UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Métodos de Pesquisa Directa: Optimização Não Linear

Aldina Isabel de Azevedo Correia

Tese Submetida para a obtenção do grau de
Doutor em Ciências Matemáticas

Escola de Ciências e Tecnologia
Departamento de Matemática

Outubro de 2010

"Mesmo as noites totalmente sem estrelas
podem anunciar a aurora de uma grande realização."

Martin Luther King

Tese Submetida à Universidade de Trás-os-Montes e Alto Douro
para obtenção do grau de Doutor em Ciências Matemáticas
sob orientação do
Prof. Doutor João Luís Honório Matias
(Professor Auxiliar da Escola de Ciências e Tecnologia)
e co-orientação do
Prof. Doutor Pedro Miguel Mestre Alves da Silva
(Professor Auxiliar da Escola de Ciências e Tecnologia)

Ao Rogério ...

Resumo

Os Problemas de Optimização aparecem frequentemente em diversas áreas tais como a Engenharia, Economia, Química, entre outras. Nestas áreas aparecem usualmente Problemas onde as funções envolvidas (função objectivo e restrições) podem ser não suaves, as suas derivadas não são conhecidas, têm expressões complexas ou até casos em que as suas expressões analíticas não podem ser determinadas, seja pela sua complexidade ou pelo seu custo (monetário, computacional, temporal,...). Nestes casos os métodos que usam derivadas não são os mais apropriados para os resolver e os métodos que usam modelos para aproximar as funções mostram-se muitas vezes ineficazes.

Neste trabalho estudam-se, implementam-se e comparam-se Métodos de Pesquisa Directa, isto é, métodos que usam apenas informação sobre os valores das funções, progredindo em direcção à solução óptima, comparando estes valores em determinados pontos, sem recorrer ao uso de derivadas, suas aproximações ou modelos que aproximem as funções envolvidas.

Inicialmente será feita a apresentação de uma síntese sobre os métodos propostos na literatura da especialidade. Estes métodos serão posteriormente implementados e testadas algumas modificações, tendo em vista à melhoria da sua eficiência.

No que respeita à Optimização sem Restrições foram estudados os métodos clássicos de Pesquisa Directa e apresentam-se novas metodologias, adoptadas de desenvolvimentos recentes nesta área, tendo os correspondentes algoritmos sido implementados, analisados e comparados.

O mesmo sucedeu para os Métodos de Optimização de Problemas com Restrições, para os quais se adaptaram e apresentam alternativas de melhoria de métodos já usados na Optimização por Pesquisa Directa, como é o caso dos Métodos de Penalidade e Barreira. São também desenvolvidas técnicas que se consideram como possíveis alternativas de resolução deste tipo de problemas, como é o caso do Método dos Filtros, que dispensando a criação e uso de uma função de Penalidade/Barreira, bem como a escolha de parâmetros de penalidade, se mostrou como uma alternativa válida.

A implementação destes algoritmos, com recurso à Tecnologia Java, correspondeu ao desenvolvimento de uma API que foi usada para realizar os testes numéricos e onde se encontram implementados os algoritmos e variantes aqui propostos.

Palavras-Chave: Optimização não Linear, Métodos de Pesquisa Directa, Métodos de Penalidade ou Barreira, Método dos Filtos.

Abstract

Nonlinear Optimization Problems are usual in many areas such as Engineering, Economics, Chemistry, among others. In these areas it often appears Optimization Problems in which the involved functions (objective and/or constraints) might be non smooth, its derivatives are not know or have complex expressions, or even cases where their analytical expressions can not be determined either due to its complexity or its cost (monetary, computational, time,...). Thus derivative based methods are not the most appropriate for its resolution, and methods that use models to approximate such functions turn out to be sometimes ineffective.

In this work were studied, implemented and compared Direct Search Methods, i.e., methods that only need information about the functions values. They advance towards an optimal solution based on the comparison of the functions values in several points, without using derivatives or their approximations or models that approximate the involved functions.

Initially it will be presented an overview of methods proposed in the literature. These methods will then be implemented and some modifications are going to be tested in order to improve their efficiency.

For Unconstrained Optimization the classical methods were studied and new methodologies were presented. These methodologies have been presented in recent developments in this area, and the corresponding algorithms were implemented, analysed and compared.

The same is applied to the Constrained Optimization, for which were adapted and presented some alternatives with the goal to improve methods already used in Optimization by Direct Search, such asPenalty and Barrier Methods. Some techniques, which were considered as possible alternatives to solve this kind of problems, such as Filter Methods, that avoids the use of Penalty/Barrier functions and the choice of penalty parameters, were also developed.

The implementation of these algorithms, using Java Technology, corresponded to the development of an API which was used to perform numerical tests and where the algorithms and its proposed variants are implemented.

Key-Words: Nonlinear Optimization, Direct Search Optimization, Penalty or Barrier Methods, Filter Methods.

Agradecimentos

Agradeço ao Instituto Politécnico do Porto pela atribuição da bolsa PROTEC/PRODOC (Programa de Bolsas de Doutoramento do IPP). A dispensa parcial de serviço docente durante o último ano de execução deste trabalho foi essencial para a sua conclusão.

Agradeço particularmente à Escola Superior de Tecnologia e Gestão de Felgueiras por ter sustentado a atribuição dessa bolsa e me ter reconhecido aptidão para concretizar os objectivos a que me propus. Queria ainda deixar uma palavra de apreço em especial à sua Presidência pela confiança e apoio constante em assuntos relacionados com a boa execução deste trabalho.

Não podia também deixar de agradecer à Escola Superior de Estudos Industriais e de Gestão pelo apoio financeiro na fase inicial deste trabalho para a participação nos primeiros congressos internacionais.

Aos meus orientadores, Prof. Doutor João Matias e Prof. Doutor Pedro Mestre, deixo o meu agradecimento pelas pessoas fantásticas que são, pelo apoio e confiança que sempre depositaram em mim, pela paciência com as minhas infindáveis questões, pela partilha de momentos de desespero e pela conquista de momentos de sucesso e por tudo o que com eles aprendi, cientificamente e como ser humano.

Agradeço ainda aos Prof. Doutor Carlos Serôdio pela sua confiança e disponibilidade quando aceitou fazer parte deste projecto, pelas constantes palavras de apoio e pela cooperação na execução e revisão de trabalhos produzidos.

Aos meus alunos e ex-alunos agradeço pela paciência, em momentos de grande tensão, na realização deste trabalho, pela compreensão e palavras de amizade em momentos mais difíceis, pelas coisas que me ensinam e pelas emoções que me permitem viver, mas principalmente por me fazerem sentir tão realizada profissionalmente quando mostram a sua capacidade de aprender comigo.

Aos meus colegas um especial reconhecimento pela sua paciência com a minha espontaneidade e com as minhas dúvidas usuais de interdisciplinaridade.

Aos meus amigos agradeço por acreditarem que sou capaz, por me fazerem sorrir, por me apoiarem sempre... por existirem. Um especial agradecimento à minha amiga Salomé que com todo o amor que lhe é natural se ofereceu e fez questão de ler a minha tese, ainda numa fase muito embrionária e que com as suas dúvidas apontadas a lápis nos recantos das páginas impressas me mostrou que não estava sozinha.

Á minha família agradeço por viver comigo a grande vontade de concretizar mais este objectivo, adoptando-o como se fosse um objectivo próprio, partilhando todas as conquistas e hesitações como sendo suas. Em especial agradeço aos meus pais que me deram tudo, ensinaram-me a viver, a estabelecer as minhas metas, a concretizá-las, a ser forte e persistente, a querer ser sempre mais e a procurar sempre novas metas para atingir. São sem dúvida o meu exemplo de vida e reconheço cada um dos seus esforços para que eu tenha o melhor. Muito obrigada por estarem sempre presentes! Ainda ao meu mano e à minha avó um agradecimento com o abraço fechado que têm sempre disponível para mim.

E por último, mas não menos importante, agracio com uma grande admiração o meu marido Rogério por ser o meu alicerce, a base de todas as minhas construções e conquistas, pela incondicional compreensão e capacidade de se adaptar às especificidades de cada dia e principalmente pela sua grande contribuição para que eu seja uma pessoa muito feliz.

Índice

Resumo vii

Abstract ix

Agradecimentos xi

Índice de Figuras xvii

Índice de Tabelas xix

Lista de Abreviaturas xxiii

I Introdução 1

Enquadramento . . . 3
Objectivos . . . 6
Metodologia . . . 7
Estrutura . . . 8

II Optimização não linear 9

1 Introdução à Optimização não Linear 11
1.1 Problemas de Optimização . . . 11
1.2 Formulação Matemática do Problema . . . 13
1.3 Características Gerais do Problema . . . 15
1.3.1 Número de Funções Objectivo . . . 15
1.3.2 Dimensão do Problema . . . 15
1.3.3 Restrições do Problema . . . 16
1.3.4 Natureza das variáveis envolvidas . . . 17
1.3.5 Características da Função Objectivo e das Restrições . . . 17
1.3.5.1 Linearidade . . . 17
1.3.5.2 Convexidade . . . 19

1.3.5.3 Continuidade e Diferenciabilidade 20
1.3.6 Analisador de Problemas 20
1.4 Classificação de Mínimos 21
1.5 Condições de Optimalidade 22
1.5.1 Condições de Optimalidade para Problemas de Optimização sem Restrições 24
1.5.2 Condições de Optimalidade para Problemas de Optimização com Restrições 27
1.5.2.1 Notações/Definições 27
1.5.2.2 Condições de qualificação das restrições 29
1.5.2.3 Condições Optimalidade 30
1.6 Métodos de Optimização 32
1.6.1 Métodos de Optimização sem Restrições 36
1.6.2 Métodos de Optimização com Restrições 39
1.7 Critérios de Paragem 41

2 Optimização não Linear sem Restrições 47
2.1 Introdução 47
2.2 Métodos de Pesquisa Directa Clássicos 50
2.3 Métodos de Pesquisa em Padrão 51
2.3.1 Caracterização 51
2.3.2 Método de Pesquisa Coordenada 53
2.3.3 Convergência e Variantes 55
2.3.3.1 Algoritmo de Hooke e Jeeves 56
2.3.3.2 Outras Variantes 57
2.4 Métodos Simplex 62
2.4.1 Caracterização 62
2.4.2 Método de Nelder-Mead 64
2.4.2.1 Descrição geral do Método de Nelder-Mead 64
2.4.3 Convergência e Variantes 68
2.5 Métodos com Conjuntos de Direcções de Pesquisa Adaptativas 78

3 Optimização não Linear com Restrições 81
3.1 Introdução 81
3.2 Métodos de Penalidade e Barreira 83
3.2.1 Métodos de Barreira 86
3.2.1.1 Função Barreira Extrema e Progressiva 87
3.2.1.2 Método de Barreira Inversa 89
3.2.1.3 Método de Barreira Logarítmica 89
3.2.2 Métodos de Penalidade 90
3.2.2.1 Funções de Penalidade Clássicas 91
3.2.2.2 Métodos de Penalidade Exacta 92
3.2.2.3 Classificação dos Métodos de Penalidade existentes 93
Métodos de Penalidade de Optimização Global – GOPM . . 94
Métodos de Penalidade de Optimização Local – LOPM . . . 99
Uma nova classe de Métodos de Penalidade Dinâmica 101
3.3 Lagrangeana Aumentada 103

3.4 Método dos Filtros 106

4 Tecnologia de Programação 115
4.1 Escolha da Tecnologia de Programação 115
4.2 Criação de Programas em Java 116
4.3 História do Java 117
4.4 Algumas Características da linguagem Java 118
4.5 Implementação doa Algoritmos em Java 119

III Desenvolvimento e Implementação dos Algoritmos 121

5 Desenvolvimento e Implementação dos Algoritmos 123
5.1 Apresentação Geral do Trabalho Desenvolvido 123
5.2 Métodos Implementados para Resolução de Problemas Sem Restrições . . 124
5.3 Métodos Implementados para Resolução de Problemas Com Restrições . . 126
5.4 Implementação 131

6 Pormenores de Implementação dos Algoritmos para Optimização sem Restrições 133
6.1 Algoritmo de Pesquisa Coordenada 134
6.2 Algoritmo de Hooke e Jeeves 136
6.3 Uma versão dos algoritmos de Audet et. al. 138
6.4 Algoritmo de Nelder-Mead 140
6.5 Algoritmo Simplex Convergente 144
6.6 Resultados Numéricos 147
6.6.1 Problemas 147
6.6.2 Comparação de Resultados e Adequação de Parâmetros de Entrada 147
6.6.2.1 Problemas de Bagirov 149
6.6.2.2 Problemas de Schittkowski 156
6.6.2.3 Problemas de Maratos 161
6.6.2.4 Análise final dos Resultados Numéricos 163

7 Pormenores de Implementação dos Algoritmos para Optimização com Restrições 169
7.1 Métodos de Penalidade e Barreira 170
7.2 Método dos Filtros 177
7.3 Resultados Numéricos 190
7.3.1 Problemas 190
7.3.2 Comparação de Resultados 191
7.3.2.1 Problemas da Colecção Cute 196
7.3.2.2 Problema PA 209
7.3.2.3 Problemas de Schittkowski 210
7.3.2.4 Análise final dos Resultados Numéricos 234

8 API - *Application Programming Interface* 247
8.1 Estrutura Geral da API 248
8.1.1 Problemas com Restrições 253

8.1.2 Problemas sem Restrições 262
8.2 Utilização da API 268

IV Conclusões, Principais Contribuições e Trabalhos Futuros 271

9 Conclusões, Principais Contribuições e Trabalhos Futuros 273
9.1 Conclusões e Considerações Finais 273
9.2 Sumário das Principais Contribuições 279
9.3 Trabalho Futuro 281

A Problemas 283
A.1 Problemas sem Restrições 284
A.1.1 Problemas de Bagirov 284
A.1.2 Problemas de Schittkowski 286
A.1.3 Problemas de Maratos 288
A.2 Problemas com Restrições 289
A.2.1 Problemas CUTE 289
A.2.2 Problema PA 289
A.2.3 Problemas de Schittkowski 290

Bibliografia 295

Índice de Figuras

1.1 Ciclo de Modelação e Resolução de um Problema de Optimização 12
1.2 Exemplos de minimizantes globais e locais a uma dimensão 22
1.3 Árvore de Sub-áreas de Optimização 34
1.4 Árvore de alguns Métodos de Optimização 34

2.1 Alguns Exemplos de Padrões ou Grelhas 51
2.2 Padrão do Método de Pesquisa Coordenada 54
2.3 Passo Padrão – Algoritmo de Hooke e Jeeves 56
2.4 Movimento de reflexão de Spendley et. al. 63
2.5 Movimentos de Nelder-Mead 64

3.1 Métodos de Penalidade/Barreira – Processos iterativos interno e externo . 84
3.2 Uma Classificação dos Métodos de Penalidade existentes 93
3.3 Um Filtro com quatro pontos 110
3.4 Envelope do Filtro 112

4.1 Processo de criação, compilação e execução dum programa em Java 116

5.1 Métodos de Optimização Implementados 124

6.1 Algoritmo de Pesquisa Coordenada 135
6.2 Algoritmo de Hooke e Jeeves 137
6.3 Versão implementada dos algoritmos de Audet et. al. 141
6.4 Algoritmo de Nelder-Mead implementado 142
6.5 Algoritmo Simplex Convergente 145
6.6 Pontos de decisão do Algoritmo Simplex Convergente 146

7.1 Algoritmo de Penalidade/Barreira Implementado 172
7.2 Critérios de Aceitação no Filtro 179
7.3 Funcionamento Geral do Método dos Filtros 181
7.4 Representação gráfica de alguns pontos de SFA e NSFA 183
7.5 Funcionamento Geral do Método dos Filtros Generalizado 184
7.6 Algoritmo dos Filtros Implementado 186
7.7 Iterações para o Problema C801, usando o MF 200
7.8 Filtro para o Problema C801 201
7.9 Posição das aproximações à solução – MF/CS (Problema C801) 202
7.10 Comportamento de f e h – MF/CS (Problema C801) 203
7.11 Comportamento de f e h – CP/HJ (Problema C801) 204
7.12 Iterações para o Problema C802, usando o MF 207

7.13 Filtro para o Problema C802 207
7.14 Comportamento de f e h – MF/NM (Problema C802) 208
7.15 Filtro para o Problema S226 216
7.16 Filtro para o Problema S226, com $h_{max} = 2$ 217
7.17 Posição das aproximações à solução – MF/SC (Problema S227) 219
7.18 Posição das aproximações à solução – MF/HJ (Problema S228) 221
7.19 Posição das aproximações à solução – MF/HJ (PS228 e $\alpha = 2$) 222
7.20 Posição das aproximações à solução – MF/AA (Problema S234) 225
7.21 Comportamento de f e V – PenBar/SC (Problema S264) 228
7.22 Comportamento de f e h – MF/SC (Problema S264) 229

8.1 Estrutura Geral da API 248
8.2 Escolher um problema guardado 248
8.3 API/GUI - Definição de Problema sem Restrições 249
8.4 API/GUI - Definição de Problema com Restrições 250
8.5 Interface Problema Geral 251
8.6 Interface Constraint 252
8.7 Opção escrever um novo problema 252
8.8 Problema a resolver 253
8.9 Métodos para Optimização com Restrições 254
8.10 API/GUI - Opções para Optimização com Restrições 254
8.11 API/GUI - Opções para o método a usar no processo interno 255
8.12 API/GUI - Opções para a Função de Penalidade/Barreira 256
8.13 API/GUI - Painéis Correspondentes às Função de Penalidade/Barreira . . 257
8.14 API/GUI - Apresentação dos Resultados - Optimização com Restrições . . 258
8.15 API/GUI - Opções para a medida de h 259
8.16 API/GUI - Opções para o método a usar no processo interno 259
8.17 API/GUI - Painéis Correspondentes as medidas no Método dos Filtros . . 261
8.18 Métodos para Optimização sem Restrições 262
8.19 API/GUI - Painéis Correspondentes aos Métodos de Pesquisa Directa . . 263
8.20 API/GUI - Opções de execução do algoritmo de Pesquisa Directa 264
8.21 API/GUI - Apresentação dos Resultados - Optimização sem Restrições . . 267
8.22 Interface da aplicação Web para Optimização sem Restrições 269
8.23 Interface da aplicação Web para Optimização com Restrições 270

Índice de Tabelas

2.1 Algoritmo de Pesquisa Coordenada 55
2.2 Algoritmo de Hooke e Jeeves 58
2.3 Algoritmo de Audet 61
2.4 Algoritmo de Nelder-Mead 69
2.5 Variante do Algoritmo de Nelder-Mead Globalmente Convergente 76

3.1 Algoritmo do Método de Penalidade ℓ_1 Clássico 101
3.2 Algoritmo do Método dos Filtros genérico 111
3.3 Algoritmo do Método dos Filtros de Audet 113

6.1 Resultados para o Problema B1 149
6.2 Resultados para o Problema B2 150
6.3 Resultados para o Problema B3 151
6.4 Resultados para o Problema B4 152
6.5 Resultados para o Problema B5 153
6.6 Resultados para o Problema B6 154
6.7 Resultados para o Problema B7 154
6.8 Resultados para o Problema B8 155
6.9 Resultados para o Problema B9 155
6.10 Resultados para o Problema B10 156
6.11 Resultados para o Problema S201 157
6.12 Resultados para o Problema S202 157
6.13 Resultados para o Problema S205 157
6.14 Resultados para o Problema S206 157
6.15 Resultados para o Problema S207 158
6.16 Resultados para o Problema S208 158
6.17 Resultados para o Problema S211 158
6.18 Resultados para o Problema S212 159
6.19 Resultados para o Problema S213 159
6.20 Resultados para o Problema S240 159
6.21 Resultados para o Problema S246 160
6.22 Resultados para o Problema S256 161
6.23 Resultados para o Problema S257 161
6.24 Resultados para o Problema M500 162
6.25 Resultados para o Problema M501 162
6.26 Qualidade das soluções encontradas com os parâmetros por defeito 163
6.27 Qualidade das soluções encontradas com os parâmetros alterados 164
6.28 Melhores valores para $f(x_k)$, usando os parâmetros por defeito 165

6.29 Melhores valores para $fEvals$, usando os parâmetros por defeito 165
6.30 Melhores valores para k, usando os parâmetros por defeito 166
6.31 Melhores valores para $f(x_k)$, $fEvals$ k em simultâneo 166

7.1 Função a optimizar no Processo Interno 171
7.2 Parâmetros usados no processo Externo 173
7.3 Resultados da primeira versão dos algoritmos PenBar 176
7.4 Resultados da versão NSFA em comparação com a anterior SFA 182
7.5 Medidas de h 187
7.6 Medidas para h adaptadas e testadas 194
7.7 Resultados para o Problema C801, usando PenBar 197
7.8 Resultados para o Problema C801, usando MF 199
7.9 Resultados para o Problema C802, usando PenBar 205
7.10 Resultados para o Problema C802, usando MF 206
7.11 Resultados para o Problema PA, usando PenBar 209
7.12 Resultados para o Problema PA, usando MF 210
7.13 Resultados para o Problema S224, usando PenBar 211
7.14 Resultados para o Problema S224, usando MF 211
7.15 Resultados para o Problema S224, usando MF com 100 iterações 212
7.16 Resultados para o Problema S225, usando PenBar 213
7.17 Resultados para o Problema S225, usando MF 214
7.18 Resultados para o Problema S226, usando PenBar 214
7.19 Resultados para o Problema S226, usando MF 215
7.20 Resultados para o Problema S226, usando o MF com $h_{max} = 2$ 216
7.21 Resultados para o Problema S227, usando PenBar 218
7.22 Resultados para o Problema S227, usando MF 218
7.23 Resultados para o Problema S228, usando PenBar 220
7.24 Resultados para o Problema S228, usando MF 220
7.25 Resultados para o Problema S231, usando PenBar 221
7.26 Resultados para o Problema S231, usando MF 222
7.27 Resultados para o Problema S233, usando PenBar 223
7.28 Resultados para o Problema S233, usando MF 223
7.29 Resultados para o Problema S234, usando PenBar 224
7.30 Resultados para o Problema S234, usando MF 224
7.31 Resultados para o Problema S249, usando PenBar 226
7.32 Resultados para o Problema S249, usando MF 226
7.33 Resultados para o Problema S264, usando PenBar 227
7.34 Resultados para o Problema S264, usando MF 227
7.35 Resultados para o Problema S270, usando PenBar 229
7.36 Resultados para o Problema S270, usando MF 230
7.37 Resultados para o Problema S323, usando PenBar 230
7.38 Resultados para o Problema S323, usando MF 231
7.39 Resultados para o Problema S324, usando PenBar 231
7.40 Resultados para o Problema S324, usando MF 232
7.41 Resultados para o Problema S325, usando PenBar 232
7.42 Resultados para o Problema S325, usando MF 233
7.43 Resultados para o Problema S326, usando PenBar 233

7.44 Resultados para o Problema S326, usando MF 234
7.45 Qualidade das soluções encontradas usando o algoritmo PenBar 235
7.46 Qualidade das soluções encontradas usando o Método EB 235
7.47 Qualidade das soluções encontradas usando o Método PB 236
7.48 Qualidade das soluções encontradas usando o Método CP 237
7.49 Qualidade das soluções encontradas usando o Método DP 237
7.50 Qualidade das soluções encontradas usando o Método ℓ_1 237
7.51 Qualidade das soluções encontradas usando PenBar 239
7.52 Qualidade das soluções encontradas usando o MF 241
7.53 Qualidade das soluções encontradas usando a medida N1 242
7.54 Qualidade das soluções encontradas usando a medida N2 242
7.55 Qualidade das soluções encontradas usando a medida NP 243
7.56 Qualidade das soluções encontradas usando a medida NEB 243
7.57 Qualidade das soluções encontradas usando o MF 244

Lista de Abreviaturas

AA	Versão dos algoritmos de **A**udet et. **a**l.
ALM	Métodos de Lagrangeana Aumentada (**A**ugmented **L**agrangian **M**ethods)
API	**A**pplication **P**rogramming **I**nterface
BM	Métodos de Barreira (**B**arrier **M**ethods)
CS	Algoritmos de Procura Coordenada (**C**ordinate **S**earch)
CGM	Mínimo Global de um Problema com Restrições (**C**onstrained **G**lobal **M**inimum)
CLM	Mínimo Local dum Problema com Restrições (**C**onstrained **L**ocal **M**inimum)
CP	Penalidade Clássica (**C**lassical **P**enalty)
DP	Penalidade Dinâmica (**D**ymanic **P**enalty)
EB	Barreira Extrema (**E**xtreme **B**arrier)
GOPM	Métodos de Penalidade de Optimização Global (**G**lobal **O**ptimal **P**enalty **M**ethods)
GUI	**G**raphical **U**ser **I**nterface
FSM	Máquinas de Estados Finitos (**F**inite **S**tate **M**achines)
FSQP	Programação Quadrática Sequencial Admissível (**F**easible **S**equential **Q**uadratic **P**rogramming)
GPS	Procura em Padrão Generalizada (**G**eneralized **P**attern **S**earch)
GSS	Procura com Conjuntos Geradores (**G**enerating **S**et **S**earch)
GRNMA	Algoritmo de Nelder-Mead com Rede (**G**rid **R**estrained **N**elder-**M**ead **A**lgorithm)
HJ	Algoritmo de **H**ooke e **J**eeves
JDK	**J**ava **D**evelopment **K**it
JVM	Máquina Virtual Java (**J**ava **V**irtual **M**achine)

KKT	**K**arush-**K**uhn-**T**ucker
ℓ_1	Penalidade ℓ_1
LICQ	Condição de Independência Linear
	(**L**inear **I**ndependence **C**onstraint **Q**ualification)
LOPM	Métodos de Penalidade de Optimização Local
	(**L**ocal **O**ptimal **P**enalty **M**ethods)
LP	Problemas de Optimização Linear (**L**inear **P**rogramming)
MADS	Procura Directa com Grelha Adaptativa
	(**M**esh **A**daptive **D**irect **S**earch)
MASD	Métodos com Conjuntos de Direcções de Pesquisa Adaptativas
	(**M**ethods with **A**daptive **S**ets of Search **D**irections)
MF	**M**étodo dos **F**iltros (Filters Method)
MFCQ	Condição de Mangasarian-Fromovitz
	(**M**angasarian-**F**romovitz **C**onstraint **Q**ualification)
N1	**N**orma 1
N2	**N**orma 2
NCP	**N**orma/Medida Penalidade Clássica (**C**lassical **P**enalty)
NEB	**N**orma/Medida Barreira Extrema (**E**xtreme **B**arrier)
Nℓ_1	**N**orma/Medida Penalidade ℓ_1
NP	**N**orma/Medida que representa as medidas
	Barreira Progressiva, Penalidade Clássica e Penalidade Estática/Dinâmica
NPB	**N**orma/Medida Barreira Progressiva (**P**rogressive **B**arrier)
NSDP	**N**orma/Medida Penalidade Estática/Dinâmica (**S**tatic/**D**ynamic **P**enalty)
NEOS	**N**etwork-**E**nabled **O**ptimization **S**ystem
NLP	Problemas de Optimização não Linear (**N**on**l**inear **P**rogramming)
NM	Algoritmo de **N**elder-**M**ead
NSFA	**N**ew **S**implex **F**ilter **A**lgorithm
	(Novo Algoritmo Filtro Simplex)
OSD	**O**ptimização **S**em **D**erivadas
PB	Barreira Progressiva (**P**rogressive **B**arrier)
PenBar	**P**enalidade/**B**arreira
PM	Métodos de Penalidade (**P**enalty **M**ethods)
PS	Procura em Padrão (**P**attern **S**earch)

PSM	Métodos de Pesquisa em Padrão (**P**attern **S**earch **M**ethods)
RGM	Métodos de Gradiente Reduzido (**R**educed-**g**radient **M**ethods)
SC	Algoritmo **S**implex **C**onvergente
SCQ	Condição de Slater (**S**later's **C**onstraint **Q**ualification)
SDNMA	Algoritmo de Nelder-Mead com Descida Suficiente (**S**uficient **D**escent **N**elder-**M**ead **A**lgorithm)
SFA	**S**implex **F**ilter **A**lgorithm (Algoritmo Filtro Simplex)
SM	Métodos Simplex (**S**implex **M**ethods)
SP	Penalidade Estática (**S**tatic **P**enalty)
SQP	Programação Quadrática Sequencial (**S**equential **Q**uadratic **P**rogramming)

Parte I

Introdução

Enquadramento

A Optimização, também designada por Programação Matemática, é muito utilizado em processos de tomada de decisão. Nestes processos pretende determinar-se de que forma se pode dar a melhor utilização aos recursos disponíveis, no sentido de obter os melhores resultados para uma dada realidade. Deste modo, a Optimização tem sido usada durante séculos nas mais diversas áreas, podendo destacar-se a Engenharia, a Gestão, a Medicina, a Estatística, entre outras.

A Optimização requer, então, a Modelação Matemática da realidade e a procura de um óptimo. Os modelos que definem um Problema de Optimização são constituídos por uma (ou várias, no caso da optimização multi-objectivo) função que se pretende maximizar ou minimizar (optimizar), usualmente designada por *função objectivo*, de uma ou várias variáveis, também designadas por *variáveis de decisão*. Essa função a optimizar pode, potencialmente, estar sujeita a algumas condições impostas às variáveis. Nesta situação o Problema de Optimização terá ainda um conjunto de funções que definem as referidas condições, ditas *funções restrição* do problema.

Feita a formulação matemática do problema, torna-se necessário encontrar métodos capazes de o resolver. Assim, têm sido desenvolvidos métodos para determinar o óptimo (máximo ou mínimo da função objectivo). Infelizmente, dada a complexidade da realidade, nem todos os problemas podem ser resolvidos analiticamente, ou seja, usando directamente as *condições de optimalidade*, pelo que se têm desenvolvido *métodos iterativos* para encontrar aproximações à solução o mais próximas possível da solução desejada.

A maioria das vezes, em problemas reais, seria impossível fazer a formulação matemática perfeita e encontrar a solução perfeita. No caso de problemas muito complexos ou com variáveis que mudam constantemente, quando se obtivesse a solução ela já estaria desadequada da nova realidade. Assim, é preferível obter boas aproximações que possam ser úteis em tempo real, pelo que se procuram algoritmos eficientes e robustos, que conduzam a uma boa aproximação da solução do problema em tempo útil, pelo que a rapidez de execução computacional também é crucial.

De acordo com as características da função objectivo e das restrições, os Problemas de Optimização podem ser divididos em duas grandes categorias: *Problemas de Optimização Linear*, para os quais as funções envolvidas são lineares e *Problemas de Optimização não*

Linear onde a função objectivo e/ou as restrições podem ser funções não lineares nas variáveis.

Os Problemas de Optimização Linear são, em geral, mais simples de resolver, no sentido em que os métodos existentes para a sua resolução são eficientes e robustos, no entanto no caso dos Problemas de Optimização não Linear isso nem sempre acontece, sendo impossível indicar um método que seja ideal para resolver todos os problemas. Para a resolução deste tipo de problemas existem, assim, métodos muito diferenciados, adequados para a resolução de diferentes tipos de problemas. Nesta classe alguns problemas são mais fáceis de resolver, nomeadamente se forem Problemas Quadráticos [1], Convexos [2] ou Problemas sem Restrições. Mas se o problema tem restrições e não é nenhum dos casos anteriores, a capacidade de assegurar a convergência para a solução global é uma tarefa muito difícil. Existem ainda outras características que podem distinguir os problemas, nomeadamente a diferenciabilidade e a continuidade das funções envolvidas, a dimensão do problema (número de variáveis), a natureza das variáveis (discreta ou contínua), o número de restrições e o número de funções objectivo (uni-objectivo ou multi-objectivo). Esta diversidade de problemas requer uma escolha adequada de métodos de resolução.

Neste trabalho abordam-se métodos que permitem resolver problemas onde a função objectivo pode ser não suave [3], não linear, descontínua, não convexa e com muitos mínimos locais.

Este tipo de problemas surge em diversas situações reais, por exemplo, quando os valores da função objectivo ou das restrições são resultado de um complexo e demorado processo de simulação, quando são obtidos por observação em experiências ou em fenómenos naturais, quando têm expressões analíticas complexas ou não estão disponíveis, quando são afectadas por ruído, quando as suas derivadas não estão definidas em determinados pontos ou são difíceis de calcular, entre outras, pelo que é de extrema importância desenvolver métodos que os permitam resolver.

Os métodos baseados em derivadas não são, deste modo, os métodos mais adequados para a sua resolução, pelo que será necessário utilizar nestas situações Métodos de Pesquisa Directa, ou seja, Métodos de Optimização não Linear sem Restrições, que não utilizem

[1] Problemas Quadráticos são problemas onde a função objectivo é uma função quadrática e as restrições são todas lineares.

[2] Problemas Convexos são problemas para os quais as funções envolvidas são todas funções convexas.

[3] Uma função diz-se *suave* se é continuamente diferenciável até à segunda ordem.

derivadas nem aproximam as funções envolvidas através de modelos. Estes métodos só utilizam nos seus algoritmos informação relativa ao valor da função objectivo, progredindo em direcção ao óptimo com base na comparação dos valores da função objectivo em diversos pontos. Estes métodos também podem ser utilizados quando as derivadas da função apresentam descontinuidades em pontos na vizinhança do óptimo, quando as derivadas não existem numa vizinhança do óptimo, quando são difíceis de calcular ou quando exigem um grande esforço computacional.

As restrições do problemas também podem apresentar estas características, pelo que também serão estudados Métodos de Optimização não Linear com Restrições, sem utilização de derivadas.

Muitos dos Métodos de Optimização sem Derivadas, designados usualmente na literatura por *Derivative-free Optimization Methods* usam modelos (lineares, quadráticos, convexos, entre outros) para aproximar as funções envolvidas. Estes métodos, sendo muitas vezes bastante eficazes, têm o inconveniente de ser possível encontrar soluções óptimas dos modelos que não são óptimos da função objectivo original. Além disso, podem exigir mais avaliações das funções envolvidas do que as que seriam necessárias usando Métodos de Pesquisa Directa, uma vez que, para a construção de modelos, é necessário avaliar logo à partida o valor das funções em alguns pontos (quanto mais pontos se avaliarem é de esperar que melhor seja o modelo), e, depois de construído o modelo, é ainda necessário aplicar o respectivo método de optimização que implica ainda mais avaliações do modelo ou da própria função.

Os métodos mais utilizados para Optimização com Restrições, em conjugação com Métodos de Pesquisa Directa, são os métodos de Penalidade ou Barreira. No entanto estes métodos exigem, em geral, a escolha ou estimação de um grande número de parâmetros. Para evitar este inconveniente têm sido introduzidas novas estratégias, nomeadamente a técnica dos Filtros. Esta técnica tem vindo a ser usada em diversas áreas de Optimização com Restrições, no entanto em conjugação com Métodos de Pesquisa Directa é ainda um assunto muito pouco estudado. Deste modo é de particular interesse averiguar a aplicabilidade do Método dos Filtros neste âmbito.

Assim, este trabalho insere-se na área da Optimização não Linear com e sem Restrições, sem uso de derivadas ou suas aproximações, nem de modelos que aproximem as funções do problema. Pretende dar-se ênfase particular ao Método dos Filtros, por ser um método

ainda pouco estudado nesta área, nomeadamente no que diz respeito às suas características e formas de implementação, tendo em vista a possibilidade de aplicação enquanto Método de Pesquisa Directa para Optimização com Restrições ou como motivação para fazer adaptações a outros métodos que tratam o mesmo tipo de problemas.

Objectivos

Neste trabalho pretende estudar-se, implementar e comparar diferentes algoritmos para resolver Problemas de Optimização não Linear sem e com Restrições, sem recorrer ao uso de derivadas das funções envolvidas ou suas aproximações, nem à aproximação das funções através de modelos.

Os Métodos de Pesquisa Directa (para Optimização sem Restrições) que são mais utilizados não têm sofrido grandes desenvolvimentos, mantendo as linhas gerais desde já há algum tempo. As mais recentes abordagens sobre este assunto têm sido acerca de problemas de convergência. Neste sentido têm sido apresentadas novas abordagens, por diversos autores, para métodos já existentes, sendo apresentadas algumas sugestões de implementação com o objectivo da garantia dessa convergência. Pretende-se, assim neste aspecto, investigar essas sugestões e a sua eficiência no desempenho dos Métodos de Pesquisa Directa quando comparadas com os métodos clássicos.

No que respeita a Métodos para Optimização com Restrições pretende averiguar-se a aplicabilidade e eficiência de novas metodologias, nomeadamente do Método dos Filtros, a qual não é bem conhecida para Optimização por Pesquisa Directa.

Conhecidas as características dos métodos já implementados e estudados pretende ainda averiguar-se a possibilidade de desenvolver novas metodologias de resolução de problemas, seja adaptando procedimentos já existentes nos métodos correspondentes ou adoptando procedimentos de outros métodos.

De modo a permitir uma fácil interacção com os métodos e metodologias desenvolvidas, pretende-se que estes sejam suportados por uma aplicação de *software*, na forma de uma API (*Application Programming Interface*), desenvolvida com recurso à Tecnologia Java, onde todos os métodos e algoritmos estudados e desenvolvidos estejam implementados.

Ainda se pretende que alguns problemas teste previamente seleccionados estejam disponíveis para o utilizador da aplicação poder testar as metodologias desenvolvidas e que a progressão do processo iterativo, durante a execução, bem como os resultados obtidos possam ser visualizados no final dessa execução.

Metodologia

A investigação realizada para o presente trabalho teve início numa pesquisa sobre Optimização, por forma a fazer um enquadramento geral da área de estudo. Nesta primeira fase evidenciou-se a Formulação Matemática de Problemas de Optimização e a sua Classificação, no sentido de enquadrar o tema em estudo e conhecer a sub-área em investigação.

Tendo enquadrado o tema de estudo na sub-área de Optimização não Linear com Restrições procedeu-se ao estudo dos seus conceitos gerais, nomeadamente no que se refere à Caracterização de Soluções e às Condições de Optimalidade, conceitos essenciais à avaliação dos métodos, no que diz respeito à convergência para o óptimo.

A resolução de Problemas não Lineares com Restrições envolve o conhecimento de métodos de Optimização Não Linear sem e com Restrições. Tendo em conta que o âmbito deste trabalho prevê o estudo de métodos que não utilizam derivadas foi aprofundado o estudo dos Métodos de Pesquisa Directa (Métodos de Optimização Não Linear sem Restrições, sem uso de derivadas nem modelos), tendo-se realizado uma procura de informação mais direccionada no que se refere aos Métodos de Pesquisa em Padrão e aos Métodos Simplex, que se têm mostrado os Métodos de Pesquisa Directa considerados clássicos e mais utilizados pelos investigadores das mais diversas áreas.

O conhecimento de quais os métodos utilizados para Optimização sem Restrições permitiu prosseguir para a pesquisa dos Métodos de Optimização Não Linear com Restrições. Os métodos que nesta pesquisa se verificaram ser mais utilizados para a Optimização com Restrições foram os Métodos de Penalidade e Lagrangeana Aumentada, no caso de existirem restrições de igualdade. Por fim, e porque mais recentemente a comunidade científica tem vindo a aplicar o Método dos Filtros a Problemas com Restrições, com manifesto sucesso, também foi realizada uma pesquisa sobre este método.

Tendo em vista o objectivo da implementação dos métodos estudados, usando a Tecnologia Java, foi realizada uma pesquisa sobre esta tecnologia. Ainda foi estudada a forma de modelação através de Máquina de Estados Finitos (Finite State Machine), uma vez que se verificou que os algoritmos implementados eram compostos por estados, transições e acções, características que se adequam a este tipo de modelação.

Estrutura

O presente trabalho é constituído por quatro partes. Na Parte I é apresentado o Enquadramento geral do tema tratado, são apresentados os Objectivos Gerais, definida a Metodologia que orientou a pesquisa bibliográfica e formalizada a presente Estrutura.

Na Parte II é apresentada uma Revisão Bibliográfica. O primeiro Capítulo desta parte é respeitante a Generalidades sobre Optimização Não Linear, onde são apresentados alguns fundamentos teóricos, incluindo conceitos e definições. Os segundo e terceiro Capítulos expõem conceitos, métodos e algoritmos usados para a resolução de Problemas de Optimização não Linear, sem e com Restrições, respectivamente, sem utilização de derivadas nem de modelos. Ainda é feita uma pequena abordagem sobre a Tecnologia Java e a modelação através de Máquina de Estados Finitos, no quarto Capítulo.

Na Parte III são apresentados os pormenores de implementação dos algoritmos desenvolvidos como alternativa aos métodos usuais de resolução de Problemas não Lineares, são apresentados os resultados numéricos obtidos e é feita uma breve descrição da API desenvolvida.

Finalmente, na Parte IV são apresentadas as Considerações Finais ou Conclusões retiradas da realização deste trabalho, pretendendo-se responder aos objectivos inicialmente propostos. Ainda é apresentado um sumário das principais contribuições alcançadas com o desenvolvimento deste trabalho e são apresentadas algumas direcções para trabalhos futuros.

Parte II

Optimização não linear

Capítulo 1

Introdução à Optimização não Linear

1.1 Problemas de Optimização

A Modelação Matemática é utilizada nas mais diversas áreas, tais como Engenharia, Economia, Química, entre outras, para estudar e compreender fenómenos e resolver problemas que deles advêm. Os problemas a resolver são de tipos e graus de complexidade variados e muitas vezes aparecem como subproblemas de outros problemas de maior dimensão.

Tipicamente, em Optimização perante um problema real inicia-se um processo de modelação (ver FIGURA 1.1). Do problema real extraem-se os elementos essenciais à formulação do modelo matemático (*Hipóteses de Modelação*). Usando esses elementos formula-se o Problema de Optimização (Modelo Matemático). Formulado o problema é, então, possível procurar a melhor solução para o problema, seleccionando o Método de Optimização apropriado. Depois há que verificar se a solução verifica as hipóteses iniciais e interpretar os resultados. Se as hipóteses são verificadas e os resultados se adequam à realidade então a solução encontrada é uma boa solução, ou seja, se o modelo for suficientemente preciso, a solução é validada e pode ser aplicada ao mundo real como uma solução para o problema real, se uma das duas situações falha, então o modelo matemático deve ser revisto.

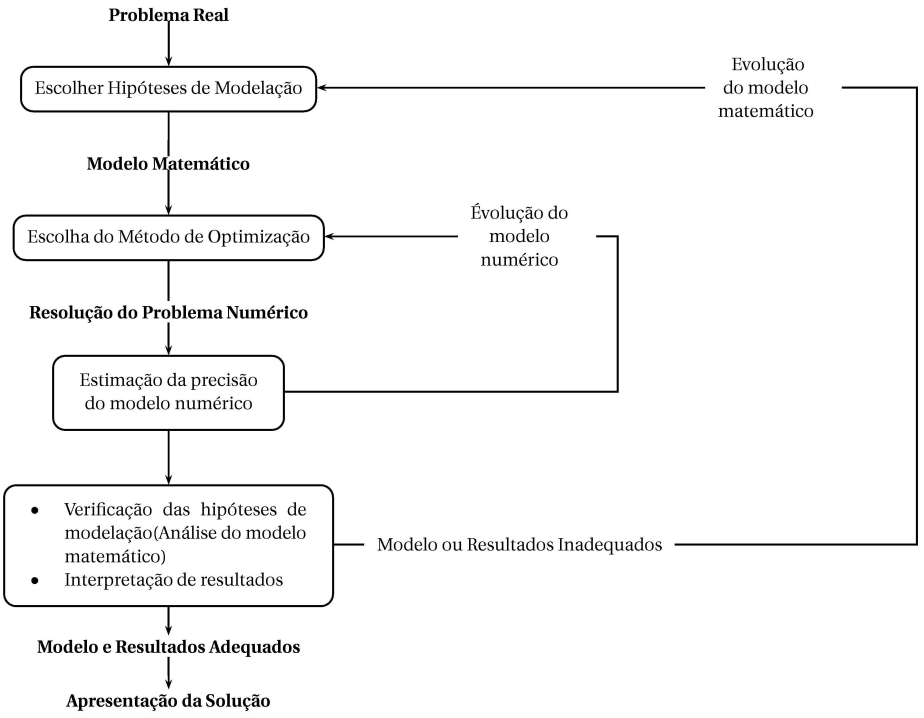

FIGURA 1.1: Ciclo de Modelação e Resolução de um Problema de Optimização

É importante notar o facto de um modelo matemático nunca ser uma representação exacta da realidade e não faz sentido considerá-lo retirado de seu contexto. É fundamental perceber que os modelos de Optimização podem ajudar a encontrar boas soluções, mas não resolvem o problema por completo. A qualidade das soluções encontradas não depende só do método de Optimização usado, mas também da qualidade da Formulação efectuada.

Os Problemas de Optimização são dos problemas que mais interesse suscitam, pois aparecem em diversas situações da vida humana e das suas actividades. Com efeito, a Optimização ocorre sempre que as pessoas, empresas ou sistemas pretendem minimizar ou maximizar algo. Assim, estes problemas aparecem frequentemente associados a processos de apoio à decisão, uma vez que a sua resolução permite encontrar soluções por forma a obter a melhor escolha de acordo com restrições existentes. A Optimização é usada, por exemplo, para projectar e optimizar a instalação de unidades industriais, onde é essencial para melhorar o desempenho económico das linhas de produção; todas as empresas têm como principal objectivo maximizar os seus lucros e minimizar os seus custos, muitos problemas na área da Medicina e da Biologia têm como objectivo maximizar ou

minimizar alguma ou várias das características de microorganismos, bactérias, animais, plantas ou objectos em observação e estudo.

Assim, um Problema de Optimização caracteriza-se por compreender uma função a optimizar, a *função objectivo*, um conjunto de variáveis ou parâmetros independentes, as *variáveis de decisão*, que afectam o valor da *função objectivo* e que geralmente estão sujeitas a algumas condições, as *restrições* do problema.

Estas condições, que estabelecem os limites das variáveis ou a forma como se relacionam, definem os *valores aceitáveis* ou *admissíveis* para essas variáveis. Os *pontos admissíveis*, são, portanto, os pontos que verificam as restrições e ao conjunto desses pontos dá-se o nome de *região admissível*.

O objectivo é, então, encontrar os valores das variáveis que minimizam ou maximizam a *função objectivo*. No caso do problema ter restrições, ainda se pretende que os valores encontrados sejam admissíveis.

1.2 Formulação Matemática do Problema

A formulação de um Problema de Optimização através de uma *função objectivo* e de possíveis restrições é um *programa ou modelo matemático* (do inglês *mathematical program*). O estudo e utilização destes modelos diz-se *programação matemática* (do inglês *mathematical programming*).

Os problemas de Optimização são diversos e com características muito distintas, mas é possível descrevê-los de uma forma geral. A forma geral de um problema de optimização pode ser encontrada em diversos textos da área. As que se apresentam aqui, são baseadas na apresentada por Fernandes em [58] e por Matias em [110].

Os *Problemas de Optimização sem Restrições* podem ser expressos matematicamente por:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1.1}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é a *função objectivo*.

Nos *Problemas de Optimização com Restrições*, a variável $x \in \mathbb{R}^n$ está sujeita a restrições. Assim, de uma forma geral, a formulação de um problema de optimização pode ser posta na forma (1.2):

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1.2a}$$

$$s.\,a.\ c_i(x) = 0, i \in \mathcal{E} \tag{1.2b}$$

$$c_i(x) \leq 0, i \in \mathcal{I} \tag{1.2c}$$

onde

- A função $f : \mathbb{R}^n \to \mathbb{R}$ (1.2a), que se pretende minimizar, é *função objectivo* e é uma função real de variáveis reais;
- x é o vector das n variáveis, $x = (x_1, x_2, ..., x_n)^T,\ x \in \mathbb{R}^n$;
- $\mathcal{E} = \{1, 2, ..., t\}$ e $\mathcal{I} = \{t+1, t+2, ..., m\}$ são dois conjuntos disjuntos de índices;
- As equações $c_i(x) = 0, i \in \mathcal{E}$ (1.2b), definem as *restrições de igualdade* do Problema;
- As equações $c_i(x) \leq 0,\ i \in \mathcal{I}$ (1.2c), representam as *restrições de desigualdade* do Problema;
- As restrições de igualdade e de desigualdade são vulgarmente conhecidas por *funções restrição do problema*;
- x deve ser admissível, isto é, deve verificar as restrições do problema (1.2b-1.2c), nesse caso, dir-se-á *ponto admissível* do problema (1.2);
- A *região admissível* do problema (1.2) é o conjunto

  $$\Omega = \{x \in \mathbb{R}^n : c_i = 0,\ i \in \mathcal{E} \ \wedge\ c_i(x) \leq 0,\ i \in \mathcal{I}\},$$

  ou seja, é o conjunto de todos os pontos admissíveis;
- O *ponto óptimo* obtido pela resolução do Problema, é representado usualmente por x^* e o correspondente valor de f diz-se *valor óptimo* e representa-se por f^*.

Note-se que a formulação (1.2) diz respeito a um problema de *minimização*, no entanto pode também representar um problema de *maximização*, dado que

$$x^* = \min\{f(x),\ x \in D\} = \max\{-f(x),\ x \in D\}, D \subset \mathbb{R}^n$$

e, portanto, um ponto onde f atinge o seu máximo é o mesmo onde $-f$ atinge o seu mínimo. Note-se ainda que uma restrição do tipo $c_i(x) \leq b$ pode ser reescrita como $c_i(x) - b \leq 0$, pelo que esta formulação também é representativa deste tipo de restrições.

Apesar de se ter apresentado a forma geral do problema de optimização, existem problemas que têm apenas restrições de igualdade, outros só restrições de desigualdade e outros sem qualquer restrição.

1.3 Características Gerais do Problema

O facto da grande maioria dos problemas de optimização poderem ser apresentados na forma geral (1.2) não significa que se possam ignorar as diferenças entre os diversos problemas. Aliás, conhecer as características especiais de cada tipo de problema conduz a uma resolução mais eficiente. Essas características são, nomeadamente, o *número de funções objectivo*, a *dimensão do problema*, as *restrições de cada problema*, a *natureza das variáveis envolvidas* e as *possíveis variações da função objectivo*.

1.3.1 Número de Funções Objectivo

Relativamente ao número de funções objectivo, se o problema tiver apenas uma função objectivo diz-se que o problema é *Uniobjectivo*, se se pretende optimizar mais do que uma função objectivo, então está-se perante um *Problema de Optimização Multiobjectivo*.

1.3.2 Dimensão do Problema

No que diz respeito à dimensão do problema, n, Fernandes em [58] começa por distinguir *problemas unidimensionais*, para os quais $n = 1$, em (1.2), e *problemas multidimensionais*, para os quais $n > 1$. Esta característica é muito importante uma vez que afecta quer a capacidade de armazenamento, quer o esforço computacional necessário para se

obter uma solução. Não existe um número fixo de variáveis a partir do qual se considera um problema de grande dimensão. A prática mostra que, dependendo das restantes características do problema ou métodos de optimização a utilizar, um problema pode ser considerado de grande dimensão e simultaneamente de pequena dimensão.

1.3.3 Restrições do Problema

Os problemas podem ainda ser classificados de acordo com a existência ou não de restrições nas variáveis, de acordo com Fernandes em [58]: os *Problemas de Optimização com Restrições*, cuja formulação genérica é a apresentada em (1.2) e os *Problemas de Optimização sem Restrições*, em que a região admissível do problema é $\Omega = R^n$, e cuja formulação assume a forma (1.1).

Um caso especial de Optimização Com Restrições são os *Problemas com Restrições do Tipo Limites Simples*. Nestes problemas as restrições têm todas a forma:

$$l_i \leq x_i \leq u_i,\ i = 1, 2, ..., n, \text{ onde } l_i \text{ e } u_i \text{ são constantes.}$$

Neste caso, se a variável pertinente é x_i, as possíveis formas de restrição são:

i. $x_i = \beta$ (x_i é fixo com o valor $\beta = l_i$ ou $\beta = u_i$);

ii. $x_i \geq l_i$ (l_i é o *limite inferior admissível* para x_i – do inglês *lower bound*);

iii. $x_i \leq u_i$ (u_i é o *limite superior admissível* para x_i – do inglês *upper bound*).

As restrições do tipo **ii.** e **iii.** dizem-se *limites simples* para as variáveis. Note-se que ambos os limites podem ser infinitos. A região admissível nestes problemas é muitas vezes designada por caixa (do inglês *box*) uma vez que é uma região rectangular.

O Problema de Optimização com Restrições do Tipo Limites Simples nas Variáveis é:

$$\begin{aligned} &\underset{x \in \mathbb{R}^n}{minimizar} && f(x) \\ &\text{s. a.} && l_i \leq x_i \leq u_i, \quad i = 1, 2, ..., n. \end{aligned} \tag{1.3}$$

1.3.4 Natureza das variáveis envolvidas

A natureza do Problema de Optimização é dependente das variáveis envolvidas, podendo estas ser Contínuas ou Discretas. Os *Problemas Contínuos* são aqueles em que as variáveis têm um carácter contínuo, enquanto os *Discretos* são basicamente aqueles em que estas não são contínuas. Alguns dos Problemas Discretos encontram-se em situações em que as variáveis apenas podem assumir um número finito de valores discretos. A classe mais importante de Optimização Discreta é a *Programação Inteira*, onde as variáveis só podem assumir valores inteiros, ou mesmo apenas valores binários. Karlof em [87] apresenta uma visão geral da Programação Inteira.

Outro aspecto a ter em conta é à aleatoriedade das variáveis. Este conceito permite distinguir a *Programação Estocástica*, que funciona num panorama de optimização que envolve incerteza introduzida por meio de variáveis aleatórias, da *Programação Determinística*, onde os parâmetros são conhecidos. No seu site Maarten [108] apresenta um extensa lista bibliográfica nesta área.

1.3.5 Características da Função Objectivo e das Restrições

As características das funções envolvidas no problema, tal como a linearidade, a convexidade, a continuidade e a diferenciabilidade, permitem distinguir vários tipos de problemas. Essa distinção é descrita sucintamente nesta secção.

1.3.5.1 Linearidade

Tal como referido anteriormente, de acordo com as características da função objectivo e das restrições, os Problemas de Optimização podem ser divididos em duas grandes categorias:

- *Problemas de Optimização Linear* (LP do inglês *Linear Programming*) – Se a função objectivo e as restrições são lineares;
- *Problemas de Optimização não Linear* (NLP do inglês *Nonlinear Programming Problems*) – Se a função objectivo e as restrições contêm funções não lineares nas variáveis.

A forma matricial de um Problema de Optimização Linear é:

$$\begin{aligned} \min \quad & c^T x \\ s.a. \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{1.4}$$

onde

- $x \in \mathbb{R}^n$ é o vector das variáveis;
- $c \in \mathbb{R}^n$ e $b \in \mathbb{R}^m$ são vectores de coeficientes reais conhecidos;
- A é a matriz $m \times n$ dos coeficientes reais das variáveis nas equações lineares das restrições.

Muitos problemas práticos de Optimização podem ser descritos pela Programação Linear. Os problemas de fluxo de rede são um exemplo prático disso. O método de Optimização mais célebre na resolução destes problemas é o conhecido por *Método Simplex*, introduzido por Dantzig em [53].

Se a função objectivo f e/ou alguma das restrições é não linear, o Problema é de Optimização não Linear. Estes problemas são usualmente mais difíceis de resolver do que os lineares, no entanto a não-linearidade é quase inevitável em muitos problemas na vida real.

Neste trabalho os Problemas de Optimização a tratar serão *Problemas de Optimização não Lineares com Restrições*, isto é, cuja função objectivo, f, ou pelo menos uma das funções restrição é não linear nas variáveis.

Os Problemas de Optimização não Lineares sem Restrições são bem mais fáceis de resolver do que os problemas com restrições, pelo que têm sido desenvolvidas técnicas para transformar estes últimos em problemas sem restrições. Essas técnicas são, por exemplo, técnicas de *Penalidade Sequencial ou de Penalização Sequencial*, que implicam a resolução de uma sequência de problemas de optimização sem restrições para se atingir uma boa aproximação à solução do problema com restrições, ou técnicas de *Penalidade Exacta ou de Penalização Exacta* em que a solução encontrada do problema sem restrições coincide, sob certas condições, com a solução do problema com restrições. Dado ser

este o âmbito deste trabalho, apresentar-se-ão mais à frente algumas considerações sobre estes métodos e as respectivas referências bibliográficas.

1.3.5.2 Convexidade

A convexidade de uma função é uma propriedade muito importante e útil em Optimização. Refira-se por exemplo o facto de os minimizantes locais de uma função convexa serem também minimizantes globais.

Pode definir-se, segundo Fernandes em [59], uma função convexa da seguinte forma:

Definição 1.1. Uma função $f : \mathbb{R} \to \mathbb{R}$ de uma variável diz-se *Convexa* se para todo o par de valores x_1 e x_2 de $\mathbb{R}$ se verifica

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \tag{1.5}$$

para todo o λ real entre 0 e 1.

Em termos gráficos, uma função f é convexa se a recta que une dois pontos quaisquer do seu gráfico fica acima ou se sobrepõe ao gráfico de f. Se a desigualdade é estrita, isto é, se

$$f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda)f(x_2) \tag{1.6}$$

f diz-se *Estritamente Convexa*. A função f é *Côncava* se $-f$ é convexa.

Estas definições podem ser estendidas à dimensão n, interpretando as variáveis x_1 e x_2 como vectores na expressão (1.5) e (1.6) e utilizando as operações de multiplicações de vectores por escalares e adição de vectores.

Um problema que envolva apenas funções convexas diz-se um *Problema Convexo*. Neste tipo de problemas a região admissível é um conjunto Convexo.

Pode definir-se, segundo Fernandes em [59], um conjunto convexo da seguinte forma:

Definição 1.2. Um conjunto C é *Convexo* se $\forall\ x_1, x_2 \in C, \quad x_1 \neq x_2$, se tem que $z = \lambda x_1 + (1-\lambda)x_2$, com $\lambda \in [0,1]$, pertence a C.

Tendo em conta a definição anterior prova-se, segundo Fernandes em [59], que:

Teorema 1.3. *Seja C um conjunto convexo e $f : C \to \mathbb{R}$ uma função convexa, então:*

- *Qualquer minimizante local é um minimizante global;*
- *Se f é estritamente convexa, existe quando muito um minimizante global de f.*

1.3.5.3 Continuidade e Diferenciabilidade

Outra forma de classificar os problemas diz respeito à continuidade das funções envolvidas. Se as funções forem contínuas está-se na presença de um *Problema Contínuo*. Se, no entanto, alguma delas é descontínua então o *Problema* é *Descontínuo*.

A informação que está disponível, durante o processo de resolução, relativamente às derivadas das funções também é um factor de extrema importância. Por exemplo, pode ocorrer que a expressão da primeira e segunda derivadas da função objectivo não estejam disponíveis ou que sejam *não suaves* (do inglês *nonsmooth*), isto é, a função objectivo pode não ser continuamente diferenciável até à segunda ordem. Neste caso não é possível utilizar Métodos de Optimização que façam uso das derivadas, sendo também difícil estabelecer resultados de convergência. A resolução deste tipo de problemas é usualmente designada por *Optimização sem Derivadas* (em inglês *Derivative-free Optimization*, *Non-differentiable Optimization* ou *Nonsmooth Optimization*).

1.3.6 Analisador de Problemas

Tendo em conta o que foi dito previamente pode concluir-se que a escolha do Método de Optimização a utilizar, para resolver um problema, nem sempre é uma tarefa fácil. De facto, essa é uma das grandes preocupações que seguem a modelação do problema real. No sentido de resolver esta dificuldade têm-se desenvolvido técnicas de classificação dos problemas. Por exemplo Fourer e Orban em [67] desenvolveram um *Analisador de Problemas* (*Dr. Ampl – a meta solver for optimization* [1]) que examina o modelo introduzido, classifica-o e indica o solver, disponível no servidor NEOS [2], mais apropriado para o resolver.

[1] Podem encontrar-se mais informações em: http://www.gerad.ca/~orban/drampl/.
[2] NEOS – *Network-Enabled Optimization System* (http://neos.mcs.anl.gov/).

1.4 Classificação de Mínimos

O principal objectivo da resolução de problemas de optimização é encontrar o ponto x^* onde a função objectivo f atinge um mínimo, f^*. Então, deve estabelecer-se a noção de mínimo de f e os diferentes tipos de mínimos que podem ocorrer.

Considere-se um problema geral de optimização (equivalente a (1.2)):

$$\min_{x \in \Omega} f(x) \tag{1.7}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é a função objectivo e Ω é a região admissível do problema.

Segundo Matias em [110], podem definir-se para f vários tipos de minimizantes:

Definição 1.4. Seja D um subconjunto de Ω e considere-se que $B(x^*, \epsilon)$, se existir, representa a *bola aberta* [3] de centro x^* e raio ϵ. Um ponto admissível x^* é um:

- *minimizante local forte* sobre D se e só se existe $\epsilon > 0$ tal que $B(x^*, \epsilon) \subset D$ e

$$f(x^*) < f(x), \ \ \forall x \in B(x^*, \epsilon) \ \text{ com } \ x \neq x^*;$$

- *minimizante local fraco* sobre D se e só se existe $\epsilon > 0$ tal que $B(x^*, \epsilon) \subset D$ e

$$f(x^*) \leq f(x), \ \ \forall x \in B(x^*, \epsilon) \ \text{ com } \ x \neq x^*;$$

- *minimizante global* se e só se

$$f(x^*) < f(x), \ \ \forall x \in D \ \text{ com } \ x \neq x^*.$$

Ao valor da função objectivo f num mimimizante chama-se *mínimo*. Se o minimizante for global, então o mínimo é *global*.

A Figura 1.2 ilustra os tipos de mínimos definidos anteriormente para uma função a uma variável representada num conjunto D.

Estas definições também são válidas para problemas sem restrições, sendo que o conjunto de pontos admissíveis do problema é $\Omega = \mathbb{R}^n$.

[3] Segundo Matias em [110], uma *bola aberta* de centro x^* e raio ϵ, $B(x^*, \epsilon)$, é o conjunto dos pontos em $\mathbb{R}^n$ situados a uma distância inferior a ϵ do ponto $x^* \in \mathbb{R}^n$.

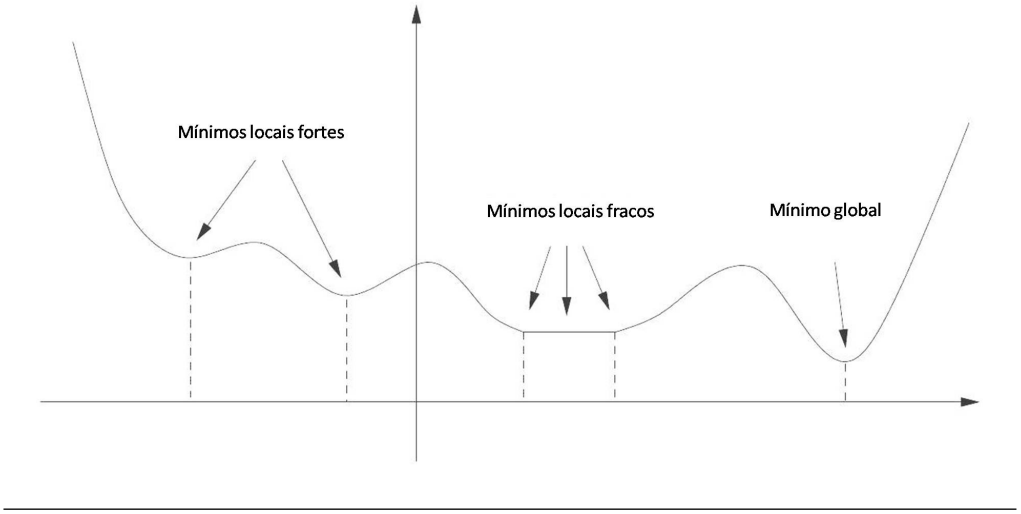

FIGURA 1.2: Exemplos de minimizantes globais e locais a uma dimensão
(Fonte: Sainvitu em [137])

Para a grande maioria dos problemas de optimização não é possível encontrar um mínimo global, no entanto para problemas práticos é, usualmente, satisfatório encontrar um minimizante numa vizinhança [4], que será o ponto que tem menor valor da função objectivo nessa vizinhança.

1.5 Condições de Optimalidade

Para verificar se um determinado ponto é um ponto óptimo de um problema é essencial conhecer as condições de optimalidade. As propriedades de um ponto óptimo fornecem informação que pode ser de grande utilidade na procura de novos algoritmos para encontrar a solução. A definição 1.4 não fornece essas informações. Seria necessário avaliar f em muitos pontos ou mesmo num número infinito de pontos admissíveis, numa vizinhança de uma possível solução, o que requeria um grande esforço computacional ou a inviabilidade de resolução do problema. Essa dificuldade é ultrapassada se a função objectivo e o conjunto das restrições possuírem determinadas propriedades/características, sendo nesse caso possível encontrar outras condições de optimalidade.

Assim, nesta secção apresentar-se-á a classificação e a identificação das condições de optimalidade de problemas de optimização sem e com restrições. A descrição das condições

[4] Seja $x \in \mathbb{R}^n$. Segundo Matias em [110], um conjunto aberto que contenha x diz-se *Vizinhança* de x.

de optimalidade, para cada tipo de problemas, pode encontrar-se em diversos textos, por exemplo em [16, 19, 58, 59, 110, 123, 137].

Antes de apresentar as condições de optimalidade considerem-se as definições seguintes, baseadas nas que foram apresentadas por Fernandes, Matias e Sainvitu em [59, 110] e [137] e onde se considera a função f definida por:

$$\begin{array}{rccc} f: & \mathbb{R}^n & \rightarrow & \mathbb{R} \\ & x = (x_1, x_2, ..., x_n) & \mapsto & f(x) \end{array}.$$

Definição 1.5. A *primeira derivada parcial de f em ordem a x_i* é dada pelo seguinte limite (se existir):

$$\frac{\partial f}{\partial x_i}(x) = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon e_i) - f(x)}{\varepsilon}$$

para $i = 1, 2, ..., n$ e onde e_i é o i-ésimo vector unitário de ordem n.

Definição 1.6. Se todas as derivadas parciais (da Definição 1.5) existirem, f diz-se *diferenciável* e o vector das n derivadas parciais de f diz-se *vector gradiente* de f e é usualmente representado por $\nabla f(x)$ ou $g(x)$ ($\in \mathbb{R}^n$):

$$g(x) = \nabla f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ ... \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix} \tag{1.8}$$

Nestas condições, prova-se que:

Teorema 1.7. *Se a função f é diferenciável num ponto x, f é contínua em x.*

Definição 1.8. Se a função f é diferenciável e as suas derivadas são funções contínuas em x, f diz-se *continuamente diferenciável* em x e escreve-se $f \in C^1$.

Podem ainda definir-se as segundas derivadas parciais:

Definição 1.9. As *segundas derivadas parciais de f*, se existirem, são as derivadas das primeiras derivadas parciais, isto é, podem definir-se, com $i, j = 1, 2, ..., n$:

- a derivada parcial de f em ordem a x_i da derivada parcial de f em ordem a x_j:

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}(x)\right), i \neq j \tag{1.9}$$

- a segunda derivada parcial de f em ordem a x_i:

$$\frac{\partial^2 f(x)}{\partial x_i^2} = \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}(x)\right), i = j. \tag{1.10}$$

Definição 1.10. Se existirem e forem contínuas, num ponto x, todas as segundas derivadas parciais de f, (1.9) e (1.10) $\forall i,j : i \geq 1 \wedge j \leq n$, então diz-se que f é *continuamente diferenciável até à segunda ordem* ou que f é *suave* e escreve-se $f \in C^2$.

Note-se que se $f \in C^2$, então $\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial^2 f(x)}{\partial x_j \partial x_i},\ i,j = 1,...,n.$

Definição 1.11. Se $f \in C^2$, à matriz, quadrada de ordem n e simétrica [5], das n^2 derivadas parciais de ordem dois dá-se o nome de *Matriz Hessiana* de $f(x)$, e usualmente representa-se por $\nabla^2 f(x)$ ou $G(x)$:

$$G(x) = \nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix} \tag{1.11}$$

Com as definições apresentadas anteriormente podem estabelecer-se as Condições de Optimalidade para Problemas de Optimização sem e com restrições.

1.5.1 Condições de Optimalidade para Problemas de Optimização sem Restrições

Nesta secção e tendo em conta o que é exposto por Fernandes, Matias e Kelley em [59, 89, 110], serão apresentadas as Condições de Optimalidade para Problemas de Optimização sem Restrições, cuja forma geral se enunciou em (1.1).

[5] Uma matriz A é simétrica se $A = A^T$, onde A^T é a matriz transposta de A, que se obtém de trocar as linhas de A com as suas colunas.

Teorema 1.12. (*Condição necessária de 1ª ordem*)

Seja x^* um minimizante do problema (1.1) e f continuamente diferenciável até à segunda ordem numa vizinhança aberta de x^*. Então:

$$\nabla f(x^*) = 0, \tag{1.12}$$

ou seja, x^* é um ponto estacionário de f.

A condição (1.12) é uma condição necessária para que x^* seja um minimizante (ou maximizante), mas não é suficiente, pois define apenas *pontos estacionários* ou *pontos críticos*, que podem ser minimizantes, maximizastes ou pontos sela. Então a condição só estabelece que qualquer minimizante tem que ser um ponto estacionário, mas nem todos os pontos estacionários são minimizantes. É, portanto, necessário definir as condições suficientes para que x^* seja um minimizante local forte. Os Teoremas seguintes, apresentados nos trabalhos dos mesmos autores citados anteriormente ([59, 89, 110]), estabelecem essas condições.

Teorema 1.13. (*Condições necessárias de 2ª ordem*)

Seja x^* um minimizante do problema (1.1) e f continuamente diferenciável até à segunda ordem, numa vizinhança aberta de x^*. Então $\nabla f(x^*) = 0$, ou seja, x^* é um ponto estacionário de f, e $\nabla^2 f(x)$ é semidefinida positiva [6].

Definição 1.14. Nas condições definidas no Teorema 1.13 diz-se que x^* é um *ponto estacionário de segunda ordem.*

O teorema seguinte estabelece as condições para que x^* seja um minimizante local da função objectivo f.

Teorema 1.15. (*Condições suficientes de 2ª ordem*)

Seja f continuamente diferenciável até à segunda ordem, numa vizinhança aberta de x^* e suponha-se que $\nabla f(x^*) = 0$ ou seja, que x^* é um ponto estacionário de f, e que $\nabla^2 f(x)$ é definida positiva [7]*, então x^* é um mínimo local forte do problema* (1.1).

[6] Uma matriz A, simétrica, diz-se *semidefinida positiva* se: $x^T Ax \geq 0$, $\forall x \in \mathbb{R}^n$, ou ainda, se os determinantes das submatrizes principais de A são não negativos.

[7] Diz-se que uma matriz A é *definida positiva* se $x^T Ax > 0$, $\forall x \in \mathbb{R}^n$, $x \neq 0$, ou seja, os determinantes das submatrizes principais de A são positivos.

Note-se que se x^* é um ponto crítico de primeira ordem mas a Hessiana é *indefinida* [8], diz-se que x^* é um *ponto sela* (do inglês *saddle point*) ou *ponto de descanso*.

Conclusão: (Fonte: Fernandes em [59].)
Seja f uma função continuamente diferenciável até à ordem 2 e x^* um ponto para o qual $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*)$ uma matriz não nula, então:

- Se $\nabla^2 f(x^*)$ é definida positiva, então x^* é minimizante;
- Se $\nabla^2 f(x^*)$ é definida negativa, então x^* é maximizante;
- Se $\nabla^2 f(x^*)$ é semidefinida positiva, então x^* é minimizante ou ponto sela;
- Se $\nabla^2 f(x^*)$ é semidefinida negativa, então x^* é maximizante ou ponto sela;
- Se $\nabla^2 f(x^*)$ é indefinida então, x^* é ponto sela.

Em programação convexa as condições de optimalidade são mais simples, uma vez que, se a função objectivo f for convexa, qualquer mínimo local é um mínimo global. Isto permite estabelecer, segundo Fernandes e Sainvitu em [59] e [137], a condição de optimalidade necessária e suficiente para este tipo de problemas.

Teorema 1.16. *(Condição necessária e suficiente para problemas convexos)*

Seja f convexa e continuamente diferenciável. Então x^* é um minimizante de (1.1) se e só se $\nabla f(x^*) = 0$, ou seja, se x^* é um ponto estacionário de f.

Finalmente, é de salientar o facto das condições de optimalidade fundamentarem o desenvolvimento e análise de algoritmos iterativos. No caso de problemas sem restrições, o que se pretende é encontrar algoritmos que encontrem pontos onde o gradiente se anule. Na prática, os algoritmos iterativos, para este tipo de problemas, nem sempre são capazes de o fazer, pelo que é satisfatório terminar o processo quando se obtém um valor do gradiente suficientemente próximo de zero. Esta será a medida que permite parar o processo iterativo e que é usualmente designada por *medida de estacionaridade*.

[8] Uma matriz A é *indefinida* se $\exists x, y \in \mathbb{R}^n$ tais que $x^T Ax > 0$ e $y^T Ay < 0$, ou seja, se não for definida positiva, nem semidefinida positiva, nem definida negativa (neste caso os determinantes das submatrizes principais têm sinais alternados, sendo o de ordem 1 negativo), nem semidefinida negativa (neste caso pelo menos um dos determinantes das submatrizes principais é zero e os outros têm sinais alternados, sendo o de ordem 1 negativo).

1.5.2 Condições de Optimalidade para Problemas de Optimização com Restrições

Nesta secção e tendo em conta o que é exposto por Fernandes, Matias, Sainvitu e Kelley em [59, 89, 110, 137], serão apresentadas as Condições de Optimalidade para Problemas de Optimização sem Restrições, cuja forma geral foi apresentada em (1.2).

1.5.2.1 Notações/Definições

No sentido de estabelecer as condições de optimalidade para Problemas de Optimização com Restrições é necessário apresentar algumas notações e definições complementares. Nesta secção apresentam-se assim, para o problema (1.2) uma classificação das restrições e definem-se gradientes, matrizes Hessianas e a Função Lagrangeana.

A classificação das restrições apresentada na definição que se segue baseia-se na exposta por Matias em [110].

Definição 1.17. Seja $\bar{x}$ um ponto de $\mathbb{R}^n$, t um valor real e $c : \mathbb{R}^n \to \mathbb{R}$ uma função genérica, tais que $c(x) \leq t$ é uma restrição de um problema de optimização. Nestas condições diz-se que:

- $\bar{x}$ *viola a restrição* $c(x) \leq t$ se e só se $c(\bar{x}) > t$;
- $c(x) \leq t$ é uma *restrição activa* em $\bar{x}$ se e só se $c(\bar{x}) = t$;
- $c(x) \leq t$ é uma *restrição não activa* em $\bar{x}$ se e só se $c(\bar{x}) < t$.

Assim, podem, segundo Sainvitu em [137], definir-se os seguintes conjuntos de índices:

Definição 1.18. Considere-se o conjunto de índices $\mathcal{A}(x) = \{i \in \mathcal{E} \cup \mathcal{I} : c_i(x) = 0\}$. Então $\mathcal{A}(x)$ representa o conjunto de todos os índices das restrições activas do problema e diz-se *conjunto activo* num ponto admissível x.

As restrições inactivas em x são, desta forma, as restrições cujos índices não pertencem ao conjunto activo, isto é, cujos índices $j \in \mathcal{E} \cup \mathcal{I}$ são tais que $j \notin \mathcal{A}(x)$.

Se as restrições forem funções C^2, isto é, continuamente diferenciáveis até à segunda ordem, podem definir-se, segundo Fernandes em [59], os vectores gradientes das restrições e as matrizes Hessianas das restrições conforme as definições seguintes.

Definição 1.19. Se todas as derivadas parciais das restrições existirem os m vectores *gradientes das restrições* $c_i(x), i = 1, ..., m$ podem representar-se pelos m vectores n-*dimensionais* :

$$\nabla c_1(x) = \begin{pmatrix} \frac{\partial c_1(x)}{\partial x_1} \\ \frac{\partial c_1(x)}{\partial x_2} \\ \cdots \\ \frac{\partial c_1(x)}{\partial x_n} \end{pmatrix}, \quad \nabla c_2(x) = \begin{pmatrix} \frac{\partial c_2(x)}{\partial x_1} \\ \frac{\partial c_2(x)}{\partial x_2} \\ \cdots \\ \frac{\partial c_2(x)}{\partial x_n} \end{pmatrix}, ..., \quad \nabla c_m(x) = \begin{pmatrix} \frac{\partial c_m(x)}{\partial x_1} \\ \frac{\partial c_m(x)}{\partial x_2} \\ \cdots \\ \frac{\partial c_m(x)}{\partial x_n} \end{pmatrix} \tag{1.13}$$

A *matriz que contém os gradientes das restrições ao longo das colunas* é uma matriz de dimensão $n \times m$:

$$\nabla C(x) = \begin{pmatrix} \nabla c_1(x) & \cdots & \nabla c_m(x) \end{pmatrix} = \begin{pmatrix} \frac{\partial c_1}{\partial x_1} & \cdots & \frac{\partial c_m}{\partial x_1} \\ \cdots & \cdots & \cdots \\ \frac{\partial c_1}{\partial x_n} & \cdots & \frac{\partial c_m}{\partial x_n} \end{pmatrix} \tag{1.14}$$

As *matrizes Hessianas* das restrições, se existirem e forem contínuas as derivadas parciais de segunda ordem de todas as restrições, são as matrizes simétricas de ordem n ($\mathbb{R}^{n \times n}$):

$$\nabla^2 c_1(x) = \begin{pmatrix} \frac{\partial^2 c_1}{\partial x_1^2} & \cdots & \frac{\partial^2 c_1}{\partial x_n \partial x_1} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 c_1}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 c_1}{\partial x_1^2} \end{pmatrix}, \cdots, \quad \nabla^2 c_m(x) = \begin{pmatrix} \frac{\partial^2 c_m}{\partial x_1^2} & \cdots & \frac{\partial^2 c_m}{\partial x_n \partial x_1} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 c_m}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 c_m}{\partial x_1^2} \end{pmatrix} \tag{1.15}$$

Para o problema (1.2) pode ainda definir-se a Função Lagrangeana. A definição desta função, do gradiente e da matriz Hessiana, apresentadas a seguir foram baseadas nas apresentadas por Fernandes, Matias e Sainvitu em [59, 110, 137] [9].

Definição 1.20. A função

$$L(x, \lambda) = f(x) + \lambda^T c(x) = f(x) + \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x), \tag{1.16}$$

[9] Em [59, 110] consideram-se as restrições $c_i(x) \leq 0, i \in \mathcal{I}$ na forma $c_i(x) \geq 0, i \in \mathcal{I}$, pelo que a função Lagrangeana considerada é: $L(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x)$.

diz-se *Função Lagrangeana* ou *Função de Lagrange* do problema (1.2), onde o vector dos *multiplicadores de Lagrange* associados às restrições é $\lambda = (\lambda_1, \cdots, \lambda_m)^T$. As componentes de x dizem-se *variáveis primais* e os multiplicadores de Lagrange são conhecidos por *variáveis duais*.

Definição 1.21. Seja L a Função Lagrangeana para o problema (1.2), $\nabla f(x)$ o gradiente de f, $\nabla c_i(x)$ os gradientes das restrições, $\nabla^2 f(x)$ a matriz Hessiana da função f e $\nabla^2 c_i(x)$ as matrizes Hessianas das restrições, com $i = 1, 2, ..., m$.

- A *matriz do gradiente* de L, que contém as derivadas parciais de primeira ordem de L em relação à variável x, denota-se por ∇L e é dada por:

$$\nabla L(x, \lambda) = \nabla f(x) + \nabla c(x)^T \lambda = \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla c_i(x) \tag{1.17}$$

- A *matriz Hessiana* de L, que contêm as derivadas parciais de segunda ordem de L em relação à variável x, denota-se por $\nabla^2 L$ e é dada por:

$$\nabla^2 L(x, \lambda) = \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(x). \tag{1.18}$$

1.5.2.2 Condições de qualificação das restrições

Para se estabelecerem as condições necessárias e suficientes para a existência de mínimos locais é necessário que certas condições para as restrições se verifiquem. Estas condições são conhecidas por *Condições de qualificação das restrições* (do inglês *Constraint Qualification Conditions*).

Segundo Fernandes em [59], a verificação destas condições garante a não ocorrência de comportamentos indesejáveis em x^* enquanto solução do problema. Elas garantem que, próximo do presumível mínimo, qualquer restrição não linear é aproximada razoavelmente pela sua expansão de Taylor de primeira ordem [10].

[10] Se a função f admite primeira e segunda derivadas em todos os pontos numa vizinhança de x então pode aproximar-se o valor da função em todos os pontos da vizinhança através do polinómio de Taylor, nomeadamente através do polinómio de Taylor de primeira ordem: $f(x+s) \approx f(x) + \nabla f(x)^T s$, com $(x+s)$ na vizinhança considerada, ou pelo polinómio de Taylor de segunda ordem: $f(x+s) \approx f(x) + \nabla f(x)^T s + \frac{1}{2} s^T \nabla^2 f(x) s$.

Existem diversas Condições de qualificação das restrições, por exemplo, a Condição de Independência Linear (LICQ – *Linear Independence Constraint Qualification*), a Condição de Slater (SCQ – *Slater's Constraint Qualification*), a Condição de Mangasarian-Fromovitz (MFCQ – *Mangasarian-Fromovitz Constraint Qualification*). Freund em [69] e Nocedal e Wrigth em [123] descrevem estas últimas condições. A Condição de Independência Linear, apresentada de seguida, é exposta por Fernandes em [59], Freund em [69], Matias em [110] e Sainvitu em [137].

Condição 1.22. *(Condição de Independência Linear)*

Considere-se um ponto x^* e o correspondente conjunto activo $\mathcal{A}(x^*)$. Se os gradientes das restrições que estão activas na solução x^*, calculados em x^*, são linearmente independentes, isto é, se os vectores do conjunto $\{\nabla c_i(x^*) : i \in \mathcal{A}(x^*)\}$ são linearmente independentes, diz-se que x^* é regular e que se verifica a Condição de Independência Linear (LICQ – do inglês Linear Independence Constraint Qualification).

1.5.2.3 Condições Optimalidade

A Condição de Independência Linear, referida anteriormente, permite estabelecer as Condições necessárias de primeira ordem para o problema (1.2)conforme se estabelece no Teorema seguinte (baseado em diversas fontes: [59, 68, 110, 123, 137]).

