_{i \in \{o+1, \ldots, q\}} [h_i(x)_+] \right).$$

Destes exemplos, as funções diferenciáveis são as funções de penalidade l_2 e l_2^2.

Apresenta-se agora o algoritmo conceptual dos métodos de penalidade exterior.

Algoritmo 5.2.1 *Métodos de penalidade exterior*

1. *Dados $\mu^0 > 0$, uma aproximação inicial x^0 e seja o contador de iterações $k = 0$.*

2. *Enquanto o critério de paragem não for satisfeito, fazer:*

   (a) $x^{k+1} = argmin\ \phi(x^k, \mu^k)$

   (b) Actualizar (aumentar) μ^k

   (c) $k \leftarrow k + 1$.

Ao longo do processo iterativo, este algoritmo mantêm a solução das funções penalidade (passo $2(a)$) não admissível no problema original, sendo provocada uma penalização com o aumento do parâmetro penalidade (passo $2(b)$) e, consequentemente, forçando os minimizantes a deslocarem-se para a região admissível.

A prova de convergência do Algoritmo 5.2.1 pode ser consultada em [65, 71]. Nas provas de convergência deste algoritmo é imposta a condição de que o parâmetro penalidade μ^k deve aumentar, simbolicamente, $\mu \longrightarrow \infty$, mas o aumento exagerado do parâmetro de penalidade gera sub-problemas difíceis de resolver, no sentido da matriz Hessiana da função do sub-problema tornar-se mal condicionada.

Como foi visto, os métodos de penalidade tipo barreira ou externa geram sub-problemas que se podem tornar mal condicionados à medida que se vão aproximando da solução do problema original. O mal condicionamento da matriz Hessiana torna os algoritmos muito lentos e na maioria das vezes não solúveis. Esta dificuldade pode ser evitada se for possível obter a solução do problema com restrições através da solução de um sub-problema sem restrições para um valor finito do parâmetro de penalidade. Os métodos de penalidade exacta dão-nos essa garantia.

5.3 Métodos de penalidade exacta

Nas secções anteriores constatou-se que os métodos de penalidade têm a desvantagem do parâmetro de penalidade aumentar ou diminuir (dependendo se se trata de penalidade

externa ou barreira) de uma forma ilimitada provocando instabilidade numérica no processo iterativo. Esta desvantagem é ultrapassada com os métodos de penalidade exacta. Os métodos de penalidade exacta podem ser divididos em duas classes [78]: métodos baseados em *funções penalidade exacta* ou métodos baseados em *funções Lagrangeana aumentada*.

Definição 5.3.1 *A função de penalidade $\phi(x,\mu)$ define-se exacta quando existe um μ^* finito, tal que um minimizante local de $\phi(x,\mu)$, $x^*(\mu)$, é solução de* (5.0.1)*, x^*, para $\mu > \mu^*$.*

Utilizando uma função de penalidade exacta há a garantia de determinar a solução do problema sem restrições para um valor finito do parâmetro de penalidade evitando o possível mau condicionamento da matriz Hessiana. As funções de penalidade exactas mais conhecidas são a l_1 e l_∞.

Os métodos de penalidade exacta podem ser também baseados na função Lagrangeana aumentada. Estes métodos usam a função Lagrangeana do problema com restrições, são mais estáveis e convergem mais rapidamente para a solução do que os métodos clássicos.

A função Lagrangeana associada ao problema (5.2.1) é:

$$L(x,\lambda) = f(x) + \sum_{i=1}^{o} h_i(x)\lambda_i + \sum_{i=o+1}^{q} h_i(x)\lambda_i$$

em que $\lambda_i \geq 0$, para $i = 1,\ldots,q$ são os multiplicadores de Lagrange associados às restrições de igualdade e desigualdade. Uma função Lagrangeana aumentada para o problema (5.2.1) define-se:

$$\phi(x,\lambda,\mu) = f(x) + \sum_{i=1}^{o} h_i(x)\lambda_i + \sum_{i=o+1}^{q} h_i(x)\lambda_i + \frac{1}{2}\mu\left(\sum_{i=1}^{o}(h_i(x))^2 + \sum_{i=o}^{q}[h_i(x)]_+^2\right)$$

O algoritmo conceptual do método de penalidade com a função Lagrangeana aumentada aplicado ao problema (5.2.1) é descrito da seguinte forma:

Algoritmo 5.3.1 *Método de penalidade com a função Lagrangeana Aumentada [50]*

Dados λ^0, μ^0, $\varepsilon^0 > 0$

Iteração externa *: Para* $k = 0, 1, 2, \ldots$

- ***Iteração interna:*** *A partir da aproximação inicial* x^k*:*

$$x^+(\lambda^k, \mu^k) \simeq arg \min_{x \in R^n} \phi(x, \lambda^k, \mu^k)$$

  de acordo com o critério de paragem que depende de ε^k
  $x^{k+1} \leftarrow x^+(\lambda^k, \mu^k)$

- ***Se*** *as condições de convergência final se verificam* ***então*** *terminar* $x^* \leftarrow x^{k+1}$*;*
- ***senão*** *actualizar os multiplicadores de Lagrange, para obter* λ^{k+1}*; actualizar o parâmetro de penalidade, para obter* μ^{k+1} *(aumentar); actualizar* ε^k *(reduzir).*

A iteração interna é o passo com maior trabalho computacional e a minimização da função ϕ pode não ser exacta. Nalgumas variantes só se implementa uma iteração do método de Newton. A convergência do algoritmo poderá ser mais rápida se os multiplicadores de Lagrange forem actualizados ao longo do processo iterativo, em vez de se manterem constante.

5.4 Funções penalidade para a PSI

Considere-se o problema de PSI só com restrições infinitas definido da seguinte forma:

$$\begin{aligned} &\min_{x \in R^n} \ f(x) \\ s.a \ \ &g_i(x,t) \leq 0 \text{ para } i = 1, \ldots, m \text{ e } \forall t \in T \ . \end{aligned} \tag{5.4.1}$$

$f(x)$ é a função objectivo e $g_i(x,t)$, para $i = 1, \ldots, m$ são as funções de restrição infinita. Assume-se que ambas as funções são duas vezes continuamente diferenciáveis em todos os argumentos.

Nas últimas décadas foram propostas várias funções de penalidade (de norma l_1 e l_∞) para problemas de PSI definidos na forma (5.4.1) [8, 88, 89, 90, 137].

Uma definição possível de uma função de penalidade do tipo l_∞ para o problema (5.4.1) é:

$$\phi_\infty(x,\mu) = f(x) + \mu \max_{i\in\{1,\ldots,m\}} \left(\max_{t\in T} [g_i(x,t)]_+ \right)$$

onde μ é o parâmetro de penalidade positivo.

Os métodos de redução baseiam-se neste tipo de funções de penalidade e para poder usá-las precisam de todos os maximizantes globais e locais das restrições infinitas. A determinação de todos os maximizantes requeridos é conseguida usando um algoritmo de optimização global. Os trabalhos [88, 89, 90, 137] fazem referência a esses algoritmos.

A.R. Conn e N.I.M. Gould, em [8], apresentam uma função de penalidade exacta para PSI que é uma generalização da função de penalidade exacta de norma l_1 para problemas de programação não-linear. A função apresentada é a seguinte:

$$\phi_{CG}(x,\mu) = f(x) + \mu \sum_{i=1}^{m} \frac{\left(\sum_{j=1}^{s_i} \left(\int_{\Omega_{ij}(x)} g_i(x,t)dt \right) \right)}{\sum_{j=1}^{s_i} \left(\int_{\Omega_{ij}(x)} dt \right)}, \tag{5.4.2}$$

onde para todo x, existe um conjunto finito de $\Omega_{ij}(x)$ tal que:

1. $\Omega_{ij} \subseteq T, 1 \leq j \leq s_i = s_i(x) < \infty$,
2. $g_i(x,t) \geq 0$, $\forall t \in \Omega_{ij}$ e $g_i(x,t) < 0$,$\forall t \in T_i \setminus \bigcup_{j=1}^{s_i} \Omega_{ij}(x)$,
3. $\Omega_{ij} \cap \Omega_{ik}(x) = \varnothing$ se $j \neq k$, e
4. $\Omega_{ij}(x)$ é ligado e não-trivial, isto é, $\int_{\Omega_{ij}(x)} dt > 0$

O denominador foi incluído para que a não admissibilidade fosse suficientemente penalizada e a função de penalidade pudesse tornar-se exacta. A maior desvantagem da função penalidade (5.4.2) é a determinação dos conjuntos Ω_{ij}, $i = 1,\ldots,m$, $j = 1,\ldots,s_i$. Uma alternativa a esta definição pode ser:

$$\phi_{CG}(x,\mu) = f(x) + \mu \sum_{i=1}^{m} \frac{\left(\sum_{j=1}^{s_i} \left(\int_T [g_i(x,t)]_+ dt \right) \right)}{\sum_{j=1}^{s_i} \left(\int_T [sign\,(g_i(x,t))]_+ dt \right)}, \tag{5.4.3}$$

que é de certa forma equivalente a (5.4.2) [88]. De qualquer modo, a função penalidade (5.4.3) é não suave e não diferenciável sendo necessário usar um algoritmo sem derivadas na sua minimização. Ver, por exemplo, em [80] uma revisão sobre optimização não diferenciável e em [141] o método de Powell sem derivadas.

K.L. Teo e C.J. Goh em [109] propuseram a transcrição das restrições do problema (5.4.1) para um problema finito não linear, onde as restrições de desigualdade infinita foram transformadas em restrições de igualdade finitas na forma:

$$\mathcal{G}_i(x) \equiv c \int_T [g_i(x,t)]_+^2 dt = 0, \text{ para } i = 1, \ldots, m,$$

em que c é um factor de ponderação empírica usado para melhorar a precisão numérica. Uma das dificuldades é que as restrições do problema PNL resultante não satisfazem a condição de regularidade usual. De facto, o valor do gradiente para todos pontos estritamente admissíveis é zero e a convergência de alguns métodos numéricos não pode ser garantida. Apesar disto, os autores relatam o sucesso obtido das experiências numéricas, com dois exemplos.

L.S. Jennings e K.L. Teo, em [43], mostraram como o problema (5.4.1) podia ser substituído por um problema aproximado (semelhante ao de [109]) com restrições diferenciáveis contínuas de primeira ordem em x. O problema aproximado é então,

$$\begin{aligned} &\min_{x \in R^n} \ f(x) \\ &s.a \ \ \mathcal{G}_{i,\epsilon}(x) \equiv \int_T g_{i,\epsilon}(x,t) dt = 0, \ i = 1, \ldots, m \end{aligned} \tag{5.4.4}$$

com,

$$g_{i,\epsilon}(x,t) = \begin{cases} 0, & \text{se } g_i(x,t) < -\epsilon; \\ \frac{(g_i(x,t)+\epsilon)^2}{4\epsilon}, & \text{se } -\epsilon \leq g_i(x,t) \leq \epsilon; \\ g_i(x,t), & \text{se } g_i(x,t) > \epsilon, \end{cases}$$

onde ϵ é um parâmetro pequeno positivo de suavização e

$$\nabla_x g_{i,\epsilon}(x,t) = \begin{cases} 0, & \text{se } g_i(x,t) < -\epsilon; \\ \frac{\nabla_x g_i(x,t)(g_i(x,t)+\epsilon)}{2\epsilon}, & \text{se } -\epsilon \leq g_i(x,t) \leq \epsilon; \\ \nabla_x g_i(x,t), & \text{se } g_i(x,t) > \epsilon. \end{cases}$$

As restrições de igualdade $\mathcal{G}_{i,\epsilon}(x) = 0$ do problema (5.4.4), mais uma vez, não satisfazem a condição de regularidade usual e não é aconselhável resolver o problema com a formulação (5.4.4). Em alternativa pode resolver-se o seguinte problema aproximado

$$\begin{aligned} &\min_{x \in R^n} \ f(x) \\ &s.a \ \ \mathcal{G}_{i,\epsilon}(x) < \varrho, \ i = 1, \ldots, m \end{aligned}$$

para ϱ suficientemente pequeno.

G.A. Watson em [137, 138] propõe uma função de penalidade exacta de norma l_1 na resolução de problemas de PSI contendo apenas uma restrição infinita e T é um subconjunto finito de R. O autor sugere o uso da função penalidade exacta de norma l_1 combinada com métodos de programação quadrática sequencial. Sendo $I = I_1 \cup I_2$ o conjunto de restrições activas, em que I_1 é um conjunto de pontos que pertencem ao interior de T e I_2 o conjunto de pontos que estão na fronteira de T. A função usada é a seguinte:

$$\phi(x, \mu) = f(x) + \mu \sum_{i \in I} h_i(x)_+ \tag{5.4.5}$$

onde,

$$h_i(x) = g(x, t_i(x)), \ i \in I_1,$$

$$h_i(x) = g(x, t_i), \ i \in I_2$$

e μ é um escalar positivo. O autor aplica estratégias de globalização na resolução dos sub-problemas quadráticos para encontrar direcções de descida da função de penalidade exacta.

C.J. Price e I.D. Coope, em [88, 89, 90], propõem uma função penalidade aumentada baseada em norma infinita. Os autores consideram um problema de PSI apenas com uma restrição infinita descrito como em (2.0.1). A função penalidade aumentada usa dois parâmetros de penalidade para controlo da admissibilidade e é definida da seguinte forma:

$$\phi_P(x, \mu, \nu) = f(x) + \mu\theta_\infty(x) + \frac{1}{2}\nu\theta_\infty^2(x), \quad \text{em que } \theta_\infty(x) = \max_{t \in T}[g(x, t)]_+. \tag{5.4.6}$$

Os parâmetros de penalidade são $\mu > 0$ e $\nu \geq 0$. O segundo termo foi adicionado para prevenir o aumento exagerado do μ, fazendo com que a função seja exacta. O termo $\theta_\infty(x)$

é de norma infinita na violação das restrições, tornando a função de penalidade ϕ_P contínua $\forall x \in R^n$.

Capítulo 6

O algoritmo de redução local

Neste capítulo apresenta-se o algoritmo de redução local baseado na técnica de penalidade e os respectivos fundamentos teóricos. A função de penalidade proposta é apresentada e são demonstradas as suas propriedades teóricas.

Uma parte do trabalho desenvolvido consistiu na proposta de um algoritmo de redução local baseado na técnica de penalidade. Neste capítulo começa-se por fazer uma revisão de alguns conceitos usados nas secções subsequentes. Na Secção 6.2 é apresentada a função de penalidade proposta e as respectivas propriedades. Na última secção faz-se a descrição detalhada do algoritmo de redução local.

6.1 Revisão de conceitos

Recorde-se a formulação matemática de um problema de PSI na forma mais geral:

$$\begin{aligned} & \min_{x \in R^n} f(x) \\ \text{s.a } & g_i(x,t) \leq 0 \quad i = 1, \ldots, m \\ & h_i(x) = 0 \quad i = 1, \ldots, o \\ & h_i(x) \leq 0 \quad i = o+1, \ldots, q \\ & \forall t \in T \subset R^p \end{aligned} \tag{6.1.1}$$

e também a dos problemas multi-locais:

$$\max_{t \in T} \; g_i(\bar{x}, t), \; i = 1, \ldots, m. \tag{6.1.2}$$

6.2 Técnica de penalidade

6.2.1 Escolha da função de penalidade

A função de penalidade considerada no algoritmo é uma extensão da função proposta em [88, 90] permitindo a inclusão de restrições finitas. A adopção da função de penalidade baseou-se num estudo empírico envolvendo quatro funções de penalidade diferentes. As funções consideradas foram: ϕ_1, ϕ_2, ϕ_∞ e ϕ_P. As três primeiras funções são de norma 1, 2 e ∞, respectivamente. A função de penalidade ϕ_P foi proposta por Price e combina dois termos baseados na norma-∞ das restrições.

A função de penalidade ϕ_1 para um problema do tipo (6.1.1) com $o = q = 0$ tem a seguinte forma:

$$\phi_1(x, \mu) = f(x) + \mu\theta_1(x)$$

onde

$$\theta_1(x) = \sum_{i=1}^{m} \left(\max_{t \in T} [g_i(x, t)]_+ \right),$$

μ é o parâmetro de penalidade positivo. A maior desvantagem desta função de penalidade é a não diferenciabilidade originada pela função $[]_+$.

A função de penalidade ϕ_2 é definida por:

$$\phi_2(x, \mu) = f(x) + \mu\theta_2(x)$$

onde

$$\theta_2(x) = \sum_{i=1}^{m} \left(\max_{t \in T} [g_i(x, t)]_+^2 \right).$$

Uma desvantagem desta função é não ser exacta. Para obter uma solução do problema (6.1.1) o parâmetro μ pode tomar valores elevados, resultando num possível mau condicionamento dos problemas a serem resolvidos. Em contrapartida esta função tem a vantagem de ser uma vez continuamente diferenciável.

A função de penalidade ϕ_∞ de norma-∞ nas restrições descreve-se como:

$$\phi_\infty(x,\mu) = f(x) + \mu\theta_\infty(x)$$

onde

$$\theta_\infty(x) = \max_{i\in\{1,\ldots,m\}} \left(\max_{t\in T} [g_i(x,t)]_+ \right).$$

A função de penalidade ϕ_P foi proposta por Price e Coope [88, 90] e é uma função de penalidade ϕ_∞ aumentada. A função usa dois parâmetros de penalidade para controlar a admissibilidade e é definida da seguinte forma:

$$\phi_P(x,\mu,\nu) = f(x) + \mu\theta_\infty(x) + \frac{1}{2}\nu\theta_\infty^2(x), \tag{6.2.1}$$

onde $\mu > 0$ e $\nu \geq 0$ são os parâmetros de penalidade. O segundo termo foi adicionado para prevenir o aumento exagerado do parâmetro de penalidade μ. Sob determinadas hipóteses pode-se mostrar que o óptimo do problema (6.1.1) (com $m = 1$ e $o = q = 0$) é ponto crítico de $\phi_P(x,\mu,\nu)$ e assim como o recíproco [90]. Esta função é conhecida como sendo uma função de penalidade exacta. No entanto, como usualmente os parâmetros μ^* e ν^* não são conhecidos, não é possível uma única minimização da função de penalidade. Normalmente, para este tipo de funções, os parâmetros de penalidade estão relacionados com os multiplicadores de Lagrange das restrições activas na solução do problema (original) com restrições. Na prática, os parâmetros μ e ν são actualizados ao longo das iterações considerando novas estimativas dos multiplicadores de Lagrange, caso na iteração actual não haja admissibilidade para o problema (6.1.1).

Realizou-se um estudo com estas quatro funções aplicadas a problemas de PSI com apenas restrições infinitas e verificou-se que os melhores resultados computacionais foram obtidos com a função penalidade ϕ_P. Os resultados desta experiência podem ser consultados no próximo capítulo.

O uso da função ϕ_P para o caso mais geral é baseado nesses resultados e consiste na extensão da função ϕ_P modificando o termo θ_∞ para incluir a violação das restrições finitas, *i.e.*, considerando,

$$\bar{\theta}_\infty(x) = \max\left(\max_{1\leq i\leq m}\max_{t\in T}[g_i(x,t)]_+, \max_{1\leq i\leq o}|h_i(x)|, \max_{o+1\leq i\leq q}[h_i(x)]_+\right) \tag{6.2.2}$$

em substituição de θ_∞ na expressão (6.2.1), sendo a nova função de penalidade designada por $\bar{\phi}_P$.

6.2.2 A função de penalidade proposta e propriedades

Esta secção começa por apresentar algumas hipóteses e conceitos necessários para a verificação das propriedades, que vão ser enunciadas, da função de penalidade proposta.

Seja F o conjunto de pontos admissíveis do problema (6.1.1).

$$\begin{aligned} F = \{x \in R^n | g_i(x,t) \leq 0, \quad i = 1,\ldots,m, \forall t \in T, \\ h_i(x) = 0,\ i = 1,\ldots,o, \quad h_i(x) \leq 0,\ i = o+1,\ldots,q\}. \end{aligned} \tag{6.2.3}$$

Considere-se ainda as seguintes hipóteses:

Hipótese 6.2.1 *int*$(F) \neq \emptyset$

Hipótese 6.2.2 *O conjunto F é compacto.*

A função de penalidade é então definida como:

$$\bar{\phi}_P(x,\mu,\nu) = f(x) + \mu\bar{\theta}_\infty(x) + \frac{1}{2}\nu\bar{\theta}^2_\infty(x),\ \mu > 0 \text{ e } \nu \geq 0, \tag{6.2.4}$$

onde $\bar{\theta}_\infty(x)$ é definido como em (6.2.2).

O princípio da técnica de penalidade externa é verificado porque sendo $\bar{\theta}_\infty(x)$ de norma infinita, é uma função contínua que se anula no conjunto de pontos admissíveis e é positiva fora dele.

Lema 6.2.1 *Seja* $P(x) = \mu\bar{\theta}_\infty(x) + \frac{1}{2}\nu\bar{\theta}_\infty(x)^2$. *Para a função de penalidade* (6.2.4) *a seguinte propriedade verifica-se:*

$$\begin{cases} P(x) = 0, & \text{se } x \in F \\ P(x) > 0, & \text{se } x \notin F \end{cases}$$

Prova. Se x é um ponto admissível do problema (6.1.1), *i.e.*, $x \in F$, então tem-se que $g_i(x,t) \leq 0,\ i = 1,\ldots,m,\ \forall t \in T,\ h_i(x) = 0$, para $i = 1,\ldots,o$ e $h_i(x) \leq 0$, para $i = o+1,\ldots,q$ o que implica $\bar{\theta}_\infty(x) = 0$ e consequentemente $P(x) = 0$.

Se x não é ponto admissível, *i.e.*, $x \notin F$, então existe pelo menos uma restrição violada. Nas restrições infinitas $\exists i \in \{1,\ldots,m\}$ tal que $g_i(x,t) > 0$, ou nas restrições de igualdade $\exists i \in \{1,\ldots,o\}$ tal que $h_i(x) \neq 0$ ou nas restrições de desigualdade $\exists i \in \{o+1,\ldots,q\}$ tal que $h_i(x) > 0$. Como a violação das restrições é dada por $|\cdot|$ nas restrições de igualdade e $[]_+$ para as restantes, o valor de $\bar{\theta}_\infty$, por definição, é positivo. O $P(x)$ é então uma soma de valores nulos em que pelo menos um termo é positivo e resulta que $P(x) > 0$. ■

A introdução dos parâmetros de penalidade permite que com o aumento destes a violação das restrições seja cada vez mais penalizada, de tal forma que as soluções da função penalidade, para uma sequência controlada de aumentos dos parâmetros de penalidade, produza uma sequência cujos os pontos de acumulação resolvem o problema original (com restrições).

O problema de PSI é localmente equivalente a um problema de PNL e a existência de um mínimo do problema finito será uma aproximação local ao problema de PSI. Veja-se em que condições se pode reduzir o problema de PSI a um problema PNL. Comece-se por assumir que $f, g \in \mathcal{C}^2$. Seja $\bar{x} \in F$ (*i.e.*, $\bar{x}$ é um ponto admissível). Pela Hipótese 6.2.2 o número de minimizantes é finito. Sejam $\bar{t}_{i,1},\ldots,\bar{t}_{i,q(\bar{x})}$ para todo $i = 1,\ldots,m$ as soluções locais dos problemas:

$$\max_{t\in T} g_i(\bar{x},t),\ i = 1,\ldots,m. \tag{6.2.5}$$

Assumindo que existe uma vizinhança $\overline{\mathcal{U}}$ de $\bar{x}$ tal que:

$$t_{i,j} : \overline{\mathcal{U}} \to T,\ t_{i,j}(x) = \bar{t}_{i,j},\ i = 1,\ldots,m \text{ e } j = 1,\ldots,q_i(\bar{x}),$$

os $t_{i,j}(\bar{x})$ são as soluções locais do problema (6.2.5). Então, pode definir-se

$$G_{i,j}(x) = g_i(x, t_{i,j}(x)),\ i = 1, \ldots, m \text{ e } j = 1, \ldots, q_i(x),$$

em que $G_{i,j} \in \mathcal{C}^2(\overline{\mathcal{U}})$.

O conjunto admissível é agora:

$$\begin{aligned}\overline{\mathcal{F}} = F \cap \overline{\mathcal{U}} = \{x \in \overline{\mathcal{U}} | & G_{i,j}(x) \leq 0\ i = 1, \ldots, m \text{ e } j = 1, \ldots, q_i(x), \\ & h_i(x) = 0,\ i = 1, \ldots, o,\ h_i(x) \leq 0,\ i = o+1, \ldots, q\}.\end{aligned}$$

Podendo substituir-se localmente o problema de PSI (6.1.1) por um problema finito (PF) da forma:

$$\begin{aligned} & \min_{x \in R^n} f(x) \\ \text{s.a } & G_{i,j}(x) \leq 0 \quad i = 1, \ldots, m \text{ e } j = 1, \ldots, q_i(x), \\ & h_i(x) = 0 \quad i = 1, \ldots, o, \\ & h_i(x) \leq 0 \quad i = o+1, \ldots, q. \end{aligned} \tag{6.2.6}$$

Seja $tot = \sum_{i=1}^{m} q_i(x) + q - o$ o número total de restrições de desigualdade do problema (6.2.6). O PF pode ser re-definido como:

$$\begin{aligned} & \min_{x \in R^n} f(x) \\ \text{s.a } & \bar{h}_i(x) \leq 0 \quad i = 1, \ldots, tot, \\ & h_i(x) = 0 \quad i = 1, \ldots, o. \end{aligned} \tag{6.2.7}$$

em que f é a função objectivo, $\bar{h}_i(x)$ para $i = 1, \ldots, tot$ são todas as funções das restrições de desigualdade h_i e $G_{i,j}(x)$ do problema (6.2.6) e $h_i(x)$, $i = 1, \ldots, o$ as funções das restrições de igualdade.

Portanto, o problema de PSI (6.1.1) pode ser localmente reduzido a um problema de PNL definido em (6.2.7).