Teorema 1.23. (*Condições necessárias de primeira ordem*)

Seja x^* uma solução do problema (1.2) e x^* regular (verifica LICQ). Então existe um vector de multiplicadores de Lagrange λ^*, com componentes $\lambda_i^*(i \in \mathcal{E} \cup \mathcal{I})$ tal que (x^*, λ^*) satisfaz as condições:

$$\nabla L(x^*, \lambda^*) = 0, \tag{1.19a}$$

$$c_i(x^*) = 0, \forall i \in \mathcal{E}, \tag{1.19b}$$

$$c_i(x^*) \leq 0, \forall i \in \mathcal{I}, \tag{1.19c}$$

$$\lambda_i^* \geq 0, \forall i \in \mathcal{I}, \tag{1.19d}$$

$$\lambda_i^* c_i(x^*) = 0, \forall i \in \mathcal{E} \cup \mathcal{I}. \tag{1.19e}$$

Tendo em conta o teorema podem fazer-se as seguintes observações:

- As condições (1.19a - 1.19e) dizem-se *Condições KKT (Karush-Kuhn-Tucker)* e o par (x^*, λ^*) chama-se *par KKT*;
- Um ponto x^* que satisfaz as condições KKT diz-se *Ponto KKT* ou ponto crítico de primeira ordem para o problema (1.2);
- A condição (1.19e) é usualmente designada por *Condição de Estacionaridade*;
- As condições (1.19b) e (1.19c) chamam-se *Condições Admissíveis (primais)*;
- A condição (1.19e) é a designada por *Condição de Complementaridade* e significa que:
  - Ou a restrição $c_i(x)$ está activa na solução, isto é, $c_i(x^*) = 0$;
  - Ou o multiplicador que lhe está associado é nulo, isto é, λ_i^*=0;
  - Em particular, qualquer restrição que não esteja activa na solução tem multiplicador nulo;
  - Se os multiplicadores que correspondem a restrições activas são todos positivos, a complementaridade diz-se *estrita*;
  - Se um multiplicador que corresponde a uma restrição activa, é positivo, a *restrição* diz-se *não degenerada*;
  - Se um multiplicador que corresponde a uma restrição activa, é zero, a *restrição* diz-se *degenerada*.
- A condição (1.19a) pode ser reescrita da seguinte forma:

$$\nabla L(x^*, \lambda^*) = \nabla f(x^*) + \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* \nabla c_i(x^*) = 0. \tag{1.20}$$

Definição 1.24. Considerem-se os vectores $x \in \mathbb{R}^n$ e $\lambda \in \mathbb{R}^{t+m}$. Pode definir-se, tendo também em conta o que foi definido no problema (1.2), o conjunto $\mathcal{N}_+(x, \lambda)$:

$$\mathcal{N}_+(x,\lambda) = \left\{ s \in \mathbb{R}^n \;\middle|\; \begin{array}{c} s^T \nabla c_i(x) = 0, \forall i \in \mathcal{E} \cup \{j \in \mathcal{A}(x) \cap \mathcal{I} : \lambda_j > 0\} \\ s^T \nabla c_i(x) \leq 0, \forall i \in \{j \in \mathcal{A}(x) \cap \mathcal{I} : \lambda_j = 0\} \end{array} \right\}$$

Usando a informação das derivadas de segunda ordem e a definição anterior podem estabelecer-se as condições de optimalidade de segunda ordem para o problema (1.2), segundo vários autores: [59, 68, 110, 123, 137].

Teorema 1.25. *(Condições necessárias de segunda ordem)*

Seja x^* uma solução do problema (1.2) e x^* regular (verifica LICQ) e λ^* um multiplicador de Lagrange tal que as condições KKT (1.19a - 1.19e) se verificam. Então

$$s^T \nabla^2 L(x^*, \lambda^*) s \geq 0, \forall s \in \mathcal{N}_+(x^*, \lambda^*) \tag{1.21}$$

A condição (1.21) significa que a curvatura da Lagrangeana não deve ser negativa em nenhuma das direcções de $\mathcal{N}_+(x^*, \lambda^*)$.

Um ponto que satisfaz condição (1.21) diz-se *Ponto crítico forte de segunda ordem.*

As Condições suficientes de segunda ordem são estabelecidas do Teorema seguinte (baseado em diversas fontes: [59, 68, 110, 123, 137]).

Teorema 1.26. (*Condições suficientes de segunda ordem*)

Seja x^* um ponto admissível do problema (1.2) e suponha-se que existem multiplicadores de Lagrange tais que as condições KKT (1.19a - 1.19e) se verificam. Considere-se, ainda, que:

$$s^T \nabla^2 L(x^*, \lambda^*) s > 0, \forall s \in \mathcal{N}_+(x^*, \lambda^*)(s \neq 0). \tag{1.22}$$

Então x^* é um minimizante local forte do problema (1.2).

1.6 Métodos de Optimização

A complexidade dos Problemas de Optimização cria diversas barreiras à sua resolução. A primeira diz respeito às dificuldades na sua formulação matemática. Depois é necessário fazer a sua classificação ou adaptação de formulação a um de entre os tipos de Problemas de Optimização para os quais se conhecem métodos de resolução. Seguidamente é necessário seleccionar e aplicar os métodos mais adequados para os resolver. Infelizmente, dada a complexidade da realidade, nem todos os problemas podem ser resolvidos

recorrendo às condições de optimalidade, apresentadas na Secção anterior, pelo que os investigadores têm desenvolvido *métodos iterativos* para encontrar soluções o mais adequadas possível à realidade.

Os métodos iterativos não resolvem, assim, o problema analiticamente e não usam directamente as condições de optimalidade, em vez disso produzem uma sucessão de iterações $x_1, x_2, ..., x_k, ...$ que se espera que convirja para a solução (minimizante) do problema. Para garantir essa convergência, o processo de cálculo de iterações seguintes nos métodos iterativos deve usar conceitos associados às condições de optimalidade, ou seja, deve ter em conta a verificação, ainda que por aproximação, destas condições.

A escolha do método iterativo mais adequado para resolver um problema pode ser uma tarefa difícil de ultrapassar, seja pela especificidade do problema ou pela dificuldade em escolher, de entre os métodos disponíveis, o que melhor permite verificar/aproximar as Condições de optimalidade.

Tendo em conta esta diversidade de características dos problemas têm-se desenvolvido diferentes sub-áreas de Optimização e existem inúmeros *solvers* disponíveis. Esta diversidade de sub-áreas de Optimização fica bem ilustrada na árvore (que se reproduz na FIGURA 1.3) e que era apresentada no servidor NEOS[11].

Aqui não se faz uma classificação tão pormenorizada como a que é feita na WikiNEOS, faz-se apenas uma abordagem geral dos vários Métodos de Optimização, referindo os métodos apresentados na FIGURA 1.4 e dando especial relevo aos métodos que não usem derivadas nem modelos, dado ser este o tema deste trabalho.

Em termos gerais os Métodos de Optimização podem ser:

- *Métodos de Optimização Estocásticos ou Heurísticos* – Métodos que usam gerações de números pseudo-aleatórios e o conceito de aleatoriedade, como, por exemplo, Estratégias Evolutivas (Beyer, [20]), Algoritmos Genéticos (Deb, [49]), Colónias ou Enxames de Partículas (Parsopoulos, [126]). Segundo Conn et. al., em [32], estes métodos são usados como um último recurso e só consideram que se devam aplicar

[11] Esta árvore estava disponível em *NEOS Guide Optimization Tree* – http://www-new.mcs.anl.gov/otc/Guide/OptWeb/index.html, actualmente **não está disponível**. Esta figura servia como página de entrada para um guia on-line onde eram introduzidas as diferentes sub-áreas da Optimização, indicando pacotes de software disponíveis neste servidor, na área respectiva. Esta informação está agora disponível na WikiNEOS, em http://wiki.mcs.anl.gov/NEOS/index.php/Optimization_Tree.

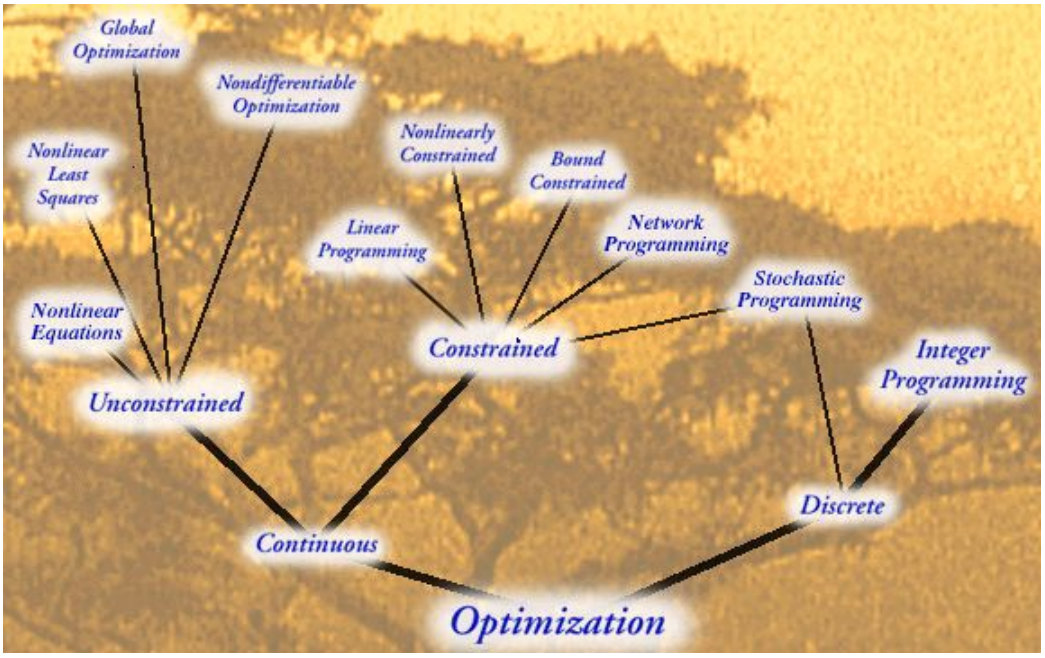

FIGURA 1.3: Árvore de Sub-áreas de Optimização
(Fonte: NEOS Guide Optimization Tree)

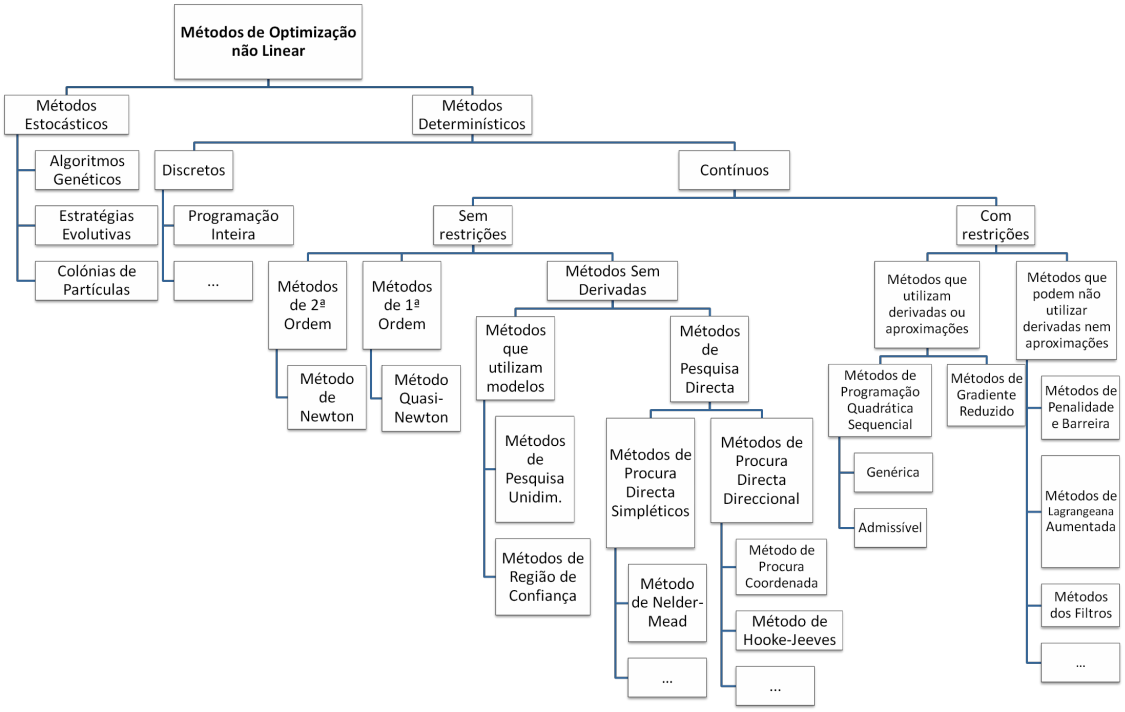

FIGURA 1.4: Árvore de alguns Métodos de Optimização

a problemas onde o espaço é necessariamente grande, complexo, ou não se conhece bem e uma análise matemática mais sofisticada não é aplicável;

- *Métodos de Optimização Determinísticos* – Métodos que usam aritmética intervalar ou enumeração exaustiva, entre outras (as referências correspondentes a estes métodos serão apresentadas mais à frente, para os diversos Métodos Determinísticos);
- *Métodos de Optimização Híbridos* – Combinam características ou métodos dos dois tipos anteriores (por exemplo, o Método de Colónia de Partículas de Vaz e Vicente apresentado em [151]).

Quanto ao tipo de mínimo que permitem encontrar, os métodos podem ser:

- *Métodos de Optimização Global* (*Global Optimization Methods*)[12] – Métodos que têm como objectivo determinar o óptimo global do problema;
- *Métodos de Optimização Local* (*Local Optimization Methods*) – Métodos que permitem encontrar apenas um mínimo local, o que estiver mais próximo da aproximação inicial.

No que diz respeito à natureza das variáveis envolvidas podem ter-se:

- *Métodos de Optimização Discretos* (*Discrete Optimization Methods*) – Adequados para resolver problemas onde as variáveis são discretas (por exemplo valores inteiros);
- *Métodos de Optimização Contínuos* (*Continous Optimization Methods*) – Adequados para resolver problemas onde as variáveis são contínuas (podem assumir qualquer valor real).

Na classe de Optimização Discreta podem destacar-se os métodos de *Programação Inteira*, onde as variáveis assumem valores inteiros ou mesmo binários. Karlof em [87] faz uma introdução da Programação Inteira, dando uma visão geral desta área de Optimização.

Os Métodos de Optimização Contínua podem dividir-se em:

[12]Para uma visão de Optimização Global pode ver-se o site http://www.mat.univie.ac.at/~neum/glopt.html, que contém links para software, problemas teste, artigos, entre outros, na área.

- *Métodos de Optimização Sem Restrições* (*Unconstrained Optimization Methods*) – São métodos desenhados para resolver Problemas de Optimização sem Restrições, ou seja, problemas do tipo (1.1);

- *Métodos de Optimização Com Restrições* (*Constrained Optimization Methods*) – São métodos desenhados para resolver Problemas de Optimização com Restrições, ou seja, problemas do tipo (1.2).

Quanto à convergência podem ter-se:

- *Métodos de Convergência Local* [13] – O mais célebre é o Método de Newton. Fletcher em [61] e Kelley em [89] apresentam este método com algum pormenor.

- *Métodos de Convergência Global* [14] – Podem destacar-se os Métodos de Procura Unidimensional (*Line-Search Methods*) e Métodos de Região Admissível ou de Região de Confiança (*Trust-Region Methods*). Kelley em [89], apresenta as linhas gerais e principais resultados destes métodos.

  Os Métodos de Procura Unidimensional e os Métodos de Região de Confiança são adequados para problemas de dimensão não muito grande, porque em cada iteração pode ser necessário fazer até n^2 cálculos da função objectivo.

  Para problemas de grande dimensão são usados usualmente Métodos de Newton Truncados, que usam modelos quadráticos para aproximar a função objectivo f.

1.6.1 Métodos de Optimização sem Restrições

Os Métodos de Optimização sem Restrições compreendem:

- *Métodos de Segunda Ordem* (*Second Order Methods*) – Métodos que requerem o conhecimento da Matriz Hessiana da função objectivo f, por exemplo, o Método de Newton (apresentado , por exemplo, por Kelley em [89]).

- *Métodos de Primeira Ordem* (*First Order Methods*) – Métodos que não requerem o conhecimento da Matriz Hessiana da função objectivo f, apenas o seu gradiente,

[13] Por Métodos de Convergência Local entendem-se métodos que requerem que o ponto ou valor inicial inicial x_0 esteja próximo do minimizante local x^*, para que se mostre a convergência do método.

[14] Não se devem confundir com Métodos de Optimização Global. Por Métodos de Convergência Global entendem-se métodos que para qualquer iteração inicial x_0 convergem para um minimizante local x^*.

e incluem métodos que usam aproximações das segundas derivadas, por exemplo, através de diferenças finitas das primeiras derivadas da função objectivo, como os Métodos Quasi-Newton. Kelley em [89], apresenta conceitos essenciais e resultados para estes métodos.

- *Métodos de Ordem Zero ou Métodos de Optimização sem Derivadas* (*Derivative-Free Methods*) – São métodos que não requerem o conhecimento das derivadas (não utilizam nem a Matriz Hessiana, nem o gradiente) da função objectivo, nem calculam estimativas para estas derivadas. Estes métodos usam apenas os valores da função objectivo para o processo de optimização. Conn et. al. em [32] e Fletcher em [61] apresentam as linhas gerais e os principais resultados destes métodos.

  Estes métodos requerem, em geral, mais iterações para encontrar a solução, por não usarem informações relativamente às derivadas da função objectivo, mas são os únicos métodos que se podem utilizar quando a função objectivo é não suave, não se conhece a sua expressão, está sujeita a ruído ou quando é impossível calcular as suas derivadas. Como o poder de cálculo computacional dos computadores actuais já é considerável, e tem tendência a aumentar, este facto tem cada vez menos importância na escolha de um algoritmo, a não ser que essa diferença tenha como consequência uma grande alteração de tempos de execução.

  Nos casos em que se tem alguma informação sobre as derivadas de f é mais adequado usar Métodos de Primeira ou Segunda Ordem.

  Segundo Vaz em [152], os problemas que podem ser resolvidos com técnicas OSD são em geral de pequenas dimensões (na ordem das poucas centenas) e só se pode esperar alguns dígitos de precisão (na ordem de 10^{-5}).

  Estes métodos incluem várias classes de algoritmos, ainda segundo Vaz em [152], nomeadamente:

  1. *Métodos de Procura (ou Pesquisa) Directa Direccional* (*Direccional Direct Search*) – Métodos que se baseiam em bases positivas ou conjuntos geradores positivos, progredindo no sentido do melhor ponto (em termos da função objectivo) usando grelhas ou padrões. Esta classe inclui algoritmos como a Procura Coordenada (CS – *Cordinate Search*), a Procura em Padrão (PS – *Pattern Search*), a Procura em Padrão Generalizada (GPS – *Generalized Pattern Search*), a Procura com Conjuntos Geradores (GSS – *Generating Set*

Search) e a Procura Directa com Grelha Adaptativa (MADS – *Mesh Adaptive Direct Search*).

2. *Métodos de Procura Directa Simpléticos* (*Simplicial Direct Search*) – Métodos que garantem o decréscimo da função objectivo usando operações com *simplexes*, caminhando na direcção oposta ao pior ponto. Esta classe inclui o algoritmo de Nelder-Mead e as suas variantes.
3. *Métodos de Procura Unidimensional* (*Line-Search*) – Métodos que usam gradientes simpléticos, como direcção de procura, em conjunto com uma estratégia de procura unidimensional. Esta classe inclui o algoritmo de Filtro Implícito (*Implicit Filter*).
4. *Métodos de Região de Confiança* (*Trust Region*) – Métodos que aplicam as técnicas de região de confiança a modelos lineares ou quadráticos da função objectivo. Esta classe inclui algoritmos baseados em modelos polinomiais ou de funções base radiais (*radial basis functions*).
5. Podem ainda destacar-se outros Métodos de Optimização sem Derivadas célebres como:
   - O Método de Hooke-Jeeves, que é um caso particular da GPS;
   - O algoritmo DIRECT, que usa a constante de Lipschitz (para funções Lipschitz contínuas [15]) para eliminar intervalos não promissores para o óptimo global.

Note-se que alguns métodos dos anteriores incluem na sua designação a expressão *Pesquisa Directa*. Esta expressão é muitas vezes usada para designar os Métodos de Ordem Zero ou Métodos de Optimização sem Derivadas, no entanto a designação *Métodos de Pesquisa Directa* é menos abrangente. Assim podem distinguir-se:

- *Métodos de Optimização sem Derivadas* (*Derivative-Free Methods*) – Esta designação abrange todos os Métodos de Optimização de Ordem Zero, ou seja, todos os que não calculam nem aproximam derivadas, mas que podem aproximar a função objectivo através de modelos e usar esses modelos para efectuar a procura;

[15] Segundo Sainvitu em [137], diz-se que uma função $f : \mathcal{S} \to \mathbb{R}^n$, onde $\mathcal{S}$ é um subconjunto aberto de $\mathbb{R}^n$ é *Contínua à Lipschitz* em $\mathcal{S}$ se, para alguma norma $\|.\|$, existe uma constante $M > 0$ tal que $\|f(x) - f(y)\| \leq M \|x - y\|, \forall x, y \in \mathcal{S}$. A M chama-se constante de Lipschitz.

- *Métodos de Pesquisa Directa* (*Direct Search Methods*) – São os Métodos de Optimização sem Derivadas que avaliam a função objectivo num conjunto finito de pontos em cada iteração e decidem qual a melhor acção baseando-se exclusivamente nesses valores e sem explicita ou implicitamente aproximarem as derivadas ou construírem modelos.

  Dos Métodos de Optimização sem Derivadas, apresentados anteriormente, são Métodos de Pesquisa Directa os Métodos de Procura Directa Direccional ou Métodos de Pesquisa em Padrão e os Métodos de Procura Directa Simpléticos ou Métodos Simplex.

Tendo em vista o objectivo principal deste trabalho, será aprofundado, no Capítulo 2, o estudo dos Métodos de Pesquisa Directa. O estudo incide mais sobre Métodos de Pesquisa em Padrão e aos Métodos Simplex, uma vez que se têm mostrado os Métodos de Pesquisa Directa mais utilizados pelos investigadores das mais diversas áreas e já considerados clássicos.

1.6.2 Métodos de Optimização com Restrições

Existe um grande número de Métodos de Optimização com Restrições. No âmbito deste trabalho interessam particularmente os Métodos para Optimização não Linear que compreendem, nomeadamente:

- *Métodos de Programação Quadrática Sequencial* (SQP – *Sequential Quadratic Programming*) – Estes métodos consistem em generalizações do Método de Newton para Optimização sem Restrições na medida em que determinam a próxima iteração minimizando, através deste método, um modelo quadrático do problema;

- *Métodos de Penalidade e Barreira* (*Penalty and Barrier Methods*) – Os métodos de penalidade e barreira foram criados para resolver um problema com restrições, resolvendo uma sequência de problemas sem restrições, ou seja, é construída uma nova função objectivo, Φ, que contém informação relativa à função objectivo inicial, f, e simultaneamente às restrições do problema. São, desta forma, construídos sucessivamente problemas sem restrições, que dependem de um parâmetro positivo, r, parâmetro de penalidade, cujas correspondentes soluções $x^*(r)$ se pretende que

convirjam para a solução do problema inicial x^*. Assim, é efectuada uma transformação ao problema original e é feita a resolução de uma sequência de outros problemas, sem restrições, derivados do inicial, pelos métodos conhecidos para este tipo de problemas. Assim, a optimalidade (minimização da função objectivo) e a viabilidade (minimização da violação das restrições) são tratadas em conjunto;

- *Métodos de Lagrangeana Aumentada* (ALM – *Augmented Lagrangian Methods*) (apresentados, por exemplo, por Bertsekas em [18]) – Estes algoritmos têm por base a minimização sucessiva da função Lagrangena Aumentada, cujos parâmetros podem ser actualizados em cada iteração. Nesta formulação, inicialmente proposta apenas para problemas com restrições de igualdade, as restrições de desigualdade são convertidas em restrições de igualdade através da introdução de variáveis de folga não negativas, que passam a fazer parte das restrições de limites simples do problema. Esta estratégia tem o inconveniente de aumentar a dimensão do problema. Por este motivo foram desenvolvidas estratégias (variantes desta) que tentam evitar esse aumento.

  Esta abordagem é relativamente fácil de implementar, porque, em cada iteração, minimiza-se a função Lagrangeana Aumentada que é uma função suave, com restrições apenas de limites simples;

- *Métodos de Gradiente Reduzido* (RGM – *Reduced-gradient Methods*) – Estes algoritmos evitam o uso de parâmetros de penalidade procurando o mínimo do problema em curvas próximas do conjunto admissível;

- *Métodos de Programação Quadrática Sequencial Admissível* (FSQP – *Feasible Sequential Quadratic Programming*) – Nestes métodos todas as iterações são pontos admissíveis (tal como o seu nome sugere). Estes métodos exigem mais cálculos da função objectivo, mas são adequados para situações em que não é possível ou é difícil calcular valores da função em pontos não admissíveis ou quando não é aceitável que o processo de optimização pare num ponto não admissível;

- *Método dos Filtros* (*Filter Methods*) – O Método dos Filtros surgiu em alternativa aos métodos anteriores, nomeadamente aos métodos que têm o inconveniente de ser necessário estimar uma série de parâmetros em cada iteração.

Ao contrário desses métodos, o Método dos Filtros utiliza o conceito de dominância da optimização multi-objectivo, considerando a optimalidade e a viabilidade separadamente. Assim, o problema é transformado num problema biobjectivo, com dois problemas de minimização sem restrições. Em cada iteração há, assim e por esta ordem, duas fases: uma fase de viabilidade e uma fase de optimalidade.

No Capítulo 3 faz-se uma breve revisão a estes métodos, nomeadamente aos Métodos de Penalidade e Barreira, de Lagrangeana Aumentada e Método dos Filtros, por serem os que permitem a resolução dos problemas sem utilização de derivadas.

1.7 Critérios de Paragem

Antes de iniciar a descrição dos Métodos de Optimização, convém notar que são métodos iterativos. Assim descreve-se o que se entende por métodos iterativos e respectivos Critérios de Paragem, de acordo com Matias em [110] e Fernandes em [58]:

Definição 1.27. Uma sequência (ou sucessão) diz-se definida por iteração se a função F, que a define, é independente de k. A sequência resultante $x_k = F(x_{k-1}, ...)$ chama-se *sequência iterativa* gerada por F.

Definição 1.28. Uma sequência iterativa de números x_k converge para o limite x^* se, dada uma tolerância $\epsilon > 0$, existe um índice $N = N(\epsilon)$ de tal modo que para todos os $k \geq N$ se tem $|x_k - x^*| \leq \epsilon$. Assim, é *convergente* se $\lim_{k\to\infty} x_k = x^*$, sendo x^* um ponto fixo da equação, isto é $x^* = F(x^*)$.

Definição 1.29. Um *método iterativo* é definido por uma equação iterativa, com o qual se constroem aproximações à solução do problema. A implementação da equação iterativa obriga ao conhecimento de uma aproximação inicial e à definição de um conjunto de condições que dêem a garantia de que uma nova aproximação calculada, numa certa iteração, se encontra suficientemente perto da solução. Quando estas condições forem verificadas, pode parar-se o processo iterativo.

Assim, tal como a maioria dos métodos mais utilizados, na resolução de problemas de minimização de uma função não linear sem restrições, os Métodos de Pesquisa Directa (Capítulo 2), são, efectivamente, métodos iterativos: sendo conhecida a aproximação

inicial ao mínimo de f, $x^{(0)}$, o processo gera uma sucessão de aproximações, $x^{(k)}$, a x^*, mimimizante da função objectivo, f. Associada à sucessão $x^{(k)}$ existe a correspondente sucessão de valores de f, $f^{(k)}$.

Existem, segundo Fernandes em [58], duas questões muito importantes relacionadas com um método iterativo, para as quais convém obter resposta antes de se iniciar o processo:

- A primeira diz respeito à convergência – interessa saber se o método iterativo converge ou não para a solução procurada. Devem, assim, sempre que possível, ser analisadas as condições necessárias e/ou suficientes de convergência de um método.
- Tendo a garantia de que o método vai convergir para a solução, a segunda questão a colocar consiste em saber qual a razão de convergência.

Ainda, segundo Fernandes em [58], a *razão de convergência do método*, em termos relativos, é medida normalmente em função do número de iterações que este leva até convergir para o mínimo.

A implementação de um método iterativo exige a realização de um número infinito de operações para se chegar à solução, no entanto, face à limitação de tempo e dos recursos disponíveis, o processo iterativo tem que ser terminado após um número finito de operações. Esta paragem deve ser feita com a ajuda de condições que, sendo verificadas, dão a garantia de que se está tão perto da solução quanto se pretende.

Definição 1.30. As condições, que permitem parar o processo iterativo após um número finito de operações e que garantem que se está perto da solução, definem o que é designado por *critério de paragem* de um processo iterativo.

As sucessões $x^{(k)}$ e $f^{(k)}$, resultantes da implementação dos Métodos de Pesquisa Directa, na grande maioria dos casos, também não são finitas. Impõe-se, assim, a definição de um critério de paragem do processo iterativo, que garanta que a aproximação calculada na iteração corrente, $x^{(k)}$, seja considerada suficientemente próxima da solução, de acordo com uma tolerância pretendida. Essa proximidade é medida pela norma da diferença entre a aproximação $x^{(k)}$ e a solução x^*, $\left\|x^{(k)} - x^*\right\|$.

Se a sucessão de valores, $x^{(k)}$, se aproxima cada vez mais de x^*, a norma da diferença tende para zero, quando $k \to +\infty$. Como o valor de x^* não é conhecido é de esperar

que $x^{(k+1)}$ seja a melhor aproximação de x^* conhecida, e toma-se $\left\|x^{(k+1)} - x^{(k)}\right\|$ como uma estimativa do erro $\left\|x^* - x^{(k)}\right\|$ na iteração k. Assim, o processo iterativo pode terminar se verificar as *condições de proximidade*, (segundo Matias em [110]), ou seja, quando $\left\|x^{(k+1)} - x^{(k)}\right\|$ atinge um valor com a precisão pré-definida. O uso individual deste critério não é aconselhável, uma vez que a variação de $|f^* - f^{(k)}|$ pode ainda ser grande. Torna-se então necessário introduzir outras condições, uma delas, tal como no caso anterior, poderá ser a quantidade

$$|f^{(k+1)} - f^{(k)}| \tag{1.23}$$

que se espera que assuma valores pequenos, pois, como o objectivo é minimizar f, pretende-se que esta decresça de iteração para iteração, formando uma sucessão de valores decrescente em k, convergente para $f(x^*)$.

Dependendo do método que se está a utilizar, e da informação disponível, pode ser usado um critério de paragem diferente. Fernandes em [58], propõe o critério de Himmelblau e o critério de Gill e Murray, para o caso de se conhecerem e serem contínuas as primeiras derivadas da função objectivo.

Nos Métodos de Pesquisa Directa não se faz uso directo das derivadas, pelo que estes critérios não se aplicam neste trabalho. No método de Nelder-Mead, por exemplo, o último vértice aceite na iteração k, $x^{(rec)}$, nem sempre é o de menor valor da função. Além disso, de uma iteração para outra o melhor vértice pode manter-se inalterável, verificando-se $x^{(k+1)} = x^{(k)}$ (melhores aproximações nas iterações $k+1$ e k), com a consequente paragem inapropriada do processo iterativo.

Assim, Matias em [110], optou por introduzir uma expressão que compara a melhor aproximação com a mais recente, $\left\|x^{(k+1)} - x^{(rec)}\right\|$, como estimativa do erro $\left\|x^* - x^{(k)}\right\|$ na iteração k. Não esquecendo o uso de condições de proximidade em termos relativos, elegeu as condições:

$$\frac{\left\|x^{(k+1)} - x^{(k)}\right\|}{\left\|x^{(k+1)}\right\|} \leq \epsilon_1 \tag{1.24}$$

e

$$\frac{\left\|x^{(k+1)} - x^{(rec)}\right\|}{\left\|x^{(k+1)}\right\|} \leq \epsilon_2 \tag{1.25}$$

para ϵ_1 e ϵ_2 quantidades positivas próximas de zero.

E, para problemas que convergem para soluções nulas, substitui as relações (1.24) e (1.25), respectivamente por:

$$\left\|x^{(k+1)} - x^{(k)}\right\| \leq \epsilon_1 \tag{1.26}$$

e

$$\left\|x^{(k+1)} - x^{(rec)}\right\| \leq \epsilon_2 \tag{1.27}$$

Para evitar a implementação das duas alternativas, Matias em [110], usou a desigualdade $\left\|x^{(k+1)} - x^{(k)}\right\| \leq \epsilon_1 \left\|x^{(k+1)}\right\|$, como critério de paragem. Esta desigualdade pode ser usada em ambas as situações, sem ter que se verificar a possível aproximação a soluções nulas.

Para medir a proximidade dos valores da sucessão, a expressão $|f^{(k+1)} - f^{(k)}|$, segundo Matias em [110], também não se adequa à especificidade do método de Nelder-Mead, pelo que, à semelhaça do trabalho de Wolfe em [158], utilizou a variância dos valores de f, nos $n+1$ vértices:

$$var(f) = \frac{1}{n}\sum_{i=1}^{n+1} (\bar{f} - f_i)^2 \leq \epsilon_3 \tag{1.28}$$

em que $\bar{f}$ é a média dos f_i, $i = 1, ..., n+1$.

No artigo original [119], Nelder e Mead usaram esta condição, referindo que possuía limitações. Em alguns casos o processo iterativo parava mesmo antes de se aproximar do mínimo do problema. Além de (1.28) Matias em [110] também usou a diferença entre os valores extremos da função, no sentido de recolher mais informação sobre a proximidade de valores da sucessão f_k. Esta diferença foi proposta em diversos textos, tais como os

de Kelley em [88] e [89]:

$$|f_p - f_m| \leq \epsilon_4 \tag{1.29}$$

As constantes positivas ϵ_3 e ϵ_4 são escolhidas de acordo com a precisão que se pretender atingir.

Ainda podem ser adoptados outros critérios, como os apresentados por Dennis e Woods em [51] e por Fernandes em [58], onde se considera como critério de paragem deste processo iterativo a medida do tamanho relativo do simplex.

Se

$$\frac{1}{\Delta} \underset{2\leq i\leq n+1}{max} \|x_i - x_1\| \leq \epsilon \tag{1.30}$$

onde x_i, $i = 1, ..., n+1$ são os vértices do simplex e ϵ uma quantidade pequena e positiva, então Fernades em [58] afirma que o minimizante local está muito perto e o processo iterativo pode ser parado, caso contrário, o processo iterativo deve continuar. O valor de Δ é definido por $max(1, ||x_i||)$, sendo $||.||$ a norma 2 de um vector e sendo x_1 o melhor vértice do simplex.

Em [148] Tseng apresenta um critério de paragem baseado no diâmetro do simplex e na diferença entre valores da função em determinados vértices do simplex:

$$diam(S) \leq \epsilon \tag{1.31}$$

e

$$||\tilde{g}_f|| \leq \epsilon, \tag{1.32}$$

Esta última condição identifica aproximações a pontos de estacionaridade, para uma certa tolerância $\epsilon > 0$, em que $\tilde{g}_f$ é uma aproximação ao gradiente do simplex .

Assim, a escolha do critério de paragem deve ser adequada ao método que se está a utilizar, visando a garantia da convergência para a solução. Dentro destas condições a escolha e utilização dos critérios são facultativas.

Capítulo 2

Optimização não Linear sem Restrições

2.1 Introdução

Muitos dos problemas que aparecem tanto no mundo industrial, como no meio académico apresentam obstáculos difíceis de ultrapassar, nomeadamente no que diz respeito à sua formulação matemática e à escolha do método adequado para os resolver.

Os problemas de optimização não são alheios a estes obstáculos. A função objectivo é muitas vezes, por exemplo, resultado de medições físicas, químicas ou económicas e portanto, não se conhece a sua expressão analítica, ou é demasiado complexa ou dispendiosa (seja em tempo, custo computacional ou mesmo monetário), e o cálculo das derivadas é muito difícil ou nem sequer existem em pontos de interesse. Noutros casos pode ainda demorar muito tempo a encontrar um bom modelo para a função objectivo ou para o encontrar pode ser necessária a resolução de outros problemas.

Essas funções são encontradas frequentemente em problemas reais. Por estes motivos tem havido uma forte insistência dos especialistas para encontrar algoritmos e *software* que os permitam resolver o melhor possível.

Nestes casos torna-se essencial utilizar Métodos de Optimização em que o cálculo, ou a verificação da existência, das derivadas não seja necessário: os *Métodos de Optimização*

sem Derivadas. Conn et. al. em [32] apresentam alguns exemplos de funções deste tipo e suas aplicações.

Se, além disso, ainda for difícil encontrar um bom modelo para a função objectivo (por exemplo, para se obter um modelo quadrático duma função objectivo são necessários $\frac{(n+1)(n+2)}{2}$ cálculos da função objectivo, segundo Conn et. al. em [32] e Vaz em [152], onde n representa a dimensão do problema) então, nem todos os Métodos de Optimização sem Derivadas são adequados para resolver o problema. Nesse caso, devem usar-se métodos que além de não requererem o cálculo de derivadas também não requerem a aproximação de funções através de modelos: os *Métodos de Pesquisa Directa*.

Apesar de já terem surgido há muito tempo, a maioria destes métodos são eficazes a encontrar óptimos e fáceis de programar, talvez por isso continuem a ser tema de interesse. De notar, que não são o melhor caminho na resolução de todos os problemas de optimização, mas podem ser a única forma de abordagem em alguns.

Não é fácil dizer o que é um Método de Pesquisa Directa, até porque as abordagens neste sentido são diversas, mas a designação *Pesquisa Directa* (*Direct Search*) apareceu por Hooke e Jeeves em [83]. Eles descreveram a pesquisa directa como sendo:

Definição 2.1. A designação *Pesquisa Directa* representa a análise de soluções testadas sequencialmente, comparando cada uma delas com a "melhor" solução obtida até aí, conjuntamente com uma técnica de determinação de qual será a próxima solução a testar (usando valores anteriores da função objectivo).

Depois desta definição de Hooke e Jeeves apareceram diversas outras definições de Métodos de Pesquisa Directa, considerando que para o ser bastava que não usassem derivadas, mas poderiam usar aproximações destas. Segundo Trosset em [147] esta descrição não caracteriza completamente o que é a Pesquisa Directa.

Considere-se o Problema de Optimização sem Restrições (1.1), que se reproduz a seguir:

$$\underset{x\in\mathbb{R}^n}{minimizar}\ f(x) \tag{2.1}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é a função objectivo, da qual se assume que f é continuamente diferenciável numa vizinhança do minimizante do problema x^*.

A maioria das abordagens para optimização usam a análise clássica da expansão em série de Taylor da função objectivo. De facto, pode classificar-se a maioria dos métodos numéricos de optimização de acordo com quantos termos da expansão são usados. O método de Newton, que assume o conhecimento das 1^{as} e 2^{as} derivadas de f e que usa o Polinómio de Taylor de 2^a ordem para construir aproximações locais quadráticas de f, é um método de 2^a ordem. O método de passos descendentes (*Steepest descent*) que assume o conhecimento das primeiras derivadas e usa o Polinómio de Taylor de 1^a ordem para construir aproximações lineares locais de f, é um método de 1^a ordem. Nesta taxonomia, métodos de ordem zero (*zero-order methods*) não requerem informação sobre as derivadas e não constroem aproximações de f. Estes são os Métodos Directos, que são frequentemente chamados métodos de ordem zero na comunidade de optimização de engenharia.

Hooke e Jeeves ainda consideraram que a Pesquisa Directa também implica que não seja necessário usar valores numéricos da função objectivo, sendo que é suficiente a sua posição relativa. Assim, dados os pontos $x_1, x_2, x_m \in \mathbb{R}^m$, basta conhecer os índices $i_1, i_2, ..., i_m$ tais que: $f(x_{i_1}) \leq f(x_{i_2}) \leq ... \leq f(x_{i_m})$, não se pressupondo o acesso a valores numéricos da função.

Neste trabalho consideram-se Métodos de Pesquisa Directa de acordo com a definição inicial de Hooke e Jeeves.

Apesar dos Métodos de Pesquisa Directa terem aparecido há muito tempo existem diversas razões para ainda continuarem a ser usados:

- Porque na prática funcionam bem e análises recentes demonstraram garantir razoavelmente a convergência global;
- Têm sucesso quando outras abordagens falham;
- São suficientemente directos para implementar e podem ser aplicados quase imediatamente a muitos Problemas de Optimização Não Linear;
- Os requisitos para a sua utilização são mínimos e os próprios algoritmos requerem a escolha de poucos parâmetros.

Em 1991, o interesse pelos Métodos de Pesquisa Directa aumentou com a análise da convergência, por Torczon em [144]. Este artigo permitiu verificar que, para além de

muitas vezes serem a única opção para a resolução de Problemas de Optimização, para a maioria também se garante a convergência.

Neste Capítulo apresentam-se sucintamente os Métodos de Pesquisa Directa Clássicos com principal incidência nos *Métodos de Pesquisa em Padrão* e nos *Métodos Simplex*, que se têm mostrado os Métodos de Pesquisa Directa considerados clássicos e mais utilizados pelos investigadores das mais diversas áreas.

Segundo Conn et. al. em [32] estes dois tipos de métodos estão relacionados, na medida em que se pode construir facilmente uma base maximal (ver Definição 2.4) para qualquer simplex (ver Definição 2.5) de $n+1$ vértices e, reciprocamente, dada uma base positiva é fácil identificar simplexes de $n+1$ vértices.

2.2 Métodos de Pesquisa Directa Clássicos

Os Métodos de Pesquisa Directa Clássicos, para minimização sem restrições, podem ser subdivididos, segundo Lewis et. al. em [99], em três tipos básicos, que também são consideradas por Kolda et. al. em [91]:

- Métodos de Pesquisa em Padrão (PSM – *Pattern Search Methods*);
- Métodos Simplex (SM – *Simplex Methods*);
- Métodos com Conjuntos de Direcções de Pesquisa Adaptativas (MASD – *Methods with Adaptive Sets of Search Directions*).

Os recentes desenvolvimentos em Métodos de Pesquisa Directa, e os Métodos de Pesquisa Directa mais usuais, podem, segundo os autores em [99], ser subdivididos nestes três tipos, pois estes métodos nunca sofreram modificações das premissas básicas: todos eles se caracterizam por terem subjacente a selecção de direcções de pesquisa mais promissoras na procura de um óptimo.

Existem ainda outros métodos, um exemplo será o de Lucidi e Sciandrome, apresentados em [102] e [103], que utilizam propriedades e características dos anteriores (neste caso, dos métodos de Pesquisa Padrão e dos métodos de Conjuntos de Direcções de Pesquisa Adaptativas).

2.3 Métodos de Pesquisa em Padrão

2.3.1 Caracterização

Os Métodos de Pesquisa em Padrão (PSM) foram introduzidos por volta de 1952 por Fermi e Metropolis, mas foram designados por Métodos de Pesquisa Directa, pela primeira vez, em 1961 por Hooke e Jeeves em [83]. Estes métodos determinam possíveis pontos óptimos usando direcções fixas ao longo do processo iterativo: partindo duma iteração x_k, a próxima iteração será encontrada procurando num padrão ou grelha de pontos, nas direcções d, a uma distância δ_k, dita *tamanho do passo*. Podem ver-se na FIGURA 2.1 alguns exemplos de padrões ou grelhas.

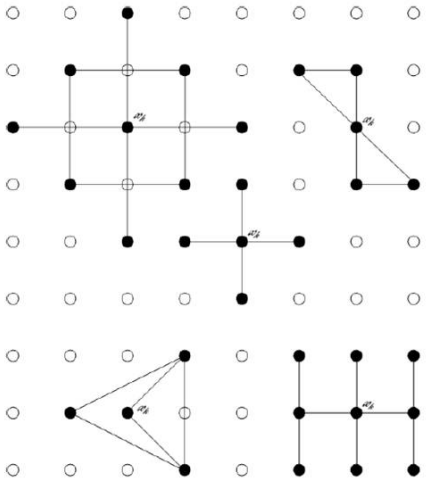

FIGURA 2.1: Alguns Exemplos de Padrões ou Grelhas
(Fonte: Espírito Santo em [56])

Segundo Lewis et. al. podem definir-se Métodos de Pesquisa em Padrão como sendo:

Definição 2.2. Os *Métodos de Pesquisa em Padrão* caracterizam-se por séries de movimentos exploratórios que consideram o comportamento da função objectivo, f, ao longo de um padrão de pontos, independentes de f.

Os movimentos exploratórios são efectuados tendo em conta um padrão de pontos e bases positivas, conceitos que se apresentam de seguida, de acordo com o que foi exposto por Conn et. al. em [32] e Vaz em [152].

Definição 2.3. Um conjunto de vectores v_i $(i = 1, ..., r)$ de $\mathbb{R}^n$ diz-se um *Conjunto Gerador Positivo* do Conjunto $A \in \mathbb{R}^n$ se qualquer vector de A pode ser escrito como

combinação linear positiva dos vectores v_i $(i = 1, ..., r)$, isto é:

$$A = \{v \in \mathbb{R}^n : v = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r, \alpha_i \geq 0, i = 1, 2, \cdots, r\}. \tag{2.2}$$

Nesse caso diz-se que o Conjunto Gerador Positivo gera positivamente A.

Definição 2.4. Um *Padrão de Pontos*, P em $\mathbb{R}^n$, é um conjunto:

$$P = \{P_i \in \mathbb{R}^n : P_i = x + v_i\}. \tag{2.3}$$

onde x é o ponto inicial de $\mathbb{R}^n$ e v_i $(i = 1, ..., r)$ são vectores de $\mathbb{R}^n$ que geram positivamente o espaço $\mathbb{R}^n$.

Uma *Base Positiva* em $\mathbb{R}^n$ é o conjunto positivamente independente, ou seja, nenhum dos seus vectores é combinação positiva dos restantes, cujo conjunto positivamente gerado é $\mathbb{R}^n$. Note-se que uma base positiva de $\mathbb{R}^n$ é formada no mínimo por $n + 1$ vectores (designada por *Base Minimal*) e não pode ser formada por mais do que $2n$ vectores (designada por *Base Maximal*).

Por exemplo, em $\mathbb{R}^2$ pode definir-se uma base positiva maximal a partir dos vectores da base canónica de $\mathbb{R}^2$, $\{e_1, e_2\} = \{(1, 0), (0, 1)\}$:

$$B_{\oplus M} = \{e_1, e_2, -e_1, -e_2\}.$$

De uma forma geral tem-se para $\mathbb{R}^n$:

$$B_{\oplus M} = \{\pm e_i, i = 1, ..., n\}. \tag{2.4}$$

Um exemplo, base positiva minimal em $\mathbb{R}^2$ é:

$$B_{\oplus m} = \{e_1, e_2, (-1, -1)\}.$$

Os conjuntos geradores positivos e as bases positivas são muito usados nos Métodos de Pesquisa em Padrão, uma vez que, sendo D um conjunto gerador positivo de $\mathbb{R}^n$, dado

um vector não nulo $v \in \mathbb{R}^n$, existe pelo menos um vector d em D tal que v e d formam um ângulo recto. Este facto é fundamental em optimização, pois sendo $v = -\nabla f(x)$, com f uma função continuamente diferenciável em x, qualquer vector d que forme um ângulo recto com v é uma direcção descendente de f em x (ou seja, uma direcção para a qual existe um $\bar{\alpha} > 0$ tal que $f(x + \alpha d) < f(x), \forall \alpha \in]0, \bar{\alpha}]$).

Assim, para minimizar a função objectivo f basta avaliar os pontos $x+\alpha d, \forall d \in D, \alpha > 0$ e, desde que $v = -\nabla f(x)$ seja não nulo, existe um α positivo e um vector d em D para os quais $f(x + \alpha d) < f(x)$. Repetindo o processo para valores cada vez menores de α caminha-se na direcção de um mínimo e o processo pode ser terminado depois de um número finito de reduções do parâmetro α. Deste modo, nestes métodos, pode considerar-se como critério de paragem o α ser suficientemente pequeno.

2.3.2 Método de Pesquisa Coordenada

O Método de Pesquisa Coordenada, que se passa a descrever, é o Método de Pesquisa Padrão mais básico, que, segundo Conn et. al. em [32], se deve a Fermi e Metropolis, Box e Wilson.

Seja x_k a iteração actual e α_k o tamanho do passo ou parâmetro da malha (rede). Neste método, o padrão é definido por:

$$P_k = \{P_i \in \mathbb{R}^n : P_i = x_k + \alpha_k d, d \in B_{\oplus M}\}$$

(a construção deste Padrão está ilustrado na FIGURA 2.2, onde $s_k = \alpha_k d$).

O ponto P_i, a uma distância α_k, na direcção d, da iteração anterior, que verifica uma redução simples ($f(P_i) < f(x_k)$) da função objectivo é a aproximação seguinte. Se em nenhuma das direcções houver redução do valor da função objectivo, então o passo é reduzido (usualmente para metade: $\alpha_{k+1} \leftarrow \frac{1}{2}\alpha_k$) e é testado um novo padrão de pontos, nas mesmas direcções, mas a uma distância menor do ponto inicial.

A ordem pela qual se pesquisa o padrão de pontos é, em geral, arbitrária, mas é fixada no início e mantém-se ao longo do processo. Algumas abordagens alteram essa ordem, usando informação das iterações anteriores, e iniciam a pesquisa pela direcção que na

iteração anterior se mostrou ser uma direcção de sucesso. Esta alteração da ordem de pesquisa não tem influência nos resultados de convergência, mas estes métodos não são considerados como os Métodos de Pesquisa Coordenada, mas sim como suas variações.

Existem ainda alguns, ditos *Métodos Oportunistas*, que aceitam o primeiro ponto do padrão que verifique uma redução simples da função objectivo, não fazendo a pesquisa em todos os pontos do padrão.

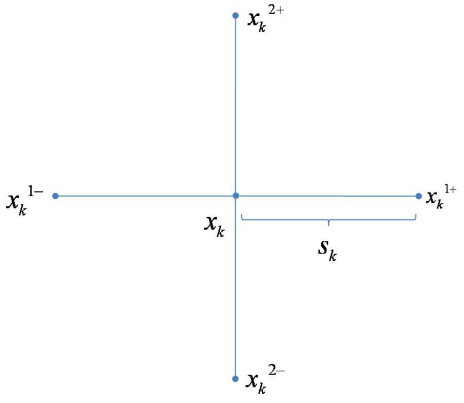

FIGURA 2.2: Padrão do Método de Pesquisa Coordenada

Dependendo da variante do método, o passo α_k pode ser reduzido, por multiplicação por um factor positivo, menor do que a unidade, pré-definido quando há insucesso, ou seja, não foi encontrado um ponto para o qual a função objectivo tenha um valor menor do que o anterior, podendo até aumentar, se houver sucesso. Na TABELA 2.1 apresenta-se o algoritmo genérico da Pesquisa Coordenada.

Kolda et. al. em [91] apontam vantagens e limitações dos métodos de Pesquisa em Padrão. Por um lado são fáceis de descrever e de implementar e inicialmente têm um progresso rápido na direcção da solução, mas apesar de rápidos na aproximação do minimizante podem ser lentos a detectá-lo de facto, ou seja, podem ser lentos a convergir. Outra vantagem dos Métodos de Pesquisa Padrão é que funcionam bem com uma grande variedade de problemas, de acordo com Lewis et. al. em [96] e Torczon em [145].

Estas características, nomeadamente as suas vantagens, incentivaram a criação e o estudo da convergência de diversos Métodos de Pesquisa em Padrão.

TABELA 2.1: Algoritmo de Pesquisa Coordenada

Algoritmo de Pesquisa Coordenada Fonte: Conn et. al. em [32].
Dados: Conhecidos o ponto inicial x_0, o tamanho do passo $\alpha_0 > 0$, o número máximo de iterações $nmax$ e os critérios de paragem.
Para $k = 0, 1, 2, ...$ fazer:

1. **Passo de Pesquisa** (ou Movimento Exploratório):
   - Considerar o padrão
   $$P_k = \{P_i \in \mathbb{R}^n : P_i = x_k + \alpha_k d, d \in B_{\oplus M}\};$$
   - Calcular o valor de f nos pontos do padrão seguindo a ordem pré-estabelecida;
     - Se é encontrado um ponto do padrão $x_k + \alpha_k d$ tal que $f(x_k + \alpha_k d) < f(x_k)$, então parar a pesquisa e fazer $x_{k+1} \leftarrow x_k + \alpha_k d$ e declarar a iteração e o passo de pesquisa como um *sucesso*;
     - Senão declarar a iteração e o passo de pesquisa como um *insucesso* e fazer $x_{k+1} \leftarrow x_k$;
2. **Actualização do parâmetro** ou tamanho do passo:
   - Se a iteração foi um sucesso fazer $\alpha_{k+1} \leftarrow \alpha_k$ (ou $\alpha_{k+1} \leftarrow \beta\alpha_k$, com $\beta > 1$ – usualmente $\beta = 2$);
   - Senão $\alpha_{k+1} \leftarrow \frac{1}{2}\alpha_k$ (ou $\alpha_{k+1} \leftarrow \beta\alpha_k$, com $0 < \beta < 1$ – usualmente $\beta = \frac{1}{2}$).
3. Enquanto não se atingir o número máximo de iterações e não se verificar o critério de paragem definido, voltar a 1., com $k \leftarrow k + 1$, senão concluir fazendo $x^* \leftarrow x_{k+1}$, $f^* \leftarrow f_{k+1}$.

2.3.3 Convergência e Variantes

Os Métodos de Pesquisa em Padrão podem, assim, ser diferenciados de acordo com:

- O padrão adoptado para os movimentos exploratórios nas sucessivas iterações;
- As direcções de procura utilizadas, d;
- O comprimento do passo, α_k.

Este último influencia directamente o deslocamento, s_k ($s_k = \alpha_k\ d$), e a forma da sua actualização. Em geral, o comprimento do passo mantém-se sempre que se verificar uma redução do valor da função, caso contrário, é reduzido por multiplicação de um factor menor do que um. O valor de α_k pode ser diferente para cada elemento do vector se a ordem de grandeza entre os seus elementos assim o justificar, desde que mantenha a estrutura algébrica dos possíveis iterandos, sendo o mais usual, tal como é detalhadamente apresentado por Lewis et. al. em [100] e Torczon em [146], reduzi-lo para metade quando o passo não é aceite e mantê-lo ou duplicá-lo quando é bem sucedido.

O conjunto de direcções de pesquisa, em algumas abordagens, permanece constante ao longo de todo o processo iterativo e noutras (por exemplo Torczon em [146]) dependem do passo, mantendo, em qualquer dos casos, a estrutura da rede definida pelo padrão.

Os Métodos de Pesquisa em Padrão podem usar um padrão pré-especificado, ou um que vai sendo adaptado, com o objectivo de pesquisar o espaço das variáveis e calcular soluções, segundo Espírito Santo em [55].

Note-se que o padrão é meramente conceptual, efectivamente nunca é construído. À medida que se vai fazendo a pesquisa vão-se testando pontos do padrão, mas eles não são definidos à partida.

Segundo Lewis et. al. em [99], por volta de 1971 surgiu uma demonstração, de Polak, da convergência global para um algoritmo simples de Pesquisa em Padrão e, no mesmo ano, Céa também publicou um texto no qual apresentou uma demonstração da convergência global do conhecido como algoritmo de Pesquisa em Padrão de Hooke e Jeeves.

2.3.3.1 Algoritmo de Hooke e Jeeves

O algoritmo de Hooke e Jeeves, é o algoritmo de Pesquisa em Padrão mais célebre e mais usado nas diversas áreas. Este algoritmo, que introduziu o conceito de Pesquisa Directa, difere do Método de Pesquisa Coordenada, introduzindo um novo passo, designado por *passo padrão* (ilustrado na FIGURA 2.3), no sentido de acelerar o processo, ao explorar a informação conseguida na pesquisa em iterações anteriores bem sucedidas . Por essa razão, o método é reconhecido como *oportunista*, segundo Espírito Santo em [55].

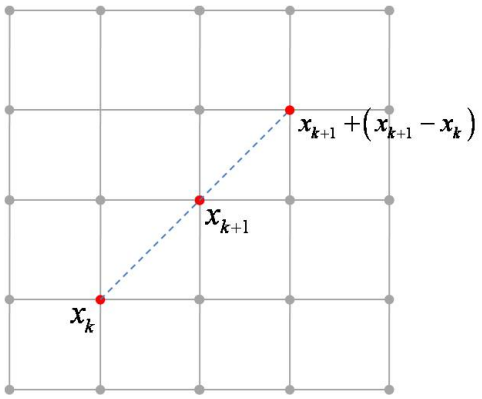

FIGURA 2.3: Passo Padrão – Algoritmo de Hooke e Jeeves

Neste algoritmo, existem, então, duas etapas:

- ***Passo Padrão*** – Se a iteração anterior for bem sucedida (produziu decréscimo da função objectivo) então a direcção $x_{k+1}-x_k$ é considerada uma direcção promissora, pelo que a pesquisa em vez de ser em torno de x_{k+1}, como na Pesquisa Coordenada, é feita em torno de uma iteração provisória dada por: $x_p = x_{k+1} + (x_{k+1} - x_k)$.

- ***Movimento Exploratório*** – É feita a Pesquisa Coordenada em torno da iteração provisória, x_p. Se a procura coordenada nesse ponto for bem sucedida, o ponto é aceite, senão é feita a pesquisa coordenada em torno de x_{k+1}.

Se a iteração anterior não foi bem sucedida $x_{k+1} \leftarrow x_k$ e a iteração limita-se à procura coordenada em torno de x_{k+1}. Na TABELA 2.2 apresenta-se o algoritmo de Hooke e Jeeves.

2.3.3.2 Outras Variantes

Depois do artigo de Céa, a teoria geral de pesquisa directa ter-se-á estendido à análise da convergência global, por Torczon em [144, 146], entre outros, e aos algoritmos de pesquisa multidireccional, pelo mesmo autor em [143], onde tal como os algoritmos simplex, da próxima secção, a pesquisa se processa por reflexão do simplex directamente do centróide de uma das faces.

Em [146] Torczon apresentou resultados relativos à convergência para um ponto de estacionaridade quando aplicado a problemas suaves, de dimensão n, em que os vértices do poliedro se mantêm sobre uma rede de pontos (*rational lattice*).

Audet e Dennis em [7] designam os algoritmos definidos e analisados por Torczon em [146] como Métodos de Pesquisa em Padrão Generalizados (GPS) para optimização sem restrições e sem utilização de derivadas.

Nesse trabalho Torczon impõe restrições adicionais ao conjunto das direcções de pesquisa, nos procedimentos para o cálculo dos movimentos exploratórios e para a actualização do comprimento do passo, para estabelecer a convergência.

A análise de convergência realizada por Torczon em [146] considera o gradiente de f, por isso pressupõe que f é continuamente diferenciável numa vizinhança do conjunto de nível compacto relativo a x_0, aproximação inicial do processo.

TABELA 2.2: Algoritmo de Hooke e Jeeves

Algoritmo de Hooke e Jeeves Fonte: Matias em [110].
Ao iniciar o algoritmo supõem-se conhecidos a função objectivo f, a dimensão do problema n, uma estimativa inicial x_0, o número máximo de iterações $nmax$, os critérios de paragem, o factor δ e as direcções $\xi_i, i = 1, ..., n$.

1. Iniciar $x_a \leftarrow x_0$, fazer $f_a \leftarrow f(x_0)$ e $k \leftarrow 1$;
2. Iniciar o índice, fazendo $i \leftarrow 1$;
3. Fazer: $x_b \leftarrow x_a + \delta\xi_i$ e $f_b \leftarrow f(x_b)$;
4. Se houver um decréscimo no valor da função, isto é, se $f_b < f_a$:
   (a) Então, fazer: $x_a \leftarrow x_b$, $f_a \leftarrow f_b$, $x_b \leftarrow x_a + \delta\xi_i$ e $f_b \leftarrow f(x_b)$;
   Se $f_b > f_a$, avançar para o passo 5;
   Senão, verificou-se uma nova diminuição no valor da função, actualizar os valores, fazendo: $x_a \leftarrow x_b$ e $f_a \leftarrow f_b$ e avançar para o passo 5;
   (b) Senão, fazer: $x_b \leftarrow x_a - \delta\xi_i$ e $f_b \leftarrow f(x_b)$;
   Se $f_b > f_a$, avançar para o passo 5;
   Senão, verificou-se um decréscimo da função, actualizar os valores, fazendo: $x_a \leftarrow x_b$, $f_a \leftarrow f_b$, $x_b \leftarrow x_a - \delta\xi_i$ e $f_b \leftarrow f(x_b)$;
   - Se $f_b > f_a$, avançar para o passo 5;
   - Senão, verificou-se um decréscimo da função, actualizar os valores, fazendo: $x_a \leftarrow x_b$ e $f_a \leftarrow f_b$ e continuar;
5. Se $i \neq n$, incrementar i fazendo $i \leftarrow i + 1$ e voltar ao passo 3;
   Caso Contrário:
6. Se o critério de paragem, T1 não for verificado (ver (1.7))
   (a) Então:
   Se $k \geq nmax$(T2), finalizar avançando para o passo 11;
   Senão, avançar para o passo 7:
   (b) Senão: Parar, pois já se está perto do mínimo, finalizando com o passo 11.
7. Preparar nova iteração, incrementando o valor de k, isto é, $k \leftarrow k + 1$; Definir uma direcção prévia de pesquisa unidimensional, $\xi_a \leftarrow x_a - x_0$; Reiniciar o ponto de partida, fazendo $x_0 \leftarrow x_a$;
8. Actualizar δ;
9. Fazer $x_b \leftarrow x_a + \delta\xi_a$ e $f_b \leftarrow f(x_b)$
10. Se houver um decréscimo do valor da função, isto é, se $f_b < f_a$:
    (a) Então, fazer $x_a \leftarrow x_b$, $f_a \leftarrow f_b$, $x_b \leftarrow x_a + \delta\xi_a$ e $f_b \leftarrow f(x_b)$;
    Se $f_b > f_a$, voltar ao passo 2;
    Senão, verificou-se um novo decréscimo da função, actualizar os valores fazendo: $x_a \leftarrow x_b$ e $f_a \leftarrow f_b$ e voltar ao passo 2;
    (b) Senão, fazer $x_b \leftarrow x_a + \frac{\delta}{2}\xi_a$ e $f_b \leftarrow f(x_b)$;
    Se $f_b > f_a$, voltar ao passo 2;
    Senão, verificou-se um novo decréscimo da função, actualizar os valores fazendo: $x_a \leftarrow x_b$ e $f_a \leftarrow f_b$ e voltar ao passo 2;
11. Enquanto não se atingir o número máximo de iterações e não se verificar o critério de paragem definido, voltar a 1., com $k \leftarrow k + 1$, senão concluir fazendo $x^* \leftarrow x_a$, $f^* \leftarrow f_a$.

Em 2002, Lewis e Torczon em [100] utilizam o Método de Pesquisa em Padrão para resolver problemas com restrições, introduzindo variáveis de folga. Nesta área podem ainda referir-se os trabalhos dos mesmos autores [94, 96, 97, 97]. Em [94] ainda introduzem a ideia de usar direcções positivas com Métodos de Pesquisa em Padrão Generalizados (GPS). Em [97], mostraram que se f é continuamente diferenciável e se o conjunto de direcções for apropriadamente escolhido, de acordo com a região admissível, então a teoria GPS e de convergência estende-se à optimização com restrições. Em [98] mostram os mesmos resultados para problemas com um número finito de restrições lineares. Ambas as extensões usam a estratégia de "barreira" ou seja, não permitem que a última iteração seja não admissível.

Em [7] Audet e Dennis apresentam uma análise dos métodos propostos por Lewis e Torczon em [97, 98] e [146] e estabelecem uma relação entre o algoritmo, as direcções de pesquisa e as propriedades de continuidade diferenciável local da função objectivo em pontos limite específicos do algoritmo.

O que distingue o Método de Pesquisa em Padrão de Lewis e Torczon em [94] para problemas sem restrições e o Método para problemas com limites simples nas variáveis, dos mesmos autores em [98], são as duas restrições que são adicionadas para garantir a convergência para um ponto de estacionaridade, isto é, um ponto admissível que verifica a condição necessária de optimalidade de primeira ordem e, portanto, mantém a admissibilidade ao longo das pesquisas. A outra restrição garante que existe pelo menos uma direcção admissível e de descida quando o ponto admissível calculado não é ainda ponto de estacionaridade.

O algoritmo dos Métodos de Pesquisa em Padrão para problemas com restrições, presente em [97], não requer qualquer redução significativa dos valores da função, embora assegure pelo menos uma direcção admissível descendente em qualquer ponto admissível de não estacionaridade.

Este método continua a ser um Método de Pesquisa em Padrão, no entanto, é adicionada uma condição que atribui um valor à função arbitrariamente grande a qualquer passo que leve a pesquisa para fora da região admissível. Mais uma vez a análise de convergência global de Lewis e Torczon em [97] estabelece que se o conjunto de nível de f, $L(x_0)$, para uma aproximação inicial admissível x_0, for compacto, se f for continuamente diferenciável numa vizinhança aberta contida em $L(x_0)$ e se os movimentos exploratórios,

a actualização do comprimento do passo e as restrições geométricas do próprio padrão verificarem certas condições, a sequência de iterandos gerada, pelo método generalizado de pesquisa em padrão, para problemas com restrições lineares, converge para um ponto admissível e de estacionaridade (segundo Conn et. al. em [32] e Lewis et. al. em [99]).

Além destes estudos não se pode ficar alheio a trabalhos em Pesquisa em Padrão, como, entre muitos outros, o de Hart em [79], de Alberto et. al. em [3], o de Custódio e Vicente em [47], o de Hedar et. al. em [80] e Vaz e Vicente em [151] e Vicente em [153], que permitem afirmar que os Métodos de Pesquisa Padrão são tema de grande interesse para a investigação.

As abordagens anteriores possibilitam provas de convergência para o caso de f ser suave. No caso de não se ter uma função desse tipo não é tão fácil provar a convergência global para um ponto estacionário. Conn et. al. em [32] apresentou alguns exemplos de funções não suaves para as quais a Pesquisa em Padrão pode falhar.

Em 2006 Audet e Dennis em [10] apresentam um método de pesquisa em padrão notável, o MADS (*Mesh Adaptive Direct Search*). Este trabalho é notável porque os autores mostram a convergência para o caso de problemas não suaves (com e sem restrições). De facto, nenhum outro método ainda permitiu estas provas. Segundo os autores este trabalho consiste numa extensão dos métodos GPS de Torczon em [144] e Coope and Price em [34]. Eles afirmam que a principal vantagem do método MADS relativamente aos métodos GPS é que a pesquisa, no passo de Sondagem, é feita num conjunto denso em $\mathbb{R}^n$ e não se restringe a um conjunto finito de direcções.

Posteriormente, Audet et. al. em [2, 11], apresentam duas versões desse método (MADS-PB e OrthoMADS) na tentativa de optimizar as direcções pesquisadas, tendo em conta a forma como lidam com o tratamento das restrições do problema. Em [12] Audet et. al. apresentam uma comparação entre estas três versões do método no que diz respeito às diferentes estratégias para lidar com restrições.

Estes métodos de pesquisa em padrão caracterizam-se por procurar uma nova iteração x_{k+1} em duas fases:

- ***Passo de Sondagem*** (*Search Step*):

O Passo de Sondagem é opcional, e não tem implicações na convergência dos métodos, e consiste no cálculo do valor da função objectivo num número finito de pontos. A escolha destes pontos é arbitrária desde que sejam em número finito.

- ***Passo de Pesquisa*** (*Poll Step*):

  O Passo de Pesquisa só é implementado se o Passo de Sondagem não obteve sucesso. Este passo consiste na usual Pesquisa Coordenada em torno da iteração actual.

Na TABELA 2.3 apresenta-se a forma geral dos algoritmos de Audet et. al. (*Directional Direct-Search Methods*).

TABELA 2.3: Algoritmo de Audet

Algoritmo de Audet Fonte: Conn et. al. em [32].
Dados: Conhecidos x_0, $\alpha_0 > 0$, $0 < \beta_1 \leq \beta_2 < 1$, $\gamma \geq 1$ o número máximo de iterações e o critério de paragem e sendo $\mathcal{D}$ um conjunto de bases positivas.
Para $k = 0, 1, 2, ...$ fazer:

1. **Passo de Sondagem**: Tentar encontrar um ponto x tal que $f(x) < f(x_k)$ calculando o valor de f num número finito de pontos (em geral é pesquisado o ponto na direcção que na iteração anterior foi um sucesso, neste caso o método é *oportunista*). Se esse ponto for encontrado então $x_{k+1} \leftarrow x$, declara-se a iteração e o passo de sondagem com sucesso e avança-se para o passo de pesquisa, senão faz-se o passo de pesquisa em torno de x_k.
2. **Passo de Pesquisa**:
   - Escolher uma base positiva D_k de $\mathcal{D} \subset \mathbb{R}^n$;
   - Considerar o padrão $P_k = \{P_i \in \mathbb{R}^n : P_i = x_k + \alpha_k d, d \in B_{\oplus M}\}$;
   - Calcular o valor de f nos pontos do padrão seguindo a ordem pré-estabelecida;
     - Se é encontrado um ponto do padrão $x_k + \alpha_k d$ tal que $f(x_k + \alpha_k d) < f(x_k)$ então parar a pesquisa e fazer $x_{k+1} \leftarrow x_k + \alpha_k d$ e declarar a iteração e o passo de pesquisa como um *sucesso*;
     - Senão declarar a iteração e o passo de pesquisa como um *insucesso* e fazer $x_{k+1} \leftarrow x_k$;
3. **Actualização do parâmetro da grelha**:
   - Se a iteração foi um sucesso então manter ou aumentar o tamanho do passo: $\alpha_{k+1} \in [\alpha_k, \gamma\alpha_k]$;
   - Senão diminuir o tamanho do passo $\alpha_{k+1} \in [\beta_1\alpha_k, \beta_2\alpha_k]$.
4. Enquanto não se atingir o número máximo de iterações e não se verificar o critério de paragem definido, voltar a 1., com $k \leftarrow k + 1$, senão concluir fazendo $x^* \leftarrow x_{k+1}$, $f^* \leftarrow f_{k+1}$.

O Critério de paragem usual nestes métodos é parar o processo quando $\alpha_k < \alpha_{tol}$, para uma dada tolerância $\alpha_{tol} > 0$ (geralmente $\alpha_{tol} = 10^{-5}$, segundo Conn et. al. em [32]). O estudo dos aspectos essenciais da Convergência Global deste método foi apresentado pelos autores, nos vários trabalhos desenvolvidos, e sumariamente apresentado por Conn et. al. em [32].

2.4 Métodos Simplex

2.4.1 Caracterização

Num artigo datado de 1962, Spendley et. al. em [142], apresentaram o primeiro método simplex conhecido. Eles observaram que podiam utilizar não mais do que $n+1$ valores da função objectivo para identificar a direcção de subida (descida) rápida, em vez dos usuais $2n$ ou 2^n, dos anteriores métodos de Pesquisa Directa, segundo Lewis and Torczon em [99]. Esta é a ideia principal da pesquisa simplex: construir um simplex não degenerado em $\mathbb{R}^n$ e usá-lo para conduzir a pesquisa.

A definição de Simplex apresentada de seguida baseia-se nas apresentadas por Matias, Kelley e Conn et. al. em [32, 89, 110]:

Definição 2.5. Um *Simplex* S, é um conjunto de $n+1$ pontos, $\{x_i\}_{i=1}^{n+1}$, em $\mathbb{R}^n$, ou seja, é um poliedro em $\mathbb{R}^n$ com $n+1$ faces, onde x_i é o i-ésimo *Vértice do Simplex* S.

Assim tem-se, como exemplos de simplex, um segmento de recta em $\mathbb{R}$, um triângulo em $\mathbb{R}^2$, um tetraedro em $\mathbb{R}^3$, etc. Note-se ainda que, segundo Matias em [110]:

Definição 2.6. Um simplex $\{x_1, x_2, \cdots, x_{n+1}\}$ é *não degenerado* se o conjunto

$$\{x_1 - x_i, x_2 - x_i, \cdots, x_{i-1} - x_i, x_{i+1} - x_i, \cdots, x_{n+1} - x_i\}, \forall i \in 1, 2, ..., n$$

for linearmente independente, isto é, se o conjunto de arestas adjacentes a qualquer vértice do simplex (neste caso x_i) formar uma base para o espaço $\mathbb{R}^n$.

Neste caso, qualquer ponto no domínio de pesquisa pode ser escrito como combinação linear das arestas adjacentes de qualquer vértice.

Os Métodos Simplex caracterizam-se por modificarem as direcções de pesquisa no final de cada iteração, notando-se ainda que se se substituir um dos vértices pela reflexão desse vértice a partir do centróide da face oposta, então o resultado também é um simplex (simplex reflectido), o que significa que na pesquisa de um óptimo pode simplesmente reflectir-se um vértice de cada vez.

Uma vez construído o primeiro simplex, o único movimento especificado no algoritmo simplex original de Spendley et. al. é a reflexão isométrica [1] do pior vértice em relação ao centróide dos n melhores vértices (conforme se ilustra FIGURA 2.4). Este movimento primeiro identifica o *pior* vértice no simplex (isto é, o que está mais longe do valor pretendido) e depois reflecte o pior vértice a partir do centróide da face oposta. Este movimento do *pior* vértice na direcção geral dos restantes vértices tem a expectativa de eventuais melhorias no valor da função objectivo.

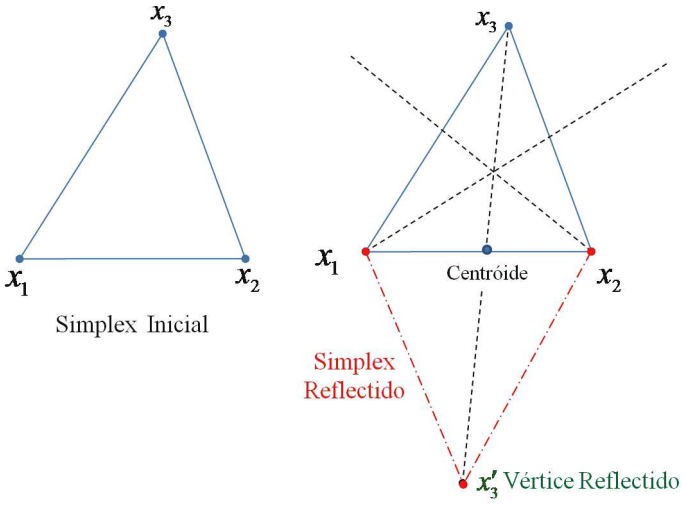

FIGURA 2.4: Movimento de reflexão de Spendley et. al.

Se o vértice reflectido continua a ser o *pior* vértice, então escolhe-se o segundo pior vértice e repete-se o processo. Quando um vértice reflectido produz um decréscimo no valor da função objectivo relativamente ao melhor vértice, tem-se um novo candidato a minimizante.

No final também se substitui o *melhor* vértice (isto é, o que tem um valor mais próximo do objectivo) ou considera-se que esse vértice é o candidato a minimizante.

Para o caso em que a reflexão fica atrás do pior vértice, no simplex original, Spendley, Hext e Himsworth sugeriram duas alternativas: reduzir ambos os comprimentos das arestas adjacentes do *melhor* vértice, reduzindo o espaço de procura, ou recorrer a um método de ordem superior para obter uma mais rápida convergência local.

Note-se que, neste método, o conjunto de ângulos internos do simplex permanece constante, mantendo uma geometria semelhante.

[1] Uma reflexão é isométrica se o ponto reflectido está à mesma do eixo de reflexão.

2.4.2 Método de Nelder-Mead

O Método de Nelder-Mead foi apresentado por Nelder e Mead em 1965 em [119] e é um método simplex com movimentos adicionais aos conhecidos até então, no sentido de acelerar a procura do óptimo do problema. O que Nelder e Mead introduziram, além do movimento de reflexão isométrica, foram movimentos não isométricos criados para acelerar a pesquisa. Estes movimentos são conhecidos como movimentos de expansão e contracção (para o interior e para o exterior), acompanhados de um passo de redução de todo o simplex ou passo encolhido em direcção ao melhor vértice, usado quando tudo o resto falha (estes movimentos são ilustrados na FIGURA 2.5, onde x_k representa o melhor vértice do simplex). Neste método o simplex pode assumir diversas formas e, consequentemente, os seus ângulos internos podem tornar-se arbitrariamente pequenos/ grandes. Esta flexibilidade torna extremamente difícil obter provas de convergência.

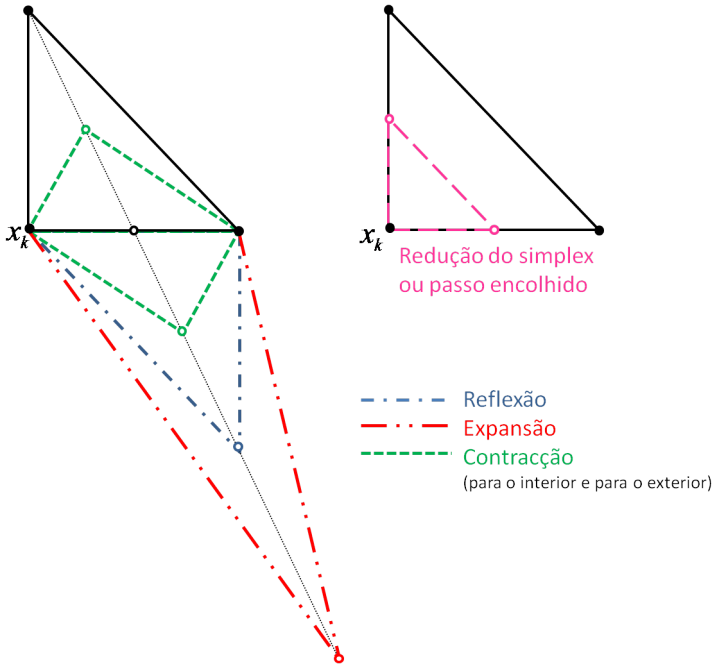

FIGURA 2.5: Movimentos de Nelder-Mead
(Fonte: Lewis et. al. em [99])

2.4.2.1 Descrição geral do Método de Nelder-Mead

Considere-se um problema de optimização sem restrições de dimensão n, tal como foi apresentado em (2.1).

Considerem-se conhecidos:

- Um ponto de partida $x^{(0)}$ (aproximação inicial à solução do problema, x^*);
- Um comprimento s para a medida da aresta do poliedro;
- Uma base de $\mathbb{R}^n$ (por exemplo, a base os vectores $e_i,\ i = 1, ..., n$, que formam a base ortonormada de $\mathbb{R}^n$).
- Os valores dos parâmetros α, γ e β, que devem verificar as condições:

$$0 < \gamma < \alpha \leq 1, \beta > 1.$$

O método começa com a construção dum poliedro em $\mathbb{R}^n$ com $n+1$ vértices distintos, isto é, define-se o simplex regular inicial $V = (v_i, f_i)$ tal que:

$$v_1 = x^{(0)}, v_i = x^{(0)} + s\ e_{i-1}\ (i = 2, ..., n+1)$$

(Note-se que se o poliedro possuir arestas com o mesmo comprimento, o simplex diz-se *regular*)

e calculam-se os correspondentes valores da função objectivo:

$$f_i = f(v_i),\ (i = 1, ..., n+1).$$

Para prosseguir o processo, em cada iteração, deve substituir-se um dos vértices do simplex (o pior) por outro novo ponto, por forma a se obter um novo simplex mais próximo da solução do problema. Para tal é necessário identificar os vértices que possuem os valores extremos da função em V:

- O *pior vértice*, v_p, que possui o valor mais alto da função objectivo

$$f_p = \underset{1 \leq i \leq n+1}{max} \{f_i\};$$

- O *melhor vértice*, v_m, que possui o menor valor da função objectivo

$$f_m = \underset{1 \leq i \leq n+1}{min} \{f_i\};$$

- O *segundo pior vértice*, v_{sp}, em que f_{sp} é o segundo valor mais alto da função objectivo

$$f_{sp} = \max_{\substack{1 \leq i \leq n+1 \\ i \neq p}} \{f_i\}.$$

Nota: Alguns autores (incluindo a versão original de Nelder-Mead) preferem ordenar os vértices do simplex por ordem crescente dos valores da função, tendo-se neste caso $v_m = v_1$, $v_{sp} = v_n$ e $v_p = v_{n+1}$. Este procedimento obriga a um maior número de ciclos e de comparações de valores relativamente ao aqui apresentado e que foi implementado por Matias em [110]. Nesta implementação apenas dois ciclos são necessários, sendo o primeiro utilizado para identificar os índices m e p e o segundo para o índice sp.

Após a identificação anteriormente referida é necessário determinar novos pontos e decidir qual a substituição a efectuar. A aceitação ou rejeição desses novos pontos será feita comparando os respectivos valores da função objectivo, tendo como base o centróide dos n melhores vértices.

As expressões que permitem calcular o centróide e os vértices auxiliares são apresentados na definição seguinte, seguindo o que foi exposto por Matias em [110].

Definição 2.7. Considere-se o simplex $\{v_i\}_{i=1}^{n+1}$ em $\mathbb{R}^n$ e que o pior vértice foi identificado como sendo o vértice de índice $p \in \{1, 2, ..., n+1\}$, então o *Centróide dos n melhores vértices* do simplex é dado por:

$$v_c = \frac{1}{n} \sum_{\substack{i=1 \\ i \neq p}}^{n+1} v_i, \tag{2.5}$$

e os pontos auxiliares serão:

- O *Vértice Reflectido*, obtido pela reflexão do pior vértice, v_p, no sentido do centróide dos restantes vértices:

$$v_r = v_c + \alpha(v_c - v_p) \tag{2.6}$$

  em que α é o parâmetro de reflexão (o valor usado usualmente para este parâmetro é $\alpha = 1$);

- O *Vértice Expandido*:

$$v_e = v_c + \beta(v_r - v_c) \tag{2.7}$$

em que β é o parâmetro de expansão (o valor usado usualmente para este parâmetro é $\beta = 2$);

- O *Vértice Contraído para o Exterior*:

$$v_{ce} = v_c + \gamma(v_r - v_c) \tag{2.8}$$

em que γ é o parâmetro de contracção (o valor usado usualmente para este parâmetro é $\gamma = \frac{1}{2}$);

- O *Vértice Contraído para o Interior*:

$$v_{ci} = v_c + \gamma(v_p - v_c) \tag{2.9}$$

Nota: Alguns autores definem estes vértices através de apenas uma expressão, equivalente a estas:

$$v = v_c + \delta(v_c - v_p), \tag{2.10}$$

onde o valor de δ indica o tipo de iteração. Desta forma, se:

- $\delta = 1$ tem-se uma reflexão;
- $0 < \delta < 1$ tem-se uma contracção (por vezes considera-se $-1 < \delta < 1, \delta \neq 0$, sendo que a valores de δ inferiores a zero corresponde a contracção para o interior e a valores de δ superiores a zero corresponde a contracção para o exterior);
- $\delta > 0$ tem-se a expansão;

A operação de *Redução do Simplex*, fixa apenas o melhor vértice v_m e substitui cada um dos outros vértices pelo ponto médio do segmento que o une a v_m:

$$v_i' = \frac{v_i + v_m}{2},\ i = 1, ..., n+1, i \neq m. \tag{2.11}$$

(ou, se $v_0 = v_m$, então $v_i' = v_0 + \gamma^s (v_i - v_0), i = 1, ..., n$, com γ^s o índice de redução do simplex, usualmente $\gamma^s = \frac{1}{2}$, segundo Conn et. al. em [32]).

Estes pontos auxiliares não são calculados à partida, mas de acordo com uma certa sequência, que depende dos correspondentes valores da função e da comparação com os valores da função nos vértices do simplex.

Após a inicialização, o método de Nelder-Mead começa por determinar o vértice reflectido. Se este novo ponto for indicador de uma boa progressão, o vértice expandido é calculado e comparado, a nível dos valores da função, com o reflectido com a consequente aceitação do vértice reflectido ou do expandido, como substituto do pior vértice do simplex.

Se, pelo contrário, a progressão não for satisfatória, os vértices contraídos para o exterior e interior são calculados e mais uma vez, comparados com os dois piores vértices do simplex. Se um dos vértices contraídos for melhor do que v_{sp}, ele é aceite e vai substituir o pior vértice, caso contrário efectua-se a redução do simplex.

Qualquer que seja o simplex final, os correspondentes valores de f são calculados procedendo-se novamente à identificação dos vértices v_m, v_p e v_{sp}, por forma a iniciar nova iteração.

Na TABELA 2.4 apresenta-se o algoritmo de de Nelder-Mead apresentado por Matias em [110].

Por todas estas características o algoritmo simplex de Nelder-Mead ganhou grande popularidade, sendo de todos os Métodos de Pesquisa Directa, o algoritmo simplex mais encontrado em *packages* de *software* numérico. Registe-se ainda que e o artigo original de Nelder e Mead é uma citação científica clássica, com centenas de referências na literatura científica em inúmeros jornais, tendo disso surgido inúmeras variantes deste método.

2.4.3 Convergência e Variantes

Em 1978, Wolfe em [158] introduziu alterações ao método original de Nelder-Mead, não adoptando a opção de contracção para o exterior, mas acrescenta um algoritmo de interpolação quadrática tendo como base o último simplex obtido.

Por sua vez em 1987, Dennis e Woods em [51] e em 1998, McKinnon em [113] apresentam uma variante que aceita o vértice expandido se o correspondente valor da função objectivo

TABELA 2.4: Algoritmo de Nelder-Mead

Algoritmo de Nelder-Mead Fonte: Matias em [110].
Dados: Ao iniciar o algoritmo supõem-se conhecidos a função objectivo f, a dimensão do problema n, o ponto de partida $x^{(0)}$ e os parâmetros α, β, γ e s, bem como o número máximo de iterações $nmax$.

1. Fazer $v_1 \leftarrow x^{(0)}$ e calcular o valor da função objectivo em v_1, $f_1 \leftarrow f(v_1)$ e Iniciar o índice, fazendo $k \leftarrow 1$;
2. Iniciar o simplex, calculando para $i = 2, ..., n+1$, $v_i \leftarrow v_1 + s\ e_{i-1}$, $f_i \leftarrow f(v_i)$;
3. Identificar os vértices com melhor, pior e segundo pior valores da função, ou seja, determinar m, p e sp.
4. Calcular o centróide, v_c, usando (2.5);
5. Calcular o vértice reflectido, v_r, usando (2.6) e o respectivo valor da função, $f_r \leftarrow f(v_r)$;
6. Se $f_r < f_{sp}$ então:
   (a) Se $f_r < f_m$, então:
      i. Calcular o vértice expandido utilizando (2.7) e o respectivo valor da função, $f_e \leftarrow f(v_e)$;
      ii. Se $f_e < f(r)$ então aceitar a expansão fazendo $v_p \leftarrow v_e$ e $f_p \leftarrow f_e$, avançando para o passo 8.
      iii. Senão aceitar a reflexão fazendo $v_p \leftarrow v_r$ e $f_p \leftarrow f_r$, avançando para o passo 8.
   (b) Senão aceitar a reflexão fazendo $v_p \leftarrow v_r$ e $f_p \leftarrow f_r$, avançando para o passo 8.
7. Senão:
   (a) Se $f_r < f_p$, então:
      i. Calcular o vértice contraído para o exterior usando (2.8) e fazer $f_{ce} \leftarrow f_{v_{ce}}$
      ii. Se $f_{ce} < f_p$, então aceitar a contracção para o exterior fazendo $v_p \leftarrow v_{ce}$ e $f_p \leftarrow f_{ce}$, avançando para o passo 8.
      iii. Senão fazer a Redução de todo o simplex, utilizando (2.11), calculando os respectivos valores da função, nos novos pontos, $f_i \leftarrow f_{v_i'}$ e avançando para o passo 8.
   (b) Senão calcular o vértice contraído para o interior, utilizando (2.9) e fazer $f_{ci} \leftarrow f(v_{ci})$
      i. Se $f_{ci} < f_p$, então aceitar a contracção para o interior fazendo $v_p \leftarrow v_{ci}$ e $f_p \leftarrow f_{ci}$, avançando para o passo 8.
      ii. Senão: fazer a redução de todo o simplex utilizando (2.11), calculando os respectivos valores da função, nos novos pontos, $f_i \leftarrow f_{v_i'}$ e avançando para o passo 8.
8. Identificar os vértices com o melhor, pior e segundo pior valores da função, ou seja, determinar os novos parâmetros m, p e sp.
9. Actualizar o processo iterativo com a melhor aproximação: $x^{(k)} \leftarrow v_m$ e o respectivo valor da função, $f(x^{(k)}) \leftarrow f_m$
10. Se o critério de paragem, T1 não for verificado (ver (1.7))
    (a) Então se $k \geq nmax$ (T2), parar; Senão, fazer $k \leftarrow k+1$ e voltar ao passo 4.
    (b) Senão parar, pois já se está perto do mínimo.
11. Concluir fazendo: $x^* \leftarrow x^{(k)}$ e $f^* \leftarrow f(x^{(k)})$

for inferior ao menor valor da função nos vértices, que corresponde ao melhor vértice, do simplex original.

McKinnon em [113] mostra, com vários exemplos, as limitações existentes em provar a convergência global do método de Nelder-Mead. Ele apresenta uma série de problemas com funções de duas variáveis, convexas e contínuas, com derivadas contínuas, para os quais o método converge para um ponto de não estacionaridade, a partir de um determinado simplex inicial. Nestes exemplos o método repete sucessivamente a contracção para o interior mantendo sempre o melhor vértice, ou seja, ocorre o que McKinnon chama repetição de contracção interna concentrada (do inglês *repeat focused inside contraction*).

A sucessão de simplexes assim obtida converge para uma linha que é ortogonal à direcção de descida máxima ($-\nabla f(x)$) e apresenta ângulos internos que tendem para zero.

No mesmo ano Lagarias et.al. em [93], apresentam propriedades de convergência do Método de Nelder-Mead para pequenas dimensões, nomeadamente para funções convexas de dimensão um. No que diz respeito a funções a duas dimensões provam alguns resultados, mas não foi possível contra argumentar sobre os exemplos de McKinnon. Neste trabalho apresentam uma descrição particularmente clara e cuidadosa das propriedades de convergência do Método Simplex de Nelder-Mead, esforçando-se por investigar o que pode ser provado sobre o comportamento assimptótico do método.

Na tentativa de resolver a situação referida por McKinnon, em 1999, Kelley em [88] propõe um novo passo para o método, designado por recomeço orientado (*oriented restart*), que reinicia o simplex com um simplex mais pequeno com arestas ortogonais, sempre que não ocorrer a condição de redução significativa, em relação à média dos valores da função nos vértices do simplex, de uma iteração para a seguinte, quando o processo estagna, testando-a com os problemas de McKinnon.

Os resultados mostram que em $\mathbb{R}$, o algoritmo é robusto; sobre condições padrão, a convergência para um ponto estacionário é garantida. Algumas propriedades gerais também podem ser provadas, mas nenhuma delas garante a convergência para problemas de grandes dimensões.

No que diz respeito à aceitação de vértices também existem diversas abordagens. Em algumas delas o vértice da contracção interna é aceite se for estritamente melhor do que o pior dos vértices do simplex (que é melhor do que o vértice reflectido). É a abordagem

de Kelley em [88] e [89], Lagarias em [93], McKinnon em [113], Wolfe em [158] e no artigo original de Nelder e Mead [119].

Noutras, onde ocorre mais frequentemente a redução do simplex só é aceite um vértice de contracção interna se ele for estritamente melhor do que o segundo pior vértice do simplex, como a apresentada por Dennis e Woods em [51] e Fernandes em [58] e a implementada por Matias [110].

A abordagem de Kelley em [88] e [89] e de Lagarias et. al. em [93] só aceita o vértice expandido se o respectivo valor da função for inferior ao valor da função no vértice reflectido, sendo, portanto, mais restritiva.

Lagarias et. al. em [93] introduzem uma notação que se torna relevante na análise de convergência. Assim, definem uma matriz M de ordem n, cuja coluna j representa a aresta do simplex S entre o vértice j, v_j, e o pior vértice, v_p:

$$M = \left[\begin{array}{cccc} v_1 - v_p & v_2 - v_p & \cdots & v_n - v_p \end{array} \right] \tag{2.12}$$

o volume do simplex S:

$$vol(S) = \frac{|det(M)|}{n!} \tag{2.13}$$

(que depende das coordenadas dos vértices e não da ordenação) e o diâmetro:

$$diam(S) = \max_{i \neq j} \|v_i - v_j\|_2 \tag{2.14}$$

Considerando esta notação Matias em [110] define simplex não degenerado, como se apresenta na definição que se segue.

Definição 2.8. Nas condições anteriores, um simplex S diz-se *não degenerado* se a matriz M for não singular, ou seja, se $vol(S) > 0$.

Ainda segundo Matias em [110], se o processo iterativo de Nelder-Mead for iniciado por um simplex não degenerado, todos os simplex subsequentes serão não degenerados. Outra propriedade também importante relaciona-se com o facto da versão do algoritmo

de Nelder-Mead, apresentado em [93], nunca gerar a redução total do simplex se a função for estritamente convexa e se o simplex inicial for não degenerado.

A análise de convergência dos Métodos de Pesquisa Directa impunha inicialmente que as arestas do simplex fossem linearmente independentes, em todas as iterações, e exigia a verificação de uma condição de redução mais forte do que a simples redução do valor da função objectivo. As diferentes variantes do algoritmo de Nelder-Mead, em geral, falham numa destas condições.

Mas, Lagarias et al. em [93] estabelecem condições para uma convergência global:

- Para uma função objectivo estritamente convexa de *apenas uma variável*, com conjuntos ou curvas de nível limitados:
  - Se:
    - Os parâmetros de reflexão, α, de expansão, β, e de contracção, γ, verificarem respectivamente: $\alpha > 0$; $\beta > 1$; $\beta > \alpha$; $\alpha\beta \geq 1$; $0 < \gamma < 1$.
    - O simplex inicial for não degenerado;
  - Então:
    - Algoritmo é convergente para um minimizante;
    - A razão de convergência a partir de uma certa iteração é linear de n em n passos, para o parâmetro de reflexão, $\alpha = 1$, isto é, a distância do melhor vértice ao óptimo diminui, de n em n passos, pelo menos de uma quantidade fixa menor do que um.
- Para uma função objectivo estritamente convexa de *duas variáveis*, com conjuntos de nível limitados:
  - Se:
    - Os parâmetros tiverem os valores: parâmetro de reflexão $\alpha = 1$, parâmetro de expansão $\beta = 2$ e parâmetro de contracção $\gamma = \frac{1}{2}$;
    - O simplex inicial for não degenerado;
  - Então a versão do algoritmo de Nelder-Mead gera simplexes cujos diâmetros tendem para zero (isto é, colapsam num ponto), o que não implica que convirjam para um ponto de acumulação, x^*.

Na sequência do método de Nelder-Mead, Tseng em [148] propõe um método de pesquisa, introduzindo uma condição de descida fortificada, mais forte do que a redução dos valores de f, e exigindo independência linear uniforme das arestas do simplex, para provar convergência para um ponto de estacionaridade de funções gerais de n variáveis. No referido trabalho, Tseng ainda inclui uma revisão geral de artigos da literatura inglesa, russa e chinesa sobre os algoritmos simplex.

O método de Tseng, designado em [148] por *Método de Pesquisa com Descida Fortificada*(do inglês *fortified-descent simplicial search*), segundo Matias em [110]:

- É flexível na escolha do número de vértices, que pode ir de 1 a n, que se mantém para a iteração seguinte;
- Usa um critério novo para a aceitação do simplex, baseado numa descida fortificada, que é análoga à redução significativa dos métodos baseados no gradiente;
- É flexível no número de cálculos adicionais de valores da função.

A análise de convergência do método de pesquisa com descida fortificada aponta no sentido de uma convergência para um ponto de estacionaridade de f.

Price et al. em [130] apresentam uma variante do algoritmo de Nelder-Mead, sem resultados de convergência, mas com bons resultados numéricos, que altera o simplex por forma a prevenir a sua degeneração. Esta abordagem baseia-se na tese de mestrado de Byatt [130] (SDNMA – *Sufficient Descent Nelder-Mead Algorithm* (Algoritmo de Nelder-Mead com Descida Suficiente)) que tem como motivação a classe de métodos descritos por Coope and Price em [33].

Nazareth et. al. em [118] propõem uma variante do algoritmo de Nelder-Mead provando ser convergente para o caso convexo. Para o caso de funções a várias variáveis esses resultados são conseguidos modificando a condição de descida estrita pela condição de descida fortificada de Tseng em [148].

As abordagens que requerem *descida suficiente* não se afastam muito do método de Nelder-Mead que também só exige uma descida simples entre iterações consecutivas.

Rykov, já em 1983, tinha publicado, em [133], uma classe de métodos, derivados do método de Nelder-Mead, convergentes, onde ideias como a pseudo-expansão de Coope, Price e Byatt aparecem pela primeira vez, de uma forma mais genérica.

Ultimamente as propriedades de convergência dos Métodos de Pesquisa Directa de Descida (*Descent Methods*) têm recebido muita atenção. A maioria desta atenção vai para a família de métodos que aceitam um ponto se o valor da função objectivo nesse ponto é menor do que melhor valor da função objectivo encontrado até então. O algoritmo original de Nelder-Mead e o método SDNMA concentram-se em melhorar o pior vértice do simplex.

Actualmente existem duas possibilidades para assegurar a convergência, do método de Nelder-Mead, para um mínimo local, da função objectivo, f:

- A primeira baseia-se numa restrição de descida suficiente imposta a valores de f em iterações consecutivas (por exemplo o método SDNMA). A grandeza da descida suficiente tende supostamente para zero, enquanto o método converge para um conjunto de pontos estacionários de f. É o caso da apresentada por Coope e Price em [33].
- A segunda exige simplesmente uma descida simples para aceitar um vértice, mas por outro lado todos os pontos têm que estar numa rede ou padrão de pontos (*Grid*), como a de Coope e Price em [34] e Burmen et. al. em [25]. A rede é então refinada gradualmente, no sentido de aumentar a precisão da pesquisa, enquanto o método se aproxima de um ponto estacionário de f.

A maioria dos especialistas da área consideram que neste momento não existe nenhuma variante do algoritmo de Nelder-Mead melhor do que a abordagem baseada numa rede (*grid-based approach*).

O trabalho de Burmen et. al. [25], apresenta um método deste tipo, possuindo uma performance similar ao método SDNMA, provando a convergência para funções objectivo de classe C^1. A abordagem de A. Burmen et al., em *Grid Restrained Nelder-Mead Algorithm* (GRNMA), não impõe uma condição descida suficiente (*sufficient descent*) em cada iteração, mas exige que os pontos estejam numa rede ou padrão.

Refira-se ainda o trabalho de Han et. al. em [78] onde os autores estudam o efeito da dimensão no desempenho do método de Nelder-Mead.

O método de Nelder-Mead, na sua forma original ou adaptado, continua a ser um dos mais usados na optimização não linear sem restrições, nomeadamente em campos como

a química, a engenharia e a medicina. São exemplos disso os trabalhos de Ghiasi et. al. em [72], Panigrahi et. al. em [125], Prsa et. al. em [131] e Walters em [157].

A razão para tal acontecer prende-se com o facto de quando o método funciona, pode funcionar de facto muito bem, encontrando frequentemente uma solução em muito menos avaliações da função objectivo do que os outros Métodos de Pesquisa Directa, contudo este método também pode falhar.

Assim, o método de Nelder-Mead pode falhar, devido à falta de garantia de descida suficiente ou convergir para um ponto não estacionário se se verifica uma deterioração da geometria do simplex.

Tendo em conta estes factos, Conn et. al. em [32] apresentam uma variante do método, globalmente convergente, baseada no controlo da geometria do simplex, através do seu diâmetro e volume normalizado e implementando um passo de salvaguarda para quando o simplex não verifica as condições de geometria estabelecidas. Este passo consiste em fazer uma rotação de 180^o, em torno do melhor vértice, dos restantes vértices do simplex.

Estes autores (também em [32]) definem o volume normalizado como sendo:

Definição 2.9. Considere-se o simplex S de vértices $(v_0, v_1, ..., v_n) = (v_m, ..., v_{sp}, v_p)$. O *volume normalizado* do simplex S é definido por:

$$\text{von}(S) = \text{vol}\left(\frac{1}{\text{diam}(S)}S\right) = \frac{|\det(M)|}{n!\text{diam}(S)^n} \tag{2.15}$$

com *vol* a representar o volume definido em (2.13), *diam* o diâmetro definido em (2.14) e M a matriz (2.12).

Na TABELA 2.5 apresenta-se o algoritmo de Nelder-Mead, globalmente convergente, apresentado por Conn et. al. em [32].

Este algoritmo é baseado, essencialmente, no trabalho de Tseng [148]. Neste algoritmo a geometria do simplex é controlada em todas as operações, através da análise do diâmetro e do volume normalizado do Simplex.

A operação de Redução do Simplex mantém o volume normalizado do Simplex, mas nas restantes operações não há garantia disto acontecer, pelo que é acrescentada, para cada

TABELA 2.5: Variante do Algoritmo de Nelder-Mead Globalmente Convergente

Variante do Algoritmo de Nelder-Mead Globalmente Convergente Fonte: Conn et. al. em [32].

Início:
Escolher $\xi > 0$ e um simplex inicial $S = \left\{v_0^{(0)}, v_1^{(0)}, ..., v_n^{(0)}\right\}$ tal que $von(S) \geq \xi$. Calcular o valor de f nos pontos do Simplex S.
Escolher os parâmetros de expansão, δ^e; de reflexão, δ^r; de contracção para o exterior, δ^{oc}; de contracção para o interior, δ^{ic}; o índice de redução do simplex, γ^s; e o parâmetro de controle do diâmetro do simlex δ^e, tais que: $0 < \gamma^s < 1 < \gamma^e, -1 < \delta^{ic} < 0 < \delta^{oc} < \delta^r < \delta^e$.
Para $k = 0, 1, 2, ...$

0. Fazer $S_k = S$.

1. **Ordenar:** Ordenar os $n+1$ vértices de $S = \left\{v_0^{(0)}, v_1^{(0)}, ..., v_n^{(0)}\right\}$, de modo que:

$$f_0 = f(v_0) \leq f_1 = f(v_1) \leq ... \leq f_n = f(v_n).$$

   Determinar $\Delta = diam(S)$.

2. **Reflexão:** Calcular o vértice reflectido isométrico ($\delta^r = 1$): $v_r = v_c + \delta^r(v_c - v_n)$.

   Se

$$\begin{array}{c} \text{diam}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_r\}\right) \leq \gamma^e \Delta \\ \text{von}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_r\}\right) \geq \xi \end{array} \tag{2.16}$$

   então calcular $f_r = f(v_r)$. Se $f_r \leq f_{n-1} - \rho(\Delta)$ [a], então fazer a expansão (e aceitar o melhor dos dois vértices, o reflectido ou o expandido). Senão fazer a contracção.

   Passo de salvaguarda: Se a reflexão isométrica não satisfaz (2.16) então rodar o simplex em torno do melhor vértice v_0:

$$v_{rot,i} = v_0 - (v_i - v_0), \quad i = 1, ..., n.$$

   Calcular $f(v_{rot,i}), \quad i = 1, ..., n$, e fazer $f_{rot} = \min\{f(v_{rot,i}) : i = 1, ..., n\}$.

   Se $f_{rot} \leq f_0 - \rho(\Delta)$, então terminar a iteração e considerar o simplex com rotação: $S_{k+1} = \{v_0, v_{rot,1}, ..., v_{rot,n}\}$. Senão fazer a contracção.

3. **Expansão:** Calcular o vértice expandido: $v_e = v_c + \delta^e(v_c - v_n)$.

   Se

$$\begin{array}{c} \text{diam}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_e\}\right) \leq \gamma^e \Delta \\ \text{von}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_e\}\right) \geq \xi \end{array} \tag{2.17}$$

   então calcular $f_e = f(v_e)$ e se $f_e \leq f_r$ substituir o vértice v_n pelo vértice expandido v_e e terminar a iteração: $S_{k+1} = \{v_0, v_1, ..., v_{n-1}, v_e\}$. Senão substituir v_n pelo vértice reflectido v_r e terminar a iteração: $S_{k+1} = \{v_0, v_1, ..., v_{n-1}, v_r\}$.

4. **Contracção:** Calcular um vértice contraído (pode ser para o interior ou exterior, $\delta = \delta^{oc}$ ou $\delta = \delta^{ic}$): $v_{cc} = v_c + \delta(v_c - v_n)$.

   Se

$$\begin{array}{c} \text{diam}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_{cc}\}\right) \leq \Delta \\ \text{von}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_{cc}\}\right) \geq \xi \end{array} \tag{2.18}$$

   então calcular $f_{cc} = f(v_{cc})$ e se $f_{cc} \leq f_n - \rho(\Delta)$ substituir o vértice v_n pelo vértice contraído v_{cc} e terminar a iteração: $S_{k+1} = \{v_0, v_1, ..., v_{n-1}, v_{cc}\}$. Senão implementar a redução do simplex.

5. **Redução do Simplex:** Fixar o melhor vértice e calcular os restantes n vértices do simplex reduzido: $v_s = v_0 + \gamma^s(v_i - v_0), \ i = 1, ..., n$. Calcular f nos n pontos do simplex reduzido. Escolher o melhor destes valores f_s.

   Se $f_s \leq f_0 - \rho(\Delta)$, então aceitar o simplex reduzido e terminar a iteração: $S_{k+1} = \{v_0 + \gamma^s(v_i - v_0), \ i = 0, 1, ..., n\}$. Senão voltar ao passo 0 com: $S_{k+1} = \{v_0 + \gamma^s(v_i - v_0), \ i = 0, 1, ..., n\}$.

---

[a] ρ é uma função continua e positiva, dita *forcing function*, tal que: $\rho :]0, +\infty[\to \mathbb{R}^+$; $\lim_{t \to 0^+} \frac{\rho(t)}{t} = 0$ e $\rho(t_1) \leq \rho(t_2)$ se $t_1 < t_2$. Um exemplo simples de uma função deste tipo é a função definida por $\rho(t) = t^2$.

uma delas uma condição do tipo $von(S_{k+1} \geq \xi)$ com $\xi > 0$, para garantir que o volume normalizado do Simplex não diminui drasticamente. De facto, com estas operações, o diâmetro do simplex (2.14) pode aumentar, o que implica uma redução do seu volume normalizado, segundo Conn et. al. em [32], ou seja, a geometria do simplex pode ser deteriorada.

No caso disto acontecer na operação de reflexão, é implementado o passo de salvaguarda, proposto por Dennis e Torczon em [52] no MDS (*Multidirectional Search Method*), uma vez que as reflexões são essenciais para assegurar a convergência global.

Outra questão importante, no que diz respeito à convergência, é o uso da condição $f(x_{k+1}) < f(v_m) - \rho(\Delta)$, com $\Delta = diam(S)$, ou seja, a imposição de descida suficiente do valor da função objectivo.

O passo de Redução do Simplex também é um pouco diferente do método original de Nelder-Mead, na medida em que, se não existir descida suficiente, este passo é aplicado repetidamente, havendo a possibilidade de ser implementado infinitamente. Neste caso prova-se que o algoritmo converge para um ponto estacionário, segundo Conn et. al. em [32] e Tseng em [148].

Mais recentemente têm surgido variantes do método de Nelder-Mead híbridas, usando-o em conjunto com abordagens estocásticas, por exemplo:

- Chelouah et. al. em [28] propõe um método híbrido que combina a Procura Tabu (*Tabu Search*) com o método de Nelder-Mead para optimização global de funções com muitos mínimos locais;
- Fan et al. em [57] integra o método de Nelder-Mead com um algoritmo genético (GA – *Genetic Algorithm*) e optimização por Enxame de Partículas (PSO – *Particle Swarm Optimization*);
- Yang et. al. em [161], para melhorar a performance de optimização do MBO (*Marriage in Honey Bees Optimization*), combina-o com o método de Nelder-Mead;
- Koscianski et. al. em [92] e Luersen et al. em [104] combinam o método de Nelder-Mead e o método de Powell, com uma inicialização do processo de optimização através de duas estratégias para globalizar pesquisa: a primeira é baseado na função

densidade de probabilidade, enquanto a segunda utiliza um algoritmo rápido para amostragem uniforme no espaço;

- Zahara et. al em [162] conjugam o método de Nelder-Mead com um algoritmo de Enxame de Partículas (*Particle Swarm*), para optimização com restrições.

2.5 Métodos com Conjuntos de Direcções de Pesquisa Adaptativas

Muitos autores, incluindo Lewis et. al. em [99], consideram os Métodos com Conjuntos de Direcções de Pesquisa Adaptativas (MASD) como Métodos de Optimização sem Derivadas, mas não os consideram como Métodos de Pesquisa Directa, uma vez que estes últimos usam somente e explicitamente valores da função objectivo. Esta classe inclui os métodos como o de Rosenbrock e Powell. Deste modo, neste trabalho só se faz uma apresentação genérica destes métodos, dando mais relevo às outras duas classes apresentadas anteriormente (PSM e SM). Os algoritmos MASD usam informação sobre a curvatura da função objectivo, obtida durante a pesquisa, para tentar acelerar o processo de procura do mínimo. Destes métodos, o primeiro foi criado por Rosenbrock [136], em 1960, para trabalhar com as características da função conhecida por "função banana" (*banana function*) de Rosenbrock. O método de Rosenbrock consiste em pesquisar n direcções ortogonais e linearmente independentes em cada iteração, usando na iteração seguinte informação adquirida sobre a função objectivo.

Powell em [129] apresentou uma variação do método de Rosenbrock, hoje conhecido por Método de Powell ou Método das direcções conjugadas de Powell. Este método tem como intuito encontrar o mínimo de uma função quadrática estritamente convexa, realizando no máximo n pesquisas unidimensionais sucessivas ao longo de direcções conjugadas. O algoritmo de Powell pode ser visto como uma versão do método dos gradientes conjugados, mas sem utilização de derivadas. Powell mostrou que se a função objectivo é quadrática convexa, então o conjunto das direcções adicionadas no último passo de cada etapa constitui um conjunto de direcções conjugadas (linearmente independentes).

Sucintamente a abordagem do método de Powell consiste em:

(1) Definir as direcções conjugadas;

(2) Realizar pesquisas unidimensionais para encontrar o minimizante exacto da interpolação quadrática calculada para cada direcção:

- As primeiras n pesquisas são sobre cada uma das direcções linearmente independentes, onde a última aproximação calculada é o ponto de partida para a pesquisa na direcção seguinte;

- A última pesquisa é sobre a direcção que liga o ponto obtido no fim das n primeiras pesquisas com o ponto inicial da etapa, chamado direcção resultante da progressão;

(3) No fim da etapa, uma das n direcções de pesquisa é substituída pela última direcção de pesquisa. O processo repete-se.

Capítulo 3

Optimização não Linear com Restrições

3.1 Introdução

A Optimização não Linear com Restrições não é tão simples como a Optimização sem Restrições, no entanto, em casos reais as variáveis estão muitas vezes sujeitas a restrições, seja por limitações de espaço, de dinheiro, de recursos ou pela natureza da realidade que a variável representa. Por exemplo, se uma variável representa o tempo decorrido ou a medida de um objecto, estas grandezas não podem ser negativas. Nestes casos o problema é um problema com restrições. Além disso, não é raro acontecer que a função objectivo ou as restrições sejam funções não lineares nas variáveis,vindo assim a originar um Problema de Optimização não Linear com Restrições.

Neste capítulo são apresentados métodos para resolver Problemas de Optimização com Restrições, do tipo apresentado em (1.2), que se reproduz a seguir, sem calcular ou estimar directamente as derivadas das funções envolvidas. Apesar de não se usarem derivadas nestes processos é usual assumir que as derivadas das funções que definem a região admissível Ω existem, por questões de prova de convergência.

Note-se que no problema (1.2) tanto a função objectivo como as restrições podem ser não lineares, ou seja o problema é um problema de Programação não Linear (NLP do inglês *NonLinear Programming*).

O que se pretende é optimizar a função f em Ω, mas por vezes tanto f como as restrições estão definidas fora de Ω, nesses casos é possível usar métodos que "funcionam" fora da região admissível, mas que se pretende que convirjam para um ponto admissível. No entanto, outras vezes, fora da região admissível nem sequer é possível calcular o valor de uma ou mais das funções envolvidas no problema. Assim, podem distinguir-se, segundo Conn et. al. em [31] e Audet et. al. em [12]:

Definição 3.1.

- *Restrições Relaxáveis*, flexíveis ou abertas (*Relaxable Constraints*) – são restrições que podem ser satisfeitas apenas aproximadamente ou assimptoticamente, sendo possível calcular o valor das funções envolvidas no problema fora da região admissível por elas definida, considerando uma certa tolerância.

- *Restrições não Relaxáveis*, inflexíveis ou fechadas (*Unrelaxable Constraints*) – são restrições que têm que ser satisfeitas em todas as iterações, uma vez que são restrições que não sendo satisfeitas inviabilizam o cálculo de uma ou várias das funções envolvidas no problema.

Os métodos usados para resolver problemas NLP não são simples, conforme se pode verificar nas aplicações reais apresentadas por Bartholomew-Biggs em [14] e Onwubolu e Babu em [124], além disso a informação existente não é totalmente clara relativamente ao método mais adequado em cada caso ou tipo de problema. Na secção 1.6.2 foram indicadas algumas das estratégias usadas para os resolver.

As técnicas mais usuais consistem em transformar os problemas com restrições em problemas sem restrições, de resolução mais fácil, cuja solução é igual ou se relaciona, de alguma forma, com a solução do problema original. Esta metodologia possibilita, assim, a utilização de outro tipo de abordagens ou algoritmos, nomeadamente a classe dos Métodos de Pesquisa Directa, tipicamente utilizados em problemas sem restrições.

Dependendo da abordagem, as restrições são tratadas como relaxáveis ou não relaxáveis. Alguns métodos, por exemplo, convertem as restrições de igualdade em duas restrições de desigualdade. Na prática, este procedimento pode não ser uma boa estratégia. Outros prosseguem apenas em pontos admissíveis. Estes métodos têm vantagens, porque se garante que o processo nunca pára num ponto não admissível, mas para tal é necessário que também o ponto inicial seja admissível. No entanto, em casos reais, nomeadamente

com regiões admissíveis muito pequenas, encontrar um ponto admissível para iniciar o processo pode ser bastante difícil. Além disso, em problemas onde os valores de f são calculados através dum *package* de simulação, apesar de não se saber à partida, pode não ser possível calculá-los para alguns parâmetros. Estas restrições, segundo Conn et. al. em [31], são designadas por *restrições escondidas* (*hidden constraints*). Neste caso a única alternativa é utilizar técnicas heurísticas ou usar a função barreira extrema, que é apresentada na secção 3.2.1.1.

Tendo em vista o objectivo principal deste trabalho, neste Capítulo faz-se uma breve revisão a estes métodos, nomeadamente aos Métodos de Penalidade e Barreira, de Lagrangeana Aumentada e Método dos Filtros, por serem os que permitem a resolução dos problemas sem utilização de derivadas.

3.2 Métodos de Penalidade e Barreira

As funções de Penalidade e Barreira, originalmente designadas por funções de Penalidade Exterior e Interior, antes ainda de serem estudadas matematicamente, foram amplamente utilizadas em problemas de engenharia, tendo, por esse motivo, adquirido outras denominações relacionadas com essas aplicações.

Nos últimos anos surgiu um grande interesse sobre este tema: Byrd et. al. em [27], Chen et al. em [29], Fletcher et. al. em [62], Gould et. al. em [75], Leyffer et. al. em [101], Klatte et. al. em [90], Mongeau et. al. em [117] e Zaslavski em [163], por causa da sua capacidade em lidar com problemas degenerados e em linearizar restrições. Os Métodos de Penalidade Exacta foram usados com sucesso para resolver problemas com restrições de Complementaridade (MPCCs – *Mathematical Programs with Complementary Constrains*) por Benson et. al. em [17], Leyffer et. al. em [101], Rodrigues e Monteiro em [134] e Rodrigues et. al. em [135] e também foram usados em programação não linear com restrições (CNLP – *Constrained NonLinear Programming*) para assegurar a admissibilidade de subproblemas e melhorar a robustez da iteração por Byrd et. al. em [27] e Chen et. al. em [29].

Considere-se o problema, P definido em (1.2). Os Métodos de Penalidade e Barreira foram criados para resolver o problema P, resolvendo uma sequência de problemas sem restrições especialmente escolhidos. Ou seja, é efectuada uma transformação ao problema

inicial originando o aparecimento e a resolução de uma sequência de outros problemas (*Processo externo*), sem restrições, derivados do inicial, pelos métodos conhecidos para este tipo de problemas (*Processo interno*). Assim, podem ser usados como o primeiro passo para a resolução de problemas de optimização, permitindo a resolução de problemas com restrições por métodos tipicamente utilizados em problemas sem restrições. Na FIGURA 3.1 apresenta-se esquematicamente o processo.

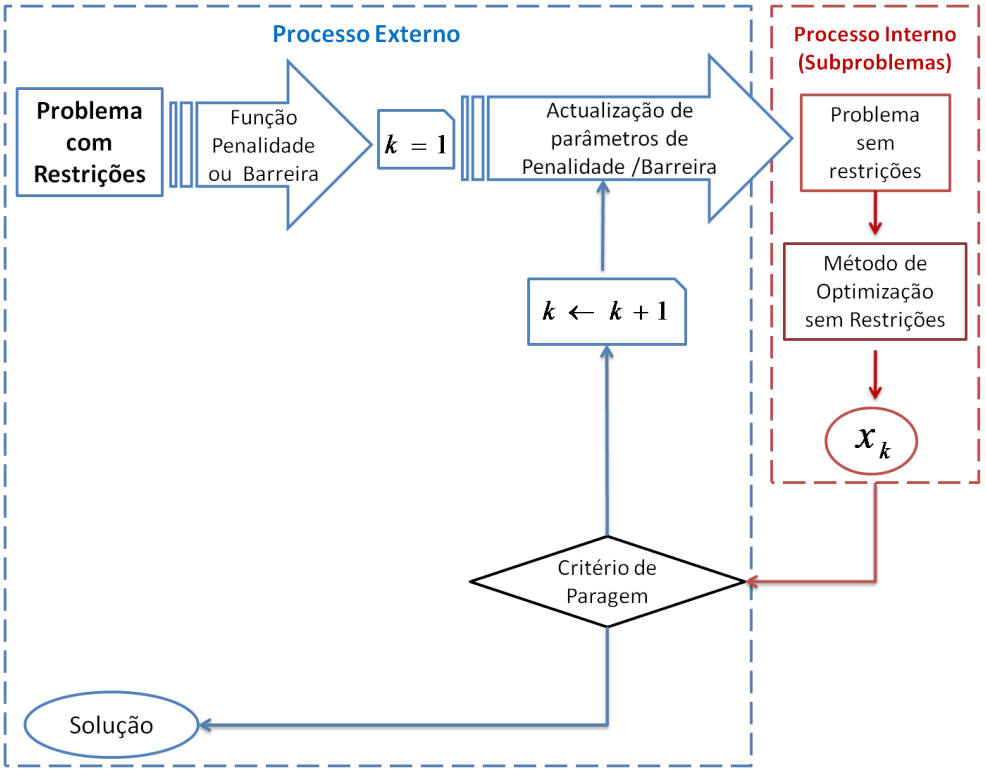

FIGURA 3.1: Métodos de Penalidade/Barreira – Processos iterativos interno e externo

Nestes métodos, é construída uma nova função objectivo, Φ, que contém informação relativa à função objectivo inicial, f, e simultaneamente às restrições do problema, logo, a admissibilidade e optimalidade são tratadas em conjunto. São desta forma sucessivamente construídos problemas sem restrições, que dependem de um parâmetro positivo, r_k, cujas correspondentes soluções $x^*(r_k)$ convergirão para a solução do problema inicial x^*.

A região admissível de P é o conjunto Ω definido pelas condições: $c_i(x) = 0,\ i \in \mathcal{E}$ e $c_i(x) \leq 0,\ i \in \mathcal{I}$.

Nos Métodos de Penalidade, a região admissível, Ω, é expandida a todo o $\mathbb{R}^n$, mas é aplicada uma grande penalização à função objectivo nos pontos que estão fora da região

admissível original, Ω.

Nos Métodos de Barreira, pressupõe-se que é conhecido um ponto inicial da região admissível, Ω, e impõe-se uma grande penalização nos pontos admissíveis que estão muito próximos da fronteira de Ω, recusando pontos não admissíveis ao longo das iterações. Desta forma cria-se uma barreira ao processo iterativo para este não sair da região admissível. Estes métodos, quando consideram parâmetros adequados, preservam iterações admissíveis.

Os Métodos de Penalidade e Barreira são, muitas vezes, designados apenas por Métodos de Penalidade uma vez que ambos se baseiam na mesma estratégia: constroem uma nova função objectivo, que inclui informação da função objectivo original e das restrições, e resolvem um ou uma sequência de novos problemas sem restrições.

Convertendo as restrições, $c_i(x) = 0$ em $c_i(x) \leq 0$ e $-c_i(x) \leq 0$, pode assumir-se o problema P (1.2) na forma:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & f(x) \\ \text{s. a.} & g_i(x) \leq 0,\ i = 1, 2, ..., m + t \end{array} \tag{3.1}$$

considerando que $g_i(x) \leq 0,\ i = 1, 2, ..., m + t$ representa o conjunto de todas as $m + t$ restrições do problema. Assim, o problema original, é agora um problema com restrições apenas de desigualdade da forma *menor ou igual*.

Então, pode construir-se nova função objectivo, Φ, que contém informação relativa à função objectivo inicial f e, simultaneamente, às restrições do problema:

$$\Phi(x, r) = f(x) + \Theta(x, r), \tag{3.2}$$

onde

- r é um parâmetro real positivo, dito *parâmetro de penalidade* ou *barreira*;
- A função $\Theta : \mathbb{R}^{n+1} \to \mathbb{R}^n$ é uma função dependente de r e da função de penalidade ou barreira, dependendo da forma como se inclui a informação relativa às restrições.

3.2.1 Métodos de Barreira

Os Métodos de Barreira são adequados para problemas onde é exigida uma aproximação à solução que seja admissível, mas também exigem que se forneça uma aproximação inicial admissível para iniciar o processo. Estes métodos foram amplamente estudados por Doyle em [54].

Tendo em conta a definição de Freund em [68] pode definir-se uma Função Barreira como sendo:

Definição 3.2. Uma *Função Barreira* para o problema (3.1) é qualquer função $b : \mathbb{R}^n \to \mathbb{R}$, que satisfaz as condições:

- $b(x) = \infty$, para todo o x que não satisfaz $g_i(x) < 0$ para algum $i \in 1, 2, ..., m + t$;
- $0 \leq b(x) \leq +\infty$, para todo o x que satisfaz $c_i(x) < 0,\ \forall i = 1, 2, ..., m + t$;
- $b(x) \to \infty$ quando $\max\limits_{i} \{g_i(x)\} \to 0$.

Esta definição impõe que o processo se desenvolva tendo em conta três regras:

- Qualquer ponto não admissível é rejeitado, uma vez que é atribuído à Função Barreira o valor infinito;
- Os pontos admissíveis devem ser aceites para comparação com os obtidos em iterações anteriores, uma vez que a Função Barreira assume um valor finito não negativo $b(x)$;
- Os pontos próximos da fronteira da região admissível devem assumir valores da Função Barreira muito grandes (que tendem para o infinito à medida que se aproximam dela).

A ideia principal do Método de Barreira é rejeitar pontos não admissíveis e dissuadir qualquer aproximação à fronteira da região admissível.

Assim, o problema a resolver em cada iteração k, que substitui o problema (3.1), é um problema, $B(r_k)$:

$$B(r_k) : \min_{x \in \mathbb{R}^n} f(x) + r_k b(x) \tag{3.3}$$

sendo $r = \{r_k\}_{k=1}^{+\infty}$ uma sucessão tal que $r_k \to +\infty$.

Pelo que, a nova função objectivo, Φ, que contém informação relativa à função objectivo inicial f e, simultaneamente, às restrições do problema, é $\Phi(x, r_k) = f(x) + r_k b(x)$.

O resultado seguinte, apresentado por Freund em [68], diz-se Teorema da Convergência dos Métodos de Barreira.

Teorema 3.3. *(Teorema da Convergência dos Métodos de Barreira)*

Seja $\{x_k\}_{k=1}^{+\infty}$ uma sucessão de soluções de $B(r_k)$ e $N(\epsilon, x)$ a vizinhança de raio $\epsilon > 0$, centrada no ponto $x \in \Omega$. Suponha-se, ainda, que a função objectivo f do problema (3.1), as restrições do problema $g_i(x)$, com $i = 1, 2, ..., m + t$ e a função barreira b, definida de acordo com a Definição 3.2, são funções contínuas e que existe uma solução óptima do problema (3.1) para $N(\epsilon, x) \cap \{x : g_i(x) < 0,\ i = 1, 2, ..., m + t\} \neq \{\}, \forall \epsilon > 0$.

Então qualquer ponto limite $\bar{x}$ de $\{x_k\}_{k=1}^{+\infty}$ é solução óptima do problema (3.1).

Nas secções seguintes apresentam-se algumas Funções Barreira, nomeadamente aquelas que se verificaram ser mais utilizadadas em conjugação com Métodos de Pesquisa Directa.

3.2.1.1 Função Barreira Extrema e Progressiva

Uma Função Barreira muito utilizada com Métodos de Pesquisa Directa, por exemplo por Audet et. al. em [7, 9–13], é a *Função Barreira Extrema* definida por:

$$\Phi(x) = b_\Omega(x) = \begin{cases} f(x) & \text{se} \quad x \in \Omega \\ +\infty & \text{se} \quad x \notin \Omega \end{cases} \tag{3.4}$$

Esta função funciona bem quando associada com os Métodos de Pesquisa Directa porque estes métodos usam o valor da função objectivo apenas em termos comparativos nos pontos em estudo. Se um ponto está fora da região admissível ou se aproxima da fronteira pelo exterior então $\Phi = +\infty$, pelo que o ponto é rejeitado. Já com métodos que constroem modelos da função este pode não ser o método mais adequado, pois é um método

inflexível que elimina pontos que poderão ser uma boa aproximação para a função objectivo propriamente dita e não o ser para o modelo que aproxima a função objectivo.

O princípio usado nesta função também é considerado nos Métodos de Penalidade de Recusa apresentados na secção 3.2.2.3.

A primeira versão do método MADS de Audet et. al. em [10], usa a função (3.4) para lidar com as restrições, no entanto, a exigência de uma aproximação inicial admissível para iniciar o processo motivou a versão seguinte deste método, apresentada pelos mesmos autores em [11]. Nesta segunda versão é usada uma função de medida de violação das restrições relaxáveis $b_X : X \subset \mathbb{R}^n \to \mathbb{R}_+$, é imposto um limite máximo para o valor dessa violação e os pontos que ultrapassem esse limite são rejeitados. Os autores designam esta abordagem por *Barreira Progressiva* e o método por MADS-PB (*Mesh Adaptive Direct Search – Progressive Barrier*). Este método tem a vantagem de permitir iniciar o processo com pontos que violem as restrições relaxáveis e admite tanto pontos teste como iterações com violações "*aceitáveis*" destas restrições.

Neste método os autores consideram o conjunto X definido pelas restrições não relaxáveis do problema. Se um ponto não está em X é rejeitado, se está em X então as restrições relaxáveis ($r_j(x) \leq 0$, com $j = 1, ..., p$ e onde p é o número de restrições relaxáveis) são tratadas usando a função $b_X : \mathbb{R}^n \to \mathbb{R}$:

$$b_X(x) = \begin{cases} \sum_{j=1}^{p} \left(\max\left(r_j(x), 0\right)\right)^2 & \text{se} \quad x \in X \\ +\infty & \text{se} \quad x \notin X \end{cases} \tag{3.5}$$

Assim, $b_X(x) = 0$ se e só se x verifica todas as restrições relaxáveis do problema e se x viola alguma das restrições relaxáveis do problema então $0 < b_X(x) < +\infty$.

Esta abordagem é chamada de barreira progressiva porque é fixado um valor máximo para a violação em cada iteração $h_{k_{max}}$, que é actualizado ao longo das iterações de acordo com uma relação de não dominância entre pontos testados, utilizada no Método dos Filtros (apresentado na Secção 3.4), considerando-se $h_{k_{max}} \to 0$ quando $k \to +\infty$.

Ambas as versões usam como direcções de pesquisa, no passo de sondagem, direcções geradas aleatoriamente, pelo que as soluções obtidas em duas execuções do método podem ser (e em geral são) diferentes. Além disso, pode acontecer que grande parte do espaço

de variáveis não seja explorado, mesmo no fim de várias iterações. Assim, Abramson et. al. em [2] usam um gerador *quasi-random* para gerar conjuntos de direcções ortogonais, tentando minimizar as direcções não exploradas. Este método é chamado pelos autores de OrthoMADS e é um método determinístico.

3.2.1.2 Método de Barreira Inversa

A Função Barreira Inversa, de potência p, é uma função barreira muito usada em problemas com restrições do tipo desigualdade, segundo Matias em [110].

Considerando o parâmetro de barreira $r_k \in \mathbb{R}^{m+t}$ podem construir-se, para o problema (3.1), os Problemas de Barreira para cada iteração k:

$$\min_{x \in \mathbb{R}^n} \Phi_k(x, r_k) \tag{3.6}$$

com

$$\Phi_k(x, r_k) = f(x) + r_k \sum_{i=1}^{m+t} \frac{1}{[g_i(x)]^p}. \tag{3.7}$$

A ideia subjacente a esta Função Barreira é criar uma barreira na fronteira da região admissível, no sentido de "motivar" iterandos admissíveis. O problema surge quando, no decorrer do processo iterativo, se obtém um iterando não admissível. Nesse caso, esta função dificulta a aproximação à região admissível. Além disso, se existir um iterando x tal que $g_i(x) = 0$ para algum $i = 1, ..., m + t$ a função não está definida para x.

3.2.1.3 Método de Barreira Logarítmica

A Função Barreira Logarítmica é usada para problemas com região admissível de interior não vazio e quando o problema original só possui restrições de desigualdade. Esta função deve ser usada com métodos que garantam que todas as iterações são admissíveis (métodos de pontos admissíveis ou métodos de pontos interiores), uma vez que a função não está definida fora da região admissível, segundo Matias em [110].

Considerando o parâmetro de barreira $r_k \in \mathbb{R}^{m+t}$ podem construir-se, para o problema (3.1), os Problemas de Barreira para cada iteração k:

$$\min_{x \in \mathbb{R}^n} \Phi_k(x, r_k) \tag{3.8}$$

com

$$\Phi_k(x, r_k) = f(x) + r_k \sum_{i=1}^{m+t} ln(-g_i(x)). \tag{3.9}$$

Note-se que os valores atribuídos ao parâmetro barreira r_k, nos subproblemas devem tender para infinito, quando k aumenta.

Esta Função tem o mesmo problema que a Função Barreira Inversa se existir um iterando x tal que $g_i(x) = 0$, para algum $i = 1, ..., m + t$, uma vez que nesse caso também não está definida para x.

3.2.2 Métodos de Penalidade

Esta secção, que se baseia essencialmente nos artigos de Correia et. al. [36, 40], pretende apresentar os Métodos de Penalidade. Estes métodos não exigem que, em cada iteração, a aproximação seja admissível, podendo ser iniciados com pontos não admissíveis, mas efectuando a procura de pontos onde a soma dos valores da função objectivo e da violação das restrições seja cada vez menor. Assim, o processo pode parar em pontos não admissíveis, o que pode não ser grave no caso de restrições relaxáveis. A violação das restrições é "medida" por uma função de penalidade que faz o balanço entre a minimização da função objectivo e a procura dum ponto admissível.

O papel das funções de penalidade é penalizar a violação das restrições do problema, permitindo que no processo iterativo possam ocorrer pontos não admissíveis, embora penalizados, em vez de criar uma barreira na fronteira da região admissível.

Tendo em conta a definição de Freund em [68] pode definir-se uma Função Penalidade como sendo:

Definição 3.4. Uma função $p : \mathbb{R}^n \to \mathbb{R}$ diz-se *função de penalidade* para o problema P, se verifica:

- $p(x) = 0$ se $c_i(x) \leq 0$;

- $p(x) > 0$ se $c_i(x) > 0$.

Assim, o problema a resolver em cada iteração, substituindo o problema P, é um problema, $P(r_k)$, com uma nova função objectivo, sem restrições:

$$P(r_k) : \min_{x \in \mathbb{R}^n} f(x) + r_k p(x) \tag{3.10}$$

sendo $r = \{r_k\}_{k=1}^{+\infty}$ uma sucessão tal que $r_k \to +\infty$.

Pelo que, a nova função objectivo, Φ, que contém informação relativa à função objectivo inicial f e, simultaneamente, às restrições do problema, é:

$$\Phi(x, r_k) = f(x) + \Theta(x, r_k), \tag{3.11}$$

com $\Theta(x, r_k) = r_k p(x)$.

A função Φ atribui, assim, uma penalização $\Theta(x, r_k)$ às restrições violadas em x, em que $r = \{r_k\}_{k=1}^{+\infty}$ é uma sucessão de escalares que se designam por *parâmetros de penalidade*. Como $r_k \to +\infty$, as violações são penalizadas de forma cada vez mais severa, para forçar a admissibilidade da solução.

O resultado seguinte, apresentado por Freund em [68], diz-se Teorema da Convergência dos Métodos de Penalidade.

Teorema 3.5. *Teorema da Convergência dos Métodos de Penalidade*

Sejam $r = \{r_k\}_{k=1}^{+\infty}$ a sucessão de parâmetros de penalidade tal que $r_{k+1} > r_k$, $\forall k$ e $\lim_{k \to +\infty} r_k = +\infty$; $\Phi(x, r_k) = f(x) + r_k p(x)$ a função objectivo e x_k^* a solução exacta do problema $P(r_k)$ e x^* a solução óptima de P. Suponha-se, ainda, que a função objectivo f do problema P, as restrições do problema $c_i(x)$, com $i = 1, 2, ..., m$ e a função Φ são funções contínuas e que $\{x_k^*\}_{k=1}^{+\infty}$ é uma sucessão de soluções de $P(r_k)$.

Então qualquer ponto limite $\bar{x}$ de $\{x_k^*\}_{k=1}^{+\infty}$ é solução óptima de P.

3.2.2.1 Funções de Penalidade Clássicas

As Funções de Penalidade clássicas incluem as funções do tipo:

$$p(x) = \sum_{i=1}^{m} [\max\{0, c_i(x)\}]^q = \sum_{i=1}^{m} ([c(x)]^+)^q, \tag{3.12}$$

com $q \geq 1$ e onde $[c(x)]^+ = \max\{0, c_i(x)\}, i = 1, 2, ..., m$.

- Se $q = 1$, em (3.12), a função $p(x)$ diz-se *função de penalidade linear* e o problema a resolver em cada iteração, que substitui o problema P, é um problema, $P(r_k)$, com uma nova função objectivo e sem restrições, do tipo:

$$P(r_k) : \min_{x \in \mathbb{R}^n} f(x) + r_k \sum_{i=1}^{m} [c(x)]^+ \tag{3.13}$$

  sendo $r = \{r_k\}_{k=1}^{+\infty}$ uma sucessão tal que $r_k \to +\infty$.

  Nota: Esta função pode não ser diferenciável nos pontos onde $c_i(x) = 0$, para algum i.

- A expressão (3.12) com $q = 2$ diz-se *função de penalidade quadrática* e o problema a resolver em cada iteração, que substitui o problema P, é um problema $P(r_k)$ com uma nova função objectivo, sem restrições, do tipo:

$$P(r_k) : \min_{x \in \mathbb{R}^n} f(x) + r_k \sum_{i=1}^{m} ([c(x)]^+)^2 \tag{3.14}$$

  sendo $r = \{r_k\}_{k=1}^{+\infty}$ uma sucessão tal que $r_k \to +\infty$.

Além destas funções mais básicas, existem outras funções de penalidade, consideradas clássicas, usadas para resolver o problema (3.1), nomeadamente, a Função de Penalidade Perda, a Função de Penalidade Exponencial, entre outras (apresentadas por Matias em [110] e Pereira em [127]).

3.2.2.2 Métodos de Penalidade Exacta

A característica principal dos Métodos de Penalidade Exacta é que, ao contrário dos Métodos de Penalidade Inexacta, apresentados anteriormente, que transformavam o problema original numa sequência infinita de problemas, agora a sequência de problemas é

finita, ou seja, o problema NLP é transformado num número finito de problemas sem restrições.

Segundo Freund em [68] e Zaslavski em [163] pode definir-se um Método de Penalidade Exacta da seguinte forma:

Definição 3.6. Um Método de Penalidade Exacta é um método que escolhe a função de penalidade $p(x)$ e o parâmetro r ($r = \{r_k\}_{k=1}^{+\infty}$) de tal forma a que a solução óptima $\bar{x}$ da sequência finita de problemas sem restrições, $P(r_k)$, com k finito, seja também a solução óptima, x^*, do correspondente problema com restrições, P.

Alguns métodos de penalidade, dependendo dos parâmetros utilizados, podem ser exactos ou inexactos, assim, na secção seguinte apresentam-se mais alguns Métodos/Funções de Penalidade, procedendo-se a uma classificação que tem em conta este conceito.

3.2.2.3 Classificação dos Métodos de Penalidade existentes

Nesta secção faz-se uma revisão e apresenta-se uma classificação para os Métodos de Penalidade existentes. São ainda apresentados alguns dos seus pressupostos e limitações. Na FIGURA 3.2 apresenta-se esquematicamente essa classificação.

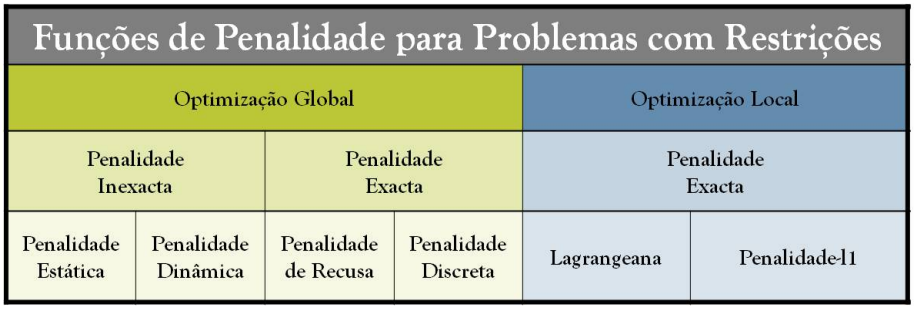

FIGURA 3.2: Uma Classificação dos Métodos de Penalidade existentes.

O objectivo dos Métodos de Penalidade é encontrar parâmetros de penalidade convenientes de forma a que a solução $\bar{x}$, que minimiza o problema sem restrições corresponda, de alguma forma, ao mínimo do respectivo problema com restrições, x^*.

Se o mínimo do problema sem restrições é admissível e correspondente ao mínimo global do problema com restrições (CGM, do inglês *Constrained Global Minimum*), ou seja, se é o ponto da região admissível com melhor valor da função objectivo, diz-se que o método

utilizado é um *Método de Optimização Global*. Se esse mínimo, correspondente a um mínimo local do problema com restrições (CLM, do inglês *Constrained Local Minimum*), ou seja, se é o ponto de uma vizinhança admissível pré-definida, com melhor valor da função objectivo, diz-se que o método utilizado é um *Método de Optimização Local*.

Assim, os Métodos de Penalidade podem ser classificados como:

- *Métodos de Penalidade de Optimização Global*, (GOPM, do inglês *Global Optimal Penalty Methods*), se permitem obter soluções globais, CGM, de P;
- *Métodos de Penalidade de Optimização Local*, (LOPM, do inglês *Local Optimal Penalty Methods*), se permitem obter soluções locais, CLM, de P.

Por outro lado, podem ser segundo Bertsekas em [19]:

- *Métodos de Penalidade Inexacta*, nos quais a minimização da nova função objectivo, Φ, não encontra os pontos CGM e CLM exactos, apenas caminha para pontos infinitamente próximos destes, pela minimização sucessiva dos problemas obtidos;
- *Métodos de Penalidade Exacta*, se permitem encontrar os CGM e CLM exactos, através de uma sequência finita de problemas de penalidade.

Métodos de Penalidade de Optimização Global – GOPM

Os GOPM podem ser Inexactos ou Exactos. Nesta classe de métodos estão Métodos de Penalidade Estática e Métodos de Penalidade Dinâmica.

Os *Métodos de Penalidade Estática* foram propostos inicialmente por Homaifar et al. em [82]. Nestes métodos é considerada uma família de níveis da violação para cada tipo de restrição. Cada nível da violação impõe um nível diferente da penalidade. A desvantagem deste método é o número de parâmetros a ser seleccionado. O número de parâmetros aumenta rapidamente com o número de restrições e de níveis da violação.

Nestes métodos é fixado um valor de penalidade, r para todo o processo. Existem duas dificuldades associadas a esta abordagem, segundo Deb em [50] e Godfrey et. al. em [73]:

- A solução da função objectivo depende do parâmetro de penalidade, r.

  Diversos autores procuraram encontrar por tentativa o melhor valor para r a usar nestes métodos para conduzir a procura dentro da região admissível. Alguns tentaram usar diferentes valores de r, dependendo do nível de violação das restrições, enquanto outros actualizaram o parâmetro de penalidade a cada iteração a partir de parâmetros que fixavam o raio de evolução (métodos de penalidade dinâmica);

- A inclusão do termo de penalidade distorce a função objectivo, segundo Deb em [50]. Para valores pequenos de r, a distorção é pequena, mas o óptimo de Φ pode não estar perto do verdadeiro óptimo. Por outro lado, se é usado um r grande, o óptimo de Φ é próximo do mínimo, mas a distorção pode ser tão grande que Φ pode ter mínimos fictícios.

Com os vectores de penalidade $\alpha \in \mathbb{R}^t$ e $\beta \in \mathbb{R}^{m-t}$ podem construir-se, para o problema (1.2), os Problemas de Penalidade para cada iteração k, com $\rho \geq 1$:

$$\min_{x \in \mathbb{R}^n} L_k(x, \alpha, \beta) \tag{3.15}$$

com

$$L_k(x, \alpha, \beta) = f(x) + \sum_{i=1}^{t} \alpha_i \left| c_i(x) \right|^\rho + \sum_{i=t+1}^{m} \beta_i [\max(0, c_i(x)]^\rho. \tag{3.16}$$

Este Método de Penalidade pode ser Exacto ou Inexacto. Se no Problema (3.15) se tiver $\rho = 1$ em (3.16), é um Método de Penalidade Exacto, se $\rho > 1$ é um Método de Penalidade Inexacto, Segundo Bertsekas em [19].

Isto é, quando $\rho = 1$ existem valores de penalidade α e β tais que o ponto que minimiza a função de penalidade é exactamente o CGM de P, (1.2).

Contudo, quando $\rho > 1$, o Método de Penalidade é um Método Inexacto e convergirá para o CGM como aproximação infinita dos valores de penalidade.

O Método de Penalidade Estática de Homaifar em [82] resolve um problema semelhante a (3.15), fixando os vectores de penalidade α e β. Este procedimento requer a escolha à partida de um número muito grande de parâmetros (um para cada restrição) e, além disso, é um Método de Penalidade Inexacto ($\rho > 1$).

Assim, a limitação comum de todos os métodos de penalidade estática é que geralmente é muito difícil escolher os valores apropriados de penalidade estaticamente. Além disso, estes métodos foram desenvolvidos para encontrar CGM e não permitem encontrar um CLM de P, (1.2). Portanto, são computacionalmente dispendiosos, porque envolvem a procura de um mínimo global de uma função de penalidade não linear.

Como alternativa à procura de parâmetros de penalidade por tentativa e erro existem os Métodos de Penalidade Dinâmica.

Os *Métodos de Penalidade Dinâmica*, por exemplo o de Wang et. al. em [159], incrementam as penalidades gradualmente, em (3.15) com L_k definida em (3.16), encontrando o mínimo global $\bar{x}$ de (3.15), para cada combinação de penalidades e param quando $\bar{x}$ é uma solução admissível de P, (1.2).

Tal como os Métodos de Penalidade Estáticos, os Métodos de Penalidade Dinâmica podem ser métodos exactos ou inexactos, dependendo do valor de ρ. Além disso, têm uma limitação comum com os anteriores, uma vez que também só permitem encontrar um mínimo global de funções não lineares.

Existem muitas variantes dos Métodos de Penalidade Dinâmica. Uma variante amplamente conhecida é o *Método de não estacionaridade*, que resolve uma sequência de problemas do tipo de (3.15) com L_k definida em (3.16) e $\rho > 1$, actualizando os parâmetros de penalidade em cada iteração k, segundo as condições que se seguem, com $C > 0$ um parâmetro constante:

$$\alpha_i(k+1) = \alpha_i(k) + C.\left|c_i(x)\right|, i = 1, ..., t \tag{3.17}$$

$$\beta_i(k+1) = \beta_i(k) + C.[\max(0, c_i(x))], i = t+1, ..., m \tag{3.18}$$

Outro Método de Penalidade Dinâmica é o *Método de Penalidade Adaptativa*, que faz uso do *feedback* do processo de pesquisa. Este método resolve o seguinte problema na iteração k:

$$\min_{x \in \mathbb{R}^n} L_k(x, \alpha, \beta) \tag{3.19}$$

com

$$L_k(x,\alpha,\beta) = f(x) + \sum_{i=1}^{t} \alpha_i(k)(c_i(x))^2 + \sum_{i=t+1}^{m} \beta_i(k)[\max(0, c_i(x))]^2 \tag{3.20}$$

onde os parâmetros de penalidade $\alpha_i(k), i = 1, ..., t$, são tais que:

- Se as restrições de igualdade são verificadas nas últimas l iterações os parâmetros são *decrementados*:

$$\alpha_i(k+1) \leftarrow \alpha_i(k)/\lambda_1 \text{ se } c_i(x) = 0 \text{ para } i \in \{k-l+1, ..., k\};$$

- Se as restrições de igualdade não são verificadas nas últimas l iterações os parâmetros são *incrementados*:

$$\alpha_i(k+1) \leftarrow \lambda_2.\alpha_i(k) \text{ se } c_i(x) \neq 0 \text{ para } i \in \{k-l+1, ..., k\};$$

- Caso contrário os parâmetros *mantêm-se*: $\alpha_i(k+1) \leftarrow \alpha_i(k)$.

onde l é um número inteiro positivo, $\lambda_1, \lambda_2 > 1$ e $\lambda_1 \neq \lambda_2$ (para evitar ciclos nas actualizações). Os parâmetros $\beta_i(k), i = t+1, ..., m$ são actualizados de forma similar.

Existem outras duas variações de GOPM, ambas abordagens exactas, são os Métodos de Penalidade de Recusa e os Métodos de Penalidade Discreta.

Os *Métodos de Penalidade de Recusa* (*Death penalty Methods*), usado na função de barreira extrema apresentada na equação (3.4), são métodos de penalidade que simplesmente rejeitam todos os pontos não admissíveis. Começam com um ou mais pontos admissíveis e procuram novos pontos, se o novo ponto não for admissível é rejeitado. Pode dizer-se que são Métodos de Barreira. São apresentados aqui, no sentido de os classificar. Como exemplos destes métodos podem referir-se os apresentados por Hu et. al. em [84] e Zhang et. al. em [164].

Nesta categoria, a dificuldade reside em gerar os pontos admissíveis iniciais e posteriormente em determinar computacionalmente novos pontos admissíveis, nomeadamente quando a região admissível é muito pequena.

Assim, este método resolve o Problema (1.2) considerando, em cada iteração k:

$$\min_{x\in\mathbb{R}^n} L_k(x,\alpha,\beta) \tag{3.21}$$

com

$$L_k(x,\alpha,\beta) = f(x) + \begin{bmatrix} \alpha_1 \\ \dots \\ \alpha_t \end{bmatrix}^T \cdot P \begin{bmatrix} c_1(x) \\ \dots \\ c_t(x) \end{bmatrix} + \begin{bmatrix} \beta_{t+1} \\ \dots \\ \beta_m \end{bmatrix}^T \cdot Q \begin{bmatrix} c_{t+1}(x) \\ \dots \\ c_m(x) \end{bmatrix},$$

onde P e Q são funções barreira extrema, isto é:

$$P \begin{bmatrix} c_1(x) \\ \dots \\ c_t(x) \end{bmatrix} = \begin{bmatrix} P_1 \\ \dots \\ P_t \end{bmatrix}$$

onde

$$P_i = \begin{cases} 0 & \text{se} \quad c_i(x) = 0 \\ +\infty & \text{se} \quad c_i(x) \neq 0 \end{cases}, i = 1, ..., t$$

e

$$Q \begin{bmatrix} c_{t+1}(x) \\ \dots \\ c_m(x) \end{bmatrix} = \begin{bmatrix} Q_{t+1} \\ \dots \\ Q_m \end{bmatrix}$$

onde

$$Q_i = \begin{cases} 0 & \text{se} \quad c_i(x) \leq 0 \\ +\infty & \text{se} \quad c_i(x) > 0 \end{cases}, i = t+1, ..., m$$

Este método é de Penalidade Exacta, uma vez que, para quaisquer valores finitos dos parâmetros de penalidade α_i e β_i, o ponto mínimo da função penalidade será admissível e terá o menor valor da função objectivo, sendo dessa forma exactamente o CGM de P.

Outro Método de Penalidade Exacta é o *Método de Penalidade Discreta* que usa o número de restrições que foram violadas em vez do grau de violações na função penalidade. Este

tipo de métodos é usada muitas vezes em Métodos de Elementos Finitos, tal como fizeram Dai et. al. em [48].

Conclui-se assim, que os Métodos de Optimização Global de (1.2) têm aplicação prática limitada, porque a procura do mínimo global é computacionalmente dispendiosa e técnicas de optimização global, como o Método de não estacionaridade, também são lentas pois só alcançam óptimos globais com convergência assimptótica.

Métodos de Penalidade de Optimização Local – LOPM

Para evitar a dispendiosa optimização global, têm sido desenvolvidos métodos de optimização local. Estes incluem, por exemplo, os Métodos dos Multiplicadores de Lagrange e os Métodos de Penalidade ℓ_1, que são ambos Métodos de Penalidade Exacta. Estes métodos foram criados para resolver problemas de optimização não linear contínuos, isto é, problemas do tipo (1.2), onde f é contínua e diferenciável e $c_i, i = 1, 2, ..., m$ podem ser descontínuas, não-diferenciáveis. Nestes métodos o objectivo é encontrar um mínimo local.

Os *Métodos dos Multiplicadores de Lagrange* tradicionais funcionam apenas para problemas (1.2) onde as funções envolvidas são contínuas e diferenciáveis.

A Função Lagrangeana do problema (1.2) com multiplicadores de Lagrange $\lambda \in \mathbb{R}^t$ e $\mu \in \mathbb{R}^{m-t}$ é definida por $L(x, \lambda, \mu) = f(x) + \lambda^T d(x) + \mu^T g(x)$, onde:

$$d(x) = \begin{bmatrix} c_1(x) \\ \cdots \\ c_t(x) \end{bmatrix} \text{ e } g(x) = \begin{bmatrix} c_{t+1}(x) \\ \cdots \\ c_m(x) \end{bmatrix}.$$

Assumindo a continuidade e diferenciabilidade, o CLM satisfaz a condição necessária de Karush-Kunh-Tucker(KKT) e a condição suficiente de ponto sela, segundo Bertsekas em [19]. Esta abordagem é, portanto, limitada à resolução de CNLPs com funções contínuas e diferenciáveis e não pode ser aplicada a problemas discretos ou a problemas onde não se conheçam as derivadas das funções. Esta limitação deve-se ao facto da existência dos multiplicadores de Lagrange depender da existência dos gradientes das restrições e da função objectivo e da regularidade das restrições (independência linear dos gradientes das restrições) nos pontos solução.

Além disso, é difícil encontrar pontos e multiplicadores que satisfaçam a condição suficiente de ponto sela, porque esta é expressa como um sistema de desigualdades não lineares, que em geral não é fácil de resolver. Assim, esta condição é usada simplesmente para verificar se as soluções encontradas pela condição KKT são, efectivamente, soluções óptimas.

O *Método de Penalidade* ℓ_1, foi proposto inicialmente por Pietrzykowski em [128], mas tem sido estudada e usada por muitos autores, por exemplo, Gould et. al. em [75] e Byrd et. al. em [27], além disso, serviu de base a muitos métodos de penalidade propostos na literatura.

Este método é um Método de Penalidade Exacta de minimização local e que permite, portanto, resolver CNLPs. Este método resolve, em cada iteração k, o problema:

$$\min_{x\in\mathbb{R}^n} \ell_1^{(k)}(x,\mu_k) \tag{3.22}$$

com

$$\ell_1^{(k)}(x,\mu_k) = f(x) + \mu_k \sum_{i=1}^{t} |c_i(x)| + \mu_k \sum_{i=t+1}^{m} \max[c_i(x),0],\ \mu_k \to +\infty. \tag{3.23}$$

Segundo Bertsekas em [19], a teoria desenvolvida à volta desta abordagem mostra que existe uma correspondência bijectiva entre o CLMs e o mínimo global da função ℓ_1 (3.23), quando μ_k é suficientemente grande.

Para valores apropriados do parâmetro de penalidade μ_k, os pontos estacionários de $\ell_1(x,\mu_k)$ são também pontos KKT do problema não linear (1.2) ou pontos estacionários não admissíveis, segundo Byrd et. al. em [26]. Esta propriedade é a mais apelativa destes Métodos de Penalidade Exacta, porque uma escolha adequada de μ_k serve todo o processo de minimização.

Este, como os outros Métodos de Penalidade Exacta, são, assim, menos dependentes do parâmetro de penalidade do que os métodos de penalização quadrática para os quais é necessário resolver uma sucessão de subproblemas com séries divergentes de parâmetros de penalidade.

Na TABELA 3.1 apresenta-se Algoritmo do Método de Penalidade ℓ_1 Clássico exposto por Byrd et al. em [27].

TABELA 3.1: Algoritmo do Método de Penalidade ℓ_1 Clássico

Algoritmo do Método de Penalidade ℓ_1 Clássico Fonte: Byrd et. al. em [27].
Dados: Conhecidos o ponto inicial x_0^s, o parâmetro de penalidade inicial $\mu_0 > 0$, o número máximo de iterações k_{max} e a tolerância $\tau > 0$.
Para $k = 0, 1, 2, ..., k_{max}$:

1. Determinar um minimizante aproximado x_k de $\ell_1(x, \mu_k)$, começando em x_k^s;

   - Se $\sum_{j=1}^{t} |c_j(x_k)| + \sum_{i=t+1}^{m} max[c_i(x_k), 0] \leq \tau$, considerar a solução aproximada x_k e ir para o passo 2;
   - Senão:
     - Escolher um novo parâmetro de penalidade $\mu_{k+1} > \mu_k$;
     - Actualizar a função de penalidade;
     - Fazer $k \leftarrow k + 1$, $x_k^s \leftarrow x_{k+1}^s$ e, enquanto não se atingir o número máximo de iterações, voltar a 1.

2. Concluir fazendo $x^* \leftarrow x_k$, $f^* \leftarrow f_k$

A minimização da função de penalidade ℓ_1 é difícil, porque é não diferenciável, pelo que não se podem aplicar algoritmos de optimização suave a esta função, além disso é difícil encontrar um valor/expressão para μ_k que garanta que o desempenho deste método seja consistente para todos os subproblemas.

O facto de ℓ_1 ser não diferenciável não é preocupante, no âmbito deste trabalho, uma vez que os Métodos de Optimização sem Restrições estudados não usam informação relativa às derivadas da função, no entanto as provas de convergência do método não estão acessíveis.

As alternativas a estes métodos têm vindo a aparecer, nomeadamente no que diz respeito à procura de soluções para o problema da escolha de parâmetros de penalidade.

Uma nova classe de Métodos de Penalidade Dinâmica

Segundo Byrd et al. em [27], os métodos de penalidade cresceram em três etapas de desenvolvimento desde que foram introduzidas em 1950. Primeiro foram vistos como meio de resolução de problemas de optimização com restrições através de técnicas de optimização sem restrições. Esta abordagem não provou ser efectiva, excepto para classes especiais de aplicações.

Na segunda etapa, os problemas de penalidade foram substituídos por uma sucessão de subproblemas com restrições lineares. Esta abordagem, usada na programação quadrática sequencial, é muito mais efectiva do que a anterior, mas deixa em aberto a questão de como escolher a sucessão de parâmetros de penalidade. Esta dificuldade, da escolha dos parâmetros, teve como consequência um decréscimo do interesse dos investigadores nos métodos de penalidade, durante os anos 90, levando ao aparecimento de outros métodos, como o Método dos Filtros, que não exige a escolha de parâmetros de penalidade.

Na etapa mais recente do desenvolvimento, os métodos de penalidade ajustam o parâmetro de penalidade em cada iteração, com o intuito de atingir um nível estipulado de admissibilidade linear. A escolha do parâmetro de penalidade deixa de ser heurística e torna-se parte integrante do passo de cálculo. Um exemplo deste tipo de abordagem é a apresentada por Byrd et. al. em [27], inspirada por um trabalho anterior dos mesmos autores, no contexto da progressão linear quadrática sequencial (SLQP). Neste artigo é feita uma análise e generalização desta estratégia para a sua aplicação em outros métodos.

A estratégia consiste em ajustar o parâmetro de penalidade dinamicamente, promovendo um progresso balanceado de optimalidade e admissibilidade através do controlo do grau de linearidade admissível considerado em cada iteração.

Em contraste com as aproximações clássicas, neste método a escolha do parâmetro de penalidade deixa de ser heurística e passa a ser feita através da resolução de um subproblema adicional em cada iteração. A nova estratégia de actualização da penalidade é apresentada no contexto de métodos de programação quadrática sequencial (SQP) e programação linear quadrática sequencial (SLQP), métodos que usam regiões admissíveis para promover a convergência.

A abordagem apresentada em [27] por Byrd et. al., consiste em fazer uma reformulação do problema, linearizando as restrições e exigindo que em cada passo haja progresso na admissibilidade linear (*Linear Feasibility*) que deve ser proporcional ao possível progresso para o óptimo. A nova estratégia incrementa automaticamente o parâmetro de penalidade, anulando a dificuldade de escolher valores apropriados para os parâmetros de penalidade.

Outras estratégias de actualização de parâmetros de penalidade foram propostas recentemente. Chen et. al. em [29], propuseram regras que actualizam o parâmetro de

penalidade, baseadas na admissibilidade e no tamanho dos multiplicadores. No mesmo ano Leyffer et. al. em [101], consideram métodos de penalidade descrevendo critérios dinâmicos para actualizar parâmetros de penalidade, baseados no decréscimo médio das restrições penalizadas.

3.3 Lagrangeana Aumentada

Os algoritmos ou *Métodos de Lagrangeana Aumentada* (*Augmented Lagrangian Methods*-ALM) foram apresentados inicialmente por Hestenes em [81] com o objectivo de resolver problemas com restrições de igualdade e baseiam-se na minimização sucessiva de uma Função Lagrangeana Aumentada. Posteriormente foram apresentadas diversas variantes deste método.

Conn et. al. em [31], por exemplo, considerando um problema só com restrições de igualdade e de limites simples, do tipo:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ s.\,a. \quad & c_i(x) = 0,\ i \in \mathcal{E}' \\ & l \leq x \leq u \end{aligned}$$

apresentam a seguinte Função Lagrangeana Aumentada para a sua resolução:

$$L(x,\lambda,S,\mu) = f(x) + \sum_{i\in\mathcal{E}'} \lambda_i c_i(x) + \frac{1}{2\mu}\sum_{i\in\mathcal{E}'} s_{ii} c_i^2(x) \tag{3.24}$$

podendo actualizar-se λ e s em cada iteração, onde λ é o vector dos multiplicadores de Lagrange estimados para as restrições de igualdade, μ é o parâmetro de penalidade e s_{ii} são elementos positivos da diagonal duma matriz S, designados pelos autores por *shifts*. Estes autores, usam depois um Método de Pesquisa em Padrão Generalizado para o cálculo da iteração seguinte, x_{k+1}, resolvendo o subproblema:

$$\min_{x\in\mathbb{R}^n} \ L_k(x_k,\lambda_k,\mu_k,s_k) \tag{3.25a}$$

$$s.t.\ l \leq x \leq u \tag{3.25b}$$

Nesta formulação as restrições de desigualdade são convertidas em restrições de igualdade através da introdução de variáveis de folga não negativas, que passam a fazer parte das restrições de limites simples do problema, (3.25b). O conjunto de índices $\mathcal{E}'$ em (3.24) inclui, assim, os índices correspondentes às restrições não só de igualdade, mas também às de desigualdade, convertidas em restrições de igualdade usando este processo. Esta estratégia tem o inconveniente de aumentar a dimensão do problema, pela introdução de variáveis de folga e de mais restrições de limites simples.

Esta abordagem é relativamente fácil de implementar, porque minimiza-se em cada iteração a função Lagrangeana Aumentada que é uma função suave, com restrições apenas de limites simples.

Conn et. al. em [31] propõe, ainda, a actualização do vector dos multiplicadores em (3.24) através da fórmula:

$$\bar{\lambda}(x, \lambda, S, \mu) = \lambda + \frac{S \cdot c\,(x)}{\mu}, \tag{3.26}$$

e usam como medida para o critério de paragem:

$$\theta(\lambda, \mu) = \left(1 + \|\lambda\| + \frac{1}{\mu}\right)^{-1}, \tag{3.27}$$

que é uma função que tende para zero quando $\left(1 + \|\lambda\| + \frac{1}{\mu}\right) \to \infty$.

A estratégia de Lagrangeana Aumentada, motivou o desenvolvimento de diversas abordagens, nomeadamente:

- Para programação convexa, conforme é apresentado por Bertsekas em [18];
- Para problemas não convexas com restrições, foi estudada por Andreani et. al. em [4, 5], Birgin et. al. em [21], Conn et. al. em [31] e Lewis et. al. em [98, 100];
- A convergência foi amplamente estudada por Luo et. al. em [105, 106] e mais recentemente por Luo et. al. em [107] e Wu et. al. em [160].

As técnicas usadas na abordagem de Conn et. al. em [30], para optimizar sucessivamente a Função Lagrangeana Aumentada, eram técnicas baseadas em derivadas. Lewis

e Torczon em [100], partindo desta abordagem consideram Métodos de Lagrangeana Aumentada, conjugados com Métodos de Pesquisa em Padrão com limites simples, para a resolução de problemas não lineares com restrições de igualdade e limites simples, sem usar, portanto, derivadas. Neste trabalho apresentam, na secção final, algumas considerações para o problema geral. Tendo em conta este trabalho, Espírito Santo em [55], apresenta, para o problema

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c_i(x) = 0, i \in \mathcal{E} \\ & c_i(x) \leq 0, i \in \mathcal{I} \\ & l \leq x \leq u \end{aligned} \tag{3.28}$$

com $\mathcal{E} = \{1, 2, ..., t\}$ e $\mathcal{I} = \{t+1, t+2, ..., m\}$, a Função Lagrangeana Aumentada:

$$L(x, \lambda, \delta, \mu) = f(x) + \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \frac{1}{2\mu} \sum_{i\in\mathcal{E}} c_i^2(x) + \frac{\mu}{2} \sum_{i\in\mathcal{I}} \left(\max\left(0, \delta_i + \frac{c_i(x)}{\mu}\right)^2 - \delta_i^2 \right) \tag{3.29}$$

onde $\lambda \in \mathbb{R}^t$ é o vector de dos multiplicadores de Lagrange para as restrições de igualdade, $\delta \in \mathbb{R}^{m-t}$ é o vector dos multiplicadores de Lagrange para as restrições de desigualdade e μ é o parâmetro de penalidade positivo. Assim, a iteração seguinte, x_{k+1}, é calculada resolvendo o subproblema:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & L(x, \lambda_k, \delta_k, \mu_k) \\ s.t. \quad & l \leq x \leq u \end{aligned} \tag{3.30}$$

através do método de Pesquisa Padrão Generalizado.

Os autores em [100] propõem estratégias de actualização dos parâmetros, mostrando resultados de convergência para este tipo de problema. A actualização do vector dos multiplicadores associados às restrições de igualdade é feita através da fórmula (3.26), proposta por Conn et. al. em [31].

A actualização do vector dos multiplicadores associados às restrições de desigualdade é feita através da fórmula:

$$\bar{\delta}(x,\delta,\mu,S) = \max\left(0,\ \delta + \frac{S\cdot c_i\left(x\right)}{\mu}\right), \tag{3.31}$$

sendo $\bar{\delta}$ o vector dos multiplicadores da iteração $k+1$ e δ o vector dos multiplicadores da iteração k.

Esta estratégia também foi usada por Espírito Santo em [55], com as devidas adaptações para o critério de paragem apresentado em (3.27), proposta por Conn et. al. em [31]:

$$\theta(\lambda,\mu) = \left(1 + \|\lambda\| + \|\delta\| + \frac{1}{\mu}\right)^{-1}, \tag{3.32}$$

que é uma função que tende para zero quando $\left(1 + \|\lambda\| + \|\delta\| + \frac{1}{\mu}\right) \to \infty$.

3.4 Método dos Filtros

Para resolver um problema de optimização não linear com restrições (NLP), (1.2), deve ter-se em conta que se pretende minimizar a função objectivo e a violação das restrições, que deverá ser zero ou, pelo menos, tender para zero. Assim, esta metodologia envolve dois conceitos: a *optimalidade* (em que a finalidade é minimizar a função objectivo f) e a *viabilidade* ou *admissibilidade* (que tem como propósito minimizar os valores das violações das restrições $c_i,\ i = 1,2,...,t+m$).

A forma mais usual de resolver os problemas NLP consiste em os transformar num problema sem restrições, cuja solução é igual ou se relaciona de alguma forma com a solução do problema NLP, como nos métodos apresentados anteriormente. Nestes métodos a optimalidade e a viabilidade são tratadas em conjunto, ou seja, é construída uma nova função objectivo (ou uma sucessão delas) que envolve f e as restrições c_i. O problema sem restrições é, então, resolvido usando métodos de optimização sem restrições.

Outra alternativa para a resolução do problema NLP (1.2) é o Método dos Filtros. Ao contrário dos métodos anteriores, o Método dos Filtros utiliza o conceito de dominância

da optimização multi-objectivo, considerando a optimalidade e a viabilidade separadamente. Assim, o problema NLP é transformado num problema bi-objectivo, com dois problemas de minimização sem restrições. Em cada iteração há, assim e por esta ordem, duas fases: uma fase de viabilidade e uma fase de optimalidade.

O Método dos Filtros foi introduzido por Fletcher e Leyffer em [63], para globalizar métodos SQP e com a motivação de evitar a dificuldade da estimação de parâmetros de penalidade e de multiplicadores de Lagrange, e, a partir daí, tem sido uma das técnicas mais usadas nas mais diversas áreas da optimização, com restrições, linear e não linear.

Uma revisão sobre o Método dos Filtros foi apresentada por Fletcher et al. em [66]. Neste trabalho eles expõem as ideias principais dos Métodos dos Filtros e os principais resultados de convergência e ainda indicam áreas onde estes métodos foram usados com sucesso, até então. Os mesmos autores, já em [63], tinham apresentado resultados sobre a convergência global em problemas SLP (Sequential Linear Programming) e em [65], sobre a convergência global em problemas SQP (Sequential Quadratic Programming) e de região admissível. Ribeiro et. al. em [132] apresentam resultados de convergência mais gerais no âmbito da Programação não linear. Sainvitu em [137] estuda-os no contexto de Métodos de Regiões de Confiança ou Regiões Admissíveis e Silva e Monteiro em [139, 140] e Wachter et. al. em [154–156] no contexto de Métodos de Procura Linear (*Line Search*).

Esta técnica também foi aplicada, entre outros:

1. Em Pesquisa Directa por Audet e Dennis em [8], onde usam o Método dos Filtros com Métodos de Pesquisa em Padrão e por Correia et. al. em [36], onde se apresenta um novo método de Pesquisa Directa para Optimização não Linear com Restrições, que combina o Método Simplex e o Método dos Filtros. Este método, à semelhança do método de Audet et. al. em [8], não necessita de calcular ou aproximar qualquer derivada, parâmetros de penalidade ou multiplicadores de Lagrange;

2. Num contexto de optimização não contínua e não diferenciável por Fletcher et. al. em [64], por Gonzaga et. al. em [74] e por Karas et. al. em [86];

3. Gould et al. em [76] apresentaram um filtro multidimensional e em [77] introduziram uma extensão desse filtro.

4. A SQP por Antunes e Monteiro em [6] e por Nie et. al. em [121];

5. Em algoritmos de pontos interiores por Ulbrich et al. em [149];

6. Num contexto de optimização com variáveis mistas (*Mixed Variable Constrained Optimization Problems*) por Abramson et. al. em [1];

7. Mais recentemente, podem-se referir os trabalhos de Friedlander et al. em [70], de Luo et al. em [105–107] e de Silva et. al. em [141].