Considere-se o problema (6.2.7). As funções de penalidade $\bar{\phi}_\infty$ e $\bar{\phi}_P$ para este problema definem-se agora da seguinte forma:

$$\bar{\phi}_\infty(x, \mu) = f(x) + \mu \bar{\bar{\theta}}_\infty(x),\ \mu > 0,$$

$$\bar{\phi}_P(x,\mu,\nu) = f(x) + \mu\bar{\bar{\theta}}_\infty(x) + \frac{1}{2}\nu\bar{\bar{\theta}}^2_\infty(x),\ \mu > 0 \text{ e } \nu \geq 0$$

onde $\bar{\bar{\theta}}_\infty(x)$ é definido como:

$$\bar{\bar{\theta}}_\infty(x) = \max\left(\max_{1\leq i\leq tot}[\bar{h}_i(x)]_+, \max_{1\leq i\leq o}|h_i(x)|,\right) \tag{6.2.8}$$

O seguinte corolário é consequência do Lema 6.2.1.

Corolário 6.2.1

$$\begin{cases} \bar{\bar{\theta}}_\infty(x) = 0, & \textit{se } x \in \bar{F} \\ \bar{\bar{\theta}}_\infty(x) > 0, & \textit{se } x \notin \bar{F}. \end{cases}$$

Teorema 6.2.1 *Considere-se o problema PNL* (6.2.7). *Sejam f, $\bar{h}$ e h continuamente diferenciáveis numa vizinhança de um minimizante local estrito $\bar{x}$ do problema finito. Suponha-se que se verifica a seguinte condição de regularidade em $\bar{x}$.*

$\nabla\bar{h}_i(\bar{x}),\ i = 1,\ldots,tot$ são linearmente independentes e existe um $z \in R^n$ tal que:

$$\nabla\bar{h}_i(\bar{x})z < 0,\ i \in I = \{i|\bar{h}_i(\bar{x}) = 0,\ i = 1,\ldots,tot\}$$

e

$$\nabla h_i(\bar{x})z = 0,\ i = 1,\ldots,o.$$

Então existe $\bar{\mu} \geq 0$ tal que para $\mu \geq \bar{\mu}$, $\bar{x}$ é um minimizante local de $\bar{\phi}_P(x,\mu,\nu)$.

Prova. Usando o Teorema 4.4 em [32] tem-se que, para $\mu > \bar{\mu}$,

$$\bar{\phi}_\infty(\bar{x},\mu) \leq \bar{\phi}_\infty(x,\mu).$$

Considerando que

$$\bar{\phi}_P(\bar{x},\mu,\nu) = \bar{\phi}_\infty(\bar{x},\mu) + \frac{1}{2}\nu\bar{\bar{\theta}}^2_\infty(\bar{x})$$

resulta em

$$\bar{\phi}_P(\bar{x},\mu,\nu) = \bar{\phi}_\infty(\bar{x},\mu)$$

uma vez que $\bar{x}$ é admissível e usando o Corolário 6.2.1 vem que $\bar{\bar{\theta}}_\infty(\bar{x}) = 0$.

Tem-se então que

$$\bar{\phi}_P(\bar{x},\mu,\nu) = \bar{\phi}_\infty(\bar{x},\mu) \leq \phi_\infty(x,\mu) \leq \bar{\phi}_\infty(x,\mu) + \frac{1}{2}\nu\bar{\bar{\theta}}^2_\infty(x) = \bar{\phi}_P(x,\mu,\nu)$$

uma vez que $\nu \geq 0$ e $\bar{\bar{\theta}}_\infty(\bar{x}) \geq 0$ pelo Corolário 6.2.1.

■

Teorema 6.2.2 *Suponha-se que existe um $\bar{\mu} \geq 0$ tal que para todo $\mu \geq \bar{\mu}$, $\bar{\phi}_P(\bar{x},\mu,\nu) \leq \bar{\phi}_P(x,\mu,\nu)$ para todo x numa vizinhança aberta $N(\bar{x})$ no qual contém algum ponto admissível do problema* (6.2.7)*. Se f, $\bar{h}$ e h são diferenciáveis em $\bar{x}$ então $\bar{x}$ e algum $\bar{u} \in R^{tot}$ e $\bar{v}) \in R^o$ satisfazem as condições de optimalidade necessárias KKT do problema* (6.2.7)*, ou seja:*

$$\exists \bar{u},\bar{v}: \quad \nabla f(\bar{x}) + \sum_{i=1}^{tot} \bar{u}_i \nabla \bar{h}_i(\bar{x}) + \sum_{i=1}^{o} \bar{v}_i \nabla h_i(\bar{x}) = 0$$
$$\bar{u} \geq 0 \quad \bar{u}\bar{h}(\bar{x}) = 0, \quad \bar{h}(\bar{x}) \leq 0, \quad h(\bar{x}) = 0.$$

Prova. Seja $\bar{\mu} \geq 0$ tal que para todo $\mu \geq \bar{\mu}$:

$$\bar{\phi}_P(\bar{x},\mu,\nu) \leq \bar{\phi}_P(x,\mu,\nu)$$

Considerando que

$$\bar{\phi}_P(\bar{x},\mu,\nu) = \bar{\phi}_\infty(\bar{x},\mu) + \frac{1}{2}\nu\bar{\bar{\theta}}^2_\infty(\bar{x})$$

resulta em

$$\bar{\phi}_P(\bar{x},\mu,\nu) = \bar{\phi}_\infty(\bar{x},\mu)$$

uma vez que $\bar{x}$ é ponto admissível ($\bar{\bar{\theta}}_\infty(\bar{x}) = 0$).

Tem-se ainda que:

$$\begin{aligned}\bar{\phi}_P(x,\mu,\nu) &= f(x) + \mu\bar{\bar{\theta}}_\infty(x) + \frac{1}{2}\nu\bar{\bar{\theta}}^2_\infty(x) \\ &= f(x) + \bar{\bar{\theta}}_\infty(x)\left[\mu + \frac{1}{2}\nu\bar{\bar{\theta}}_\infty(x)\right]\end{aligned}$$

Como $\bar{\bar{\theta}}_\infty(x)$ é limitado, tem-se que:

$$\forall x \in N(\bar{x}) \; \exists \bar{\nu} \text{ finito tal que } \bar{\bar{\theta}}_\infty(x) \leq \bar{\nu}$$

e, portanto,

$$\bar{\phi}_P(x, \mu, \nu) = \bar{\phi}_\infty(x, \mu + \frac{1}{2}\nu\bar{\nu}).$$

Sem perda de generalidade, para $\left(\mu + \frac{1}{2}\nu\bar{\nu}\right) \geq \bar{\mu}$:

$$\bar{\phi}_\infty(\bar{x}, \mu + \frac{1}{2}\nu\bar{\nu}) \leq \bar{\phi}_\infty(x, \mu + \frac{1}{2}\nu\bar{\nu})$$

e aplicando o Teorema 4.8 de [32] obtém-se o resultado. ■

O uso de dois parâmetros de penalidade na função $\bar{\phi}_P$ permite que o valor de $\bar{\mu}$ seja menor do que o necessário para $\bar{\phi}_\infty$, obtendo-se problemas com melhor condicionamento.

6.3 Algoritmo de penalidade

O algoritmo de penalidade está dividido em dois tipos de iterações: iterações internas e externas.

As iterações internas estão relacionadas com a minimização da função de penalidade para um determinado valor fixo de μ e ν.

As iterações externas estão relacionadas com a resolução da sequência de sub-problemas formados pela função de penalidade $\bar{\phi}_P$ parametrizada por μ e ν. Nesta fase são calculadas todas as soluções globais (assim como algumas locais) dos problemas definidos em (6.1.2).

A sequência $\{x^k\}$, onde k é o contador de iteração externo, é formado pela sequência de soluções dos sub-problemas

$$x^{k+1} = arg \min_{x \in R^n} \bar{\phi}_P(x, \mu, \nu)$$

parametrizados pelos parâmetros de penalidade μ e ν.

Se o critério de paragem é verificado (*i.e.*, o ponto x^{k+1} é admissível e o erro relativo é menor do que um determinado valor pequeno) então o algoritmo pára com uma aproximação à solução do problema (6.1.1), caso contrário os parâmetros de penalidade são actualizados e inicia-se uma nova iteração externa.

6.3.1 Inicialização dos parâmetros de penalidade

Na inicialização do parâmetro de penalidade μ é considerada a não admissibilidade do ponto inicial x^0. Esta escolha pretende reduzir os cálculos computacionais na minimização do problema, quando as escalas entre os valores da função objectivo e das funções de restrição são bastante díspares.

Para calcular a não admissibilidade começa-se por determinar os conjuntos $\mathcal{A}_i^0$ para $i = 1, \ldots, m$, formados pelos maximizantes (globais e locais) dos sub-problemas definidos na equação (6.1.2). Se o menor valor das violações das restrições (infinitas e finitas) for inferior a um determinado valor pequeno, então o valor inicial de μ é 1. Caso contrário, μ é, em valor absoluto, o quociente entre a função objectivo em x^0 e o menor valor da violação das restrições. O algoritmo de inicialização do parâmetro de penalidade μ é o seguinte:

Algoritmo 6.3.1 *Inicialização do parâmetro de penalidade* μ

Seja $\xi = min(min_{i\in\{1,\ldots,m\},t\in\mathcal{A}_i^0}[g_i(x^0,t)]_+, min_{1\leq j\leq o}|h_j(x^0)|, min_{o+1\leq j\leq q}[h_j(x^0)]_+)$*;*

Se $\xi \approx 0$

então $\mu = 1$.

senão $\mu = \left|\frac{f(x^0)}{\xi}\right|$.

O parâmetro de penalidade ν é inicializado a 1.

6.3.2 Actualização dos parâmetros penalidade

Para motivar a equação de actualização dos parâmetros de penalidade para a função $\bar{\phi}_P$ é introduzida a função Langrangeana, $L(x, u, v)$ do problema (6.2.7).

Seja

$$L(x,u,v) = f(x) + \sum_{i=1}^{tot} u_i \bar{h}_i(x) + \sum_{i=1}^{o} v_i h_i(x),$$

onde u_i e v_i são os multiplicadores de Lagrange relativos às restrições de desigualdade e igualdade, respectivamente. No caso das restrições de desigualdade, para além das restrições finitas consideram-se as restrições associadas aos pontos t_i fazendo com que a restrição

$g(x,t) \leq 0$ seja activa. Note-se que t_{ij}, $i = 1,\ldots,m$ e $j = 1,\ldots,q_i(x)$, são os óptimos globais e alguns locais para o problema (6.1.2).

Considerando também a função de penalidade $\bar{\phi}_P$,

$$\begin{aligned}\bar{\phi}_P(x,\mu,\nu) &= f(x) + \mu\bar{\bar{\theta}}_\infty(x) + \frac{1}{2}\nu\bar{\bar{\theta}}^2_\infty(x)\\ &= f(x) + \bar{\bar{\theta}}_\infty(x)\left(\mu + \nu\bar{\bar{\theta}}_\infty\right),\end{aligned}$$

onde $\bar{\bar{\theta}}_\infty(x)$ é definido como em (6.2.8).

Se $\|\cdot\|$ é uma norma de R^n então existe uma norma dual correspondente $\|\cdot\|'$ na qual está definida como:

$$\|x\|' = \max_{y\neq 0} x^T\frac{y}{\|y\|}, \forall x \in R^n$$

A partir da definição de norma dual chega-se à chamada desigualdade de Cauchy generalizada:

$$|x^T y| \leq \|x\|'\|y\|, \forall x, y \in R^n.$$

Para relacionar o ponto estacionário do problema (6.2.7) com ponto estacionário da função de penalidade é necessário a expressão de derivada direccional da função de penalidade, num ponto admissível do problema (6.2.7).

Lema 6.3.1 *(Lema 2.1 em [9])*

Se as funções $\{\bar{h}_i(x)\}_{i=1}^{tot}$ *e* $\{h_i(x)\}_{i=1}^{o}$ *são continuamente diferenciáveis e* $\bar{\bar{\theta}}_\infty(x)$ *é definido como em* (6.2.8)*, então para alguma direcção* $d \in R^n$*, a derivada direccional* $D_d\bar{\bar{\theta}}_\infty(x)$ *existe e é definida como:*

$$D_d\bar{\bar{\theta}}_\infty(x) = \begin{cases} \max_{\{i\in I, j\in J\}}\left(d^T[\nabla\bar{h}_i(x)]_+, d^T\nabla h_j(x)\right) & \text{se } \bar{\bar{\theta}}_\infty(x) > 0 \text{ e } I(x)\neq\emptyset \text{ ou } J(x)\neq\emptyset\\ \max_{\{i\in I, j\in J\}}\left(d^T[\nabla\bar{h}_i(x)]_+, d^T\nabla h_j(x)\right) & \text{se } \bar{\bar{\theta}}_\infty(x) = 0 \text{ e } I(x)\neq\emptyset \text{ ou } J(x)\neq\emptyset\\ 0 & \text{se } I(x)=\emptyset \text{ e } J(x)=\emptyset\end{cases}$$

onde

$$I(x) = \{i \in \{1,\ldots,tot\} : max[\bar{h}_i(x)]_+ = \bar{\bar{\theta}}_\infty(x)\} \text{ e}$$

$$J(x) = \{j \in \{1,\ldots,o\} : max|h_j(x)| = \bar{\bar{\theta}}_\infty(x)\}.$$

Definição 6.3.1 *(Definição 2.1 em [9]) Para valores fixos de μ e ν, um ponto $\bar{x}$ é um ponto crítico de $\bar{\phi}_P(x,\mu,\nu)$ se e só se, para todo $d \in R^n$, a derivada direccional de $D_d\bar{\phi}_P(\bar{x})$ é não negativa.*

Teorema 6.3.1 *Seja $\bar{x}$ um ponto estacionário KKT do problema* (6.2.7) *com multiplicadores de Lagrange correspondentes $(\bar{u},\bar{v}) \in R_+^{tot} \times R^o$, ou seja,*

$$\exists \bar{u},\bar{v}: \quad \nabla f(\bar{x}) + \sum_{i=1}^{tot} \bar{u}_i \nabla_i \bar{h}_i(\bar{x}) + \sum_{i=1}^{o} \bar{v}_i \nabla_i h_i(\bar{x}) = 0$$

$$\bar{u} \geq 0 \quad \bar{u}\bar{h}(\bar{x}) = 0, \quad \bar{h}(\bar{x}) \leq 0, \quad h(\bar{x}) = 0.$$

Se $\mu + \nu\bar{\bar{\theta}}_\infty(x) \geq \|u,v\|_1$ então x^* é um ponto estacionário de $\bar{\phi}_P(x,\mu,\nu)$.

Prova. *Usando a Definição 6.3.1, um ponto $\bar{x}$ é ponto crítico de $\bar{\phi}_P(x,\mu,\nu)$ se a derivada direccional de $D_d\bar{\phi}_P(\bar{x})$ é não negativa. Seja*

$$D_d\bar{\phi}_P(\bar{x},\mu,\nu) = \nabla f(\bar{x})^T d + \mu D_d\bar{\bar{\theta}}_\infty(\bar{x}) + \nu D_d\bar{\bar{\theta}}_\infty(\bar{x})\bar{\bar{\theta}}_\infty(\bar{x})$$

$$= \nabla f(\bar{x})^T d + D_d\bar{\bar{\theta}}_\infty(\bar{x})(\mu + \nu\bar{\bar{\theta}}_\infty(\bar{x}))$$

$$= -(\bar{u})^T[\nabla\bar{h}(\bar{x})d]_+ - (\bar{v})^T\nabla h(\bar{x})d + D_d\bar{\bar{\theta}}_\infty(\bar{x})(\mu + \nu\bar{\bar{\theta}}_\infty(\bar{x})) \tag{6.3.1}$$

$$\geq -\|\bar{u},\bar{v}\|_1\|[\nabla\bar{h}(\bar{x})]_+, \nabla h(\bar{x})\|_\infty + D_d\bar{\bar{\theta}}_\infty(\bar{x})(\mu + \nu\bar{\bar{\theta}}_\infty(\bar{x})) \tag{6.3.2}$$

$$= (\mu + \nu\bar{\bar{\theta}}_\infty(\bar{x}) - \|\bar{u},\bar{v}\|_1)D_d\bar{\bar{\theta}}_\infty(\bar{x}). \tag{6.3.3}$$

A expressão (6.3.1) *resulta do gradiente da função Lagrangeana ser zero. Em* (6.3.2) *é usada a desigualdade de Cauchy generalizada e pelo Lema 6.3.1 como $\|[\nabla\bar{h}(\bar{x})]_+, \nabla_h(\bar{x})\|_\infty = D_d\bar{\bar{\theta}}_\infty(\bar{x})$, obtém-se* (6.3.3).

Então, para $\mu + \nu\bar{\bar{\theta}}_\infty(\bar{x}) \geq \|\bar{u},\bar{v}\|_1$ tem-se que:

$$D_d\bar{\phi}_P(\bar{x},\mu,\nu) \geq 0, \ \forall d \in R^n.$$

E portanto, $\bar{x}$ é um ponto estacionário de $\bar{\phi}_P(\bar{x},\mu,\nu)$. ■

Considerando o teorema anterior, $n \times \left(\mu + \nu\bar{\bar{\theta}}_\infty(x)\right)$ é um limite superior de $\|u,v\|_1$.

O seguinte algoritmo é então utilizado para actualizar os parâmetros de penalidade da função de penalidade $\bar{\phi}_P$.

Algoritmo 6.3.2 *Actualização dos parâmetros de penalidade μ e ν.*

Seja $\theta_{crossover}$ um parâmetro dado.

- *Se* $\bar{\bar{\theta}}_\infty(x^{k+1}) \leq \theta_{crossover}$
  - $\mu^{k+1} = n \times \left(\mu^k + \nu^k \bar{\bar{\theta}}_\infty(x^{k+1})\right)$, $\nu^{k+1} = \nu^k$
- *senão*
  - $\mu^{k+1} = \mu^k$, $\nu^{k+1} = \frac{n \times \left(\mu^k + \nu^k \bar{\bar{\theta}}_\infty(x^{k+1})\right)}{\bar{\bar{\theta}}_\infty(x^{k+1})}$.

6.3.3 Critério de Paragem

O critério de paragem é verificado através da conjunção de duas condições. Uma delas está relacionada com número de iterações externas. Se o número de iterações externas do algoritmo ultrapassar um determinado número pré-estabelecido, o algoritmo termina a sua execução. A outra condição de paragem considera o erro relativo da solução encontrada. Caso o erro relativo Δ da solução encontrada seja próxima de zero, o algoritmo pára.

Para calcular o erro relativo, Δ, da solução encontrada na iteração k, procede-se da seguinte forma. Seja x^k a solução actual do problema, e seja x^{k+1} a solução encontrada. O erro relativo Δ é calculado da seguinte forma:

Se $||x^{k+1} - x^k|| \approx 0$

então $\Delta^k = ||x^{k+1} - x^k||$,

senão $\Delta^k = \frac{||x^{k+1} - x^k||}{||x^{k+1}||}$.

6.3.4 Iterações externas

Nesta fase determina-se a sequência $\{x^k\}$, onde k é um contador das iterações externas, através da resolução da sequência de sub-problemas

$$x^{k+1} \quad = \quad arg \min_{x \in R^n} \bar{\phi}_P(x, \mu, \nu)$$

parametrizados pelos parâmetros de penalidade μ e ν da função $\bar{\phi}_P$.

Se o critério de paragem é verificado o algoritmo pára com uma solução aproximada do problema (6.1.1), caso contrário, os parâmetros de penalidade são actualizados seguindo o Algoritmo 6.3.2 e inicia-se uma nova iteração externa.

O algoritmo com a iteração externa é apresentado de seguida.

Algoritmo 6.3.3 *O algoritmo de penalidade*

1. *Seja maxiter o número máximo de iterações externas.*

2. *Seja x^0 uma aproximação inicial do problema* (6.1.1). *Seja $k = 0$.*

3. *Calcular os conjuntos $\mathcal{A}_i^k$, $i = 1, \ldots, m$, contendo todos os maximizantes globais e alguns maximizantes locais para o problema* (6.1.2) *com $\bar{x}$ substituído por x^k.*

4. *Determinar μ^0 usando o Algoritmo 6.3.1 e seja $\nu^0 = 1$.*

5. *Usar x^k como valor inicial para resolver o problema de optimização da função de penalidade sem restrições:*

$$\min_{x \in R^n} \quad \bar{\phi}_P(x, \mu, \nu) \tag{6.3.4}$$

   onde o conjunto T substituído por $\mathcal{A}_i^k \cup E$ em $\bar{\bar{\theta}}_\infty(x)$. E é uma grelha estabilizadora (grelha uniformemente distribuída em R^p).

   Seja $x^{k+1} = \arg\min \bar{\phi}_P(x, \mu, \nu)$ o minimizante da função de penalidade encontrado.

6. *Verificar a admissibilidade de x^{k+1} calculando os conjuntos $\mathcal{A}_i^{k+1}$, $i = 1, \ldots, m$ (resolvendo* (6.1.2) *com $\bar{x}$ substituído por x^{k+1}).*

   Se o critério de paragem é verificado em x^{k+1}

   - *então pára sendo x^{k+1} uma solução aproximado do problema* (6.1.1).
   - *caso contrário actualizar os parâmetros de penalidade de acordo com o Algoritmo 6.3.2.*

7. *Seja $k = k + 1$ e voltar para o passo 5.*

6.3.5 Iterações internas

Nas iterações internas faz-se a minimização da função de penalidade $\bar{\phi}_P(x, \mu, \nu)$. Uma vantagem da função $\bar{\phi}_P(x, \mu, \nu)$ é a de ser exacta, dando-nos a garantia de que os parâmetros de penalidade não crescem ilimitadamente e evitando desta forma uma possível instabilidade numérica. Outra vantagem desta função é a de ser contínua. A função é de norma-∞ e não havendo pontos de descontinuidade o algoritmo converge para uma solução do problema PSI. No entanto, a função é não diferenciável levando-nos a optar por um procedimento que não usa derivadas na sua minimização.

6.3.6 Convergência do algoritmo

Para provar a convergência do algoritmo, são consideradas as seguintes hipóteses (algumas já referidas anteriormente)

(i) As funções f, g e h são contínuas em R^n.

(ii) O conjunto $\{x : x \in F : f(x) \leq \alpha\}$ é limitado para algum α finito.

(iii) O interior da região admissível F^0 é não vazio.

(iv) F é um subconjunto fechado de F^0.

Lema 6.3.2 *Suponha-se que o problema* (6.1.1) *satisfaz as Hipóteses (i)-(iv). Suponha-se ainda que, para cada k, a função* $\bar{\phi}_p(x, \mu^k, \nu^k)$ *tem um minimizante pertencente a* F^0. *Seja* x^k *o minimizante global do problema* (6.3.4)*, então:*

$$\bar{\phi}_P(x^k, \mu^k, \nu^k) \leq \bar{\phi}_P(x^{k+1}, \mu^{k+1}, \nu^{k+1}) \tag{6.3.5}$$

$$\bar{\bar{\theta}}_\infty(x^{k+1}) \leq \bar{\bar{\theta}}_\infty(x^k) \tag{6.3.6}$$

$$f(x^k) \leq f(x^{k+1}). \tag{6.3.7}$$

Prova. Como para todo o k tem-se $0 < \mu^k \leq \mu^{k+1}$, $0 \leq \nu^k \leq \nu^{k+1}$ e x^k é o minimizante global do problema (6.3.4), ou seja,

$$x^k = argmin\ \bar{\phi}_P(x, \mu^k, \nu^k)$$

tem-se que $\bar{\phi}_P(x^k, \mu^k, \nu^k) \leq \bar{\phi}_P(x^{k+1}, \mu^k, \nu^k)$. Então,

$$\begin{aligned}
\bar{\phi}_P(x^k, \mu^k, \nu^k) &= f(x^k) + \mu^k \bar{\bar{\theta}}_\infty(x^k) + \frac{1}{2}\nu^k \bar{\bar{\theta}}^2_\infty(x^k) \\
&\leq f(x^{k+1}) + \mu^k \bar{\bar{\theta}}_\infty(x^{k+1}) + \frac{1}{2}\nu^k \bar{\bar{\theta}}^2_\infty(x^{k+1}) \\
&\leq f(x^{k+1}) + \mu^{k+1} \bar{\bar{\theta}}_\infty(x^{k+1}) + \frac{1}{2}\nu^{k+1} \bar{\bar{\theta}}^2_\infty(x^{k+1}) \\
&= \bar{\phi}_P(x^{k+1}, \mu^{k+1}, \nu^{k+1})
\end{aligned}$$

Portanto,

$$\bar{\phi}_P(x^k, \mu^k, \nu^k) \leq \bar{\phi}_P(x^{k+1}, \mu^{k+1}, \nu^{k+1})$$

Veja-se agora,

$$\begin{aligned}
\bar{\phi}_P(x^k, \mu^k, \nu^k) &= f(x^k) + \mu^k \bar{\bar{\theta}}_\infty(x^k) + \frac{1}{2}\nu^k \bar{\bar{\theta}}^2_\infty(x^k) \\
&\leq f(x^{k+1}) + \mu^k \bar{\bar{\theta}}_\infty(x^{k+1}) + \frac{1}{2}\nu^k \bar{\bar{\theta}}^2_\infty(x^{k+1}).
\end{aligned} \tag{6.3.8}$$

Como x^{k+1} é minimizante de $\bar{\phi}_P(x^{k+1}, \mu^{k+1}, \nu^{k+1})$ então

$$\bar{\phi}_P(x^{k+1}, \mu^{k+1}, \nu^{k+1}) \leq \bar{\phi}_P(x^k, \mu^{k+1}, \nu^{k+1}),$$

e,

$$\begin{aligned}
\bar{\phi}_P(x^{k+1}, \mu^{k+1}, \nu^{k+1}) &= f(x^{k+1}) + \mu^{k+1} \bar{\bar{\theta}}_\infty(x^{k+1}) + \frac{1}{2}\nu^{k+1} \bar{\bar{\theta}}^2_\infty(x^{k+1}) \\
&\leq f(x^k) + \mu^{k+1} \bar{\bar{\theta}}_\infty(x^k) + \frac{1}{2}\nu^{k+1} \bar{\bar{\theta}}^2_\infty(x^k).
\end{aligned} \tag{6.3.9}$$

Somando (6.3.8) a (6.3.9), vem:

$$(\mu^{k+1} - \mu^k)\bar{\bar{\theta}}_\infty(x^{k+1}) + \frac{1}{2}(\nu^{k+1} - \nu^k)\bar{\bar{\theta}}^2_\infty(x^{k+1}) \leq (\mu^{k+1} - \mu^k)\bar{\bar{\theta}}_\infty(x^k) + \frac{1}{2}(\nu^{k+1} - \nu^k)\bar{\bar{\theta}}^2_\infty(x^k). \tag{6.3.10}$$

Passando todos os termos para o primeiro termo da desigualdade, tem-se:

$$(\mu^{k+1}-\mu^k)\left(\bar{\bar{\theta}}_\infty(x^{k+1})-\bar{\bar{\theta}}_\infty(x^k)\right)+\frac{1}{2}(\nu^{k+1}-\nu^k)\left(\bar{\bar{\theta}}^2_\infty(x^{k+1})-\bar{\bar{\theta}}^2_\infty(x^k)\right)\leq 0.$$

Colocando em evidência $\left(\bar{\bar{\theta}}_\infty(x^{k+1})-\bar{\bar{\theta}}_\infty(x^k)\right)$, obtém-se:

$$\left(\bar{\bar{\theta}}_\infty(x^{k+1})-\bar{\bar{\theta}}_\infty(x^k)\right)\left((\mu^{k+1}-\mu^k)+\frac{1}{2}(\nu^{k+1}-\nu^k)\left(\bar{\bar{\theta}}_\infty(x^{k+1})+\bar{\bar{\theta}}_\infty(x^k)\right)\right)\leq 0. \quad (6.3.11)$$

Como $(\mu^{k+1}-\mu^k)\geq 0$ e $(\nu^{k+1}-\nu^k)\geq 0$ e $\bar{\bar{\theta}}_\infty\geq 0$ segue que

$$\bar{\bar{\theta}}_\infty(x^{k+1})\leq\bar{\bar{\theta}}_\infty(x^k).$$

Finalmente, usando (6.3.6) tem-se

$$f(x^k)+\mu^k\bar{\bar{\theta}}_\infty(x^k)+\frac{1}{2}\nu^k\bar{\bar{\theta}}^2_\infty(x^k)\leq f((x^{k+1})+\mu^k\bar{\bar{\theta}}_\infty(x^{k+1})+\frac{1}{2}\nu^k\bar{\bar{\theta}}^2_\infty(x^{k+1})$$

$$\leq f(x^{k+1})+\mu^k\bar{\bar{\theta}}_\infty(x^k)+\frac{1}{2}\nu^k\bar{\bar{\theta}}^2_\infty(x^k),$$

ou seja, $f(x^k)\leq f(x^{k+1})$. ■

Lema 6.3.3 *Seja x^* um minimizante global do problema* (6.1.1)*, então para $k=1,2,\ldots,$ tem-se*

$$f(x^k)\leq\bar{\phi}_P(x^k,\mu^k,\nu^k)\leq f(x^*). \quad (6.3.12)$$

Como consequência, $x^k\in F$ se e só se, é uma solução global do problema (6.1.1).