O Método dos Filtros considera o problema NLP (1.2) como um problema biobjectivo, tendo em vista a minimização da função objectivo, f, (optimalidade) e uma função contínua h, que agrega as m funções restrição do problema (admissibilidade).

É óbvio que a prioridade deve ser dada a h, uma vez que não é razoável ter como solução do problema um ponto não admissível, ou seja, um ponto que não verifique as restrições.

Considere-se o problema definido em (1.2). Tendo em conta que:

$$c_i(x) = 0 \Leftrightarrow c_i(x) \leq 0 \wedge c_i(x) \geq 0 \Leftrightarrow c_i(x) \leq 0 \wedge -c_i(x) \leq 0, \tag{3.33}$$

o problema (1.2) pode escrever-se da seguinte forma:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{3.34a}$$

$$s.\ a.\ c_i(x) \leq 0, i \in 1, 2, ..., w \tag{3.34b}$$

em que $w \in \mathbb{N}$ é o número de restrições de desigualdade obtidas ($w = m + t$, no caso de existirem restrições de igualdade, senão $w = m$). Assim, a função h deve ser tal que:

$$h(x) \geq 0 \text{ com } h(x) = 0 \text{ se e só se } x \text{ é admissível.}$$

Deste modo define-se h como sendo:

$$h(x) = \|C_+(x)\|, \tag{3.35}$$

onde $\|.\|$ é a norma de um vector e $C_+(x)$ é o vector dos w valores das restrições em x, isto é, $c_i(x)$ para $i = 1, 2, ..., w$:

$$C_+(x) = \begin{cases} c_i(x) & \text{se} \quad c_i(x) > 0 \\ 0 & \text{se} \quad c_i(x) \leq 0 \end{cases}$$

Considerando a norma 1, por exemplo, tem-se: $h(x) = \|C_+(x)\|_1 = \sum_{i=1}^{w} \max(0, c_i(x))$, considerando a norma 2 tem-se:

$$h(x) = \|C_+(x)\|_2 = \sqrt{\sum_{i=1}^{w} \max(0, c_i(x))^2}. \tag{3.36}$$

Segundo Karas et. al. em [86], o Método dos Filtros define uma *região proibida*, memorizando pares $(f(x_k), h(x_k))$, com bom desempenho nas iterações anteriores, evitando pontos *dominados* (no sentido definido na regra de Pareto usual, apresentada na Definição 3.7) pelos pontos deste conjunto, nas iterações seguintes:

Definição 3.7. Um ponto $x \in \mathbb{R}^n$ *domina* $y \in \mathbb{R}^n$, e escreve-se $x \prec y$, se $f(x) \leq f(y)$ e $h(x) \leq h(y)$.

Definição 3.8. Um *filtro*, $\mathcal{F}$, é um conjunto finito de pontos em que nenhum par de pontos, x e y do conjunto $\mathcal{F}$, tem a relação $x \prec y$, ou seja, o filtro é constituído por pontos tal que nenhum deles domina outro.

Assim, um ponto é aceite para o filtro se e só se não for dominado por nenhum outro ponto do filtro e a sua inclusão elimina do filtro todos os pontos que ele dominar, ou seja, o filtro é um conjunto dinâmico. Deste modo, o filtro funciona como um critério na aceitação da iteração.

Considere-se a FIGURA 3.3, baseada na apresentada por Ribeiro et. al. em [132], que ilustra um filtro com quatro pontos (representados por a, b, c e d). Os pontos a, b, c e d do Filtro definem uma região proibida, representada a sombreado. Se o novo ponto a ser testado no filtro for o representado por y, como este está na região proibida não será aceite no filtro. Já o ponto representado por z está fora de região proibida, pelo que seria incluído no filtro. O mesmo acontece com o ponto representado por w, no entanto, neste caso, haveria ainda lugar à eliminação dos pontos representados por c e d dos filtro, uma vez que estão na região proibida definida por w, ou seja, c e d são dominados por w.

Na TABELA 3.2 apresenta-se o Algoritmo do Método dos Filtros genérico, apresentado por Karas em [85] e Ribeiro et. al. em [132], sem especificar os algoritmos internos usados

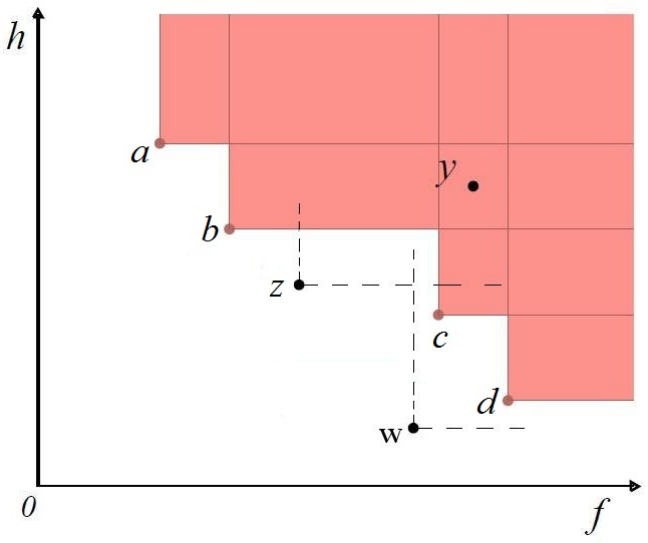

FIGURA 3.3: Um Filtro com quatro pontos
(Fonte: Karas et. al. em [85] e Ribeiro et. al. em [132])

nos passos de viabilidade e optimalidade. Neste método é construída uma sucessão de Filtros $F_0, F_1, ..., F_k$, constituídos por pares $(f(x_k), h(x_k)) \in \mathbb{R}^2$.

Note-se que na definição do par temporário para o filtro, no Algoritmo apresentado TABELA 3.2, Karas em [85] utiliza a igualdade $\left(\tilde{f}_0^j, \tilde{h}^j\right) = \left(f_0(x^k) - \alpha h(x^k), (1-\alpha)h(x^k)\right)$, tendo subtraído $\alpha h(x^k)$ a cada componente. Esta modificação evita a aceitação de pares muito próximos de iterações anteriores. Assim, inclui o conceito de *envelope*. Este conceito foi introduzido tendo em vista evitar que se acumulem pontos não estacionários no filtro, exigindo um decréscimo significativo em f ou h para a entrada no filtro.

Na FIGURA 3.4 apresenta-se um exemplo de um envelope, apresentado por Ribeiro et. al. em [132], que adiciona uma pequena margem à volta da fronteira definida pela regra de Pareto. Todos os pontos acima da linha tracejada passam a ser pontos proibidos como se pertencessem à região proibida.

A obtenção de resultados de convergência, para o Método dos Filtros em SQP, exige que se verifiquem três condições, segundo Sainvitu em [137]:

1. As funções f e c são continuamente diferenciáveis de ordem 2 em $\mathbb{R}^n$;

2. Os iterandos x^k permanecem num domínio fechado e limitado de $\mathbb{R}^n$;

TABELA 3.2: Algoritmo do Método dos Filtros genérico

Algoritmo do Método dos Filtros genérico Fonte: Karas em [85] e Ribeiro et. al. em [132].
Conhecidos o ponto inicial $x^0 \in \mathbb{R}^n$, o filtro que define a região de rejeição inicial $F_0 = \emptyset$ (inicialmente vazio), o filtro que define a região de aceitação inicial $\mathcal{F}_k = \emptyset$ (inicialmente vazio) (e no caso de Karas [85] $\alpha \in (0,1)$);
Fazer $k = 0$;
REPETIR
Definir o par temporário para o filtro $\left(\tilde{f}_0^j, \tilde{h}^j\right) \in \mathbb{R}^2$
(no caso de Karas em [85] é definido por: $\left(\tilde{f}_0^j, \tilde{h}^j\right) = \left(f_0(x^k) - \alpha h(x^k), (1-\alpha)h(x^k)\right)$);
Construir o conjunto $\bar{F}_k = F_k \cup \left\{\left(\tilde{f}_0, \tilde{h}\right)\right\}$, ou seja, introduz-se temporariamente o par no filtro;
Definir o conjunto $\bar{\mathcal{F}}_k = \mathcal{F}_k \cup \left\{x \in \mathbb{R}^n : f_0(x) \geq \tilde{f}_0, h(x) \geq \tilde{h}\right\}$;

- *Fase de viabilidade ou admissibilidade*:
  - se $h\left(x^k\right) = 0$ então fazer $z^k = x^k$, porque x^k é um ponto admissível;
  - senão, calcular o ponto intermédio z^k, a partir do ponto x^k, *mais admissível*, que não esteja no filtro, isto é, $h(z^k) < h(x^k)$ e $z^k \notin \bar{\mathcal{F}}_k$;
    se isto for impossível, parar esta fase com insucesso e passar à fase seguinte.
- *Fase de optimalidade*:
  - se z^k for um ponto estacionário de f, então parar com sucesso
  - senão, calcular um ponto x^{k+1}, com melhor valor da função f $\left(f\left(x^{k+1}\right) < f\left(z^k\right)\right)$ a partir do ponto intermédio z^k, tal que $x^{k+1} \notin \bar{\mathcal{F}}_k$.
- *Actualização do filtro*:
  Nesta fase decide-se se o par $\left(f\left(x^k\right), h\left(x^k\right)\right)$ é mantido no filtro ou se é removido, isto é:
  - se $f\left(x^{k+1}\right) < f\left(z^k\right)$, ou seja, se o ponto x^{k+1} tem um valor da função objectivo menor do que z^k, então permanece no filtro:
    $F_{k+1} = F_k$ e $\bar{\mathcal{F}}_{k+1} = \mathcal{F}_k$
  - senão, é removido:
    $F_{k+1} = \bar{F}_k$ e $\bar{\mathcal{F}}_{k+1} = \bar{\mathcal{F}}_k$
- Incrementar k: $k = k + 1$

3. Para todo o k o modelo da função objectivo é diferenciável até à ordem 2 em $\mathbb{R}^n$ e admite uma Hessiana limitada uniformemente.

Audet e Dennis em [8] utilizaram pela primeira vez o Método dos Filtros em conjunto com Métodos de Pesquisa Directa, nomeadamente com Métodos de Pesquisa em Padrão, apresentando alguns resultados de convergência. Este filtro difere dos filtros usuais em três aspectos, segundo Fletcher et. al. em [66]:

1. Só exige decréscimo simples, analogamente aos Métodos de Pesquisa em Padrão;
2. O centro de procura (*poll center*) é admissível ou é a iteração não admissível com menor valor da violação das restrições;
3. O filtro inclui um par $(0, f^F)$ correspondente a uma iteração admissível.

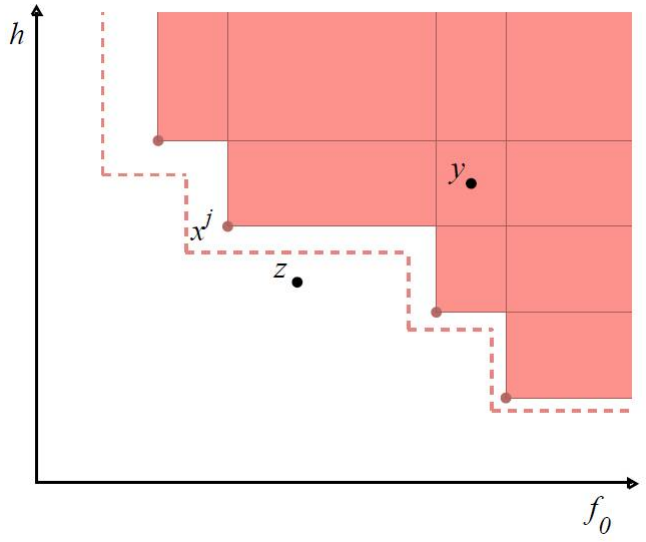

FIGURA 3.4: Envelope do Filtro
(Fonte: Ribeiro et. al. em [132])

Considere-se f^F o valor da função objectivo no melhor ponto admissível encontrado até então. O filtro de Audet aceita o ponto x^k se e só se: $h(x^k) = 0 \wedge f(x^k) < f^F$ ou $h(x^k) < h_l \vee f(x^k) < f_l, \forall (h_l, f_l) \in \mathcal{F}_k$.

Além disso, é estabelecido um limite máximo para os valores de h de pontos aceites pelo filtro, h_{max}. Só pontos x^k tais que $h(x^k) < h_{max}$ é que podem ser incluídos no filtro. Esta condição evita que o filtro aceite pontos, que apesar de terem um valor aceitável para f, estejam muito distantes da região admissível.

Na TABELA 3.3 apresenta-se o Algoritmo do Método dos Filtros de Audet apresentado por Audet et. al. em [8].

TABELA 3.3: Algoritmo do Método dos Filtros de Audet

Algoritmo do Método dos Filtros de Audet Fonte: Audet et. al. em [8]

INICIALIZAÇÃO

Seja x_0 um ponto admissível do conjunto de soluções iniciais. Incluam-se todas essas soluções no filtro $\mathcal{F}_0$, em conjunto com $h_{max} > h(x_0)$. Estabelecer o tamanho da malha (padrão) $\Delta_0 > 0$, $(\Delta_k \to 0, k \to +\infty)$ e iniciar o contador k em 0.

DEFINIÇÃO DAS SOLUÇÕES

Definir (se possível)

f_k^F: o menor valor da função objectivo de todas as soluções admissíveis encontrado até ao momento;

$h_k^I > 0$: o menor valor positivo da violação das restrições encontrado até ao momento;

f_k^I: menor valor da função objectivo dos pontos encontrados até ao momento cujo valor da violação das restrições é igual a h_k^I.

PASSOS DE SONDAGEM E PROCURA

Fazer o Passo de Sondagem e possivelmente o Passo de Procura (ou só parte dos passos) até ser encontrado um ponto x_{k+1} que é aceite no filtro, ou quando se verificou que todos os pontos de sondagem foram rejeitados por $\mathcal{F}_k$:

- PASSO DE SONDAGEM (*SEARCH STEP*): Calcular o valor de h e f no conjunto de pontos da grelha actual M_k (a estratégia usada para escolher os pontos de sondagem é fornecida pelo utilizador).

- PASSO DE PROCURA (*POLL STEP*): Calcular o valor de h e f nos pontos da pesquisa centrados em p_k, onde p_k verifica $(h(p_k), f(p_k)) = (0, f_k^F)$ ou $(h(p_k), f(p_k)) = (h_k^I, f_k^I)$

ACTUALIZAÇÃO DO PARÂMETRO

Se o Passo de Sondagem ou o Passo de Procura produziram uma iteração aceite pelo filtro:

$$x_{k+1} \in \mathcal{F}_{k+1},$$

então declarar a iteração um *sucesso* e actualizar $\Delta_{k+1} \geq \Delta_k$.

Senão, fazer $x_{k+1} = x_k$, declarar a iteração como um *insucesso* e actualizar $\Delta_{k+1} < \Delta_k$.

Incrementar k, $k \leftarrow k + 1$, e voltar ao passo DEFINIÇÃO DAS SOLUÇÕES.

Capítulo 4

Tecnologia de Programação

4.1 Escolha da Tecnologia de Programação

Tendo em vista o objectivo geral deste trabalho surge a necessidade de desenvolver uma API (*Application Programming Interface*), onde todos os métodos e algoritmos estudados e desenvolvidos estejam implementados.

Para implementar a API era necessário fazer a escolha da linguagem de programação a utilizar. Para essa escolha era de extrema importância ter em conta parâmetros como a portabilidade e o desempenho.

Uma das tecnologias a ser considerada quando falamos de portabilidade é a Tecnologia Java, que tem suporte para a maioria das plataformas de computação utilizadas, segundo Mestre em [115].

Segundo Bull et. al. em [24], a Tecnologia Java foi avaliada em relação a outras linguagens de programação utilizadas neste tipo de aplicações, tais como C e FORTRAN, e concluiu-se que em relação às outras tem um bom desempenho. Na análise por elementos finitos tem mesmo um desempenho comparável ao C, segundo Nikishkov et. al. em [122].

A escolha da Tecnologia Java, para desenvolver esta aplicação, apareceu assim antes de iniciar este trabalho, e de forma bastante imediata, na sequência de se ter constatado a diversidade de sistemas operativos usados actualmente. Assim, a portabilidade da aplicação, ou seja, a independência do sistema operativo, foi a principal motivação para a escolha desta tecnologia, suportada depois pelo bom desempenho demonstrado.

A Tecnologia Java é portável, porque depois de compilar uma aplicação uma vez, ela pode ser executada em qualquer computador, independentemente do sistema operativo. Assim, não é necessário existir uma versão compilada para cada sistema operativo em que se deseje executar a aplicação, ao contrário do que acontece com a utilização de compiladores tradicionais. A ideia forte associada à linguagem Java é *"write once, run anywhere"* (*escrever um código uma vez e executá-lo em qualquer parte*), Martins em [109].

O compilador de Java, ao contrário dos compiladores tradicionais, não gera código executável, mas sim *bytecodes*. Assim, desde que cada computador tenha um programa que possa interpretar e executar *bytecodes*, não é necessário voltar a compilar o código fonte. Este programa é um intérprete de Java, existindo um por cada plataforma. Ao interpretador *Java bytecodes* dá-se o nome de Máquina Virtual Java (JVM – *Java Virtual Machine*), segundo Mestre em [115].

A FIGURA 4.1 ilustra o processo de criação, compilação e execução dum programa em Java.

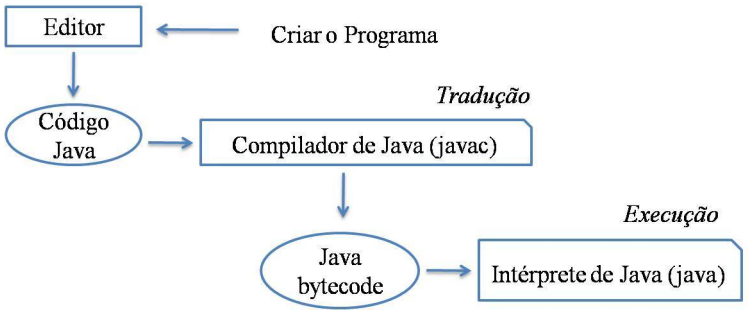

FIGURA 4.1: Processo de criação, compilação e execução dum programa em Java (Fonte: Mendes et. al. em [114])

4.2 Criação de Programas em Java

Para criar programas em Java é necessário ter o *Java Development Kit* (JDK) instalado no computador, disponível na página da *Sun Microsystems*[1]. Para o desenvolvimento de apliações em Java são também usados ambientes integrados de desenvolvimento, como

[1] http://java.sun.com/

os ambientes *open-source NetBeans IDE*[2] ou *Eclipse* [3]. Estes ambientes incluem diversas ferramentas como editor de texto para escrever programas, funcionalidades para compilar e executar programas e janelas de visualização dos resultados da compilação e execução. Neste trabalho utilizou-se o ambiente *NetBeans IDE*, mas a API resultante pode ser utilizada em qualquer outro.

4.3 História do Java

A Tecnologia Java começou a ser desenvolvida no início dos anos 90 por uma equipa da *Sun Microsystems* liderada por James Gosling, no âmbito do projecto *Green*, cujo objectivo era desenvolver uma pequena linguagem de equipamentos electrónicos com *chips* programáveis, segundo Mendes et. al. em [114], Mestre em [115] e Booch et. al. em [23].

A equipa desenvolveu então a linguagem Java, inicialmente designada por *Oak*. O primeiro produto lançado foi um controlo remoto extremamente inteligente, o *7 – *StarSeven*, que não teve o sucesso esperado e merecido. No entanto, em 1993, com o aparecimento da *World Wide Web*, o Java torna-se uma linguagem fundamental. A sua portabilidade era ideal para o desenvolvimento de aplicações que se pretendessem ser executadas em qualquer máquina ligada à *internet*. Além disso, integrava alguns mecanismos de segurança, era fácil de integrar nos navegadores da Web (*browsers*) existentes e mostrou-se ser a única forma de obter alguma dinâmica nas páginas, através dos *applets* Java. Assim aparece o *browser HotJava* em 1994, o primeiro a executar *applets* e, consequentemente, objectos dinâmicos e animados, mutáveis em conteúdo e aparência[4].

Em 1996, a *Netscape* integra a capacidade de executar código Java no seu *browser*, como acontece actualmente com todos os *browsers*. Esta integração e o aumento da importância da *internet* impulsionou a popularização da linguagem, tornando-a uma das mais utilizadas.

[2] http://www.netbeans.org/
[3] http://www.eclipse.org/
[4] http://java.sun.com/features/1998/05/birthday.html

4.4 Algumas Características da linguagem Java

A linguagem JAVA, tal como as linguagens C, Pascal, C++, entre outras, é uma linguagem considerada de *alto nível*, na medida em que é mais próxima das linguagens humanas do que as linguagens máquina ou *assembly*. É, ainda, uma linguagem *orientada aos objectos*, não sendo exactamente uma linguagem por objectos "pura", dado que nem todos os seus componentes são objectos, segundo Mendes et. al. em [114]. Nesta medida, a programação em Java é feita a dois níveis:

- O *nível dos tipos primitivos*, em que a programação se faz declarando variáveis e constantes, como sendo de um dado *tipo* (inteiro, real, caractere, booleano, etc.) e manipulando estes valores usando *operadores* definidos para esse tipo (Adição +, Subtracção −, Divisão /, Maior do que >, Igual a ==, Disjunção ||, etc.) e *estruturas de controlo* condicionais e repetitivas (*If/else, switch, for, while, do/while*);
- O *nível dos objectos*, baseado em criação de objectos e classes.

Os objectos são, então, tipos não primitivos, tal como os *arrays*, porque são tratados através de *referência*, ou seja, através duma variável que, contendo o seu endereço, permite o acesso ao seu valor efectivo. As variáveis de tipos primitivos guardam valores e não endereços de valores. Estes tipos, são designados por *tipos referenciados*.

De uma forma geral, segundo Mestre et. al. em [116], podem definir-se objectos como sendo:

Definição 4.1. Os *objectos* são entidades constituídas por código (métodos) e dados, tendo uma analogia com os objectos da vida real. Os objectos utilizados em POO (Programação Orientada a Objectos) têm características semelhantes aos objectos que utilizamos na vida real, ou seja, têm comportamentos/funções e têm estados.

Um objecto pode ser caracterizado por três componentes, ainda segundo Mestre et. al. em [116] e Mendes et. al. [114]:

- *Interface* – permite ao programador e ao sistema identificar o objecto;
- *Atributos (ou variáveis)* – permitem caracterizar e guardar o estado do objecto;

- *Comportamentos (ou Métodos)* – definem o conjunto de funcionalidades que pode executar por opção própria ou a pedido de outro objecto.

Assim, não é possível definir objectos sem definir a *classe* a que pertencem. A definição seguinte baseia-se nas apresentadas por Martins em [109], Mendes et. al. em [114], Mestre em [115] e Mestre et. al. em [116]. Tanto este conceito, como o conceito de objecto, bem como todos os outros do Java são amplamente apresentados no Tutorial sobre Java da Oracle[5].

Definição 4.2. Uma *classe* é uma especificação abstracta de propriedades, usada para criar objectos, então Classes são tipos que definem os atributos e comportamentos dos seus objectos. Funcionam como um "molde" para construir objectos dessa classe. Assim, a definição de uma classe implica a especificação dos atributos e dos comportamentos que os objectos criados a partir dela devem possuir.

Deste modo, um objecto é uma *instância* duma classe e possui os atributos (atributos de instância) e comportamentos (métodos de instância) definidos pela classe, respondendo ao mesmo conjunto de *mensagens*. Assim, uma mesma entidade (classe) reúne os seus atributos e métodos, o que permite modelar de uma forma mais simples e aproximada os elementos do mundo real. Esta é a principal vantagem da programação orientada aos objectos.

4.5 Implementação doa Algoritmos em Java

Após análise da complexidade dos algoritmos concluiu-se que estes têm uma estrutura dificilmente implementável utilizando a programação "tradicional". A forma de ultrapassar esta questão passa pela implementação de Máquinas de Estados Finitos (FSM – *Finite State Machines*).

O conceito subjacente a uma Máquina de Estados é a ideia de que um sistema evolui através de uma sequência de mudanças ou transições de estado, segundo Neto em [120]. Assim, uma máquina de estados é uma forma de fazer a modelação dum sistema composto por estados, transições e acções.

[5] http://download.oracle.com/javase/tutorial/

Um estado é indicativo de tudo o que ocorreu até àquele momento, isto é, a presença num estado implica que se tenha passado por uma série de estados anteriores e feito determinadas mudanças, desde a entrada no sistema. Uma transição consiste numa mudança de estado e implica a existência de uma condição, que tem que ser verificada para que se faça essa transição. Para a condição ser verificada é, por vezes, necessário desempenhar alguma tarefa ou acção.

As acções produzem sinais e podem ser de:

- *Entrada* – são acções que se executam para entrar no sistema ou num dado estado e sinais que entram no estado;
- *Saída ou transição* – são acções que implicam a saída do estado, que podem implicar a transição para um dos estados seguintes ou a saída do sistema, dependendo se o sinal produzido pelas acções é de saída ou transição;
- **Reacção ou actualização** – são acções que são executadas internamente num estado, em resposta ao sinal de entrada nesse estado, e que constroem o sinal de saída passo a passo pela observação do sinal de entrada.

Assim, pode definir-se o conjunto de Entrada (Saída) num estado, que é constituído por todos os sinais de entrada (saída) num estado.

Note-se que a cada sinal de entrada corresponde um e um só sinal de saída e uma mudança de estado, podendo mudar-se para o mesmo estado, neste caso, diz-se que o sinal é nulo (*absent*).

Quando o número de estados é reduzido e quando os sinais de entrada e de saída são finitos e pequenos, as Máquinas de Estados Finitos podem ser representadas através de um diagrama de transição de estados.

Parte III

Desenvolvimento e Implementação dos Algoritmos

Capítulo 5

Desenvolvimento e Implementação dos Algoritmos

5.1 Apresentação Geral do Trabalho Desenvolvido

A resolução de Problemas de Optimização Não Linear Sem Restrições, sem recorrer ao cálculo ou aproximações de derivadas nem à construção de modelos das funções envolvidas, implica a utilização de Métodos de Pesquisa Directa, nomeadamente os Métodos de Pesquisa/Procura Directa Direccional ou *Métodos de Pesquisa em Padrão – PSM (Pattern Search Methods)* e os Métodos de Pesquisa/Procura Directa Simpléticos ou *Métodos Simplex – SM (Simplex Methods)*, que se têm mostrado os Métodos de Pesquisa Directa considerados clássicos e mais utilizados pelos investigadores das mais diversas áreas, tal como foi apresentado no Capítulo 2.

Para Problemas de Optimização Não Linear Com Restrições, verificou-se, no Capítulo 3, que os métodos que os permitem resolver, sem recorrer ao cálculo ou aproximações de derivadas, nem construir modelos das funções envolvidas, são os Métodos de Penalidade e Barreira, os Métodos de Lagrangeana Aumentada e o Método dos Filtros.

Note-se que, tal como já foi referido anteriormente, estes métodos não são os mais adequados para resolver todos os Problemas de Optimização. No caso de se estar perante um problema linear, convexo, quadrático ou para o qual se conhecem e são fáceis de calcular as derivadas das função objectivo e das restrições, o mais adequado será utilizar

métodos que usem derivadas, uma vez que serão mais robustos e até eficazes a encontrar o óptimo. No entanto, se as funções envolvidas são não suaves, ou se apresentam como resultado de simulações computacionais complicadas ou se são simplesmente *caixas pretas*, então a única possibilidade de resolução são métodos que não recorrem à informação das derivadas na procura de um óptimo.

Assim, neste trabalho, dos algoritmos apresentados na FIGURA 1.4, apenas faz sentido implementar os métodos da FIGURA 5.1.

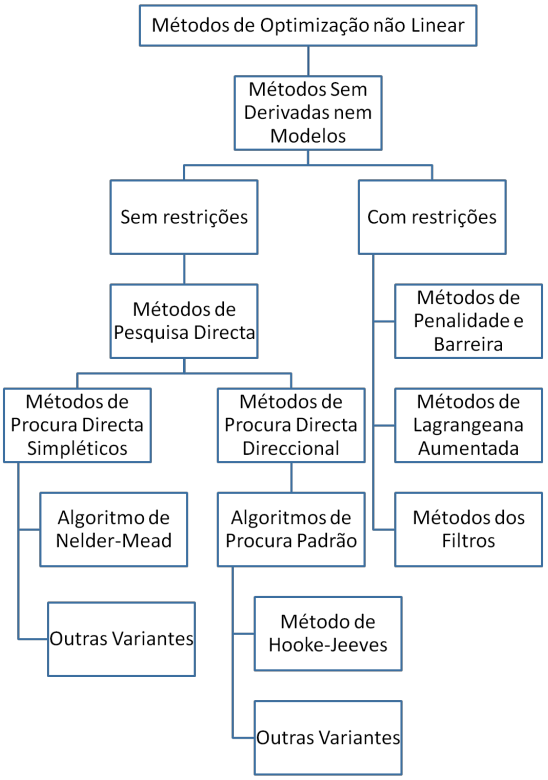

FIGURA 5.1: Métodos de Optimização Implementados

5.2 Métodos Implementados para Resolução de Problemas Sem Restrições

Para Optimização Não Linear Sem Restrições foram implementados cinco algoritmos: três Métodos de Pesquisa em Padrão (PSM) e dois Métodos Simplex (SM).

Deste modo, a cada um destes algoritmos corresponde uma classe na API desenvolvida. Os pormenores sobre estas classes serão apresentados no Capítulo 8.

O primeiro PSM implementado neste trabalho foi um Método de Pesquisa Coordenada, por ser de entre estes o mais simples e muito usado para resolver Problemas de Optimização, seja directamente ou fazendo parte de fases internas de algoritmos mais sofisticados. O método implementado está descrito mais em pormenor na secção 6.1.

O segundo PSM implementado é o Método de Hooke-Jeeves, uma vez que tem a especificidade de introduzir o passo padrão que permite, em muitos casos, uma progressão mais rápida para o óptimo. As especificações de implementação deste método são apresentadas na secção 6.2.

O último PSM é uma versão dos Algoritmos de Audet. et. al. para Optimização Não Linear Sem Restrições. Estes autores utilizaram estes métodos com grande sucesso para resolver tanto problemas sem restrições como com restrições. Neste último caso foram usados em geral, métodos de barreira extrema ou progressiva e projecção para o interior da região admissível, quando a iteração encontrada é não admissível. Estes métodos diferem dos anteriores por introduzirem um passo de sondagem na pesquisa. A versão mais recente destes métodos usa como direcções de pesquisa, no passo de sondagem, direcções geradas aleatoriamente sendo, por isso, um Método Estocástico. Neste trabalho estudam-se Métodos Determinísticos, pelo que as direcções para o passo de sondagem são escolhidas à partida, são fixas e em número finito. Nos Algoritmos de Audet. et. al. é considerada a possibilidade dessas direcções serem em número infinito (ou a tender para) o que implica um número crescente de avaliações da função objectivo. No caso dessa avaliação ser feita através de processos morosos, por exemplo se resultar de complexos processos de simulação, esta pode ser uma característica desfavorável à boa execução do método. Os detalhes de implementação, da versão implementada aqui, apresentam-se na secção 6.3.

O quarto Método de Pesquisa Directa implementado é o Método de Nelder-Mead e é um SM. Este método é um dos Métodos de Pesquisa Directa mais utilizados e, em alguns problemas, revela-se extremamente eficaz a encontrar o óptimo. No entanto, possui algumas limitações relacionadas com problemas de geometria do simplex ao longo do processo de optimização em alguns problemas. Na generalidade dos problemas, nomeadamente de pequenas dimensões, o desempenho deste método é dos melhores entre os aqui apresentados, mas o facto de não se garantir que a geometria se mantém estável ao longo do processo faz com que em alguns problemas este método falhe, convergindo para pontos

não estacionários. A demonstração da sua convergência também não tem sido obtida em parte devido à falta de garantia de descida suficiente. O método de Nelder-Mead aqui apresentado difere do original, sendo o método implementado por Matias em [110]. Os pormenores de implementação deste método estão descritos na secção 6.4.

Tendo presentes as limitações do Método de Nelder-Mead, foi implementado um segundo SM, apresentado por Conn et. al em [32], denominado por Simplex Convergente. Neste algoritmo a geometria do simplex é controlada através do seu diâmetro e volume normalizado e é introduzido um passo de salvaguarda para situações em que as condições de geometria não se verificam. Ainda é introduzida neste método uma condição que impõe uma descida suficiente do valor da função objectivo. Com estas novas condições os autores provam que o algoritmo converge para um ponto estacionário. Os detalhes de implementação apresentam-se na secção 6.5.

Depois de implementados estes cinco Métodos de Pesquisa Directa, usando a API que é constituída por todos os algoritmos necessários, implementados usando a Tecnologia Java, foram resolvidos 25 problemas (ver 6.6.1), escolhendo os parâmetros de entrada correspondentes e o mais similares possível, no sentido de comparar o seu desempenho. Os resultados obtidos são discutidos na secção 6.6.2. Esta comparação será útil para conhecer a eficiência e eficácia de cada um dos Métodos de Optimização Não Linear Sem Restrições, uma vez que estes serão usados nos processos internos dos Métodos de Optimização Com Restrições.

5.3 Métodos Implementados para Resolução de Problemas Com Restrições

No que diz respeito à Optimização Não Linear Com Restrições foram implementados apenas dois dos Métodos Possíveis: Métodos de Penalidade/Barreira e Método dos Filtros.

A cada um destes algoritmos corresponde uma classe na API desenvolvida. Os pormenores sobre estas classes e sobre classes auxiliares à sua execução serão apresentados no Capítulo 8.

O Método de Lagrangeana Aumentada, tal como foi exposto no Capítulo 3, secção 3.3, transforma o Problema Com Restrições num Problema Com Restrições apenas de Igualdade e Limites Simples, usando variáveis de folga. Estas variáveis são introduzidas nas restrições de limites simples do problema. O Problema original é então transformado numa sequência de problemas de limites simples, mas com maior dimensão, uma vez que as variáveis de folga são introduzidas como variáveis do problema. As primeiras abordagens usando a Função Lagrangeana Aumentada usavam informação sobre derivadas das funções envolvidas. Outras abordagens foram posteriormente desenvolvidas, como a de Lewis et. al. que é uma abordagem de Pesquisa Directa, uma vez que no processo de optimização dos problemas de limites simples usa o Método de Pesquisa em Padrão Generalizado (GPS). Espírito Santo, seguindo a sugestão apresentada por estes autores e na tentativa de evitar o inconveniente do aumento de dimensão do problema, não usa variáveis de folga, mas define uma nova Função Lagrangeana Aumentada. Esta nova função incorpora informação das restrições de igualdade e desigualdade, reduzindo o problema a problemas de limites simples. Estes problemas são depois resolvidos usando o Método de Pesquisa em Padrão de Hooke-Jeeves Generalizado. Neste trabalho não foram implementados métodos de Pesquisa em Padrão nem Métodos Simplex Generalizados, capazes de lidar com problemas com restrições apenas do tipo limites simples, uma vez que as restrições do tipo limite simples podem ser transformadas em restrições de desigualdade:

$$l \leq x \leq u \Leftrightarrow -x + l \leq 0 \wedge x - u \leq 0 \tag{5.1}$$

pelo que não é possível implementar o Método de Lagrangeana Aumentada, tal como é apresentado nesses trabalhos. A possibilidade disponível é transformar as restrições que estão na forma $l \leq x \leq u$, usando a equivalência (5.1). Neste caso o método não seria muito diferente dos Métodos de Penalidade implementados.

Esta poderá ser uma das estratégias a implementar no futuro, bem como a generalização dos Métodos de Pesquisa Directa.

Assim, neste trabalho serão tratados problemas cuja forma geral é a apresentada em (1.2).

Na secção 3.2 foram apresentados alguns dos Métodos de Penalidade e Barreira. As funções Barreira Extrema (3.4) e Progressiva (3.5) têm sido muito utilizadas com Métodos de Pesquisa Directa, pelo que ambos os métodos foram implementados. Os Métodos

de Barreira Inversa (3.6 e 3.7) e Barreira Logarítmica (3.8 e 3.9) têm o inconveniente de não estarem definidas para alguns valores das restrições, nomeadamente quando o seu valor é zero, no caso da Barreira Inversa, e quando o seu valor é zero ou negativo, no caso da Barreira Logarítmica. Assim, optou-se por não implementar estes métodos. Sabendo que os Métodos de Barreira são mais adequados para usar quando os pontos iniciais são admissíveis, foram realizados testes numéricos com 18 problemas, alguns com pontos iniciais admissíveis e outros com pontos iniciais não admissíveis, no sentido de comparar o seu desempenho nas duas situações.

Dos Métodos de Penalidade apresentados na secção 3.2 só não foi implementado o Método de Penalidade Adaptativa (3.19 e 3.20) por ser similar ao Método de Penalidade Dinâmica (3.15, 3.16, 3.17 e 3.18).

Os Métodos de Penalidade têm-se mostrado eficazes a resolver problemas com restrições nas mais diversas áreas da optimização, mas têm o grande inconveniente de ser necessário, na maioria dos métodos, definir à partida um número grande de parâmetros para iniciar o processo. Esta escolha de parâmetros não tem regras pré-definidas, sendo muitas vezes feita empiricamente. Além disso, uma escolha de parâmetros pode ser adequada para um problema e não o ser no problema seguinte. Deste modo os métodos aqui implementados usam por defeito valores de parâmetros similares para todos os métodos, não se fazendo um estudo exaustivo da adequabilidade dos parâmetros, no entanto permite-se que o utilizador os defina. Assim a API está desenhada para que este possa ser um dos trabalhos a ser realizado no futuro.

O Método dos Filtros, cujas características principais foram descritas na secção 3.4, evitam a definição ou escolha de parâmetros de penalidade. Esta foi a principal motivação para terem surgido, por Fletcher e Leyffer, em 2002. Depois disso, este método foi utilizado nas mais diversas áreas da Optimização, mas em Pesquisa Directa só tinha sido aplicado por Audet (e outros autores que trabalharam com ele). Assim, o interesse de estudar a aplicabilidade deste método em Pesquisa Directa foi considerado de extrema importância e mostrou-se ser um dos assuntos centrais deste trabalho, não só por ainda não ter sido implementado em conjugação com outros Métodos de Pesquisa Directa, mas também porque algumas das suas características serviram de motivação para adaptações que foram efectuadas no algoritmo de Penalidade/Barreira. Os conceitos associados à sua implementação são relativamente simples e os resultados numéricos obtidos têm sido

bastante satisfatórios. No entanto os algoritmos disponíveis (TABELAS 3.2 e 3.3) são demasiado genéricos, não especificando claramente os passos de implementação, mas mostrando apenas as ideias gerais da implementação. Assim, foi necessário repensar a forma de implementação deste método. O esquema geral de implementação bem como os restantes pormenores de implementação e versões que foram sendo desenvolvidas são expostas na secção 7.2.

Nos métodos de Penalidade/Barreira os resultados de saída usuais são: número de iterações efectuadas no processo externo, número de vezes que foi calculada o valor da função penalidade/barreira, última iteração efectuada, valor da função penalidade/barreira na última iteração efectuada e valor da função objectivo na última iteração efectuada.

No Método dos Filtros não é encontrada apenas uma única solução, mas sim o Filtro final, ou seja, o conjunto de soluções não dominadas, designado usualmente por Frente de Pareto. Assim, o decisor, perante um conjunto de soluções possíveis para o problema pode escolher a que melhor se adeqúe à situação real em questão. Além disso, ao longo do processo iterativo, à semelhança dos métodos dos filtros de Audet, o método implementado neste trabalho guarda a melhor solução admissível encontrada (se existir), ou seja, das iterações que não violam as restrições, a que tem menor valor da função objectivo e a melhor solução não admissível encontrada (se existir), ou seja, das iterações que violam as restrições, a que tem menor valor de violação.

No sentido de ampliar o tipo de soluções encontradas pelos Métodos de Penalidade/Barreira e com a motivação das vantagens oferecidas pela existência de diversas opções de solução do Método dos Filtros, foram acrescentados procedimentos nos Métodos de Penalidade/Barreira implementados, para guardar informação sobre a melhor solução admissível encontrada (se existir), a melhor solução não admissível encontrada (se existir) e ainda o valor da violação das restrições na melhor solução não admissível, $\Phi(x_{ki}) - f(x_{ki})$. Deste modo, se forem encontradas iterações nestas condições, ficam disponíveis para o utilizador três soluções, para que ele escolha a que melhor se adequa à sua realidade. Esta alteração ainda permite que, havendo a exigência da solução ser admissível, se a última iteração não o for (apesar de ter o melhor valor da Função Penalidade/Barreira) estará disponível uma solução admissível, neste caso a melhor encontrada no processo iterativo. Podem ser vistos mais detalhes de implementação dos Métodos de Penalidade/Barreira na secção 7.1.

Nos Métodos de Penalidade/Barreira a violação das restrições é penalizada (ou implica a rejeição de pontos), ou seja, dependendo da admissibilidade da iteração, a função objectivo da sucessão de problemas sem restrições incorpora, além da função objectivo do problema, uma medida para a violação das restrições. No Método dos Filtros o objectivo é minimizar a função objectivo e simultaneamente uma função h que agregue as funções restrições do problema. Esta função é, em geral, definida utilizando a norma 2. No entanto a única exigência para esta função é que seja contínua e que: $h(x) \geq 0$ com $h(x) = 0$ se e só se x é admissível.

Tendo em conta este facto, foram implementadas outras medidas para a violação das restrições, nomeadamente medidas semelhantes às usadas nos Métodos de Penalidade/Barreira para penalizar ou rejeitar iterações.

Tanto nos Métodos de Penalidade/Barreira como o Método dos Filtros é necessário, no processo interno, resolver problemas sem restrições, ou seja, problemas do tipo:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{5.2}$$

onde n é a dimensão do problema, $f : \mathbb{R}^n \to \mathbb{R}$ é a função objectivo (no caso dos métodos de Penalidade/Barreira a função a optimizar é Φ_k e no Método dos Filtros optimizam-se h e f). No âmbito deste trabalho serão usados para resolver estes problemas os cinco Métodos de Pesquisa Directa apresentados anteriormente. Note-se que no caso do Método dos Filtros este procedimento só foi feito por Audet, utilizando apenas Métodos de Pesquisa em Padrão, nomeadamente o Método de Pesquisa Coordenada e outras versões deste método.

Depois de implementados estes métodos, a que podemos chamar *Métodos de Pesquisa Directa para Optimização Com restrições*, usando a API desenvolvida, foram resolvidos 18 problemas (ver 7.3.1), escolhendo os parâmetros de entrada correspondentes e o mais similares possível, no sentido de comparar o seu desempenho. A escolha dos parâmetros foi feita tendo em conta os valores que costumam ser mais utilizados nos diversos trabalhos da área e que foram apresentados aos longo do Capítulo 2.

5.4 Implementação

Nos Capítulos 6 e 7 apresentam-se os pormenores de implementação dos algoritmos referidos, respectivamente, para Optimização sem Restrições e para Optimização com Restrições.

Deste modo, estes capítulos são iniciados com a apresentação dos algoritmos implementados modelados sob a forma de Máquina de Estados. Seguidamente, são apresentados resultados numéricos que permitem comparar o comportamento dos algoritmos implementados e são testadas as estratégias alternativas implementadas.

No capítulo 8 apresenta-se com mais pormenor a API, nomeadamente no que diz respeito a classes e respectivos construtores (correspondentes aos algoritmos implementados), à aplicação gráfica desenvolvida e ao seu funcionamento. Neste capítulo também se poderão visualizar com mais pormenor os dados de entrada e saída de cada um dos métodos.

Capítulo 6

Pormenores de Implementação dos Algoritmos para Optimização sem Restrições

Neste trabalho foram implementados, usando Máquinas de Estados Finitos, cinco algoritmos para Optimização sem Restrições:

1. Um algoritmo de Pesquisa Coordenada – CS (Secção 2.3.2);
2. O algoritmo de Hooke e Jeeves – HJ (Secção 2.3.3.1);
3. Uma versão dos algoritmos de Audet et. al. – AA (Secção 2.3.3.2);
4. O algoritmo de Nelder-Mead – NM (Secção 2.4.2);
5. Um algoritmo Simplex Convergente – SC (Secção 2.4.3).

Estes algoritmos são compostos por diversos estados, sendo que o Estado 1 é o estado de entrada no sistema e o Estado final é o estado de saída do sistema.

Os algoritmos evoluem através de uma sequência de mudanças ou transições de estado, que são consequência do cumprimento da tarefa do estado e/ou dos valores das variáveis à saída do estado e que produzem as respectivas reacções.

Optou-se por uma representação dos algoritmos implementados através de fluxograma, salientando-se através de rectângulos os estados, por se considerar que seria a forma mais elucidativa de apresentar as acções a realizar ao longo dos processos.

6.1 Algoritmo de Pesquisa Coordenada

O algoritmo de Pesquisa Coordenada aqui implementado é um algoritmo oportunista, baseado no apresentado genericamente na TABELA 2.1. Este algoritmo é constituído por 11 estados e o seu funcionamento é apresentado no fluxograma da FIGURA 6.1.

Nesta implementação a base utilizada como conjunto de direcções de pesquisa foi a base canónica de $\mathbb{R}^n$, (2.4), onde $n = dim$ é a dimensão do problema, sendo a ordem de pesquisa a seguinte: $e_1, -e_1, e_2, -e_2, ...$

O comprimento do passo foi mantido, no caso de iterações bem sucedidas, e reduzido a metade, no caso de iterações mal sucedidas.

Foram utilizados quatro critérios de paragem do processo iterativo:

- Três deles têm que ser verificados simultaneamente (designados no fluxograma por CP):
  - A condição de proximidade (1.24) – com tolerância $T1$;
  - A condição de controle dos valores da função objectivo (1.23) – com tolerância $T2$;
  - Um limite mínimo para o comprimento do passo α_{min} – com tolerância $T3$;
- O quarto critério, no Estado 9, controla o número Máximo de iterações k_{max} do processo iterativo.

Note-se que nos critérios de paragem foram usadas desigualdades estritas, ou seja, o processo iterativo pára se $d < T1 \ \wedge \ df < T2 \ \wedge \ \alpha < T3$, ou seja, se a distância entre duas iterações consecutivas for menor do que a tolerância $T1$, a distância entre os valores da função objectivo em duas iterações consecutivas for menor do que a tolerância $T2$ e o tamanho do passo for menor do que a tolerância $T3$. O processo iterativo também

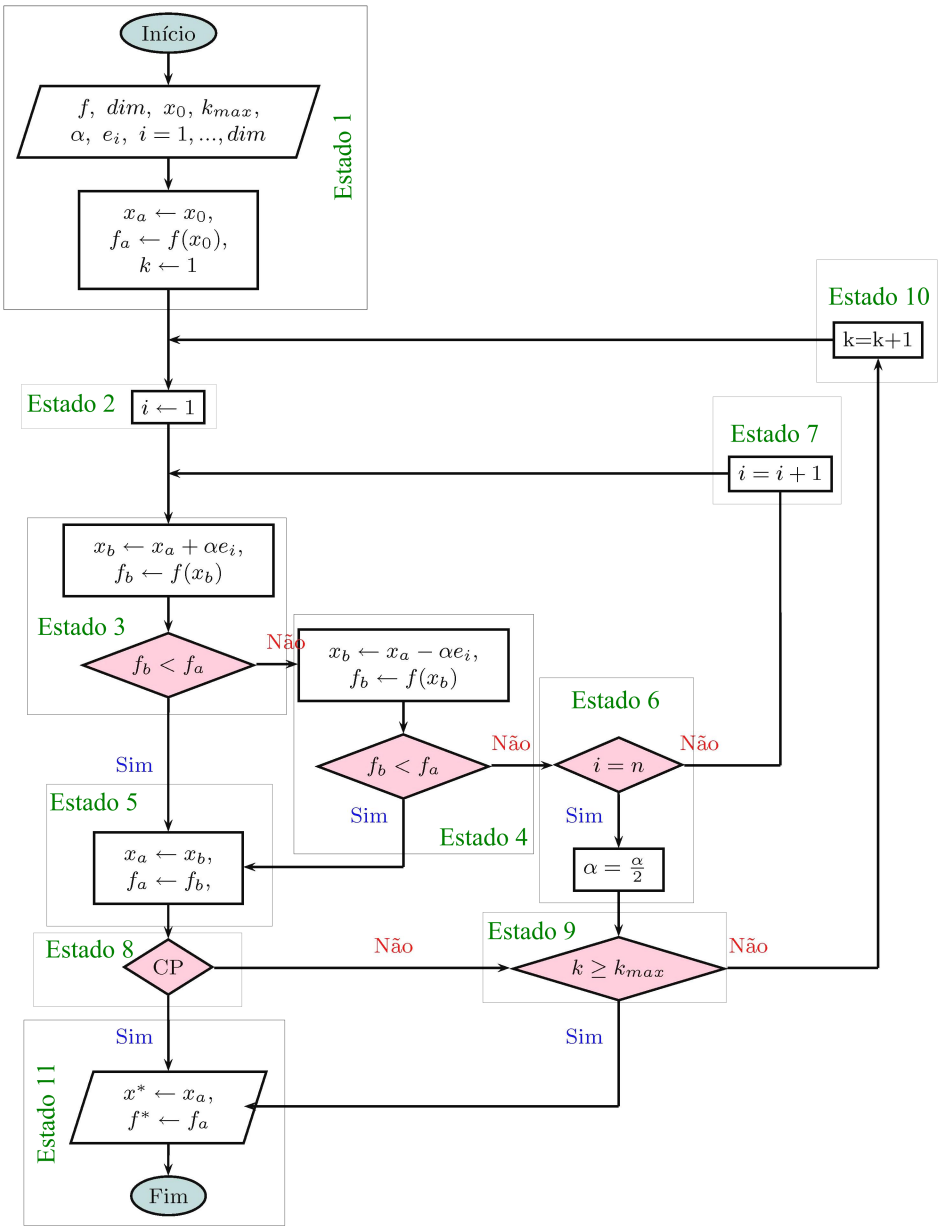

FIGURA 6.1: Algoritmo de Pesquisa Coordenada implementado

pára se o número de iterações efectuadas atinge o número Máximo de iterações definido inicialmente, k_{max}.

Os valores de entrada para este algoritmo são:

- Função objectivo, f;
- Ponto inicial, x_0 (do qual se define também a dimensão do problema – dim);
- Número Máximo de iterações (por defeito $k_{max} = 100$);

- Tamanho do passo inicial (por defeito $\alpha = 1$);
- Tolerância para a distância entre duas iterações (por defeito $T1 = 0.00001$);
- Tolerância para a distância entre dois valores consecutivos da função objectivo (por defeito $T2 = 0.00001$);
- Valor mínimo para o tamanho do passo (por defeito $T3 = 0.001$).

Note-se que os valores definidos por defeito podem ser alterados pelo utilizador da API.

Este algoritmo devolve os seguintes resultados:

- Número de vezes que a função objectivo foi calculada, $fEvals$;
- Número de iterações efectuadas, k;
- Iteração em que foi encontrada a solução, $ksuc$;
- Número de iterações bem sucedidas ao longo do processo iterativo, suc;
- Número de iterações mal sucedidas ao longo do processo iterativo, $insuc$;
- Últimos valores calculados para os critérios de paragem, d, df e α (com tolerâncias, $T1$, $T2$ e $T3$, respectivamente);
- Aproximação à solução encontrada, x^*;
- Valor da função objectivo na aproximação à solução encontrada, $f(x^*)$.

6.2 Algoritmo de Hooke e Jeeves

O algoritmo de Hooke e Jeeves implementado baseia-se no algoritmo exposto genericamente na TABELA 2.2. O esquema geral de funcionamento do algoritmo implementados é apresentado no fluxograma da FIGURA 6.2 e é constituído por 17 estados.

Nesta implementação também foi utilizada a base canónica de $\mathbb{R}^n$ como conjunto de direcções de pesquisa e a mesma ordem de pesquisa, com $n = dim$ a dimensão do problema.

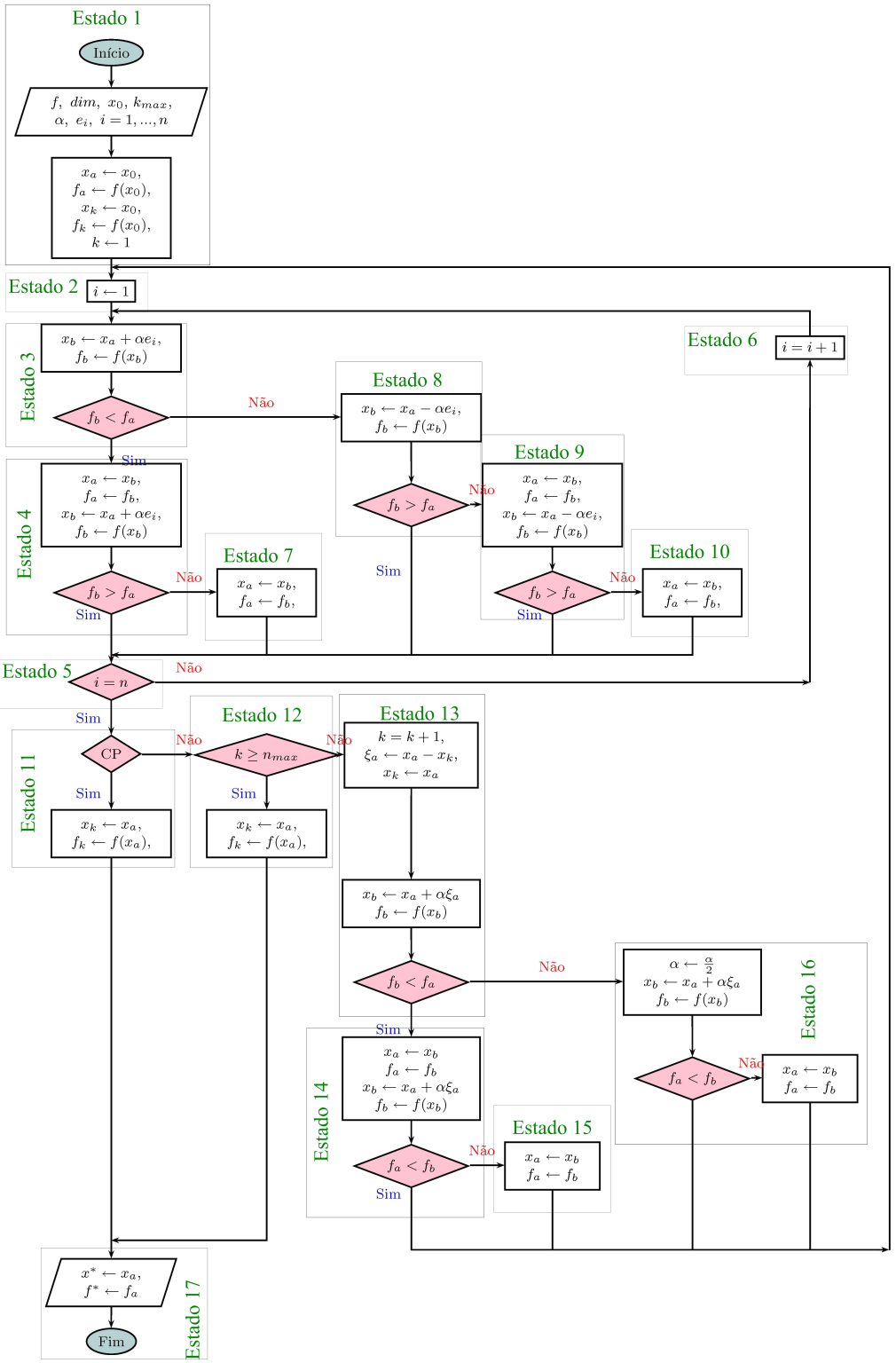

FIGURA 6.2: Algoritmo de Hooke e Jeeves implementado

O comprimento do passo também foi mantido, no caso de iterações bem sucedidas, e reduzido a metade, no caso de iterações mal sucedidas.

Foram utilizados os mesmos quatro critérios de paragem do algoritmo de Pesquisa Coordenada, assim como os mesmos dados de entrada por defeito e o algoritmo devolve os mesmos resultados.

6.3 Uma versão dos algoritmos de Audet et. al.

Os algoritmos de Audet et. al.: MADS [10], MADS-PB [11] e OrthoMADS [2]; referidos nas secções 2.3.3.2 e 3.2.1.1 foram desenvolvidos para resolver problemas com restrições. No entanto, é possível implementar um método de Optimização sem restrições usando uma das ideias base destes métodos: a existência de duas fases de pesquisa (Passo de Sondagem e Passo de Pesquisa), ilustradas na Tabela 2.3.

Os algoritmos MADS usam como direcções de pesquisa, no Passo de Sondagem, direcções geradas aleatoriamente, formando um conjunto denso de direcções. Para as calcular é gerada uma matriz B não singular triangular inferior de números inteiros e as suas colunas são permutadas aleatoriamente. As colunas da matriz resultante B_k formam uma base que é considerada na construção da base maximal $D_k = [B_k \ -B_k]$ utilizada no Passo de Sondagem. A geração de um conjunto denso de direcções permite provar a convergência do método, no entanto na prática nunca é gerado um conjunto infinito de direcções, apenas um conjunto de direcções geradas aleatoriamente que apesar de poderem ser distintas nunca são em número infinito. Esta forma de geração de direcções de pesquisa tem como consequência que execuções diferentes do algoritmo, para os mesmos dados de entrada, podem conduzir a resultados diferentes.

No algoritmo aqui implementado a escolha das direcções de pesquisa é determinística, de acordo com o requisito inicial de se estudarem apenas métodos determinísticos. As direcções de pesquisa escolhidas são as colunas duma matriz triangular inferior não singular de uns, ou seja, a matriz B é uma matriz $n \times n$ do tipo:

$$B = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

pelo que D_k será constituída pelos vectores coluna da matriz D:

$$D = \begin{bmatrix} 1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 & -1 & -1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & \cdots & 1 & -1 & -1 & \cdots & -1 \end{bmatrix}$$

Assim, a versão dos algoritmos de Audet et. al. aqui implementado difere do Algoritmo de Pesquisa Coordenada porque inclui, antes da pesquisa coordenada, um passo de sondagem nestas direcções.

Note-se que o último vector de B é um vector da base canónica de $\mathbb{R}^n$, e_n, com $n = dim$ a dimensão do problema, no entanto a pesquisa não será feita no mesmo ponto nos dois passos, uma vez que, havendo lugar a duas fases de pesquisa, haverá dois comprimentos do passo diferentes, uma da malha (no Passo de Sondagem) α_k^m e outro do padrão da pesquisa coordenada (no Passo de Pesquisa) α_k^p. Estes comprimentos devem verificar as seguintes condições, segundo os autores em [12]:

- $0 < \alpha_k^m \leq \alpha_k^p$;
- Se $\alpha_k^m \to 0$ quando $k \to +\infty$ então $\alpha_k^p \to 0$ quando $k \to +\infty$.

Para alcançar este objectivo a actualização dos comprimentos dos passos α_k^m e α_k^p é feita utilizando o processo que propõe Audet et. al. em [12]:

$$\begin{cases} \alpha_k^p = \sqrt{\alpha_k^m} & se \quad \alpha_k^m < 1 \\ \alpha_k^p \geq 1 & se \quad \alpha_k^m = 1 \end{cases} \tag{6.1}$$

A base utilizada como conjunto de direcções de pesquisa, no Passo de Pesquisa, foi a base canónica de $\mathbb{R}^n$, (2.4), sendo a ordem de pesquisa a seguinte: $e_1, -e_1, e_2, -e_2, ...$ e $n = dim$ a dimensão do problema.

O comprimento dos passos da malha (no Passo de Sondagem) e do padrão da pesquisa coordenada (no Passo de Pesquisa) foram mantidos, no caso de iterações bem sucedidas. No caso de iterações mal sucedidas o passo da malha α_k^m foi reduzido a metade e o passo do padrão α_k^p foi actualizado de acordo com (6.1), sendo que, no caso de $\alpha_k^m = 1$ considerou-se $\alpha_k^p = 1.5$ (Estes valores também foram utilizados na API como valores iniciais dos passos por defeito, mas podem ser alterados pelo utilizador).

A versão dos algoritmos de Audet et. al. implementado, exposto genericamente na TABELA 2.3, é apresentado no fluxograma da FIGURA 6.3 e é constituído por 18 estados. No fluxograma da FIGURA 6.3, para simplificação de escrita, em vez de α_k^p e α_k^m foram usadas as designações α_p e α_m; os vectores coluna de D são designados por D_i, onde $i = 1, ..., dim$.

Foram utilizados os mesmos quatro critérios de paragem do processo iterativo utilizados nos algoritmos de Pesquisa Coordenada e de Hooke-Jeeves (com α_m em vez de α) e o algoritmo tem os mesmos dados de entrada (à excepção do comprimento do passo α, uma vez que, neste caso, serão dois comprimentos do passo α_p e α_m), devolvendo no final os mesmos resultados.

6.4 Algoritmo de Nelder-Mead

O algoritmo de Nelder-Mead aqui implementado, exposto genericamente na TABELA 2.4, é constituído por 17 estados e é apresentado no fluxograma da FIGURA 6.4.

Nesta implementação é utilizada a base canónica de $\mathbb{R}^n$ para iniciar o simplex, onde $n = dim$ é a dimensão do problema. Os $n + 1$ vértices que constituem o Simplex inicial são, então, o ponto inicial $x_0 = v_1$ e os restantes n vértices, que são calculados através da fórmula: $v_i = v_1 + se_{i-1}$, $i = 2, ..., n + 1$, com s a medida da aresta do simplex. Nesta implementação considerou-se, por defeito, $s = 1$, no entanto este é um dos parâmetros que pode ser alterado pelo utilizador.

Os parâmetros de reflexão, expansão e contracção utilizados por defeito foram, respectivamente, $\alpha = 1$, $\gamma = 2$ e $\beta = 0.5$, mas também podem ser alterados pelo utilizador.

Foram utilizados cinco critérios de paragem do processo iterativo:

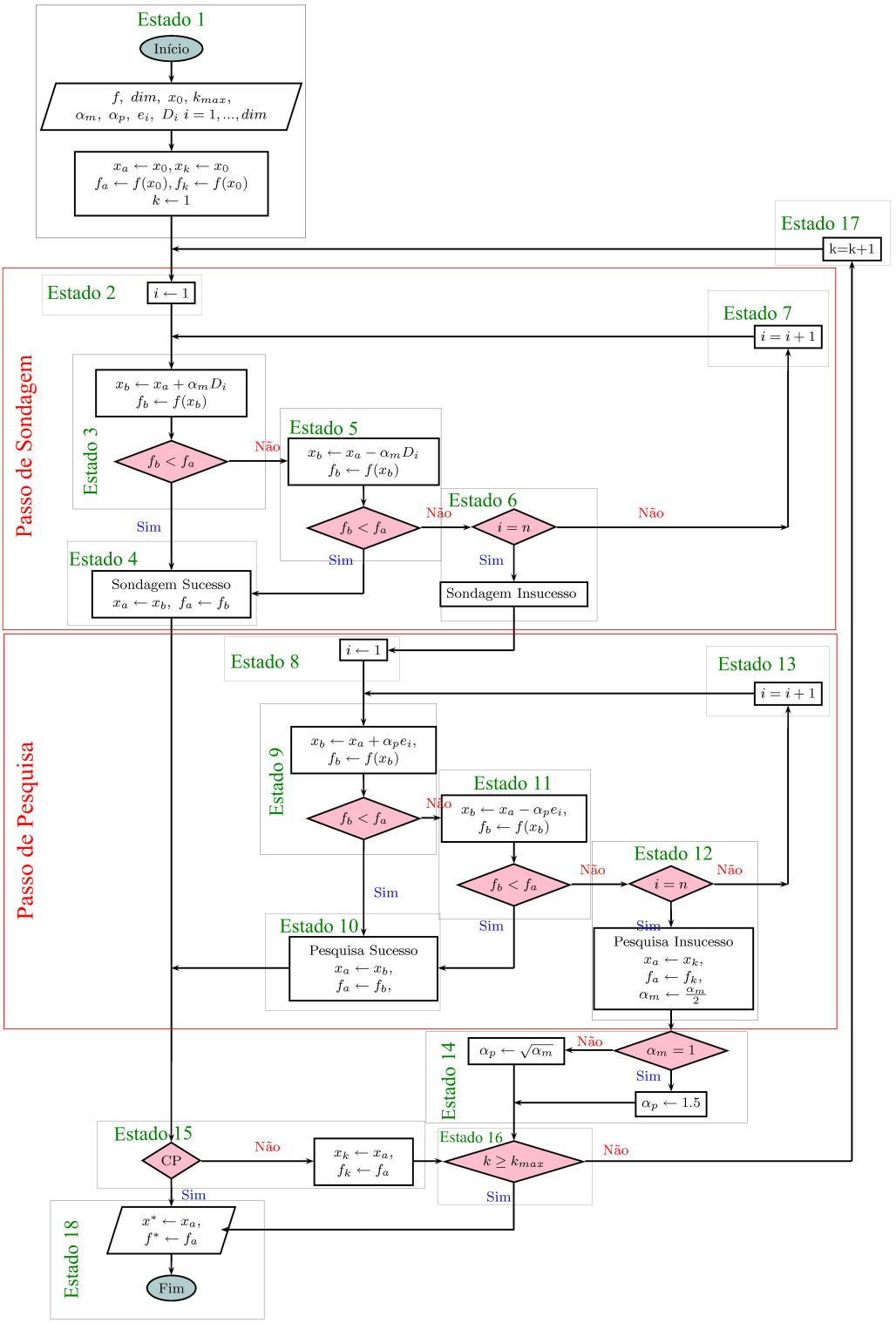

FIGURA 6.3: Versão dos algoritmos de Audet et. al.

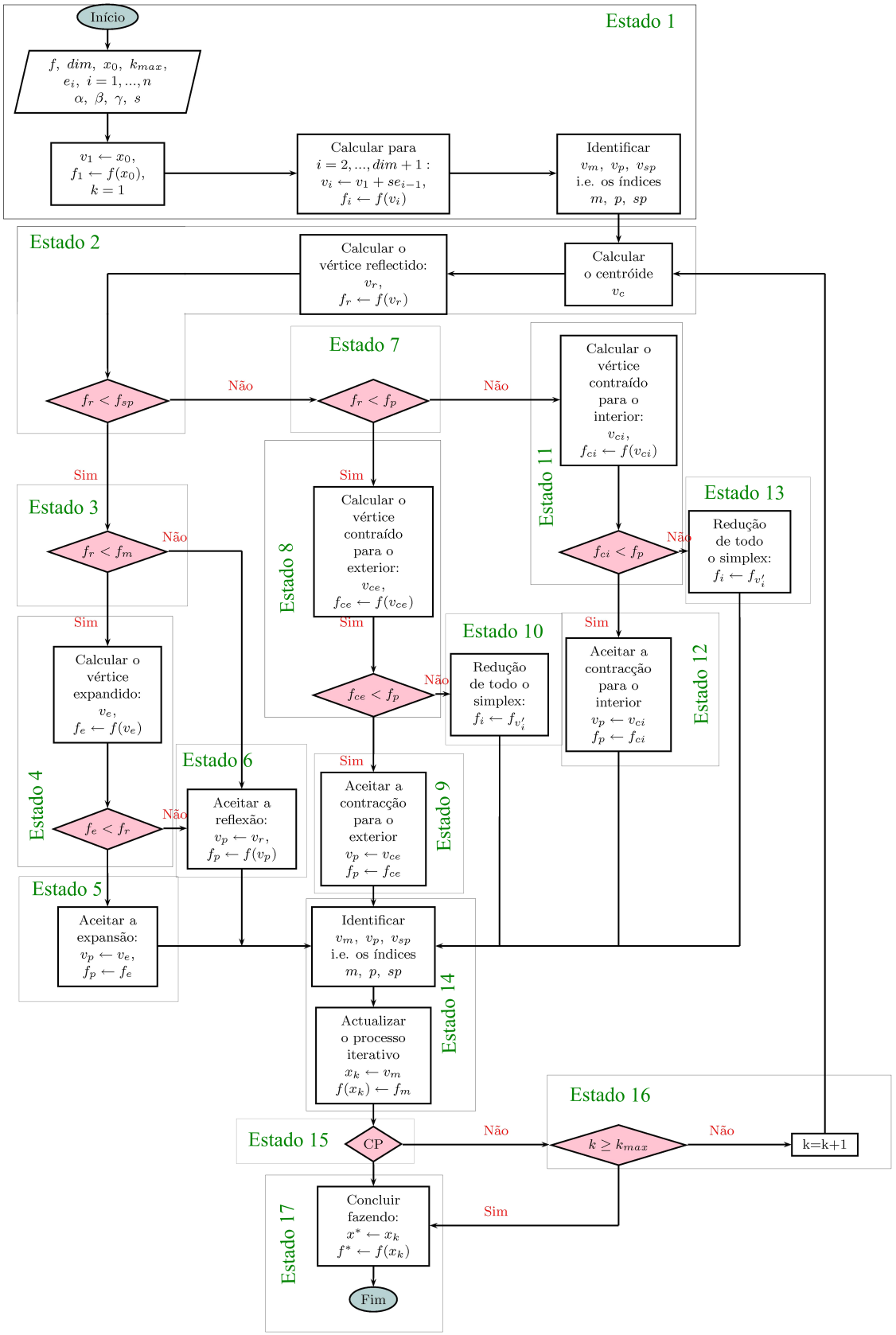

FIGURA 6.4: Algoritmo de Nelder-Mead implementado

- Quatro deles têm que ser verificados simultaneamente (designados no fluxograma por CP):
  - A condição de proximidade (1.24) (ou (1.26) para problemas que convergem para soluções nulas) – com tolerância $T1$;
  - A condição de proximidade (1.25) (ou (1.27) para problemas que convergem para soluções nulas) – com tolerância $T4$;
  - A variância dos valores de f (1.28) deve ser menor do que uma dada tolerância $T5$;
  - A diferença entre valores extremos da função nos vértices do simplex da iteração actual (1.29) – com tolerância $T6$;
- O quinto critério controla o número Máximo de iterações k_{max} do processo iterativo.

Os valores utilizados por defeito para as tolerâncias são $T1 = T4 = T5 = T6 = 0.00001$.

Neste algoritmo também foram usadas desigualdades estritas nos critérios de paragem, ou seja, o processo iterativo pára se $d < T1 \ \wedge \ d2 < T4 \ \wedge \ var < T5 \ \wedge \ df < T6$, ou seja, se a distância entre as duas iterações anteriores consecutivas for menor do que a tolerância $T1$, a distância entre os vértices com valores extremos da função objectivo no simplex for menor do que a tolerância $T4$, a variância dos valores da função objectivo for menor do que a tolerância $T5$ e a distância entre valores extremos da função objectivo nos vértices do simplex da iteração actual for menor do que a tolerância $T6$. O processo iterativo também pára se o número de iterações efectuadas ultrapassa ou é igual ao número Máximo de iterações definido inicialmente, k_{max}.

Este algoritmo devolve os seguintes resultados:

- Número de vezes que a função objectivo foi calculada, $fEvals$;
- Últimos valores calculados para os critérios de paragem, d, $d2$, var e df (com tolerâncias, $T1$, $T4$, $T5$ e $T6$, respectivamente);
- Aproximação à solução encontrada, x^*;
- Valor da função objectivo na aproximação à solução encontrada, $f(x^*)$.

6.5 Algoritmo Simplex Convergente

O algoritmo Simplex Convergente implementado, exposto genericamente na TABELA 2.5, difere do algoritmo de Nelder-Mead, nomeadamente, no controle da geometria do simplex, através do seu diâmetro e volume normalizado e na implementação de um passo de salvaguarda para quando o simplex não verifica as condições de geometria estabelecidas. O algoritmo implementado é constituído por 21 estados e é apresentado no fluxograma da FIGURA 6.5.

Nesta implementação é utilizada a base canónica de $\mathbb{R}^n$, onde $n = dim$ é a dimensão do problema, para iniciar o simplex. Os $n+1$ vértices que constituem o Simplex inicial são, então, o ponto inicial $x_0 = v_1$ e os restantes n vértices, com s a medida da aresta do simplex, calculados através da formula: $v_i = v_1 + se_{i-1},\ i = 2, ..., n+1$. Nesta implementação também se considerou $s = 1$, mas neste algoritmo s também é um dado de entrada.

A primeira diferença deste algoritmo relativamente ao de Nelder-Mead apresenta-se logo no Estado 1, onde se controla o volume normalizado do simplex inicial, (2.15). Se o volume normalizado é maior do que uma certa tolerância ξ o processo continua, senão o processo pára, sendo aconselhada a escolha de um novo ponto de partida.

Os parâmetros de reflexão, expansão e contracção utilizados foram, respectivamente, $\alpha = 1$, $\gamma = 2$ e $\beta = 0.5$ e o parâmetro de redução $\gamma_s = 0.5$, no entanto a API permite que o utilizador defina outros valores para estes parâmetros.

Foram utilizados os mesmos cinco critérios de paragem do processo iterativo (designados no fluxograma da FIGURA 6.5 por CP e 7) considerados para o algoritmo de Nelder-Mead.

Os passos do fluxograma na FIGURA 6.5 representados por losangos com a descrição dv1, dv2 e dv3 consistem em pontos de controle do diâmetro (2.14), $\Delta = diam(S_k)$, e volume normalizado (2.15) do simplex actual S_k, de acordo com as desigualdades (2.16), (2.17) e (2.18), respectivamente, estabelecidas na TABELA 2.5, ou seja:

- dv1:

$$\begin{aligned} \text{diam}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_r\}\right) &\leq \gamma^e \Delta \\ \text{von}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_r\}\right) &\geq \xi \end{aligned} \tag{6.2}$$

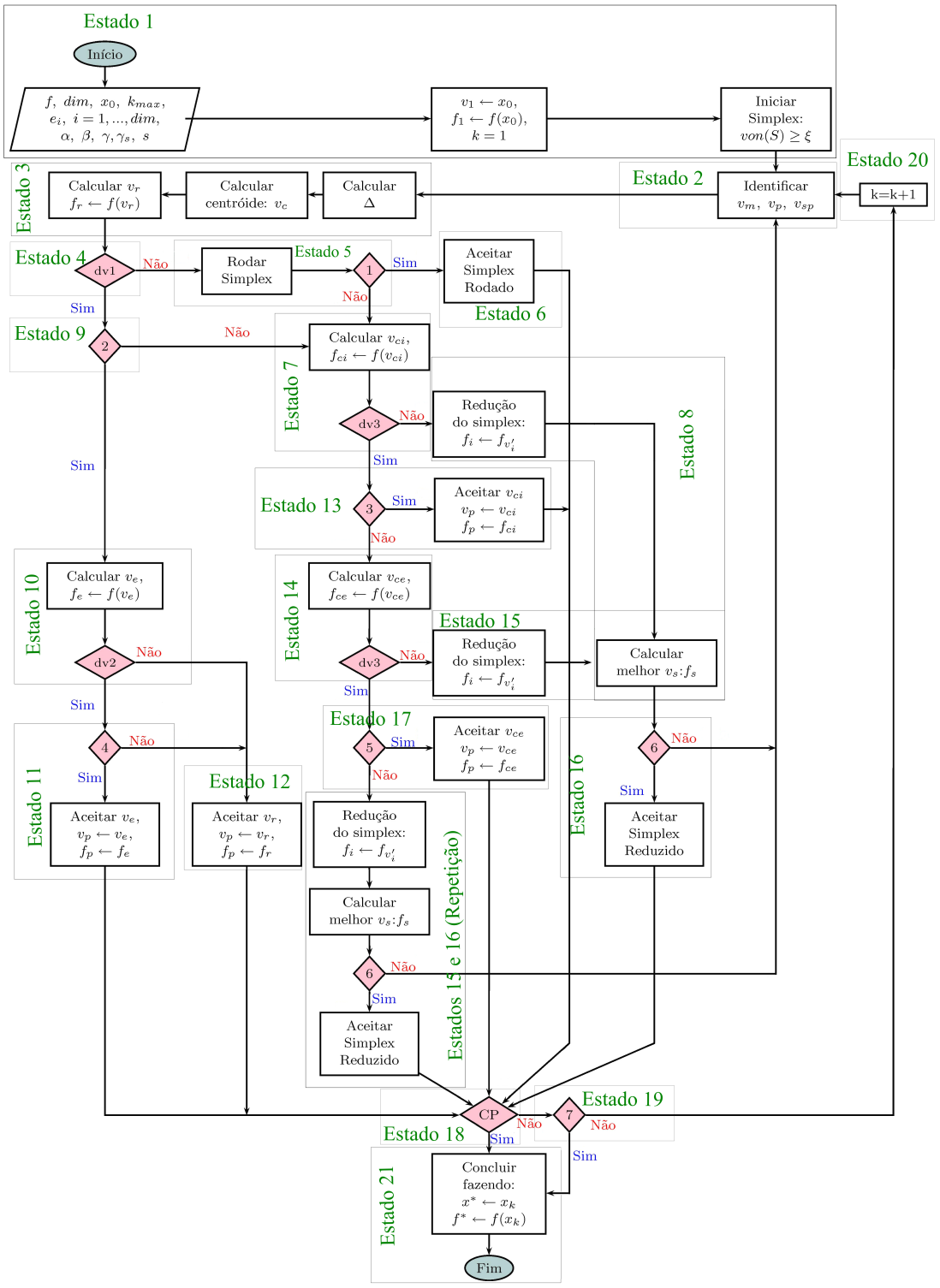

FIGURA 6.5: Algoritmo Simplex Convergente implementado

- dv2:

$$\begin{aligned} \operatorname{diam}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_e\}\right) &\leq \gamma^e \Delta \\ \operatorname{von}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_e\}\right) &\geq \xi \end{aligned} \tag{6.3}$$

- dv3:

$$\begin{aligned} \operatorname{diam}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_{cc}\}\right) &\leq \Delta \\ \operatorname{von}\left(\{v_0, v_1, v_{n-1}\} \cup \{v_{cc}\}\right) &\geq \xi \end{aligned} \tag{6.4}$$

Nesta implementação considerou-se $\gamma^e = 2$.

Os restantes losangos representam pontos de decisão, de acordo com a FIGURA 6.6, onde a função ρ é uma função continua e positiva, dita *forcing function*, tal como foi apresentada na TABELA 2.5, tal que: $\rho :]0, +\infty[\to \mathbb{R}^+$; $\lim_{t \to 0^+} \frac{\rho(t)}{t} = 0$ e $\rho(t_1) \leq \rho(t_2)$ se $t_1 < t_2$. A função utilizada foi a sugerida pelos autores em [32], definida por $\rho(t) = t^2$.

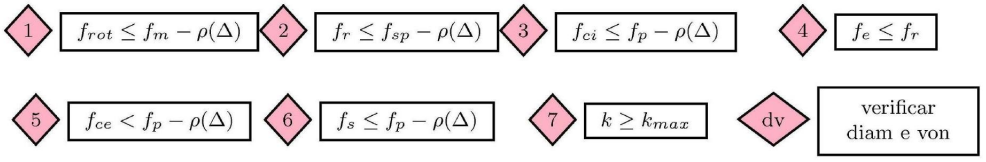

FIGURA 6.6: Pontos de decisão do Algoritmo Simplex Convergente

Este algoritmo devolve os seguintes resultados:

- Número de vezes que a função objectivo foi calculada, $fEvals$;
- Últimos valores calculados dos critérios de paragem (CP);
- Aproximação à solução encontrada, x^*;
- Valor da função objectivo na aproximação à solução encontrada, $f(x^*)$.

Note-se que, à semelhança do algoritmo de Nelder-Mead, nesta implementação não se fez a organização dos vértices do simplex pela ordem dos valores da função objectivo, como os autores apresentam em [32]. Nesta implementação foi utilizado o procedimento sugerido por Matias em [110] para o Método de Nelder-Mead: faz-se a identificação dos índices m, p e sp correspondentes à posição ocupada no simplex pelos melhor, pior e segundo melhor vértices no simplex.

Além disso, em vez de se implementar apenas a contracção para o interior ou a contracção para o exterior (tal como sugeriam Conn et.al. em [32] cujo algoritmo é o apresentado

na TABELA 2.5), foram implementados ambos os movimentos. Se a contracção para o interior não for bem sucedida ainda é testado o vértice contraído para o interior e somente em caso de insucesso é implementado o passo de Redução do simplex.

6.6 Resultados Numéricos

Nesta secção são apresentados resultados numéricos que permitem comparar o comportamento dos algoritmos. Para fazer essa comparação são primeiro apresentados os problemas teste, depois os resultados obtidos na resolução de cada um deles e por fim faz-se uma análise final dos resultados obtidos.

6.6.1 Problemas

Com o objectivo de testar os Métodos de Pesquisa Directa para Optimização Sem Restrições implementados, era necessário realizar alguns testes computacionais. Assim, tal como já foi referido na secção 5.1, foram resolvidos 25 problemas (que são expostos no Apêndice A), escolhendo os parâmetros de entrada, de acordo com o exposto nas secções anteriores, o mais similares possível, no sentido de comparar o seu desempenho de forma mais equitativa.

Dos 25 problemas seleccionados:

- 10 são problemas apresentados por Bagirov em [15]: B1, B2, B3, B4, B5, B6, B7, B8, B9 e B10;
- 13 são de Schittkowski em [138]: S201, S202, S205, S206, S207, S208, S211, S212, S213, S240, S246, S256 e S257;
- os dois últimos são problemas de Maratos apresentados por Matias em [110]: M500 e M501.

6.6.2 Comparação de Resultados e Adequação de Parâmetros de Entrada

Os resultados numéricos finais obtidos da resolução dos problemas da secção anterior, usando os parâmetros por defeito, são apresentados nesta secção. Para ver os resultados

obtidos ao longo do processo iterativo, que conduzem a estes resultados finais, basta executar os algoritmos correspondentes usando a API ou a sua componente gráfica, inserindo os respectivos dados de entrada. Os ficheiros resultantes da execução efectuada para a presente secção estão disponíveis no CD anexo a este trabalho.

Nesta secção discutem-se e comparam-se esses resultados, no sentido de conhecer a eficiência de cada um dos Métodos de Optimização Não Linear Sem Restrições implementados. Estes resultados também são importantes nos algoritmos de Optimização Com Restrições, uma vez que serão utilizados nos seus processos internos.

Com o objectivo de efectuar uma comparação dos resultados obtidos, com os diversos algoritmos, são aqui apresentados os resultados mais significativos, nomeadamente o número de avaliações da função objectivo $fEvals$ e o número de iterações efectuadas k, uma vez que estes, no caso de funções para as quais estes algoritmos são desenvolvidos, podem ser dados decisivos para a "qualidade" da solução encontrada. Ainda é incluído o valor da função objectivo na solução encontrada através do respectivo algoritmo, $f(x_k)$ e o valor da função objectivo apresentado originalmente como valor mínimo para o respectivo problema, $f(x^*)$ (exposto no Apêndice A), para ser mais simples averiguar se os resultados obtidos são próximos do objectivo.

Note-se que os parâmetros ou valores de entrada utilizados são os parâmetros definidos por defeito, que foram apresentados anteriormente. Nesta secção discutem-se estes resultados e testam-se alterações de parâmetros, com o objectivo de melhorar o desempenho dos métodos que se mostrarem menos eficazes. Essas alterações incidem essencialmente sobre os comprimentos dos passos iniciais, uma vez que é o parâmetro de entrada comum a todos os métodos.

No sentido de poder estabelecer-se se uma aproximação à solução é uma "Boa" aproximação ou não, fixa-se o valor $0.0001 = 10^{-5}$ como tolerância para a distância dos valores de f, ao valor exacto da função objectivo no mínimo, $|f(x^*) - f(x_k)|$. Este valor foi o seleccionado para medir a qualidade da solução porque é o mais usado para as tolerâncias neste tipo de métodos.

Assim,

- Se $|f(x^*) - f(x_k)| \geq 0.0001$ considera-se a aproximação à solução como uma *Boa* aproximação;

- Se $0.0001 < |f(x^*) - f(x_k)| \leq 0.01$ considera-se a aproximação à solução como uma aproximação *Média*.
- Se $|f(x^*) - f(x_k)| > 0.01$ considera-se a aproximação à solução *Má*;

Tendo em conta estes critérios foi efectuada a classificação das aproximações obtidas. Os cálculos correspondentes podem ser vistos no ficheiro *Parametros+Classificacao.xlsx* ou *Parametros+Classificacao.ods* no CD anexo a este trabalho.

Por uma questão de simplificação de escrita usaram-se abreviaturas para referir os algoritmos implementados, nomeadamente:

- CS – Algoritmo de Pesquisa Coordenada;
- HJ – Algoritmo de Hooke e Jeeves;
- AA – Versão dos algoritmos de Audet et. al.;
- NM – Algoritmo de Nelder-Mead;
- SC – Algoritmo Simplex Convergente.

6.6.2.1 Problemas de Bagirov

Problema B1	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	105	35	1.9522262775970682	**0.000002277597060063**	**Boa**
HJ	170	27	1.9680690421896776	0.015845042189670100	Má
AA	152	28	2.2414810835524905	0.28925708355249000	Má
NM	96	48	1.9522244981595898	**0.000000498159580031**	**Boa**
SC	199	51	1.9675740609777739	0.015350060977770100	Má
Objectivo	$f(x^*) = 1.952224$				

Tabela 6.1: Resultados para o Problema B1

Os Métodos de Pesquisa Directa implementados permitem todos resolver razoavelmente o Problema B1, de acordo com os dados da Tabela 6.1, no entanto, tendo em conta o critério de classificação definido, as aproximações encontradas pelos algoritmos HJ, AA e SC não são *Boas* aproximações. Assim, será de analisar os parâmetros usados na implementação destes métodos, nomeadamente os tamanhos dos passos utilizados.

No que diz respeito ao algoritmo HJ, dos vários testes efectuados o melhor valor obtido para f foi para um comprimento do passo inicial $\alpha = 2.3$. Os valores obtidos com este

passo inicial são $f(x_k) = 1.9523614599596226$ com 93 avaliações da função objectivo e 14 iterações. O melhor resultado obtido com o algoritmo AA foi considerando $\alpha_m = 2.75$ e $\alpha_p = 0.4$, obtendo-se $f(x_k) = 1.9529704302621242$, com 201 avaliações de f e 31 iterações. Assim, não se encontrou uma *Boa* aproximação à solução com nenhum dos dois algoritmos, de acordo com o critério definido.

Note-se que apesar da actualização dos valores de atribuídos α_m e α_p ter sido feita usando as condições estabelecidas em 6.1, inicialmente não verificam a condição $0 < \alpha_k^m \leq \alpha_k^p$, no entanto foram os valores que permitiram chegar ao melhor valor encontrado com o algoritmo AA. Este facto volta a ocorrer em alguns dos problemas apresentados, pelo que não se fará referência a este facto, por ser óbvio por observação identificar quando tal sucede.

Usando o passo inicial $s = 2.75$ no algoritmo SC a aproximação à solução encontrada é uma *Boa* aproximação, uma vez que para este comprimento do passo se obtém o valor da função objectivo $f(x_k) = 1.9522244957935837$ ($|f(x^*) - f(x_k)| < 10^{-5}$), ao fim de 117 avaliações de f e 49 iterações.

Assim, foi possível encontrar uma *Boa* aproximação à solução para o Problema B1, usando os algoritmos CS, NM e SC. As aproximações obtidas usando os algoritmos HJ e AA, mesmo tentando uma melhora, alterando os tamanhos dos passos iniciais, não devolveram *Boas* aproximações à solução, de acordo com o critério definido.

Problema B2	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\lvert f(x^*) - f(x_k)\rvert$	Class.
CS	398	100	16.0	8.800000000000000000	Má
HJ	72	11	8.0	0.800000000000000000	Má
AA	198	34	7.971825880678733	0.771825880678730000	Má
NM	96	49	7.200003551011001	**0.000003551010999914**	**Boa**
SC	205	93	7.200000018962497	**0.000000018962490245**	**Boa**
Objectivo	$f(x^*) = 7.2$				

TABELA 6.2: Resultados para o Problema B2

Os resultados obtidos para o Problema B2 (ver TABELA 6.2) já mostram algumas diferenças. Os Métodos Simplex (NM e SC) encontram uma *Boa* aproximação à solução, com o valor da função objectivo próximo do desejado. Os valores obtidos pelos algoritmos de Audet e HJ também se aproximam do valor pretendido, mas constituem *Boas* aproximações. Assim, para estes algoritmos é necessário definir outros parâmetros iniciais.

No caso do Algoritmo de Pesquisa Coordenada, basta considerar $\alpha = 2.6$ que o valor obtido é $f(x_k) = 7.20000823975788$, com 99 avaliações da função objectivo e 31 iterações.

Relativamente ao algoritmo HJ, dos vários testes efectuados o melhor valor obtido para f foi para um comprimento do passo inicial $\alpha = 2.6$. Com este passo inicial obtém-se $f(x^*) = 7.200189790951651$, $fEvals = 98$ e $k = 15$. Assim, não se encontrou uma *Boa* aproximação à solução com este algoritmo, de acordo com o critério definido.

Alterando os passos no algoritmo de Audet, por exemplo, para $\alpha_m = 1.6$ e $\alpha_p = 2.6$ obtém-se $f(x^*) = 7.199999999999999$, $fEvals = 433$ e $k = 58$.

Assim, foi possível encontrar uma *Boa* aproximação à solução para o Problema B2, usando todos os algoritmos, de acordo com o critério definido, excepto como algoritmo HJ.

Problema B3	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\lvert f(x^*) - f(x_k)\rvert$	Class.
CS	794	100	10.0	10.0	Má
HJ	111	10	10.0	10.0	Má
AA	799	75	2.1052537491073053E-10	**2.1052537491073053E-10**	**Boa**
NM	197	100	9.211172647652553E-4	9.211172647652553E-4	Média
SC	232	100	4.719706419107953E-4	4.719706419107953E-4	Média
Objectivo	$f(x^*) = 0$				

TABELA 6.3: Resultados para o Problema B3

Para o Problema B3 apenas o algoritmo AA encontra uma *Boa* aproximação à solução com os parâmetros por defeito (ver TABELA 6.3). Os valores obtidos com os Métodos Simplex (NM e SC) também não estão longe da solução, mas pelo critério definido não são *Boas* aproximações à solução.

No entanto, considerando $\alpha = 2$, o algoritmo de CS encontra a solução indicada no problema original, com 789 avaliações de f e 100 iterações; o algoritmo de HJ encontra-a ao fim de 123 avaliações da função objectivo e 11 iterações.

Relativamente ao algoritmo NM, dos vários testes efectuados o melhor valor obtido para f foi para um comprimento do passo inicial $s = 3.9$. Com este passo inicial obtém-se $f(x_k) = 1.6289976335256924E-4$, $fEvals = 196$ e $k = 100$. Assim, não se encontrou uma *Boa* aproximação à solução com este algoritmo, de acordo com o critério definido.

As alterações ao tamanho do passo para o algoritmo SC não permitiram uma melhoria da aproximação à solução que já tinha sido encontrada.

Problema B4	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	1000	100	16.0	16.0	Má
HJ	225	15	25.0	25.0	Má
AA	1989	100	9.0	9.0	Má
NM	181	100	0.008251692421031426	0.008251692421031426	Média
SC	980	54	2.583077715151689	2.583077715151689	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.4: Resultados para o Problema B4

Segundo os dados da TABELA 6.4, nenhum dos algoritmos, usando os parâmetros por defeito, permite encontrar uma *Boa* aproximação à solução do Problema B4. O algoritmo de NM é o que melhor a aproxima, mas os restantes dão resultados pouco satisfatórios. Assim, para este problema será necessário testar outros tamanhos do passo iniciais.

No caso do algoritmo de HJ o melhor valor encontrado foi para um passo $\alpha = 8.42$ para o qual $f(x_k) = 0.03779496476996102$, depois de 391 avaliações da função objectivo e 27 iterações; com o mesmo tamanho do passo inicial o algoritmo de CS encontra uma aproximação à solução mais próxima do pretendido com o valor da função objectivo $f(x^*) = 4.198030638961514E-18$ e considerada *Boa*, de acordo com o critério definido, ao fim de 697 avaliações da função objectivo e 100 iterações. No algoritmo AA, usando o mesmo valor para o passo $\alpha_m = 8.42$ e $\sqrt{\alpha_m} \approx 2.9 = \alpha_p$, obtém-se $f(x^*) = 1.1353141070843003E-5$, efectuando 1144 avaliações da função objectivo e fazendo 100 iterações.

No algoritmo de NM, considerando o tamanho do passo inicial $s = 8.42$ o seu desempenho também melhora, obtendo-se $f(x^*) = 3.107792000099394E-5$, com 196 avaliações da função objectivo e 100 iterações; o mesmo acontece ao algoritmo SC com o qual se obtém, ao fim do mesmo número de iterações, para $s = 8.42$, uma aproximação à solução com valor da função objectivo $f(x^*) = 3.052907845647507E-5$, ao fim de 346 avaliações da função objectivo.

Note-se que o valor para o passo inicial 8.42 foi usado em todos os métodos porque, dos vários valores testados, era o que permitia obter o melhor resultado usando o algoritmo de HJ. Nos restantes algoritmos existiam outros valores que também permitiam obter *Boas* aproximações à solução, mas por uma questão de coerência na comparação, este foi o usado em todos os algoritmos.

Assim, foi possível encontrar uma *Boa* aproximação à solução para o Problema B4, usando todos os algoritmos, de acordo com o critério definido, excepto como algoritmo HJ.

Problema B5	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	1000	100	4.0	4.0	Má
HJ	225	15	5.0	5.0	Má
AA	1989	100	3.0	3.0	Má
NM	181	100	0.0908388266163287	0.0908388266163287	Má
SC	908	40	2.636334495507589	2.636334495507589	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.5: Resultados para o Problema B5

Os resultados apresentados na TABELA 6.5 não constituem *Boas* aproximações à solução do Problema B5, pelo que, à semelhança do que foi feito para os problemas anteriores, se testaram outros valores para os passos iniciais.

O valor de α, de entre os testados, que devolveu melhor valor de f para o algoritmo HJ, analogamente ao que aconteceu no problema anterior, foi $\alpha = 8.42$, para o qual $f(x_k) = 0.19440927130659438$, ao fim de 391 avaliações de f e 27 iterações.

Usando este tamanho do passo no algoritmo CS obtém-se uma *Boa* aproximação à solução ao fim de 697 avaliações de f e 100 iterações, com $f(x_k) = 2.0489096219603036E - 9$.

No algoritmo AA, usando o mesmo valor para o passo $\alpha_m = 2.75$ e $\sqrt{\alpha_m} \approx 1.66 = \alpha_p$, obtém-se $f(x_k) = 0.002639660795666819$, efectuando 972 avaliações da função objectivo e fazendo 100 iterações.

Usando o algoritmo de NM o melhor resultado obtido para a função objectivo foi $f(x_k) = 2.748972661094526E - 4$, $fEvals = 200$ e $k = 100$, para $s = 7.18$.

O melhor resultado obtido com o algoritmo SC foi usando o tamanho do passo $s = 9.2$ obtendo-se $f(x_k) = 0.0016055947016297193$ com $fEvals = 372$ e k=100.

Assim, apenas se consegui uma *Boa* aproximação à solução do problema B5, de acordo com o critério definido, usando o algoritmo CS, com $\alpha = 8.42$, com os restantes algoritmos foi possível melhorar as aproximações à solução, mas não foram encontradas *Boas* aproximações à solução.

Problema B6	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	488	100	0.9712266204375002	0.9712266204375002	Má
HJ	1267	100	0.6024268886867177	0.6024268886867177	Má
AA	492	100	8.053301799463416	8.053301799463416	Má
NM	197	100	0.01570932226037624	0.01570932226037624	Má
SC	286	100	0.026113770851487677	0.026113770851487677	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.6: Resultados para o Problema B6

Na TABELA 6.6 pode ver-se que apesar de todos os algoritmos, à excepção do AA, aproximarem a solução, de acordo com o critério definido, não são *Boas* aproximações da solução do Problema B6.

No entanto, com os valores testados para os valores dos passos iniciais, não se encontraram aproximações *Boas*, houve melhoria mas pouco significativa (considerando $\alpha = 0.75$ o algoritmo HJ devolveu $f(x_k) = 0,113747463261829$; com $\alpha_m = 0.20$ e $\alpha_p = 0.45$ o algoritmo AA devolveu $f(x_k) = 0,375970770130362$; com $s = 0.35$ obteve-se $f(x_k) = 0,0120925466250873$ com o algoritmo de NM e não se encontrou um valor para s que tivesse melhorado o desempenho do algoritmo SC).

Com o algoritmo CS, considerando um passo inicial $\alpha = 0.2$ encontrou-se uma *Boa* aproximação à solução, com $f(x_k) = 1.9454956053016614E-5$, $fEvals = 375$ e $k = 62$.

Problema B7	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	115	36	2.00000120401382	2.00000120401382	Má
HJ	73	11	2.0193359375000046	2.0193359375000046	Má
AA	211	36	2.00000120401382	2.00000120401382	Má
NM	197	100	1.000000275711803	1.000000275711803	Má
SC	254	100	1.4513710019966242	1.4513710019966242	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.7: Resultados para o Problema B7

Os resultados apresentados na TABELA 6.7 não constituem *Boas* aproximações à solução do Problema B7, no entanto considerando o tamanho do passo inicial $\alpha = 2.2$ os algoritmos CS e AA (com $\alpha_m = 2.2$ e α_p definido por defeito) devolvem exactamente a solução do problema. No algoritmo CS ao fim de 216 avaliações da função objectivo e 55 iterações e no algoritmo AA ao fim de 429 avaliações de f e com $k = 55$.

Considerando s com o mesmo valor inicial ($s = 2.2$) os algoritmos HJ, NM e SC também permitem obter *Boas* soluções, todas com $f(x_k) = 2.24265050974281E-14$. Para tal o

algoritmo HJ faz 77 avaliações de f e 12 iterações, com o algoritmo NM obtém-se 197 avaliações de f e $k = 100$ e com o algoritmo SC 114 avaliações de f e $k = 152$.

Assim, pode concluir-se que, com os parâmetros de entrada adequados, todos os Métodos de Pesquisa Directa permitem encontrar uma *Boa* aproximação à solução do Problema B7.

Problema B8	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	783	100	22.2	22.2	Má
HJ	110	10	22.2	22.2	Má
AA	1501	100	0.0	**0.0**	**Boa**
NM	193	100	2.5467266448866694	2.5467266448866694	Má
SC	207	100	27.071263339821524	27.071263339821524	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.8: Resultados para o Problema B8

Dos resultados apresentados na TABELA 6.8 apenas o algoritmo AA permite encontrar exactamente a solução. Assim, foram testadas outras possibilidades para os valores dos comprimentos do passo iniciais em cada um dos algoritmos.

Com os algoritmo NM e SC não se encontrou um valor para s que tivesse melhorado o desempenho dos algoritmos.

Para o algoritmo CS, dos valores testados, o valor $\alpha = 2.93$ foi o que permitiu obter o melhor resultado, no entanto a correspondente aproximação à solução não é uma *Boa* aproximação, de acordo com o critério estabelecido ($f(x_k) = 0,0441269531250315$). Também foi possível melhorar a aproximação à solução obtida com o algoritmos HJ (com $\alpha = 45.65$), no entanto também não é uma *Boa* aproximação à solução ($f(x_k) = 2,34417725806599$).

Deste modo, apenas se consegui uma *Boa* aproximação à solução do problema B8, de acordo com o critério definido, usando o algoritmo AA e com os passos definidos por defeito.

Problema B9	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	992	100	16.0	16.0	Má
HJ	210	15	25.0	25.0	Má
AA	1953	100	9.0	9.0	Má
NM	193	100	0.05467105623136796	0.05467105623136796	Má
SC	242	100	0.0029876383892611436	0.0029876383892611436	Média
Objectivo	$f(x^*) = 0$				

TABELA 6.9: Resultados para o Problema B9

Os resultados apresentados na TABELA 6.9, obtidos para o Problema B9, podem ser melhorados com outros valores para os passos iniciais, nomeadamente, para $\alpha = 7.5$ usando o algoritmo CS obtém-se $f(x_k) = 4.55191440096314E-14$, $fEvals = 716$ e $k = 100$; para o algoritmo HJ, com o mesmo valor do passo inicial tem-se $f(x_k) = 2.84223193312042E-12$, $fEvals = 333$ e $k = 24$; com $s = 7.5$ o algoritmo NM devolve $f(x_k) = 5.36183750990827E-5$, $fEvals = 838$ e $k = 100$ e algoritmo SC $f(x_k) = 1.98194230700271E-5$, $fEvals = 188$ e $k = 100$. O algoritmo AA com $\alpha_m = 2.75$ e $\alpha_p = 1.66$, à semelhança do que foi considerado no Problema B5, permite encontrar os seguintes resultados: $f(x_k) = 7.29602570524535E-5$, $fEvals = 306$ e $k = 100$.

Assim, todos os algoritmos permitem encontrar uma *Boa* aproximação à solução do problema B9, de acordo com o que foi estabelecido.

Problema B10	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	377	58	1.153887348327638	1.153887348327638	Má
HJ	176	13	0.3074445312499995	0.3074445312499995	Má
AA	500	38	1.5421425227051628	1.5421425227051628	Má
NM	218	100	0.31635508369046295	0.31635508369046295	Má
SC	276	100	0.06256266187125432	0.06256266187125432	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.10: Resultados para o Problema B10

Os resultados para o Problema B10, apresentados na TABELA 6.10, mostram que nenhum dos algoritmos permite encontrar uma *Boa* aproximação à solução. Assim, foram testadas outras possibilidades para os valores dos comprimentos do passo iniciais em cada um dos algoritmos.

As melhorias obtidas nos resultados não devolvem qualquer aproximação à solução que seja uma *Boa* aproximação, de acordo com o critério estabelecido. Deste modo, não se conseguiu encontrar uma *Boa* aproximação à solução do problema B10, usando os algoritmos de Pesquisa Directa.

6.6.2.2 Problemas de Schittkowski

Nas TABELAS 6.11, 6.12, 6.13, 6.14 e 6.15 pode ver-se os resultados obtidos correspondem todos a *Boas* aproximações à solução dos respectivos problemas.

Assim, todos os algoritmos implementados devolvem *Boas* aproximações às soluções dos problemas S201, S202, S205, S206 e S207, usando os parâmetros por defeito.

Problema S201	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	395	100	0.0	0.0	Boa
HJ	67	10	0.0	0.0	Boa
AA	628	100	0.0	0.0	Boa
NM	78	38	2.3760179238165266E-9	2.3760179238165266E-9	Boa
SC	119	48	0.0	0.0	Boa
Objectivo	$f(x^*) = 0$				

TABELA 6.11: Resultados para o Problema S201

Problema S202	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	338	100	0.0	0.0	Boa
HJ	73	11	0.0	0.0	Boa
AA	628	100	0.0	0.0	Boa
NM	67	32	0.0	0.0	Boa
SC	88	39	5.3770259170010006E-15	5.3770259170010006E-15	Boa
Objectivo	$f(x^*) = 0$				

TABELA 6.12: Resultados para o Problema S202

Problema S205	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	178	100	3.49482962448898E-5	3.49482962448898E-5	Boa
HJ	343	56	1.1980687983775554E-5	1.1980687983775554E-5	Boa
AA	250	94	1.2604603385420328E-8	1.2604603385420328E-8	Boa
NM	76	37	5.7909911781634624E-11	5.7909911781634624E-11	**Boa**
SC	81	41	1.2773504784835512E-10	1.2773504784835512E-10	Boa
Objectivo	$f(x^*) = 0$				

TABELA 6.13: Resultados para o Problema S205

Problema S206	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	171	52	9.336509094469523E-10	9.336509094469523E-10	**Boa**
HJ	120	17	9.336509094469523E-10	9.336509094469523E-10	**Boa**
AA	359	100	2.0822471584310544E-5	2.0822471584310544E-5	Boa
NM	85	41	1.0131413651051473E-9	1.0131413651051473E-9	Boa
SC	88	43	1.9802097000643977E-9	1.9802097000643977E-9	Boa
Objectivo	$f(x^*) = 0$				

TABELA 6.14: Resultados para o Problema S206

Na TABELA 6.16, correspondente aos resultados obtidos para o Problema S208, nenhuma das aproximações constitui uma *Boa* aproximação à solução, à excepção da obtida com o algoritmo NM, pelo que, se testaram outros valores para os passos iniciais.

Considerando $\alpha = 2.2$, tanto o algoritmo CS como o HJ devolvem *Boas* aproximações à solução, de acordo com o critério estabelecido. Com este passo inicial obtém-se, usando o algoritmo CS: $f(x_k) = 4.93038065763132E-32$, $fEvals = 213$ e $k = 54$; usando o algoritmo HJ: $f(x_k) = 1.97708264371016E-29$, $fEvals = 77$ e $k = 12$. Estabelecendo $\alpha_m = 2.2$ e $\alpha = 1.5$, o algoritmo AA devolve $f(x_k) = 4.93038065763132E-32$,

Problema S207	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	223	84	1.9790813611415002E-10	1.9790813611415002E-10	Boa
HJ	378	62	1.0351450250545388E-6	1.0351450250545388E-6	Boa
AA	340	100	1.7150063300505907E-6	1.7150063300505907E-6	Boa
NM	87	42	9.437663406249846E-12	**9.437663406249846E-12**	**Boa**
SC	196	42	7.79958958117555E-5	7.79958958117555E-5	Boa
Objectivo	$f(x^*) = 0$				

TABELA 6.15: Resultados para o Problema S207

Problema S208	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	316	100	4.358675386942925	4.358675386942925	Má
HJ	686	100	3.553297571613234	3.553297571613234	Má
AA	288	100	4.331582939666962	4.331582939666962	Má
NM	187	96	2.230971210084903E-11	**2.230971210084903E-11**	**Boa**
SC	254	47	3.633741095095621	3.633741095095621	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.16: Resultados para o Problema S208

$fEvals = 422$ e $k = 54$; se $s = 2.2$ o algoritmo obtém-se com o algoritmo SC: $f(x_k) = 1.97708264371016E-29$, $fEvals = 99$ e $k = 38$.

Pode então concluir-se que, com os parâmetros de entrada adequados, todos os Métodos de Pesquisa Directa permitem encontrar uma *Boa* aproximação à solução do Problema S208.

Problema S211	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	251	100	0.029329594939720326	0.029329594939720326	Má
HJ	240	39	0.030123556747536057	0.030123556747536057	Má
AA	324	100	0.03567795751494598	0.03567795751494598	Má
NM	135	71	5.913266176743808E-10	**5.913266176743808E-10**	**Boa**
SC	250	69	0.1846588159949892	0.1846588159949892	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.17: Resultados para o Problema S211

À semelhança do que acontece no problema anterior, os resultados apresentados na TABELA 6.17 não constituem *Boas* aproximações de soluções para o Problema S211, à excepção da obtida com o algoritmo NM, pelo que, se testaram outros valores para os passos iniciais.

Considerando $\alpha = 2.2$, tanto o algoritmo CS como o HJ devolvem *Boas* aproximações à solução, de acordo com o critério estabelecido. Com este passo inicial obtém-se, usando o algoritmo CS: $f(x_k) = 4.979684464207637E-30$, $fEvals = 209$ e $k = 53$; usando o algoritmo HJ: $f(x_k) = 4.442272972525823E-29$, $fEvals = 77$ e $k =$

12. Estabelecendo $\alpha_m = 2.2$ e $\alpha = 0.5$, o algoritmo AA devolve os valores $f(x_k) = 4.979684464207637E-30$, $fEvals = 422$ e $k = 54$; se $s = 2.2$ obtém-se com o algoritmo SC: $f(x_k) = 4.442272972525823E-29$, $fEvals = 99$ e $k = 44$.

Assim, pode concluir-se que, com os parâmetros de entrada adequados, todos os Métodos de Pesquisa Directa permitem encontrar uma *Boa* aproximação à solução do Problema S211.

Problema S212	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	397	100	0.0	0.0	Boa
HJ	66	10	0.0	0.0	Boa
AA	789	100	0.0	0.0	Boa
NM	162	82	1.4340275895963465E-20	1.4340275895963465E-20	Boa
SC	180	100	0.0	0.0	Boa
Objectivo	$f(x^*) = 0$				

TABELA 6.18: Resultados para o Problema S212

Os resultados apresentados na TABELA 6.18, para o Problema S212, correspondem todos a *Boas* aproximações à solução do respectivo problema. Assim, todos os algoritmos implementados devolvem *Boas* aproximações à solução dos problemas S212, usando os parâmetros por defeito.

Problema S213	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	397	100	0.0	0.0	Boa
HJ	66	10	0.0	0.0	Boa
AA	789	100	0.0	0.0	Boa
NM	79	39	4.848125237601807E-37	4.848125237601807E-37	Boa
SC	250	69	0.1846588159949892	0.1846588159949892	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.19: Resultados para o Problema S213

A TABELA 6.19 apresenta *Boas* aproximações à solução do problema S213, à excepção do algoritmo SC. No entanto, considerando $s = 2.2$ obtém-se uma *Boa* aproximação à solução, com: $f(x_k) = 1.7066509389413067E-6$, $fEvals = 239$ e $k = 51$.

Problema S240	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	202	100	16.75	16.75	Má
HJ	216	22	0.0	**0.0**	**Boa**
AA	202	100	20.75	20.75	Má
NM	195	100	8.259159012889974E-8	8.259159012889974E-8	Boa
SC	745	100	1058.4169741950982	1058.4169741950982	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.20: Resultados para o Problema S240

Os resultados apresentados na TABELA 6.20 não constituem todos *Boas* aproximações de soluções para o Problema B5, pelo que, à semelhança do que foi feito para os problemas anteriores, se testaram outros valores para os passos iniciais.

Tendo em conta que os os algoritmos CS, AA e SC não devolvem *Boas* aproximações à solução, para este problema, foram estes os algoritmos em que foram testados novos valores para os passos iniciais. Assim, com $\alpha = 2.2$ e com o algoritmo CS obteve-se $f(x_k) = 3.0919909391301015E-9$ ao fim de 310 avaliações da função objectivo e 100 iterações; com $\alpha_m = 2.2$, mantendo α_p por defeito, com o algoritmo AA obteve-se $f(x_k) = 1.6570196936071216E-8$ ao fim de 417 avaliações da função objectivo e 100 iterações; com $s = 2.2$.

Com o algoritmo SC e com os valores testados não foi possível encontrar uma *Boa* aproximação à solução, sendo que a melhor aproximação encontrada foi considerando o valor do passo inicial $s = 12.4$, com o qual se obteve $f(x_k) = 51.918700933095806$ ao fim de 636 avaliações da função objectivo e 100 iterações.

Problema S246	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\lvert f(x^*) - f(x_k)\rvert$	Class.
CS	268	100	0.3619683409109712	0.3619683409109712	Má
HJ	801	100	0.08784395887144672	0.08784395887144672	Má
AA	313	100	0.17535233497619626	0.17535233497619626	Má
NM	179	94	5.497384551185125E-9	5.497384551185125E-9	Boa
SC	195	98	3.4786676722717363E-9	**3.4786676722717363E-9**	**Boa**
Objectivo	$f(x^*) = 0$				

TABELA 6.21: Resultados para o Problema S246

Para o problema S246 foi possível encontrar *Boas* aproximações à solução usando os algoritmos NM e SC. As restantes aproximações, conforme se pode observar na TABELA 6.21 não constituem *Boas* aproximações de soluções para o Problema S246.

Assim, testaram-se outros valores para os passos iniciais, à semelhança do que foi feito nos problemas anteriores, não se tendo obtido grandes melhorias nem *Boas* aproximações à solução do problema, de acordo com o critério definido.

Os resultados apresentados na TABELA 6.22, para o Problema S256, constituem todos *Boas* aproximações de soluções para o Problema S256, à excepção da obtida pelo algoritmo AA. Nos testes efectuados não foi possível encontrar uma *Boa* aproximação à solução com este algoritmo, mas encontrou-se uma aproximação melhor do que a encontrada com os parâmetros por defeito. A melhor aproximação encontrada foi considerando

Problema S256	Resultados				
Métodos	$fEvals$	k	$f(x_k)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	618	100	6.773522966341261E-5	6.773522966341261E-5	Boa
HJ	642	57	8.542094139649887E-5	8.542094139649887E-5	Boa
AA	313	100	0.17535233497619626	0.17535233497619626	Má
NM	180	100	2.9791958090691317E-5	**2.9791958090691317E-5**	**Boa**
SC	162	100	7.81806787538548E-5	7.81806787538548E-5	Boa
Objectivo	$f(x^*) = 0$				

TABELA 6.22: Resultados para o Problema S256

$\alpha_m = 2.75$ e $\alpha_p = 0.7$, para os quais $f(x_k) = 7.402270156553639E-4$ ao fim de 493 avaliações da função objectivo e 100 iterações.

Problema S257	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	486	100	0.014886787725845351	0.014886787725845351	Má
HJ	1042	100	-33.62171072790248	**-33.62171072790248**	
AA	496	100	4.646929398179054	4.646929398179054	Má
NM	177	100	-33.32588405022699	**-33.32588405022699**	
SC	285	100	1.8446543314821326	1.8446543314821326	Má
Objectivo	$f(x^*) = 0$				

TABELA 6.23: Resultados para o Problema S257

No que diz respeito ao Problema S257 os resultados apresentados na TABELA 6.23 mostram que os algoritmos HJ e NM foram capazes de encontrar uma solução melhor do que a apresentada por Schittkowski.

Assim, testaram-se outros valores para os passos iniciais, sendo os melhores resultados obtidos os seguintes. Usando o algoritmo CS a função objectivo assume o valor $f(x_k) = -35.3664121517539$ com 485 avaliações, $k = 100$ e $\alpha = 2.6$; usando o algoritmo HJ obtém-se: $f(x_k) = -41.04717190250285$, $fEvals = 1033$ e $k = 99$, com $\alpha = 2.6$; o algoritmo AA devolve: $f(x_k) = -38.93973390036328$, $fEvals = 561$ e $k = 100$, com $\alpha_m = 8.5$ e $\alpha_p = 2.6$; se $s = 12.75$ obtém-se usando o algoritmo NM: $f(x_k) = -41.192991503558915$, $fEvals = 204$ e $k = 100$ e se $s = 7.5$ e usando o algoritmo SC tem-se: $f(x_k) = -35.682895336428395$, $fEvals = 498$ e $k = 100$.

6.6.2.3 Problemas de Maratos

Os resultados apresentados na TABELA 6.24 não constituem *Boas* aproximações de soluções para o Problema M500, pelo que, à semelhança do que foi feito para os problemas anteriores, se testaram outros valores para os passos iniciais. No entanto, apesar

Problema M500	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	287	100	0.9308057479653508	1,93080574796535	Má
HJ	666	100	0.9056967589057991	1,90569675890580	Má
AA	283	100	-0.5077403169847092	0,49225968301529	Má
NM	194	100	-1.0006241632854342	0,00062416328543	Média
SC	169	100	-0.21073703328211793	0,78926296671788	Má
Objectivo $f(x^*) = -1$					

Tabela 6.24: Resultados para o Problema M500

de terem sido encontradas soluções melhores, continuam a não ser *Boas* aproximações à solução.

Das alternativas testadas só se obteve uma *Boa* aproximação à solução, de acordo com o critério definido, usando o algoritmo CS, com $\alpha = 2.15$, $f(x_k) = -0.999937280906977$, tendo efectuado 303 avaliações da função objectivo e 100 iterações.

Com os restantes algoritmos as melhores aproximações à solução encontradas foram: com o algoritmo HJ, com $\alpha = 2.15$, $f(x_k) = -0.999593511652988$, tendo efectuado 622 avaliações da função objectivo e 99 iterações; com o algoritmo AA, com $\alpha_m = 2.15$ e α_p definido por defeito, $f(x_k) = -0.9886959579016861$, tendo efectuado 324 avaliações da função objectivo e 100 iterações; com o algoritmo NM, não se obteve uma aproximação à solução melhor do que a encontrada com o valor do passo por defeito; com o algoritmo SC, com $s = 2.15$, $f(x_k) = -0,972450931564772$, tendo efectuado 303 avaliações da função objectivo e 72 iterações.

Problema M501	Resultados				
Métodos	$fEvals$	k	$f(x^*)$	$\|f(x^*) - f(x_k)\|$	Class.
CS	304	100	0.974780426314383	1,97478042631438	Má
HJ	74	11	0.975035732067772	1,97503573206777	Má
AA	353	100	0.42638930925171314	1,42638930925171	Má
NM	184	100	-0.09755437373783789	0,90244562626216	Má
SC	214	83	0.4299242862201531	1,42992428622015	Má
Objectivo	$f(x^*) = -1$				

Tabela 6.25: Resultados para o Problema M501

À semelhança do que acontece no problema anterior, os resultados apresentados na Tabela 6.25 não constituem *Boas* aproximações de soluções para o Problema M501, pelo que, se testaram outros valores para os passos iniciais.

Das alternativas testadas só não se obteve uma *Boa* aproximação à solução, de acordo com o critério definido, usando o algoritmo SC, no entanto foi possível melhorar a aproximação considerando com $s = 2.15$, com o qual se obteve $f(x_k) = -0.9888650009624702$,

$fEvals = 184$ e $k = 56$.

Os restantes algoritmos devolvem *Boas* aproximações à solução, de acordo com o critério estabelecido.

Considerando $\alpha = 2.1$ obtém-se, usando o algoritmo CS: $f(x_k) = -1.0000057537325282$, $fEvals = 98$ e $k = 26$; usando o algoritmo HJ, com $\alpha = 2.2$ tem-se: $f(x_k) = -0.9999999999999964$, $fEvals = 86$ e $k = 13$; estabelecendo $\alpha_m = 2.1$ e considerando α definido por defeito, o algoritmo AA devolve $f(x_k) = -1.000004802653367$, $fEvals = 225$ e $k = 31$; se $s = 2.1$ obtém-se com o algoritmo NM: $f(x_k) = -1.0000062499065685$, $fEvals = 141$ e $k = 70$.

6.6.2.4 Análise final dos Resultados Numéricos

Tendo em conta os resultados numéricos, discutidos nas secções anteriores, é possível fazer uma análise final do desempenho dos vários algoritmos implementados.

Uma das questões a ter em conta nesta análise é a qualidade das aproximações às soluções encontradas (apresentadas no CD anexo e, sumariamente, na secção anterior).

Na TABELA 6.26 pode ver-se o número de soluções *Boas* e *Más*, de acordo com o critério definido, que foram obtidas na resolução dos 25 problemas sem restrições, usando os parâmetros por defeito; na TABELA 6.27 apresenta-se o resultado das alterações de parâmetros efectuadas, na qualidade dessas aproximações. Não se contabilizaram as aproximações à solução *Médias*, por representarem um número reduzido de aproximações (2 aproximações ao longo de toda a presentação de resultados).

Métodos	Nº de Soluções Boas	Nº de Soluções Más	Total
CS	9	16	25
HJ	9	16	25
AA	9	16	25
NM	13	12	25
SC	8	17	25

TABELA 6.26: Qualidade das soluções encontradas com os parâmetros por defeito

Nesta Tabela é possível verificar que a maioria dos métodos não devolvem soluções *Boas*, com os parâmetros por defeito. O algoritmo de Nelder-Mead foi o que permitiu resolver

Métodos	Nº de Soluções Boas	Nº de Soluções Más	Total
CS	13	3	16
HJ	7	9	16
AA	9	7	16
NM	5	7	12
SC	8	9	17

TABELA 6.27: Qualidade das soluções encontradas com os parâmetros alterados

mais problemas com os parâmetros por defeito, sendo este número superior ao número de vezes que foi necessário testar alternativas de parâmetros.

As alterações aos parâmetros só foram efectuadas se a aproximação à solução obtida não fosse, segundo o critério definido, uma *Boa* aproximação. Com estas alterações foi possível melhorar muitos resultados, conforme se pode observar na TABELA 6.27. No entanto, não foi possível estabelecer um padrão para as alterações que deviam ser efectuadas, uma vez que, como se viu na secção anterior, para problemas diferentes são mais adequados parâmetros distintos.

Analisando os valores obtidos pode mesmo dizer-se que os parâmetros com mais sucesso na resolução dos problemas foram os parâmetros por defeito, uma vez que com eles é possível resolver um número razoável de problemas, enquanto que os parâmetros alterados, permitem resolver um problema específico, mas não funcionam bem para os problemas seguintes.

Note-se que para o problema S257 foram encontradas aproximações à solução melhores do que a solução apontada pelo autor. Este problema foi contabilizado em todos os algoritmos, com alteração dos parâmetros iniciais, como conducente a *Boas* aproximações à solução.

Interessa ainda analisar, de uma forma geral, qual dos algoritmos teve melhor desempenho. Este desempenho não depende apenas do número de *Boas* aproximações às soluções que conseguiu identificar, mas também dependendo do número de avaliações das funções objectivo e iterações que foram necessárias.

Na TABELA 6.28 apresenta-se um registo do número de problemas, de entre os testados, para os quais cada um dos algoritmos encontrou o resultado mais próximo do pretendido, usando os parâmetros por defeito.

Métodos	$f(x_k)$ Melhor	$f(x_k)$ Melhor em simultâneo	Total
CS	0	5	5
HJ	2	5	7
AA	2	4	6
NM	12	1	13
SC	4	2	6

TABELA 6.28: Melhores valores para $f(x_k)$, usando os parâmetros por defeito

Existem vários problemas para os quais mais do que um algoritmo devolve o mesmo resultado (o melhor obtido) para $f(x_k)$, neste caso foram registados todos os que conduziam a esse resultado na coluna com a designação $f(x_k)$ *melhor em simultâneo*, da TABELA 6.28. Nessa tabela pode observar-se que o algoritmo de Nelder-Mead foi o que obteve mais melhores aproximações às soluções, tendo encontrado 13 melhores soluções, uma das quais também encontrada por outros algoritmos. Os restantes algoritmos permitiram encontrar as melhores aproximações obtidas num menor número de problemas, sendo de destacar que o algoritmo de Pesquisa Coordenada não foi o método mais eficaz, no que diz respeito ao valor da função objectivo, em nenhum dos problemas, tendo-o sido apenas em simultâneo com outros algoritmos.

No que diz respeito ao número de avaliações da função objectivo, o algoritmo de Nelder-Mead também é o que tem melhor desempenho, efectuando o menor número de avaliações da função objectivo em 14 problemas, sendo que 5 desses são comuns ao número de avaliações efectuadas por outros algoritmos, como se pode observar na TABELA 6.29.

Métodos	$fEvals$ Melhor	$fEvals$ Melhor em simultâneo	Total
CS	0	0	0
HJ	6	3	9
AA	0	0	0
NM	5	9	14
SC	2	0	2

TABELA 6.29: Melhores valores para $fEvals$, usando os parâmetros por defeito

Note-se que os algoritmos CS e AA efectuam, em todos os problemas, mais avaliações da função objectivo do que os restantes, portanto nunca são algoritmos com o melhor/menor valor de avaliações da função objectivo.

No que respeita ao número de iterações efectuadas, característica não tão pertinente como as anteriores, mas também aqui resgistada, como se pode ver na TABELA 6.30, o algoritmo com maior número de problemas nos quais necessitou de menos iterações é o algoritmo HJ, obtendo um total de 21 problemas com o menor valor de iterações efectuadas para encontra a aproximação à solução, sendo 7 em simultâneo com outros algoritmos.

Métodos	k Melhor	k Melhor em simultâneo	Total
CS	3	0	3
HJ	14	7	21
AA	3	0	3
NM	2	4	6
SC	4	1	5

TABELA 6.30: Melhores valores para k, usando os parâmetros por defeito

Em alguns dos problemas os resultados apresentados nas tabelas anteriores foram verificados em simultâneo, ou seja, um algoritmo devolveu o melhor valor de $f(x_k)$, o menor número de avaliações da função objectivo e o menor número de iterações necessárias. Isto só aconteceu com os algoritmos HJ e NM, como se pode observar na TABELA 6.31.

Métodos	Acumula os 3 anteriores
CS	0
HJ	3
AA	0
NM	3
SC	0

TABELA 6.31: Melhores valores para $f(x_k)$, $fEvals$ k em simultâneo

Tendo em conta estes resultados é de prever que os algoritmos de Nelder-Mead e de Hooke-Jeeves sejam os que permitem encontra melhores aproximações à solução, no entanto, também se pode ver que em alguns problemas os outros algoritmos são melhor sucedidos. Deste modo não se pode indicar o melhor algoritmo para todos os problemas, pelo que, na presença de um problema que possa ser resolvido usando estes métodos/algoritmos será de testar todos os algoritmos escolhendo a aproximação à solução que melhor se adeqúe à situação em questão.

Ainda se pode concluir da análise destes resultados, que estes algoritmos, usando apenas informação sobre o valor da função objectivo em diversos pontos e progredindo por comparação destes valores, nem sempre permitem encontrar *Boas* aproximações à soluçaõ, no entanto a alteração de parâmetros, nomeadamente do tamanho do passo inicial, permitiu melhorar muitas dessas aproximações, pelo que também será uma boa opção testar diferentes valores para os parâmetros iniciais.

Capítulo 7

Pormenores de Implementação dos Algoritmos para Optimização com Restrições

Neste trabalho foram implementados, usando máquinas de Estados Finitos, dois tipos de Métodos para Optimização com Restrições:

1. Métodos de Penalidade e Barreira – PenBar (Secção 7.1);
2. Método dos Filtros – MF (Secção 7.2);

Pode chamar-se aos métodos implementados *Métodos de Pesquisa Directa para Optimização Com Restrições* uma vez que não usam derivadas, nem aproximações de derivadas nem modelos para aproximar as funções envolvidas.

Tal como foi explicado no Capítulo 5 não foi implementado o Método de Lagrangeana Aumentada por não se terem implementado Métodos de Optimização Sem Restrições para problemas com limites simples, ou seja, Métodos de Pesquisa Directa Generalizados.

À semelhança do que aconteceu com os algoritmos de optimização sem restrições (Capítulo 6), estes algoritmos são compostos por diversos estados, pelo que se optou, mais uma vez, por uma representação dos algoritmos através de fluxogramas, salientando-se através de rectângulos os estados.

7.1 Métodos de Penalidade e Barreira

Os Métodos de Penalidade e Barreira, tal como foi apresentado na FIGURA 3.1, são constituídos por dois processos:

- Processo externo – onde é criada uma sequência de problemas sem restrições;
- Processo interno – onde são resolvidos os problemas sem restrições.

com o objectivo resolver o problema com restrições P apresentado em (1.2).

Assim, a diferença destes métodos dos de resolução de problemas sem restrições está no processo externo, na construção da função Φ e na actualização de parâmetros de penalidade/barreira.

Dos Métodos de Penalidade e Barreira, apresentados na secção 3.2, foram implementados todos à excepção dos Métodos de Barreira Inversa (3.6 e 3.7) e Barreira Logarítmica (3.8 e 3.9), uma vez que existem alguns pontos para os quais as suas funções barreira não estão definidas, nomeadamente, no caso da Barreira Inversa quando o valor de uma dada restrição nesse ponto é zero e no caso da Barreira Logarítmica quando o valor de uma dada restrição nesse ponto é zero ou positivo e do Método de Penalidade Adaptativa (3.19 e 3.20) por ser similar ao Método de Penalidade Dinâmica (3.15, 3.16, 3.17 e 3.18).

Assim, os Métodos de Penalidade/Barreira implementadas neste trabalho são:

1. Barreira Extrema (EB) (3.4);
2. Barreira Progressiva (PB) (3.5);
3. Penalidade Clássica (CP) (3.12);
4. Penalidade Estática/Dinâmica (SP) (3.16);
5. Penalidade ℓ_1 (ℓ_1) (3.16, 3.17 e 3.18).

Nestes métodos é construída uma nova função objectivo, Φ, usando informação sobre a função objectivo inicial, f, e as restrições do problema $c_i,\ i = 1, ..., m$. Assim, é criada uma sucessão de Problemas sem Restrições que dependem de um parâmetro positivo

r_k (ou vector de parâmetros positivos), dito parâmetro de Penalidade/Barreira, cujas soluções convergem para a solução do problema inicial (Processo Externo).

Estes problemas sem restrições são então resolvidos usando Métodos de Pesquisa Directa (Processo Interno), onde o problema a ser resolvido, em cada iteração k, pode ser representado genericamente por:

$$\min_{x_k \in \mathbb{R}^n} \Phi(x_k, r_k),$$

com

$$\Phi(x_k, r_k) = f(x_k) + r_k p(x)$$

e p uma função que penaliza (Penalidade) ou recusa (Barreira) pontos que violem as restrições. Deste modo, nestes métodos, a optimalidade e a admissibilidade são tratadas em conjunto.

O algoritmo de Penalidade/Barreira aqui implementado, apresentado genericamente na FIGURA 3.1, é constituído por 9 estados e é apresentado no fluxograma da FIGURA 7.1.

Para definir a sucessão de funções Φ, a optimizar no processo interno, foram, então, implementados cinco métodos de Penalidade/Barreira distintos, apresentados no Capítulo 3, cujas funções a optimizar no Processo Interno se apresentam resumidamente na TABELA 7.1.

Método	Função Φ a optimizar no Processo Interno
Barreira Extrema	$\Phi(x) = b_\Omega(x) = \begin{cases} f(x) & \text{se} \quad x \in \Omega \\ +\infty & \text{se} \quad x \notin \Omega \end{cases}$ onde Ω é a região admissível
Barreira Progressiva	$\Phi(x) = f(x) + b_X(x) = f(x) + \begin{cases} \sum_{i=1}^{m} (\max(c_i(x), 0))^2 & se \quad x \in X \\ +\infty & se \quad x \notin X \end{cases}$ onde $X \subset \Omega$ é o conjunto definido pelas restrições de igualdade
Penalidade Clássica	$\Phi(x) = f(x) + p(x) = f(x) + r_k \sum_{i=1}^{m} [\max\{0, c_i(x)\}]^q, \quad q \geq 1$
Penalidade Estática	$\Phi(x) = L_k(x, \alpha, \beta) = f(x) + \sum_{i=1}^{t} \alpha_i \lvert c_i(x)\rvert^q + \sum_{i=t+1}^{m} \beta_i [\max(0, c_i(x)]^q, \quad q \geq 1$
	Actualiza os parâmetros de forma estática: parametro $= \{\gamma.\alpha_1, ..., \gamma.\alpha_t, \gamma.\beta_1, ..., \gamma.\beta_{m-t}\}, \quad \gamma > 1$
Penalidade Dinâmica	Actualiza os parâmetros de forma dinâmica (neste caso tem que se ter $q > 1$): $C = \gamma(\mathrm{C} = \gamma > 0)$ $\alpha_i(k+1) = \alpha_i(k) + C.\lvert c_i(x)\rvert, i = 1, ..., t$ $\beta_i(k+1) = \beta_i(k) + C.[\max(0, c_i(x))], i = t+1, ..., m$
Penalidade ℓ_1	$\Phi(x) = \ell_1(x, \mu) = f(x) + \mu \sum_{i=1}^{t} \lvert c_i(x)\rvert + \mu \sum_{i=t+1}^{m} \max[c_i(x), 0], \mu \to +\infty$

TABELA 7.1: Função a optimizar no Processo Interno

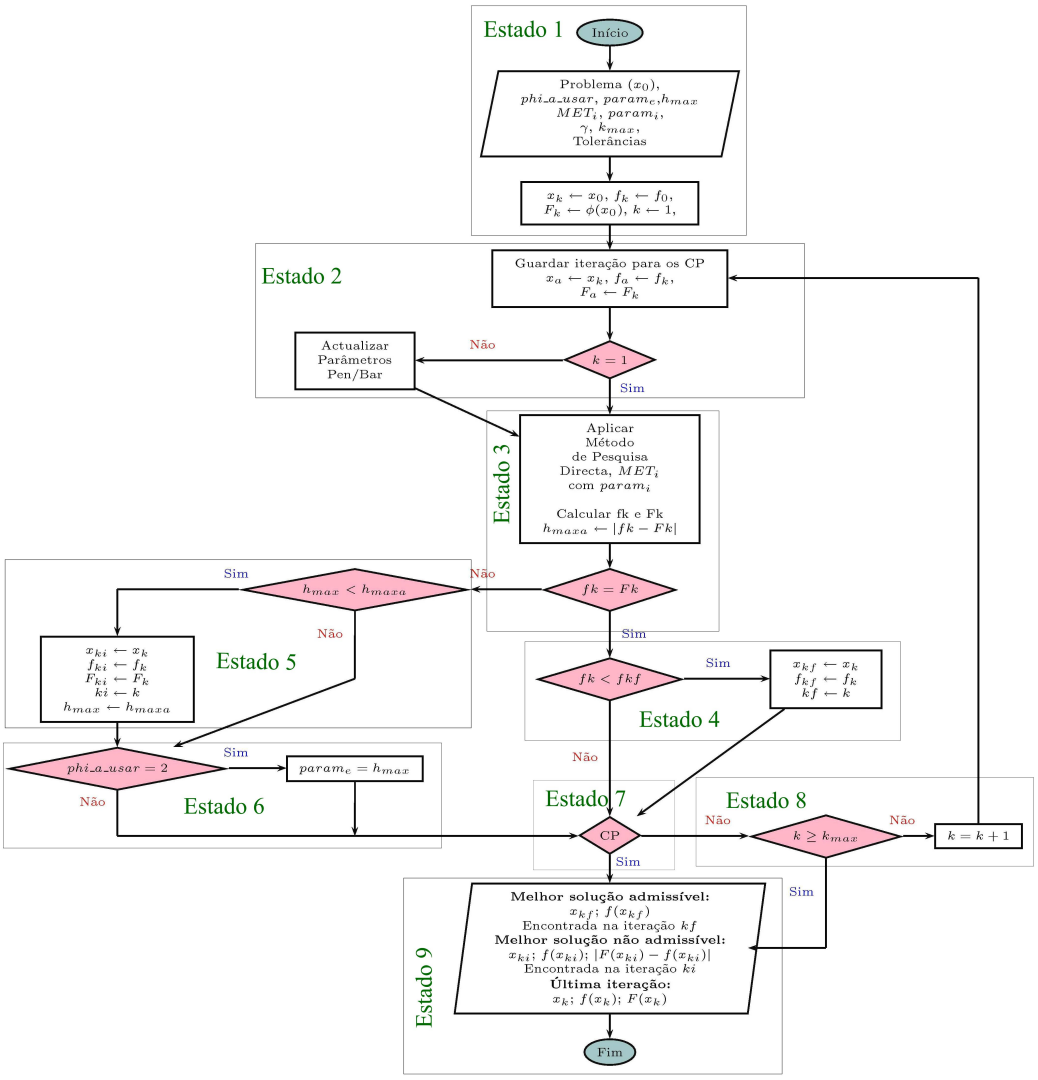

FIGURA 7.1: Algoritmo de Penalidade/Barreira Implementado

Para os Métodos de Barreira implementados não é necessário definir parâmetros de barreira, mas os Métodos de Penalidade têm o grande inconveniente de ser necessário, na maioria dos casos, definir à partida um número grande de parâmetros de penalidade para iniciar o processo. Esta escolha de parâmetros não tem regras pré-definidas, sendo muitas vezes efectuada empiricamente. Além disso, uma escolha de parâmetros pode ser adequada para um problema e não o ser no problema seguinte. Deste modo os métodos aqui implementados usam por defeito valores de parâmetros similares para todos os métodos, não se fazendo um estudo exaustivo da adequabilidade dos parâmetros, no entanto permitem que o utilizador os defina. Assim a API está desenhada para que este possa ser um dos trabalhos a ser realizado no futuro. Na TABELA 7.2, terceira coluna,

apresentam-se os parâmetros iniciais utilizados por defeito nesta implementação, no entanto o utilizador pode alterá-los. O valor para o factor de actualização dos parâmetros usado foi $\gamma = 2$, que também pode ser alterado.

Método	**Parâmetro**	**Parâmetro Inicial**	**Actualização**
Barreira Extrema	*	*	*
Barreira Progressiva	h_{max}	$h_{max} = +\infty$	$\Phi(x_k) < h_{\max} \Rightarrow h_{\max} = \Phi(x_k)$
Penalidade Clássica	r_k	1	$\underbrace{\gamma.r_k}_{r_{k+1}}, \gamma > 1$
Penalidade Estática	$[\alpha_1, ..., \alpha_t, \beta_1, ..., \beta_{m-t}]$	$\underbrace{[1, 1, ..., 1]}_{m \text{ components}}$	$[\gamma.\alpha_1, ..., \gamma.\alpha_t, \gamma.\beta_1, ..., \gamma.\beta_{m-t}]$
Penalidade Dinâmica	$[\alpha_1, ..., \alpha_t, \beta_1, ..., \beta_{m-t}]$	$\underbrace{[1, 1, ..., 1]}_{m \text{ components}}$	Ver Tabela 7.1
Penalidade ℓ_1	μ	1	$\gamma.\mu$
$* \rightarrow$ Não são utilizados quaisquer parâmetros neste método			

Tabela 7.2: Parâmetros usados no processo Externo

Os critérios de paragem utilizados no processo interno foram os definidos para os algoritmos de resolução de problemas sem restrições, descritos no Capítulo 6. Para o processo externo foram utilizados, três critérios de paragem do processo iterativo:

- Dois deles têm que ser verificados simultaneamente:
  - A condição de proximidade (1.24) (ou (1.26) para problemas que convergem para soluções nulas) $\rightarrow$ $|x_k - x_{k+1}| < T1 = 10^{-5}$;
  - A diferença entre os valores da função Φ em duas iterações sucessivas $\rightarrow$ $|\Phi(x_k) - \Phi(x_{k+1})| < T2 = 10^{-5}$;
- O terceiro critério controla o número máximo de iterações k_{max} do processo iterativo externo, tendo-se considerado $k_{max} = 40$.

Assim, o algoritmo implementado permite que o utilizador defina os seguintes parâmetros de entrada:

- Problema a resolver (função objectivo, restrições);
- Ponto inicial;
- A função de penalidade/barreira a usar (número inteiro de 1 a 6, correspondente aos métodos de Penalidade/Barreira definidos acima e de acordo com a ordem apresentada) – phi_a_usar;

- Parâmetros de penalidade/barreira iniciais (de acordo com a TABELA 7.2);
- Número máximo de iterações no processo externo – k_{max};
- Tolerância para a distância entre 2 iterações – $T1$;
- Tolerância para a distância entre 2 valores da função objectivo entre iterações consecutivas – $T2$;
- Método a usar no processo interno (número inteiro de 1 a 5, correspondente aos métodos de Pesquisa Directa, apresentados no Capítulo 6 e de acordo com a ordem apresentada) – MET_i;
- Parâmetros do processo interno (todos os que foram apresentados no Capítulo 6);
- Valor máximo da violação das restrições – h_{max};
- Factor de actualização dos Parâmetros de penalidade/barreira – γ (apresentado na TABELA 7.2);

Os algoritmos de Penalidade/Barreira devolvem genericamente os seguintes resultados:

- Número de iterações efectuadas no processo externo – k;
- Número de vezes que foi calculada o valor da função penalidade/barreira – $nEvals$;
- Última iteração efectuada (aproximação à solução encontrada) – $x^* = x_k$;
- Valor da função penalidade/barreira na última iteração efectuada – $\Phi(x_k)$;
- Valor da função objectivo na última iteração efectuada – $f(x_k)$.

No entanto, tal como foi referido no Capítulo 5, verificou-se que no Método dos Filtros é possível encontrar um conjunto de soluções (Filtro) que possuem entre elas a propriedade de não dominância. Esta característica tem a vantagem de permitir ao decisor escolher a solução que melhor se adeqúe a uma situação real, do conjunto de soluções possíveis.

Assim, no sentido de ampliar o tipo de soluções encontradas pelos Métodos de Penalidade/Barreira e com a motivação das vantagens oferecidas pela existência de diversas opções de solução do Método dos Filtros, considerou-se que, na impossibilidade de usar

o conceito de dominância, devido às especificidades destes métodos, seria de todo o interesse guardar aproximações encontradas no processo iterativo, que não sendo as melhores no que diz respeito ao valor da função barreira/penalidade, tivessem valores com interesse (com o menor valor possível da função objectivo e da violação das restrições).

Deste modo, foram acrescentados procedimentos nos Métodos de Penalidade/Barreira implementados, para devolver, ainda informação sobre:

- A melhor aproximação à solução admissível encontrada (se existir) – x_{kf} – esta aproximação é, das aproximações admissíveis encontradas no processo iterativo ($|\Phi(x_{ki}) - f(x_{ki})| = 0$), a que tem menor valor da função objectivo, ou seja, a que tem melhor valor de f;
- O valor da função objectivo na melhor aproximação à solução admissível encontrada – $f(x_{kf})$;
- A melhor aproximação à solução não admissível encontrada (se existir) – x_{ki} – esta aproximação é, das aproximações não admissíveis encontradas no processo iterativo ($|\Phi(x_{ki}) - f(x_{ki})| \neq 0$), a que tem menor valor da violação das restrições, ou seja, a que tem menor valor de $|\Phi(x_{ki}) - f(x_{ki})|$;
- O valor da função objectivo na melhor aproximação à solução não admissível encontrada – $f(x_{ki})$;
- O valor da violação das restrições na melhor aproximação à solução não admissível – $|\Phi(x_{ki}) - f(x_{ki})|$.

Deste modo, se forem encontradas iterações nestas condições, ficam disponíveis para o utilizador três soluções, para que escolha a que melhor se adequa à sua realidade. Esta alteração ainda permite que, havendo a exigência da solução ser admissível, se a última iteração não o for (apesar de ter o melhor valor da Função Penalidade/Barreira) estará disponível uma solução admissível, neste caso a melhor encontrada no processo iterativo.

Note-se que a possibilidade de identificação das melhores aproximações à solução admissível e não admissível nos métodos de Penalidade/Barreira, que neste trabalho fazem parte dos dados de saída disponíveis e apresentados na Tabelas de resultados da secção 7.3, não são dados de saída usuais nestes algoritmos.

	Prob. C801		Prob. C802	
Initial Point infeasible	$(0.1, -0.1)$	$f_0 = 160.87$	$(0.1, -0.1)$	$f_0 = 295.1$
Numerical Results	**SFA**	**PA**	**SFA**	**PA**
Number of evaluations of h/Φ	101	5516	101	4922
Number of evaluations of f/Φ	110	5516	115	4922
Last Successful iteration	4	100	34	100
Best feasible solution -x_{kf}	$(5.35, 1.15)$	Not found	$(2.6, 1.4)$	Not found
Objective value - $f(x_{kf})$	8.9	Not found	87.2	Not found
Best infeasible solution x_{ki}	$(3.1, -0.1)$	$(4.87, 1.57)$	$(0, 0)$	$(5.0, 3.6)$
Violation Value - $h(x_{ki})$	0.1	24.62	1.01	66.5
Objective Value - $f(x_{ki})$	38.47	6.02	295.1	-29.12
Penalty Function Value - $\Phi(x_{ki})$	-	3.12	-	8.44
Objective Value - $f(x^*)$	7.563	7.563	-97.31	-97.31

SFA - Simplex Filter Algorithm; **PA** - Penalty Algorithm

TABELA 7.3: Resultados da primeira versão implementada dos algoritmos de Penalidade/Barreira. (Fonte: Correia et. al. em [39])

Aliás, quando se iniciou a sua implementação, à semelhança do que faziam outros autores, estes dados não eram incluídos nas tabelas de resultados. Veja-se por exemplo, a TABELA 7.3 que foi apresentada em [39], onde se compara o desempenho do SFA (*Simplex Filter Algotithm*), uma das primeiras versões do algoritmo dos Filtros implementada, já com algumas alterações em relação à primeira versão apresentada em [38], com o algoritmo de Penalidade ℓ_1 (PA) que estava implementado na altura.

Nesta implementação o algoritmo de Penalidade não devolvia as melhores aproximações à solução, devolvia apenas a última iteração. Por esse motivo, por exemplo para o Problema C801, aparece a designação *Not found* na *Best Feasible Solution* para o algoritmo PA, porque a última iteração correspondia a uma aproximação não admissível.

Esta identificação da admissibilidade da aproximação à solução era determinada calculando apenas no final do processo o valor da diferença $|\Phi(x_k) - f(x_k)|$, com x_k a última aproximação encontrada. Se a diferença fosse nula considerava-se a aproximação admissível, senão seria não admissível, não sendo guardada nem identificada mais nenhuma aproximação obtida no processo.

7.2 Método dos Filtros

O Método dos Filtros, que foi apresentado genericamente na secção 3.4 surgiu, por Fletcher e Leyffer, em 2002, com a motivação de evitar a definição ou escolha à partida dos parâmetros de penalidade/barreira e tem sido utilizado nas mais diversas áreas da Optimização. No entanto em Pesquisa Directa só tinha sido utilizado pelo grupo de trabalho de Charles Audet. Assim, o interesse de estudar a aplicabilidade deste método em Pesquisa Directa foi considerado de extrema importância e, tal como se disse no Capítulo 5, mostrou-se ser um dos assuntos centrais deste trabalho, não só por ainda não ter sido implementado em conjugação com outros Métodos de Pesquisa Directa, mas também porque algumas das suas características serviram de motivação para adaptações que foram efectuadas nos algoritmos de Penalidade/Barreira.

Os algoritmos apresentados pelo grupo de trabalho de Charles Audet (Tabelas 3.2 e 3.3) apenas definem os procedimentos de forma genérica, não especificando claramente os passos de implementação. Assim, foi necessário estudar a melhor forma de fazer esta implementação, pelo que o trabalho desenvolvido passou por diversas fases, tendo sido implementadas diferentes versões.

A primeira versão testada deste método foi exposta no artigo de Correia et. al. [38]. Neste artigo foi implementado o Método dos Filtros em combinação com o método de Hooke e Jeeves, um método de pesquisa em padrão, à semelhança do que fizeram Audet e Dennis em [8], e em combinação com o método de Nelder-Mead. Estas duas combinações foram analisadas e comparadas concluindo-se que o segundo, que se denominou de Algoritmo Filtro Simplex (AFS), pode ser considerado como uma alternativa para a resolução de problemas não lineares com restrições, tendo-se obtido resultados bastante satisfatórios.

Esta primeira versão do algoritmo, para pontos admissíveis do simplex em pesquisa, testava imediatamente a entrada para o Filtro enquanto para pontos não admissíveis aplicava apenas três das operações do método de Nelder-Mead (Reflexão, Contracção e Expansão) e só se não houvesse nenhum ponto/iteração bem sucedida (aceite no Filtro) é que o passo encolhido ou redução do simplex era aplicado, voltando-se à pesquisa do simplex.

De uma forma geral o processo desenvolvia-se em 5 etapas principais:

1. Construção de um Filtro inicial com o ponto inicial;
2. Construção de um simplex inicial, a partir do ponto inicial;
3. Pesquisa do simplex, ponto por ponto:
   - Se o ponto fosse admissível era testada a sua entrada para o Filtro;
   - Senão eram implementadas as 3 operações do método de Nelder-Mead, pela ordem usual, e era testada a entrada do ponto obtido no Filtro;
4. Se o ponto fosse aceite, a iteração era considerada bem sucedida e o processo era repetido até se verificar o critério de paragem, senão implementava-se a redução do simplex e voltava-se à fase de pesquisa do simplex.
5. A aproximação à solução apresentada consistia no ponto admissível com menor valor da função objectivo encontrado no processo.

Esta metodologia tinha o inconveniente de só aceitar no Filtro pontos admissíveis, pelo que o processo foi alterado, tendo-se em conta a ideia inicial do método que consiste em aceitar no Filtro pontos não dominados.

Tendo em vista este objectivo desenvolveu-se um novo esquema de implementação cujo correspondente algoritmo foi designado por NSFA (*New Simplex Filter Algorithm*).

As alterações principais deste algoritmo, em relação ao SFA, são:

- O Filtro aceita pontos não admissíveis desde que verifiquem a condição de não dominância;
- Aplica-se o método de Nelder-Mead (as quatro operações do método) a h (definida em (3.35)) no caso do ponto considerado na pesquisa do Simplex não ser admissível, tendo em vista a *admissibilidade*;
- Aplica-se o método de Nelder-Mead (as quatro operações do método) a f (função objectivo), no caso do ponto ser admissível, com o objectivo da *optimalidade*.

As várias experiências efectuadas tendo por base estas alterações foram apresentadas em diversos trabalhos: [37, 39, 111].

Em [111] foi feita uma comparação do NSFA com a primeira versão do mesmo método, SFA [38]. Este trabalho foi depois actualizado e melhorado em [41].

Em [39] apresentou-se o NSFA em comparação com a função de penalidade exacta l_1 (Algoritmo da TABELA 3.1) e em [37] fez-se uma breve apresentação do que já tinha sido desenvolvido até ao momento, no âmbito deste trabalho.

Nestas implementações ainda foi criada uma nova classe na API exclusivamente dedicada à aceitação de uma nova iteração no Filtro. Esta aceitação depende de um conjunto de critérios, baseados nas indicações dos trabalhos de Audet, que podem ser observados no fluxograma da FIGURA 7.2.

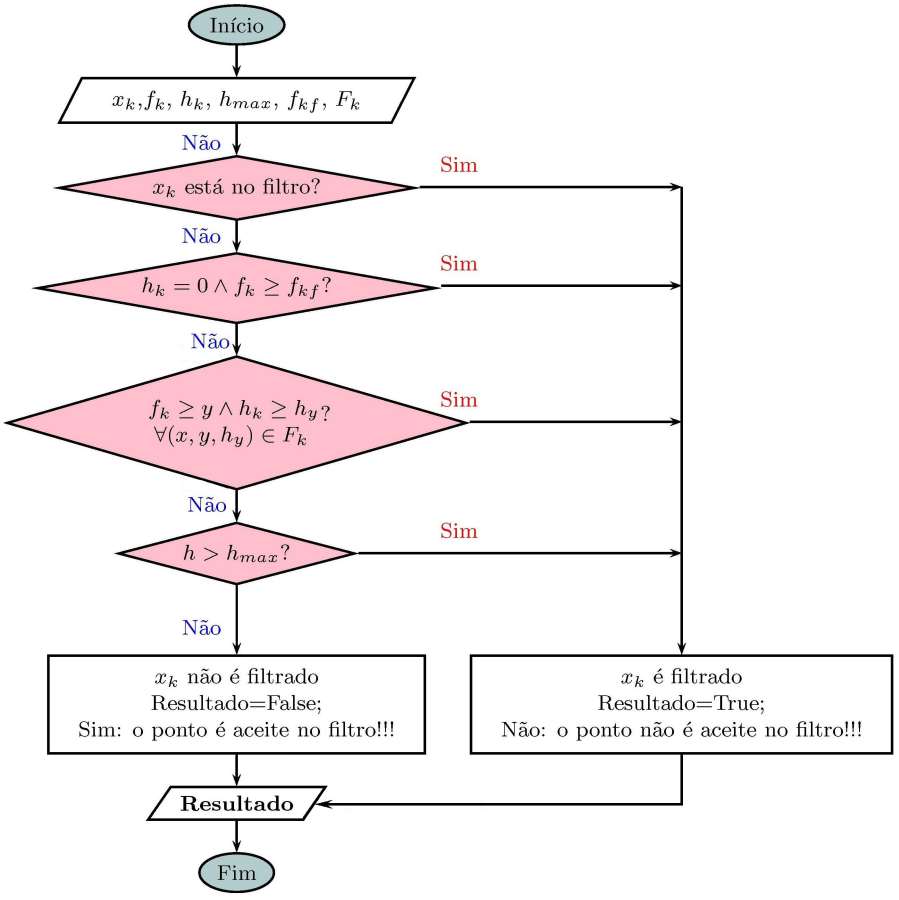

FIGURA 7.2: Critérios de Aceitação no Filtro

Assim, enquanto o algoritmo inicial só permitia encontrar soluções admissíveis, porque só admitia no Filtro pontos admissíveis, com este novo algoritmo foi possível identificar todas as soluções não dominadas, incluindo-as no Filtro.

O resultado ou dado de saída principal do Método dos Filtros é usualmente o Filtro, ou seja, o conjunto de aproximações à solução, encontradas no processo iterativo, que possuem a propriedade de não dominância entre elas, pelo que a identificação da admissibilidade das aproximações à solução, tal como foi feita neste trabalho, também não é um procedimento usual neste método.

O mais usual é a identificação da melhor aproximação admissível, não se dando importância a soluções não admissíveis, no entanto, pelas razões que já foram apresentadas anteriormente a melhor aproximação não admissível, no algoritmo dos filtros implementado, também faz parte dos resultados ou dados de saída do algoritmo.

Assim, à semelhanças do que se fez para o algoritmo de Penalidade/Barreira, foram implementados procedimentos para identificar a melhor aproximação à solução admissível x_{kf}, ou seja, o ponto encontrado no processo iterativo que não viola as restrições e tem melhor (menor) valor da função objectivo, como também a melhor aproximação à solução não admissível x_{ki}, ou seja, o ponto encontrado no processo iterativo, não admissível, com menor valor de violação das restrições. Esta segunda solução pode ser importante no caso de problemas relaxáveis, nos quais é possível aceitar uma solução cujo valor da violação não ultrapassa um dado valor máximo, de violação estipulado h_{max}, ou seja, o algoritmo guarda das iterações efectuadas, além do conjunto de pontos não dominados, os que possuem melhores valores para f e h.

Deste modo, além de todos os outros pontos que constituem o Filtro, são apresentados no final do processo os pontos referidos. Esta alteração teve a motivação de guardar, de entre os várias pontos testados no processo iterativo, os mais significativos, no que diz respeito à admissibilidade e optimalidade.

O funcionamento geral adoptado, e que constitui a versão final do Método dos Filtros implementado, pode ser ilustrado pela FIGURA 7.3.

O NSFA começa com um filtro inicial que contém apenas o ponto inicial. Depois é construído um Simplex Inicial, S_k, que contém $n+1$ vértices, a partir deste ponto, isto é:

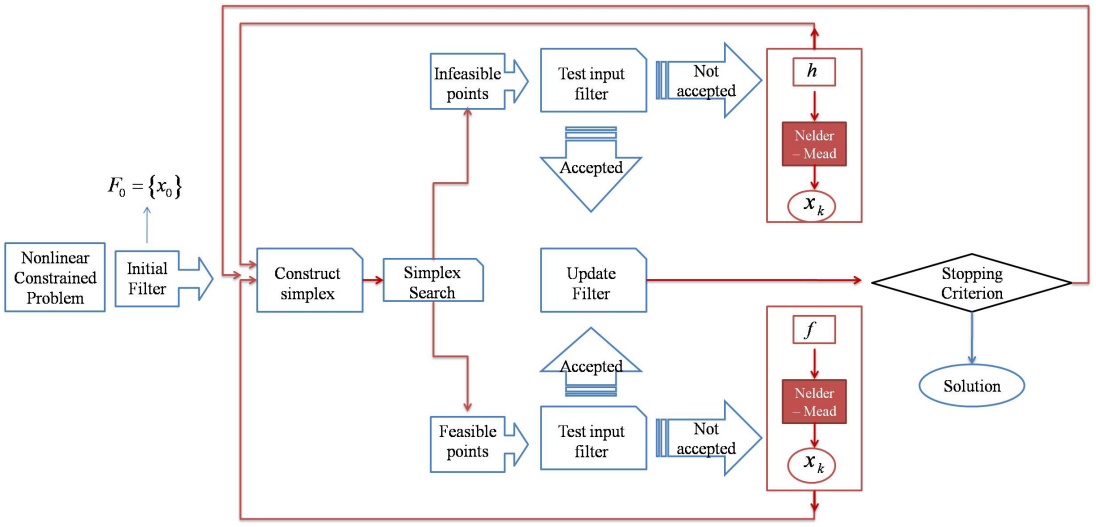

FIGURA 7.3: Funcionamento Geral do Método dos Filtros Implementado em [111]

$$S_0 = \{x_0 = v_0\} \cup \{v_i : v_i = x_0 + e_i, i = 1, ..., n\}$$
$$\vdots$$
$$S_k = \{x_k\} \cup \{v_i : v_i = x_k + e_i, i = 1, ..., n\}$$

onde $e_i, i = 1, ..., n$ representam os vectores da base canónica em $\mathbb{R}^n$, iniciando a Pesquisa no Simplex para $i = 0, ..., n$.

Se o vértice, que está a ser analisado, é um ponto admissível é testada a entrada no Filtro (de acordo com a Definição 3.8):

- Se o ponto não é aceite no Filtro:
  - O método de Nelder-Mead Method é aplicado à função f;
  - Um novo ponto é obtido;
  - Voltando-se à construção do simplex, a partir deste ponto.
- Se o ponto é aceite no Filtro:
  - O Filtro é actualizado e esse ponto é considerado a nova aproximação à solução, ou seja, a nova iteração;
  - Se o critério de paragem é verificado, esta aproximação é a aproximação à solução, senão volta-se à construção do simplex, a partir deste ponto.

Se o vértice, que está a ser analisado, é um ponto não admissível é testada a entrada no Filtro:

- Se o ponto não é aceite no Filtro:
  - O método de Nelder-Mead é aplicado à função h;
  - Um novo ponto é obtido;
  - Voltando-se à construção do simplex, a partir deste ponto.
- Se o ponto é aceite no Filtro:
  - O Filtro é actualizado e esse ponto é considerado a nova aproximação à solução, ou seja, a nova iteração;
  - Se o critério de paragem é verificado, esta aproximação é a aproximação à solução, senão volta-se à construção do simplex, a partir deste ponto.

Os testes numéricos nos trabalhos correspondentes a esta implementação, Correia et. al. [39, 41, 111], mostraram que esta metodologia é bastante eficaz, tanto em comparação com o algoritmo anterior, como em comparação com a implementação semelhante, mas usando o método de Hooke e Jeeves, bem como em comparação com a função de penalidade exacta l_1. A norma usada nestes trabalhos, para definir h foi a norma 2, apresentada em (3.36).

	Prob. C801		Prob. C802	
Infeasible initial Point	$(0.1, -0.1)^T$		$(0.1, -0.1)^T$	
Numerical Results	**SFA**	**NSFA**	**SFA**	**NSFA**
Number of evaluations of h	1226	41	1226	55
Number of evaluations of f	1526	41	1526	41
Unsuccessful iterations	39	24	40	23
Successful iterations	1	15	0	16
Last successful iteration	1.st	39.st	0	39.st
Best feasible solution -x_{kf}	$(0.1; -0.09)^T$	$(5.1; 0.9)^T$	$(0.1, -0.1)^T$	$(2.5, 1.4)^T$
Function value - $f(x_{kf})$	160.87	9.67	295.1	87.2
Best infeasible solution x_{ki}	*	$(3.1; -0.1)^T$	*	$(0, 0)^T$
Violation value - $h(x_{ki})$	*	0.1	*	1.01
Function value - $f(x_{ki})$	*	38.47	*	295.1

SFA - Simplex Filter Algorithm, (Correia *et al.*, 2009); **NSFA** - New Simplex Filter Algorithm
* Algorithm rejects infeasible solutions
Successful Iterations - Iterations accepted in the Filter

TABELA 7.4: Resultados da versão NSFA em comparação com a anterior SFA (Fonte: Correia et. al. em [41])

Pode ver-se por exemplo a FIGURA 7.4 onde os resultados foram obtidos para os problemas C801 e C802 (apresentados em [41]) mostram uma clara melhoria em relação à primeira versão que tinha sido implementada em [38]. Esta melhoria fica bem evidenciada na representação gráfica (apresentada na FIGURA 7.4) de alguns pontos dos processos iterativos correspondentes. Efectivamente o NSFA dá prioridade à admissibilidade, uma vez que as aproximações obtidas no processo iterativo estão mais próximas ou no interior da região admissível, apresentada na figura a cinza, do que a anterior versão do algoritmo.

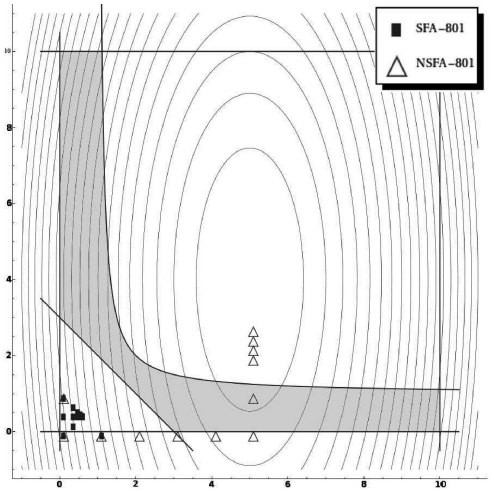

Fig. 4. Problem C-801 with infeasible initial points.

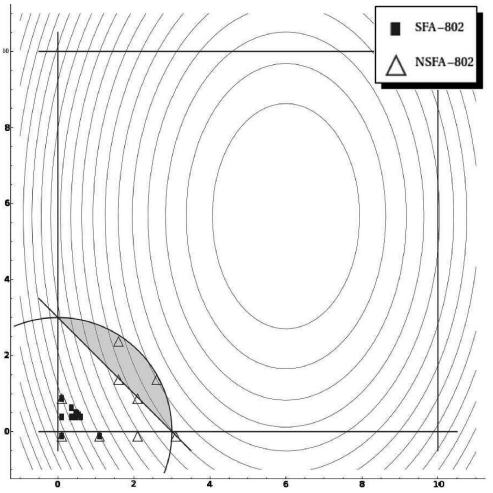

Fig. 5. Problem C-802 with infeasible initial points.

FIGURA 7.4: Representação gráfica de alguns pontos
dos processos iterativos SFA e NSFA.
(Fonte: Correia et. al. em [41])

Estes resultados motivaram a implementação do Método dos Filtros de forma genérica, ou seja, de forma a poder ser aplicado não só o Método de Nelder-Mead na optimização de h e a f, como todos os métodos de Optimização Sem Restrições apresentados no Capítulo 6, aliás tal como se fez nos algoritmos de Penalidade/Barreira implementados.

Assim, o esquema de implementação apresentado na FIGURA 7.3 foi actualizado, sendo a versão implementada neste trabalho a apresentada na FIGURA 7.5.

As alterações efectuadas são simplesmente adaptações e generalizações, pois o funcionamento é similar. As adaptações dizem respeito às designações *Conjunto inicial de pesquisa* e *Pesquisa do Conjunto* que constituem apenas uma adaptação das designações *Simplex inicial* e *Pesquisa do Simplex*, uma vez que, tanto a construção como a pesquisa é feita usando o mesmo procedimento utilizado no algoritmo anterior. As

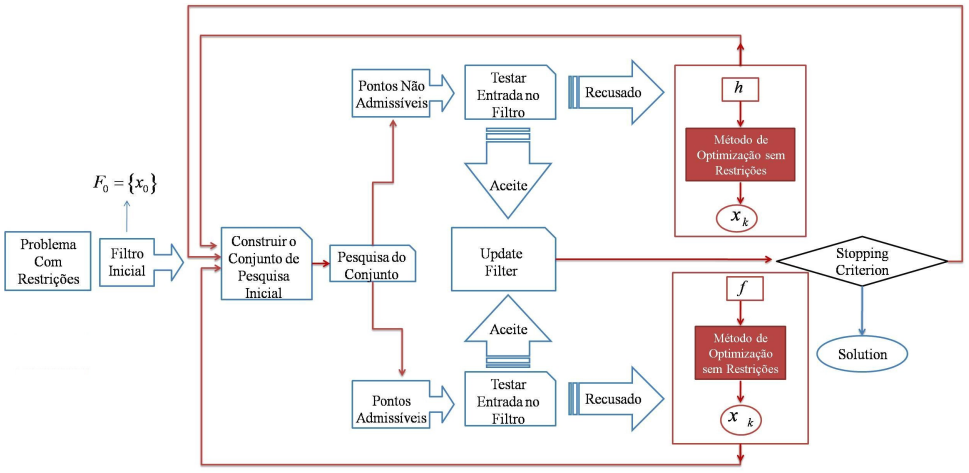

FIGURA 7.5: Funcionamento Geral do Método dos Filtros Generalizado Implementado

generalizações dizem respeito ao processo interno, ou seja, ao processo de optimização sem restrições, que no algoritmo anterior era feito usando exclusivamente o algoritmo de Nelder-Mead e nesta implementação é possível usar qualquer um dos cinco algoritmos de optimização sem restrições implementados. Note-se que o Método dos Filtros só foi usado desta forma, como Método de Pesquisa Directa, por Audet et. al., mas utilizando apenas Métodos de Pesquisa em Padrão.

Na fase final deste trabalho ainda foi feita uma alteração à forma como o número de iterações é contabilizado. Nas versões apresentadas anteriormente só eram contabilizadas as iterações se houvesse lugar à aceitação do ponto no Filtro, ou seja, iterações com sucesso, no entanto considerou-se que as iteração sem sucesso também deviam ser contabilizadas.

Esta alteração pode aumentar o inconveniente de, se o número máximo de iterações for pequeno, o número de iterações permitidas poder ser atingido sem que se proceda à efectiva optimização de f e/ou h. Isto pode acontecer se na *Pesquisa do Conjunto* os pontos forem sendo aceites no Filtro, nesse caso a iteração é contada e volta-se à construção do simplex a partir do mesmo ponto, repetindo a partir desse ponto o processo de construção do novo conjunto de pesquisa e o processo de pesquisa, portanto nunca se optimiza f nem h.

Uma forma de evitar este inconveniente, principalmente quando o seu desempenho é comparado com outro método, por exemplo, com os métodos de Penalidade/Barreira, é usar neste método um número de iterações superior. Esta diferença na maioria das vezes não constitui, em termos de número de avaliações da função objectivo e da violação das

restrições, um aumento efectivo do custo do processo, uma vez que parte dele se desenrola sem optimizar f e h (ou seja, com um número reduzido de avaliações), enquanto que todas as iterações efectuadas nos métodos de Penalidade/Barreira são constituídas (no processo interno) pela optimização da função Penalidade/Barreira, que agrega f e a violação das restrições e consequentemente exige a avaliação destas funções em todas as iterações.

A execução do algoritmo dos Filtros implementado pode ser feita usando os parâmetros por defeito ou alterando-os, tanto no processo interno (correspondente à optimização sem restrições de f ou h) como no externo (correspondente ao Método dos Filtros e à construção do Filtro).

O algoritmo dos Filtros implementado neste trabalho é, então, à semelhança dos algoritmos anteriores, apresentado no fluxograma da FIGURA 7.6. Neste fluxograma o losango com a designação "Filtro" representa que será efectuado o teste de entrada para o Filtro, ou seja, será implementado o procedimento ilustrado pelo fluxograma da FIGURA 7.2, sendo que se neste procedimento o resultado é "False", ou seja, o ponto não é filtrado e portanto é aceite no Filtro, o algoritmo dos Filtros prossegue para o Estado correspondente à resposta "Sim", caso contrário segue para o Estado correspondente à resposta "Não". Este algoritmo é constituído por 16 estados.

Note-se que no estado 6, antes de verificar se $i \geq n$ testa-se se $k \geq kmax$ e no caso desta condição se verificar o processo iterativo pára, sendo devolvidos os resultados apresentados no estado 16. Este passo não foi representado no fluxograma por uma questão de simplicidade.

Após a implementação do Método dos Filtros de forma genérica, sendo possível usar todos os Métodos de Pesquisa Directa implementados, outra questão que também foi tida em atenção na implementação do algoritmo foi a forma como é medida a violação das restrições.

Os Métodos dos Filtros até agora implementados usam a norma 2 para esta medida, no entanto na definição da função h apenas se exige que seja contínua e que $h(x) \geq 0$ com $h(x) = 0$ se e só se x é admissível.