Prova. Como $\mu^k>0$ e $\nu^k\geq 0$ para todo $x\in R^n$ e x^k é minimizante global de (6.3.4) tem-se:

$$f(x^k)\leq f(x^k)+\mu^k\bar{\bar{\theta}}_\infty(x^k)+\frac{1}{2}\nu^k\bar{\bar{\theta}}^2_\infty(x^k)$$

$$\leq f(x^*)+\mu^k\bar{\bar{\theta}}_\infty(x^*)+\frac{1}{2}\nu^k\bar{\bar{\theta}}^2_\infty(x^*)=f(x^*).$$

■

Teorema 6.3.2 *Suponha-se que o problema* (6.2.6) *satisfaz as Hipóteses (i)-(iv) e seja* $\bar{\bar{\theta}}_\infty$ *contínua. Seja* $\{x^k\}$, $k = 1, \ldots, \infty$ *uma sequência de soluções do problema* (6.3.4)*, gerada pelo Algoritmo 6.3.3 com* $\mu \to \infty$ *e* $\nu \to \infty$*. Então, o ponto limite* $\bar{x}$ *da sequência* $\{x^k\}$ *é uma solução óptima do problema* (6.2.6)*.*

Prova. Seja $\bar{x}$ um ponto limite da sequência $\{x^k\}$, então tem-se que $\lim_{k\to\infty} x^k = \bar{x}$. Pelo Lema 6.3.3

$$q^* := \lim_{k\to\infty} \bar{\phi}_P(x^k, \mu^k, \nu^k) \leq f(x^*).$$

Então,

$$\lim_{k\to\infty} \mu^k \bar{\bar{\theta}}_\infty(x^k) + \frac{1}{2}\nu^k \bar{\bar{\theta}}^2_\infty(x^k) =$$
$$\lim_{k\to\infty} \left[\bar{\phi}_P(x^k, \mu^k, \nu^k) - f(x^k)\right] = q^* - f(\bar{x}).$$

Mas como $\mu^k \to \infty$ e $\nu^k \to \infty$, consequentemente vem:

$$\lim_{k\to\infty} \bar{\bar{\theta}}_\infty(x^k) = 0.$$

Portanto, a partir da continuidade das funções, tem-se que $\bar{\bar{\theta}}_\infty(\bar{x}) = 0$, isto é, $\bar{x}$ é uma solução admissível do problema (6.2.6). Finalmente, pelo Lema 6.3.3 tem-se que $f(x^k) \leq f(x^*)$ para todo k e, portanto, $f(\bar{x}) \leq f(x^*)$, podendo-se concluir que $\bar{x}$ é uma solução óptima de (6.2.6).

■

Capítulo 7

Implementação e resultados computacionais

Neste capítulo são apresentados os detalhes de implementação do algoritmo de redução local baseado na técnica de penalidade. Os resultados obtidos nas experiências computacionais também são aqui apresentados.

Neste trabalho foi implementada uma versão do algoritmo apresentado no Capítulo 6. Na implementação do *solver* algumas decisões como, por exemplo, a escolha do parâmetro $\theta_{crossover}$ ou da função de penalidade basearam-se, para além dos fundamentos teóricos, nos resultados empíricos que vão ser mostrados neste capítulo.

Começa-se por descrever alguns detalhes de implementação do *solver*. Na segunda secção é feita uma referência à base de dados dos problemas teste usados nas experiências computacionais. Na terceira secção são apresentados os resultados de uma análise de sensibilidade, em que se fez variar os parâmetros ϵ, $\theta_{crossover}$e μ nas funções de penalidade ϕ_∞ e ϕ_P. Na quarta secção comparam-se os resultados obtidos pelo *solver* aplicado a diferentes funções de penalidade. Na última secção apresenta-se os resultados do *solver* para problemas de PSI na forma mais geral.

7.1 Detalhes de implementação

O *solver* desenvolvido é designado por SIRedAl (*Semi-Infinite Reduction Algorithm*). Este *solver* foi implementado em MATLAB e é capaz de resolver problemas de PSI na forma mais geral com dimensão infinita máxima de 2 ($p = 2$). O código do *solver* usa dois algoritmos diferentes na minimização da função de penalidade e dois na resolução dos problemas multi-locais.

Resolução dos problemas multi-locais

Os conjuntos $\mathcal{A}_i^k$, $i = 1, \ldots, m$, são formados pelos maximizantes (globais e locais) dos sub-problemas definidos na equação (6.1.2). O código do *solver* está preparado para usar os seguintes *solvers* na resolução destes problemas: MLOCPSOAm ou MLOGAMO. O último `m` em MLOCPSOAm refere-se à versão MATLAB [66].

Sempre que possível estes *solvers* usam como população inicial as aproximações *epsilon* globais determinadas na iteração anterior (o conjunto $\{t \in \mathcal{A}_i^k : g_i(x^k, t) > \epsilon\}$) como aproximações iniciais para resolver o sub-problema i, $i = 1, \ldots, m$.

Minimização da função de penalidade

A minimização dos problemas de penalidade sem restrições pode ser feita usando o *solver* do MATLAB `fminsearch` ou o *solver* PSwarm [131]. O `fminsearch` usa o algoritmo Nelder-Mead enquanto que o PSwarm utiliza um algoritmo híbrido de pesquisa padrão, com colónias de partículas.

A grelha estabilizadora E usada na substituição do conjunto T, juntamente com os conjuntos $\mathcal{A}_i^k$, para $i = 1, \ldots, m$ é estática e é formada por 10 pontos uniformemente distribuídos por dimensão. Por uma questão de implementação são produzidas grelhas até dimensão 2.

7.2 Problemas teste

Os problemas teste pertencem à base de dados SIPAMPL [128]. O SIPAMPL disponibiliza à comunidade científica uma base de dados de problemas de PSI codificados na linguagem de modelação AMPL [22]. A base de dados SIPAMPL contém mais de 160 problemas de PSI de pequena/média dimensão. Foram testados no *solver* SIRedAl todos aqueles que tinham dimensão infinita menor ou igual a 2. No Apêndice A encontra-se uma descrição sumária dos problemas teste usados.

Para além da base de dados, uma outra funcionalidades disponível do pacote SIPAMPL é a interface entre os problemas codificados em AMPL e o MATLAB. O *solver* SIRedAl utiliza rotinas do SIPAMPL para obter e resolver problemas codificados em AMPL.

7.3 Estudo com várias funções de penalidade

Nesta secção o algoritmo de redução baseado na técnica de penalidade é usado com as funções de penalidade ϕ_1, ϕ_2, ϕ_∞ e ϕ_P. Foram usados 39 problemas teste da base de dados SIPAMPL com apenas uma restrição infinita ($m = 1$ e $o = q = 0$), em que a dimensão do espaço infinito foi de 1 e 2 ($p = 1, 2$). Este estudo permitiu comparar a eficiência e robustez do algoritmo com as diferentes funções e decidir por aquela que melhor se adapta aos problemas de PSI. Na minimização dos problemas sem restrições, *i.e.*, na minimização das funções de penalidade foram usados dois *solvers*: o `fminsearch` e o PSwarm. Para cada um deles foi escolhida a função de penalidade cujo algoritmo de redução obteve melhores resultados, tendo pesado nesta decisão o valor da função objectivo, o número de iterações externas, o número de avaliações da função objectivo e o número de avaliações das funções de restrições.

7.3.1 Detalhes de implementação e parâmetros

Actualização dos parâmetros de penalidade

Na actualização dos parâmetros de penalidade da função ϕ_P usou-se o Algoritmo 6.3.2. Nas restantes funções, o parâmetro de penalidade μ foi actualizado da seguinte forma:

$$\mu^{k+1} = 10\mu^k,$$

em que k é o contador da iteração externa.

Problemas multi-locais

Na resolução dos problemas multi-locais usou-se a versão MATLAB do MLOCPSOA. Os valores dos parâmetros neste *solver* estão descritos na Tabela 7.1.

Tabela 7.1: Valores dos parâmetros no MLOCPSOA

Parâmetros	Valor
Tamanho da população (número de pontos ou partículas)	100
Número máximo de avaliações da função objectivo	2000
Número máximo de iterações do algoritmo	2000

Parâmetros do algoritmo de redução

Os parâmetros usados no algoritmo de redução foram os indicados na Tabela 7.2.

7.3.2 Resultados computacionais

Para uma maior compreensão dos resultados optou-se por apresentá-los através do gráfico de perfis de desempenho (*performance profile*) [15] em termos do valor de algumas métricas

Tabela 7.2: Valores dos parâmetros do algoritmo de redução

Parâmetros	Valor
Número máximo de iterações externas	200
$\theta_{crossover}$	0.1
ϵ	0.001

consideradas (nomeadamente, o valor da função função objectivo obtido) usando uma estratégia como em A.I.F. Vaz e L. Vicente [131, 132].

A função usada nos gráficos de perfis de desempenho é chamada função de desempenho (*function profile*) e mede a eficiência e robustez das várias versões do *solver* em termos dos valores obtidos de uma métrica. Para explicar a determinação dos perfis, considere-se $\mathcal{P}$ um conjunto de problemas teste e $\mathcal{S}$ o conjunto com as várias versões do *solver*. Define-se $r_{p,s}$ como sendo o valor da métrica obtida pelo *solver* s na resolução do problema p, em que $p \in \mathcal{P}$ e $s \in \mathcal{S}$. O valor de $r_{p,s}$ é $+\infty$ sempre que há uma falha, *i.e.*, quando um *solver* s não converge para um ponto admissível do problema p ou não é capaz de obter um ponto admissível para o problema p. A função de desempenho $\rho_s(\tau)$ de um *solver* $s \in \mathcal{S}$ define-se como uma fracção de problemas onde o valor médio da métrica é menor do que τ

$$\rho_s(\tau) = \frac{1}{|\mathcal{P}|}|M|,$$

em que $M = \{p \in \mathcal{P} : r_{p,s} < \tau\}$.

Os algoritmos usados na resolução dos problemas multi-locais e na minimização da função de penalidade (quando utilizado o PSwarm) são estocásticos, por esta razão os problemas foram executados 10 vezes em cada *solver*. Procedeu-se a uma análise dos gráficos de perfis de desempenho, comparando as várias versões do algoritmo (que diferem apenas na função de penalidade) para um mesmo *solver*. Nessa análise considerou-se o valor mínimo, médio e máximo da função objectivo, assim como o número de iterações externas, número de avaliações da função objectivo e avaliações das restrições. Nesta secção apresenta-se apenas os gráficos de perfis de desempenho relativos à média da função objectivo nas várias versões do algoritmo para os *solvers* `fminsearch` e o PSwarm, respectivamente. Os

restantes gráficos de perfis de desempenho encontram-se no Apêndice B.

Considere-se o exemplo da Figura 7.1. Uma forma simples de interpretar o gráfico de perfis de desempenho é observá-lo em torno de $\tau = 1$, em que nos é dada a percentagem de problemas, para cada uma das versões, que obtiveram melhores valores médios da função objectivo. Observando a Figura 7.1 e considerando a versão do algoritmo que usou a função de penalidade ϕ_P, pode-se verificar que dos 39 problemas teste, aproximadamente 83% foram resolvidos com melhor valor médio da função objectivo comparados com o valor médio da função objectivo das 4 versões.

Analisando o gráfico de perfis de desempenho quando $\tau \to \infty$ pode-se inferir sobre a robustez do algoritmo para cada versão. A robustez entende-se como a capacidade, para determinados valores dos parâmetros (neste exemplo é considerado o valor médio da função objectivo), de obter uma solução aproximada (mesmo que a solução apresentada esteja afastada do melhor valor da função obtido entre todas as versões consideradas).

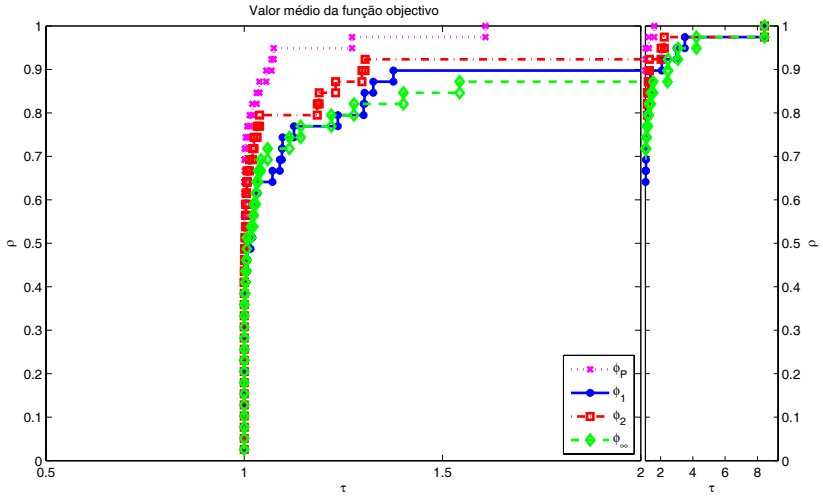


Figura 7.1: Perfis de desempenho em termos da média dos valores da função objectivo usando `fminsearch`.

Observando a Figura 7.1 verifica-se que o algoritmo que usa a função de penalidade ϕ_P é o mais robusto.

O gráfico da Figura 7.2 mostra os resultados obtidos usando a função `fminsearch` na

minimização das funções de penalidade. Observando a figura verifica-se que as curvas dos perfis de desempenho das 4 versões do algoritmo estão muito próximas em termos do valor médio da função objectivo, significando que os algoritmos são semelhantes em eficiência e robustez.

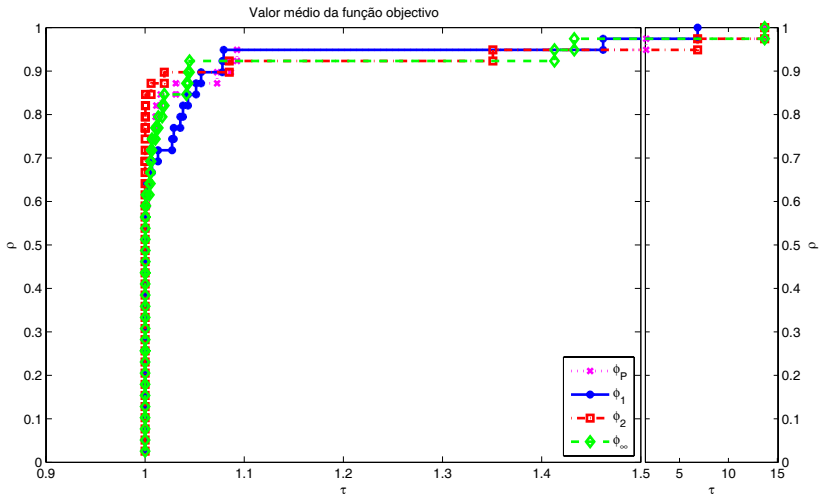


Figura 7.2: Perfis de desempenho em termos da média dos valores da função objectivo usando o PSwarm.

Fez-se um estudo análogo aos gráficos de perfis de desempenho relativos ao valor mínimo e ao máximo dos valores da função objectivo e aos número mínimo, médio e máximo do número de iterações externas, do número de avaliações da função objectivo e do número avaliações das restrições (Apêndice B). Constatou-se que nas versões em que o `fminsearch` foi usado para a minimização da função de penalidade, a função ϕ_P é mais eficiente e robusta. Nas versões do PSwarm, não foram verificadas discrepâncias significativas entre as diferentes funções de penalidade. As várias versões apresentam eficiência e robustez muito semelhantes.

Com base na análise realizada aos gráficos de perfis de desempenho optou-se pelas versões que utilizam a função de penalidade ϕ_P, em ambos os *solvers*. O gráfico de perfis de desempenho da Figura 7.3 compara os valores médios da função objectivo dos algoritmos entre os *solvers* `fminsearch` e PSwarm.

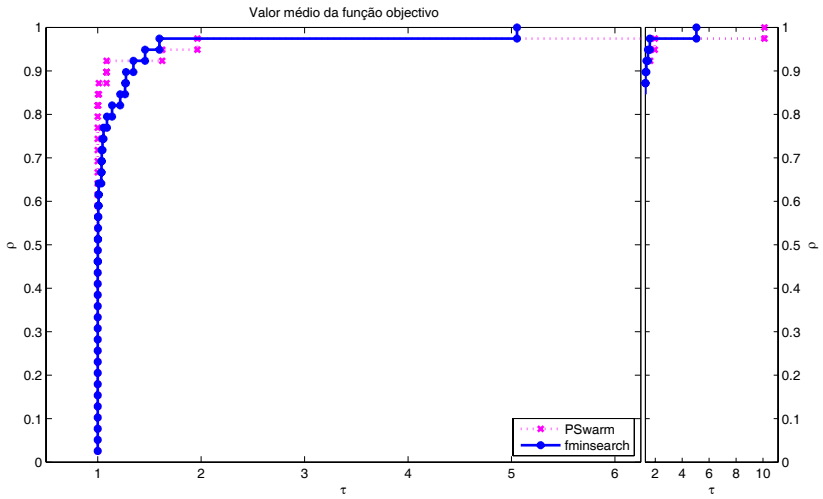


Figura 7.3: Comparação entre a média dos valores da função objectivo usando PSwarm *vs* `fminsearch` na minimização de ϕ_P.

Embora os algoritmos sejam equivalentes em termos de eficiência, quando avaliados no valor médio da função objectivo, como se confirma na Figura 7.3, a versão do `fminsearch` mostra-se mais robusta. Nos restantes parâmetros analisados a versão `fminsearch` é mais eficiente e robusta, exceptuando no número de avaliações da função objectivo, em que a versão PSwarm faz mais avaliações. Esta diferença pode ser justificada pelo facto do PSwarm ser um *solver* mais vocacionado para optimização global.

7.3.3 Conclusões

Este estudo permitiu destrinçar a função de penalidade que torna o algoritmo de redução mais eficiente e robusto quando são só consideradas restrições infinitas. Tendo sido com a função de penalidade ϕ_P que o algoritmo auferiu melhores resultados na resolução dos problemas de PSI.

7.4 Análise de sensibilidade

Nesta secção são apresentados os resultados de uma análise de sensibilidade aplicada a duas funções de penalidade, ϕ_∞ e ϕ_P. Foram usadas estas funções porque são muito próximas e permitindo analisar o comportamento da função de penalidade com mais um parâmetro. Na análise fez-se variar os parâmetros ϵ, $\theta_{crossover}$ e μ_0. Para o efeito foram usados 55 problemas teste da base de dados SIPAMPL, com uma restrição infinita e com variável infinita de dimensão 1 e 2.

7.4.1 Detalhes de implementação

Na actualização dos parâmetros de penalidade da função ϕ_P seguiu-se o Algoritmo 6.3.2. A expressão de actualização do parâmetro de penalidade da função ϕ_∞ foi a seguinte:

$$\mu^{k+1} = n\mu^k$$

em que k é o contador da iteração externa e n o número de variáveis finitas.

Na minimização das funções de penalidade utilizou-se a função `fminsearch` do MATLAB. Os problemas multi-locais foram resolvidos conforme a descrição apresentada na Secção 7.3.1.

7.4.2 Variação dos parâmetros

Os parâmetros escolhidos para análise de sensibilidade foram o ϵ, $\theta_{crossover}$ e μ_0. Em cada execução (versão) fez-se variar pelo menos um desses parâmetros, perfazendo um total de 5 combinações diferentes. A Tabela 7.3 mostra os valores dos parâmetros para cada execução.

Observando a primeira linha da Tabela 7.3, interpreta-se que na execução 1 para cada uma das funções de penalidade, o erro relativo é $\epsilon = 10^{-5}$, o parâmetro $\theta_{crossover} = 0.1$ e o valor inicial do parâmetro de penalidade $\mu_0 = \|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$.

Tabela 7.3: Variação dos parâmetros

	ϵ	$\theta_{crossover}$	μ_0
1	10^{-5}	0.1	$\|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$
2	10^{-5}	0.1	$10 \times \|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$
3	10^{-5}	0.1	$10000 \times \|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$
4	10^{-3}	0.1	$\|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$
5	10^{-5}	0.2	$\|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$

O valor inicial do parâmetro de penalidade ν da função ϕ_P, em todas as execuções foi 1.

7.4.3 Resultados computacionais

Os resultados são apresentados por gráficos de perfis de desempenho em termos do valor da função objectivo. No gráfico da Figura 7.4 é comparado o valor da função objectivo das várias versões quando usada a função de penalidade ϕ_∞.

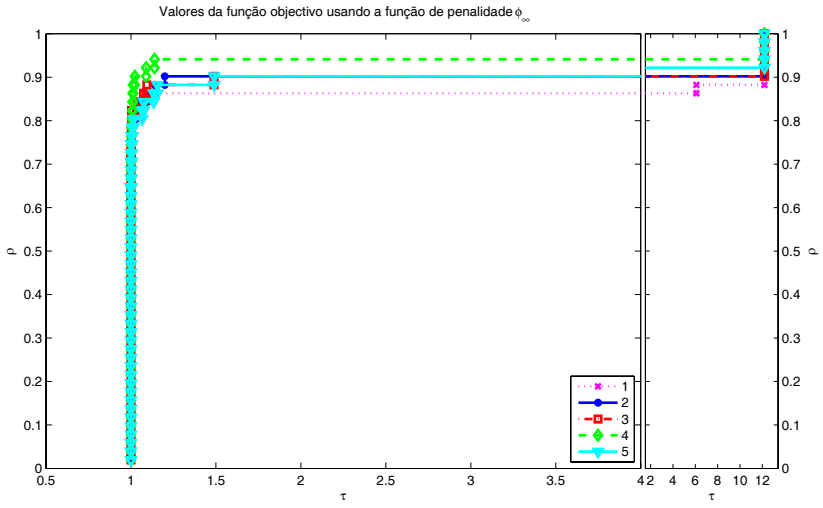


Figura 7.4: Valor da função objectivo usando a função ϕ_∞.

Observando a Figura 7.4 verifica-se que as curvas do gráfico são praticamente coincidentes. No entanto, a curva da execução 4 está ligeiramente acima das restantes. Podendo-se

inferir que esta versão será mais eficiente e robusta do que as outras.

O mesmo estudo foi feito usando a função de penalidade ϕ_p. O gráfico de perfis de desempenho da Figura 7.5 mostra os resultados obtidos.