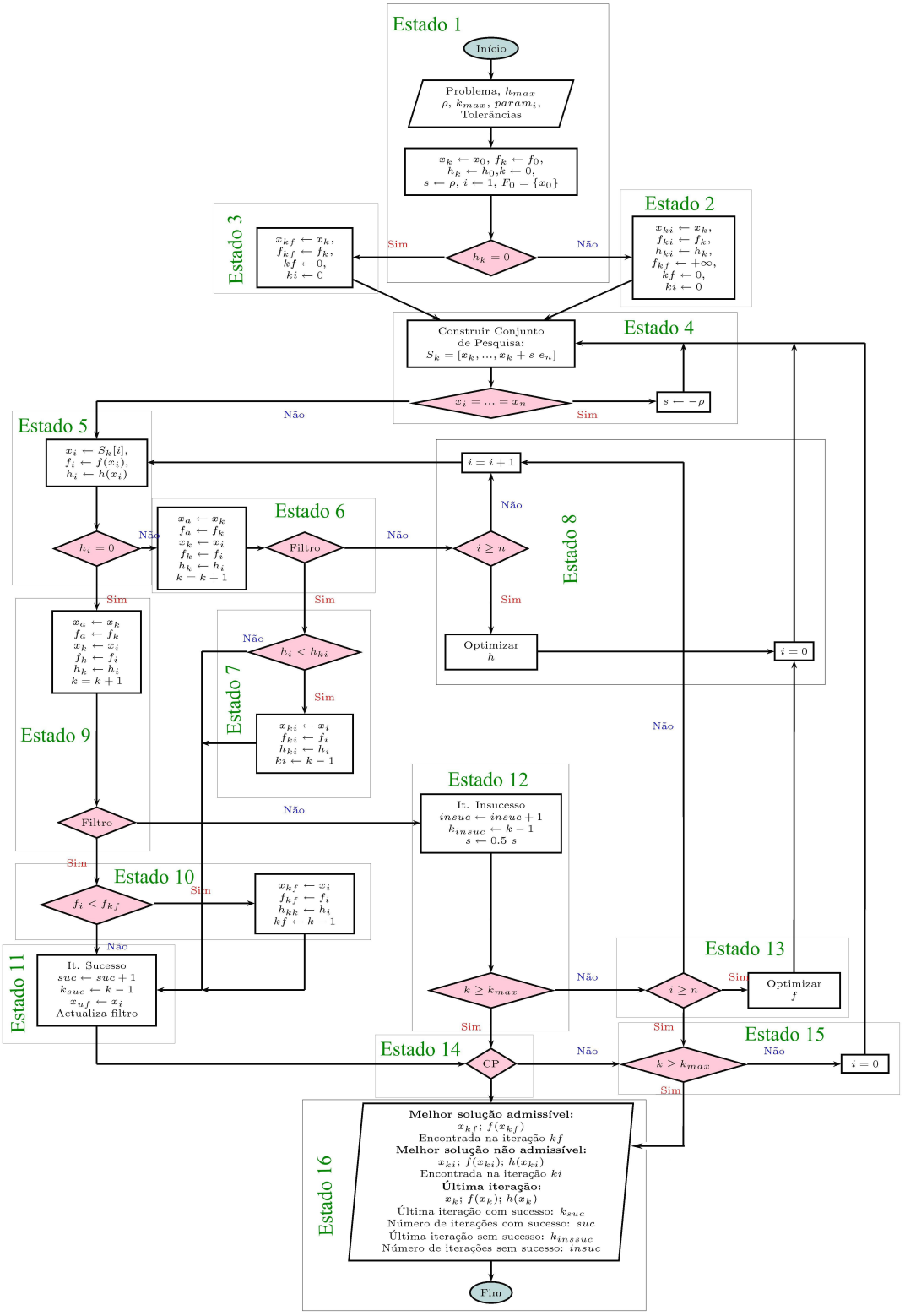

FIGURA 7.6: Algoritmo dos Filtros Implementado

Tendo em conta que nos Métodos de Penalidade/Barreira a violação das restrições é penalizada (ou implica a rejeição de pontos não admissíveis) usando funções de Penalidade/Barreira que não são mais do que medidas para a violação das restrições, foram testadas outras medidas para a violação das restrições, tendo como base as medidas de penalização ou rejeição utilizadas nos Métodos de Penalidade/Barreira. Os primeiros testes efectuados neste sentido foram apresentados em [42].

Assim, as medidas para h implementadas neste trabalho são: Norma 1, Norma 2, Barreira Extrema, Barreira Progressiva, Penalidade Clássica, Penalidade Estática/Dinâmica e Penalidade ℓ_1.

Estas medidas foram adaptadas, já que neste método não se usam parâmetros de Penalidade/Barreira, sendo as expressões utilizadas as apresentadas na Tabela 7.5.

Medida	Função h
Norma 1	$h(x) = \|C_+(x)\|_1 = \sum_{i=1}^{t+m} \max(0, c_i(x))$
Norma 2	$h(x) = \|C_+(x)\|_2 = \sqrt{\sum_{i=1}^{t+m} \max(0, c_i(x))^2}$
Barreira Extrema	$h(x) = \begin{cases} 0 & \text{se} \quad x \in \Omega \\ +\infty & \text{se} \quad x \notin \Omega \end{cases}$ onde Ω é a região admissível
Barreira Progressiva	$h(x) = \begin{cases} \sum_{i=1}^{m} (\max(c_i(x), 0))^2 & se \quad x \in X \\ +\infty & se \quad x \notin X \end{cases}$ onde $X \subset \Omega$ é o conjunto definido pelas restrições de igualdade
Penalidade Clássica	$h(x) = \sum_{i=1}^{t+m} [\max\{0, c_i(x)\}]^q, q \geq 1$
Penalidade Estática	$h(x) = \sum_{i=1}^{t} \|c_i(x)\|^q + \sum_{i=t+1}^{m} [\max(0, c_i(x)]^q, q > 1$
Penalidade ℓ_1	$h(x) = \sum_{i=1}^{t} \|c_i(x)\| + \sum_{i=t+1}^{m} \max[c_i(x), 0]$

TABELA 7.5: Medidas de h

Note-se que as medidas Norma 1, Norma 2 e Penalidade Clássica estão definidas apenas para problemas com restrições do tipo desigualdade e as medidas Barreira Extrema, Barreira Progressiva, Penalidade Estática/Dinâmica e Penalidade ℓ_1 podem ser usadas para problemas com restrições dos tipos igualdade e desigualdade.

No caso de se pretender resolver um problema com restrições dos tipos igualdade e desigualdade usando as medidas Norma 1, Norma 2 e Penalidade Clássica é necessário converter as restrições de igualdade em duas restrições de desigualdade. Este procedimento, de acordo com vários autores, não é o mais adequado uma vez que pode implicar uma dificuldade acrescida para encontrar aproximações à solução admissíveis.

Outra possibilidade, considerando as restrições de igualdade na forma $c_i(x) = 0$, com $i = 1, ..., t$, é nas expressões destas medidas considerar os valores $\max[|c_i(x)|, 0]$ em vez de $\max[c_i(x), 0]$. Esta possibilidade não foi implementada neste trabalho, mas pode ser uma das opções a implementar no futuro.

O Método dos Filtros implementado tem, assim, os seguintes parâmetros de entrada:

- Problema a resolver (função objectivo, restrições);
- Ponto inicial;
- A norma a usar (número inteiro de 1 a 7, correspondente às medidas definidos na Tabela 7.5 e de acordo com a ordem apresentada) – $norma_a_usar$;
- Número máximo de iterações no processo externo – k_{max};
- Tolerância para a distância entre 2 iterações – $T1$;
- Tolerância para a distância entre 2 valores da função objectivo entre iterações consecutivas – $T2$;
- Método a usar no processo interno (número inteiro de 1 a 5, correspondente aos métodos de Pesquisa Directa, apresentados no Capítulo 6 e de acordo com a ordem apresentada) – MET_i;
- Parâmetros do processo interno (todos os que foram apresentados no Capítulo 6);
- Valor máximo da violação das restrições – h_{max};
- Expoente a usar – q (no caso de se escolher a medida Penalidade Clássica ou Penalidade Estática);

e devolve os seguintes resultados:

1. Número de iterações efectuadas no processo externo – k;
2. Número de vezes que foi calculado o valor da função objectivo – $fEvals$
3. Número de vezes que foi calculado o valor da violação – $hEvals$
4. Última iteração efectuada – x_k;

5. Valor da função objectivo na última iteração efectuada – $f(x_k)$;
6. Melhor aproximação à solução admissível encontrada (se existir) – x_{kf} (se não existir é dada a indicação de que não foi encontrada nenhuma);
7. Valor da função objectivo na melhor aproximação à solução admissível encontrada – $f(x_{kf})$;
8. Iteração em que foi encontrada a melhor aproximação à solução admissível – kf
9. Melhor aproximação à solução não admissível encontrada (se existir) – x_{ki} (se não existir é dada a indicação de que não foi encontrada nenhuma);
10. Valor da função objectivo na melhor aproximação à solução não admissível encontrada – $f(x_{ki})$;
11. Valor da violação das restrições na melhor aproximação à solução não admissível – $h(x_{ki})$;
12. Iteração em que foi encontrada a melhor aproximação à solução não admissível – ki;
13. Filtro – $\mathcal{F}_k$.

Os valores usados por defeito são $norma_a_usar = 2$; $k_{max} = 40$; $T1 = 0.00001 = T2$; $MET_i = 1$; $h_{max} = +\infty$ e $q = 2$; os parâmetros do processo interno por defeito apresentados no Capítulo 6.

Note-se que considerando $h_{max} = +\infty$ não se está a estabelecer qualquer valor máximo para a violação das restrições, no entanto esse valor vai naturalmente sendo actualizado, à medida que o processo decorre. Isto porque só são aceites para o Filtro pontos não dominados, pelo que não serão aceites pontos com valor de h muito grande, a não ser que o valor de f nesses pontos seja mais pequeno do que qualquer outro no Filtro.

Apesar de nos testes efectuados não se ter definido um valor máximo para h, esse valor pode ser estabelecido à partida. Definir à partida h_{max} aumenta a exigência de aceitação para o Filtro, forçando a não aceitação de pontos cujo valor da violação seja superior e consequentemente a optimização de f e/ou h. Esta poderá ser uma boa opção para diminuir o inconveniente da possibilidade de nunca haver efectivamente a optimização de h e f que foi descrita anteriormente.

Na execução dos algoritmos para os problemas teste verificou-se que ao longo do processo iterativo na fase de pesquisa do conjunto havia repetição de iterações, uma vez que essa pesquisa se iniciava no último ponto aceite para o Filtro ou, no caso do ponto testado não ser aceite para o Filtro, no ponto resultante da optimização sem restrições de f ou h. Esta repetição era consequência de no primeiro caso (da pesquisa se iniciar no último ponto aceite para o Filtro) se testar a entrada no Filtro de um ponto que já tinha sido testado e aceite, consequentemente a seguir era um ponto não aceite, uma vez que já estava no Filtro, considerando-se uma iteração sem sucesso. Tendo-se verificado este problema de execução, fez-se uma alteração da a forma de iniciar o contador de posição no conjunto de pesquisa do ponto, considerando-se o contador de posição a iniciar na posição 1 ($i = 1$). Assim, no caso do ponto já ter sido testado no Filtro, constrói-se o conjunto de pesquisa a partir desse ponto, mas a pesquisa inicia-se no ponto seguinte e não nesse mesmo ponto, evitando assim repetições. Esta alteração teve como consequência uma alteração nos resultados obtidos. Na secção 7.3.2.1 serão referidas as consequências dessa alteração na execução do algoritmo dos problemas C801 e C802 em comparação com os que tinham sido obtidos em [41] e que são apresentados na TABELA 7.4.

7.3 Resultados Numéricos

Nesta secção são apresentados resultados numéricos que permitem comparar o comportamento dos algoritmos. Para fazer essa comparação, à semelhança do que foi feito no Capítulo 6 para os Algoritmos para Optimização sem Restrições, são primeiro apresentados os problemas teste, depois os resultados obtidos na resolução de cada um deles e por fim faz-se uma análise final dos resultados obtidos.

7.3.1 Problemas

Depois de implementados os Métodos de Pesquisa Directa para Optimização Com Restrições e incluídos na API desenvolvida, tal como já foi referido na secção 5.1, foram seleccionados 18 problemas teste (que são expostos no Apêndice A).

Dos 18 problemas seleccionados:

- os primeiros são problemas da colecção Cute [22]: C801; C802;

- o terceiro foi apresentado em [8] como problema para testar o método dos Filtro de de Audet et al.: PA;
- os restantes 15 são de Schittkowski em [138]: S224; S225; S226; S227; S228; S231; S233; S234; S249; S264; S270; S323; S324; S325; S326.

Sabendo que os Métodos de Barreira são mais adequados quando aplicados com pontos iniciais admissíveis, sendo de esperar que nestas condições se obtenham melhores resultados, foram realizados testes numéricos com os problemas com pontos iniciais admissíveis sem contudo deixar de se experimentar, alguns problemas com pontos iniciais não admissíveis. Os problemas com pontos iniciais não admissíveis são os problemas da colecção Cute: C801; C802; os restantes são problemas com pontos iniciais admissíveis.

7.3.2 Comparação de Resultados

Apresentam-se nesta secção os resultados numéricos finais (sem descrever as aproximações obtidas nos processos iterativos) obtidos da resolução dos problemas teste, usando os parâmetros por defeito.

Para ver os resultados obtidos ao longo do processo iterativo, que conduzem a estes resultados finais, basta executar os algoritmos correspondentes usando a API ou a sua componente gráfica, inserindo os respectivos dados de entrada. Os ficheiros resultantes da execução efectuada para a presente secção estão disponíveis no CD anexo a este trabalho.

Nesta secção discutem-se e comparam-se esses resultados, no sentido de conhecer a eficiência de cada um dos Métodos de Optimização Não Linear Com Restrições implementados. Os parâmetros de entrada foram seleccionados de acordo com o exposto nas secções anteriores e o mais similares possível, no sentido de comparar o desempenho dos métodos implementados.

Com o objectivo de efectuar uma comparação dos resultados obtidos com os diversos algoritmos, são aqui apresentados os resultados mais significativos (arredondados à 4^{a}casa decimal, exceptuando os escritos em notação científica para os quais se consideraram 3 algarismos significativos), nomeadamente: o número de avaliações da função objectivo (ou da função penalidade/barreira) $nEvals$, o número de iterações efectuadas k, o valor

da função objectivo na ultima iteração efectuada $f(xk)$ e o valor da função objectivo apresentado originalmente como valor mínimo para o respectivo problema $f(x^*)$(exposto no Apêndice A), para ser mais simples averiguar se os resultados obtidos são próximos do objectivo. Para se poder visualizar a admissibilidade da aproximação à solução encontrada, também são incluídos os valores das violações $V(x_k)$ $(= |\Phi(x_k) - f(x_k)|$ no caso dos métodos de Penalidade/Barreira e $= h(x_k)$ no caso do Método dos Filtros).

Com as alterações efectuadas aos algoritmos também é relevante conhecer os valores $f(x_{ki})$, $V(x_{ki})$, ki, correspondentes à melhor aproximação à solução não admissível, e $f(x_{kf})$, kf, correspondentes à melhor aproximação à solução admissível, pelo que estes valores também são apresentados.

Por uma questão de simplificação de escrita usaram-se abreviaturas para referir os algoritmos implementados, nomeadamente:

- Para os Métodos de Optimização Sem Restrições:
  - CS – Algoritmo de Pesquisa Coordenada;
  - HJ – Algoritmo de Hooke e Jeeves;
  - AA – Versão dos algoritmos de Audet et. al.;
  - NM – Algoritmo de Nelder-Mead;
  - SC – Algoritmo Simplex Convergente.
- Para os Métodos de Optimização Com Restrições:
  - PenBar – Algoritmo de Penalidade/Barreira, sendo que:
    1. EB – Barreira Extrema;
    2. PB – Barreira Progressiva;
    3. CP – Penalidade Clássica;
    4. SP – Penalidade Estática;
    5. DP – Penalidade Dinâmica;
    6. ℓ_1 – Penalidade ℓ_1.
  - MF – Método dos Filtros, tendo:
    1. N1 – Norma 1;
    2. N2 – Norma 2;

3. NEB – Barreira Extrema;
4. NPB – Barreira Progressiva;
5. NCP – Penalidade Clássica;
6. NSDP – Penalidade Estática/Dinâmica;
7. Nℓ_1 – Penalidade ℓ_1 .

Note-se que dos 18 problemas seleccionados apenas um deles tem uma restrição de igualdade (o problema S325). Além disso, as restrições de igualdade podem ser escritas como 2 restrições de desigualdade, isto é:

$c_i(x) = 0, i = 1, ..., t$

$\Leftrightarrow c_i(x) \leq 0 \wedge c_i(x) \geq 0, i = 1, ..., t$

$\Leftrightarrow c_i(x) \leq 0 \wedge -c_i(x) \leq 0, i = 1, ..., t;$

Assim, tal como foi dito na secção 3.4, os 18 problemas podem ser escritos na forma (3.34).

Neste trabalho e dado que existe apenas um problema com uma restrição de igualdade, consideraram-se os problemas nesta forma geral, mas mais tarde poderá ser estudado o comportamento dos algoritmos usando problemas com restrições de igualdade em vez das duas de desigualdade, uma vez que os algoritmos estão implementados para tratar com restrições dos 2 tipos. No que diz respeito à forma geral dos problemas a forma como foram escritos usando a Tecnologia Java baseia-se nesta forma geral, no entanto é possível escrevê-los distinguindo as restrições de igualdade das de desigualdade.

Considerando, $q = 2$ e as regras de actualização da Tabela 7.2 tem-se que a função Φ a optimizar nos métodos de Penalidade Clássica e Estática se reduzem exactamente à mesma função.

Levando em conta estas considerações, no caso do Método do Filtros, as 7 formas de medir a violação das restrições ficam reduzidas apenas às 4 apresentadas na Tabela 7.6, uma vez que, sem restrições de igualdade e com $q = 2$, a Norma 1 é igual à medida penalidade ℓ_1 e as medidas Penalidade Clássica, Estática e Dinâmica são a mesma medida.

Assim, consideram-se apenas 4 designações:

- N1 – para representar a Norma 1 e a medida Penalidade ℓ_1;
- N2 – para representar a Norma 2;
- NEB – para representar a medida Barreira Extrema;
- NP – para representar as medidas Barreira Progressiva, Penalidade Clássica e Penalidade Estática/Dinâmica;

Medida		h
Norma 1/Penalidade ℓ_1	N1	$h(x) = \|C_+(x)\|_1 = \sum_{i=1}^{n} \max[0,\ r_i(x)]$
Norma 2	N2	$h(x) = \|C_+(x)\|_2 = \sqrt{\sum_{i=1}^{n} \left\{\max[0,\ r_i(x)]^2\right\}}$
Barreira Extrema	NEB	$h(x) = \begin{cases} 0 & \text{if} \quad x \in \Omega \\ +\infty & \text{if} \quad x \notin \Omega \end{cases}$
Barreira Progressiva Penalidade Clássica Penalidade Estática/Dinâmica	NP	$h(x) = \sum_{i=1}^{n} \{\max[r_i(x),\ 0]\}^2$

TABELA 7.6: Medidas para h adaptadas e testadas

No sentido de se poder estabelecer se uma aproximação à solução é uma "Boa" aproximação ou não, à semelhança do que foi feito para os resultados dos algoritmos de optimização sem restrições, fixam-se os valores de referência.

Como neste caso é importante distinguir aproximações admissíveis de não admissíveis considerem-se soluções admissíveis as que possuem o valor da violação das restrições muito próximo de zero, ou seja, tão próximo quanto a precisão do computador permitir.

Seja $V(x_k) = |\Phi(x_k) - f(x_k)|$ no caso dos Métodos de Penalidade/Barreira e $V(x_k) = h(x_k)$ no caso do Método dos Filtros. Considere-se que uma aproximação é admissível quando o teste $V(x_k) = 0$ (ou seja, a violação das restrições é considerada nula) nos algoritmos implementados for aceite pelo computador como verdadeiro, ou seja, consideram-se admissíveis as aproximações que assim forem consideradas na execução dos algoritmos. Se na execução se verificar que $V(x_k) \neq 0$ (ou seja, a violação das restrições não é considerada nula) considera-se que a solução não é admissível.

Nos problemas com restrições a qualidade das soluções deve ter em conta não só os valores da função objectivo como os valores das violações das restrições, pelo que estes valores devem ser considerados para caracterizar as aproximações à solução obtidas.

Tomem-se, à semelhança do que foi feito anteriormente, os valores $0.0001 = 10^{-5}$ e $0.01 = 10^{-3}$ para as tolerâncias usadas na classificação das aproximações à solução, a seguir apresentadas.

Considera-se uma aproximação como sendo:

- *Aproximação à solução, admissível* se $|V(x_k)| = 0$, considerando-se:
  - Aproximação admissível *Boa* se: $|f(x^*) - f(x_k)| \leq 0.0001$;
  - Aproximação admissível *Média*, se: $0.0001 < |f(x^*) - f(x_k)| \leq 0.01$;
  - Aproximação admissível *Má*, se: $|f(x^*) - f(x_k)| > 0.01$.
- *Aproximação à solução, não admissível* se $|V(x_k)| \neq 0$, considerando-se:
  - Aproximação não admissível *Boa* se: $|f(x^*) - f(x_k)| \leq 0.0001 \ \wedge \ |V(x_k)| \leq 0.0001$;
  - Aproximação não admissível *Média*, se:
    - $0.0001 < |f(x^*) - f(x_k)| \leq 0.01 \ \wedge \ |V(x_k)| \leq 0.0001$;
    - ou $|f(x^*) - f(x_k)| \leq 0.0001 \ \wedge 0.0001 < |V(x_k)| \leq 0.01$;
    - ou $0.0001 < |f(x^*) - f(x_k)| \leq 0.01 \ \wedge 0.0001 < |V(x_k)| \leq 0.01$;
  - Aproximação não admissível *Má*, se: $|f(x^*) - f(x_k)| > 0.01 \ \vee \ |V(x_k)| > 0.01$.

Tendo em conta estes critérios foi efectuada a classificação das aproximações obtidas. Os cálculos correspondentes podem ser vistos nos ficheiros, no CD anexo a este trabalho:

- *ResultadosClassificacaoPenBar.xlsx* ou *ResultadosClassificacaoPenBar.ods*;
- *ResultadosClassificacaoFiltro.xlsx* ou *ResultadosClassificacaoFiltro.ods*.

Em cada uma das tabelas que se seguem apresentam-se genericamente os resultados obtidos. Os melhores resultados obtidos (valores mínimos dos resultados obtidos na execução dos algoritmos) são destacados com fundo cinza.

Nestes destaques considera-se a melhor aproximação à solução admissível a que tiver o valor de f mais próximo do objectivo. Das aproximações à solução não admissíveis destaca-se a que tiver o valor de f mais próximo do objectivo e/ou o valor de V mais

próximo de zero. Ainda são destacados o menor número de avaliações das funções, o menor número de iterações e a menor diferença entre o valor da função objectivo na aproximação e o seu valor na solução apresentada. Nas tabelas de resultados o * representa que não foi encontrada nenhuma aproximação ou que, apesar de ter sido encontrada uma aproximação, não é melhor do que o ponto inicial.

Note-se que, apesar do inconveniente descrito anteriormente, relativamente à forma de contagem das iterações do Método dos Filtros implementado, uma vez que, se o número máximo de iterações for pequeno, o número de iterações permitidas pode ser atingido sem que se proceda à efectiva optimização de f e/ou h, nestas implementações considerou-se tanto nos Métodos de Penalidade/Barreira como no Método dos Filtros o número máximo de iterações no processo externo de $k_{max} = 40$.

Assim, na resolução dos problemas em que se verificar esta ocorrência será testado o comportamento do algoritmo dos Filtros com $k_{max} = 100$. Neste caso, será ainda importante fazer uma análise do número de avaliações das funções, no sentido de averiguar se esta alteração tem como consequência um efectivo aumento do custo, neste sentido, do processo de optimização.

Na análise dos resultados das secções seguintes serão destacados os métodos que permitiram obter mais valores mínimos, ou seja, mais valores dos destacados nas tabelas de resultados com fundos cinza.

Em alguns dos problemas são ainda apresentadas figuras que ilustram o processo iterativo efectuado e os resultados finais, com o objectivo de ilustrar graficamente todo o processo.

A elaboração destes gráficos não é ainda uma das componentes da API desenvolvida, mas pretende-se que o seja no futuro, bem como a interligação com uma base de dados onde se guardem os resultados obtidos nas execuções.

7.3.2.1 Problemas da Colecção Cute

Problema C801

Os Métodos de Pesquisa Directa para Optimização com Restrições implementados não permitem obter *Boas* aproximações à solução do Problema C801, de acordo com os critérios definidos e com os resultados apresentados nas TABELAS 7.7 e 7.8.

Usando alguns dos algoritmos de Penalidade, nomeadamente a Penalidade ℓ_1 em conjugação com todos os Métodos de Pesquisa Directa para Optimização sem Restrições e a Clássica conjugada com o algoritmo HJ, é possível encontrar aproximações admissíveis à solução do problema *Médias*.

O Método de Penalidade ℓ_1 também é o que permite obter mais valores mínimos, nomeadamente no número de avaliações da função de Penalidade (Φ_{Evals}), no número de iterações necessárias no processo de optimização (k), no valor de $f(x_{kf})$, no valor de $V(x_{ki})$ e ki.

Problema		Ponto Inicial										Solução					
C801		x0=(0,1;-0,1)			f(x0)=160,87							x*=(?,?)				f(x*) = 7,563	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível					
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class	
CS	EB	16081	40	160,87	infinito	infinito	Má	*	*	*	*	*	*	*	*	*	
	PB	595	3	6,6793	infinito	infinito	Má	*	*	*	*	6,6793	0,4262	3	0,8837	Má	
	CP/SP	1715	18	7,5578	7,5578	0,0052	Média	*	*	*	*	7,5578	5,04E-06	18	0,0052	Média	
	DP	3412	40	7,4677	7,5124	0,0953	Má	*	*	*	*	7,4677	0,0448	40	0,0953	Má	
	l1	276	3	7,5581	7,5581	0,0049	Média	7,5581	3	0,0049	Média	7,5581	8,62E-06	2	0,0049	Média	
HJ	EB	32001	40	160,87	infinito	infinito	Má	*	*	*	*	*	*	*	*	*	
	PB	3434	40	6,6797	7,1055	0,8833	Má	*	*	*	*	6,6797	0,4259	2	0,8833	Má	
	CP/SP	2152	18	7,5608	7,5608	0,0022	Média	7,5608	17	0,0022	Média	7,5605	8,22E-05	14	0,0025	Média	
	DP	3590	40	7,4678	7,5125	0,0952	Má	*	*	*	*	7,4678	0,0447	40	0,0952	Má	
	l1	523	7	7,5584	7,5584	0,0046	Média	7,5584	6	0,0046	Média	7,5582	4,06E-05	2	0,0048	Média	
AA	EB	32081	40	160,87	infinito	infinito	Má	*	*	*	*	*	*	*	*	*	
	PB	1113	3	6,6792	7,1055	0,8838	Má	*	*	*	*	6,6792	0,4263	3	0,8838	Má	
	CP/SP	2812	18	7,5579	7,5579	0,0051	Média	*	*	*	*	7,5579	4,01E-06	18	0,0051	Média	
	DP	5831	40	7,4676	7,5124	0,0954	Má	*	*	*	*	7,4676	0,0448	40	0,0954	Má	
	l1	676	3	7,5578	7,5578	0,0052	Média	7,5578	3	0,0052	Média	7,5578	8,12E-07	2	0,0052	Média	
NM	EB	32241	40	160,87	infinito	infinito	Má	*	*	*	*	*	*	*	*	*	
	PB	993	3	6,67937	infinito	infinito	Má	*	*	*	*	6,6794	0,4262	3	0,8836	Má	
	CP/SP	1579	18	7,5575	7,5575	0,0055	Média	*	*	*	*	7,5575	1,01E-05	18	0,0055	Média	
	DP	2787	40	7,4676	7,5124	0,0954	Má	*	*	*	*	7,4676	0,0448	40	0,0954	Má	
	l1	260	3	7,5575	7,5575	0,0055	Média	7,5575	3	0,0055	Média	7,5575	1,57E-06	2	0,0055	Média	
SC	EB	4241	40	160,87	infinito	infinito	Má	*	*	*	*	*	*	*	*	*	
	PB	823	6	6,6903	7,1056	0,8727	Má	*	*	*	*	6,6903	0,4153	5	0,8727	Má	
	CP/SP	2445	17	7,5575	7,5575	0,0055	Média	*	*	*	*	7,5575	6,49E-06	17	0,0055	Média	
	DP	4393	40	7,4675	7,5124	0,0955	Má	*	*	*	*	7,4675	0,0449	40	0,0955	Má	
	l1	514	3	7,5575	7,5575	0,0055	Média	7,5575	2	0,0055	Média	7,5575	1,01E-06	3	0,0055	Média	
Mínimos		260	3			0,0022		7,5575	2	0,0022		6,6792	8,12E-07	2	0,0025		

TABELA 7.7: Resultados para o Problema C801, usando os algoritmos de Penalidade/Barreira

Note-se que, tal como já foi referido na secção 7.1, a possibilidade de identificação das melhores aproximações à solução, admissível e não admissível, nos métodos de Penalidade/Barreira, que neste trabalho fazem parte dos dados de saída disponíveis e que são apresentados na TABELA 7.7, não são dados de saída usuais nestes algoritmos.

Assim, os resultados usuais a apresentar, quando se usam algoritmos deste tipo, são os que constam nas colunas correspondentes à *Última Iteração*. As colunas correspondentes à *Melhor Aproximação Admissível* e à *Melhor Aproximação não Admissível* são, assim, dados adicionais que são obtidas por procedimentos implementados neste trabalho que os permitem guardar, conforme foi explicado na secção 7.1.

Nos resultados apresentados na TABELA 7.7 pode observar-se que o Método de Penalidade ℓ_1 em conjugação com o Método de Nelder-Mead é o que permite encontrar uma aproximação à solução com o menor número de iterações, 3, e com o menor valor de avaliações da função Φ, 260. Este ainda foi o método com que se obteve o melhor valor da função objectivo para uma aproximação admissível à solução e uma *Média* aproximação não admissível à solução com o menor número de iterações.

Estes resultados quando comparados com os apresentados em [39] e reproduzidos na TABELA 7.3, onde se consideraram os mesmos parâmetros de entrada à excepção do número máximo de iterações nos processos interno e externo (em [39] considerou-se $k_{maxI} = 40$ o número máximo de iterações no processo interno e $k_{maxE} = 100$ o número máximo de iterações no processo externo, e aqui considera-se $k_{maxI} = 100$ e $k_{maxE} = 40$) revelam uma melhoria em vários aspectos: diminuição do número de avaliações de Φ, diminuição muito significativa do número de iterações efectuadas, foi possível identificar e devolver uma aproximação admissível à solução e uma melhoria da qualidade da aproximação não admissível à solução, no que diz respeito tanto ao valor da função objectivo como ao valor da violação das restrições.

Deste modo, para este problema, revelou-se mais adequada a aposta num maior número de iterações na optimização sem restrições do que no número de vezes que são actualizados os parâmetros de penalidade e é construída uma nova função objectivo.

Ainda é possível verificar que, qualquer que seja o método usado no processo interno, o Método de Barreira Extrema não permite encontrar qualquer aproximação à solução melhor do que o ponto inicial, apesar de ter sido necessário calcular muitas vezes o valor de Φ. Sendo este um dos problemas com ponto inicial não admissível, não foi possível encontrar com este método pontos dentro da região admissível, pelo que as iterações realizadas foram todas recusadas como possíveis aproximações.

Os Métodos de Barreira Progressiva e Penalidade Dinâmica só permitem encontrar aproximações não admissíveis *Más*, de acordo com os critérios definidos. O Método de Penalidade Clássica só permite encontrar uma aproximação admissível à solução em conjugação com o Método de Hooke-Jeeves, tendo encontrado aproximações não admissíveis *Médias* em conjugação com qualquer um dos Métodos de Pesquisa Directa e o Método de Penalidade ℓ_1 é o único que permite encontrar aproximações admissíveis e não admissíveis em conjugação com qualquer um dos Métodos de Pesquisa Directa.

Posto isto é de concluir que para este problema o Método de Penalidade ℓ_1 foi o que se mostrou mais eficiente.

Os resultados obtidos usando o algoritmo dos Filtros, apresentados na TABELA 7.8, já tinham sido apresentados para o Método dos Filtros em conjugação com o Método de Nelder-Mead em [41] e foram reproduzidos na TABELA 7.4.

Estes novos resultados, obtidos quando o Método dos Filtros é conjugado com todos os Métodos de Pesquisa Directa, são muito semelhantes para todos os métodos usados nos processos interno e externo, diferindo genericamente apenas no número de avaliações das funções envolvidas no processo de optimização. Em todos eles, a melhor aproximação admissível à solução possui o valor aproximado de 9,6700 para a função objectivo, não sendo encontrada qualquer aproximação admissível à solução no caso de se usar a medida de violação das restrições NEB.

Esta medida, à semelhança do que aconteceu na execução do algoritmo de Penalidade/Barreira com a Barreira Extrema, não mostrou um bom desempenho como era de esperar, já que esta medida/função é mais adequada para problemas cujo ponto inicial seja um ponto admissível, o que não é o caso.

Problema		Ponto Inicial									Solução					
C801		x0=(0,1;-0,1)			f(x0)=160,87						x*=(?,?)				f(x*) = 7,563	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	N1/N2	47	1617	40	31,2458	0,0000	9,6700	7	2,1070	Má	38,4700	0,1000	3	6	30,9070	Má
	NEB	41	2848	40	15,2700	infinito	9,6700	7	2,1070	Má	*	*	*	2	*	*
	NP	47	1617	40	31,2458	0,0000	9,6700	7	2,1070	Má	38,4700	0,0100	3	6	30,9070	Má
HJ	N1/N2	66	182	40	54,8700	0,0000	9,6700	7	2,1070	Má	38,4700	0,1000	3	6	30,9070	Má
	NEB	66	176	40	54,8700	0,0000	9,6700	7	2,1070	Má	*	*	*	2	*	*
	NP	66	182	40	54,8700	0,0000	9,6700	7	2,1070	Má	38,4700	0,0100	3	6	30,9070	Má
AA	N1/N2	47	3169	40	14,4458	0,0000	9,6700	7	2,1070	Má	38,4700	0,1000	3	6	30,9070	Má
	NEB	41	5648	40	15,2700	infinito	9,6700	7	2,1070	Má	*	*	*	2	*	*
	NP	47	3169	40	14,4458	0,0000	9,6700	7	2,1070	Má	38,4700	0,0100	3	6	30,9070	Má
NM	N1/N2	83	173	40	76,1200	0,0000	9,6700	7	2,1070	Má	38,4700	0,1000	3	6	30,9070	Má
	NEB	41	5676	40	15,2700	infinito	9,6700	7	2,1070	Má	*	*	*	2	*	*
	NP	83	173	40	76,1300	0,0000	9,6700	7	2,1070	Má	38,4700	0,0100	3	6	30,9070	Má
SC	N1/N2	53	618	40	61,8442	0,0000	9,6700	7	2,1070	Má	38,4700	0,1000	3	8	30,9070	Má
	NEB	41	776	40	15,2700	infinito	9,6700	7	2,1070	Má	*	*	*	2	*	*
	NP	53	612	40	60,4784	0,0000	9,6700	7	2,1070	Má	38,4700	0,0100	3	8	30,9070	Má
Minimos		41	1617	40			9,6700	7,0000	2,1070		38,4700	0,0100	3		30,9070	

TABELA 7.8: Resultados para o Problema C801, usando o algoritmo dos Filtros

Apesar destes resultados serem piores do que os obtidos usando os Métodos de Penalidade/Barreira, apresentados na TABELA 7.7 é de salientar que este algoritmo tem incluído nos seus dados de saída o conjunto de soluções não dominadas, das quais se pode escolher a mais adequada a uma situação real modelada por um problema deste tipo.

O Filtro obtido na execução do algoritmo dos Filtros, por exemplo, em conjugação com o algoritmo de Pesquisa Coordenada e usando a medida N1, possui 6 aproximações à solução.

Estes pontos são os pontos não dominados do conjunto de iterações encontradas ao longo de todo o processo iterativo. O conjunto de iterações efectuadas encontra-se representado na FIGURA 7.7. Esta representação, bem como as seguintes, tem como objectivo ilustrar o comportamento do algoritmo, salientando-se desta forma as principais diferenças entre o algoritmo de Penalidade/Barreira e o algoritmo dos Filtros implementados.

Na FIGURA 7.7 os pontos do Filtro são apresentados através de losangos e as iterações obtidas ao longo do processo através de quadrados. Nesta figura o eixo das abcissas corresponde aos valores de f e o eixo das ordenadas aos valores de h.

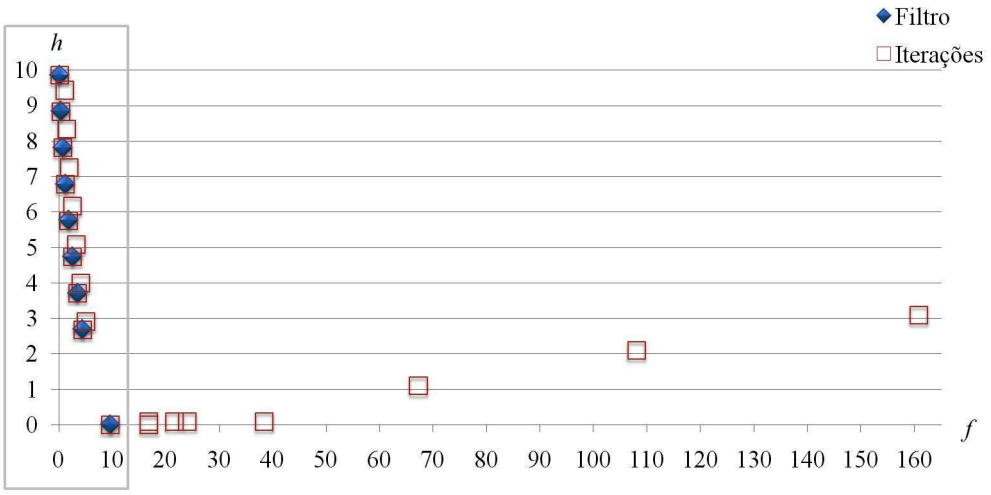

FIGURA 7.7: Iterações – Problema C801,
usando o algoritmo dos Filtros e Pesquisa Coordenada

Note-se que de todas as iterações efectuadas foram apenas aceites no Filtro pontos que não pertenciam à região proibida de pontos anteriores, de acordo com a Regra de Pareto, e pontos dominados por novas iterações foram eliminados do Filtro.

A representação gráfica do Filtro, mais em pormenor pode ser vista na FIGURA 7.8. Com esta representação é possível identificar pontos da denominada Frente de Pareto obtida na execução do algoritmo para este problema.

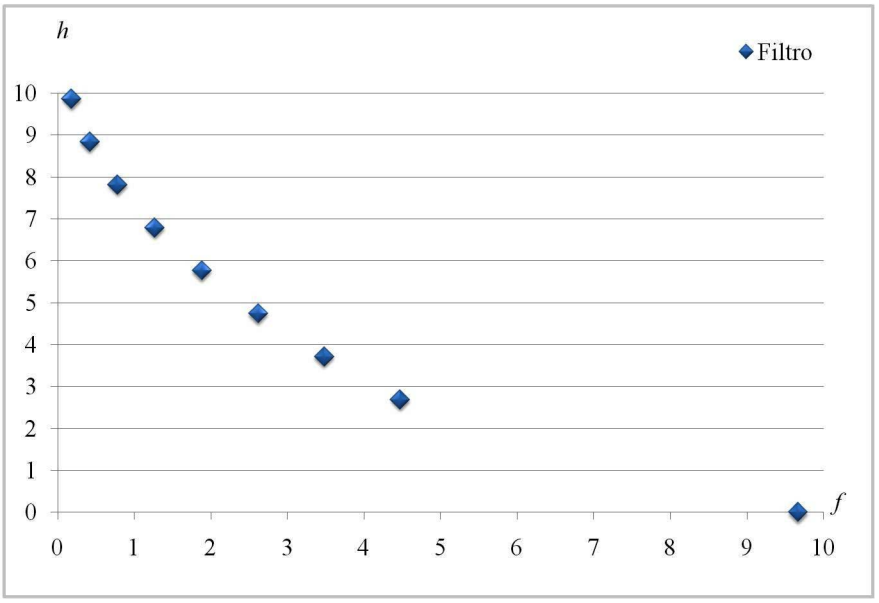

FIGURA 7.8: Filtro para o Problema C801,
usando o algoritmo dos Filtros e Pesquisa Coordenada

Na presença desta informação o decisor fica em posse das aproximações à solução que possuem melhor conjugação de valores da função objectivo e da violação das restrições, sendo possível escolher a que melhor se adequa à situação que pretende resolver.

Imagine-se, por exemplo, que os valores da função f representam gastos de uma empresa (em milhões de euros) depois de ser feito um investimento sujeito a determinadas restrições, medidas pelos valores de h. É possível que seja proveitoso optar, por exemplo, pela solução que mais minimiza f (ou seja, que possui o valor de f muito próximo de zero) e fazer um investimento que implica uma medida de violação de 10 unidades (valor correspondente de h).

Dependendo da situação em questão, e havendo ou não a possibilidade de flexibilidade na verificação das restrições, pode então ser útil apresentar um conjunto de soluções ao decisor, para que ele decida a que mais lhe convém. Nestes casos o Método dos Filtros pode ser uma boa opção.

Para este problema foi possível, usando os algoritmos de Penalidade/Barreira, encontrar aproximações à solução que, no caso do problema não ser relaxável são melhores, uma vez que são mais próximas do objectivo, no entanto a diversidade de soluções do Método dos Filtros pode ser uma mais valia em problemas relaxáveis.

Ainda se poderá analisar o comportamento do processo iterativo (considerando ainda o algoritmo dos Filtros em conjugação com o algoritmo de Pesquisa Coordenada e usando a medida N1) no que diz respeito à posição das aproximações à solução, relativamente às curvas de nível da função objectivo e à região admissível definida pelas restrições do problema (ver FIGURA 7.9) e também relativamente ao comportamento dos valores da função objectivo e da violação das restrições ao longo do processo (ver FIGURA 7.10).

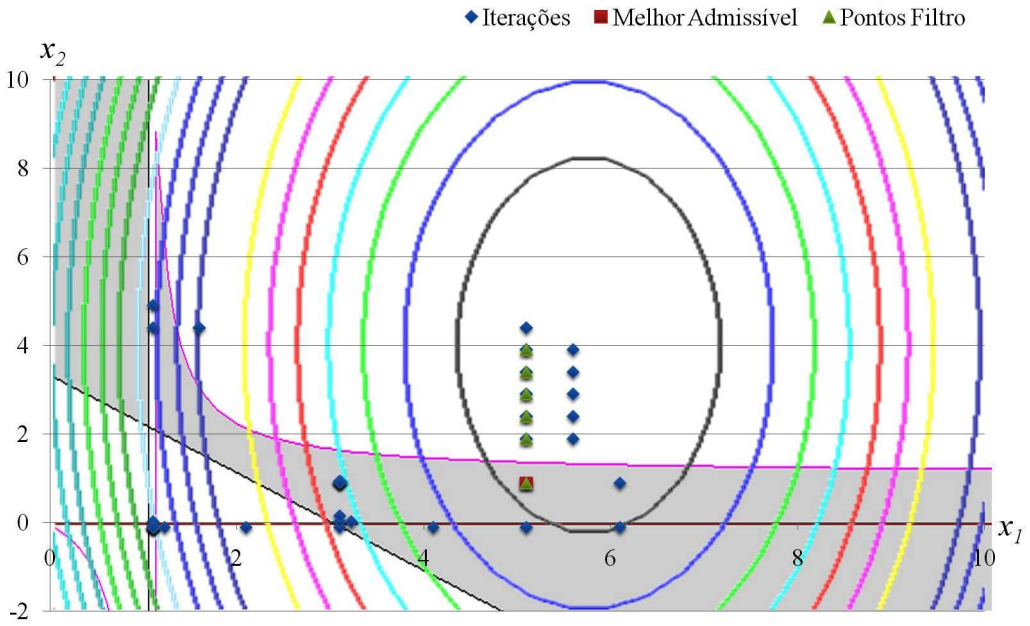

FIGURA 7.9: Posição das aproximações à solução – MF/CS (Problema C801)

Na FIGURA 7.9 a zona sombreada a cinzento representa a região admissível, os eixos das abcissas e das ordenadas representam as coordenadas da iteração, algumas das curvas de nível da função objectivo são também apresentadas e as iterações efectuadas ao longo do processo, os pontos do Filtro final e a melhor iteração admissível são representadas, respectivamente, por losangos, triângulos e um quadrado.

De notar que o ponto inicial deste problema é o ponto $(0.1, -0.1)^T$, que é um ponto não admissível, e que muitas das iterações efectuadas são pontos admissíveis.

Observando o comportamento dos valores da função objectivo e da violação das restrições ao longo do processo, na FIGURA 7.10, onde o eixo das abcissas representa o número da iteração, o eixo das ordenadas representa os valores das funções, os valores de h estão representados por triângulos e os de f por quadrados, e sabendo que a melhor aproximação

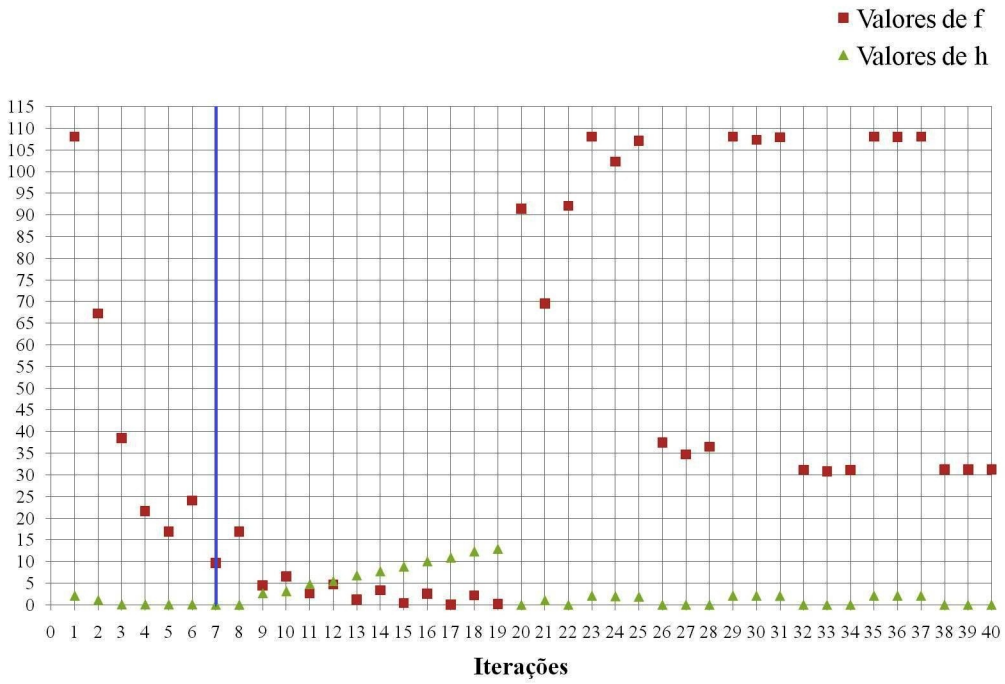

FIGURA 7.10: Comportamento dos valores da função objectivo e da violação das restrições ao longo do processo – MF/CS (Problema C801)

admissível à solução foi obtida na iteração 7 (destacada com uma linha vertical de maior espessura), pode verificar-se que houve um progresso nas iterações iniciais relativamente rápido, havendo uma diminuição tanto dos valores da função objectivo como dos valores da violação das restrições, no entanto, a partir da iteração 7, na tentativa de optimizar ora f ora h, sem levar em atenção os valores de uma enquanto a outra é optimizada, o progresso em direcção ao óptimo conduziu a iterações piores do que as que já tinham sido encontradas, no que diz respeito à regra de não dominância, usada na aceitação de novas iterações no Filtro.

Os resultados obtidos com os Métodos de Penalidade/Barreira, nomeadamente usando a função CP e o algoritmo HJ no processo interno, com os quais se obtiveram as melhores aproximações à solução (as aproximações admissível e não admissível que têm o valor da função objectivo mais próximo do valor pretendido, ou seja, as que têm os valores de $V = |f(x^*) - f(x_k)|$ menores), apresentados na FIGURA 7.11 mostram logo nas primeiras iterações há uma diminuição da violação V, apesar do aumento de f, sendo que a iteração final não é a que possui melhores valores, de acordo com a classificação que foi feita de melhor aproximação à solução admissível e não admissível. Estas aproximações foram

obtidas na iteração 14 e 17 de acordo com o que já tinha sido apresentado na TABELA 7.7.

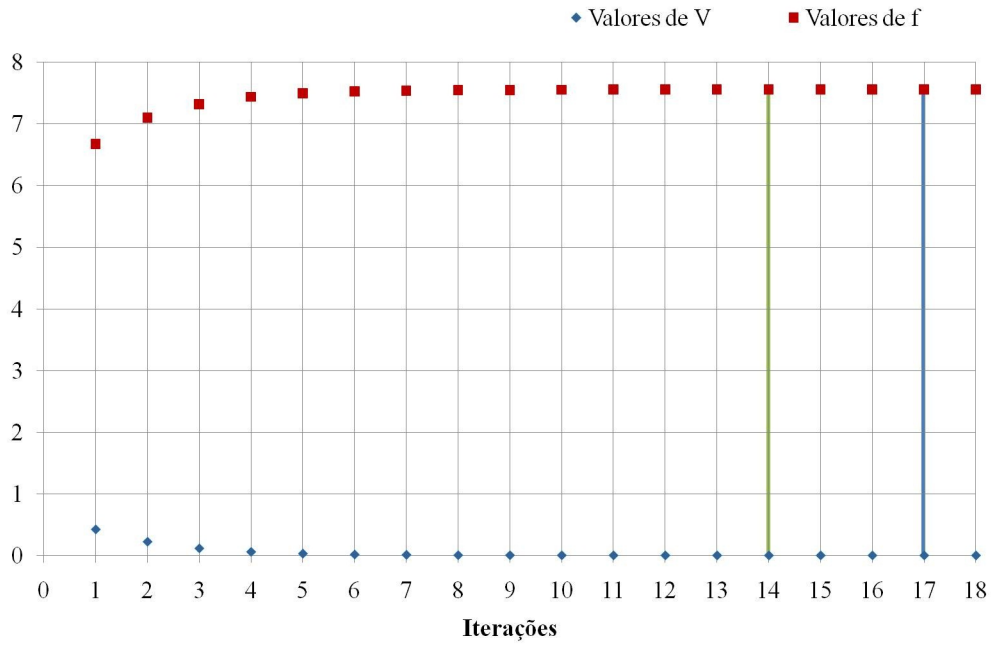

FIGURA 7.11: Comportamento dos valores da função objectivo e da violação das restrições ao longo do processo – CP/HJ (Problema C801)

Assim, para a resolução deste problema foram mais eficazes os Métodos de Penalidade/Barreira, à excepção do Método de Barreira Extrema, com o qual não foi possível encontrar qualquer aproximação à solução.

Ainda na tentativa de melhorar o desempenho do Método dos Filtros foi testada a sua execução com 100 iterações, no entanto não se obtiveram melhores aproximações do que as obtidas com 40 iterações. No caso, por exemplo, de se usar o CS no processo interno o desempenho do método foi semelhante ao verificado anteriormente, ilustrado na FIGURA 7.10, a partir da iteração 7. A partir desta iteração as tentativas de optimizar f ou h individualmente conduziram a iterações que melhoravam os valores destas funções, mas não foram encontradas mais iterações aceites no Filtro.

Como já se tinha referido no final da Secção 7.2, os resultados aqui apresentados diferem dos que tinham sido obtidos em [41] e que são apresentados na TABELA 7.4, uma vez que foi feita uma alteração na forma de iniciar o contador de posição no conjunto de pesquisa no caso da entrada do ponto no Filtro já ter sido testada. Assim, evitando

esta repetição há um consequente aumento de diferentes iterações ao longo do processo verificando-se que em 40 iterações há optimização efectiva de f e h. Isto pode ser verificado observando que os resultados agora apresentados possuem genericamente um número de avaliações das funções superior ao apresentado na TABELA 7.4. No entanto, as melhores aproximações à solução (admissível e não admissível) continuam a ser as mesmas, não havendo neste caso melhoria nas aproximações à solução.

Problema C802

Problema		Ponto Inicial										Solução				
C802		x0=(0,1;-0,1)				f(x0)=295,1						x*=(?,?)		f(x*) =	-97,30952	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB	16081	40	295,1	infinito	infinito	Má	*	*	*	*	*	*	*	*	*
	PB	602	3	58,5432	69,9274	58,5432	Má	*	*	*	*	58,5432	11,3843	3	155,85272	Má
	CP/SP	3234	23	84,6734	84,6735	84,6734	Má	84,6735	22	181,98302	Má	84,6734	0,0000234	21	181,98292	Má
	DP	6034	40	83,2361	83,9481	83,2361	Má	*	*	*	*	83,2361	0,7119	40	180,54562	Má
	l1	682	6	85,2996	85,2996	85,2996	Má	85,2996	6	182,60912	Má	85,2995	0,0000198	5	182,60902	Má
HJ	EB	32001	40	538369,9048	infinito	infinito	Má	*	*	*	*	*	*	*	*	*
	PB	1540	5	58,6162	69,9278	58,6162	Má	*	*	*	*	58,6162	11,3116	4	155,92572	Má
	CP/SP	2073	19	84,6999	84,6999	84,6999	Má	84,6999	18	182,00942	Má	84,6918	0,0015	13	182,00132	Má
	DP	3424	40	83,2306	83,949	83,2306	Má	*	*	*	*	83,2306	0,7185	40	180,54012	Má
	l1	817	9	85,2381	85,2381	85,2381	Má	85,2381	8	182,54762	Má	85,2324	0,0118	5	182,54192	Má
AA	EB	32081	40	295,1	infinito	infinito	Má	*	*	*	*	*	*	*	*	*
	PB	1513	5	58,5434	69,9275	58,5434	Má	*	*	*	*	58,5434	11,3841	5	155,85292	Má
	CP/SP	4862	23	84,7016	84,7016	84,7016	Má	84,7016	22	182,01112	Má	84,7016	5,62E-06	23	182,01112	Má
	DP	9238	40	83,2342	83,9481	83,2342	Má	*	*	*	*	83,2342	0,7139	40	180,54372	Má
	l1	1145	6	84,7977	84,7977	84,7977	Má	84,7977	5	182,10722	Má	84,7978	0,0000185	6	182,10732	Má
NM	EB	32241	40	295,1	infinito	infinito	Má	*	*	*	*	*	*	*	*	*
	PB	1295	3	58,5429	infinito	infinito	Má	*	*	*	*	58,5429	11,3845	3	155,85242	Má
	CP/SP	2214	23	84,6712	84,6712	84,6712	Má	84,6712	22	181,98072	Má	84,6712	0,0000148	20	181,98072	Má
	DP	2991	40	83,2356	83,9481	83,2356	Má	*	*	*	*	83,2356	0,7125	40	180,54512	Má
	l1	535	6	84,671	84,671	84,671	Má	84,671	5	181,98052	Má	84,671	1,2E-08	6	181,98052	Má
SC	EB	4241	40	295,1	infinito	infinito	Má	*	*	*	*	*	*	*	*	*
	PB	366	3	58,5457	69,9274	58,5457	Má	*	*	*	*	58,5457	11,3818	2	155,85522	Má
	CP/SP	3255	23	84,6712	84,6712	84,6712	Má	84,6712	21	181,98072	Má	84,6712	1,61E-06	23	181,98072	Má
	DP	6618	40	83,2361	83,9481	83,2361	Má	*	*	*	*	83,2361	0,712	40	180,54562	Má
	l1	1136	7	84,671	84,671	84,671	Má	84,671	6	181,98052	Má	74,3006	9,8267	4	171,61012	Má
Minimos		366	3			58,5432		84,671	5	181,98052		58,5429	1,2E-08	2	155,85242	

TABELA 7.9: Resultados para o Problema C802, usando os algoritmos de Penalidade/Barreira

Os Métodos de Pesquisa Directa para Optimização com Restrições implementados não permitem obter nem *Boas*, nem *Médias* aproximações à solução do Problema C802, de acordo com os critérios definidos e os dados das TABELAS 7.9 e 7.10.

Para este problema os Métodos de Penalidade/Barreira, à semelhança do que aconteceu para o problema C801, também permitem encontrar aproximações mais próximas do objectivo, no entanto não constituem *Boas* aproximações à solução. No caso do Método dos Filtros, as aproximações também são *Más*, o melhor resultado obtido foi usando o Método de Nelder-Mead no processo interno.

Comparando os dados apresentados na TABELA 7.10 com os da TABELA 7.4, onde se usou a medida N2 para h, verifica-se que houve uma melhoria na iteração em que a melhor

aproximação admissível à solução foi encontrada. No entanto foram necessárias mais avaliações da função correspondente à violação das restrições.

Nestas condições este aumento permitiu que o Filtro final possua 18 aproximações à solução em contraposição com os dados anteriormente apresentados onde o Filtro era composto por 12 aproximações.

Problema		Ponto Inicial									Solução						
C802		x0=(0,1;-0,1)				f(x0)=295,1					x*=(?,?)				f(x*) = -97,3		
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível						
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class	
CS	N1/N2	43	1214	40	217,3631	1,1131	100,7000	22	198,0095	Má	*	*	*	11	*	*	
	NEB	41	2848	40	117,1000	infinito	*	*	*	*	*	*	*	1	*	*	
	NP	43	1214	40	217,3631	1,2490	100,7000	2	198,0095	Má	*	*	*	11	*	*	
HJ	N1	41	205	40	161,5305	0,0000	125,6877	22	222,9972	Má	*	*	*	12	*	*	
	N2	41	205	40	161,5305	0,0000	125,6877	22	222,9972	Má	*	*	*	14	*	*	
	NEB	56	260	40	100,6559	0,0000	93,9844	29	191,2939	Má	*	*	*	*	*	*	
	NP	41	205	40	161,5305	0,0000	125,6877		222,9972	Má	*	*	*	14	*	*	
AA	N1	41	1480	40	53,1000	4,2200	100,7000	38	198,0095	Má	*	*	*	16	*	*	
	N2	41	1480	40	53,1000	4,2200	100,7000	38	198,0095	Má	*	*	*	17	*	*	
	NEB	41	5648	40	117,1000	infinito	*	*	*	*	*	*	*	1	*	*	
	NP	41	1480	40	53,1000	17,8084	100,7000	38	198,0095	Má	*	*	*	17	*	*	
NM	N1/N2	41	331	40	20,0000	14,1200	**87,2000**	36	**184,5095**	Má	*	*	*	18	*	*	
	NEB	41	5676	40	117,1000	infinito	*	*	*	*	*	*	*	1	*	*	
	NP	41	331	40	20,0000	199,3744	**87,2000**	36	**184,5095**	Má	*	*	*	18	*	*	
SC	N1/N2	41	520	40	29,0766	9,2403	88,6641	37	185,9736	Má	*	*	*	15	*	*	
	NEB	41	776	40	117,1000	infinito	*	*	*	*	*	*	*	1	*	*	
	NP	41	520	40	29,0766	85,3834	88,6641	37	185,9736	Má	*	*	*	15	*	*	
Minimos		**41**	**205**	**40**			**87,2**	**2**	**184,50952**		**0**	**0**	**0**		**0,000**		

TABELA 7.10: Resultados para o Problema C802, usando o algoritmo dos Filtros

O Filtro obtido na execução do algoritmo dos Filtros, por exemplo, em conjugação com o algoritmo de Pesquisa Coordenada e usando a medida N1, possui 6 aproximações à solução.

Estes pontos são os pontos não dominados do conjunto de iterações encontradas ao longo de todo o processo iterativo. O conjunto de iterações efectuadas encontra-se representado na FIGURA 7.7. Esta representação, bem como as seguintes, tem como objectivo ilustrar o comportamento do algoritmo.

Na FIGURA 7.12 os pontos do Filtro são apresentados através de quadrados e as iterações obtidas ao longo do processo através de losangos. Nesta figura o eixo das abcissas corresponde aos valores de f e o eixo das ordenadas aos valores de h.

A representação gráfica do Filtro, mais em pormenor pode ser vista na FIGURA 7.13. Com esta representação é possível identificar pontos da denominada Frente de Pareto obtida na execução do algoritmo para este problema.

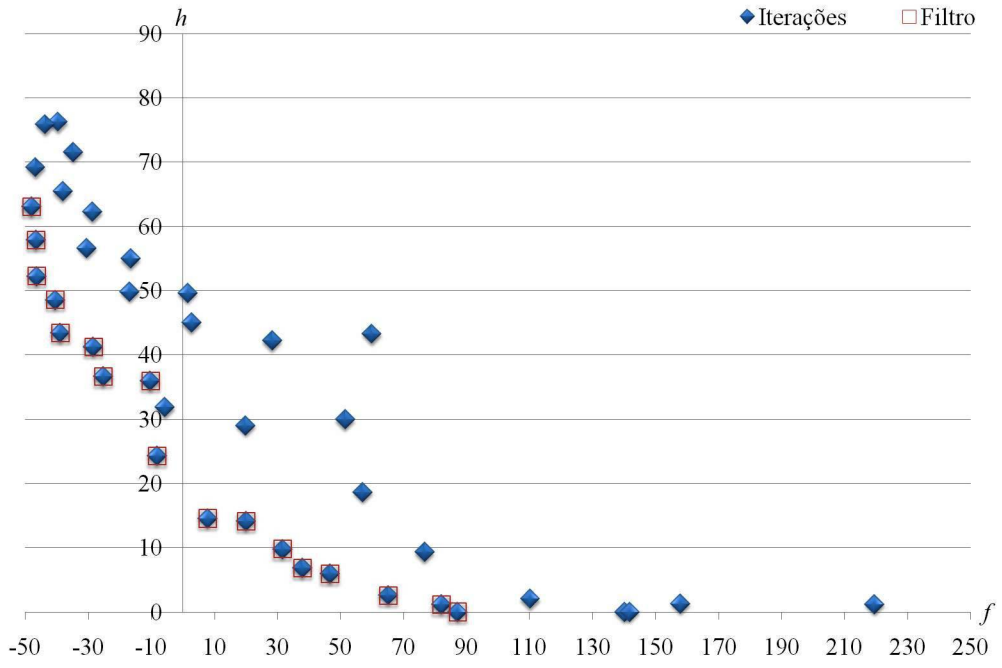

FIGURA 7.12: Iterações – Problema C802,
usando o algoritmo dos Filtros e o Método de Nelder-Mead

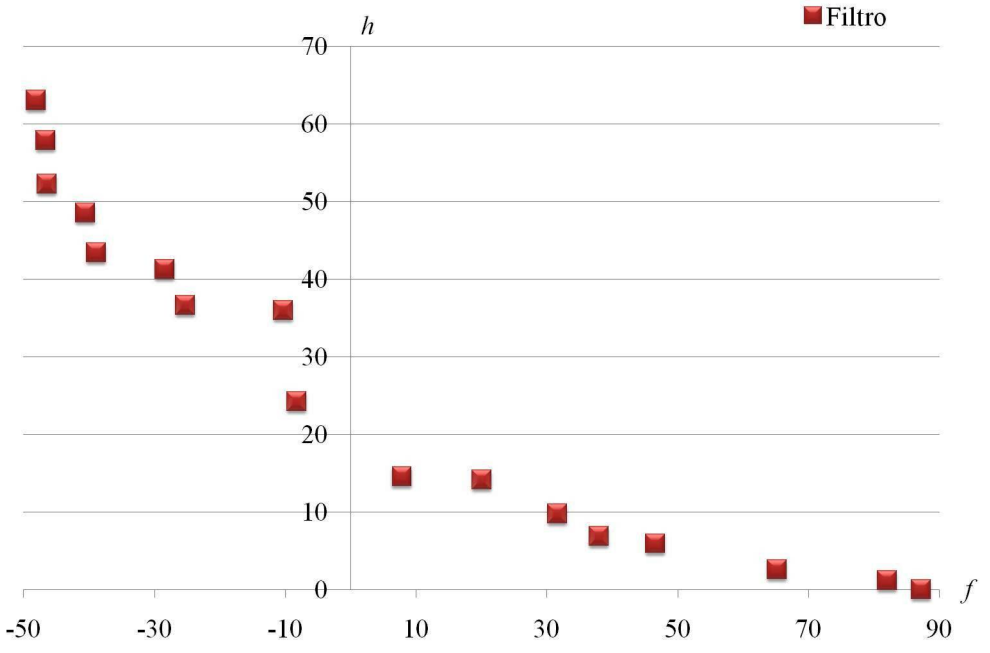

FIGURA 7.13: Filtro para o Problema C802,
usando o algoritmo dos Filtros e o Método de Nelder-Mead

Observando o comportamento dos valores da função objectivo e da violação das restrições ao longo do processo, na FIGURA 7.14, onde o eixo das abcissas representa o número da iteração, o eixo das ordenadas representa os valores das funções, os valores de h estão representados através de quadrados e os de f através de losangos, e sabendo que a melhor aproximação admissível à solução foi obtida na iteração 36 (destacada com uma linha vertical de maior espessura), pode verificar-se que houve um progresso nas iterações iniciais relativamente rápido, havendo uma diminuição dos valores da função objectivo, no entanto aumentaram os valores da violação das restrições. A partir das iterações 21 e 35 o valor da violação diminui e o valor da função objectivo aumenta, o que é consequência da optimização sem restrições de h, no entanto, a partir dessa iterações repete-se o procedimento inicial, com f a diminuir e h a aumentar.

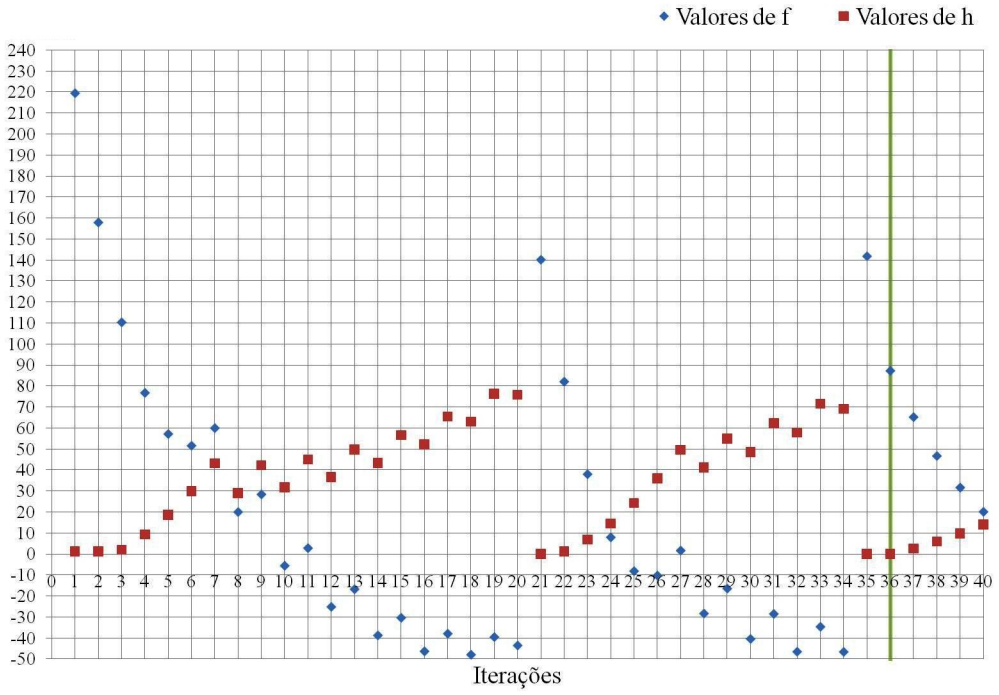

FIGURA 7.14: Comportamento dos valores da função objectivo e da violação das restrições ao longo do processo – MF/NM (Problema C802)

A execução do Método dos Filtros com 100 iterações não conduziu a melhores aproximações à solução dos as que já tinham sido encontradas.

7.3.2.2 Problema PA

Os Métodos de Penalidade/Barreira nem sempre permitem encontrar uma *Boa* aproximação à solução do Problema PA. Dependendo da conjugação entre funções de Penalidade/Barreira e métodos usados no processo interno podem obter-se aproximações admissíveis e não admissíveis *Boas*, *Médias* ou **Más**, de acordo com o apresentado na TABELA 7.11.

Problema		Ponto Inicial										Solução				
PA		x0=(0,0)				f(x0)=0						x*=(1,0)		f(x*)= -1		
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB/PB	802	2	-1	-1	0	Boa	-1	1	0	Boa	*	*	*	*	*
	CP/SP	6436	19	-1	-1	0	Boa	*	*	*	*	-1	0,0000153	18	0	Boa
	DP	4272	40	-1,2115	-1,1058	0,2115	Má	*	*	*	*	-1,2115	0,1058	40	0,2115	Má
	l1	1507	4	-1	-1	0	Boa	-1	3	0	Boa	-199	99	1	198	Má
HJ	EB/PB	133	2	-1	-1	0	Boa	-1	1	0	Boa	*	*	*	*	*
	CP/SP	1512	21	-0,9988	-0,9988	0,0012	Média	-0,9988	20	0,0012	Média	-1,0003	0,000153	14	0,0003	Média
	DP	3348	40	-1,2115	-1,1058	0,2115	Má	*	*	*	*	-1,2115	0,1058	40	0,2115	Má
	l1	4179	40	-1,01E+31	2,79E+42	1,01E+31	Má	*	*	*	*	-1,01E+31	5,07E+30	1	1,01E+31	Má
AA	EB/PB	1598	2	-1	-1	0	Boa	-1	1	0	Boa	*	*	*	*	*
	CP/SP	12628	19	-1	-1	0	Boa	*	*	*	*	-1	0,0000153	18	0	Boa
	DP	7522	40	-1,2115	-1,1058	0,2115	Má	*	*	*	*	-1,2115	0,1058	40	0,2115	Má
	l1	1826	5	-1	-1	0	Boa	-1	4	0	Boa	-99	196	3	98	Má
NM	EB/PB	572	2	-1	-1	0	Boa	-1	1	0	Boa	*	*	*	*	*
	CP/SP	2107	22	-0,9998	-0,9998	0,0002	Média	*	*	*	*	-0,9998	0,00000171	22	0,0002	Média
	DP	3230	40	-1,2115	-1,1058	0,2115	Má	*	*	*	*	-1,2115	0,1058	40	0,2115	Má
	l1	9464	40	-9,8406	3,24E+34	8,8406	Má	*	*	*	*	-9,84E+22	5,89E+22	1	9,84E+22	Má
SC	EB/PB	276	2	-1	-1	0	Boa	-1	1	0	Boa	*	*	*	*	*
	CP/SP	2916	22	-0,9957	-0,9957	0,0043	Média	*	*	*	*	-0,9957	0,000000535	21	0,0043	Média
	DP	4740	40	-1,2116	-1,1058	0,2116	Má	*	*	*	*	-1,2116	0,1058	40	0,2116	Má
	l1	770	4	-0,1	-0,1	0,9	Má	*	*	*	*	-0,1	0,00000144	3	0,9	Má
Minimos		133	2			0		-1	1	0		-1,01E+31	0,000000535	1	0	

TABELA 7.11: Resultados para o Problema PA, usando os algoritmos de Penalidade/Barreira

Para obter aproximações *Boas* admissíveis a função que se mostrou mais adequada para este problema foi a função EB, sendo o Problema PA um problema com um ponto inicial admissível esta função mostrou um bom desempenho. A função de Penalidade ℓ_1 também permitiu encontrar *Boas* aproximações, quando combinada com os Métodos CS e AA.

As únicas conjugações de métodos que permitiram encontrar *Boas* aproximações não admissíveis foram as conjugações do Método CS/SP com os Métodos CS e AA.

O Método dos Filtros devolve sempre uma *Boa* aproximação admissível à solução para o problema PA , independentemente da medida e do Método de Optimização sem Restrições usado no processo interno (ver TABELA 7.12).

Nestes resultados há a notar que, à excepção de quando se usa a medida NEB, todas as iterações foram aceites para o Filtro, apesar de não serem consideradas *Boas* aproximações de acordo com os critérios definidos. Isto aconteceu por não se ter definido um valor máximo para o valor da violação das restrições. Assim, como no processo iterativo foram

Problema		Ponto Inicial					Solução									
PA		x0=(0,0)			f(x0)=0		x*=(1,0)							f(x*)= -1		
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
Todos	N1/N2	42	42	40	-40,0000	39,0000	-1,0000	1	0,0000	Boa	-2,0000	1,0000	2	40	1,0000	Má
	NEB	42	42	40	-40,0000	infinito	-1,0000	1	0,0000	Boa	*	*	*	2	*	*
	NP	42	42	40	-40,0000	1521,0000	-1,0000	1	0,0000	Boa	-2,0000	1,0000	2	40	1,0000	Má
Minimos		42	42	40			-1,0000	1	0,0000		-2,0000	1,0000	2		1,0000	

TABELA 7.12: Resultados para o Problema PA, usando o algoritmo dos Filtros

sendo encontradas iterações com melhor valor de f o processo continuou sem restrições para os valores de h. Para evitar este inconveniente, de serem aceites no Filtro pontos que não sejam de interesse, pode estabelecer-se no início do processo, o valor pretendido para h_{max}. Este procedimento será testado para o Problema S226, para o qual acontece uma situação semelhante.

Note-se que o Problema PA é um problema linear, pelo que é um problema para o qual os métodos implementados não são os mais adequados, no entanto este problema foi usado como problema teste no trabalho de Audet and Dennis [8], referência base da Pesquisa Directa para Optimização com Restrições usando o Método dos Filtros, e tem sido usado nos diversos trabalhos apresentados ([38, 39, 41, 42, 111]) no sentido de fazer comparações de desempenho dos métodos. Desta forma, considerou-se mais pertinente testar a definição de um h_{max} no início do processo para outro problema, neste caso, para o Problema S226.

7.3.2.3 Problemas de Schittkowski

Problema S224

Usando o algoritmo de Penalidade/Barreira é possível encontrar *Boas* aproximações, admissíveis e não admissíveis, à solução do Problema S224, conforme se verifica nos dados da TABELA 7.13.

O Método de Penalidade ℓ_1 permite encontrar *Boas* aproximações admissíveis à solução usando qualquer um dos Métodos de Pesquisa Directa no processo interno.

Ao contrário do que seria de esperar, uma vez que o ponto inicial é admissível, a função Barreira Extrema nem sempre permite encontrar uma *Boa* aproximação admissível à

Problema		Ponto Inicial										Solução				
S224		x0=(0,1;0,1)				f(x0)=-8,77						x*=(4,4)		f(x*) =		-304
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB/PB	217	2	-293,17	-293,17	10,83	Má	-293,17	2	10,83	Má	*	*	*	*	*
	CP/SP	3493	26	-303,9985	-303,9985	0,0015	Média	*	*	*	*	-303,9985	2,38E-09	25	0,0015	Média
	DP	5510	40	-314,0364	-309,0515	10,0364	Má	*	*	*	*	-314,0364	4,9849	40	10,0364	Má
	I1	705	7	-304	-304	0	Boa	-304	7	0	Boa	-304,0009	0,000891	6	0,0009	Média
HJ	EB/PB	301	2	-303,5079	-303,5079	0,4921	Má	-303,5079	1	0,4921	Má	*	*	*	*	*
	CP/SP	2815	22	-303,989	-303,989	0,011	Má	-303,989	21	0,011	Má	-304,0046	0,0025	17	0,0046	Média
	DP	5168	40	-314,0351	-309,0514	10,0351	Má	*	*	*	*	-314,0351	4,9838	40	10,0351	Má
	I1	755	8	-304	-304	0	Boa	-304	7	0	Boa	-304,0312	0,0312	6	0,0312	Má
AA	EB	330	2	-304	-304	0	Boa	-304	1	0	Boa	*	*	*	*	*
	PB	432	2	-295,4023	-295,4023	8,5977	Má	-295,4023	2	8,5977	Má	*	*	*	*	*
	CP/SP	5548	27	-303,8675	-303,8675	0,1325	Má	*	*	*	*	-303,8675	1,53E-07	23	0,1325	Má
	DP	11045	40	-314,0347	-309,0505	10,0347	Má	*	*	*	*	-314,0347	4,9842	40	10,0347	Má
	I1	1272	7	-304	-304	0	Boa	-304	7	0	Boa	-304,0014	0,0014	6	0,0014	Média
NM	EB/PB	330	2	-304	-304	0	Boa	-304	1	0	Boa	*	*	*	*	*
	CP/SP	2103	26	-304	-304	0	Boa	*	*	*	*	-304	5,55E-06	25	0,0000	Boa
	DP	2762	40	-314,0373	-309,0515	10,0373	Má	*	*	*	*	-314,0373	4,9858	40	10,0373	Má
	I1	740	8	-304	-304	0	Boa	-304	7	0	Boa	-304,003	0,003	6	0,0030	Média
SC	EB	4370	40	-314,9693	infinito	infinito	Má	*	*	*	*	*	*	*	*	*
	PB	431	3	-413,419	-380,6154	109,419	Má	*	*	*	*	-413,419	32,8036	2	109,4190	Má
	CP/SP	4068	26	-304	-304	0	Boa	*	*	*	*	-304	5,89E-06	26	0,0000	Boa
	DP	4809	40	-314,0023	-309,0515	10,0023	Má	*	*	*	*	-314,0023	4,9508	40	10,0023	Má
	I1	1138	8	-304	-304	0	Boa	-304	7	0	Boa	-304,0029	0,0029	6	0,0029	Média
Minimos		217	2			0		-304	1	0		-413,419	2,38E-09	2	0	

TABELA 7.13: Resultados para o Problema S224, usando os algoritmos de Penalidade/Barreira

solução. Isso só acontece quando conjugada no processo interno com os algoritmos AA e NM.

Ainda é possível encontrar, usando os Métodos de Penalidade e Barreira implementados, algumas *Boas* aproximações não admissíveis à solução, nomeadamente com as conjugações CP/SP com NM e CP/SP com SC.

Com 11 das conjugações de métodos possíveis (30 no total, correspondentes à combinação de 6 funções de Penalidade/Barreira e 5 Métodos de Pesquisa Directa) não foi possível encontrar qualquer aproximação admissível à solução, o mesmo acontece com 6 conjugações de métodos, para aproximações não admissíveis à solução.

Problema		Ponto Inicial								Solução						
S224		x0=(0,1;0,1)				f(x0)=-8,77				x*=(4,4)					f(x*) =	-304
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
Todos	N1	41	41	40	-653,1700	68,8000	-196,7700	5	107,2300	Má	-222,3700	0,1000	6	22	81,6300	Má
	N2	41	41	40	-653,1700	41,9407	-196,7700	5	107,2300	Má	-222,3700	0,1000	6	22	81,6300	Má
	NEB	41	41	40	-653,1700	infinito	-196,7700	5	107,2300	Má	*	*	*	*	*	*
	NP	41	41	40	-653,1700	1759,0200	-196,7700	5	107,2300	Má	-222,3700	0,0100	6	22	81,6300	Má
Minimos		41	41	40			-196,7700	5	107,2300		-222,3700	0,0100	6		81,6300	

TABELA 7.14: Resultados para o Problema S224, usando o algoritmo dos Filtros

Usando o algoritmo dos Filtros, seja qual for a combinação de Métodos que se faça, não é possível encontrar qualquer *Boa* aproximação, como se pode ver na TABELA 7.14.

Problema		Ponto Inicial									Solução					
S224		x0=(0,1;0,1)			f(x0)=-8,77						x*=(4,4)				f(x*) = -304	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	N1	111	785	100	-54,3706	0	-289,2700	59	14,7300	Má	**-222,3700**	0,1000	**6**	37	**81,6300**	Má
	N2	101	715	100	-206,0700	0	-289,2700	59	14,7300	Má	**-222,3700**	0,1000	**6**	37	**81,6300**	Má
	NEB	101	6517	100	-428,7700	infinito	-196,7700	**5**	107,2300	Má	*	*	*	2	*	*
	NP	101	715	100	-206,0700	0	-289,2700	59	14,7300	Má	**-222,3700**	**0,0100**	**6**	45	**81,6300**	Má
HJ	N1	176	275	100	-171,5700	0	-196,7700	**5**	107,2300	Má	**-222,3700**	0,1000	**6**	28	**81,6300**	Má
	N2	176	274	100	-171,5700	0	-196,7700	**5**	107,2300	Má	**-222,3700**	0,1000	**6**	28	**81,6300**	Má
	NEB	101	12885	100	21.177,0650	infinito	-196,7700	**5**	107,2300	Má	*	*	*	2	*	*
	NP	176	274	100	-171,5700	0	-196,7700	**5**	107,2300	Má	**-222,3700**	**0,0100**	**6**	28	**81,6300**	Má
AA	N1	101	889	100	-225,2200	2,5	-243,0200	56	60,9800	Má	**-222,3700**	0,1000	**6**	42	**81,6300**	Má
	N2	101	690	100	-197,4200	0	-243,0200	56	60,9800	Má	**-222,3700**	0,1000	**6**	41	**81,6300**	Má
	NEB	101	12917	100	-428,7700	infinito	-196,7700	**5**	107,2300	Má	*	*	*	2	*	*
	NP	101	690	100	-197,4200	0	-243,0200	56	60,9800	Má	**-222,3700**	**0,0100**	**6**	41	**81,6300**	Má
NM	N1	206	286	100	-133,1200	0	-196,7700	**5**	107,2300	Má	**-222,3700**	0,1000	**6**	28	**81,6300**	Má
	N2	108	429	100	-133,1200	0	-274,2742	55	29,7258	Má	**-222,3700**	0,1000	**6**	42	**81,6300**	Má
	NEB	101	12981	100	-428,7700	infinito	-196,7700	**5**	107,2300	Má	*	*	*	2	*	*
	NP	108	429	100	-133,1200	0	-274,2742	55	29,7258	Má	**-222,3700**	**0,0100**	**6**	42	**81,6300**	Má
SC	N1	101	465	100	-271,1030	0	-276,6575	55	27,3425	Má	**-222,3700**	0,1000	**6**	45	**81,6300**	Má
	N2	101	474	100	-325,5481	0,95	**-299,4481**	99	**4,5519**	Má	**-222,3700**	0,1000	**6**	44	**81,6300**	Má
	NEB	101	1781	100	-428,7700	infinito	-196,7700	**5**	107,2300	Má	*	*	*	2	*	*
	NP	101	509	100	-319,4373	1,397368	-295,9662	99	8,0338	Má	**-222,3700**	**0,0100**	**6**	44	**81,6300**	Má
Minimos		**101**	**274**	**100**			**-299,4481**	**5**	**4,5519**		**-222,3700**	**0,0100**	**6**		**81,6300**	

TABELA 7.15: Resultados para o Problema S224, usando o algoritmo dos Filtros com 100 iterações

A execução do algoritmo dos Filtros com 100 como número máximo de iterações conduz a algumas melhorias nos resultados, mas ainda assim não são obtidas *Boas* aproximações à solução (ver a TABELA 7.15).

As melhorias obtidas são apenas na melhor aproximação admissível e apenas quando se usa uma das seguintes 11 combinações: N1 com CS, N2 com CS, NP com CS, N1 com AA, N2 com AA, NP com AA, N2 com NM, NP com NM, N1 com SC, N2 com SC e NP com SC. Em 9 das 20 combinações possíveis não foi possível melhorar estas aproximações. Houve também, à excepção de quando se usa a medida NEB, um aumento do número de pontos aceites e disponíveis no Filtro Final.

Relativamente ao número de avaliações de f e h necessárias no processo houve um aumento bastante significativo, no caso da função f houve um aumento correspondente ao aumento do número de iterações (aproximadamente de 40 para 100), mas no caso da função h o número de avaliações foi bastante superior. Note-se no entanto que estes valores não são muito diferentes, há caso que até são inferiores, aos que foram necessários fazer com o algoritmo PenBar.