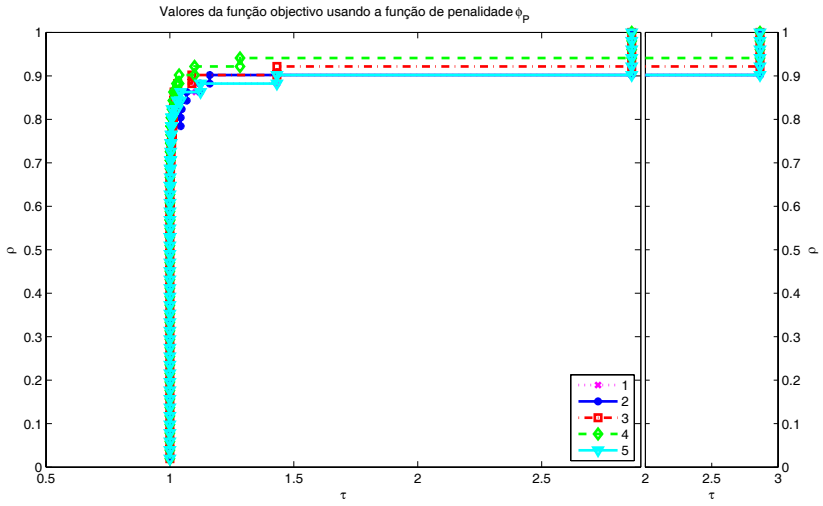


Figura 7.5: Valor da função objectivo usando a função ϕ_p.

As conclusões que se tiram observando a Figura 7.5 são análogas às anteriores.

Foram também analisados os gráficos de perfis de desempenho para cada uma das funções de penalidade, em termos do número de iterações externas, do número de avaliações da função objectivo e do número de avaliações das restrições. Podendo-se concluir que o algoritmo na versão 4 é mais eficiente e robusto independentemente da função de penalidade usada. Os gráficos podem ser consultados no Apêndice C.

Ainda neste âmbito, foram comparados os resultados do algoritmo de redução na versão 4 com as duas funções de penalidade. O gráfico de perfis de desempenho obtido em termos da valor da função objectivo é o da Figura 7.6.

O algoritmo com a função ϕ_P é mais eficiente, no entanto o da ϕ_∞ é mais robusto.

Observe-se agora o gráfico em termos do número de iterações externas representado da Figura 7.7.

Verifica-se que em $\tau = 1$, o algoritmo com a função de penalidade ϕ_P resolve aproximadamente 60% dos problemas com melhor número de iterações. Estando a função ϕ_∞

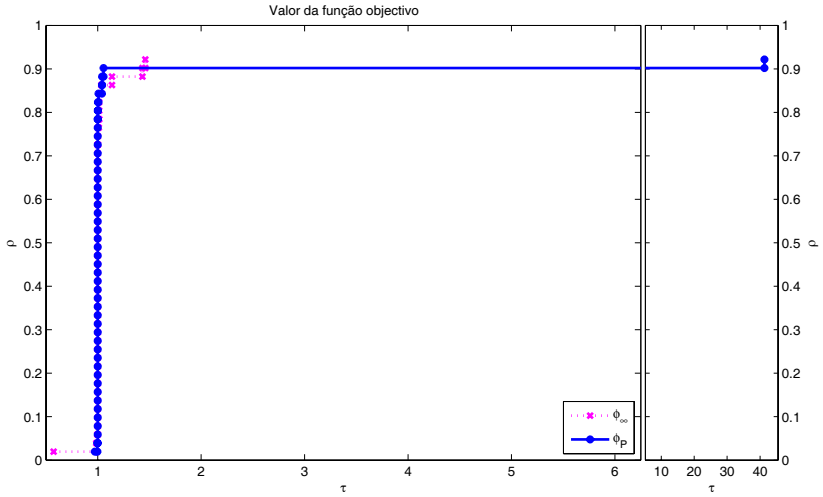


Figura 7.6: Comparação do valor da função objectivo entre ϕ_∞ *vs* ϕ_P (versão 4).

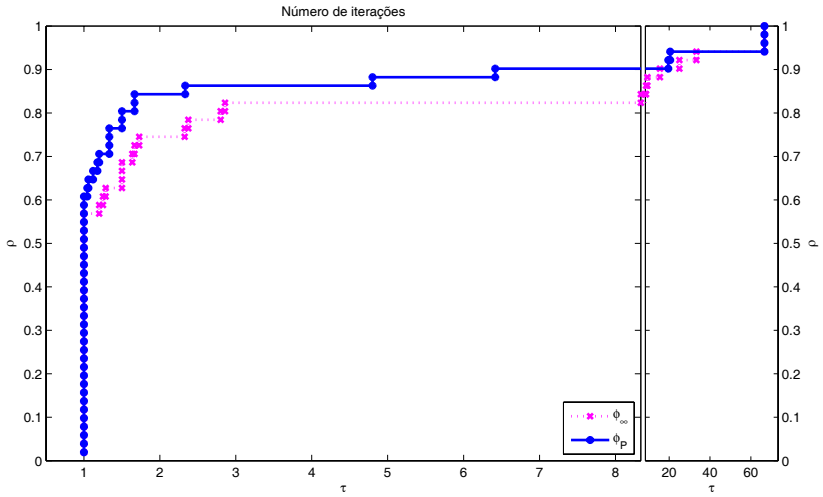


Figura 7.7: Comparação do número de iterações externas entre ϕ_∞ *vs* ϕ_P (versão 4).

ligeiramente abaixo. Podendo-se concluir que relativamente ao número de iterações externas o algoritmo com a função ϕ_P é mais eficiente. Acompanhando a curva quando $\tau \to \infty$ conclui-se ainda que o algoritmo da ϕ_P é também mais robusto.

Analise-se os resultados relativos ao número de avaliações da função objectivo e das restrições representados nas Figuras 7.8 e 7.9.

Constata-se que em $\tau = 1$ é resolvida a mesma percentagem de problemas em ambos

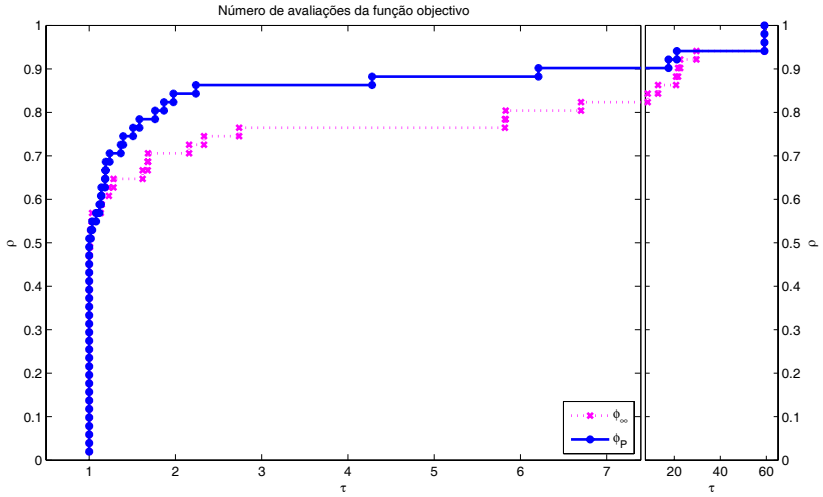


Figura 7.8: Comparação do número de avaliações da função objectivo entre ϕ_∞ *vs* ϕ_P (versão 4).

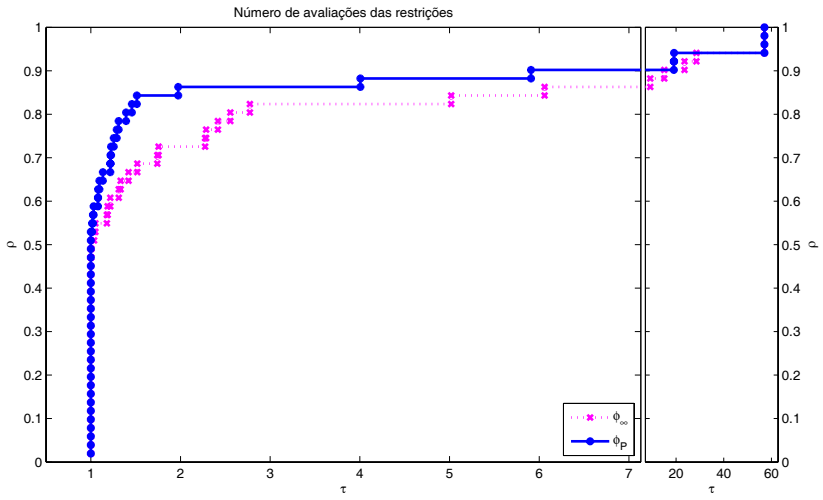


Figura 7.9: Comparação do número de avaliações das restrições entre ϕ_∞ *vs* ϕ_P (versão 4).

os algoritmos. No entanto, para valores próximos de $\tau = 1$ o algoritmo da função de penalidade ϕ_P é mais eficiente. O mesmo acontece quanto à robustez. O algoritmo da função de penalidade ϕ_P é melhor quando considerado o número de avaliações das funções e o número de avaliações das restrições.

7.4.4 Conclusões

A variação dos parâmetros ϵ, $\theta_{crossover}$ e μ_0 no algoritmo implementado para as funções de penalidade ϕ_∞ e ϕ_P permitiu analisar experimentalmente o impacto desta variação no que respeita ao valor da função objectivo, ao número de iterações externas, ao número de avaliações da função e ao número de avaliações das restrições. Podendo-se concluir que a variação dos parâmetros não influencia significativamente o resultado em termos de valor da função objectivo. No entanto, o mesmo não se passa quando contabilizado o número de iterações externas, avaliações da função objectivo e avaliações das restrições. O algoritmo de redução na versão 4, em que $\epsilon = 10^{-3}$, $\theta_{crossover} = 0.1$ e $\mu_0 = \|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$ mostrou-se mais eficiente e robusto, quando consideradas as métricas referidas anteriormente, traduzindo-se numa diminuição dos cálculos computacionais na resolução dos problemas. Este comportamento verificou-se para ambas as funções de penalidade. O algoritmo com a função ϕ_P mostrou-se mais eficiente e robusto relativamente ao algoritmo da função ϕ_∞, quando comparados na versão 4.

7.5 O SIRedAl aplicado a problemas de PSI na forma mais geral

Esta secção é dedicada aos resultados obtidos com o SIRedAl para problemas de PSI na forma mais geral. Na resolução dos problemas multi-locais foram usados dois *solvers*: MLOCPSOA e MLOCGAMO. É feita uma comparação entre resultados obtidos pelo SIRedAl (utilizando os diferentes *solvers*) com as soluções conhecidas na literatura. Para o efeito, foram usados 117 problemas teste, 5 dos quais contêm restrições finitas.

7.5.1 Detalhes de implementação e parâmetros

Os resultados numéricos foram obtidos utilizando a função `fminsearch` do MATLAB nas iterações internas. A preferência da função `fminsearch` é justificada pelos resultados ob-

tidos no estudo apresentado na Secção 7.3.

Na resolução dos problemas multi-locais foram utilizados os *solvers* MLOCPSOA e MLOCGAMO. Os parâmetros considerados nesses algoritmos estão representados Tabela 7.1.

Os valores dos parâmetros do Algoritmo 6.3.2 são os da Tabela 7.2.

7.5.2 Resultados computacionais na versão com o MLOCPSOA

Cada problema foi executado 10 vezes, uma vez que a resolução dos sub-problemas (6.1.2) envolve o uso de algoritmos estocásticos. Para cada execução, comparou-se as soluções obtidas com as soluções conhecidas na bibliografia e considerou-se sucesso quando o erro relativo entre as soluções não excedia 5%. Na Tabela 7.4 estão indicados o número de sucessos obtidos (coluna "Suc.") para cada problema.

Observando a Tabela 7.4 verifica-se que em 62% dos problemas o número de sucessos ocorreu em mais de metade das execuções. Para os restantes casos, há a salientar que para o problema hettich11 o algoritmo não convergiu em nenhuma das execuções. O problema hettich11 é definido da seguinte forma:

$$\begin{aligned} &\min_{x\in R^2} \ -(x_2+5)^2 - x_1^2 \\ &s.a \quad 2t + x_1 + x_2 - t^2 \leq 0, \ t \in [-1,1]. \end{aligned} \tag{7.5.1}$$

Sendo o problema sem restrições associado ao (7.5.1) definido como:

$$\min \phi = -(x_2+5)^2 - x_1^2 + \mu \max\{0, 2t + x_1 + x_2 - t^2\} + \frac{1}{2}\nu^2 \max\{0, 2t + x_1 + x_2 - t^2\}.$$

Analisando esta função de penalidade constata-se que o termo da função objectivo é quadrático em x, enquanto que o das restrições é linear. Nesta situação, a violação das restrições nunca poderá compensar o termo quadrático da função objectivo, e por conseguinte, está-se perante uma função de penalidade ilimitada. Justificando deste forma a não convergência do algoritmo.

Para os problemas apresentados na Tabela 7.5 não são conhecidas as soluções na bibliografia. Nesta tabela está representada a média do valor da função objectivo para a solução encontrada com uma não admissibilidade máxima de 5×10^{-3} das restrições infinitas.

7.5.3 Resultados computacionais na versão com o MLOCGAMO

Nesta experiência a resolução dos problemas multi-locais fez-se com o *solver* MLOCGAMO na versão bi-objectivo (é a por defeito do *solver*). Foi usado o mesmo conjunto de problemas teste e procedeu-se como na versão com o MLOCPSOA. Na Tabela 7.6 são apresentados o número de sucessos obtidos.

Observando a Tabela 7.6 verifica-se que em 52% dos problemas teste o número de sucessos foi igual ou superior a 5. Nos problemas watson9 e leon21 não houve sucessos porque as soluções obtidas pelo *solver* tinham uma não admissibilidade superior a 5×10^{-3}. Contudo, a média dos valores da função objectivo encontrados foi de -99.6677 e -10.3832, sendo as soluções conhecidas na bibliografia -99.6607 e -12, respectivamente. Para os problemas lobianco1 e li1 o algoritmo encontrou valores muito elevados da função objectivo, em todas as execuções. No problema watson13 o algoritmo divergiu logo na primeira iteração e para o hettich11, devido à função ser ilimitada o algoritmo não convergiu em nenhuma das execuções.

Os resultados apresentados na Tabela 7.7 são a média do valor da função objectivo para a solução encontrada com uma não admissibilidade máxima de 5×10^{-3} das restrições infinitas.

Comparando os resultados da Tabela 7.7 com os da Tabela 7.5 tem-se que em 40% dos problemas os resultados têm um erro relativo inferior a 5×10^{-3}.

7.5.4 Conclusões

A utilização dos 117 problemas teste da base de dados SIPAMPL, com dimensão infinita máxima 2, permitiu testar a potencialidade do *solver* SiReDAL. Obteve-se resultados em todos os problemas na versão com o MLOCPSOA na resolução dos problemas multi-locais, exceptuando para o problema hettich11. Este problema produz uma função de penalidade ilimitada, o que origina a não convergência do algoritmo. Comparou-se os resultados obtidos com as soluções conhecidas na literatura e verificou-se que em 62% dos problemas obteve-se sucesso em mais de metade das execuções. Foram obtidos resultados também para 38 problemas cujas soluções não estão disponíveis na literatura.

Repetiu-se a experiência usando o MLOCGAMO na resolução dos problemas multi-locais. Em 52% dos problemas teste obteve-se sucesso em mais de metade das execuções, comparando os resultados obtidos com as soluções da literatura. Para além do problema hettich11, nesta versão o algoritmo também não convergiu para os problemas lobianco1, li1 e watson13. Foram obtidos resultados para problemas em que não é conhecida a solução na bibliografia. Para estes problemas teste comparou-se os resultados entre as duas versões e verificou-se que em 40% dos problemas o erro relativo entre as soluções foi inferior a 5×10^{-3}.

Tabela 7.4: Problemas/Nr. de Sucessos obtidos na versão MLOCPSOA

Problema	Suc.	Problema	Suc.	Problema	Suc.	Problema	Suc.
andreson1	10	hettich11	0	leon2	0	matlab2	0
blankenship1	10	hettich2	10	leon20	10	priceK	10
coopeL	10	hettich5	10	leon21	10	reemtsen2	10
coopeM	10	hettich6	0	leon22	10	reemtsen3	0
coopeN	10	hettich7	0	leon23	10	reemtsen4	0
deluca1	10	hettich9	0	leon3	0	watson1	10
deluca2	10	honstede1	8	leon4	0	watson10	10
fang1	10	kortanek1	10	leon5	0	watson11	10
fang2	6	kortanek2	10	leon6	0	watson12	0
fang3	7	kortanek3	5	leon7	0	watson13	3
ferris1	0	kortanek4	10	leon8	0	watson2	10
ferris2	10	leon1	0	leon9	0	watson3	10
goerner1	3	leon10	10	li1	0	watson4a	10
goerner2	0	leon11	10	li2	0	watson4c	10
goerner3	0	leon12	10	lin2	10	watson6	10
goerner4	0	leon13	10	liu1	10	watson7	10
goerner5	1	leon14	10	liu2	10	watson8	9
goerner6	0	leon15	10	liu3	7	watson9	10
goerner7	0	leon16	0	lobianco1	10	zhou1	10
hettich10c	10	leon17	10	matlab1	0		

Tabela 7.5: Problemas e média da função objectivo da solução obtida na versão MLOCPSOA

Problema	Média	Problema	Média	Problema	Média	Problema	Média
gugat1	0.0427	gugat5b	0.3972	hettich13	-2.2355	powell1	-2.9999
gugat2	0.1876	gugat5c	0.2267	hettich14	-2.1212	still1	1
gugat3	0	gugat5d	0.9662	hettich4	1	tanaka1	-0.9999
gugat4a	3.2041	gugat5e	0.0642	hettich8	0.0832	userman	0
gugat4b	0.3261	gugat5f	0.0108	leon18	-1.75	vaz1	3665.7589
gugat4c	0.0417	gugat6	0.1103	leon19	0.7952	vaz2	0.0022
gugat4d	0.9354	gugat7	0.5961	leon24	-9.4022	vaz3	6.7223
gugat4e	1.3726	hettich1	0.1096	lin1	-1.8156	vaz4	0.0005
gugat4f	2.1077	hettich10	1	polak1	5.4450		
gugat5a	0.0325	hettich12	2.459	polak2	6.1984		

Tabela 7.6: Problemas/Nr. de Sucessos obtidos na versão com o MLOCGAMO

Problema	Suc.	Problema	Suc.	Problema	Suc.	Problema	Suc.
andreson1	10	hettich11	0	leon2	0	matlab2	0
blankenship1	10	hettich2	9	leon20	10	priceK	1
coopeL	1	hettich5	10	leon21	0	reemtsen2	0
coopeM	10	hettich6	0	leon22	10	reemtsen3	1
coopeN	10	hettich7	0	leon23	10	reemtsen4	0
deluca1	5	hettich9	0	leon3	0	watson1	10
deluca2	0	honstede1	0	leon4	4	watson10	10
fang1	5	kortanek1	8	leon5	0	watson11	10
fang2	0	kortanek2	10	leon6	0	watson12	10
fang3	8	kortanek3	7	leon7	0	watson13	0
ferris1	0	kortanek4	10	leon8	0	watson2	0
ferris2	10	leon1	0	leon9	0	watson3	10
goerner1	10	leon10	10	li1	0	watson4a	10
goerner2	0	leon11	8	li2	0	watson4c	10
goerner3	0	leon12	5	lin2	3	watson6	10
goerner4	0	leon13	10	liu1	10	watson7	10
goerner5	1	leon14	10	liu2	10	watson8	10
goerner6	0	leon15	10	liu3	6	watson9	10
goerner7	0	leon16	0	lobianco1	0	zhou1	10
hettich10c	10	leon17	10	matlab1	0		

Tabela 7.7: Problemas e média da função objectivo da solução obtida na versão com o MLOC-GAMO

Problema	Média	Problema	Média	Problema	Média	Problema	Média
gugat1	0.0432	gugat5b	0.3225	hettich13	-2.25	powell1	-1.0515
gugat2	0.1620	gugat5c	0.0540	hettich14	-2.1375	still1	1
gugat3	0	gugat5d	43.4083	hettich4	1	tanaka1	-0.9999
gugat4a	0.0735	gugat5e	0.0641	hettich8	0.0183	userman	0
gugat4b	0.3156	gugat5f	0.0455	leon18	-1.75	vaz1	3666.9402
gugat4c	0.0623	gugat6	0.0814	leon19	0.7969	vaz2	0.0003
gugat4d	45.6244	gugat7	0.4768	leon24	-9.0771	vaz3	6.7194
gugat4e	3.3473	hettich1	0.1493	lin1	-1.8126	vaz4	0.0005
gugat4f	17.0286	hettich10	1	polak1	5.4456		
gugat5a	0.0164	hettich12	4.4011	polak2	6.2024		

Capítulo 8

Conclusões e trabalho futuro

Neste capítulo são apresentadas as conclusões que se podem retirar do projecto desenvolvido. Igualmente, são apresentadas algumas sugestões úteis a concretizar no futuro.

8.1 Conclusões

Neste trabalho foi proposto um algoritmo de redução local baseado na técnica de penalidade para problemas de PSI na forma mais geral. Na resolução dos problemas finitos usou-se a técnica de penalidade de pontos exteriores. Realizou-se um estudo com várias funções de penalidade e optou-se pela função de norma-∞ aumentada, a função de penalidade ϕ_P. A pretensão de resolver problemas na forma geral conduziu-nos à extensão desta função que passou a designar-se por $\bar{\phi}_P$. Esta função tem a vantagem de ser exacta e contínua, no entanto é não diferenciável, condicionando-nos a optar por algoritmos que não utilizam derivadas na sua resolução. As propriedades teóricas da função de penalidade, assim como a convergência do algoritmo também foram estudadas.

O algoritmo proposto foi implementado no *solver* SIReDAL. O código do programa está preparado para usar dois algoritmos diferentes na minimização da função de penalidade `fminsearch` ou PSwarm. Permite também que os problemas multi-locais possam ser resolvidos pelos *solvers* MLOCPSOA ou MLOCGAMO.

O SiReDAL foi testado com 117 problemas da base de dados SIPAMPL, com dimensão infinita máxima de 2. Usou-se dois *solver* MLOCPSOA e MLOCGAMO na resolução dos problemas multi-locais. Na versão em que foi utilizado o MLOCPSOA obteve-se resultados numéricos em 116 problemas teste. O algoritmo não obteve resultados apenas num problema cuja a função de penalidade é ilimitada. Na versão com o MLOCGAMO o *solver* obteve resultados em 97% dos problemas. Os resultados numéricos confirmaram a potencialidade do algoritmo proposto.

8.2 Trabalho futuro

Na versão actual, o *solver* SiReDAL tem a capacidade de resolver problemas apenas com dimensão infinita máxima de 2. Uma melhoria possível seria torná-lo mais abrangente, no sentido de resolver também problemas com dimensão infinita superior a 2. Seria também interessante testá-lo com problemas que incluam mais restrições finitas.

Será desejável no futuro fazer a comparação do algoritmo proposto com outro *software* existente, por exemplo, com o *solver fseminf* do MATLAB.

Dada a potencialidade do algoritmo, será bastante útil a sua disponibilização ao domínio público.

Apêndices

Apêndice A

Descrição dos problemas da base de dados do SIPAMPL

Nas tabelas seguintes estão descritos os problemas teste usados nos estudos realizados ao longo deste trabalho. Na primeira coluna da tabela ('Nome do Problema') estão os nomes dos problemas teste. A segunda e terceira colunas ('Função Objectivo' e 'Funções de Restrição') correspondem ao tipo de função objectivo e das funções restrição, respectivamente. A quarta ('Var.') indica o número de variáveis do problema. A quinta e sexta colunas ('R.Desig.' e 'R.Igual.') referem-se ao número de restrições finitas de desigualdade e igualdade, respectivamente. A sétima coluna ('R.Inf.') tem o número de restrições infinitas e a última ('dim(T)') refere-se à dimensão do espaço infinito.

Nome do Problema	Função Objectivo	Funções de Restrição	Var. (n)	R.Desig. (m)	R.Igual. (o)	R.Inf. (q)	dim(T) (p)
andreson1	Linear	Linear	3	0	0	1	2
blankenship1	Linear	Generic	2	0	0	1	1
coopeL	Quadratic	Linear	2	0	0	1	1
coopeM	Quadratic	Linear	2	1	0	1	1
coopeN	Linear	Quadratic	2	0	0	1	1
deluca1	Linear	General	7	0	0	8	1
deluca2	Linear	General	7	0	0	8	1
fang1	Linear	Linear	50	0	0	1	1
fang2	Linear	Linear	50	0	0	1	1
fang3	Linear	Linear	50	0	0	1	1
ferris1	Linear	Linear	7	0	0	2	1
ferris2	Linear	Linear	7	0	0	1	1
goerner1	Linear	Generic	4	0	0	2	1
goerner2	Linear	Quadratic	5	0	0	2	1
goerner3	Linear	Generic	7	0	0	2	1
goerner4	Linear	Linear	7	0	0	2	2
goerner5	Linear	Linear	7	0	0	2	2
goerner6	Linear	Linear	16	0	0	2	2
goerner7	Linear	Generic	8	0	0	2	2
gugat1	Linear	Linear	9	0	0	4	1
gugat2	Linear	Linear	9	0	0	4	1
gugat3	Linear	Linear	7	2	0	2	1
gugat4a	Linear	Linear	9	0	0	4	1
gugat4b	Linear	Linear	9	0	0	4	1
gugat4c	Linear	Linear	9	0	0	4	1
gugat4d	Linear	Linear	9	0	0	4	1
gugat4e	Linear	Linear	9	0	0	4	1
gugat4f	Linear	Linear	9	0	0	4	1
gugat5a	Linear	Linear	7	0	0	4	1
gugat5b	Linear	Linear	7	0	0	4	1

Nome do Problema	Função Objectivo	Funções de Restrição	Var. (n)	R.Desig. (m)	R.Igual. (o)	R.Inf. (q)	dim(T) (p)
gugat5c	Linear	Linear	7	0	0	4	1
gugat5d	Linear	Linear	7	0	0	4	1
gugat5e	Linear	Linear	7	0	0	4	1
gugat5f	Linear	Linear	7	0	0	4	1
gugat6	Linear	Generic	6	0	0	4	1
gugat7	Linear	Generic	4	0	0	4	1
hettich1	Linear	Linear	9	0	0	2	2
hettich10	Linear	Quadratic	2	0	0	2	1
hettich10c	Linear	Quadratic	2	0	0	2	1
hettich11	Linear	Quadratic	2	0	0	1	1
hettich12	Linear	Linear	16	0	0	2	2
hettich13	Linear	Linear	2	0	0	1	2
hettich14	Linear	Linear	2	1	0	1	2
hettich2	Linear	Linear	3	0	0	2	1
hettich4	Linear	Linear	2	0	0	2	1
hettich5	Linear	Linear	3	0	0	2	2
hettich6	Linear	Linear	7	0	0	2	2
hettich7	Linear	Linear	7	0	0	2	2
hettich8	Linear	Linear	5	0	0	2	1
hettich9	Quadratic	Linear	11	0	0	2	2
honstede1	Linear	Linear	3	0	0	1	2
kortanek1	Linear	Linear	2	0	0	1	1
kortanek2	Linear	Linear	2	0	0	1	2
kortanek3	Linear	Linear	7	0	0	1	1
kortanek4	Linear	Linear	8	0	0	1	1
leon1	Linear	Linear	4	0	0	2	1
leon10	Linear	Linear	3	0	0	2	1
leon11	Linear	Linear	3	0	0	2	1
leon12	Linear	Linear	2	0	0	1	1
leon13	Linear	Linear	2	0	0	1	1

Nome do Problema	Função Objectivo	Funções de Restrição	Var. (n)	R.Desig. (m)	R.Igual. (o)	R.Inf. (q)	dim(T) (p)
leon14	Linear	Linear	2	0	0	1	1
leon15	Linear	Linear	2	0	0	1	1
leon16	Linear	Linear	3	0	0	1	1
leon17	Linear	Linear	3	0	0	1	1
leon18	Linear	Linear	2	0	0	1	1
leon19	Linear	Linear	5	0	0	1	1
leon2	Linear	Linear	6	0	0	2	1
leon20	Linear	Linear	2	0	0	1	1
leon21	Linear	Linear	2	0	0	2	1
leon22	Linear	Linear	2	0	0	1	1
leon23	Linear	Linear	3	0	0	4	1
leon24	Linear	Linear	4	0	0	5	1
leon3	Linear	Linear	6	0	0	2	1
leon4	Linear	Linear	7	0	0	2	1
leon5	Linear	Linear	8	0	0	2	1
leon6	Linear	Linear	5	0	0	2	1
leon7	Linear	Linear	5	0	0	2	1
leon8	Linear	Linear	7	0	0	2	1
leon9	Linear	Linear	7	0	0	2	1
li1	Quadratic	Generic	10	0	0	1	1
li2	Quadratic	Generic	6	0	0	1	1
lin1	Linear	Linear	6	0	0	1	2
lin2	Linear	Polynomial	9	0	0	36	1
liu1	Quadratic	Linear	2	0	0	1	1
liu2	Quadratic	Linear	2	0	0	1	1
liu3	Quadratic	Linear	16	0	0	2	1
lobianco1	Linear	General	11	0	0	8	1
matlab1	Quadratic	Generic	3	0	0	2	1

Nome do Problema	Função Objectivo	Funções de Restrição	Var. (n)	R.Desig. (m)	R.Igual. (o)	R.Inf. (q)	dim(T) (p)
matlab2	Quadratic	Generic	3	0	0	1	2
polak1	Linear	Quadratic	4	0	0	2	2
polak2	Linear	Generic	4	0	0	2	2
powell1	Linear	Linear	2	0	0	1	1
priceK	Quadratic	Linear	2	0	0	1	1
reemtsen2	Linear	Linear	10	0	0	2	2
reemtsen3	Linear	Linear	10	0	0	2	2
reemtsen4	Linear	Linear	37	0	0	2	2
still1	Linear	Linear	2	0	0	1	1
tanaka1	Linear	Quadratic	2	1	0	1	1
userman	Quadratic	Linear	2	2	0	1	1
vaz1	Linear	Generic	10	0	0	1	2
vaz2	Linear	Linear	1	0	0	1	2
vaz3	Linear	Linear	3	0	0	1	2
vaz4	Linear	Linear	10	0	0	1	2
watson1	Quadratic	Quadratic	2	0	0	1	1
watson10	Linear	Linear	3	0	0	1	2
watson11	Linear	Linear	3	0	0	1	2
watson12	Polynomial	Linear	3	0	0	1	2
watson13	Polynomial	Linear	3	0	0	1	2
watson2	Quadratic	Polynomial	2	0	0	1	1
watson3	Quadratic	Generic	3	0	0	1	1
watson4a	Linear	Quadratic	3	0	0	1	1
watson4c	Linear	Polynomial	8	0	0	1	1
watson6	Polynomial	Generic	2	0	0	1	1
watson7	Quadratic	Linear	3	0	0	1	2
watson8	Linear	Linear	6	0	0	1	2
watson9	Linear	Linear	6	0	0	1	2
zhou1	Linear	Linear	2	0	0	1	1

Apêndice B

Resultados com várias funções de penalidade

Neste apêndice apresentam-se os gráficos de perfis de desempenho relativos ao valor máximo e mínimo da função objectivo, ao maior, médio e menor número de iterações externas, avaliações da função objectivo e avaliações das restrições. Nestes gráficos são comparadas 4 versões do algoritmo de redução, que diferem na função de penalidade para um mesmo *solver* usado na resolução do problema sem restrições (foram usados 2 *solvers*). Mostram-se também os resultados do algoritmo com a função de penalidade ϕ_P em termos das médias dos parâmetros enunciados anteriormente, quando usados os *solvers* `fminsearch` e PSwarm na resolução da função de penalidade.

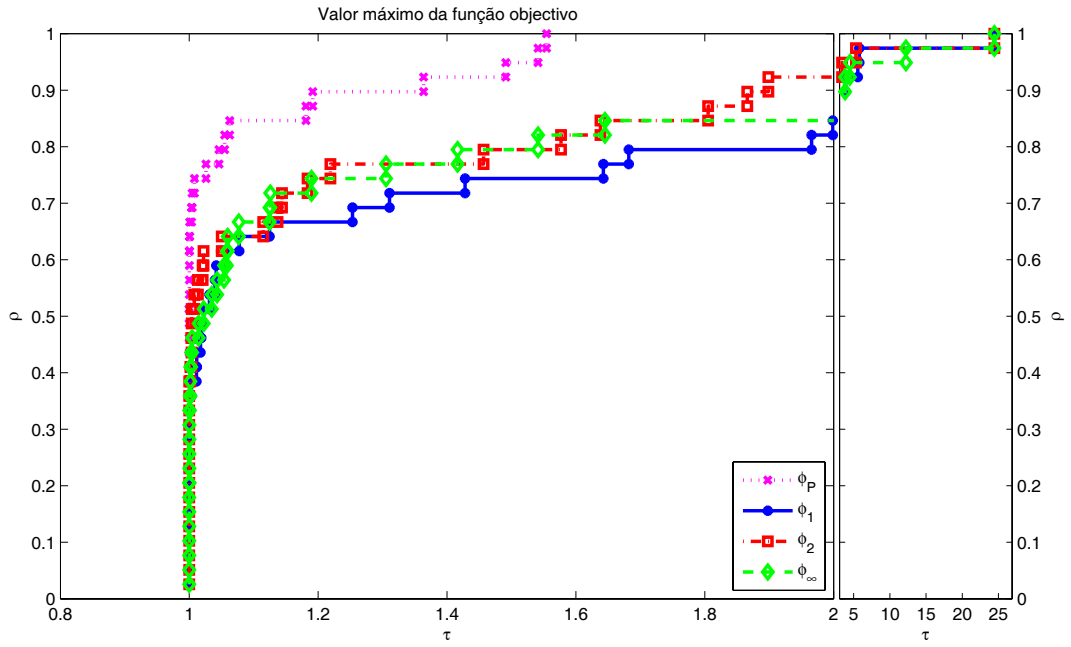


Figura B.1: Gráfico de desempenho de perfis relativo ao valor máximo obtido da função objectivo usando a função `fminsearch`.

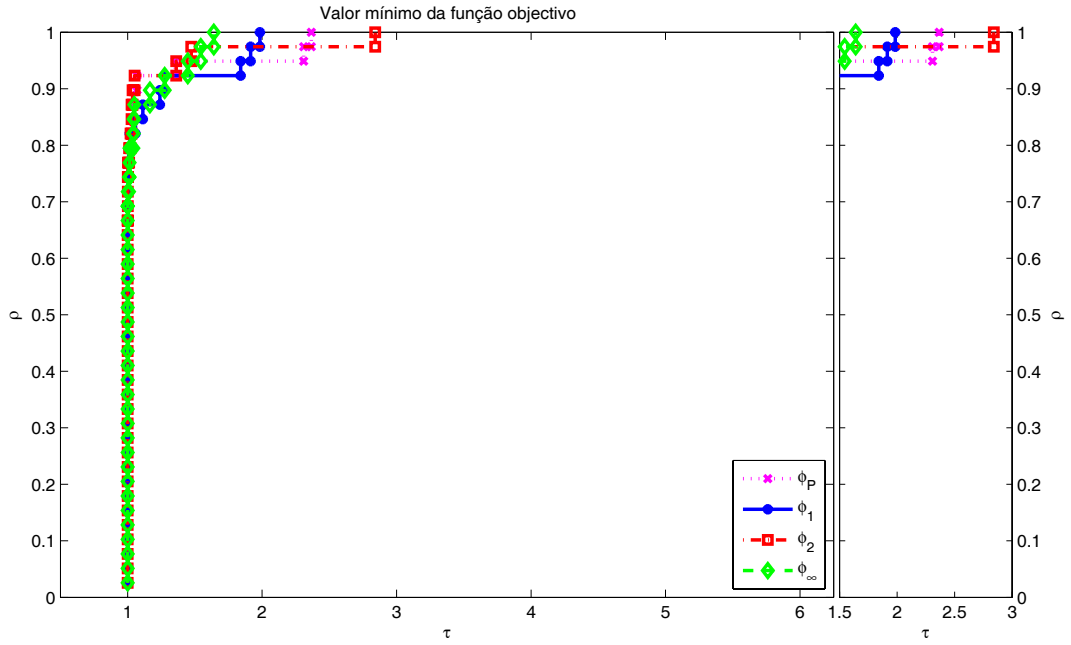


Figura B.2: Gráfico de desempenho de perfis relativo ao valor mínimo obtido da função objectivo usando a função `fminsearch`.

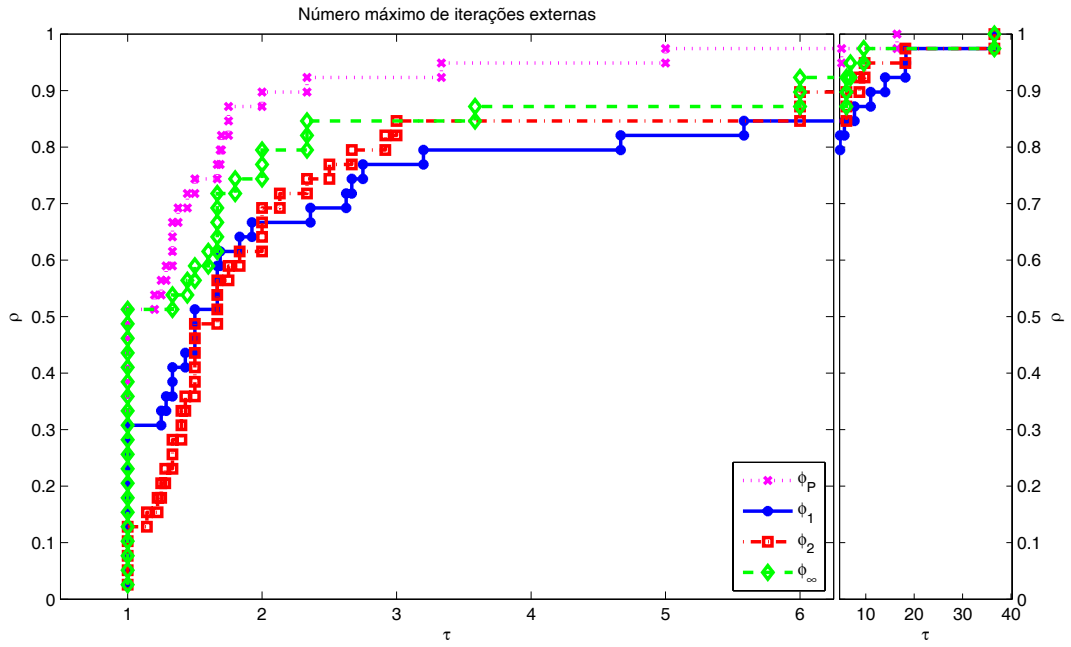


Figura B.3: Gráfico de desempenho de perfis relativo ao número máximo obtido de iterações externas usando a função `fminsearch`.

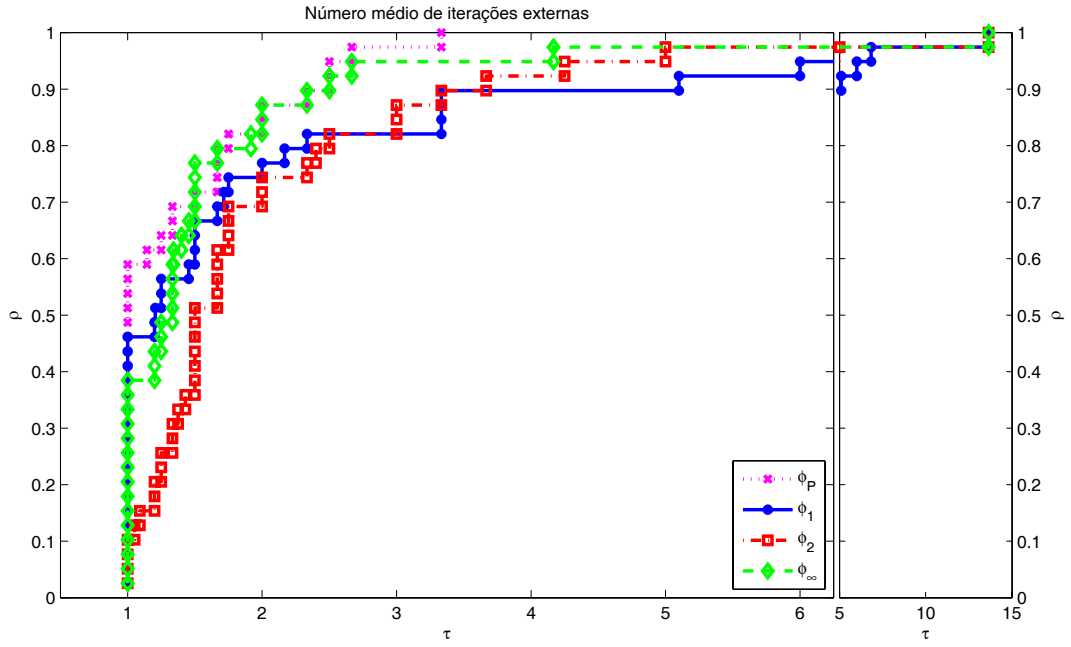


Figura B.4: Gráfico de desempenho de perfis relativo ao número médio obtido de iterações externas usando a função `fminsearch`.

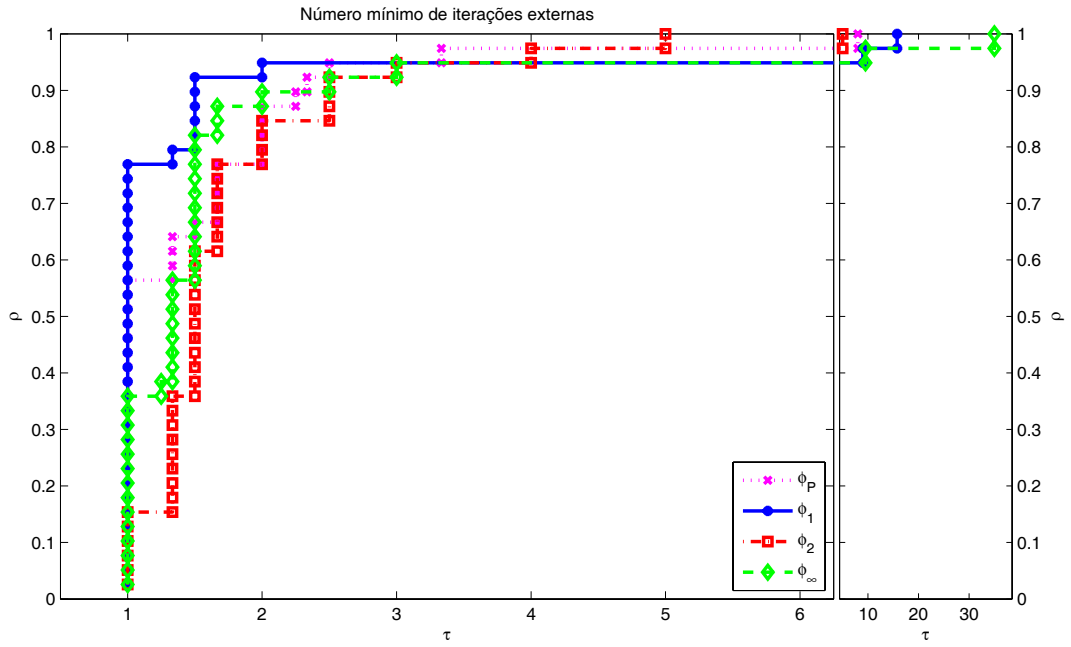


Figura B.5: Gráfico de desempenho de perfis relativo ao número mínimo obtido de iterações externas usando a função `fminsearch`.

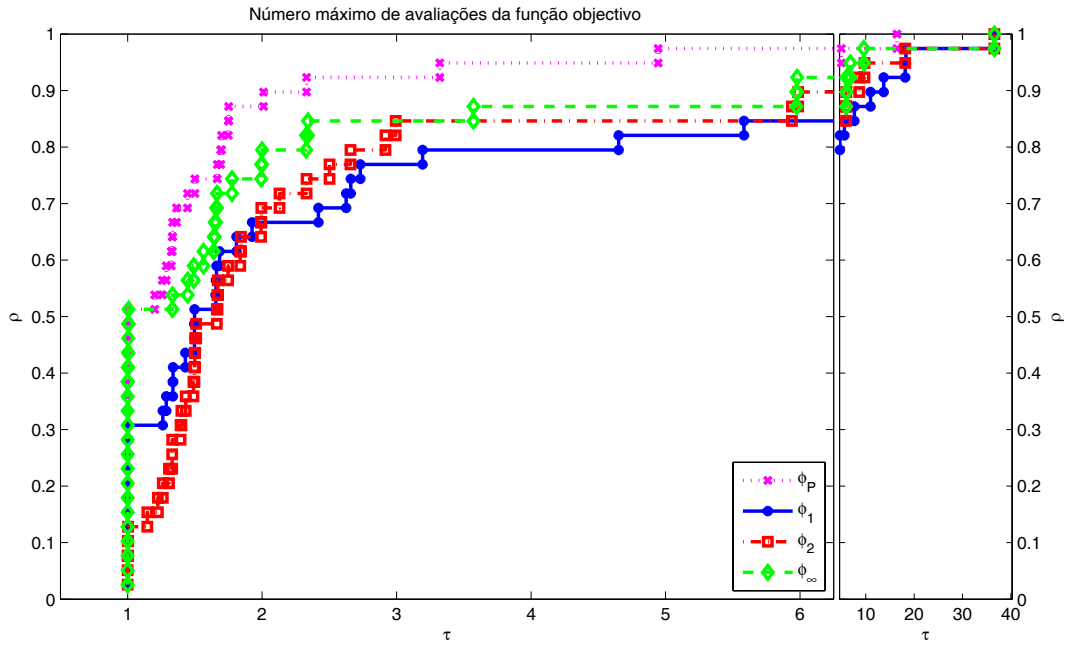


Figura B.6: Gráfico de desempenho de perfis relativo ao número máximo obtido de avaliações da função objectivo usando a função `fminsearch`.

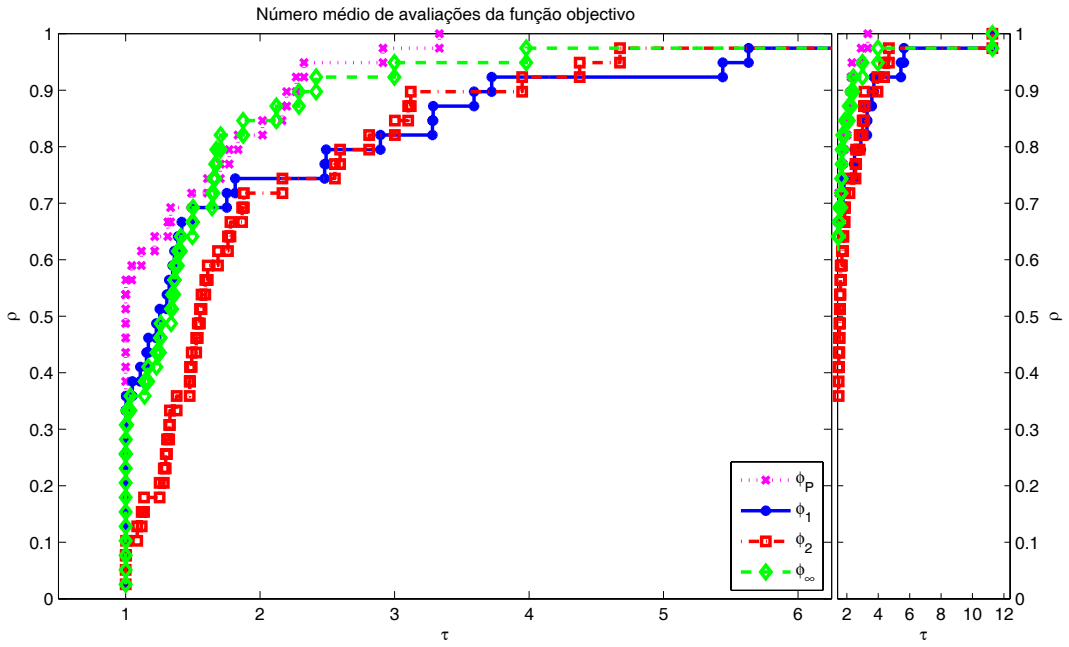


Figura B.7: Gráfico de desempenho de perfis relativo ao número médio obtido de avaliações da função objectivo usando a função `fminsearch`.

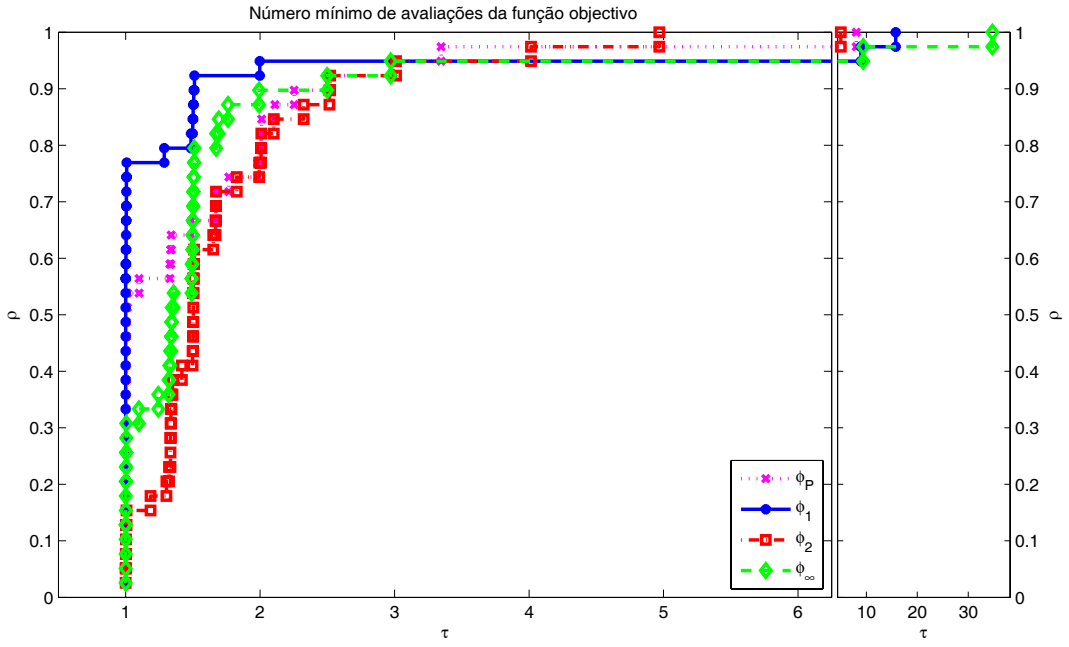


Figura B.8: Gráfico de desempenho de perfis relativo ao número mínimo obtido de avaliações da função objectivo usando a função `fminsearch`.

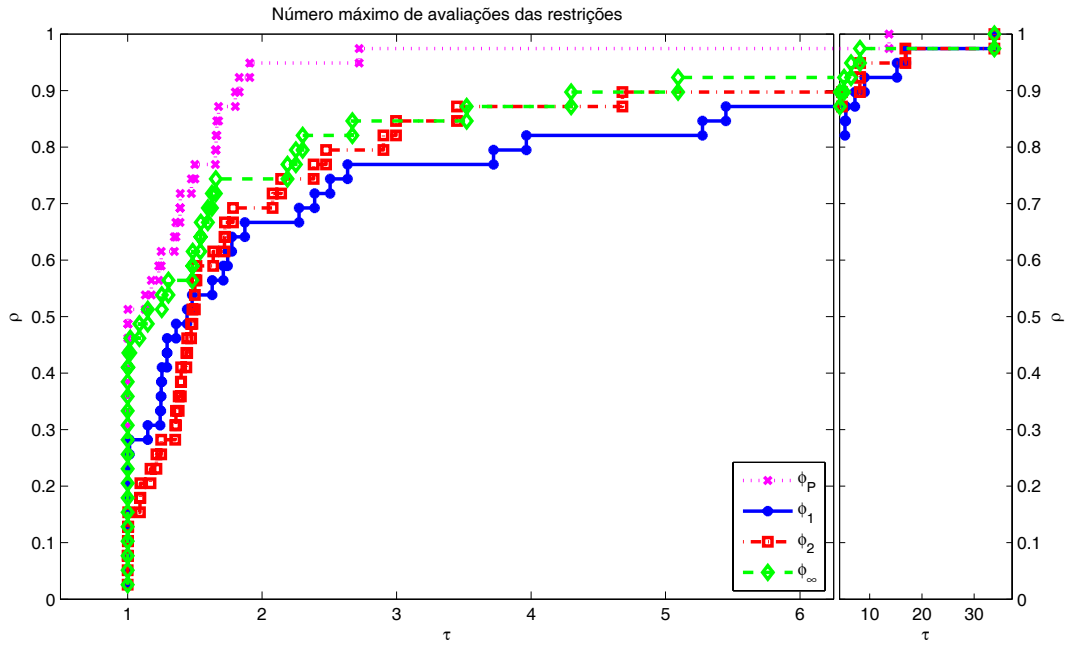


Figura B.9: Gráfico de desempenho de perfis relativo ao número máximo obtido de avaliações das funções de restrição usando a função `fminsearch`.