É ainda de ter em conta a grande diferença do resultado obtido quando se usa a medida N2 em combinação com o Método SC, cujo valor da função objectivo na melhor aproximação admissível é bastante inferior ao que tinha sido obtido com apenas 40 iterações. Assim, o aumento no número de iterações efectuado no processo iterativo constituiu, para este

problema, uma boa opção para aproximar a solução e pode ser uma boa opção para melhorar resultados na resolução de outros problemas.

Problema S225

Os Métodos de Penalidade/Barreira permitem encontrar *Boas* aproximações, admissíveis e não admissíveis, à solução do Problema S225, conforme se verifica nos dados da TABELA 7.16.

Problema		Ponto Inicial										Solução				
S225		x0=(3;1,8)				f(x0)=12,24						x*=(1,1)			f(x*) = 2	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB/PB	**263**	**2**	2	2	**0**	Boa	**2**	2	0	Boa	*	*	*	*	*
	CP/SP	2307	21	2	2	**0**	Boa	*	*	*	*	2	7,63E-06	21	**0,0000**	Boa
	DP	4144	40	1,7885	1,8258	0,2115	Má	*	*	*	*	1,7842	0,0366	32	0,2158	Má
	l1	398	4	2	2	**0**	Boa	**2**	3	0	Boa	2	6,10E-06	4	**0,0000**	Boa
HJ	EB/PB	482	**2**	2,0059	2,0059	0,0059	Média	2,0059	**1**	0,0059	Média	*	*	*	*	*
	CP/SP	3148	40	1,9942	2293869,026	0,0058	Má	*	*	*	*	1,9942	8,50E-03	12	0,0058	Média
	DP	3395	40	1,7885	1,8258	0,2115	Má	*	*	*	*	1,7829	0,0366	30	0,2171	Má
	l1	2668	40	2	3709887,743	**0**	Má	*	*	*	*	2	1,35E-05	**2**	**0,0000**	Boa
AA	EB/PB	901	4	2	2,0002	**0**	Média	2,0002	4	0,0002	Média	*	*	*	*	*
	CP/SP	4094	20	2	2	**0**	Boa	*	*	*	*	2	**3,68E-11**	19	**0,0000**	Boa
	DP	11089	40	1,7881	1,8259	0,2119	Má	*	*	*	*	1,7847	0,0369	33	0,2153	Má
	l1	939	5	2	2	**0**	Boa	**2**	4	**0**	Boa	2	2,89E-09	5	**0,0000**	Boa
NM	EB/PB	319	**2**	2	2	**0**	Boa	**2**	**1**	**0**	Boa	*	*	*	*	*
	CP/SP	1779	20	2	2	**0**	Boa	*	*	*	*	2	1,16E-05	20	**0,0000**	Boa
	DP	3100	40	1,7885	1,8258	0,2115	Má	*	*	*	*	1,7849	0,0366	33	0,2151	Má
	l1	450	4	2	2	**0**	Boa	**2**	4	**0**	Boa	2	3,12E-05	3	**0,0000**	Boa
SC	EB	4259	40	3,225	infinito	infinito	Má	*	*	*	*	*	*	*	*	*
	PB	325	3	1,4751	1,6581	0,5249	Má	*	*	*	*	**1,4751**	0,1829	**2**	0,5249	Má
	CP/SP	3162	18	2	2	**0**	Boa	*	*	*	*	2	7,64E-06	17	**0,0000**	Boa
	DP	8157	40	1,7894	1,8262	0,2106	Má	*	*	*	*	1,7834	0,0359	29	0,2166	Má
	l1	467	4	2	2	**0**	Boa	*	*	*	*	2	5,84E-06	3	**0,0000**	Boa
Minimos		**263**	**2**			**0**		**2**	**1**	**0**		**1,4751**	**3,7E-11**	**2**	**0**	

TABELA 7.16: Resultados para o Problema S225, usando os algoritmos de Penalidade/Barreira

Usando os Métodos CS, AA ou NM no processo interno é possível encontrar *Boas* aproximações admissíveis à solução com os Métodos de Penalidade/Barreira EB, PB e ℓ_1. Usando o Método HJ em conjugação com as funções EB ou PB é possível encontrar uma *Boa* aproximação admissível à solução e usando o Método SC não é possível encontrar nenhuma, apenas se encontram aproximações não admissíveis *Boas* e *Más*.

Neste problema o ponto inicial é admissível e usando o método EB foi possível encontrar *Boas* aproximações à solução, à excepção de quando se usa o SC no processo interno.

Ainda é possível encontrar, usando os Métodos de Penalidade e Barreira implementados, algumas *Boas* aproximações não admissíveis à solução, de acordo com os critérios definidos.

Com as 30 conjugações de métodos possíveis foi sempre encontrada uma aproximação (admissível ou não admissível) à solução, à excepção de quando se usa o Método SC no processo interno conjugado com a função EB no processo externo.

Problema		Ponto Inicial									Solução					
S225		x0=(3;1,8)					f(x0)=12,24				x*=(1,1)				f(x*) = 2	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS/AA	Todas	80	**41**	**40**	7,2400	0	7,2400	3	5,2400	Má	*	*	*	1	*	*
HJ	N1	55	1162	**40**	2,0315	0,0156	**2,0000**	39	**0,0000**	Boa	**2,6900**	0,3000	**9**	4	**0,6900**	Má
	N2	55	1162	**40**	2,0315	0,0156	**2,0000**	39	**0,0000**	Boa	**2,6900**	0,3000	**9**	5	**0,6900**	Má
	NEB	**48**	9629	**40**	8.253,4291	infinito	12,2400	**1**	10,2400	Má	*	*	*	2	*	*
	NP	55	1162	**40**	2,0315	2,44E-04	**2,0000**	39	**0,0000**	Boa	**2,6900**	**0,0900**	**9**	5	**0,6900**	Má
NM/SC	Todas	132	**41**	**40**	11,8400	0	11,8400	3	9,8400	Má	*	*	*	1	*	*
Minimos		**48**	**41**	**40**			**2,0000**	**1**	**0,0000**		**2,6900**	**0,0900**	**9**		**0,6900**	

TABELA 7.17: Resultados para o Problema S225, usando o algoritmo dos Filtros

Usando o algoritmo dos Filtros, só foi possível encontrar *Boas* aproximações usando o algoritmo HJ e as medidas para a violação das restrições N1, N2 e NP, como se pode ver na TABELA 7.17.

Além disso, usando os restantes métodos o Filtro Final obtido é de pequena dimensão, pelo que, mesmo que o problema seja relaxável, não são uma boa opção para o resolver.

Problema S226

Problema		Ponto Inicial											Solução					
S226		x0=(0.8,0,05)						f(x0)=-0.04					x*=(2^0,5/2;2^0,5/2)				f(x*) = -0,5	
Métodos		Última Iteração						Melhor Admissível					Melhor não admissível					
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class		f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB	162	**2**	-0,4593	-0,4593	0,0407	Má	-0,4593	2	0,0407	Má		*	*	*	*	*
	PB	12478	40	-0,4593	-0,45934	0,0407	Má	-0,4593	14	0,0407	Má		*	*	*	*	*
	CP/SP	1937	15	-0,5	-0,5	**0**	Boa	*	*	*	*		-0,5	4,88E-06	15	**0,0000**	Boa
	DP	4288	40	-0,5195	-0,5098	0,0195	Má	*	*	*	*		-0,5195	0,0098	40	0,0195	Má
	l1	183	**2**	-0,498	-0,498	0,002	Média	-0,498	2	0,002	Média		*	*	*	*	*
HJ	EB/PB	229	**2**	-0,46	-0,46	0,04	Má	-0,46	**1**	0,04	Má		*	*	*	*	*
	CP/SP	1442	15	-0,4997	-0,4997	0,0003	Média	-0,4997	14	0,0003	Média		-0,5001	9,00E-05	11	0,0001	Boa
	DP	3636	40	-0,5198	-0,5098	0,0198	Má	*	*	*	*		**-0,5198**	0,0099	39	0,0198	Má
	l1	**136**	**2**	-0,4978	-0,4978	0,0022	Média	-0,4978	**1**	0,0022	Média		*	*	*	*	*
AA	EB/PB	320	**2**	-0,4593	-0,4593	0,0407	Má	-0,4593	2	0,0407	Má		*	*	*	*	*
	CP/SP	3341	15	-0,4999	-0,4999	1E-04	Boa	*	*	*	*		-0,4999	6,91E-06	15	0,0001	Boa
	DP	10177	40	-0,5195	-0,5098	0,0195	Má	*	*	*	*		-0,5195	0,0098	40	0,0195	Má
	l1	303	**2**	-0,4946	-0,4946	0,0054	Média	-0,4946	2	0,0054	Média		-0,4946	4,22E-06	**1**	0,0054	Média
NM	EB/PB	263	**2**	-0,4963	-0,4963	0,0037	Média	-0,4963	**1**	0,0037	Média		*	*	*	*	*
	CP/SP	1312	15	-0,5	-0,5	**0**	Boa	*	*	*	*		-0,5	2,36E-06	15	**0,0000**	Boa
	DP	3326	40	-0,5195	-0,5098	0,0195	Má	*	*	*	*		-0,5195	0,0098	40	0,0195	Má
	l1	341	3	-0,5	-0,5	**0**	Boa	**-0,5**	2	**0**	Boa		*	*	*	*	*
SC	EB/PB	213	**2**	-0,04	-0,04	0,46	Má	-0,04	**1**	0,46	Má		*	*	*	*	*
	CP/SP	2207	15	-0,5	-0,5	**0**	Boa	*	*	*	*		-0,5	7,70E-07	14	**0,0000**	Boa
	DP	6245	40	-0,5195	-0,5098	0,0195	Má	*	*	*	*		-0,5195	0,0098	40	0,0195	Má
	l1	649	4	-0,5	-0,5	**0**	Boa	**-0,5**	3	**0**	Boa		-0,5	**1,88E-09**	**1**	**0,0000**	Boa
Minimos		**136**	**2**			**0**		**-0,5**	**1**	**0**			**-0,5198**	**1,9E-09**	**1**	**0**	

TABELA 7.18: Resultados para o Problema S226, usando os algoritmos de Penalidade/Barreira

Usando o algoritmo de Penalidade/Barreira é possível encontrar *Boas* aproximações, admissíveis e não admissíveis, à solução do Problema S226, de acordo com os critérios definidos e conforme se verifica nos dados da TABELA 7.18.

O Método de Penalidade ℓ_1, em conjugação com os Métodos NM e SC, permite encontrar *Boas* aproximações admissíveis à solução, com um número bastante reduzido de iterações e de avaliações da função Φ, enquanto as restantes *Boas* aproximações são não admissíveis.

Neste caso a função Barreira Extrema permitiu encontrar algumas aproximações admissíveis, mas nunca permitiu encontrar uma *Boa* aproximação à solução.

Problema		Ponto Inicial					Solução									
S226		x0=(0.8,0,05)			f(x0)=-0.04		x*=(2^0,5/2;2^0,5/2)				f(x*) = -0,5					
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
Todos	N1/N2	41	41	40	-2,0400	1663,6425	*	*	*	*	-0,0900	2,2425	1	41	0,4100	Má
	NEB	42	42	40	-2,0400	infinito	*	*	*	*	*	*	*	2	*	*
	NP	42	42	40	-2,0400	2767706,3678	*	*	*	*	-0,0900	5,0288	1	41	0,4100	Má
Minimos		41	41	40			0,0000	0	0,0000		-0,0900	2,2425	1		0,4100	

TABELA 7.19: Resultados para o Problema S226, usando o algoritmo dos Filtros

Usando o algoritmo dos Filtros, não foi possível encontrar *Boas* aproximações, como se pode ver na TABELA 7.19.

Tal como aconteceu com o Problema PA, na execução do algoritmo dos Filtros com as medidas N1, N2 e NP, foram aceites todas as iterações para o Filtro para o Problema S226, apesar de não serem consideradas *Boas* aproximações de acordo com os critérios definidos. Estas iterações foram sendo aceites porque o processo de optimização de f foi repetido, independentemente do valor de h.

Para evitar este inconveniente uma possibilidade é estabelecer logo no início do processo iterativo um valor máximo para a violação das restrições, ou seja, para h_{max}.

A representação gráfica do Filtro, sem fazer nenhuma restrição ao valor de h, por exemplo usando o algoritmo AA e a medida N1, é a apresentada na FIGURA 7.15, onde os pontos do Filtro são apresentados com um losango e as iterações com um quadrado. Nesta figura o eixo das abcissas corresponde aos valores de f e o eixo das ordenadas aos valores de h. Assim, pode ver-se nessa figura que as aproximações encontradas e os pontos do Filtro coincidem.

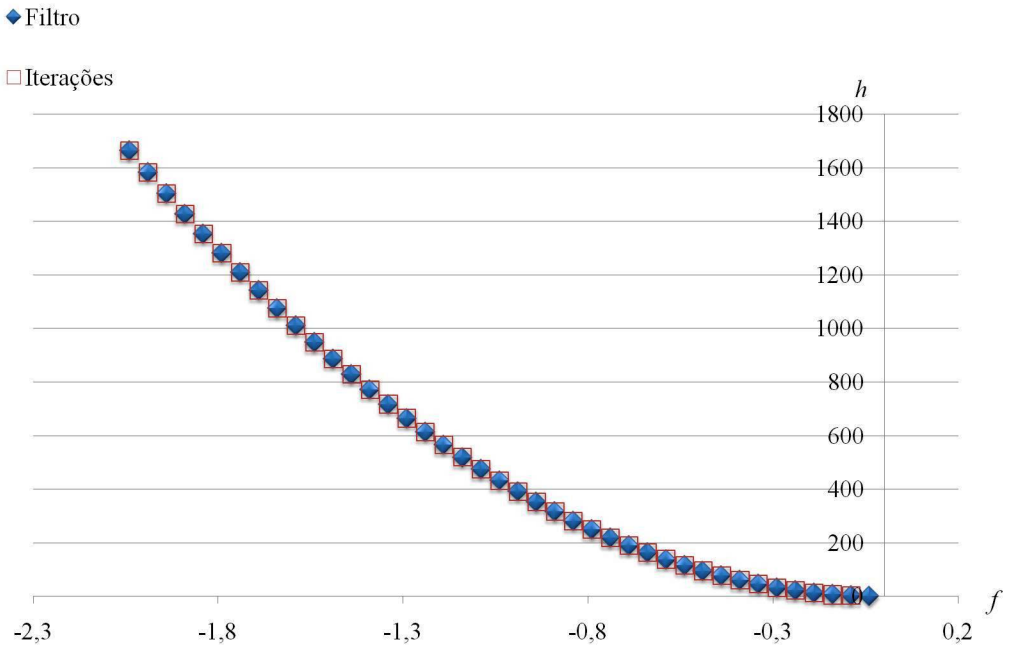

FIGURA 7.15: Filtro para o Problema S226,
usando o algoritmo dos Filtros e a versão implementada dos métodos de Audet

Considerando, por exemplo $h_{max} = 2$, os resultados obtidos são bastante diferentes (ver TABELA 7.20).

Problema		Ponto Inicial									Solução					
S226		x0=(0.8,0,05)				f(x0)=-0.04					x*=(2^0,5/2;2^0,5/2)				f(x*) = -0,5	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	N1/N2	49	2828	40	-0,0402	0	-0,4400	12	0,0600	Má	-0,8400	0,7425	7	4	0,3400	Má
	NEB	49	3241	40	-0,0900	infinito	-0,4400	11	0,0600	Má	*	*	*	1	*	*
	NP	51	2825	40	-0,0900	5,0288	-0,4400	10	0,0600	Má	-0,8400	0,5513	7	3	0,3400	Má
HJ	N1/N2	66	838	40	-6,5779E-13	0	*	*	*	*	-0,8400	0,7425	2	2	0,3400	Má
	NEB	41	10428	40	-4.335,3197	infinito	*	*	*	*	*	*	*	2	*	*
	NP	66	838	40	-6,5779E-13	0	*	*	*	*	-0,8400	0,5513	2	2	0,3400	Má
AA	N1/N2	45	5562	40	-0,0463	0	-0,2743	15	0,2257	Má	-0,3600	0,2201	16	7	0,1400	Má
	NEB	49	6421	40	-0,0900	infinito	-0,4743	12	0,0257	Má	*	*	*	1	*	*
	NP	43	5562	40	-0,0525	0	-0,4743	27	0,0257	Má	-0,5484	0,0429	28	7	0,0484	Má
NM	N1	48	1234	40	-0,0087	0	-0,4326	16	0,0674	Má	-0,5486	0,4006	20	8	0,0486	Má
	N2	55	1117	40	-0,0652	0	-0,2066	22	0,2934	Má	-0,2772	0,0434	23	8	0,2228	Má
	NEB	41	10506	40	-23,4900	infinito	*	*	*	*	*	*	*	1	*	*
	NP	62	1276	40	-0,0635	0	-0,4759	19	0,0241	Má	-0,3352	0,0789	14	5	0,1648	Má
SC	N1/N2	47	1021	40	-1,9228	3,4560	-0,4767	30	0,0233	Má	-0,6455	0,3700	31	8	0,1455	Má
	NEB	41	1406	40	-23,4900	infinito	*	*	*	*	*	*	*	1	*	*
	NP	59	1347	40	-1,8905	11,1841	-0,4613	11	0,0387	Má	-0,6698	0,1587	18	5	0,1698	Má
Minimos		41	838	40			-0,4767	10	0,0233		-0,8400	0,0429	2		0,0484	

TABELA 7.20: Resultados para o Problema S226,
usando o algoritmo dos Filtros com $h_{max} = 2$

Apesar de não se obterem *Boas* aproximações à solução, de acordo com os critérios definidos, já é possível encontrar aproximações admissíveis e, as aproximações não admissíveis possuem valores menores da função objectivo e da violação das restrições.

Com esta alteração do parâmetro h_{max} o Filtro já aceita menos aproximações, com todas as combinações de Métodos de Pesquisa Directa e medidas. A representação gráfica do Filtro, por exemplo usando o algoritmo AA e a medida N1, é a apresentada na FIGURA 7.16, onde os pontos do Filtro são apresentados com um losango e as iterações com um quadrado.

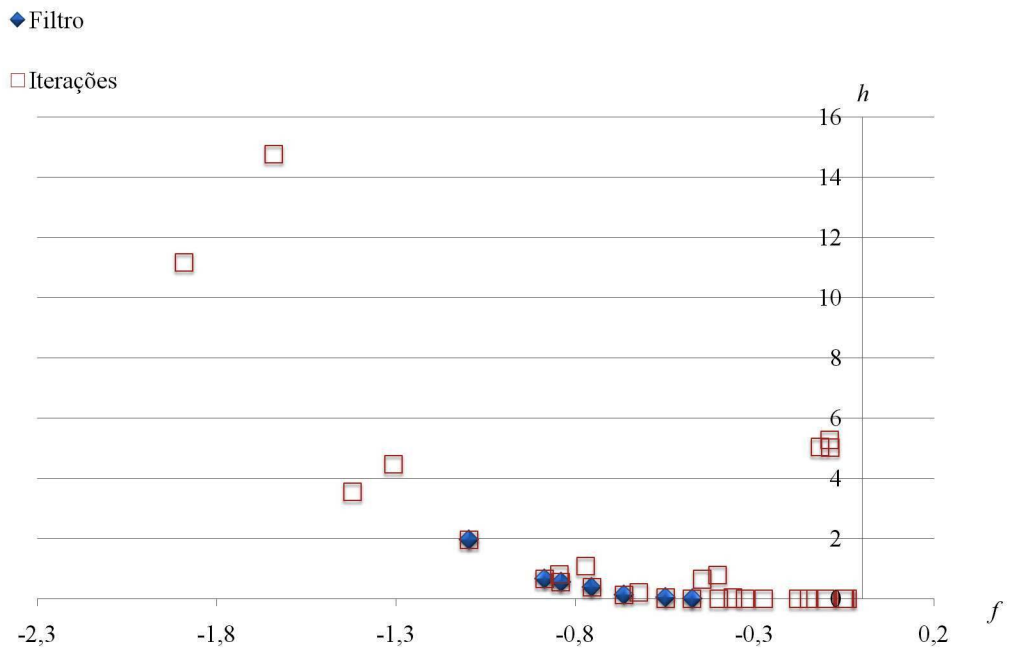

FIGURA 7.16: Filtro para o Problema S226, usando o algoritmo dos Filtros e a versão implementada dos métodos de Audet, com $h_{max} = 2$

Neste caso só foram aceites no Filtro iterações cujo valor da violação das restrições não fosse superior a $h_{max} = 2$, pelo que o Filtro ficou apenas com 7 aproximações. Esta poderá ser uma boa alternativa para problemas relaxáveis, mas com um certo limite para a tolerância da violação das restrições.

Problema S227

Os resultados obtidos, usando os Métodos de Penalidade/Barreira e o Método dos Filtros, para o Problema S227 são apresentados nas TABELAS 7.21 e 7.22.

Ambos os algoritmos permitem encontrar *Boas* aproximações à solução. O algoritmo de Penalidade/Barreira permite encontrar várias aproximações admissíveis e não admissíveis. O Método de Penalidade ℓ_1 permitiu encontrar *Boas* aproximações admissíveis à

Problema		Ponto Inicial										Solução				
S227		x0=(0,5;0,5)				f(x0)=2,5						x*=(1,1)			f(x*) = 1	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB	336	3	1	1	**0,0000**	Boa	**1**	3	0	Boa	*	*	*	*	*
	PB	5667	40	1	1	**0,0000**	Boa	**1**	34	0	Boa	*	*	*	*	*
	CP/SP	1766	18	1	1	**0,0000**	Boa	*	*	*	*	1	1,91E-06	17	**0,0000**	Boa
	DP	3795	40	0,8995	0,9452	**0,1005**	Má	*	*	*	*	0,8995	0,0457	40	0,1005	Má
	I1	799	**2**	1	1	**0,0000**	Boa	**1**	**1**	0	Boa	*	*	*	*	*
HJ	EB	613	**2**	1	1	0,0030	Média	1	**1**	0,003	Média	*	*	*	*	*
	PB	3121	40	1	1	0,0030	Média	1	**1**	0,003	Média	*	*	*	*	*
	CP/SP	3083	40	0,9989	325.400	0,0011	Má	*	*	*	*	0,997	2,10E-03	10	0,0030	Média
	DP	3577	40	0,8984	0,9453	0,1016	Má	*	*	*	*	0,8966	0,0467	37	0,1034	Má
	I1	**135**	**2**	1	1	**0,0000**	Boa	**1**	**1**	**0**	Boa	*	*	*	*	*
AA	EB	1600	**2**	1	1	**0,0000**	Boa	**1**	**1**	**0**	Boa	*	*	*	*	*
	PB	32076	40	1	1	**0,0000**	Boa	**1**	**1**	**0**	Boa	*	*	*	*	*
	CP/SP	2830	17	1	1	**0,0000**	Boa	*	*	*	*	1	1,19E-05	17	**0,0000**	Boa
	DP	6590	40	0,8995	0,9452	**0,1005**	Má	*	*	*	*	0,8995	0,0456	40	0,1005	Má
	I1	1581	**2**	1	1	**0,0000**	Boa	**1**	**1**	**0**	Boa	*	*	*	*	*
NM	EB/PB	477	3	1	1	**0,0000**	Boa	**1**	3	**0**	Boa	*	*	*	*	*
	CP/SP	1803	20	1	1	0,0005	Média	*	*	*	*	1,0005	**1,18E-06**	19	0,0005	Média
	DP	3110	40	0,8995	0,9452	**0,1005**	Má	*	*	*	*	0,8995	0,0457	40	0,1005	Má
	I1	284	3	1	1	**0,0000**	Boa	**1**	2	**0**	Boa	**0,801**	1,72E-01	**1**	0,1990	Má
SC	EB/PB	213	**2**	2,5	2,5	1,5000	Má	2,5	**1**	1,5	Má	*	*	*	*	*
	CP/SP	2090	18	1	1	**0,0000**	Boa	*	*	*	*	1	6,20E-06	17	**0,0000**	Boa
	DP	4894	40	0,8995	0,9452	**0,1005**	Má	*	*	*	*	0,8995	0,0457	40	0,1005	Má
	I1	667	4	1	1	**0,0000**	Boa	**1**	3	**0**	Boa	1	3,31E-05	2	**0,0000**	Boa
Minimos		**135**	**2**			**0,0000**		**1**	**1**	**0**		**0,801**	**1,18E-06**	**1**	**0**	

TABELA 7.21: Resultados para o Problema S227, usando os algoritmos de Penalidade/Barreira

Problema		Ponto Inicial								Solução						
S228		x0=(0,0)				f(x0)=0				x*=(0,-3)					f(x*) = -3	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	Todas	106	**41**	**40**	-1,0000	0,0000	-1,0000	**3**	2,0000	Má	*	*	*	1	*	*
HJ	Todas	**119**	**41**	**40**	-2,0000	0,0000	**-2,0000**	**3**	**1,0000**	Má	*	*	*	1	*	*
AA	Todas	158	**41**	**40**	-1,5000	0,0000	-1,5	**3**	1,5	Má	*	*	*	1	*	*
NM/SC	Todas	**119**	**41**	**40**	1,32E-23	0,0000	*	*	*	*	*	*	*	1	*	*
Minimos		**119**	**41**	**40**			**-2**	**3**	**1**		**0**	**0**	**0**		**0,000**	

TABELA 7.22: Resultados para o Problema S227, usando o algoritmo dos Filtros

solução em conjugação com qualquer um dos Métodos de Pesquisa Directa. Os Métodos de Barreira (EB e PB) também permitiram encontrar aproximações admissíveis, no entanto, em conjugação com os Métodos HJ e SC são *Médias* e *Más*, respectivamente.

O algoritmo dos Filtros mostrou-se mais eficiente em conjugação com os Métodos CS e HJ e conjugando o Método AA com a medida NEB.

Note-se que, por exemplo, usando o Método SC e a medida NP o Filtro que se obtém tem dimensão 16. Na FIGURA 7.17 pode ver-se a representação dos pontos obtidos ao longo de todo o processo iterativo, representados através de quadrados, e os pontos do Filtro final através de losangos. A zona sombreada a cinzento representa a região admissível, os eixos das abcissas e das ordenadas correspondem às coordenadas das iterações, algumas das curvas de nível da função objectivo são também apresentadas.

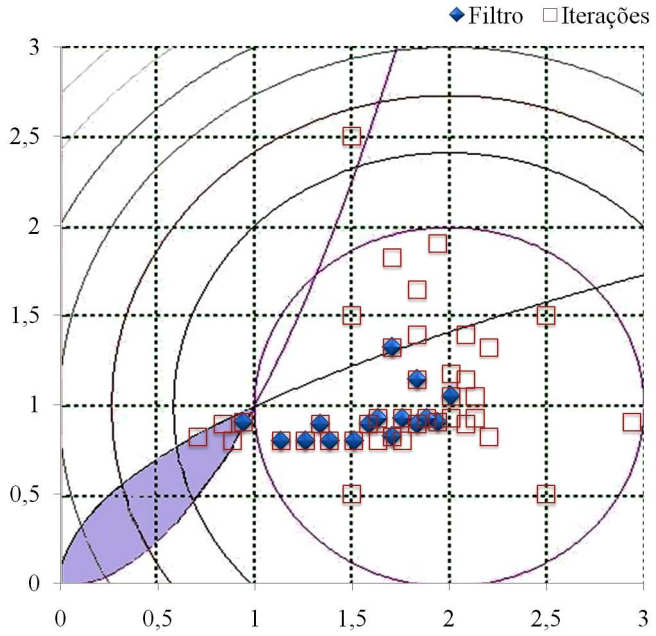

FIGURA 7.17: Posição das aproximações à solução – MF/SC (Problema S227)

Nesta representação pode ver-se que apenas um dos pontos do Filtro é admissível, os restantes pertencem todos à região não admissível. Sendo os pontos que constituem o Filtro pontos que possuem a propriedade de não dominância, se o problema for relaxável, qualquer um deles poderá ser uma boa aproximação para resolver o problema, balanceando os valores de f e h que se podem obter.

Problema S228

AS TABELAS 7.23 e 7.24 apresentam os resultados da execução dos algoritmos para o Problema S228.

Os Métodos de Penalidade/Barreira permitem encontrar todos *Boas* aproximações à solução, à excepção de quando se usa a função DP. Mas, mesmo neste caso, apesar das aproximações serem não admissíveis, são aproximações *Médias*, de acordo com os critérios definidos.

No caso do Método dos Filtros acontece exactamente o oposto, nenhuma combinação de métodos implementados permite encontrar aproximações *Boas*.

Problema		Ponto Inicial										Solução				
S228		x0=(0,0)				f(x0)=0						x*=(0,-3)			f(x*) = -3	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB/I1	805	2	-3	-3	0,0000	Boa	-3	1	0	Boa	*	*	*	*	*
	PB	16081	40	-3	-3	0,0000	Boa	-3	1	0	Boa	*	*	*	*	*
	CP/SP	915	11	-3	-3	0,0000	Boa	*	*	*	*	-3	6,11E-06	11	0,0000	Boa
	DP	3285	40	-3,0037	-3,0018	0,0037	Média	*	*	*	*	-3,0037	0,0018	40	0,0037	Média
HJ	EB/PB/I1	134	2	-3	-3	0,0000	Boa	-3	1	0	Boa	*	*	*	*	*
	CP/SP	656	9	-3	-3	0,0000	Boa	*	*	*	*	-3	1,05E-09	8	0,0000	Boa
	DP	2779	40	-3,004	-3,0018	0,0040	Média	*	*	*	*	-3,004	0,002	32	0,0040	Média
AA	EB/PB/I1	1605	2	-3	-3	0,0000	Boa	-3	1	0	Boa	*	*	*	*	*
	CP/SP	1807	12	-3	-3	0,0000	Boa	*	*	*	*	-3	4,11E-07	11	0,0000	Boa
	DP	6015	40	-3,0037	-3,0018	0,0037	Média	*	*	*	*	-3,0037	0,0019	40	0,0037	Média
NM	EB/PB/I1	270	2	-3	-3	0,0000	Boa	-3	1	0	Boa	*	*	*	*	*
	CP/SP	1136	13	-3	-3	0,0000	Boa	*	*	*	*	-3	5,82E-07	10	0,0000	Boa
	DP	2946	40	-3,0037	-3,0018	0,0037	Média	*	*	*	*	-3,0037	0,0018	40	0,0037	Média
SC	EB/PB/I1	535	3	-3	-3	0,0000	Boa	-3	2	0	Boa	*	*	*	*	*
	CP/SP	1298	11	-3	-3	0,0000	Boa	*	*	*	*	-3	6,59E-06	11	0,0000	Boa
	DP	3228	40	-3,0037	-3,0018	0,0037	Média	*	*	*	*	-3,0037	0,0018	40	0,0037	Média
Minimos		134	2			0,0000		-3	1	0		-3,004	1,05E-09	8	0	

TABELA 7.23: Resultados para o Problema S228, usando os algoritmos de Penalidade/Barreira

Problema		Ponto Inicial								Solução					
S228		x0=(0,0)			f(x0)=0					x*=(0,-3)				f(x*) = -3	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível				
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	Todas	106	41	40	-1,0000	0,0000	-1,0000	3	2,0000	Má	*	*	*	1	*	*
HJ	Todas	119	41	40	-2,0000	0,0000	-2,0000	3	1,0000	Má	*	*	*	1	*	*
AA	Todas	158	41	40	-1,5000	0,0000	-1,5	3	1,5	Má	*	*	*	1	*	*
NM/SC	Todas	119	41	40	1,32E-23	0,0000	*	*	*	*	*	*	*	1	*	*
Minimos		119	41	40			-2	3	1		0	0	0		0,000	

TABELA 7.24: Resultados para o Problema S228, usando o algoritmo dos Filtros

Se se considerar, por exemplo, o Método HJ e a medida N2, apesar de terem sido feitas 40 iterações, os pontos são muito próximos, não havendo, consequentemente, grande evolução dos valores das funções envolvidas.

Na FIGURA 7.17 pode ver-se a representação dos pontos obtidos ao longo de todo o processo iterativo a representado através de quadrados e o ponto do Filtro Final representado através de um losango. A zona sombreada a cinzento representa a região admissível, os eixos das abcissas e das ordenadas representam as coordenadas da iteração, as curvas de nível da função objectivo são também apresentadas.

Fazendo no Método de HJ, por exemplo, a alteração do parâmetro tamanho do passo para $\alpha = 2$ os resultados obtidos já são bastante diferentes, sendo encontrada, na iteração 11, a melhor aproximação admissível à solução $x_{kf} = (0.0; -3.0)$ com valor da função $f(x_{kf}) = -3.0$ e a melhor aproximação à solução não admissível $x_{ki} = (0.0, -3.25)$ com $f(x_{ki}) = -3.25$ e $h(x_{ki}) = 1.5625$, na iteração 9.

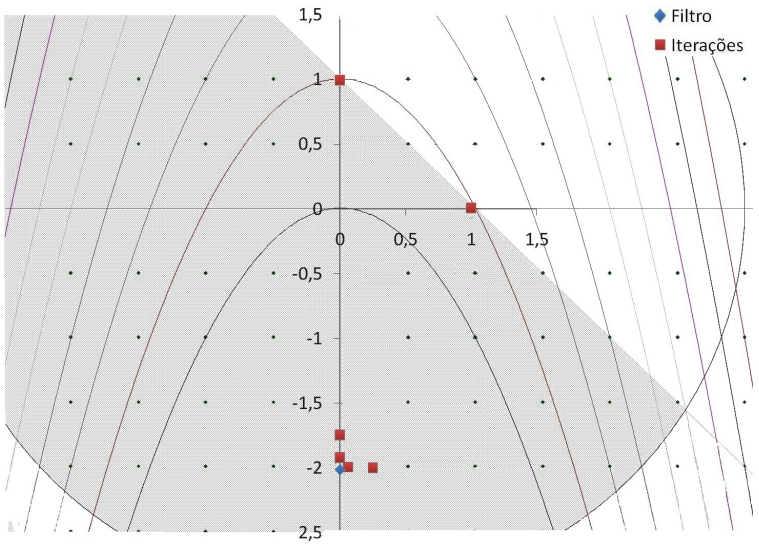

FIGURA 7.18: Posição das aproximações à solução – MF/HJ (Problema S228)

Neste caso o Filtro tem dimensão 13. A representação gráfica dos pontos obtidos ao longo de todo o processo iterativo, através de losangos, e os pontos do Filtro Final, através de quadrados, pode ser vista na FIGURA 7.19. A zona sombreada a cinzento representa a região admissível, os eixos das abcissas e das ordenadas representam as coordenadas da iteração, algumas das curvas de nível da função objectivo são também apresentadas.

Problema S231

De acordo com os dados da TABELA 7.25, foi possível encontrar *Boas* aproximações à solução do Problema S231 usando o algoritmo de Penalidade/Barreira, nomeadamente usando no processo interno os Métodos NM e SC.

Problema		Ponto Inicial										Solução				
S231		x0=(-1,2;1)				f(x0)=24,2						x*=(1,1)			f(x*) = 0	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	Todas	10274	40	1,49E-04	1,49E-04	0,0001	Média	1,49E-04	40	0,000149	Média	*	*	*	*	*
HJ	Todas	2963	6	0,0359	0,0359	0,0359	Má	0,0359	1	0,0359	Má	*	*	*	*	*
AA	Todas	12640	40	1,34E-04	1,34E-04	0,0001	Média	1,34E-04	40	0,000134	Média	*	*	*	*	*
NM	Todas	264	2	2,23E-11	2,23E-11	2,23E-11	Boa	2,23E-11	1	2,23E-11	Boa	*	*	*	*	*
SC	EB/PB	1084	6	2,66E-11	2,66E-11	2,66E-11	Boa	2,66E-11	5	2,66E-11	Boa	*	*	*	*	*
	CP/SP/DP/l1	953	6	1,77E-11	1,77E-11	1,77E-11	Boa	1,77E-11	5	1,77E-11	Boa	*	*	*	*	*
Minimos		264	2			0,0000		1,77E-11	1	1,77E-11		0	0	0	0	

TABELA 7.25: Resultados para o Problema S231, usando os algoritmos de Penalidade/Barreira

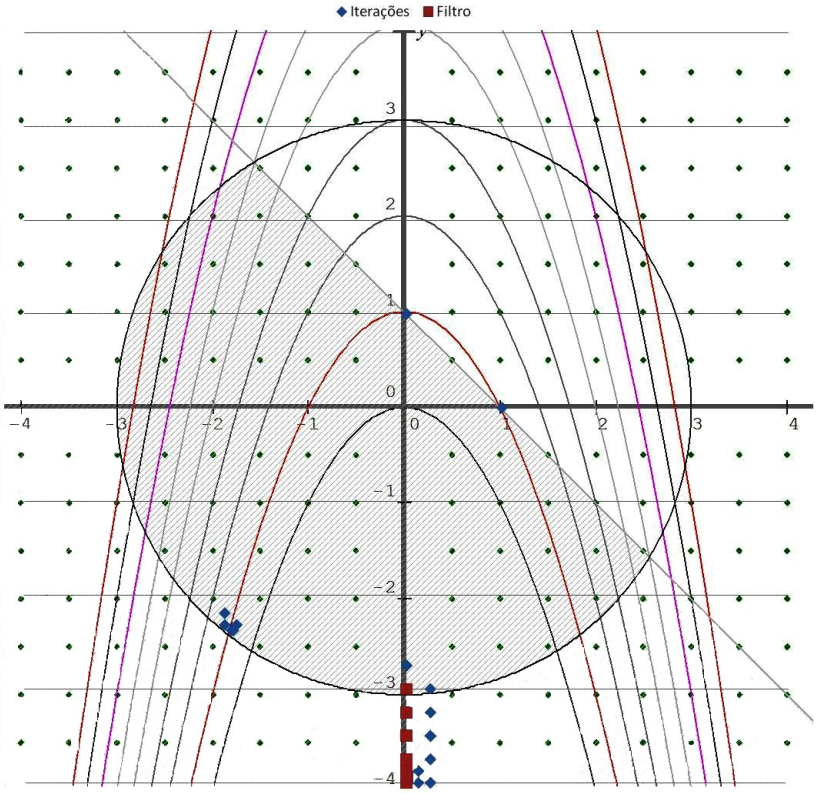

FIGURA 7.19: Posição das aproximações à solução – MF/HJ (Problema S228), com $\alpha = 2$

No entanto, de acordo com os dados da TABELA 7.26 não foi possível encontrar nenhuma usando o algoritmo dos Filtros. Contudo, basta considerar no Método dos Filtros o tamanho do passo inicial $\rho = 2.2$ que o algoritmo, por exemplo, em conjugação com o Método CS, encontra uma *Boas* aproximação admissível à solução, $x_{kf} = (1.0000; 1.0000)$, com $f(x_{kf}) = 1.98E - 29$.

Problema		**Ponto Inicial**									**Solução**					
S231		x0=(-1,2;1)					f(x0)=24,2				x*=(1,1)				f(x*) = 0	
Métodos		**Última Iteração**					**Melhor Admissível**				**Melhor não admissível**					
IP	EP	**fEv**	**hEV**	**k**	**f(xk)**	**h(xk)**	**f(xkf)**	**kf**	**|f(x*)-f(xkf)|**	**class**	**f(xki)**	**h(xki)**	**ki**	**Dim F**	**|f(x*)-f(xki)|**	**class**
CS/HJ	Todos	**106**	**41**	**40**	24,2000	0	**4,0634**	**4**	**4,0634**	Má	*	*	*	1	*	*
AA	Todos	158	**41**	**40**	24,2000	0	**4,0634**	**4**	**4,0634**	Má	*	*	*	1	*	*
NM/SC	Todos	132	**41**	**40**	24,2000	0	**4,0634**	**4**	**4,0634**	Má	*	*	*	1	*	*
Minimos		**106**	**41**	**40**			**4,0634**	**4**	**4,0634**		**0**	**0**	**0**		**0**	

TABELA 7.26: Resultados para o Problema S231, usando o algoritmo dos Filtros

Problema S233

Problema		Ponto Inicial										Solução				
S233		x0=(1,2;1)				f(x0)=19,4						x*=(1,1)			f(x*) = 0	
Métodos		Última Iteração				Melhor Admissível						Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	Todas	13849	40	2,16E-04	2,16E-04	0,000216	Média	2,16E-04	40	0,000216	Média	*	*	*	*	*
HJ	PB	2651	40	0,0396	0,0396	0,0396	Má	0,0396	1	0,0396	Má	*	*	*	*	*
	Outras	143	2	0,0396	0,0396	0,0396	Má	0,0396	1	0,0396	Má	*	*	*	*	*
AA	EB	12866	40	8,45E-06	8,45E-06	0,00000845	Boa	8,45E-06	40	0,00000845	Boa	*	*	*	*	*
	Outras	12836	40	8,45E-06	8,45E-06	0,00000845	Boa	8,45E-06	40	0,00000845	Boa	*	*	*	*	*
NM	PB	3043	40	1,26E-11	1,26E-11	1,26E-11	Boa	1,26E-11	3	1,26E-11	Boa	*	*	*	*	*
	Outras	343	4	1,26E-11	1,26E-11	1,26E-11	Boa	1,26E-11	3	1,26E-11	Boa	*	*	*	*	*
SC	PB	5820	40	1,05E-10	1,05E-10	1,05E-10	Boa	1,05E-10	14	1,05E-10	Boa	*	*	*	*	*
	Outras	3545	15	1,05E-10	1,05E-10	1,05E-10	Boa	1,05E-10	14	1,05E-10	Boa	*	*	*	*	*
Minimos		143	2			1,26E-11		1,26E-11	1	1,26E-11		0	0	0	0	

TABELA 7.27: Resultados para o Problema S233, usando os algoritmos de Penalidade/Barreira

Problema		Ponto Inicial										Solução					
S233		x0=(1,2;1)					f(x0)=19,4					x*=(1,1)				f(x*) = 0	
Métodos		Última Iteração					Melhor Admissível					Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS/HJ	Todos	46	41	40	0,462527	0	0,4625	39	0,462512398	Má	*	*	*	1	*	*
AA	N1/N2	76	4011	40	0,8000	0,2100	0,5281	6	0,5281	Má	4,4031	0,0475	4	2	4,4031	Má
	NEB	71	4011	40	0,8000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	76	4011	40	0,8000	0,0441	0,5281	6	0,5281	Má	4,4031	0,0023	4	2	4,4031	Má
NM/SC	Todos	48	41	40	0,4625	0	0,4625	39	0,462512398	Má	*	*	*	1	*	*
Minimos		46	41	40			0,4625	6	0,462512398		4,4031	0,0023	4		4,403125	

TABELA 7.28: Resultados para o Problema S233, usando o algoritmo dos Filtros

Os resultados nas TABELAS 7.27 e 7.28 mostram que sucede para o Problema S233 o mesmo que aconteceu para o Problema S231: o algoritmo de Penalidade/Barreira permite encontrar *Boas* aproximações à solução, nomeadamente usando no processo interno os Métodos NM e SC, mas não foi possível encontrar nenhuma usando o algoritmo dos Filtros, usando os parâmetros por defeito.

Problema S234

Problema		Ponto Inicial										Solução					
S234		x0=(0,5;0.5)				f(x0)=-0,5						x*=(0,2;0,2)			f(x*) = -0,8		
Métodos		Última Iteração					Melhor Admissível					Melhor não admissível					
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class	
CS	EB	223	2	-0,8	-0,8	0	Boa	-0,8	2	0	Boa	*	*	*	*	*	
	PB	239	2	-1,207	-1,0238	0,407	Má	*	*	*	*	-1,207	0,1832	2	0,4070	Má	
	CP/SP	2211	19	-0,8	-0,8	0	Boa	*	*	*	*	-8,00E-01	9,54E-07	19	0,0000	Boa	
	DP	4193	40	-0,8557	-0,8278	0,0557	Má	*	*	*	*	-0,8556	0,0278	40	0,0556	Má	
	l1	553	3	-0,8	-0,8	0	Boa	-0,8	3	0	Boa	-8,00E-01	3,81E-07	2	0,0000	Boa	
HJ	EB	317	2	-0,7996	-0,7996	0,0004	Média	-0,7996	1	0,0004	Média	*	*	*	*	*	
	PB	165	2	-1,207	-1,0238	0,407	Má	*	*	*	*	-1,207	0,1832	1	0,4070	Má	
	CP/SP	1072	14	-0,8	-0,8	0	Boa	*	*	*	*	-8,00E-01	9,96E-06	13	0,0000	Boa	
	DP	3303	40	-0,8557	-0,8278	0,0557	Má	*	*	*	*	-0,8557	0,0279	40	0,0557	Má	
	l1	743	8	-0,7995	-0,7995	0,0005	Média	-0,7995	7	0,0005	Média	-8,00E-01	4,36E-05	2	0,0000	Boa	
AA	EB	395	2	-0,8	-0,8	0	Boa	-0,8	2	0	Boa	*	*	*	*	*	
	PB	400	2	-1,207	-1,0238	0,407	Má	*	*	*	*	-1,207	0,1832	2	0,4070	Má	
	CP/SP	6118	19	-0,8	-0,8	0	Boa	*	*	*	*	-8,00E-01	9,54E-07	19	0,0000	Boa	
	DP	7668	40	-0,8556	-0,8278	0,0556	Má	*	*	*	*	-0,8556	0,0278	40	0,0556	Má	
	l1	342	2	-0,8	-0,8	0	Boa	-0,8	2	0	Boa	-8,00E-01	1,14E-06	1	0,0000	Boa	
NM	EB	283	2	-0,8	-0,8	0	Boa	-0,8	1	0	Boa	*	*	*	*	*	
	PB	249	2	-1,207	-1,0238	0,407	Má	*	*	*	*	-1,207	0,1832	1	0,4070	Má	
	CP/SP	1771	17	-0,8	-0,8	0	Boa	*	*	*	*	-8,00E-01	8,48E-07	15	0,0000	Boa	
	DP	3773	40	-0,8556	-0,8278	0,0556	Má	*	*	*	*	-0,8556	0,0278	40	0,0556	Má	
	l1	525	3	-0,8	-0,8	0	Boa	-0,8	2	0	Boa	-8,00E-01	1,26E-05	1	0,0000	Boa	
SC	EB	213	2	-0,5	-0,5	0,3	Má	-0,5	1	0,3	Má	*	*	*	*	*	
	PB	468	2	-1,207	-1,0238	0,407	Má	*	*	*	*	-1,207	0,1832	2	0,4070	Má	
	CP/SP	3893	18	-0,8	-0,8	0	Boa	*	*	*	*	-8,00E-01	1,29E-06	17	0,0000	Boa	
	DP	8396	40	-0,8557	-0,8279	0,0557	Má	*	*	*	*	-0,8557	0,0278	40	0,0557	Má	
	l1	696	4	-0,8	-0,8	0	Boa	-0,8	3	0	Boa	-8,00E-01	4,41E-06	1	0,0000	Boa	
Minimos		165	2			0		-0,8	1	0		-1,207	3,81E-07	1	0		

TABELA 7.29: Resultados para o Problema S234, usando os algoritmos de Penalidade/Barreira

Problema		Ponto Inicial									Solução						
S234		x0=(0,5;0.5)					f(x0)=-0,5				x*=(0,2;0,2)					f(x*) = -0,8	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível						
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class	
CS	Todas	91	1243	40	-0,5000	0,0000	*	*	*	*	*	*	*	1	*	*	
HJ	N1/N2	48	251	40	-0,7461	0,2000	-0,8000	3	1,08E-05	Boa	-0,8113	0,2125	39	13	0,0113	Má	
	NEB	41	8031	40	-47,0446	infinito	*	*	*	*	*	*	*	2	*	*	
	NP	48	251	40	-0,7461	0,0400	-0,8000	3	1,08E-05	Boa	-0,8113	0,0402	39	13	0,011264038	Má	
AA	N1	46	2409	40	-0,7461	0,2000	-0,7461	9	0,0539	Má	-0,8113	0,2125	39	13	0,011264038	Má	
	N2	46	2409	40	-0,7461	0,2000	-0,7461	9	0,0539	Má	-0,8113	0,2004	39	13	0,011264038	Má	
	NEB	71	5623	40	-1,5000	infinito	*	*	*	*	*	*	*	2	*	*	
	NP	46	2409	40	-0,7461	0,0400	-0,7461	9	0,0539	Má	-0,8113	0,0402	39	13	0,011264038	Má	
NM	N1	104	619	40	-0,5000	0,0000	*	*	*	*	*	*	*	1	*	*	
	N2/NP	111	474	40	-0,5000	0,0000	-0,7216	6	0,0784	Má	*	*	*	1	*	*	
	NEB	41	10506	40	20736,5000	infinito	*	*	*	*	*	*	*	1	*	*	
SC	N1/N2	104	693	40	-0,5000	0,0000	-0,7615	6	0,0385	Má	*	*	*	1	*	*	
	NEB	41	1406	40	20736,5000	infinito	*	*	*	*	*	*	*	1	*	*	
	NP	90	1012	40	-0,5000	0,0000	-0,7763	9	0,0237	Má	*	*	*	1	*	*	
Minimos		41	251	40			-0,8000	3	1,08E-05		-0,8113	0,040156	39		0,011264038		

TABELA 7.30: Resultados para o Problema S234, usando o algoritmo dos Filtros

Os resultados obtidos, usando os Métodos de Penalidade/Barreira e o Método dos Filtros, para o Problema S234 são apresentados nas TABELAS 7.29 e 7.30.

Ambas as metodologias permitem encontrar *Boas* aproximações à solução. O algoritmo de Penalidade/Barreira permite encontrar várias aproximações admissíveis e não admissíveis. Os Métodos de Penalidade/Barreira EB e ℓ_1 foram os que permitiram encontrar

Boas aproximações admissíveis à solução em conjugação com qualquer um dos Métodos de Pesquisa Directa, à excepção do EB conjugado com o SC.

O algoritmo dos Filtros mostrou-se mais eficiente em conjugação com o método HJ e usando as medidas N1, N2 e NP.

Note-se que, por exemplo, usando o Método AA e a medida N1, o Filtro que se obtém tem dimensão 13. Na FIGURA 7.20 pode ver-se a representação dos pontos obtidos ao longo de todo o processo iterativo, representados através de quadrados, e os pontos do Filtro Final, representados através de losangos. A zona sombreada a rosa representa a região admissível, os eixos das abcissas e das ordenadas representam as coordenadas da iteração, algumas das curvas de nível da função objectivo são também representadas nesta figura.

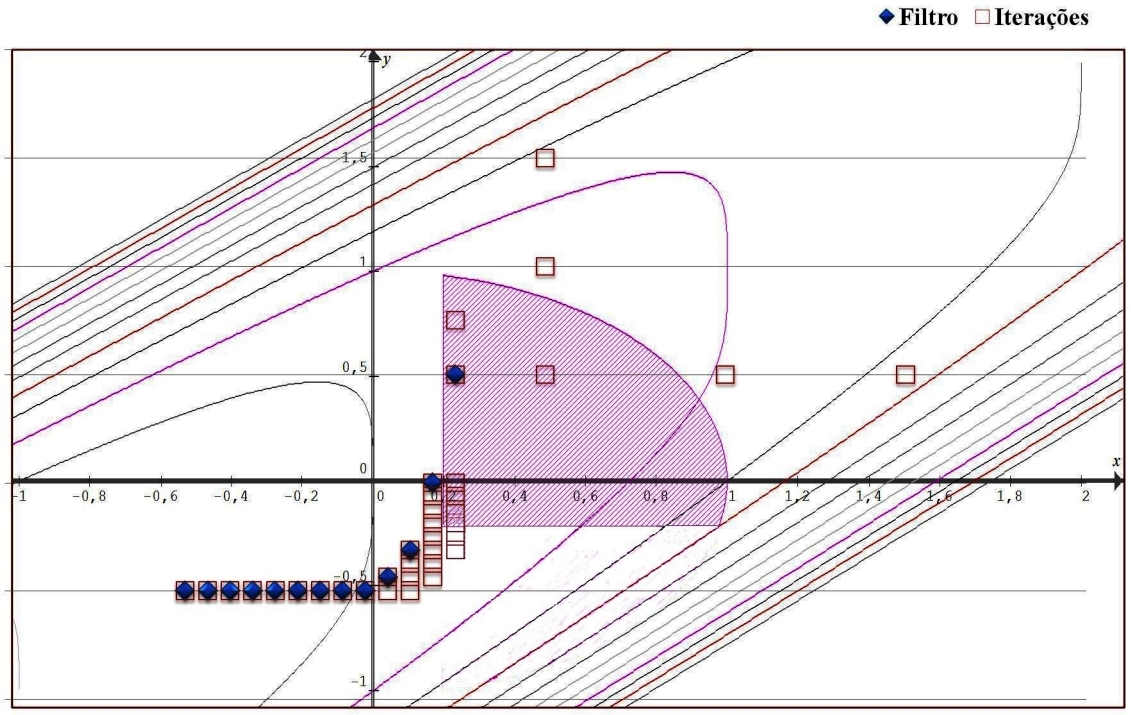

FIGURA 7.20: Posição das aproximações à solução – MF/AA (Problema S234)

Problema S249

Os resultados obtidos na resolução do Problema S249, usando os Métodos de Penalidade/Barreira e o Método dos Filtros, são apresentados nas TABELAS 7.31 e 7.32.

À semelhança do que aconteceu para o Problema anterior, ambos os algoritmos permitem encontrar *Boas* aproximações à solução. O algoritmo de Penalidade/Barreira permite

Problema		Ponto Inicial										Solução				
S249		x0=(1,1,1)				f(x0)=3						x*=(1,0,0)			f(x*) = 1	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB	1203	2	1	1	0	Boa	1	1	0	Boa	*	*	*	*	*
	PB	23819	40	0,25	0,25	0,75	Má	*	*	*	*	0,2500	0,25	1	0,7500	Má
	CP/SP	2577	18	1	1	0	Boa	*	*	*	*	1,0000	7,63E-06	18	0,0000	Boa
	DP	4999	40	0,8499	0,9219	0,1501	Má	*	*	*	*	0,8499	0,072	40	0,1501	Má
	l1	1796	3	1	1	0	Boa	1	2	0	Boa	0,2500	5,00E-01	1	0,7500	Má
HJ	EB	179	2	1	1	0	Boa	1	1	0	Boa	*	*	*	*	*
	PB	180	2	0,25	0,5	0,75	Má	*	*	*	*	0,2500	0,25	1	0,7500	Má
	CP/SP	1664	17	1,0003	1,0003	0,0003	Média	1,0003	16	0,0003	Média	0,9994	4,31E-04	13	0,0006	Média
	DP	3860	40	0,8497	0,9219	0,1503	Má	*	*	*	*	0,8497	0,0722	40	0,1503	Má
	l1	268	3	1	1	0	Boa	1	2	0	Boa	0,2500	5,00E-01	1	0,7500	Má
AA	EB	1524	2	1	1	0	Boa	1	2	0	Boa	*	*	*	*	*
	PB	932	3	0,2502	infinito	infinito	Má	*	*	*	*	0,2502	0,2498	3	0,7498	Má
	CP/SP	4980	18	1	1	0	Boa	*	*	*	*	1,0000	7,63E-06	18	0,0000	Boa
	DP	9339	40	0,85	0,9219	0,15	Má	*	*	*	*	0,8500	0,072	40	0,1500	Má
	l1	1738	4	1	1	0	Boa	1	4	0	Boa	0,2499	5,00E-01	1	0,7501	Má
NM	EB	573	3	1	1	0	Boa	1	2	0	Boa	*	*	*	*	*
	PB	657	4	0,25	0,5	0,75	Má	*	*	*	*	0,2500	0,25	3	0,7500	Má
	CP/SP	2499	18	1	1	0	Boa	*	*	*	*	1,0000	7,53E-06	18	0,0000	Boa
	DP	5383	40	0,85	0,9219	0,15	Má	*	*	*	*	0,8500	0,072	40	0,1500	Má
	l1	805	5	1	1	0	Boa	1	4	0	Boa	1,0000	9,21E-08	3	0,0000	Boa
SC	EB	1401	5	1,0006	1,0006	0,0006	Média	1,0006	4	0,0006	Média	*	*	*	*	*
	PB	542	3	0,25	infinito	infinito	Má	*	*	*	*	0,2500	0,25	3	0,7500	Má
	CP/SP	5648	24	1	1	0	Boa	*	*	*	*	1,0000	1,57E-08	24	0,0000	Boa
	DP	9414	40	0,85	0,9219	0,15	Má	*	*	*	*	0,8500	0,072	40	0,1500	Má
	l1	1157	4	1	1	0	Boa	1	3	0	Boa	1,0000	7,17E-06	2	0,0000	Boa
Minimos		179	2			0		1	1	0		0,2499	1,57E-08	1	0	

TABELA 7.31: Resultados para o Problema S249, usando os algoritmos de Penalidade/Barreira

Problema		Ponto Inicial									Solução					
S249		x0=(1,1,1)					f(x0)=3				x*=(1,0,0)				f(x*) = 1	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	N1/N2	47	637	40	2,0067	0,9180	*	*	*	*	2,76563	0,1250	11	30	1,7656	Má
	NEB	56	3021	40	3,0000	0,0000	*	*	*	*	*	*	*	2	*	*
	NP	47	637	40	2,0067	0,8427	*	*	*	*	2,76563	0,0156	11	30	1,7656	Má
HJ	N1/N2	771	226	40	7,72E-05	0,9912	1	12	0	Boa	0,76563	0,1250	11	25	0,2344	Má
	NEB	91	936	40	64,9444	0,0000	*	*	*	*	*	*	*	2	*	*
	NP	71	226	40	7,72E-05	0,9825	1	12	0	Boa	0,76563	0,0156	11	25	0,2344	Má
AA	N1/N2	47	1231	40	2,0067	0,9180	*	*	*	*	2,76563	0,1250	11	30	1,7656	Má
	NEB	56	5991	40	3,0000	0,0000	*	*	*	*	*	*	*	2	*	*
	NP	47	1231	40	2,0067	0,8427	*	*	*	*	2,76563	0,0156	11	30	1,7656	Má
NM/SC	Todos	151	41	40	3,0000	0,0000	*	*	*	*	*	*	*	1	*	*
Minimos		47	41	40			1	12	0		0,76563	0,0156	11		0,2344	

TABELA 7.32: Resultados para o Problema S249, usando o algoritmo dos Filtros

encontrar várias aproximações admissíveis e não admissíveis. Os Métodos de Penalidade/Barreira EB e ℓ_1 foram os que permitiram encontrar *Boas* aproximações admissíveis à solução em conjugação com qualquer um dos Métodos de Pesquisa Directa, à excepção do EB conjugado com o SC.

No caso do algoritmo dos Filtros os melhores resultados também são obtidos com o Método dos Filtros em conjugação com o método HJ e usando as medidas N1, N2 e NP.

Problema S264

Os resultados obtidos na resolução do Problema S264, apresentados nas TABELAS 7.33 e 7.34, não constituem *Boas* aproximações à solução, de acordo com os critérios definidos.

Problema		Ponto Inicial										Solução					
S264		x0=(0,0,0,0)				f(x0)=0						x*=(0,1,2,-1)				f(x*) = -44	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível					
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class	
CS	EB	854	2	-29	-29	15	Má	-29	2	15	Má	*	*	*	*	*	
	PB	8496	40	-46,1089	-44,456	2,1089	Má	*	*	*	*	-46,1089	1,6531	40	2,109	Má	
	CP/SP	5213	20	-42,5079	-42,508	1,4921	Má	*	*	*	*	-42,5079	3,60E-06	19	1,492	Má	
	DP	12242	40	-42,8032	-42,658	1,1968	Má	*	*	*	*	-42,8032	0,1449	40	1,197	Má	
	l1	715	4	-42,4538	-42,454	1,5462	Má	-42,4538	3	1,5462	Má	-42,4538	4,45E-06	4	1,546	Má	
HJ	EB	**221**	**2**	-29	-29	15	Má	-29	**1**	15	Má	*	*	*	*	*	
	PB	54952	40	245887,1	infinito	infinito	Má	*	*	*	*	-46,1108	1,6549	**1**	2,111	Má	
	CP/SP	4819	21	-42,5059	-42,506	1,4941	Má	-42,5059	20	1,4941	Má	-42,5072	3,00E-04	16	1,493	Má	
	DP	7932	40	-42,8033	-42,658	**1,1967**	Má	*	*	*	*	-42,8033	0,1452	40	**1,197**	Má	
	l1	1603	7	-42,4505	-42,451	1,5495	Má	-42,4505	6	1,5495	Má	-42,4519	5,60E-04	3	1,548	Má	
AA	EB	3182	**2**	-36,25	-36,25	**7,75**	Má	-36,25	**1**	7,75	Má	*	*	*	*	*	
	PB	5235	12	-42,4219	infinito	infinito	Má	*	*	*	*	-42,4219	0,362	12	1,578	Má	
	CP/SP	6431	19	-42,4677	-42,468	1,5323	Má	*	*	*	*	-42,4677	1,01E-05	19	1,532	Má	
	DP	17024	40	-42,8026	-42,658	1,1974	Má	*	*	*	*	-42,8026	0,145	40	1,197	Má	
	l1	2109	5	-42,3651	-42,365	1,6349	Má	-42,365	4	1,635	Má	-42,3651	3,42E-06	5	1,635	Má	
NM	EB	1112	3	-42,5113	-42,511	1,4887	Má	-42,5113	3	1,4887	Má	*	*	*	*	*	
	PB	1396	3	-46,1103	-44,456	2,1103	Má	*	*	*	*	-46,1103	1,6545	3	2,110	Má	
	CP/SP	4723	24	-42,5111	-42,511	**1,4889**	Má	*	*	*	*	-42,5111	3,10E-07	23	1,489	Má	
	DP	7125	40	-42,8031	-42,658	1,1969	Má	*	*	*	*	-42,8031	0,1448	40	1,197	Má	
	l1	932	5	-42,5115	-42,512	1,4885	Má	*	*	*	*	-42,5115	**2,34E-08**	4	1,489	Má	
SC	EB	492	3	-38,8493	-38,849	5,1507	Má	-38,8493	2	5,1507	Má	*	*	*	*	*	
	PB	359	**2**	-46,1672	-44,446	2,1672	Má	*	*	*	*	**-46,1672**	1,7215	**1**	2,167	Má	
	CP/SP	7171	21	-42,5111	-42,511	**1,4889**	Má	-42,5107	19	1,4893	Má	-42,5111	8,83E-06	20	1,489	Má	
	DP	9231	40	-42,803	-42,658	1,197	Má	*	*	*	*	-42,8030	0,1448	40	1,197	Má	
	l1	1868	7	-42,5114	-42,511	**1,4886**	Má	**-42,5114**	6	**1,4886**	Má	-42,5114	2,58E-04	5	1,489	Má	
Minimos		**221**	**2**			**1,1967**		**-42,5114**	**1**	**1,4886**		**-46,1672**	**2,34E-08**	**1**	**1,1967**		

TABELA 7.33: Resultados para o Problema S264, usando os algoritmos de Penalidade/Barreira

Problema		Ponto Inicial									Solução						
S264		x0=(0,0,0,0)					f(x0)=0				x*=(0,1,2,-1)					f(x*) = -44	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível						
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class	
CS	N1	**41**	1595	**40**	4,0000	0,0000	-26,0000	26	18,0000	Má	-30,0000	2,0000	27	8	14,0000	Má	
	N2	**41**	1595	**40**	4,0000	0,0000	-26,0000	26	18,0000	Má	-30,0000	2,0000	27	9	14,0000	Má	
	NEB	**41**	2444	**40**	-23,0000	infinito	**-4,0000**	**1**	**40,0000**	Má	*	*	*	2	*	*	
	NP	**41**	1595	**40**	4,0000	0,0000	-26,0000	26	18,0000	Má	-30,0000	4,0000	27	9	14,0000	Má	
HJ	N1	**41**	**162**	**40**	-28,4867	6,6383	-4,0000	**1**	40,0000	Má	-21,9046	**0,1151**	31	14	22,0954	Má	
	N2	**41**	**162**	**40**	-28,4867	6,6383	-4,0000	**1**	40,0000	Má	-21,9046	**0,1151**	31	13	22,0954	Má	
	NEB	**41**	4238	**40**	1212,2208	infinito	-4,0000	**1**	40,0000	Má	*	*	*	2	*	*	
	NP	**41**	**162**	**40**	-28,4867	44,0673	-4,0000	**1**	40,0000	Má	-21,9046	**0,0132**	31	13	22,0954	Má	
AA	N1	**41**	2965	**40**	-17,2500	0,0000	-28,2500	26	15,7500	Má	-32,2500	1,7500	27	9	11,7500	Má	
	N2	**41**	2965	**40**	-17,2500	0,0000	-28,2500	26	15,7500	Má	-32,2500	1,7500	27	9	11,7500	Má	
	NEB	**41**	4844	**40**	-23,0000	infinito	-4,0000	**1**	40,0000	Má	*	*	*	2	*	*	
	NP	**41**	2965	**40**	-17,2500	0,0000	-28,2500	26	15,7500	Má	-32,2500	3,0625	27	10	11,7500	Má	
NM	N1	**41**	461	**40**	-57,5326	61,4647	-25,8979	27	18,1021	Má	-6,0000	3,0000	**2**	11	38,0000	Má	
	N2	**41**	461	**40**	-57,5326	36,6620	-25,8979	27	18,1021	Má	-6,0000	3,0000	**2**	11	38,0000	Má	
	NEB	**41**	6692	**40**	-23,0000	infinito	-4,0000	**1**	40,0000	Má	*	*	*	2	*	*	
	NP	**41**	461	**40**	-57,5326	1344,1043	-25,8979	27	18,1021	Má	-6,0000	9,0000	**2**	11	38,0000	Má	
SC	N1	**41**	430	**40**	-65,2484	88,0663	**-39,6673**	26	**4,3327**	Má	-6,0000	3,0000	**2**	11	38,0000	Má	
	N2	**41**	466	**40**	-65,0448	55,1571	-32,5634	26	11,4366	Má	**-36,7014**	0,3054	27	11	**7,2986**	Má	
	NEB	**41**	392	**40**	-23,0000	infinito	-4,0000	**1**	40,0000	Má	*	*	*	2	*	*	
	NP	**41**	470	**40**	-60,4619	1674,4147	-24,9076	26	19,0924	Má	-27,7118	2,3894	27	11	16,2882	Má	
Minimos		**41**	**162**	**40**			**-39,6673**	**1**	**4,3327**		**-36,7014**	**0,013249**	**2**		**7,299**		

TABELA 7.34: Resultados para o Problema S264, usando o algoritmo dos Filtros

Note-se, no entanto, que tendo em conta que o ponto inicial, $x_0 = (0, 0, 0, 0)$, possuía um valor nulo da função objectivo e que o objectivo era encontrar o ponto cujo valor da função objectivo é -44 os algoritmos implementados permitiram um decréscimo dos valores da função satisfatórios.

Veja-se, por exemplo, o comportamento da função objectivo e da violação das restrições ao longo do processo com o algoritmo PenBar, nomeadamente com a função de Penalidade ℓ_1 conjugado com o SC, na FIGURA 7.22 e com o algoritmo MF e a medida N1 conjugado com o SC, na FIGURA 7.21, conjugações com as quais de obtiveram as aproximações admissíveis com valor da função objectivo mais próximo do pretendido.

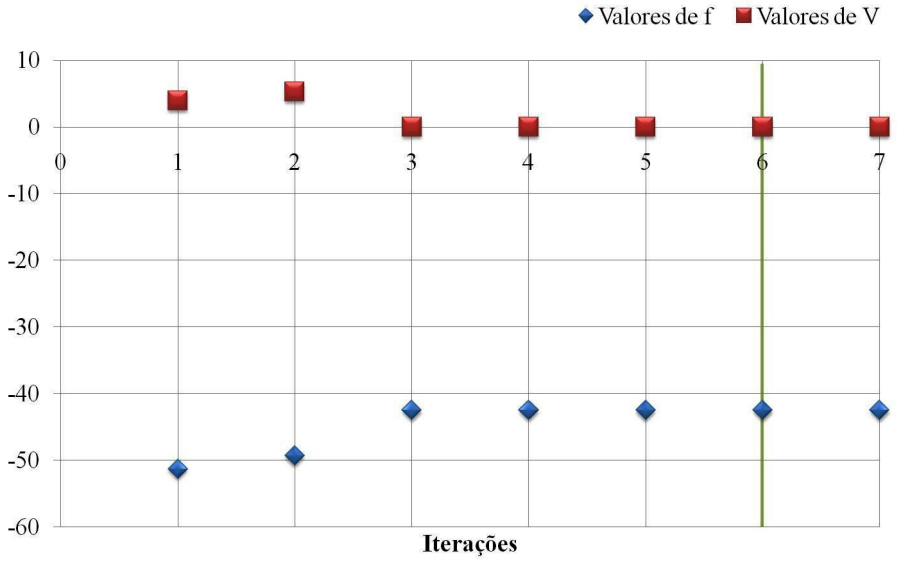

FIGURA 7.21: Comportamento dos valores da função objectivo e da violação das restrições ao longo do processo – PenBar/SC (Problema S264)

Nestas representações gráficas o eixo das abcissas representa o número da iteração, o eixo das ordenadas representa os valores das funções, os valores da violação das restrições ($V = |f(x^*) - f(x_k)|$ nos Métodos de Penalidade/Barreira e h no Método dos Filtros) estão representados por quadrados e os de f por losangos. A melhor aproximação admissível à solução foi destacada com uma linha vertical.

O algoritmo PenBar, com a função de Penalidade ℓ_1 conjugado com o SC precisou de 6 iterações até encontrar a melhor aproximação à solução admissível. Pode verificar-se que houve um progresso logo na primeira iteração relativamente rápido, dado que o valor inicial da função objectivo era zero, no entanto aumentaram os valores da violação das

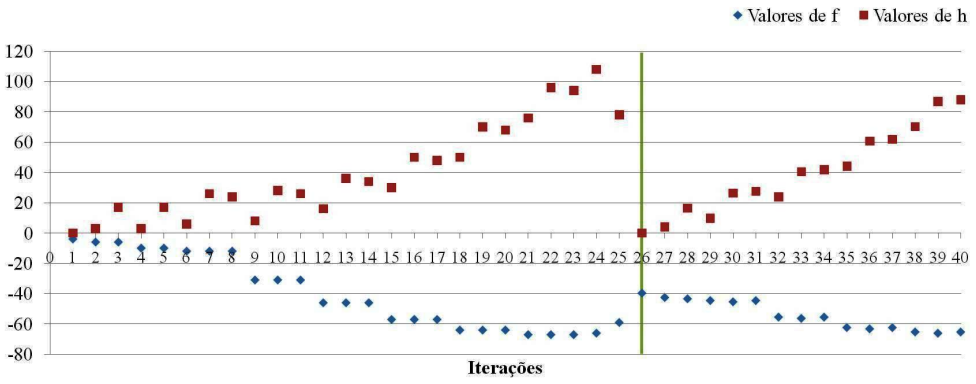

FIGURA 7.22: Comportamento dos valores da função objectivo e da violação das restrições ao longo do processo – MF/SC (Problema S264)

restrições. Nas iterações seguintes esse valor diminuiu encontrando-se uma aproximação razoavelmente perto do objectivo.

No caso do algoritmo dos Filtros já foram necessárias 26 iterações para encontra a melhor aproximação admissível, mas esta tem um valor maior da função objectivo, maior do que a encontrada usando a Função de Penalidade ℓ_1. Inicialmente a minimização dos valores de f conduziu a uma oscilação dos valores de h que na iteração 26 tomou o valor zero, sendo esta a melhor iteração admissível encontrada no processo.

Problema S270

Problema		Ponto Inicial										Solução				
S270		x0=(1,1;2,1;3,1;4,1;1)				f(x0)=1,0001						x*=(1,2,3,4,2)				f(x*) = -1
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível				
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	Todas	1258	2	-3,55E-15	-3,55E-15	1,0000	Má	-3,55E-15	1	1,0000	Má	*	*	*	*	*
HJ	Todas	510	2	1,03E-13	1,03E-13	1,0000	Má	1,03E-13	1	1,0000	Má	*	*	*	*	*
AA	EB	12390	22	-0,9981	-0,9981	0,0019	Média	-0,9981	22	0,0019	Média	*	*	*	*	*
	PB	17342	40	-4,08E-05	2,90E-05	1,0000	Má	*	*	*	*	-4,08E-05	6,98E-05	40	1,000	Má
	CP/SP	16311	40	-0,8381	0,3827	0,1619	Má	*	*	*	*	-8,38E-01	5,34E-05	7	0,162	Má
	DP	17593	40	-0,9995	-0,9994	0,0005	Média	-0,9992	36	0,0008	Média	-0,9993	4,58E-06	37	0,001	Média
	l1	13833	24	-0,9985	-0,9985	0,0015	Média	-0,9985	24	0,0015	Média	*	*	*	*	*
NM	EB	2507	6	-1	-1	**0,0000**	Boa	**-1**	6	**0,0000**	Boa	*	*	*	*	*
	PB	1300	3	-1	-1	**0,0000**	Boa	*	*	*	*	**-1**	**6,09E-08**	3	**0,000**	Boa
	CP/SP	956	5	-1	-1	**0,0000**	Boa	*	*	*	*	**-1**	2,00E-07	4	**0,000**	Boa
	DP	576	3	-1	-1	**0,0000**	Boa	*	*	*	*	**-1**	6,62E-06	2	**0,000**	Boa
	l1	1579	8	-0,9995	-0,9995	0,0005	Média	-0,9995	7	0,0005	Média	-9,88E-01	1,30E-03	**1**	0,012	Má
SC	EB	**255**	2	1,0001	1,0001	2,0001	Má	1,0001	**1**	2,0001	Má	*	*	*	*	*
	PB	5173	40	-1	infinito	infinito	Má	*	*	*	*	**-1**	1,44E-05	**1**	**0,000**	Boa
	CP/SP	1101	4	-1	-1	**0,0000**	Boa	*	*	*	*	**-1**	4,18E-07	3	**0,000**	Boa
	DP	987	4	-1	-1	**0,0000**	Boa	*	*	*	*	**-1**	9,16E-08	3	**0,000**	Boa
	l1	3104	11	-0,9964	-0,9964	0,0036	Média	-0,9974	5	0,0026	Média	-9,94E-01	1,18E-02	6	0,006	Má
Minimos		**255**	**2**			**0**		**-1**	**1**	**0,0000**		**-1**	**6,09E-08**	**1**	**0**	

TABELA 7.35: Resultados para o Problema S270, usando os algoritmos de Penalidade/Barreira

Os resultados obtidos na resolução do Problema S270, usando os Métodos de Penalidade/Barreira e o Método dos Filtros, são apresentados nas TABELAS 7.35 e 7.36.

Problema		Ponto Inicial									Solução					
S270		x0=(1,1;2,1;3,1;4,1;1)					f(x0)=1,0001				x*=(1,2,3,4,2)				f(x*) = -1	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	N1	41	1517	40	100,0033	8,2928	*	*	*	*	-0,9999	2,0400	5	2	1,00E-04	Má
	N2	41	1900	40	104,7662	7,2328	*	*	*	*	-0,9999	2,0400	5	2	1,00E-04	Má
	NEB	41	5046	40	3536,0001	infinito	*	*	*	*	*	*	*	2	*	*
	NP	41	1900	40	104,7662	52,3141	*	*	*	*	-0,9999	4,1616	5	2	1,00E-04	Má
HJ	N1	65	572	40	-17,8724	52,5274	*	*	*	*	-0,9999	2,0400	5	14	1,00E-04	Má
	N2	65	575	40	-17,5830	50,6867	*	*	*	*	-0,9999	2,0400	5	14	1,00E-04	Má
	NEB	41	8536	40	933327,4814	infinito	*	*	*	*	*	*	*	2	*	*
	NP	65	575	40	-17,5830	2569,1443	*	*	*	*	-0,9999	4,1616	5	14	1,00E-04	Má
AA	N1	41	2331	40	260,4724	10,5490	*	*	*	*	-0,9999	2,0400	5	2	1,00E-04	Má
	N2	41	2153	40	-0,2445	3,6887	*	*	*	*	0,5487	0,8458	35	3	1,5487	Má
	NEB	41	10046	40	3536,0001	infinito	*	*	*	*	*	*	*	2	*	*
	NP	41	2153	40	-0,2445	13,6062	*	*	*	*	0,5487	0,7154	35	3	1,548677984	Má
NM	N1	41	1572	40	0,4009	6,4875	0,4249	33	1,4249	Má	0,3968	1,1791	38	5	1,396798664	Má
	N2	41	1027	40	-0,9570	3,4443	*	*	*	*	0,2271	0,1818	27	6	1,227139925	Má
	NEB	41	16171	40	3536,0001	infinito	*	*	*	*	*	*	*	2	*	*
	NP	41	1027	40	-0,9570	11,8631	*	*	*	*	0,2271	0,0331	27	6	1,227139925	Má
SC	N1	41	2146	40	-0,2897	1,1287	-0,2819	11	0,7181	Má	-0,2897	0,6961	39	4	0,710313153	Má
	N2	41	2218	40	1,0263	0,9418	0,7213	11	1,7213	Má	-0,0670	0,9792	28	4	0,932956224	Má
	NEB	41	671	40	3536,0001	infinito	*	*	*	*	*	*	*	2	*	*
	NP	41	2459	40	-0,7090	2,68E-04	0,1350	17	1,1350	Má	-0,7090	2,68E-04	40	4	0,290954096	Má
Minimos		41	572	40			-0,2819	11	0,7181		-0,9999	2,7E-04	5		1E-04	

TABELA 7.36: Resultados para o Problema S270, usando o algoritmo dos Filtros

Estes resultados são semelhantes aos obtidos para os Problemas S231 e S233: o algoritmo de Penalidade/Barreira permite encontrar *Boas* aproximações à solução, nomeadamente usando no processo interno os Métodos NM e SC, mas o mesmo já não foi possível usando o algoritmo dos Filtros, usando os parâmetros por defeito.

Problema S323

Problema		Ponto Inicial						Solução									
S323		x0=(0,1)				f(x0)=5						x*=(0,5536;1,306)				f(x*) = 0	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível					
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class	
CS	Todas	799	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
HJ	Todas	134	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
AA	EB	355	2	1,16E-10	1,16E-10	0,00000	Boa	0,0	2	0,00000	Boa	*	*	*	*	*	
	Outras	1580	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
NM	EB	262	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
	PB/DP	162	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
	CP/SP	159	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
	l1	208	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
SC	EB/l1	263	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
	Outras	179	2	0,0	0,0	0	Boa	0,0	1	0	Boa	*	*	*	*	*	
Minimos		134	2			0		0	1	0		0	0	0	0,000		

TABELA 7.37: Resultados para o Problema S323, usando os algoritmos de Penalidade/Barreira

Os resultados obtidos na resolução do Problema S323, usando os Métodos de Penalidade/Barreira e o Método dos Filtros, são apresentados nas TABELAS 7.37 e 7.38.

Ambos os algoritmos permitem encontrar *Boas* aproximações à solução. O algoritmo de Penalidade/Barreira permite encontrar aproximações admissíveis usando qualquer uma das combinações de métodos possíveis.

Problema		Ponto Inicial					Solução									
S323		x0=(0,1)					f(x0)=5				x*=(0,5536;1,306)				f(x*) = 0	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS/AA	Todas	65	41	40	2,0000	0,0000	1,0000	2	1,0000	Má	*	*	*	1	*	*
HJ	Todas	113	41	40	0,0000	0,0000	0,0000	5	0,0000	Boa	*	*	*	1	*	*
NM	Todas	111	41	40	1,0000	0,0000	0,0000	12	0,0000	Boa	*	*	*	1	*	*
SC	Todas	101	41	40	1,0000	0,0000	0,0000	12	0,0000	Boa	*	*	*	1	*	*
Minimos		65	41	40			0,0000	2	0,0000		0	0	0		0	

TABELA 7.38: Resultados para o Problema S323, usando o algoritmo dos Filtros

No caso do algoritmo dos Filtros os melhores resultados são obtidos usando, no processo de Optimização sem Restrições os Métodos HJ, NM ou SC.

Problema S324

Problema		Ponto Inicial						Solução										
S324		x0=(15,3)					f(x0)=11,25					x*=(18,81;1,581)					f(x*) = 5	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível						
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class	
CS	EB/I1	163	2	9,6944	9,6944	4,6944	Má	9,6944	2	4,6944	Má	*	*	*	*	*	
	PB	1110	8	6,0431	infinito	infinito	Má	*	*	*	*	6,0431	0,0028	8	1,0431	Má	
	CP/SP	1547	10	5,8291	5,8291	0,8291	Má	*	*	*	*	5,8291	2,52E-06	10	0,8291	Má	
	DP	6267	40	4,9993	5,0018	0,0007	Média	*	*	*	*	4,9998	0,0022	37	0,0002	Média	
HJ	EB/I1	136	2	5,5625	5,5625	0,5625	Má	5,5625	1	0,5625	Má	*	*	*	*	*	
	PB	4358	40	5,1744	5,1802	0,1744	Má	*	*	*	*	5,1744	0,0058	4	0,1744	Má	
	CP/SP	3591	11	5,2009	5,2009	0,2009	Má	5,2009	10	0,2009	Má	5,2008	7,24E-05	8	0,2008	Má	
	DP	15592	40	5,0063	5,0085	0,0063	Média	*	*	*	*	5,0063	0,0021	38	0,0063	Média	
AA	EB/I1	291	2	9,6944	9,6944	4,6944	Má	9,6944	2	4,6944	Má	*	*	*	*	*	
	PB	1004	2	7,6539	7,6552	2,6539	Má	*	*	*	*	7,6539	0,0012	1	2,6539	Má	
	CP/SP	1957	10	6,6965	6,6965	1,6965	Má	*	*	*	*	6,6965	3,74E-07	10	1,6965	Má	
	DP	8027	40	5,0141	5,0164	0,0141	Má	*	*	*	*	7,6539	0,0012	1	2,6539	Má	
NM	EB/I1	226	2	5	5,0000	0	Boa	5	2	0	Boa	*	*	*	*	*	
	PB	298	3	4,98	infinito	infinito	Má	*	*	*	*	4,9800	0,01	3	0,0200	Má	
	CP/SP	967	12	5	5,0000	0	Boa	*	*	*	*	5,0000	3,96E-06	11	0,0000	Boa	
	DP	2740	40	4,9951	4,9976	0,0049	Média	*	*	*	*	4,9951	0,0024	40	0,0049	Média	
SC	EB	764	4	5,0014	5,0014	0,0014	Média	5,0014	4	0,0014	Média	*	*	*	*	*	
	PB	7518	40	5,9227	infinito	infinito	Má	*	*	*	*	5,9227	0,0017	40	0,9227	Má	
	CP/SP	1674	12	5	5,0000	0	Boa	*	*	*	*	5,0000	1,01E-05	12	0,0000	Boa	
	DP	3468	40	4,9951	4,9976	0,0049	Média	*	*	*	*	4,9951	0,0024	40	0,0049	Média	
	I1	521	3	5	5,0000	0	Boa	5	2	0	Boa	*	*	*	*	*	
Minimos		136	2			0		5	1	0		4,98	3,74E-07	1	0,000		

TABELA 7.39: Resultados para o Problema S324, usando os algoritmos de Penalidade/Barreira

De acordo com os dados da TABELA 7.39, foi possível encontrar *Boas* aproximações à solução do Problema S2324 usando o algoritmo de Penalidade/Barreira, nomeadamente usando no processo interno os Métodos NM e SC.

No entanto, de acordo com os dados da `tabela` 7.40 não foi possível encontrar nenhuma aproximação *Boa* usando o algoritmo dos Filtros.

Note-se, no entanto, que, por exemplo, usando o Método NM e a medida N1 o Filtro que se obtém tem dimensão 35, o que significa que há várias opções de aproximação à solução, no caso do problema ser relaxável.