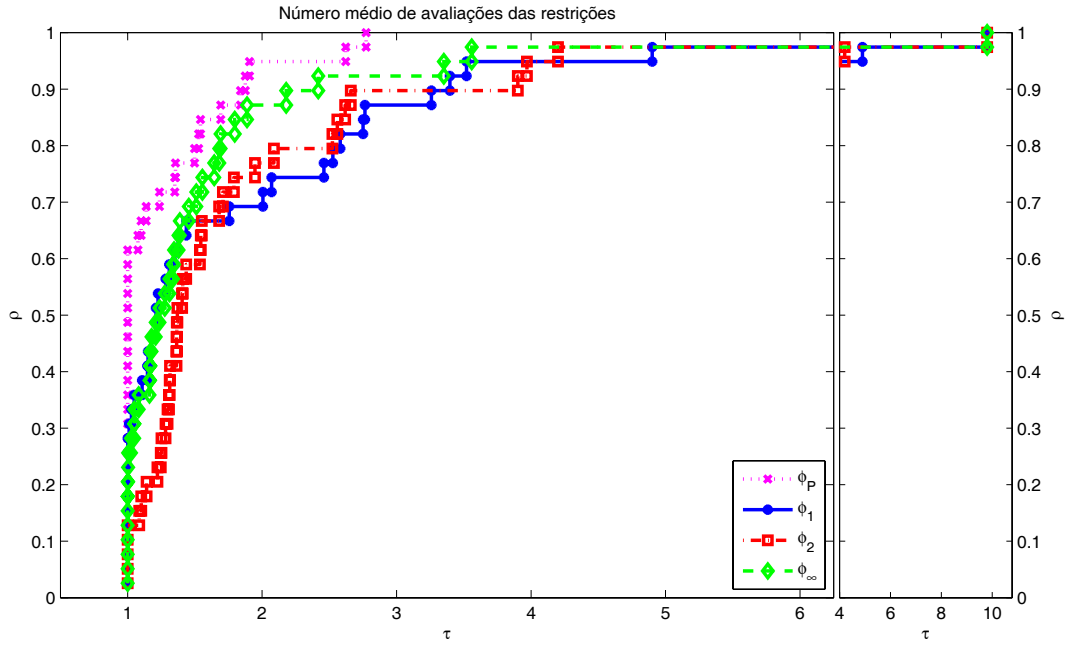


Figura B.10: Gráfico de desempenho de perfis relativo ao número médio obtido de avaliações das funções de restrição usando a função `fminsearch`.

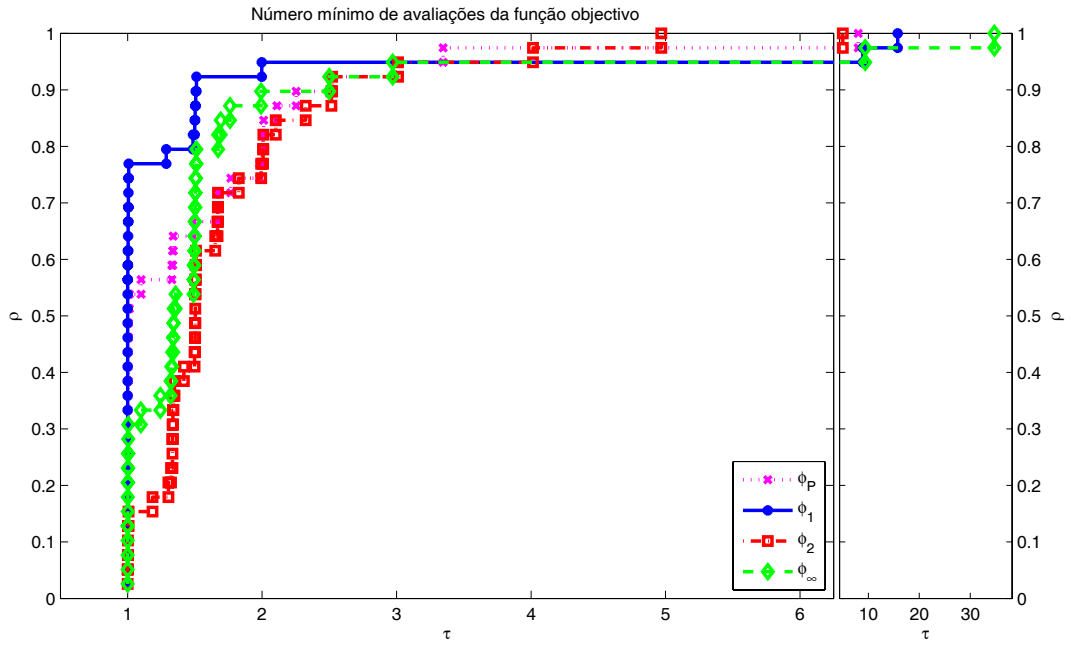


Figura B.11: Gráfico de desempenho de perfis relativo ao número mínimo obtido de avaliações das funções de restrição usando a função `fminsearch`.

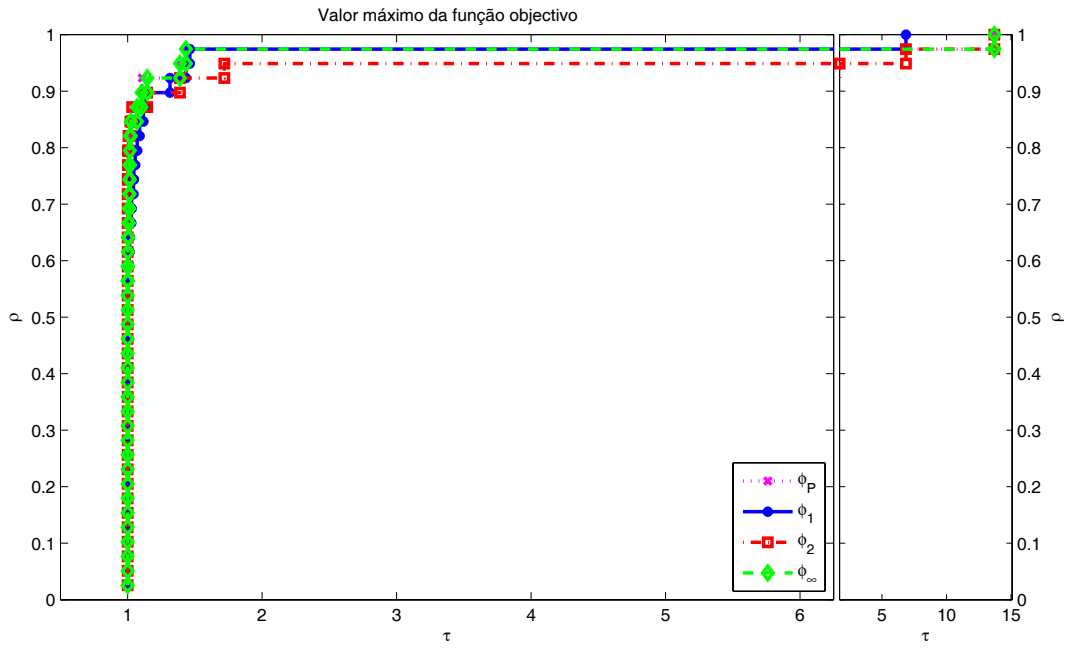


Figura B.12: Gráfico de desempenho de perfis relativo ao valor máximo obtido da função objectivo usando o *solver* PSwarm.

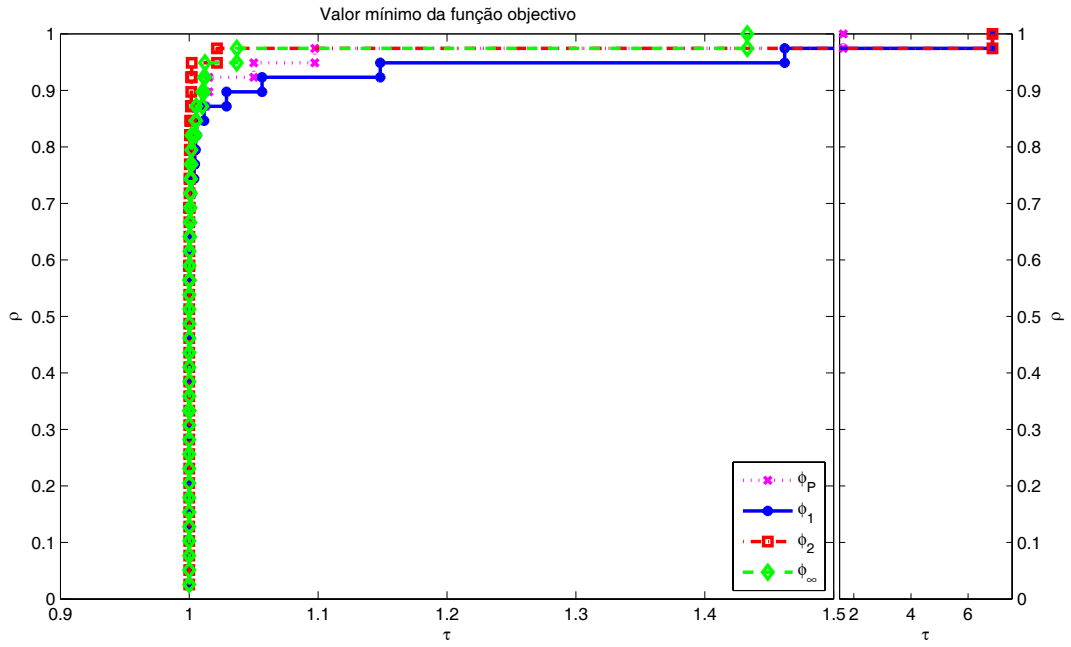


Figura B.13: Gráfico de desempenho de perfis relativo ao valor mínimo obtido da função objectivo usando o *solver* PSwarm.

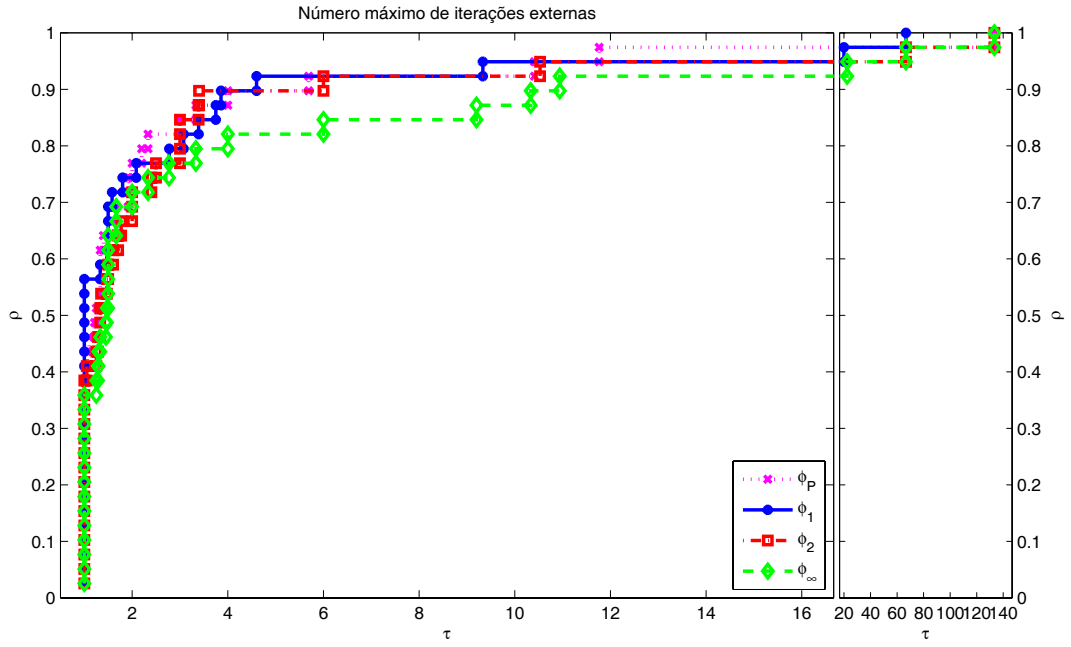


Figura B.14: Gráfico de desempenho de perfis relativo ao número máximo obtido de iterações externas usando o *solver* PSwarm.

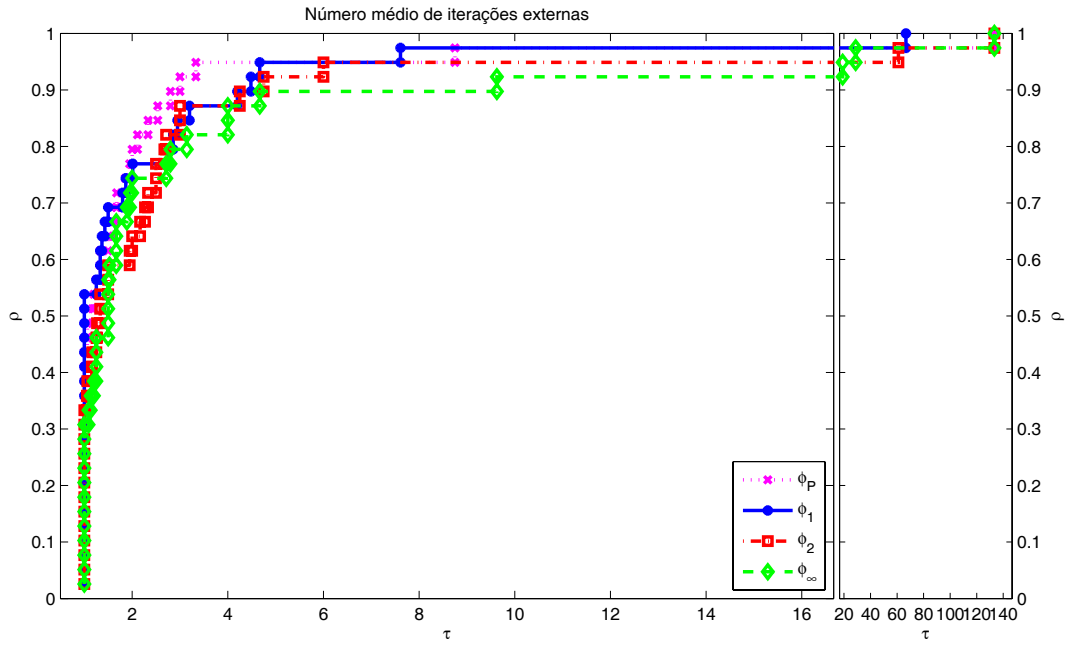


Figura B.15: Gráfico de desempenho de perfis relativo ao número médio obtido de iterações externas usando o *solver* PSwarm.

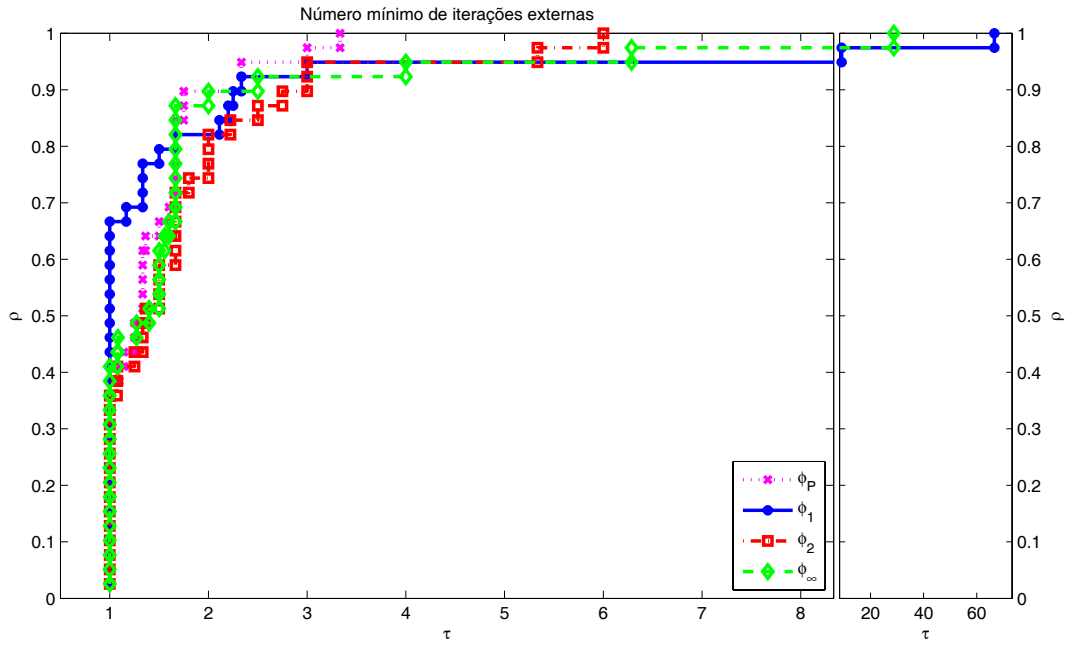


Figura B.16: Gráfico de desempenho de perfis relativo ao número mínimo obtido de iterações externas usando o *solver* PSwarm.

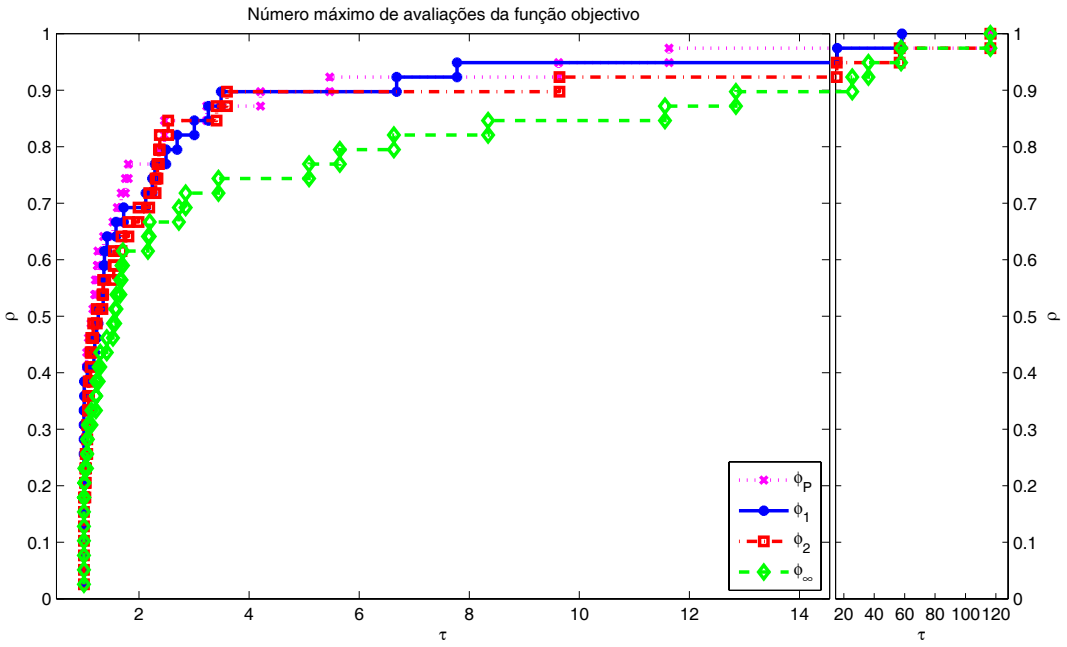


Figura B.17: Gráfico de desempenho de perfis relativo ao número máximo obtido de avaliações da função objectivo usando o *solver* PSwarm.

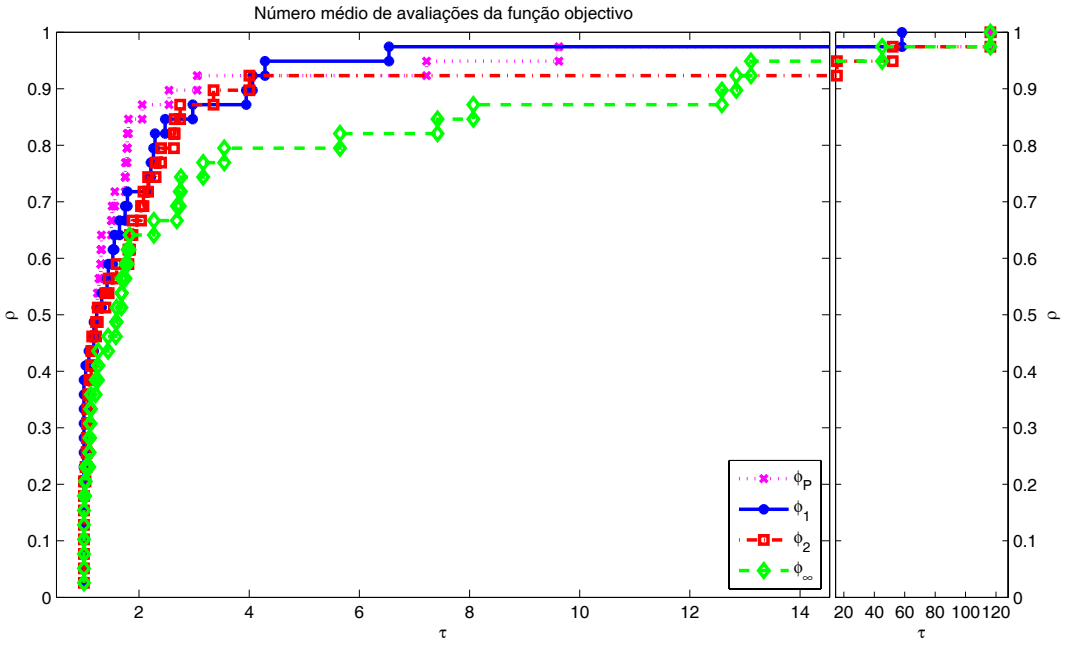


Figura B.18: Gráfico de desempenho de perfis relativo ao número médio obtido de avaliações da função objectivo usando o *solver* PSwarm.

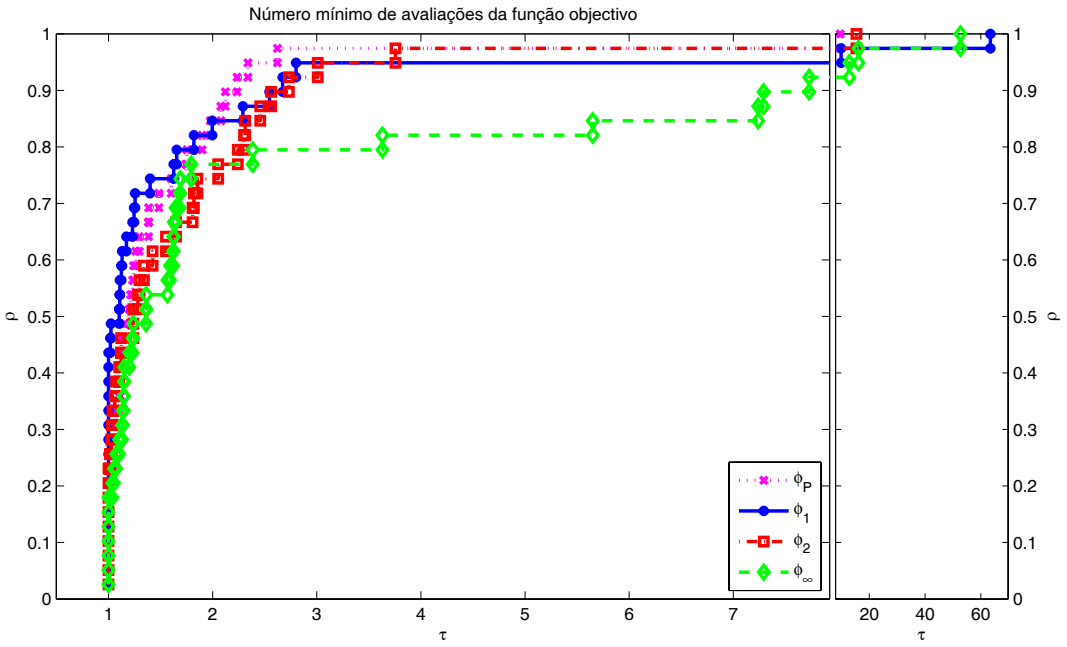


Figura B.19: Gráfico de desempenho de perfis relativo ao número mínimo obtido de avaliações da função objectivo usando o *solver* PSwarm.

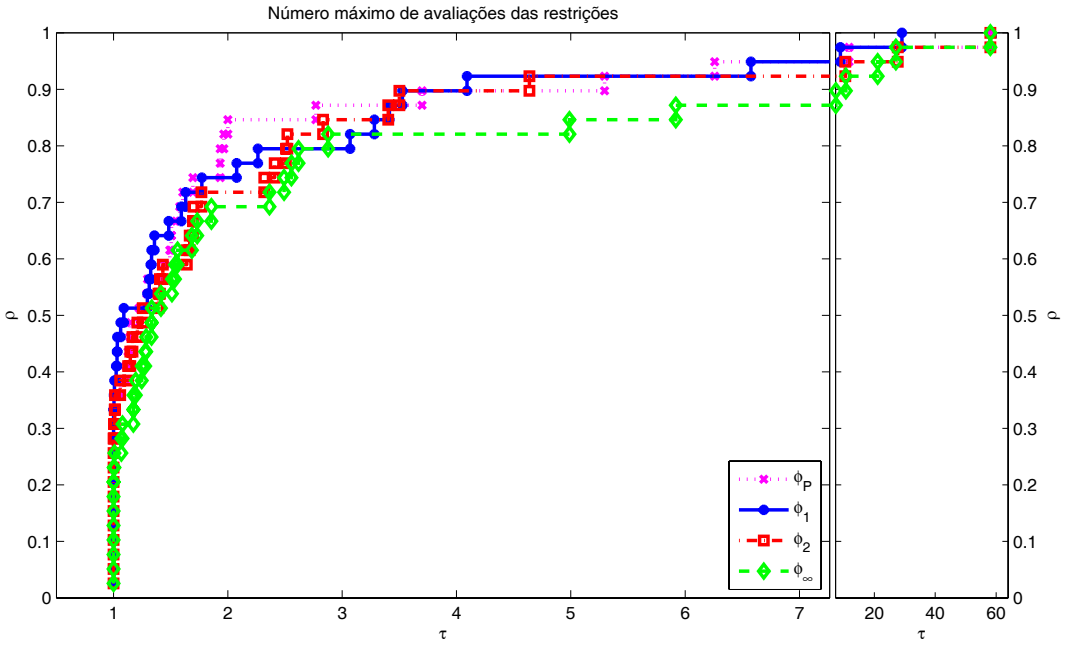


Figura B.20: Gráfico de desempenho de perfis relativo ao número máximo obtido de avaliações das funções de restrição usando o *solver* PSwarm.

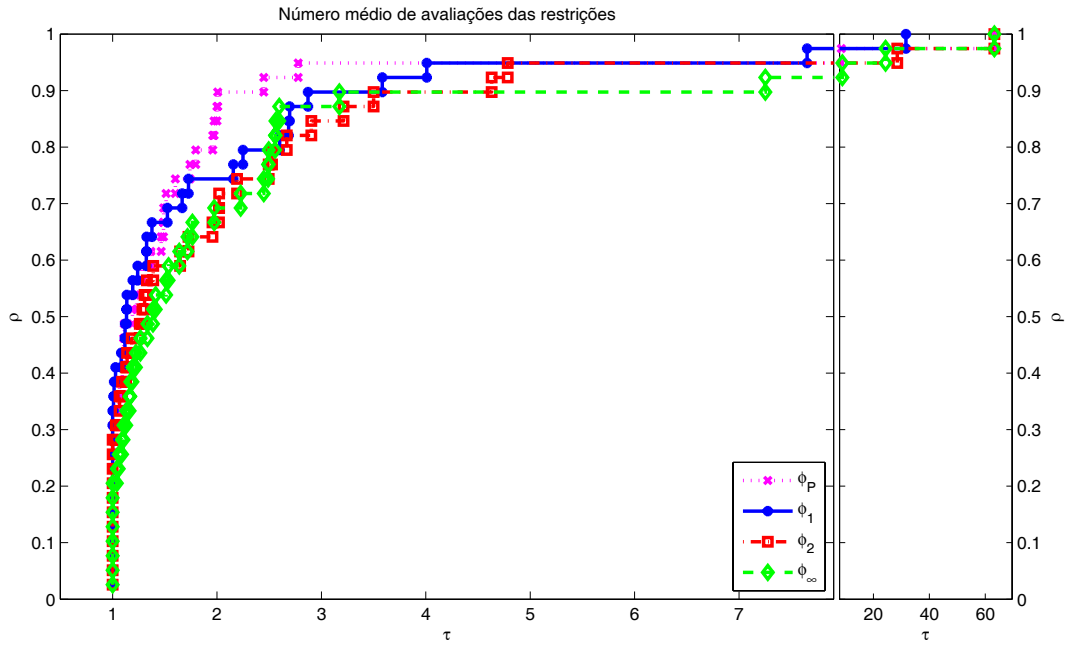


Figura B.21: Gráfico de desempenho de perfis relativo ao número médio obtido de avaliações das funções de restrição usando o *solver* PSwarm.

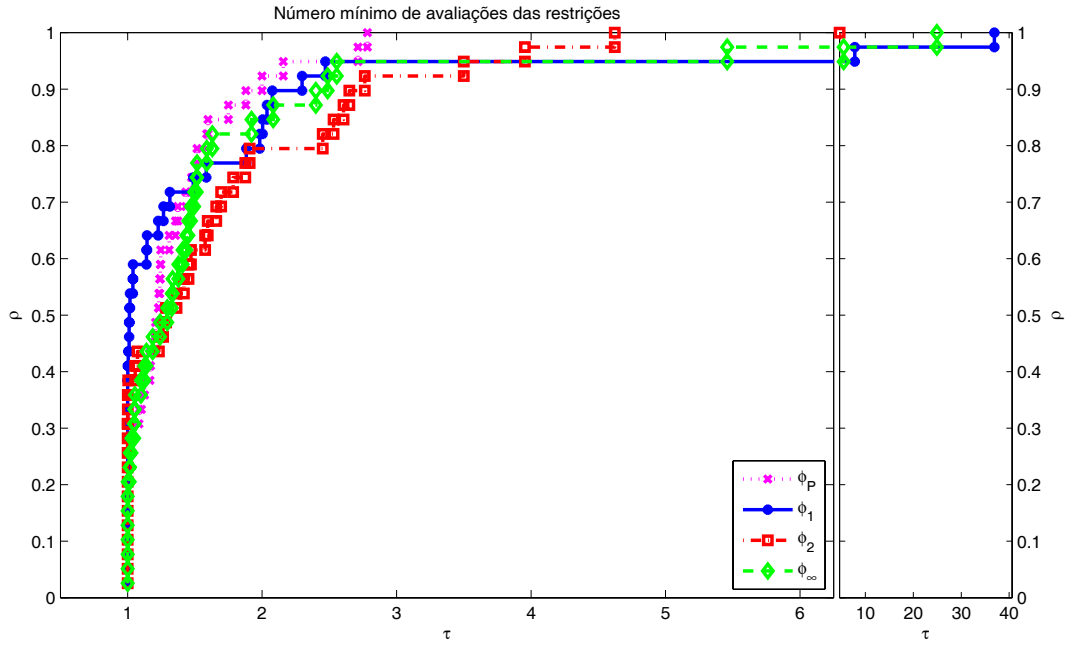


Figura B.22: Gráfico de desempenho de perfis relativo ao número mínimo obtido de avaliações das funções de restrição usando o *solver* PSwarm.

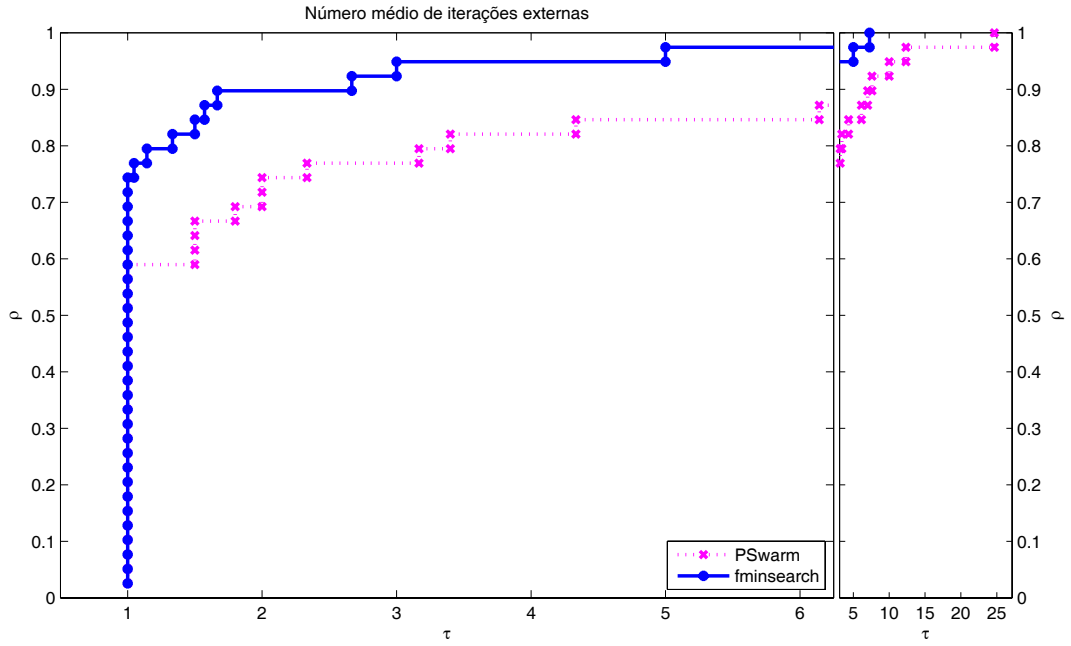


Figura B.23: Comparação entre o número médio de iterações externas entre PSwarm *vs* `fminsearch`.

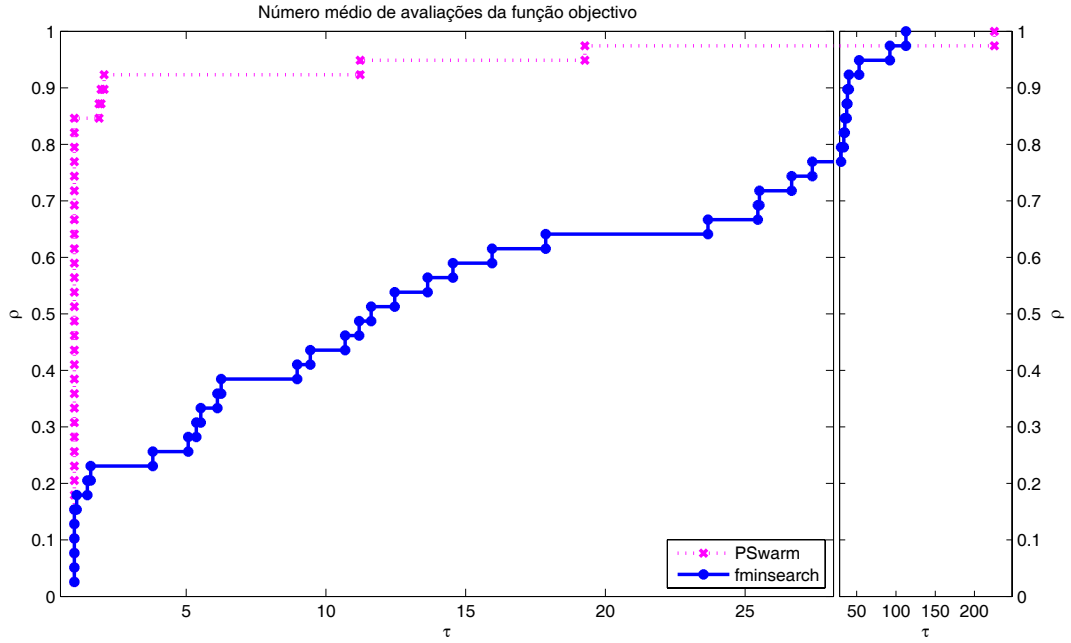


Figura B.24: Comparação entre o número médio de avaliações da função objectivo entre PSwarm *vs* `fminsearch`.

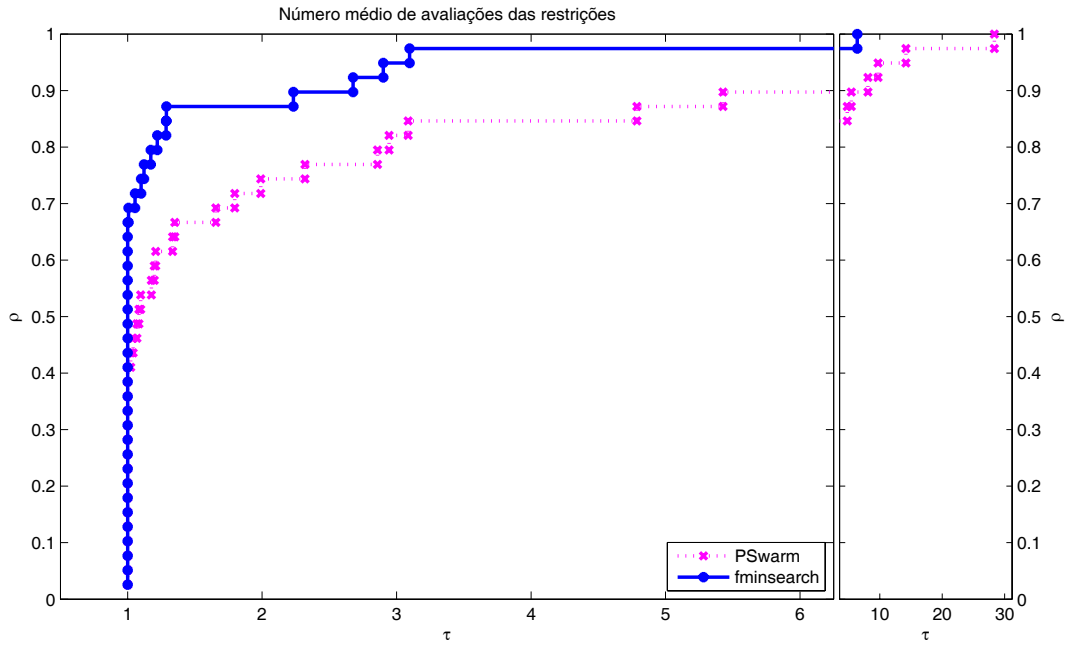


Figura B.25: Comparação entre o número médio de avaliações das funções de restrição entre PSwarm *vs* `fminsearch`.

Apêndice C

Resultados da análise de Sensibilidade

Neste apêndice apresentam-se os gráficos de perfis de desempenho obtidos de uma análise de sensibilidade feita ao algoritmo. Os gráficos comparam os resultados das 5 execuções realizadas. Em cada execução fez-se variar os parâmetros apresentados na Tabela C.1.

Tabela C.1: Variação dos parâmetros na análise de sensibilidade

Execução	ϵ	$\theta_{crossover}$	μ_0
1	10^{-5}	0.1	$\|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$
2	10^{-5}	0.1	$10 \times \|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$
3	10^{-5}	0.1	$10000 \times \|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$
4	10^{-3}	0.1	$\|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$
5	10^{-5}	0.2	$\|\frac{f(x_0)}{\max(g_i(x_0,t))}\|$

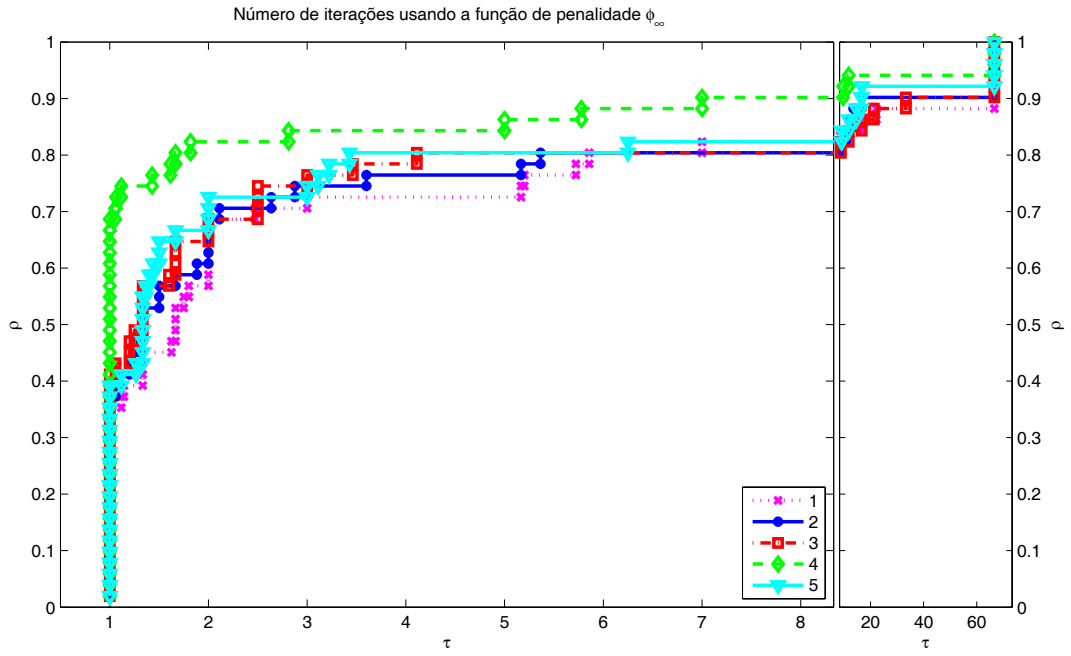


Figura C.1: Número de iterações externas usando a função ϕ_∞.

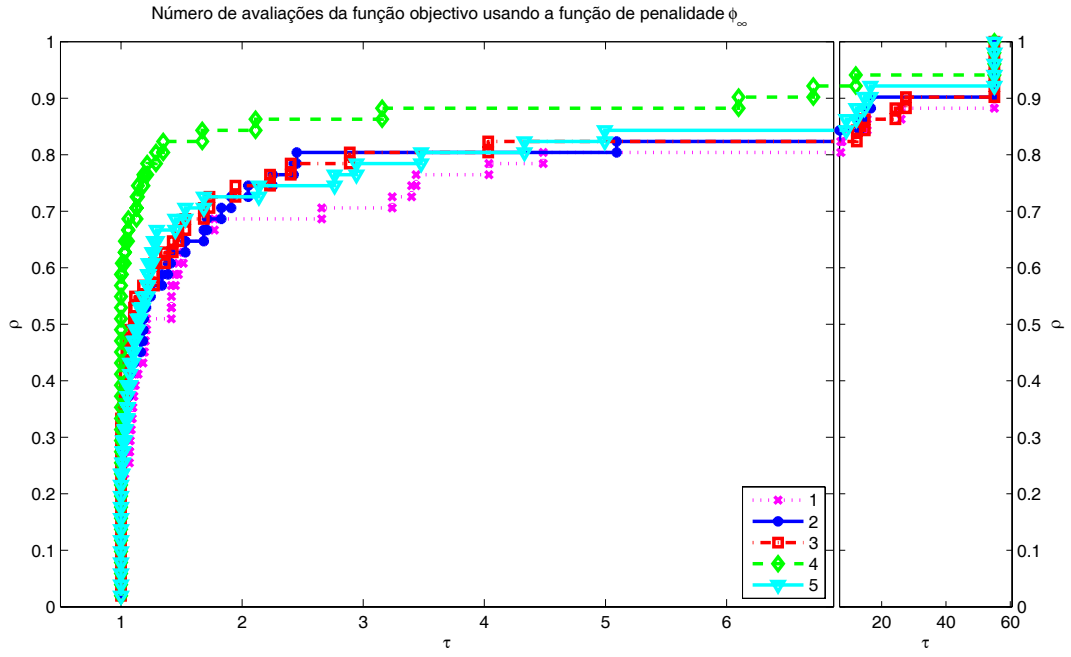


Figura C.2: Número de avaliações da função objectivo usando a função ϕ_∞.

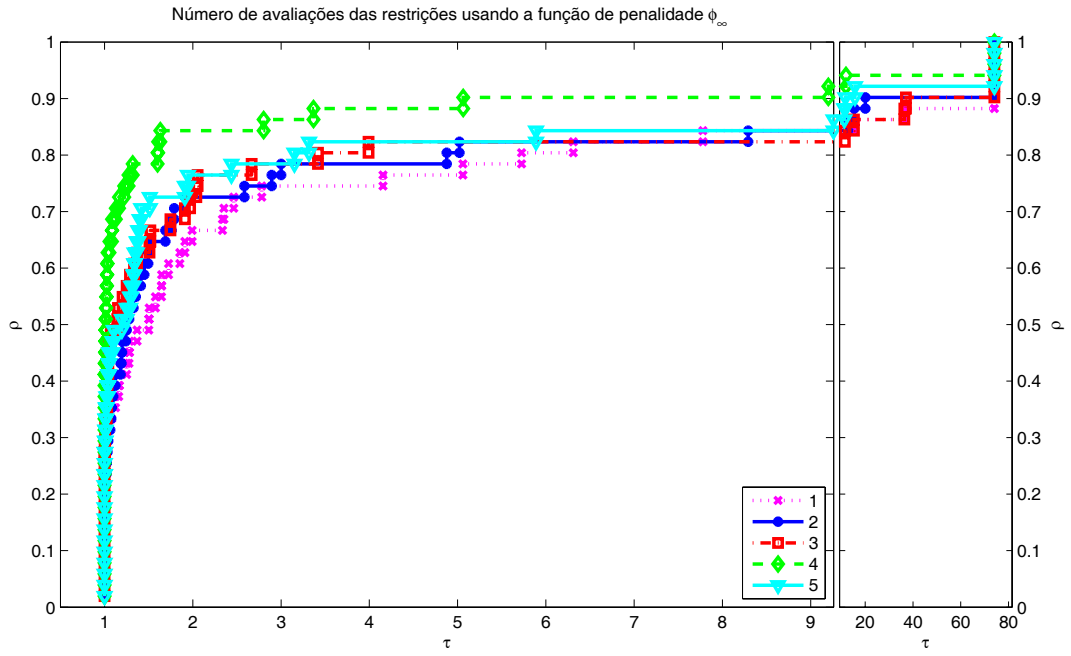


Figura C.3: Número de avaliações das funções de restrição usando a função ϕ_∞.

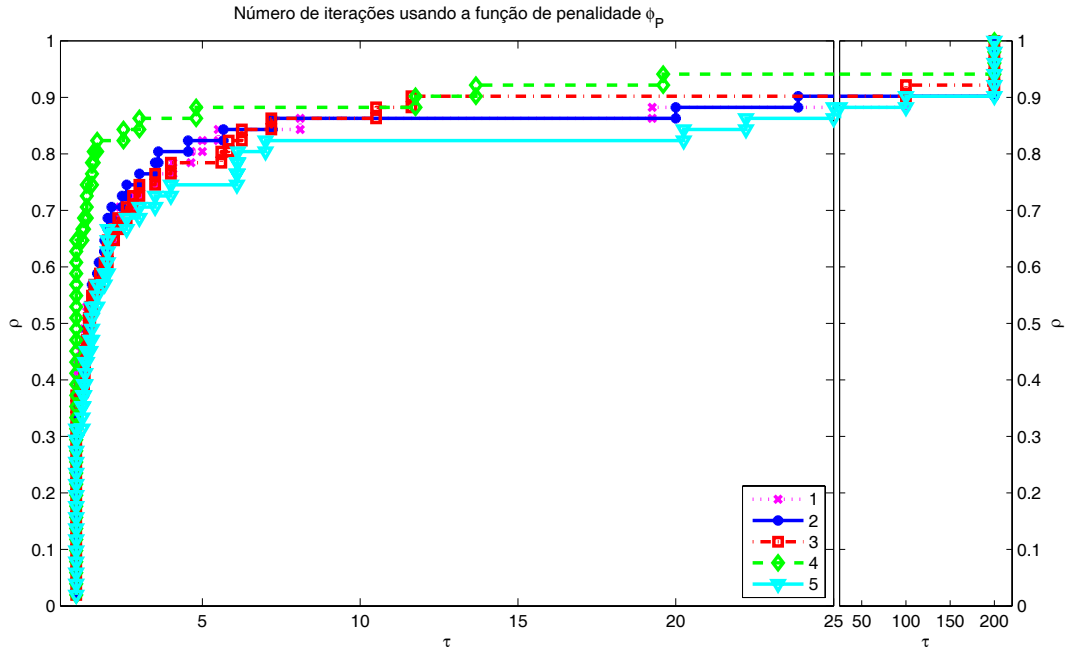


Figura C.4: Número de iterações externas usando a função ϕ_P.

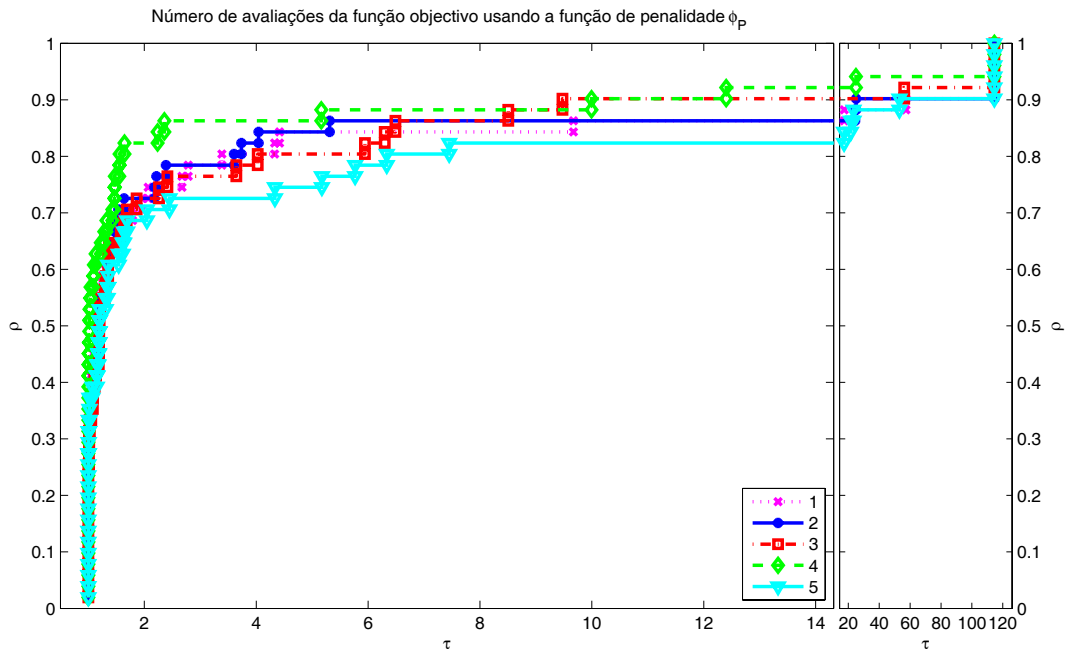


Figura C.5: Número de avaliações da função objectivo usando a função ϕ_P.

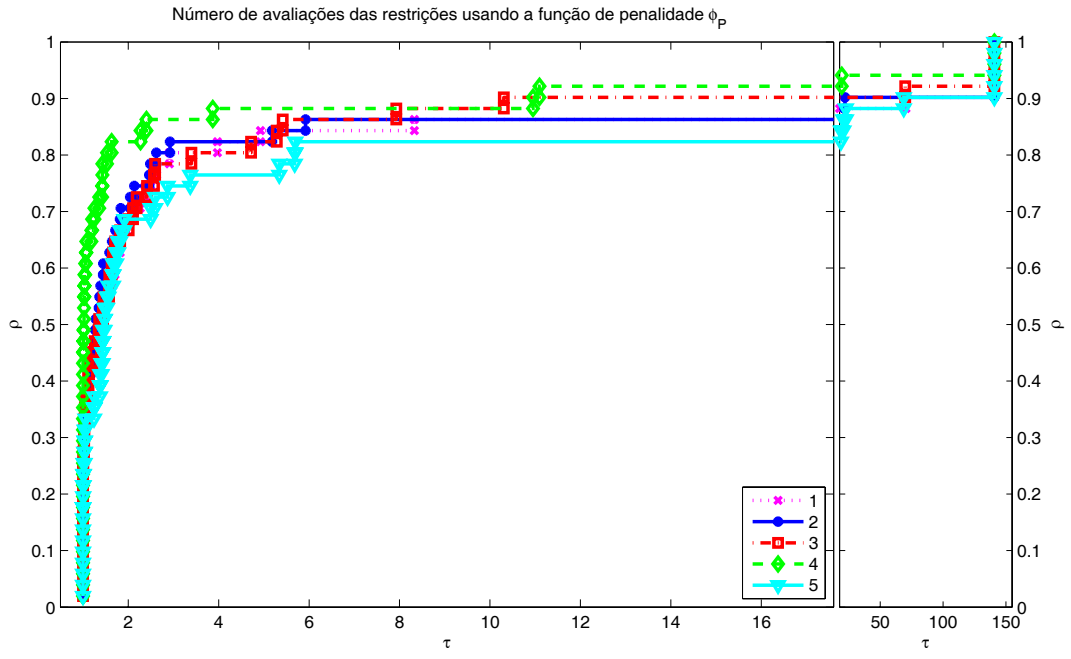


Figura C.6: Número de avaliações das funções de restrição usando a função ϕ_P.

Bibliografia

[1] M.D. Ašić e V.V. Kovačević-Vujčić. An interior semi-infinite programming method. *Journal of Optimization Theory and Applications*, 59(3):353–367, 1988.

[2] E.J. Anderson e A.S. Lewis. An extension of the simplex algorithm for semi-infinite linear programming. *Mathematical Programming*, 44:247–269, 1989.

[3] A.M. Bagirov e A.M. Rubinov. Cutting angle method and a local search. *Journal of Global Optimization*, 27(2-3):193–213, 2003.

[4] A.M. Bagirov e J. Zhang. Comparative analysis of the cutting angle and simulated annealing method in global optimization. *Optimization*, 52(4-5):363–378, 2003.

[5] B. Betrò. An accelerated central cutting plane algorithm for linear semi-infinite programming. *Mathematical Programming*, 101(3):479–495, 2004.

[6] J.W. Blankenship e J.E. Falk. Infinitely constrained optimization problems. *Journal of Optimization Theory and Applications*, 19(2):261–281, 1976.

[7] R. Brits, A. Engelbrecht, e F. van den Bergh. A niching particle swarm optimizer. In *4th Asi-Pacific Conference on Simulated Evolution and Learning(SEAL2002)*, pages 692–696, 2002.

[8] A.R. Conn e N.I.M. Gould. An exact penalty function for semi-infinite programming. *Mathematical Programming*, 37:19–40, 1987.

[9] I.D. Coope e C.J. Price. A two-parameter exact penalty function for nonlinear programming. *Journal of Optimization Theory and Applications*, 83(1):49–61, 1994.

[10] I.D. Coope e G.A. Watson. A projected Lagrangian algorithm for semi-infinite programming. *Mathematical programming*, 32(3):337–356, 1985.

[11] W.W. Cooper, H. Hemphill, Z. Huang, S. Li, V. Lelas, e D.W. Sullivan. Survey of mathematical programming models in air pollution management. *European Journal of Operations Research*, 96(1):1–35, 1997.

[12] L. Costa e A.I.F. Vaz. A genetic algorithm framework for multilocal optimization. Technical report, Universidade do Minho, Portugal, 2010.

[13] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons, Chichester, 2001.

[14] K. Deb e S. Agrawal. Simulated binary crossover for continuos search space. *Complex Systems*, 9(2):115–148, 1999.

[15] E.D. Dolan e J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(A):201–213, 2002.

[16] R. Eberhart e J. Kennedy. New optimizers using particle swarm theory. In *Proceedings of the 1995 6th International Sysmposium n Micro Machine and Human Science*, pages 39–43, 1995.

[17] S.-C. Fang, C.-J. Lin, e S.-Y. Wu. On solving convex quadratic semi-infinite programming problems. *Optimization*, 21:107–125, 1994.

[18] A.V. Fiacco e K.O. Kortanek, editors. *Semi-Infinite Programming and Applications*. Springer-Verlag, 1983.

[19] B. Fischer e J. Modersitzki. An algorithm for complex linear approximation based on semi-infinite programming. *Numerical Algorithms*, 5(6):287–297, 1993.

[20] R. Fletcher. *Pratical Methods of Optimization 2 ed.* John Wiley and Sons, New York, 1986.

[21] C. Floudas. Recent in global optimization for process synthesis, design and control: enclosure of all solutions. *Computers and Chemical Engineering*, pages 963–973, 1999.

[22] R. Fourer, D.M. Gay, e B.W. Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.

[23] P.E. Gill, W. Murray, M.A. Saunders, e M.H. Wright. *User's Guide for NPSOL: A Fortran Package for Nonlinear Programming.* Stanford University, 1986.

[24] M.S. Gockenbach e A.J. Kearsley. Optimal signal sets for non-gaussian detectors. *SIAM Journal on Optimization*, 9(2):316–326, 1999.

[25] S. Görner, A. Potchinkov, e R. Reemtsen. The direct solution of nonconvex nonlinear FIR filter design problems by a SIP method. *Optimization and Engineering*, 1:123–154, 2000.

[26] H.J. Greenberg. Mathematical programming models for environmental quality control. *Operations Research*, 43(4):578–622, 1995.

[27] P.R. Gribik. A central-cutting-plane algorithm for semi-infinite programming problems. In [34], pages 66–82, 1979.

[28] J. Guddat, H.Th. Jongen, F. NožIčKa, J. Still, e F. Twilt. Parametric optimization and related topics IV. In B. Brosowski, F. Deutsch, e J. Guddat, editors, *Approximation & Optimization*, volume 9. Verlag Peter Lang, 1995.

[29] S.-A. Gustafson. A three-phase algorithm for semi-infinite programs. In [18], pages 138–157, 1983.

[30] S.-A. Gustafson, K.O. Kortanek, e J.R. Sweigart. Numerical optimization techniques in air quality modeling: objective interpolation formulas for the spatial distribution of pollutant concentration. *Journal of Applied Metereology*, 16(12):1243–1255, 1977.

[31] E. Haaren-Retagne. *A Semi-Infinite Programming Algorithm for Robot Trajectory Planning*. Tese de doutoramento, University of Trier, 1992.

[32] S.-P. Han e O.L. Mangasarian. Exact penalty functions in nonlinear programming. *Mathematical Programming*, 17:251–269, 1979.

[33] A. Heim e O. Von Stryk. Trajectory optimization of industrial robots with application to computer-aided robotics and robot controllers. *Optimization*, 47:407–420, 2000.

[34] R. Hettich, editor. *Semi-Infinite Programming*. Springer-Verlag, 1979.

[35] R. Hettich. An implementation of a discretization method for semi-infinite programming. *Mathematical Programming*, 34(3):354–361, 1986.

[36] R. Hettich e G. Gramlich. A note on an implementation of a method for quadratic semi-infinite programming. *Mathematical Programming*, 46:249–254, 1990.

[37] R. Hettich, A. Kaplan, e R. Tichatschke. Semi-infinite programming: Numerical methods. *Encyclopedia of Optimization*, pages 3429–3434, 2009.

[38] R. Hettich e K.O. Kortanek. Semi-infinite programming: Theory, Methods, and Applications. *SIAM Review*, 35(3):380–429, 1993.

[39] R. Hettich e G. Still. Semi-infinite programming: Second order optimality conditions. *Encyclopedia of Optimization*, pages 3434–3439, 2009.

[40] R. Horst e H. Tuy. *Global Optimization - Deterministic Approach*. Springer, second edition, 1992.

[41] H. Hu. A globally convergent method for semi-infinite linear programming. *Journal of Global Optimization*, 8:189–199, 1996.

[42] L. Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics*, 25(1):33–54, 1996.

[43] L.S. Jennings e K.L. Teo. A computational algorithm for functional inequality constrained optimization problems. *Automatica*, 26(2):371–375, 1990.

[44] J. Kaliski, D. Haglin, C. Roos, e T. Terlaky. Logarithmic barrier decomposition methods for semi-infinite programming. *International Transactions in Operational Research*, 4(4):285–303, 1997.

[45] A.H.G.R. Kan e G.T. Timmer. Stochastic global optimization methods, part I: Clustering methods. *Mathematical Programming*, 39:27–56, 1987.

[46] A.H.G.R. Kan e G.T. Timmer. Stochastic global optimization methos part II: Multi level methods. *Mathematical Programming*, 39(1):57–78, 1987.

[47] J. Kennedy e R. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, Perth, Australia, 1995. http://engr.iupui.edu/ shi/Coference/psopap4-html.

[48] E. Kiseleva e T. Stepanchuk. On the efficiency of a global non-diffrentiable optimization algorithm based on the method of optimal set partitioning. *Journal of Global Optimization*, 25:209–235, 2003.

[49] T. Kolda. Generating set search methods for practical optimization, smart fields seminar. Technical report, U. Standord, Sandia National Laboratories, 2008.

[50] T. Kolda, R.M. Lewis, e V. Torczon. A generating set direct search augmented lagrangian algorithm for optimization with a combination of general and linear constraints. Technical Report SAND 2006-5315, SANDIA National Laboratories, 2006.

[51] K.O. Kortanek e H. No. A central cutting plane algorithm for convex semi-infinite programming problems. *SIAM Journal on Optimization*, 3(4):901–918, 1993.

[52] C.T. Lawrence. *A Computationally Efficient Feasible Sequencial Quadratic Programming Algorithm*. Tese de doutoramento, Institute for Systems Research, University of Maryland, 1998.

[53] C.T. Lawrence e A.L. Tits. Feasible sequential quadratic programming for finely discretized problems from SIP. In [98], pages 159–193, 1998.

[54] C.T. Lawrence e J.L. Zhou A.L. Tits. User's Guide for CFSQP Version 2.4: A C code for solving (Large Scale Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints. Technical Report ISR TR 94-16rl, Institute for Systems Research, University of Maryland, 1996.

[55] T. León, S. Sanmatías, e E. Vercher. A multi-local optimization algorithm. *TOP*, 6(1):1–18, 1998.

[56] T. León, S. Sanmatías, e E. Vercher. On the numerical treatment of linearly constrained semi-infinite optimization problems. *European Journal of Operational Research*, 121:78–91, 2000.

[57] T. León e E. Vercher. A purification algorithm for semi-infinite programming. *European Journal of Operational Research*, 57:412–420, 1992.

[58] Y. Li e D. Wang. A semi-infinite programming model for Earliness/Tardiness production planning with simulated annealing. *Mathematical and Computer Modelling*, 26(7):35–42, 1997.

[59] C.-J. Lin, S.-C. Fang, e S.-Y. Wu. An unconstrained convex programming approach to linear semi-infinite programming. *SIAM Journal on Optimization*, 8(2):443–456, 1998.

[60] Y.-C. Liou, S.-Y. Wu, e J.-C. Yao. Bilevel decision with generalized semi-infinite optimization for fuzzy mapping as lower level problems. *Fuzzy Optimization and Decision Making*, 4(1):41–50, 2005.

[61] Y. Liu e K.L. Teo. An adaptive dual parametrization algorithm for quadratic semi-infinite programming problems. *Global Optimization*, 24:205–217, 2002.

[62] Y. Liu, K.L. Teo, e S. Ito. A dual parametrization approach to linear-quadratic semi-infinite programming problems. *Optimization Methods and Software*, 10:471–495, 1999.

[63] Y. Liu, K.L. Teo, e S.Y. Wu. A new quadratic semi-infinite programming algorithm based on dual parametrization. *Global Optimization*, 29:401–413, 2004.

[64] S.P. Marin. Optimal parametrization of curves for robot trajectory design. *IEEE Transactions on Automatic Control*, 33(2):209–214, 1988.

[65] J.M. Martínez e S.A. Santos. Métodos computacionais de otimização. Technical Report ISBN 85-244-0092-7, 256 pages, 20^o Colóquio Brasileiro de Matemática, Rio de Janeiro, 1995.

[66] MathWorks. *MATLAB*. The MathWorks Inc., 1999. Version 5.4, Release 11.

[67] MathWorks. *MATLAB*. The MathWorks Inc., 2001. Version 6.1, Release 12.1.

[68] MathWorks. *MATLAB Optimization Toolbox*. The MathWorks Inc., 2001. In [67].

[69] T. Meng, T. Ray, e P. Dhar. Supplementary material on parameter estimation using particle swarm. *Preprint submitted to Elsevier Science*, 2004.

[70] S.G. Nash e A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill International Editions, 1996.

[71] J. Nocedal e S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[72] Visual Numerics. Library IMSL. http://www.vni.com/products/imsl/.

[73] K. Parsopoulos, V. Plagianakos, G. Magoulas, e M. Vrahatis. Objective function stretching to alleviate convergence to local minima. *Nonlinear Analisys*, 47:3419–3424, 2001.

[74] K. Parsopoulos e M. Vrahatis. Recent approaches to global optimization problem through particle swarm optimization. *Natural Computing*, 1:235–306, 2002.

[75] A.I.P.N. Pereira. *Caracterização da Função Penalidade Exponencial num Método de Redução para Programação Semi-Infinita*. Tese de doutoramento, Universidade do Minho, Escola de Engenharia, Portugal, 2006.

[76] A.I.P.N. Pereira e E.M.G.P. Fernandes. A new algorithm to identify all global maximizers based on simulated annealing. *Proceedings of the 6th World Congress on Structural and Multidisciplinary Optimization,(CD-ROM)*, 2005.

[77] A.I.P.N. Pereira e E.M.G.P. Fernandes. On a reduction line search filter method for nonlinear semi-infinite programming problems. *EUROPT'2008*, pages 174–179, 2008.

[78] G. Di Pillo. Exact penalty methods. In *I. Ciocco(Ed.), Algorithms for Continuous Optimization*, pages 1–45. Kluwer Acad. Publ., 1994.

[79] J.D. Pintér. *Global Optimization in Action*, volume 6 of *Nonconvex Optimization and its Applications*. Kluer Academic Publishers, 1996.

[80] E. Polak. On the mathematical foundations of nondifferentiable optimization in engineering design. *SIAM Review*, 29(1):21–89, 1987.

[81] E. Polak e J.O. Royset. On the use of augmented lagrangians in the solution of generalized semi-infinite min-max problems. *Computatinal Optimization and Applications*, 31(2):173–192, 2005.

[82] A. Potchinkov. Design of optimal linear phase FIR filters by a semi-infinite programming technique. *Signal Processing*, 58:165–180, 1997.

[83] A. Potchinkov e R. Reemtsen. A globally most violated cutting plane method for complex minimax problems with application to digital filter design. *Numerical Algorithms*, 5:611–620, 1993.

[84] A. Potchinkov e R. Reemtsen. FIR filter design in the complex domain by a semi-infinite programming technique. I. The Method. *AEÜ*, 48(3):135–144, 1994.

[85] A. Potchinkov e R. Reemtsen. FIR filter design in the complex domain by a semi-infinite programming technique. II. Examples. *AEÜ*, 48(4):200–209, 1994.

[86] A. Potchinkov e R. Reemtsen. The design of FIR filters in the complex plane by convex optimization. *Signal Processing*, 46:127–146, 1995.

[87] A. Potchinkov e R. Reemtsen. The simultaneous approximation of magnitude and phase by FIR digital filters. I: A new approach. *International Journal of Circuit Theory and Applications*, 25:167–177, 1997.

[88] C.J. Price. *Non-Linear Semi-Infinite Programming*. Tese de doutoramento, University of Canterbury, New Zealand, 1992.

[89] C.J. Price e I.D. Coope. An exact penalty function algorithm for semi-infinite programmes. *BIT*, 30:723–734, 1990.

[90] C.J. Price e I.D. Coope. Numerical experiments in semi-infinite programming. *Computational Optimization and Applications*, 6:169–189, 1996.

[91] J.-J. Rückmann e A. Shapiro. First-order optimality conditions in generalized semi-infinite programming. *Journal of Optimization Theory and Applications*, 101(3):677–691, 1999.

[92] J.-J. Rückmann e A. Shapiro. Second-order optimality conditions in generalized semi-infinite programming. *Set-Valued Analysis*, 9:169–186, 2001.

[93] R. Reemtsen. Discretization methods for the solution of semi-infinite programming problems. *Journal of Optimization Theory and Applications*, 71(1):85–103, 1991.

[94] R. Reemtsen. A cutting plane method for solving minimax problems in the complex plane. *Numerical Algorithms*, 2:409–436, 1992.

[95] R. Reemtsen. Some outer approximation methods for semi-infinite optimization problems. *Journal of Computational and Applied Mathematics*, 53:87–108, 1994.

[96] R. Reemtsen. Semi-infinite programming: Discretization methods. *Encyclopedia of Optimization*, pages 3417–3424, 2009.

[97] R. Reemtsen e A. Potchinkov. FIR filter design in regard to frequency responce, magnitude, and phase by semi-infinite programming. In [28], pages 299–314, 1995.

[98] R. Reemtsen e J.-J. Rückmann, editors. *Semi-Infinite Programming*. Kluwer Academic Publishers, 1998.

[99] K. Roleff. A stable multiple exchange algorithm for linear SIP. In [34], pages 83–96, 1979.

[100] J.O. Royset, E. Polak, e A. Der Kiureghian. Adaptive approximations and exact penalization for the solution of generalized semi-infinite min-max problems. *SIAM Journal on Optimization*, 14(1):1–34, 2003.

[101] S. Salhi e N. Queen. A hybrid algorithm for identifying global and local minima when optimizating function with many minima. *European Journal od Operations Research*, 155:51–67, 2004.

[102] O. Stein e G. Still. On optimality conditions for generalized semi-infinite programming problems. *Journal of Optimization Theory and Applications*, 103(2):443–458, 2000.

[103] G. Still. Generalized semi-infinite programming: Theory and methods. *European Journal of Operational Research*, 119:301–313, 1999.

[104] R.L. Streit. Solution of systems of complex linear equations in th linfty norm with constraints on the unknowns. *SIAM Scientif and Statistical Computing*, 7:132–149, 1986.

[105] O. Von Stryk e M. Schlemmer. Optimal control of the industrial robot manutec R3. *in: R. Bulirsch, D. Kraft (eds.), Computational Optimal Control, International Series of Numerical Mathematics*, 115:367–382, 1994.

[106] K. Suyama, K. Takada, R. Hirabayashi, e H. Iwakura. Complex chebyshev approximation for fir filters using linear semi-infinite programming method. *Electronics and Communications in Japan*, 87(3):1–8, 2004.

[107] V.B. Tadic, S.P. Meyn, e R. Tempo. Randomized algorithms for semi-infinite programming problems. *Probabilistic and Randomized Methods for Design under Uncertainty*, pages 243–261, 2006.

[108] Y. Tanaka. A trust region method for semi-infinite programming problems. *International Journal of Systems Science*, 30(2):199–204, 1999.

[109] K.L. Teo e C.J. Goh. A simple computational procedure for optimization problems with functional inequality constraints. *IEEE Transactions on Automatic Control*, AC-32(10):940–941, 1987.

[110] I. Tsoulos e I. Lagaris. Gradient-controlled, typical-distance clustering for global optimization. 2004.

[111] W. Tu e R. Mayne. Studies of multi-start clustering for global opimization. *Internation Journal Numerical Methods in Engineering*, 53(9):2239–2252, 2002.

[112] A.I.F. Vaz. *Aplicações, Métodos e Ferramentas Para Programação Semi-Infinita Não Linear*. Tese de doutoramento, Universidade do Minho, Escola de Engenharia, Portugal, 2003. Downloadable from http://www.norg.uminho.pt/aivaz/.

[113] A.I.F. Vaz. Multilocal particle swarm optimization algorithm. Technical Report ALG/IV/6-2004, Universidade do Minho, 2004.

[114] A.I.F. Vaz e E.M.G.P. Fernandes. Particle swarm algorithms for multi-local optimization. In *Proceedings of I Congresso de Estatística e Investigação Operacional da Galiza e Norte de Portugal*, Guimarães, Portugal, 2005.

[115] A.I.F. Vaz e E.M.G.P. Fernandes. Tools for robotic trajectory planning using cubic splines and semi-infinite programming. *Lectures Notes in Economics and Mathematical Systems*, 563:399–413, 2006.

[116] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Discretization methods for semi-infinite programming. *Investigação Operacional*, 21(1):37–46, 2001.

[117] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. NSIPS: Nonlinear Semi-Infinite Programming Solver. Technical Report ALG/EF/1-2001, 2001. http://www.eng.uminho.pt/∼dps/aivaz/.

[118] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Penalty function algorithms for semi-infinite programming. In *EURO 2001*, page 91, Julho 2001.

[119] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. A penalty technique for semi-infinite programming. In *Proceedings of V-SGAPEIO*, pages 235–238, Ferrol, Spain, 2001.

[120] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Robot trajectory planning with semi-infinite programming. *Proceedings of the ORP3 conference (7 pages)*, 2001. Paris.

[121] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. A sequential quadratic programming for nonlinear semi-infinite programming. In *Optimization 2001*, page 27, Julho 2001.

[122] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. NSIPS v2.0: Nonlinear Semi-Infinite Programming Solver. Technical Report ALG/EF/1-2002, 2002. http://www.eng.uminho.pt/∼dps/aivaz/.

[123] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Optimal signal sets via semi-infinite programming. *Investigação Operacional*, 22(1):87–101, 2002.

[124] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Combining an exponential multiplier method with a constraint transcritpion strategy for semi-infinite programming. In *CD-ROM of the XXXV SBPO (Simpósio Brasileiro de Pesquisa Operacional)*, 2003.

[125] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. A quasi-newton interior point method for semi-infinite programming. *Optimization Methods and Software*, 18(6):673–687, 2003.

[126] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. A sequential quadratic method with a dual parametrization approach to nonlinear semi-infinite programming. *TOP*, 11(1):109–130, 2003.

[127] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. Robot trajectory planning with semi-infinite programming. *European Journal of Operational Research*, 53(3):607–617, 2004.

[128] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. SIPAMPL: Semi-infinite programming with AMPL. *ACM Transactions on Mathematical Software*, 30(1):47–61, 2004.

[129] A.I.F. Vaz, E.M.G.P. Fernandes, e M.P.S.F. Gomes. NSIPS v2.1: Nonlinear Semi-Infinite Programming Solver. Technical Report ALG/EF/5-2002, Universidade do Minho, Braga, Portugal, 2002. http://www.norg.uminho.pt/aivaz/.

[130] A.I.F. Vaz e E.C. Ferreira. Air pollution control with semi-infinite programming. *Applied Mathematical Modelling*, 33:1957–1969, 2009.

[131] A.I.F. Vaz e L.N. Vicente. A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, 39:197–219, 2007.

[132] A.I.F. Vaz e L.N. Vicente. Pswarm: A hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods and Software*, 24:669–685, 2009.

[133] Y.V. Volkov e S.K. Zavriev. A general stochastic outer approximations method. *SIAM Journal on Control and Optimization*, 35(4):1387–1421, 1997.

[134] F.G. Vásquez, J.-J.Rückmann, O. Stein, e G. Still. Generalized semi-infinite programming: A tutorial. *Journal of Computational and Applied Mathematics*, 217(2):394–419, 2008.

[135] D. Wang e S.-C. Fang. A semi-infinite programming model for Earliness/Tardiness production planning with a genetic algorithm. *Computers and Mathematics with Applications*, 31(8):95–106, 1996.

[136] M.-H. Wang e Y.-E. Kuo. A perturbation method for solving linear semi-infinite programming problems. *Computers and Mathematic with Applications*, 37:181–198, 1999.

[137] G.A. Watson. Globally convergent methods for semi-infinite programming. *BIT*, 21:362–373, 1981.

[138] G.A. Watson. Numerical experiments with globally convergent methods for semi-infinite programming problems. In [18], pages 193–205, 1983.

[139] G.-W. Weber. Generalized semi-infinite optimization: On some foundations. *Vychislitel'nye Teknologii*, 4:41–61, 1999.

[140] R. Werner. Cascading - an adjusted exchange method for robust conic programming. *Central European Journal of Operations*, 16(2):179–189, 2008.

[141] M.A. Wolfe. *Numerical Methods for Unconstrained Optimization, an Introduction.* Van Nostrand Reinhold Company, 1978.

[142] S.-Y. Wu e S.-C. Fang. Solving convex programs with infinitely many constraints by a relaxed cutting plane method. *Computers and Mathematics with Applications*, 38:23–33, 1999.

[143] S.-Y. Wu, D.-H. Li, L. Qi, e G. Zhou. An iterative method for solving KKT system of the semi-infinite programming. *Optimization Methods and Software*, 20(6):629–643, 2005.

[144] J.L. Zhou e A.L. Tits. An SQP algorithm for finely discretized continuous minimax problems and other minimax problems with many objective functions. *SIAM Journal on Optimization*, 6(2):461–487, 1996.