Problema		Ponto Inicial									Solução					
S324		x0=(15,3)					f(x0)=11,25				x*=(18,81;1,581)				f(x*) = 5	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS/AA	Todas	80	41	40	10,9600	0,000	10,9600	3	5,9600	Má	*	*	*	1	*	*
HJ	N1/N2	48	41	40	5,9506	2,750	*	*	*	*	5,9506	2,7500	40	39	0,9506	Má
	NEB	48	9629	40	4301,1717	infinito	*	*	*	*	*	*	*	2	*	*
	NP	48	41	40	5,9506	7,563	*	*	*	*	5,9506	7,5625	40	39	0,9506	Má
NM	N1/N2	55	41	40	3,7225	8,500	7,2500	37	2,2500	Má	7,1256	0,2500	36	35	2,1256	Má
	NEB	90	713	40	3,7225	infinito	7,0636	36	2,0636	Má	*	*	*	2	*	*
	NP	55	41	40	3,7225	72,250	7,2500	37	2,2500	Má	7,1256	0,0625	36	35	2,1256	Má
SC	Todas	119	41	40	6,5600	0,000	6,5600	3	1,5600	Má	*	*	*	1	*	*
Minimos		48	41	40			6,5600	3	1,5600		5,9506	0,0625	36		0,9506	

TABELA 7.40: Resultados para o Problema S324, usando o algoritmo dos Filtros

Problema S325

Problema		Ponto Inicial										Solução					
S325		x0=(-3,0)					f(x0)=9					x*=(-2,372;-1,836)				f(x*) = 3,79134	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível					
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB	314	2	9	9,0000	5,20866	Má	9,0000	2	5,20866	Má	*	*	*	*	*
	PB	236	2	3	infinito	infinito	Má	*	*	*	*	2,961	0,411	2	0,8303	Má
	CP/SP	1541	17	4	3,7913	0,00004	Boa	*	*	*	*	3,7913	1,30E-05	17	0,0000	Boa
	DP	3553	40	4	3,7425	0,09754	Má	*	*	*	*	3,6938	0,0487	40	0,0975	Má
	I1	8121	40	4	4,0029	0,21056	Má	*	*	*	*	4,00E+00	9,77E-04	40	0,2106	Má
HJ	EB	133	2	9	9,0000	5,20866	Má	9,0000	1	5,20866	Má	*	*	*	*	*
	PB	248	2	3	3,3721	0,83034	Má	*	*	*	*	2,961	0,4111	1	0,8303	Má
	CP/SP	2947	40	4	3.248.560,0993	0,00314	Má	*	*	*	*	3,7882	1,50E-03	9	0,0031	Média
	DP	3391	40	4	3,7427	0,09834	Má	*	*	*	*	3,693	0,049	39	0,0983	Má
	I1	2826	40	2	infinito	infinito	Má	*	*	*	*	1,685	2,08E+00	1	2,1063	Má
AA	EB	822	2	9	9,0000	5,20866	Má	9,0000	2	5,20866	Má	*	*	*	*	*
	PB	615	2	3	infinito	infinito	Má	*	*	*	*	2,9613	0,4108	2	0,8300	Má
	CP/SP	3279	19	4	3,7914	0,00004	Boa	*	*	*	*	3,7913	6,22E-05	19	0,0000	Boa
	DP	6776	40	4	3,7425	0,09764	Má	*	*	*	*	3,6937	0,0488	40	0,0976	Má
	I1	8218	27	4	3,7914	0,00006	Boa	*	*	*	*	3,7914	1,19E-07	27	0,0001	Boa
NM	EB	833	2	9	9,0000	5,20866	Má	9,0000	1	5,20866	Má	*	*	*	*	*
	PB	334	3	3	3,3721	0,83014	Má	*	*	*	*	2,9612	0,4109	2	0,8301	Má
	CP/SP	1575	18	4	3,7913	0,00004	Boa	*	*	*	*	3,7913	9,96E-06	17	0,0000	Boa
	DP	2905	40	4	3,7425	0,09764	Má	*	*	*	*	3,6937	0,0488	40	0,0976	Má
	I1	312	3	4	3,7913	0,00004	Boa	*	*	*	*	3,7913	2,54E-06	2	0,0000	Boa
SC	EB	213	2	9	9,0000	5,20866	Má	9,0000	1	5,20866	Má	*	*	*	*	*
	PB	498	3	9	9,4007	5,54986	Má	*	*	*	*	9,3412	0,0595	2	5,5499	Má
	CP/SP	3310	21	7	7,4641	3,67276	Má	*	*	*	*	7,4641	5,46E-06	20	3,6728	Má
	DP	5127	40	7	7,4282	3,60106	Má	*	*	*	*	7,3924	0,0358	40	3,6011	Má
	I1	433	3	4	3,7914	0,00006	Boa	*	*	*	*	3,7914	3,04E-07	3	0,0001	Boa
Minimos		133	2			4,00E-05		9	1	5,20866		1,685	1,19E-07	1	0,000	

TABELA 7.41: Resultados para o Problema S325, usando os algoritmos de Penalidade/Barreira

Os resultados obtidos na resolução do Problema S325, usando os Métodos de Penalidade/Barreira e o Método dos Filtros, são apresentados nas TABELAS 7.41 e 7.42.

O algoritmo de Penalidade/Barreira permite encontrar *Boas* aproximações à solução não admissíveis, mas não permite encontrar *Boas* soluções admissíveis. Usando o algoritmo dos Filtros não se encontra nenhuma aproximação admissível e as aproximações não admissíveis são aproximações *Más*, de acordo com os critérios definidos.

Problema		Ponto Inicial									Solução					
S325		x0=(-3,0)					f(x0)=9				x*=(-2,372;-1,836)				f(x*) = 3,79134	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	N1	41	1493	40	8,3816	3,6921	*	*	*	*	6,2500	2,7500	10	16	2,4587	Má
	N2	41	2123	40	14,4181	7,2477	*	*	*	*	7,7009	2,5732	30	9	3,9095	Má
	NEB	41	4853	40	13,0000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	41	1575	40	14,4181	52,5297	*	*	*	*	7,7009	6,6216	36	11	3,9095	Má
HJ	N1	41	847	40	9,7617	4,4856	*	*	*	*	8,7617	0,0006	6	7	4,9704	Má
	N2	41	849	40	14,4203	7,2496	*	*	*	*	7,7025	2,5732	6	5	3,9112	Má
	NEB	41	9629	40	4.413,5397	infinito	*	*	*	*	*	*	*	2	*	*
	NP	41	849	40	14,4203	52,5561	*	*	*	*	7,7025	6,6216	6	5	3,9112	Má
AA	N1	41	4328	40	4,0000	5,0000	*	*	*	*	9,0000	3,02E-11	33	13	5,2087	Má
	N2	41	3694	40	14,4180	7,2477	*	*	*	*	7,7008	2,5732	15	15	3,9095	Má
	NEB	41	9653	40	13,0000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	41	3694	40	14,4180	52,5291	*	*	*	*	7,7008	6,6216	15	15	3,9095	Má
NM	N1	41	1220	40	9,1266	7,1103	*	*	*	*	8,4980	2,10E-11	20	11	4,7066	Má
	N2	41	1073	40	3,7852	3,9051	*	*	*	*	7,8967	4,93E-09	6	8	4,1054	Má
	NEB	41	9701	40	13,0000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	41	1073	40	3,7852	15,2499	*	*	*	*	7,8967	2,43E-17	6	8	4,1054	Má
SC	N1	41	1268	40	3,9759	4,0907	*	*	*	*	7,9181	8,99E-10	35	16	4,1268	Má
	N2	41	3170	40	8,7009	3,2520	*	*	*	*	7,5265	6,98E-08	6	6	3,7352	Má
	NEB	41	1301	40	13,0000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	31	1701	40	9,0519	27,7293	*	*	*	*	7,4880	1,13E-20	16	12	3,6966	Má
Minimos		31	847	40			0	0	0		6,2500	1,13E-20	6		2,4587	

TABELA 7.42: Resultados para o Problema S325, usando o algoritmo dos Filtros

Problema S326

Os resultados obtidos na resolução do Problema S326, apresentados nas TABELAS 7.43 e 7.44 não constituem *Boas* aproximações à solução, de acordo com os critérios definidos.

Problema		Ponto Inicial										Solução					
S326		x0=(4,3)						f(x0)=-69				x*=(5,240;3,746)				f(x*) = -79,8079	
Métodos		Última Iteração						Melhor Admissível				Melhor não admissível					
IP	EP	ΦEvals	k	f(xk)	Φ(xk)	|f(x*)-f(xkf)|	class	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	V(xki)	ki	|f(x*)-f(xki)|	class
CS	EB	801	2	-79,0000	-79,0000	0,80790	Má	-79,0000	1	0,8079	Má	*	*	*	*	*
	PB	263	2	-80,1681	-79,9921	0,36020	Má	*	*	*	*	-80,1681	0,1759	2	0,3602	Má
	CP/SP	1083	14	-79,8079	-79,8079	0,000000	Boa	*	*	*	*	-79,8079	4,69E-05	14	0,0000	Boa
	DP	2837	40	-79,8589	-79,8334	0,05100	Má	*	*	*	*	-79,8589	0,0255	40	0,0510	Má
	I1	237	2	-79,8078	-79,8078	0,00010	Média	-79,8078	2	0,0001	Média	-79,8078	1,17E-06	1	0,0001	Média
HJ	EB	139	2	-72,9375	-72,9375	6,87040	Má	-72,9375	1	6,8704	Má	*	*	*	*	*
	PB	721	5	-80,1663	-79,9921	0,35840	Má	*	*	*	*	-80,1663	0,1742	4	0,3584	Má
	CP/SP	989	13	-79,8066	-79,8066	0,00130	Média	-79,8066	12	0,0013	Média	-79,8093	7,23E-04	9	0,0014	Média
	DP	3059	40	-79,8612	-79,8332	0,05330	Má	*	*	*	*	-79,8612	0,0262	35	0,0533	Má
	I1	425	5	-79,8060	-79,8060	0,00190	Média	-79,8060	4	0,0019	Média	-79,8087	1,20E-03	1	0,0008	Média
AA	EB	1600	2	-79,0000	-79,0000	0,80790	Má	-79,0000	1	0,8079	Má	*	*	*	*	*
	PB	280	2	-80,1683	infinito	infinito	Má	*	*	*	*	-80,1683	0,1762	2	0,3604	Má
	CP/SP	2042	14	-79,8069	-79,8069	0,00100	Média	*	*	*	*	-79,8069	1,08E-05	13	0,0010	Média
	DP	5299	40	-79,8587	-79,8333	0,05080	Má	*	*	*	*	-79,8587	0,0254	40	0,0508	Má
	I1	536	3	-79,8078	-79,8078	0,00010	Média	-79,8078	3	0,0001	Média	-79,8078	4,20E-06	2	0,0001	Média
NM	EB	464	3	-79,8078	-79,8078	0,00010	Média	-79,8078	3	0,0001	Média	*	*	*	*	*
	PB	31508	40	-80,1685	infinito	infinito	Má	*	*	*	*	-80,1685	0,1763	1	0,3606	Má
	CP/SP	1500	17	-79,7982	-79,7982	0,00970	Média	*	*	*	*	-79,7982	9,28E-06	16	0,0097	Média
	DP	2813	40	-79,8588	-79,8334	0,05090	Má	*	*	*	*	-79,8588	0,0254	40	0,0509	Má
	I1	624	5	-79,8078	-79,8078	0,00010	Média	-79,8078	4	0,0001	Média	-79,8078	1,52E-05	2	0,0001	Média
SC	EB	275	2	-79,8025	-79,8025	0,00540	Média	-79,8025	1	0,0054	Média	*	*	*	*	*
	PB	4361	40	-80,1682	infinito	infinito	Má	*	*	*	*	-80,1682	0,1761	1	0,3603	Má
	CP/SP	2846	21	-79,8078	-79,8078	0,00010	Média	*	*	*	*	-79,8078	3,68E-07	17	0,0001	Média
	DP	3928	40	-79,8590	-79,8334	0,05110	Má	*	*	*	*	-79,8592	0,0255	39	0,0513	Má
	I1	468	4	-79,8078	-79,8078	0,00010	Média	-79,8078	3	0,0001	Média	-79,8078	1,45E-05	2	0,0001	Média
Minimos		139	2			0		-79,8078	1	0,0001		-80,1685	3,68E-07	1	0,000	

TABELA 7.43: Resultados para o Problema S326, usando os algoritmos de Penalidade/Barreira

Problema		Ponto Inicial									Solução					
S326		x0=(4,3)				f(x0)=-69					x*=(5,240;3,746)				f(x*) = -79,8079	
Métodos		Última Iteração					Melhor Admissível				Melhor não admissível					
IP	EP	fEv	hEV	k	f(xk)	h(xk)	f(xkf)	kf	|f(x*)-f(xkf)|	class	f(xki)	h(xki)	ki	Dim F	|f(x*)-f(xki)|	class
CS	N1/N1	43	1234	40	-78,7500	3,6825	*	*	*	*	-76,0000	0,3891	1	23	3,8079	Má
	NEB	41	4051	40	32,0000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	43	1234	40	-78,7500	13,5608	*	*	*	*	-76,0000	0,1514	1	23	3,8079	Má
HJ	N1	46	283	40	-87,0225	30,3165	*	*	*	*	-76,0000	0,3891	1	25	3,8079	Má
	N2	46	283	40	-87,0225	21,6705	*	*	*	*	-76,0000	0,3891	1	24	3,8079	Má
	NEB	41	7306	40	6.150,0330	infinito	*	*	*	*	*	*	*	2	*	*
	NP	46	283	40	-87,0225	469,6090	*	*	*	*	-76,0000	0,1514	1	24	3,8079	Má
AA	N1	43	1613	40	-81,4922	12,6309	*	*	*	*	-76,0000	0,3891	1	25	3,8079	Má
	N2	45	2351	40	-76,0468	0,4236	-72,75	16	7,0579	Má	-76,5000	0,2500	17	10	3,3079	Má
	NEB	41	8051	40	32,0000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	45	2351	40	-76,0468	0,1794	-72,75	16	7,0579	Má	-76,5000	0,0625	17	10	3,3079	Má
NM	N1	48	196	40	-88,8594	147,7882	*	*	*	*	-76,0000	0,3891	1	23	3,8079	Má
	N2	48	196	40	-88,8594	126,4903	*	*	*	*	-76,0000	0,3891	1	23	3,8079	Má
	NEB	41	8091	40	32,0000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	48	196	40	-88,8594	15.999,7965	*	*	*	*	-76,0000	0,1514	1	23	3,8079	Má
SC	N1	47	755	40	-79,0000	0,0000	-79	16	0,8079	Má	-76,0000	0,3891	1	8	3,8079	Má
	N2	47	616	40	-79,3251	0,2217	-79	16	0,8079	Má	-79,0935	0,0627	35	11	0,7144	Má
	NEB	41	1091	40	32,0000	infinito	*	*	*	*	*	*	*	2	*	*
	NP	47	616	40	-79,3251	0,0492	-79	16	0,8079	Má	-79,0935	0,0039	35	11	0,7144	Má
Minimos		41	196	40			-79	16	0,8079		-79,0935	0,0039	1		0,7144	

TABELA 7.44: Resultados para o Problema S326, usando o algoritmo dos Filtros

Note-se, no entanto, que os algoritmos implementados permitiram um decréscimo dos valores da função satisfatórios, aproximando-se dos valores pretendidos.

7.3.2.4 Análise final dos Resultados Numéricos

Com o objectivo de fazer uma análise geral dos resultados apresentados nas secções anteriores nesta secção será discutido o desempenho dos vários algoritmos para os problemas atrás apresentados.

Tendo em conta os critérios que foram definidos fez-se, para cada Problema, uma classificação da qualidade das aproximações à solução que foram encontradas, usando os vários algoritmos. Esta classificação será agora analisada genericamente, no sentido de aferir qual deles teve o melhor desempenho na resolução dos problemas propostos.

Os cálculos correspondentes podem ser vistos no CD anexo a este trabalho no ficheiro *ResultadosFinais.xlsx* ou *ResultadosFinais.ods*.

Métodos de Penalidade Barreira

Na TABELA 7.45 pode ver-se o número de aproximações à solução *Boas*, *Médias* e *Más*, de acordo com os critérios definidos, que foram obtidas na resolução dos 18 problemas com restrições, usando o algoritmo PenBar e os parâmetros por defeito.

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
CS	EB	7	2	7	16	88,9%				0	0,0%
	PB	4	2	3	9	50,0%			9	9	50,0%
	CP/SP	1	2	2	5	27,8%	8	2	4	14	77,8%
	DP	1	2	2	5	27,8%		1	13	14	77,8%
	l1	8	5	4	17	94,4%	2	3	5	10	55,6%
HJ	EB	4	3	9	16	88,9%				0	0,0%
	PB	3	2	5	10	55,6%			9	9	50,0%
	CP/SP	1	5	5	11	61,1%	4	8	3	15	83,3%
	DP	1		3	4	22,2%		2	12	14	77,8%
	l1	5	4	5	14	77,8%	3	2	6	11	61,1%
AA	EB	8	3	5	16	88,9%				0	0,0%
	PB	5	1	2	8	44,4%			8	8	44,4%
	CP/SP	2	1	1	4	22,2%	8	2	5	15	83,3%
	DP	2	2		4	22,2%		2	12	14	77,8%
	l1	9	5	3	17	94,4%	2	4	4	10	55,6%
NM	EB	12	2	2	16	88,9%				0	0,0%
	PB	7	1		8	44,4%	1		8	9	50,0%
	CP/SP	3		1	4	22,2%	9	4	2	15	83,3%
	DP	3			3	16,7%	1	2	12	15	83,3%
	l1	12	3	1	16	88,9%	4	3	5	12	66,7%
SC	EB	5	3	6	14	77,8%				0	0,0%
	PB	5		2	7	38,9%	1		10	11	61,1%
	CP/SP	3		2	5	27,8%	9	2	3	14	77,8%
	DP	3			3	16,7%	1	2	11	14	77,8%
	l1	8	4	2	14	77,8%	6	3	3	12	66,7%

TABELA 7.45: Qualidade das soluções encontradas usando o algoritmo PenBar, com os parâmetros por defeito

Esta tabela inclui a contagem de aproximações à solução obtidas com cada um dos métodos implementados. Para discutir estes resultados com mais pormenor vejam-se as tabelas parciais destes resultados.

As primeiras 5 tabelas apresentadas correspondem aos resultados obtidos por cada um dos Método de Penalidade/Barreira, as seguintes correspondem aos resultados por Método de Pesquisa Directa.

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
CS	EB	7	2	7	16	88,9%				0	0,0%
HJ	EB	4	3	9	16	88,9%				0	0,0%
AA	EB	8	3	5	16	88,9%				0	0,0%
NM	EB	12	2	2	16	88,9%				0	0,0%
SC	EB	5	3	6	14	77,8%				0	0,0%
	Total	**36**	**13**	**29**	**78**		**0**	**0**	**0**	**0**	
	%	40,0%	14,4%	32,2%	86,7%		0,0%	0,0%	0,0%	0,0%	

TABELA 7.46: Qualidade das soluções encontradas usando o Método de Barreira Extrema

Os resultados apresentados na TABELA 7.46 mostram que o Método de Pesquisa Directa usado no processo interno não tem grande influência nos resultados, quando se usa o Método EB, apesar de se verificar que com o Método NM foi possível encontrar um maior número de aproximações à solução.

Com este método houve uma dificuldade acrescida de encontrar aproximações não admissíveis. Isto acontece porque este método recusa pontos não admissíveis. A encontrar aproximações admissíveis foi dos que se mostrou mais eficaz, tendo sido possível encontrar uma *Boa* aproximação à solução em 36 combinações de problemas com um dos métodos de Pesquisa Directa (de 90 combinações possíveis, correspondentes à combinação de 5 Métodos de Pesquisa Directa, para 18 problemas), ou seja, nas tentativas feitas para resolver um problema com o Método EB em cerca de 40% foi possível encontrar uma *Boa* aproximação admissível.

Note-se ainda que com este método foi possível apresentar uma aproximação admissível à solução em cerca de 80% dos problemas (de 14 – 77,8% a 16 – 88,9% aproximações em 18 problemas), sendo que cerca de metade são *Médias* ou *Más*.

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
CS	PB	4	2	3	9	50,0%			9	9	50,0%
HJ	PB	3	2	5	10	55,6%			9	9	50,0%
AA	PB	5	1	2	8	44,4%			8	8	44,4%
NM	PB	7	1		8	44,4%	1		8	9	50,0%
SC	PB	5		2	7	38,9%	1		10	11	61,1%
	Total	**24**	**6**	**12**	**42**		**2**	**0**	**44**	**46**	
	%	26,7%	6,7%	13,3%	46,7%		2,2%	0,0%	48,9%	51,1%	

TABELA 7.47: Qualidade das soluções encontradas usando o Método de Barreira Progressiva

Usando o Método de Barreira Progressiva, conforme mostram os resultados apresentados na TABELA 7.47, o número total de aproximações admissíveis encontradas reduz cerca de 30% (de 86,7% para 46,7%). Neste caso já foi possível encontrar aproximações não admissíveis à solução, no entanto, de acordo com os critérios definidos, apenas 2 delas são *Boas* aproximações.

O Método de Penalidade Clássica e o Método de Penalidade Dinâmica (TABELAS 7.48 e 7.49) são os que se mostraram menos eficazes na resolução dos 18 problemas propostos permitindo encontrar menos aproximações admissíveis à solução, independentemente do método usado no processo interno. A excepção é a conjugação do Método HJ com a

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
CS	CP/SP	1	2	2	5	27,8%	8	2	4	14	77,8%
HJ	CP/SP	1	5	5	11	61,1%	4	8	3	15	83,3%
AA	CP/SP	2	1	1	4	22,2%	8	2	5	15	83,3%
NM	CP/SP	3		1	4	22,2%	9	4	2	15	83,3%
SC	CP/SP	3		2	5	27,8%	9	2	3	14	77,8%
	Total	**10**	**8**	**11**	**29**		**38**	**18**	**17**	**73**	
	%	11,1%	8,9%	12,2%	32,2%		42,2%	20,0%	18,9%	81,1%	

TABELA 7.48: Qualidade das soluções encontradas usando o Método de Penalidade Clássica

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
CS	DP	1	2	2	5	27,8%		1	13	14	77,8%
HJ	DP	1		3	4	22,2%		2	12	14	77,8%
AA	DP	2	2		4	22,2%		2	12	14	77,8%
NM	DP	3			3	16,7%	1	2	12	15	83,3%
SC	DP	3			3	16,7%	1	2	11	14	77,8%
	Total	**10**	**4**	**5**	**19**		**2**	**9**	**60**	**71**	
	%	11,1%	4,4%	5,6%	21,1%		2,2%	10,0%	66,7%	78,9%	

TABELA 7.49: Qualidade das soluções encontradas usando o Método de Penalidade Dinâmica

Penalidade Clássica, que permite encontrar aproximações admissíveis para 11 problemas (61,1%), no entanto apenas 1 delas é uma *Boa* aproximação.

No que diz respeito a aproximações não admissíveis a percentagem total de problemas para os quais fica disponível uma aproximação é similar nos dois métodos. Estas aproximações são maioritariamente *Médias* ou *Más*, de acordo com os critérios definidos. No entanto, com o Método de Penalidade Dinâmica há até uma incidência superior em *Más* aproximações.

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
CS	l1	8	5	4	17	94,4%	2	3	5	10	55,6%
HJ	l1	5	4	5	14	77,8%	3	2	6	11	61,1%
AA	l1	9	5	3	17	94,4%	2	4	4	10	55,6%
NM	l1	12	3	1	16	88,9%	4	3	5	12	66,7%
SC	l1	8	4	2	14	77,8%	6	3	3	12	66,7%
	Total	**42**	**21**	**15**	**78**		**17**	**15**	**23**	**55**	
	%	46,7%	23,3%	16,7%	86,7%		18,9%	16,7%	25,6%	61,1%	

TABELA 7.50: Qualidade das soluções encontradas usando o Método de Penalidade ℓ_1

O Método de Penalidade ℓ_1 (ver resultados na TABELA 7.50) foi o que permitiu obter mais *Boas* aproximações admissíveis à solução. Usando este método encontraram-se aproximações *Boas* admissíveis à solução de perto de 50% dos problemas (46,7%).

Em conjugação com o Método CS e AA é possível encontrar 17 aproximações admissíveis às 18 solução dos problemas teste, no entanto é em conjugação com o Método NM que se encontram 12 *Boas* aproximações admissíveis à solução. No que diz respeito a soluções não admissíveis foram encontradas para 61,1% dos problemas, sendo 17 *Boas* aproximações à solução.

Nas TABELAS 7.51 apresentam-se os resultados por Método de Pesquisa Directa.

O Método de Pesquisa Coordenada permitiu melhores resultados quando conjugado com os métodos externos EB e ℓ_1, permitindo encontrar aproximações admissíveis à solução de cerca de 90% dos problemas, sendo cerca de metade *Boas* aproximações. Resultados semelhantes são obtidos com o Método AA e NM.

Considerando aproximações admissíveis e não admissíveis, o Método CS permite encontrar 31 aproximações *Boas* (21 admissíveis e 10 não admissíveis), 19 *Médias* (13 admissíveis e 6 não admissíveis) e 40 *Más* (18 admissíveis e 31 não admissíveis). Assim, é dos métodos que permite obter as menores percentagens de problemas para os quais permite apresentar *Boas* aproximações, obtendo-se uma percentagem de 23,3% no caso de aproximações admissíveis e 11,1% no caso de aproximações não admissíveis.

O Método de Hooke-Jeeves também devolve melhores resultados em conjugação com os métodos externos EB e ℓ_1, mas o uso deste Método no Processo interno conduzia a mais aproximações *Médias* ou *Más*.

O Método HJ permite obter poucas *Boas* aproximações, quando comparado com os restantes métodos, tanto admissíveis como não admissíveis. À semelhança dos outros métodos, o HJ teve melhor desempenho usando os métodos externos EB e ℓ_1, obtendo-se no entanto, com este método as menores percentagens de problemas para os quais permite encontrar aproximações à solução.

Quanto a *Boas* aproximações permite encontrar 14 admissíveis (15,6%) e 7 não admissíveis (7,8%) num total de 90 combinações possíveis de problemas e métodos usados no processo externo (18 problemas e 5 métodos para o processo externo).

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
CS	EB	7	2	7	16	88,9%					0,0%
	PB	4	2	3	9	50,0%			9	9	50,0%
	CP/SP	1	2	2	5	27,8%	8	2	4	14	77,8%
	DP	1	2	2	5	27,8%		1	13	14	77,8%
	I1	8	5	4	17	94,4%	2	3	5	10	55,6%
	Total	**21**	**13**	**18**	**52**		**10**	**6**	**31**	**47**	
	%	23,3%	14,4%	20,0%	57,8%		11,1%	6,7%	34,4%	52,2%	

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
HJ	EB	4	3	9	16	88,9%				0	0,0%
	PB	3	2	5	10	55,6%			9	9	50,0%
	CP/SP	1	5	5	11	61,1%	4	8	3	15	83,3%
	DP	1		3	4	22,2%		2	12	14	77,8%
	I1	5	4	5	14	77,8%	3	2	6	11	61,1%
	Total	**14**	**14**	**27**	**55**		**7**	**12**	**30**	**49**	
	%	15,6%	15,6%	30,0%	61,1%		7,8%	13,3%	33,3%	54,4%	

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
AA	EB	8	3	5	16	88,9%				0	0,0%
	PB	5	1	2	8	44,4%			8	8	44,4%
	CP/SP	2	1	1	4	22,2%	8	2	5	15	83,3%
	DP	2	2		4	22,2%		2	12	14	77,8%
	I1	9	5	3	17	94,4%	2	4	4	10	55,6%
	Total	**26**	**12**	**11**	**49**		**10**	**8**	**29**	**47**	
	%	28,9%	13,3%	12,2%	54,4%		11,1%	8,9%	32,2%	52,2%	

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
NM	EB	12	2	2	16	88,9%				0	0,0%
	PB	7	1		8	44,4%	1		8	9	50,0%
	CP/SP	3		1	4	22,2%	9	4	2	15	83,3%
	DP	3			3	16,7%	1	2	12	15	83,3%
	I1	12	3	1	16	88,9%	4	3	5	12	66,7%
	Total	**37**	**6**	**4**	**47**		**15**	**9**	**27**	**51**	
	%	41,1%	6,7%	4,4%	52,2%		16,7%	10,0%	30,0%	56,7%	

Métodos		Aproximação à solução									
		Admissível					Não admissível				
IP	EP	Boa	Média	Má	Total	%	Boa	Média	Má	Total	%
SC	EB	5	3	6	14	77,8%				0	0,0%
	PB	5		2	7	38,9%	1		10	11	61,1%
	CP/SP	3		2	5	27,8%	9	2	3	14	77,8%
	DP	3			3	16,7%	1	2	11	14	77,8%
	I1	8	4	2	14	77,8%	6	3	3	12	66,7%
	Total	**24**	**7**	**12**	**43**		**17**	**7**	**27**	**51**	
	%	26,7%	7,8%	13,3%	47,8%		18,9%	7,8%	30,0%	56,7%	

TABELA 7.51: Qualidade das soluções encontradas usando os Métodos de Penalidade/Barreira

O Método AA mostrou-se tão eficaz como o CS em combinação com o Método de Penalidade ℓ_1, sendo possível com esta combinação obter uma aproximação admissível à solução de 17 dos 18 problemas teste. É até possível obter mais uma aproximação *Boa* a com esta combinação. A combinação com o Método EB também permitiu encontrar aproximações admissíveis *Boas* para 8 problemas.

No que diz respeito a aproximações não admissíveis o Método AA é muito similar aos anteriores, nomeadamente ao CS.

O Método de NM é o que permite obter mais aproximações admissíveis *Boas*, nomeadamente usando as funções EB e ℓ_1, com as quais se obtêm aproximações admissíveis *Boas* para 12 problemas teste. Em conjugação com outros métodos no processo interno a percentagem de sucesso a encontrar aproximações admissíveis desce bastante, sendo que em conjugação com o Método DP se obtém a percentagem mais baixa obtida (16,7%).

Na procura de aproximações não admissíveis a melhor conjugação do Método NM foi com o Método CP/SP, que permitiu encontrar *Boas* aproximações não admissíveis para 9 problemas teste.

Analisando o total de *Boas* aproximações admissíveis à solução encontradas quando se usa o Método SC concluir-se-ía que não se mostrou um método eficaz em comparação com os restantes Métodos de Pesquisa Directa, identificando apenas aproximações admissíveis para 47,8% (correspondentes a 43 num total de 90 combinações possíveis de problemas e métodos usados no processo externo), no entanto 24 destas aproximações são *Boas* aproximações, de acordo com os critérios definidos, superando neste aspecto o número de *Boas* aproximações admissíveis encontradas quando se usam os Métodos HJ e CS.

Relativamente a aproximações não admissíveis este método é muito similar ao Método NM.

Deste modo, sendo o objectivo encontrar uma *Boa* aproximação admissível à solução, nestes 18 problemas teste, as associações de Métodos que se mostraram mais eficientes foram o Método de Barreira Extrema conjugado com o Método de Nelder-Mead e o Método de Penalidade ℓ_1 com o Método de Nelder-Mead, com as quais foi possível encontrar aproximações *Boas* à solução de 12 dos 18 problemas teste. O Método de Penalidade ℓ_1 foi o que permitiu encontrar mais aproximações admissíveis *Boas*, independentemente

do método usado no processo interno, identificando 42 *Boas* aproximações admissíveis em 90 combinações de métodos e problemas possíveis.

Existindo alguma flexibilidade na violação das restrições, pode considerar-se o Método de Penalidade Clássica, com a qual foi possível identificar 38 *Boas* aproximações não admissíveis.

Método dos Filtros

Na TABELA 7.52 pode ver-se o número de soluções *Boas* e *Más*, de acordo com os critérios definidos, que foram obtidas na resolução dos 18 problemas com restrições, usando o algoritmo MF e os parâmetros por defeito. Em nenhum problema foram encontradas aproximações classificadas como *Médias*.

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
CS	N1	2	10	12	66,7%		10	10	55,6%
	N2	2	10	12	66,7%	1	9	10	55,6%
	NEB	1	9	10	55,6%			0	0,0%
	NP	2	10	12	66,7%	1	9	10	55,6%
HJ	N1	5	8	**13**	**72,2%**	1	12	**13**	**72,2%**
	N2	5	7	**12**	**66,7%**	1	12	**13**	**72,2%**
	NEB	2	8	**10**	**55,6%**			**0**	**0,0%**
	NP	4	7	**11**	**61,1%**	1	11	**12**	**66,7%**
AA	N1	1	11	12	66,7%		12	12	66,7%
	N2	1	13	14	77,8%		11	11	61,1%
	NEB	2	8	10	55,6%			0	0,0%
	NP	1	13	14	77,8%		13	13	72,2%
NM	N1	2	9	**11**	**61,1%**		10	**10**	**55,6%**
	N2	2	10	**12**	**66,7%**		10	**10**	**55,6%**
	NEB	2	7	**9**	**50,0%**			**0**	**0,0%**
	NP	2	10	**12**	**66,7%**		10	**10**	**55,6%**
SC	N1	2	12	14	77,8%		9	9	50,0%
	N2	2	12	14	77,8%		9	9	50,0%
	NEB	2	7	9	50,0%			0	0,0%
	NP	2	12	14	77,8%		9	9	50,0%

TABELA 7.52: Qualidade das soluções encontradas usando o algoritmo MF, com os parâmetros por defeito

Nos dados apresentados nesta tabela pode ver-se que nenhuma das medidas permitiu encontrar tantas aproximações à solução como as que se encontraram com o algoritmo PenBar com as funções EB, PB e ℓ_1, no entanto é de notar que a contabilização de aproximações foi feita tendo em conta a alteração do algoritmo que inclui nos resultados

finais as melhores soluções admissível e não admissível e que foi feita com a motivação do algoritmo MF implementado o permitir.

À semelhança do que foi feito na secção anterior, para os resultados do algoritmo PenBar, esta tabela inclui a contagem de aproximações à solução obtidas com cada uma das medidas e métodos implementados. Para discutir estes resultados com mais pormenor vejam-se as tabelas parciais destes resultados.

As primeiras tabelas apresentadas (TABELAS 7.53, 7.54, 7.55 e 7.56) correspondem aos resultados obtidos por cada uma das Medidas de violação implementadas e as seguintes (TABELA 7.57) correspondem aos resultados por Método de Pesquisa Directa.

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
CS	N1	2	10	12	66,7%		10	10	55,6%
HJ	N1	5	8	13	72,2%	1	12	13	72,2%
AA	N1	1	11	12	66,7%		12	12	66,7%
NM	N1	2	9	11	61,1%		10	10	55,6%
SC	N1	2	12	14	77,8%		9	9	50,0%
	Total	**12**	**50**	**62**		**1**	**53**	**54**	
	%	13,3%	55,6%	68,9%		1,1%	58,9%	60,0%	

TABELA 7.53: Qualidade das soluções encontradas usando a medida N1

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
CS	N2	2	10	12	66,7%	1	9	10	55,6%
HJ	N2	5	7	12	66,7%	1	12	13	72,2%
AA	N2	1	13	14	77,8%		11	11	61,1%
NM	N2	2	10	12	66,7%		10	10	55,6%
SC	N2	2	12	14	77,8%		9	9	50,0%
	Total	**12**	**52**	**64**		**2**	**51**	**53**	
	%	13,3%	57,8%	71,1%		2,2%	56,7%	58,9%	

TABELA 7.54: Qualidade das soluções encontradas usando a medida N2

Os resultados apresentados nas TABELAS 7.53, 7.54 e 7.55 os resultados obtidos com o MF, usando as medidas N1, N2 e NP, são muito similares. O maior número de aproximações encontradas, nestas três combinações, é superior quando é usado o Método HJ no processo interno, obtendo-se nesse caso aproximações admissíveis *Boas* para 5 problemas teste, no caso de se usar a N1 ou a N2, e 4, no caso de se usar a NP.

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
CS	NP	2	10	12	66,7%	1	9	10	55,6%
HJ	NP	4	7	11	61,1%	1	11	12	66,7%
AA	NP	1	13	14	77,8%		13	13	72,2%
NM	NP	2	10	12	66,7%		10	10	55,6%
SC	NP	2	12	14	77,8%		9	9	50,0%
	Total	**11**	**52**	**63**		**2**	**52**	**54**	
	%	12,2%	57,8%	70,0%		2,2%	57,8%	60,0%	

TABELA 7.55: Qualidade das soluções encontradas usando a medida NP

Nos três casos a percentagem de aproximações admissíveis e não admissíveis *Más* é superior à percentagem de *Boas* aproximações e o total de problemas para os quais se encontraram aproximações é bastante inferior ao obtido com o algoritmo PenBar com as funções EB, PB e ℓ_1.

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
CS	NEB	1	9	10	55,6%			0	0,0%
HJ	NEB	2	8	10	55,6%			0	0,0%
AA	NEB	2	8	10	55,6%			0	0,0%
NM	NEB	2	7	9	50,0%			0	0,0%
SC	NEB	2	7	9	50,0%			0	0,0%
	Total	**9**	**39**	**48**		**0**	**0**	**0**	
	%	10,0%	43,3%	53,3%		0,0%	0,0%	0,0%	

TABELA 7.56: Qualidade das soluções encontradas usando a medida NEB

No caso de se usar a medida NEB (ver TABELA 7.56), à semelhança do que acontece para o Método de Barreira Extrema, no qual esta medida se baseia, não é possível encontrar qualquer aproximação não admissível e os problemas para os quais foi possível encontrar aproximações admissíveis é reduzido.

O algoritmo MF não permitiu encontrar tantas aproximações à solução como as que se encontraram com o algoritmo PenBar com as funções EB, PB e ℓ_1, no entanto permite encontrar mais aproximações admissíveis do que quando são usadas as funções PB e CS usando qualquer uma das medidas à excepção da medida NEB.

Nas TABELAS 7.57 apresentam-se os resultados por Método de Pesquisa Directa. Estes resultados não diferem muito de método para método. O que mais se destaca é o Método

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
CS	N1	2	10	12	66,7%		10	10	55,6%
	N2	2	10	12	66,7%	1	9	10	55,6%
	NEB	1	9	10	55,6%			0	0,0%
	NP	2	10	12	66,7%	1	9	10	55,6%
	Total	**7**	**39**	**46**		**2**	**28**	**30**	
	%	7,8%	43,3%	51,1%		2,2%	31,1%	33,3%	

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
HJ	N1	5	8	13	72,2%	1	12	13	72,2%
	N2	5	7	12	66,7%	1	12	13	72,2%
	NEB	2	8	10	55,6%			0	0,0%
	NP	4	7	11	61,1%	1	11	12	66,7%
	Total	**16**	**30**	**46**		**3**	**35**	**38**	
	%	17,8%	33,3%	51,1%		3,3%	38,9%	42,2%	

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
AA	N1	1	11	12	66,7%		12	12	66,7%
	N2	1	13	14	77,8%		11	11	61,1%
	NEB	2	8	10	55,6%			0	0,0%
	NP	1	13	14	77,8%		13	13	72,2%
	Total	**5**	**45**	**50**		**0**	**36**	**36**	
	%	5,6%	50,0%	55,6%		0,0%	40,0%	40,0%	

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
NM	N1	2	9	11	61,1%		10	10	55,6%
	N2	2	10	12	66,7%		10	10	55,6%
	NEB	2	7	9	50,0%			0	0,0%
	NP	2	10	12	66,7%		10	10	55,6%
	Total	**8**	**36**	**44**		**0**	**30**	**30**	
	%	8,9%	40,0%	48,9%		0,0%	33,3%	33,3%	

Métodos		Aproximação à solução							
		Admissível				Não admissível			
IP	EP	Boa	Má	Total	%	Boa	Má	Total	%
SC	N1	2	12	14	77,8%		9	9	50,0%
	N2	2	12	14	77,8%		9	9	50,0%
	NEB	2	7	9	50,0%			0	0,0%
	NP	2	12	14	77,8%		9	9	50,0%
		8	**43**	**51**		**0**	**27**	**27**	
		8,9%	47,8%	56,7%		0,0%	30,0%	30,0%	

TABELA 7.57: Qualidade das soluções encontradas usando o Método dos Filtros

HJ com o qual foi possível encontrar 16 *Boas* aproximações admissíveis à solução dos problemas teste.

Considerando os totais de aproximações admissíveis à solução, os resultados obtidos com o algoritmo MF são similares aos obtidos com o algoritmo PenBar, sendo que com os métodos CS, HJ, NM é possível encontrar mais aproximações com o algoritmo PenBar e com os métodos AA e SC encontram-se mais aproximações com o algoritmo MF.

Deste modo, sendo o objectivo encontrar uma *Boa* aproximação admissível à solução, nestes 18 problemas teste, as conjugações de Métodos e medidas que se mostraram mais eficientes foram o Método HJ com as medidas N1, N2 e NP.

Tendo em conta estes resultados os algoritmos de Penalidade/Barreira são os que permitem encontrar melhores aproximações à solução, nomeadamente quando se usa o Método de Penalidade ℓ_1. No entanto este método não permitiu encontrar, por exemplo, uma aproximação à solução admissível para o Problema PA, quando conjugado com os Métodos NM e SC.

Note-se que as adaptações que foram feitas aos Métodos de Penalidade/Barreira implementados, que se basearam nas características apresentadas pelo Método dos Filtros, permitem que, havendo a exigência da solução ser admissível, se a última iteração não o for (apesar de ter o melhor valor da Função Penalidade/Barreira) fica disponível uma solução admissível, neste caso a melhor encontrada no processo iterativo. Assim, o estudo do Método dos Filtros tornou-se essencial para melhorar o desempenho dos Métodos de Penalidade/Barreira implementados.

Deste modo, à semelhança do que aconteceu para os Métodos de Pesquisa Directa não se pode indicar o melhor algoritmo para todos os problemas, pelo que, na presença de um problema que possa ser resolvido usando estes métodos/algoritmos será de testar todos os algoritmos escolhendo a aproximação à solução que melhor se adeqúe à situação em questão.

Capítulo 8

API - *Application Programming Interface*

De modo a permitir uma fácil interacção com os métodos e metodologias desenvolvidas foi implementada uma API (*Application Programming Interface*), com recurso à Tecnologia Java, onde todos os métodos e algoritmos estudados e desenvolvidos, que foram apresentados nos capítulos anteriores, estão implementados.

Esta API é constituída por um conjunto de classes correspondentes a todos os algoritmos implementados, a classes usadas na definição dos problemas teste, a classes auxiliares de execução.

Para além de implementar a API, foi ainda implementada uma aplicação com um GUI (*Graphical User Interface*) para testar a API e fornecer aos utilizadores uma forma simples de interagir com esta.

Neste Capítulo, que se baseia nos trabalhos [43–46, 112], será apresentada genericamente a estrutura da API, destacando-se as classes implementadas e as linhas gerais de como pode ser usada. Também são apresentadas algumas ideias de trabalhos futuros a desenvolver.

No que respeita às classes correspondentes aos problemas teste será apresentada a sua estrutura geral. As classes correspondentes à componente gráfica da aplicação serão apresentadas através das visualizações de execução, onde se poderão ver com mais pormenor os dados de entrada e saída de cada um dos métodos.

8.1 Estrutura Geral da API

A estrutura geral da API é apresentada na FIGURA 8.1, que foi apresentada em [112] e que ilustra todas as opções disponíveis para o utilizador. A explicação desta estrutura geral será acompanhada da explicação do funcionamento da aplicação gráfica desenvolvida paralelamente, por se ter considerado que seria a forma mais elucidativa de o fazer.

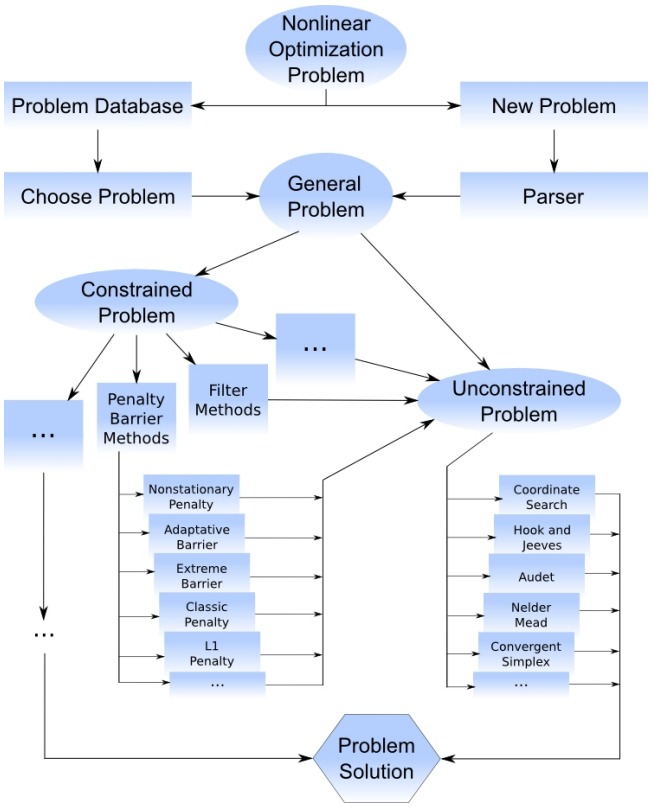

FIGURA 8.1: Estrutura Geral da API.
(Fonte: Correia et. al. em [112])

Usando as classes definidas para cada algoritmo ou usando o GUI, o utilizador pode escolher: escrever o problema a resolver ou escolher um problema da base de dados.

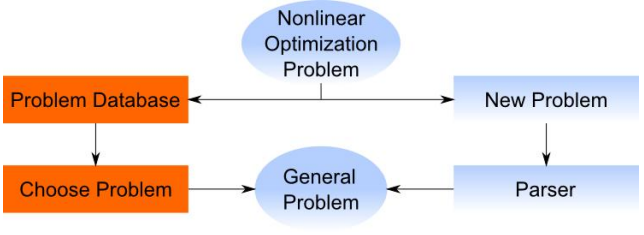

FIGURA 8.2: Escolher um problema guardado.
(Fonte: Correia et. al. em [43, 44])

Os problemas que já estão escritos usando Java são os problemas sem restrições apresentados na secção 6.6.1 e os problemas com restrições apresentados na secção 7.3.1. A variável correspondente a cada um dos problemas implementados é uma variável inteira `int PR` que pode assumir valores entre 1 e 25 no caso de problemas sem restrições e de 1 a 18 no caso de problemas com restrições.

Escolhendo esta segunda opção (FIGURA 8.2), usando a componente gráfica da aplicação deve executar-se a classe `semRestricoes` ou `comRestricoes`, dependendo do tipo de problema que se pretende resolver, seleccionar a opção *Use a Stored Problem* e seleccionar o problema que se pretende resolver, da lista de problemas disponíveis (ver FIGURA 8.3 e FIGURA 8.4).

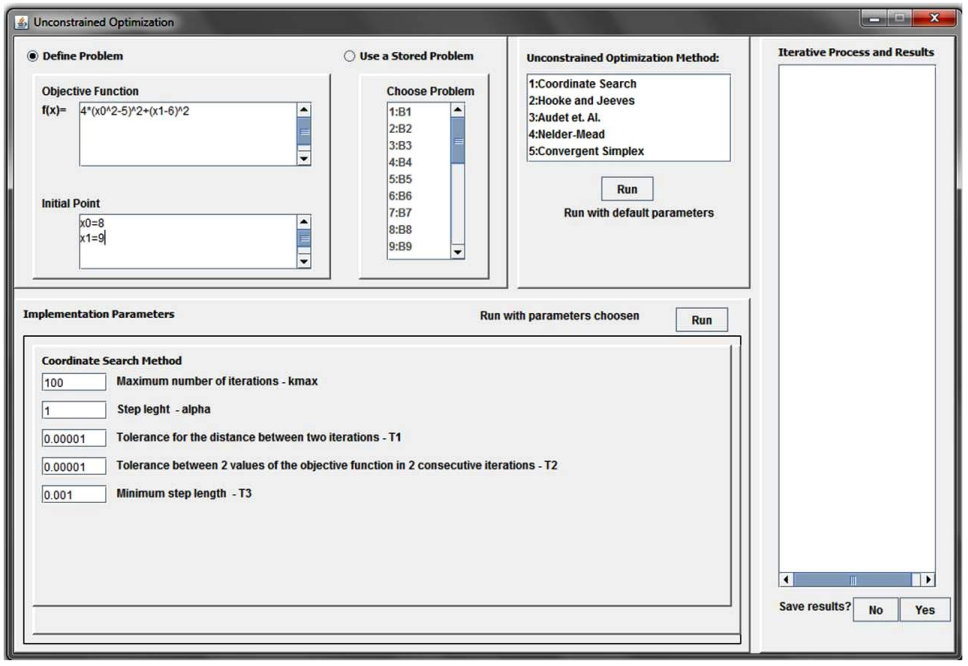

FIGURA 8.3: API/GUI - Definição de Problema sem Restrições

Escolhendo a primeira opção e se o problema a resolver é um problema sem restrições, usando a componente gráfica da aplicação deve executar-se a classe `semRestricoes`, seleccionando depois a opção *Define Problem* e definir a função objectivo e o ponto inicial. Por exemplo, se utilizador quisesse definir o problema S201, o que devia preencher era o que é apresentado na FIGURA 8.3.

A variável que representa esta escolha é uma variável inteira `intE_e` ou `int E_i` que pode assumir os valores 1 (no caso de se optar pela opção *Use a Stored Problem*) ou 2 (no caso de se optar pela opção *Define Problem*).

Se o problema que se tem para resolver é um problema com restrições, usando a componente gráfica da aplicação deve executar-se a classe `comRestricoes`, seleccionar a opção *Define Problem* e definir a função objectivo, o ponto inicial e as restrições do problema. Por exemplo, se utilizador quiser definir o problema S227, deve preencher os campos de acordo com a FIGURA 8.4.

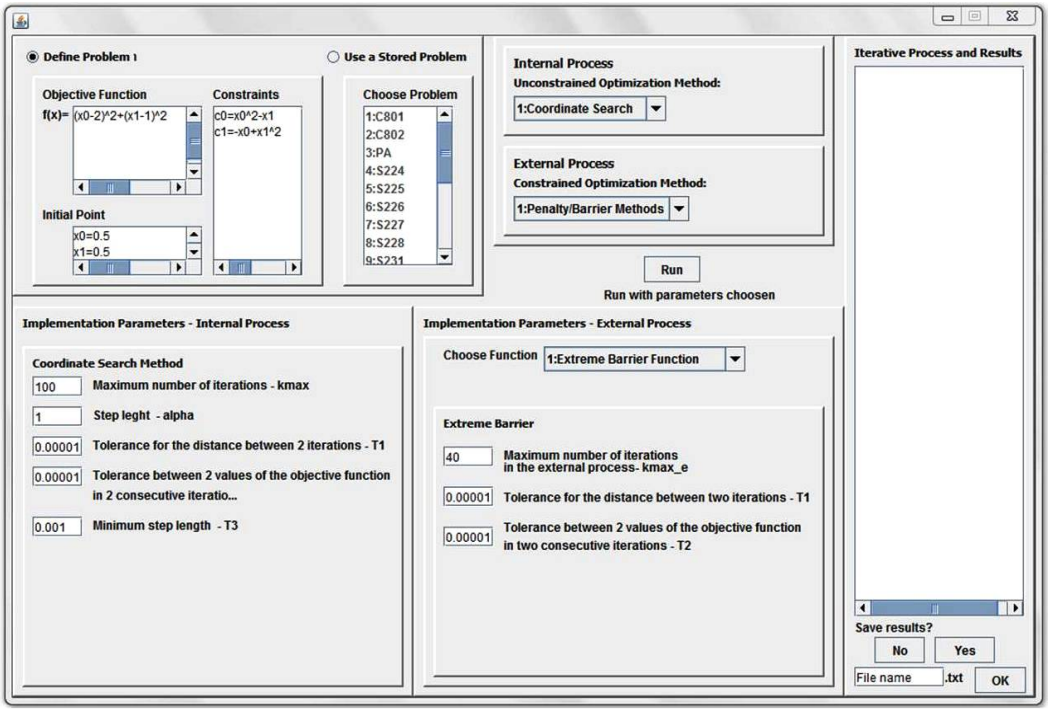

FIGURA 8.4: API/GUI - Definição de Problema com Restrições

Tendo-se optado pela definição do problema, são usadas mais 2 variáveis para definir a função objectivo e o ponto inicial: `String eq = " "` e `String initialPoint = " "`, se o problema tiver restrições é criado uma variável do mesmo tipo para cada uma delas.

Neste momento a GUI só está preparada para receber restrições do tipo $c(x) \leq 0$, mas conforme se verá abaixo, a API está preparada para lidar com qualquer tipo de restrição.

Os problemas que já estão definidos têm a estrutura da interface `ProblemaGeral`, cuja definição é apresentada na FIGURA 8.5. Assim, tendo sido definida uma interface, esta impõe à classe que declarar que a implementa, uma implementação completa de todos os métodos que ela possui pelo que todos os problemas têm um vector de restrições (vazio no caso de Problemas sem Restrições), as componentes desse vector podem ser restrições de igualdade, desigualdade ou limites simples, um ponto inicial, uma dimensão e têm que permitir, através dos seus métodos, avaliar a função objectivo, contar o número

```
package MaquinasEstado.generico;

import java.util.*;

public interface ProblemaGeral {

//O que o problema tem que ter:
    //VARIÁVEIS Obrigatórias
    //as respectivas restrições (um vector de restrições)
    public Vector<Constraint> constraints  = null;
    //tipo de restrição (igualdade, desigualdade ou limites simples)
    public int typeOf(int i);
    //um ponto inicial
    double[] x0();
    //dimensão
    int dim();

    //MÉTODOS
    //forma de avaliar a função objectivo (a expressão)
    public  double evaluatef (double[] parametros);
    //forma de contar um número de restrições
    int numerResticoes();
    int numerResticoes_igualdade();
    int numerResticoes_desigualdade();
    int numerResticoes_limiteSimples();
    //forma de avaliar  as restrições (a expressão)
    double evaluateH (int n, double[] xk);
}
```

FIGURA 8.5: Interface Problema Geral

de restrições dos vários tipos e avaliar as restrições, tal como foi definido na interface `ProblemaGeral`.

Para distinguir a implementação de problemas com restrições de problemas sem restrições criaram-se 2 sub-interfaces da interface `ProblemaGeral`: `ProblemaSemRestricoes` e `ProblemaComRestricoes`. As classes que implementem estas sub-interfaces têm obrigatoriamente que implementar todos os métodos da interface `ProblemaGeral`.

Deste modo, um problema que já esteja na base de dados já é uma classe que implementa esta interface, no caso de possuir restrições ainda usa a interface `Constraint` que é interface para as classes que correspondem às restrições dos problemas e cuja definição é apresentada na FIGURA 8.6.

Se o utilizador não pretende usar a GUI, pode definir os seus próprios problemas criando classes em Java que implementem as interfaces apresentadas.

```
package MaquinasEstado.generico;

public interface Constraint {
    //VARIÁVEIS
      //tipos de restrições
    int SIMPLE = 0;
    int EQUALLITY = 1;
    int INEQUALLITY = 2;

    //MÉTODOS para:
    //definir tipo da restrição
    void setType(int t);
    //definir a expressão da restrição (como String)
    void setEquation(String eq);
    //avaliar a restrição em xk (valor da expressão em x_k)
    double evaluate (double[] xk);
    //saber o tipo da restrição
    public int type();
    //ir buscar a expressão da restrição como String
    String getEquation();
}
```

FIGURA 8.6: Interface Constraint

Se o problema a resolver for um problema definido pelo utilizador os dados introduzidos (*Strings*) são interpretados por um *parser* que também faz parte da API, e é criado um novo Problema Geral (ver FIGURA 8.7).

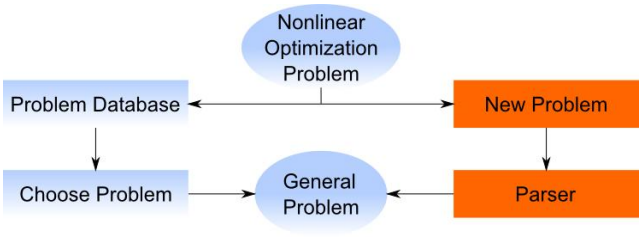

FIGURA 8.7: Opção escrever um novo problema.
(Fonte: Correia et. al. em [43, 44])

Depois de escolhido ou gerado o problema a resolver que pode ser com ou sem restrições (ver FIGURA 8.8) há que escolher o(s) método(s) que se pretende(m) usar para o resolver.

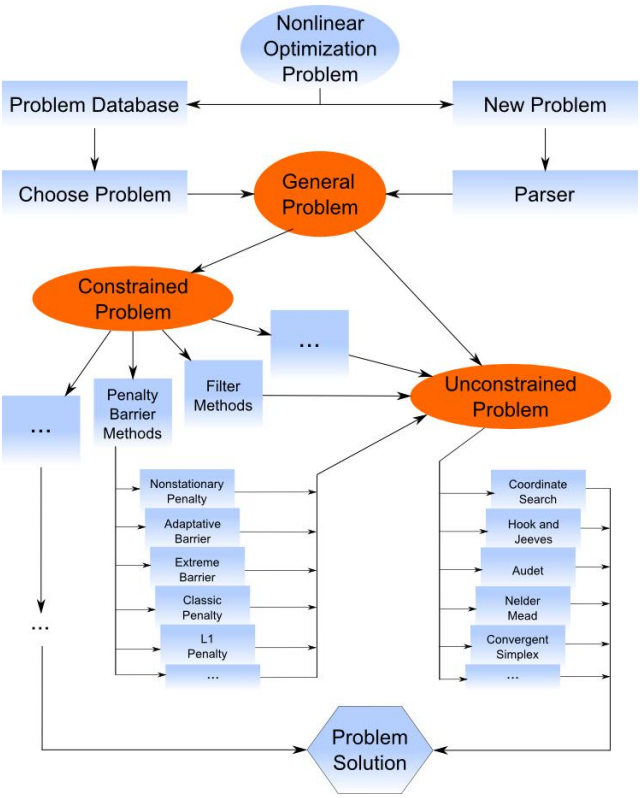

FIGURA 8.8: Problema a resolver.
(Fonte: Correia et. al. em [43, 44])

8.1.1 Problemas com Restrições

Se o problema é um Problema com Restrições pode, na versão da API implementada neste trabalho, escolher-se o algoritmo de Penalidade/Barreira ou o algoritmo dos Filtros (ver FIGURA 8.9 e FIGURA 8.10), no entanto há a possibilidade de no futuro incluir novos métodos de Optimização com Restrições.

A escolha do método a usar é efectuado através de uma variável `int MET_e` que assumirá o valor 1 se se pretende usar o algoritmo PenBar ou 2 se se pretende usar o algoritmo MF.

Sendo escolhido um ou outro para o processo externo há que escolher o métodos a usar no processo interno. Para o processo interno estão disponíveis os 5 Métodos de Pesquisa Directa implementados (ver FIGURA 8.11), no entanto há a possibilidade de no futuro incluir novos métodos de Optimização sem Restrições. Os parâmetros destes Métodos de Pesquisa Directa, que já foram apresentados no Capítulo 6 e que podem ser alterados pelo utilizador que utilize o GUI ou usando as classes nas suas próprias aplicações.

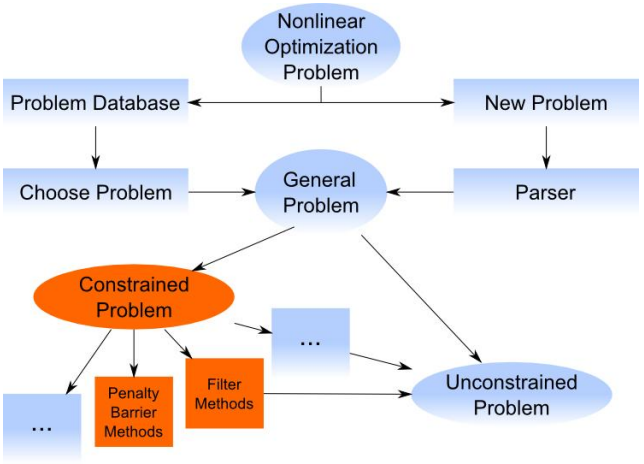

FIGURA 8.9: Métodos para Optimização com Restrições.
(Fonte: Correia et. al. em [43, 44])

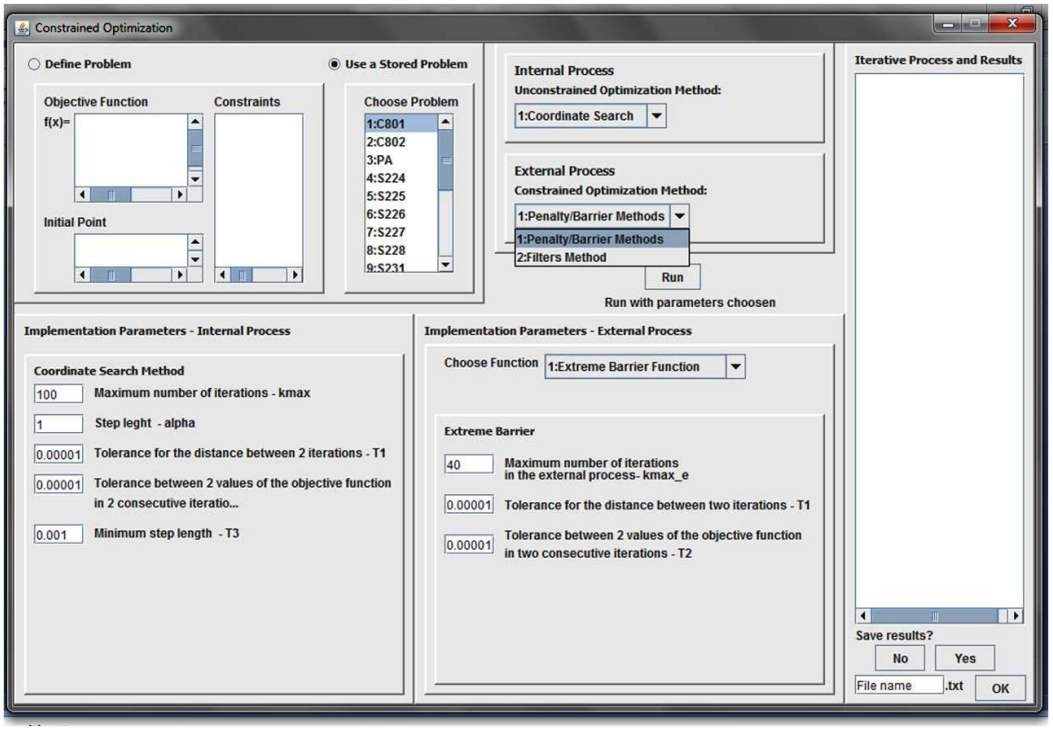

FIGURA 8.10: API/GUI - Opções para Optimização com Restrições

A variável que corresponde a esta escolha é uma variável inteira `int MET_i` que pode assumir os valores de 1 a 5, correspondendo aos Métodos de Pesquisa Directa pela ordem que é apresentada na FIGURA 8.11: CS, HJ, AA, NM e SC.

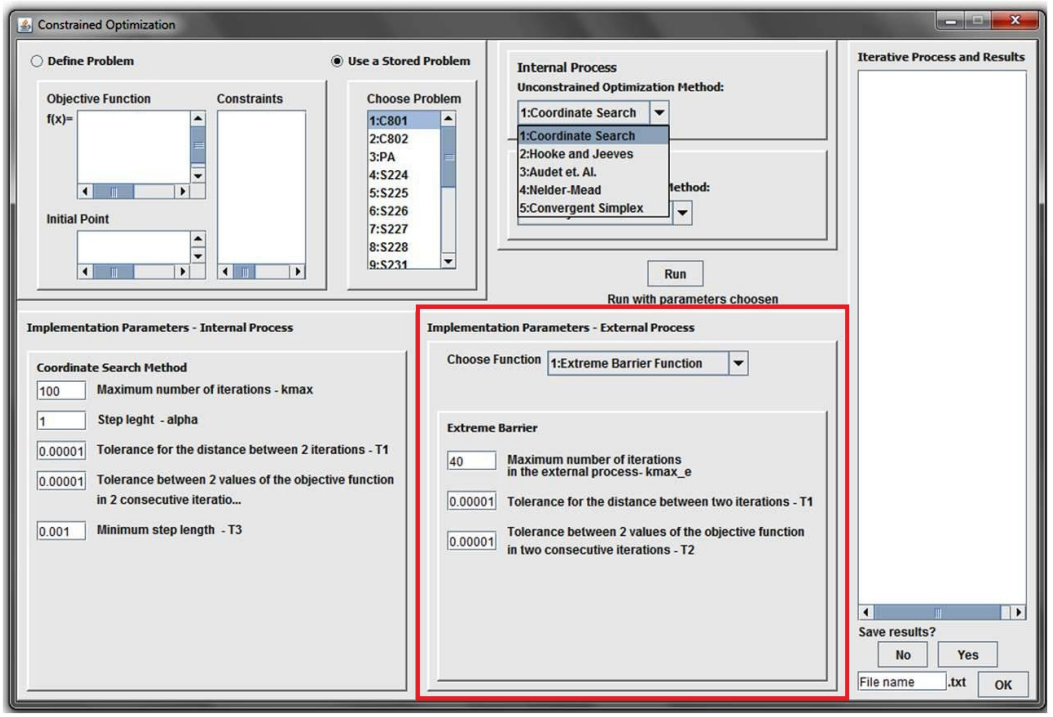

FIGURA 8.11: API/GUI - Opções para o método a usar no processo interno

Escolhendo o algoritmo de Penalidade/Barreira para o processo externo é possível escolher uma das 6 Funções de Penalidade/Barreira implementadas (ver FIGURA 8.12).

A escolha de uma dessas 6 funções corresponde à definição da variável inteira quer representa a função de penalidade a usar (assume valores de 1 a 6) ou a medida para a violação das restrições no algoritmo dos Filtros (nesse caso, assume valores de 1 a 7), `int norma_ou_phi_a_usar`.

Dependendo da Função de Penalidade/Barreira escolhida, aparece no rectângulo inferior direito o correspondente painel de parâmetros, que podem ser alterados pelo utilizador.

Seleccionando, por exemplo a função Barreira Extrema aparecem os correspondentes parâmetros como se pode ver na FIGURA 8.11, na qual se destacou com um rectângulo vermelho o painel correspondente.

Os painéis correspondentes às 6 funções implementadas podem ser vistos na FIGURA 8.13. Neste painéis estão por defeito os parâmetros que foram apresentados no Capítulo

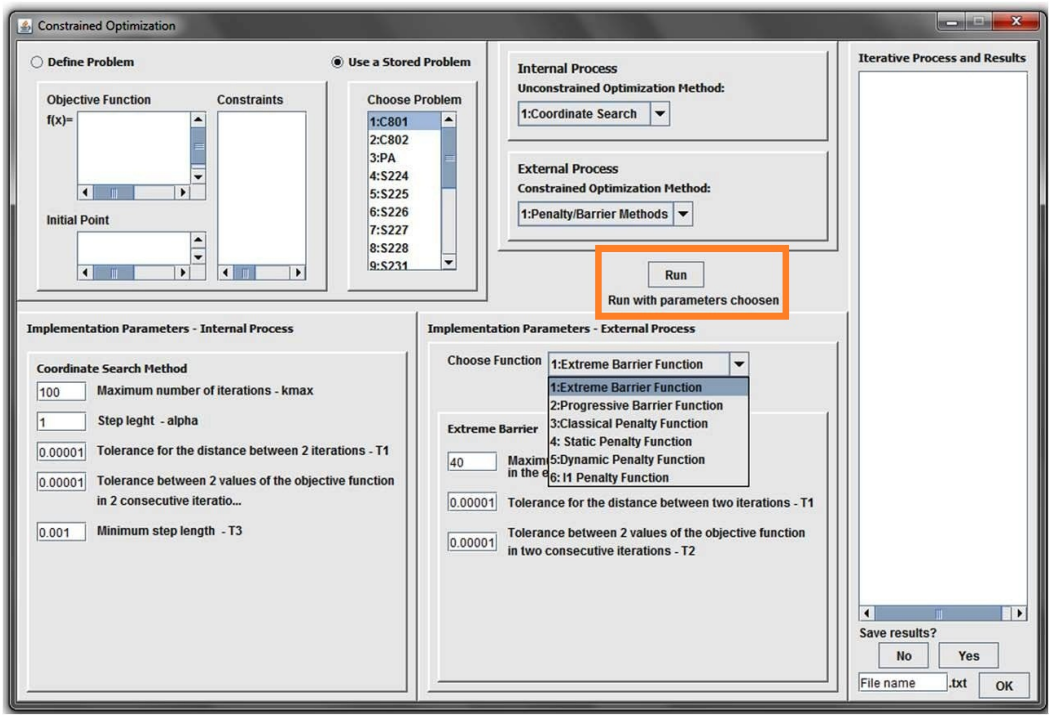

FIGURA 8.12: API/GUI - Opções para a Função de Penalidade/Barreira

7, podendo no entanto ser alterados pois a cada um destes parâmetros corresponde a uma variável na API (dentro da classe correspondente).

Para executar o algoritmo PenBar com a GUI e as respectivas opções escolhidas, o utilizador deve premir o botão *Run* que aparece na FIGURA 8.12 destacado com um rectângulo laranja.

Quando esta opção é seleccionada é criado um novo objecto da classe `Programa_PenBar`, com todos os parâmetros de entrada, usando o respectivo construtor. Para tal é usada a instrução:

```
Programa_PenBar PB=new Programa_PenBar(
 //parametros do processo externo
   pr, norma_ou_phi_a_usar, gama_e,  q,
   MET_i, T1_e,  T2_e, kmax_e, hmax,
           //parametros do processo interno
             kmax_i, alpha_i,  T1_i,  T2_i,  T3_i,
             alpha_m_i,  alpha_p_i,
             alfa_i,  beta_i,  gama_i,  s_i,  T4_i,  T5_i,  T6_i,
```

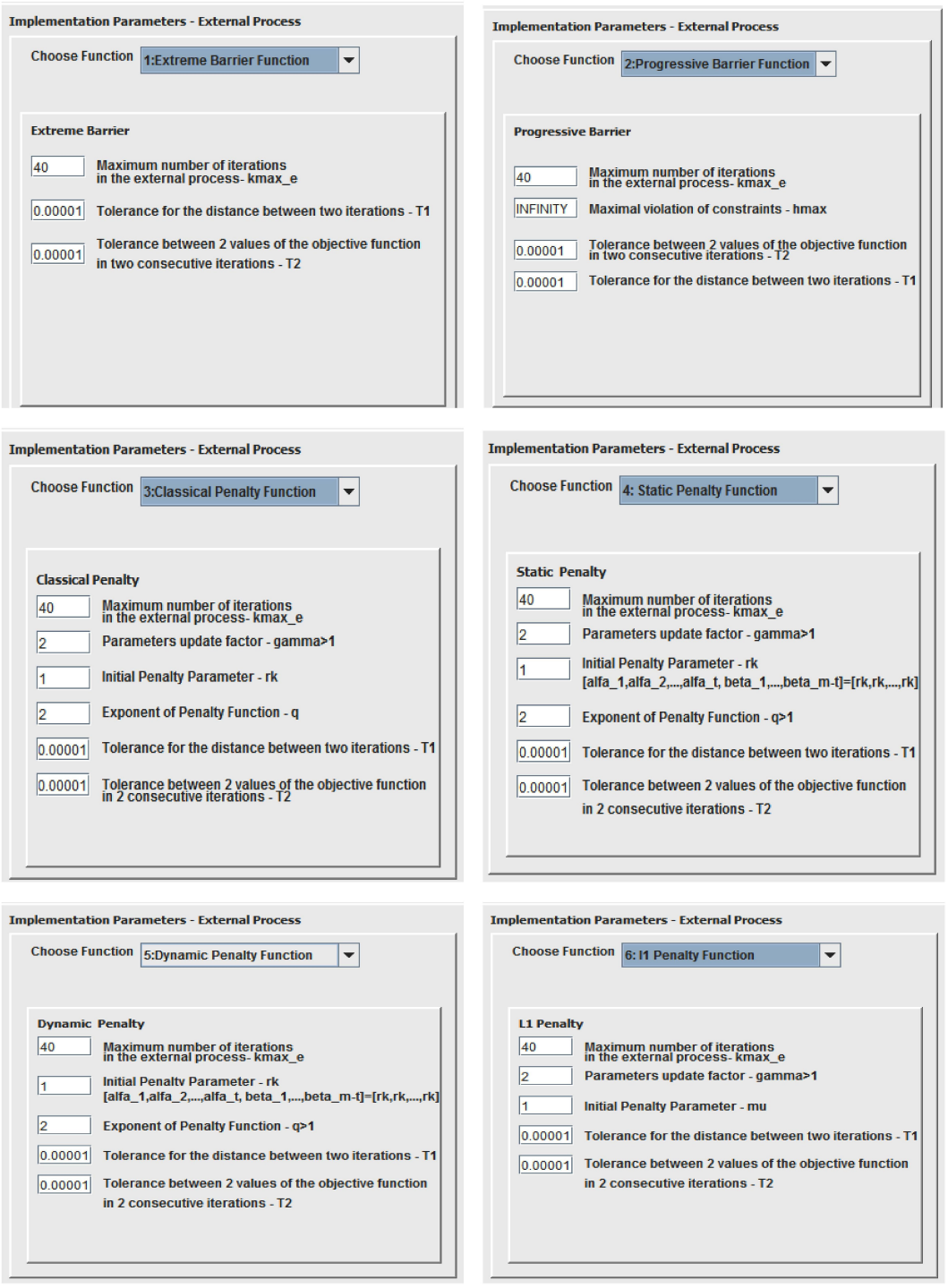

FIGURA 8.13: API/GUI - Painéis Correspondentes às Função de Penalidade/Barreira

```
gama_s_i,  xi_i,this,rk);
```

Depois de criado o objecto `PB` da Classe `Programa_PenBar` é executado o algoritmo, através da instrução `PB.run();` que executa o procedimento implementado com os dados de entrada introduzidos quando da criação do objecto.

A execução deste método ainda usa objectos das classes `CalPhi` e `Criterios`, o primeiro para calcular o valor num determinado ponto da função Penalidade/Barreira, escolhida pelo utilizador, e o segundo para calcular os valores das tolerâncias do processo iterativo.

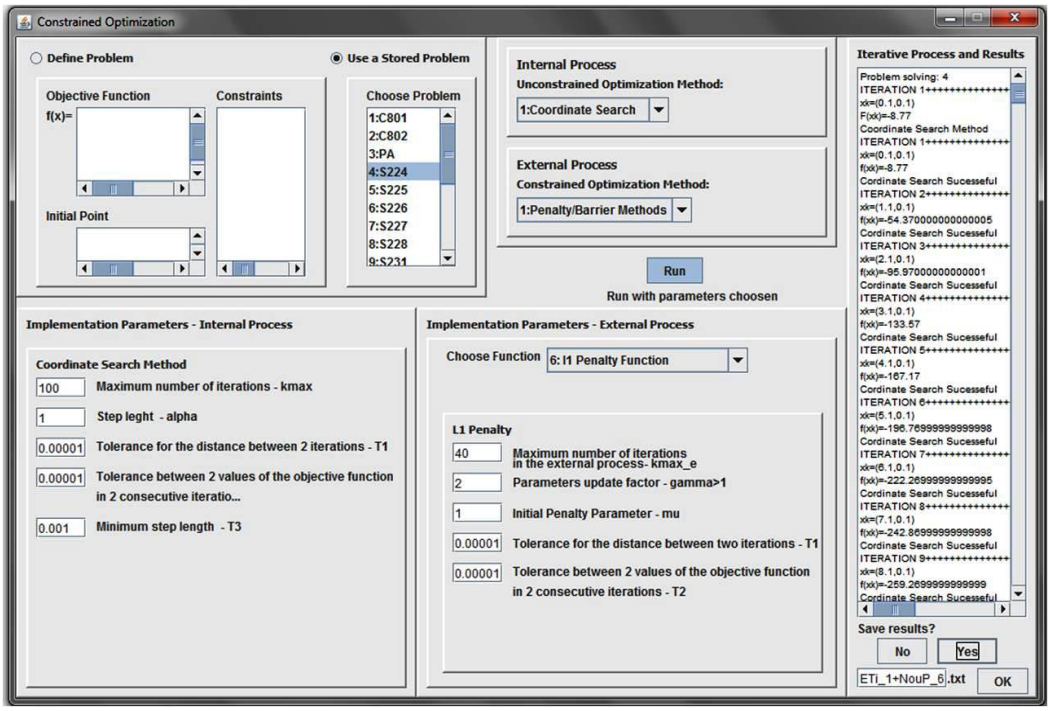

FIGURA 8.14: API/GUI - Apresentação dos resultados do Processo iterativo Optimização com Restrições

Os resultados obtidos são apresentados do lado direito, na caixa *Iterative Process and Results*, como se pode ver na FIGURA 8.14.

Posteriormente podem gravar-se os resultados da execução num ficheiro, seleccionando a opção *Yes*, escolhendo o nome que ser quer dar ao ficheiro e fazendo *OK*. Este procedimento usa a classe `EscreveFicheiro` criada especialmente para este efeito.

Escolhendo o algoritmo dos Filtros para o processo externo é possível escolher uma das 7 medidas para a violação das restrições implementadas (ver FIGURA 8.15).

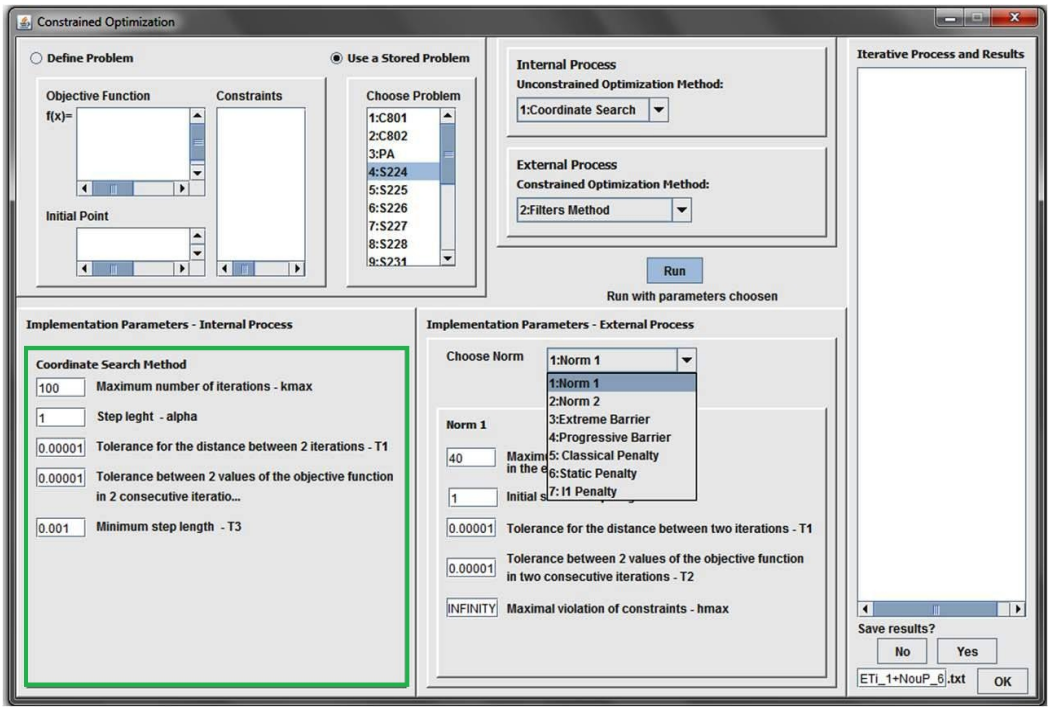

FIGURA 8.15: API/GUI - Opções para a medida de h

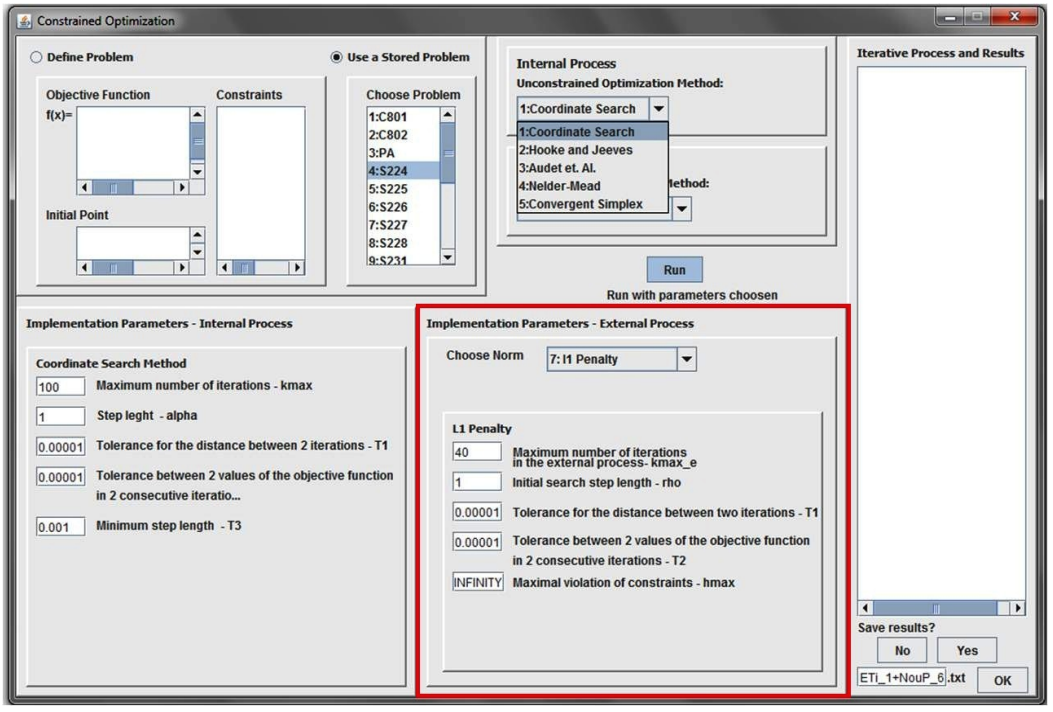

FIGURA 8.16: API/GUI - Opções para o método a usar no processo interno

Os métodos que podem ser usados no processo de optimização sem restrições (processo interno) são os mesmo que estão disponíveis para os Métodos de Penalidade/Barreira (ver FIGURA 8.16).

Dependendo da medida escolhida, aparece no rectângulo inferior direito o correspondente painel de parâmetros, que podem ser alterados pelo utilizador.

Seleccionando, por exemplo a medida Mℓ_1 aparecem os correspondentes parâmetros como se pode ver na FIGURA 8.16, na qual se destacou com um rectângulo vermelho o painel correspondente.

Os restantes painéis, correspondentes às restantes 6 medidas implementadas, podem ser vistos na FIGURA 8.17. Nestes painéis estão por defeito os parâmetros que fora apresentados no Capítulo 7, podendo no entanto ser alterados. Cada um destes parâmetros corresponde a uma variável na API.

O procedimento para executar o algoritmo MF com a GUI e as respectivas opções escolhidas é semelhante ao usado para o algoritmo PenBar. A diferença está na classe e respectivo construtor que são usados. Neste caso a GUI usa o construtor da classe Programa_Filtro, com todos os parâmetros de entrada, para criar um novo objecto, usando-se para isso a instrução:

```
Programa_Filtro F=new Programa_Filtro(
//parametros do processo externo
  pr, norma_ou_phi_a_usar, q, MET_i,
              T1_e, T2_e, kmax_e, rho, hmax,
            //parametros do processo interno
              kmax_i, alpha_i, T1_i, T2_i, T3_i,
              alpha_m_i, alpha_p_i,
              alfa_i, beta_i, gama_i,  s_i,  T4_i,  T5_i,  T6_i,
              gama_s_i,  xi_i, this);
```

Assim, é criado o objecto F e é executado o algoritmo MF através da instrução `F.run();` que executa o método da classe com os dados de entrada introduzidos quando da criação do objecto.

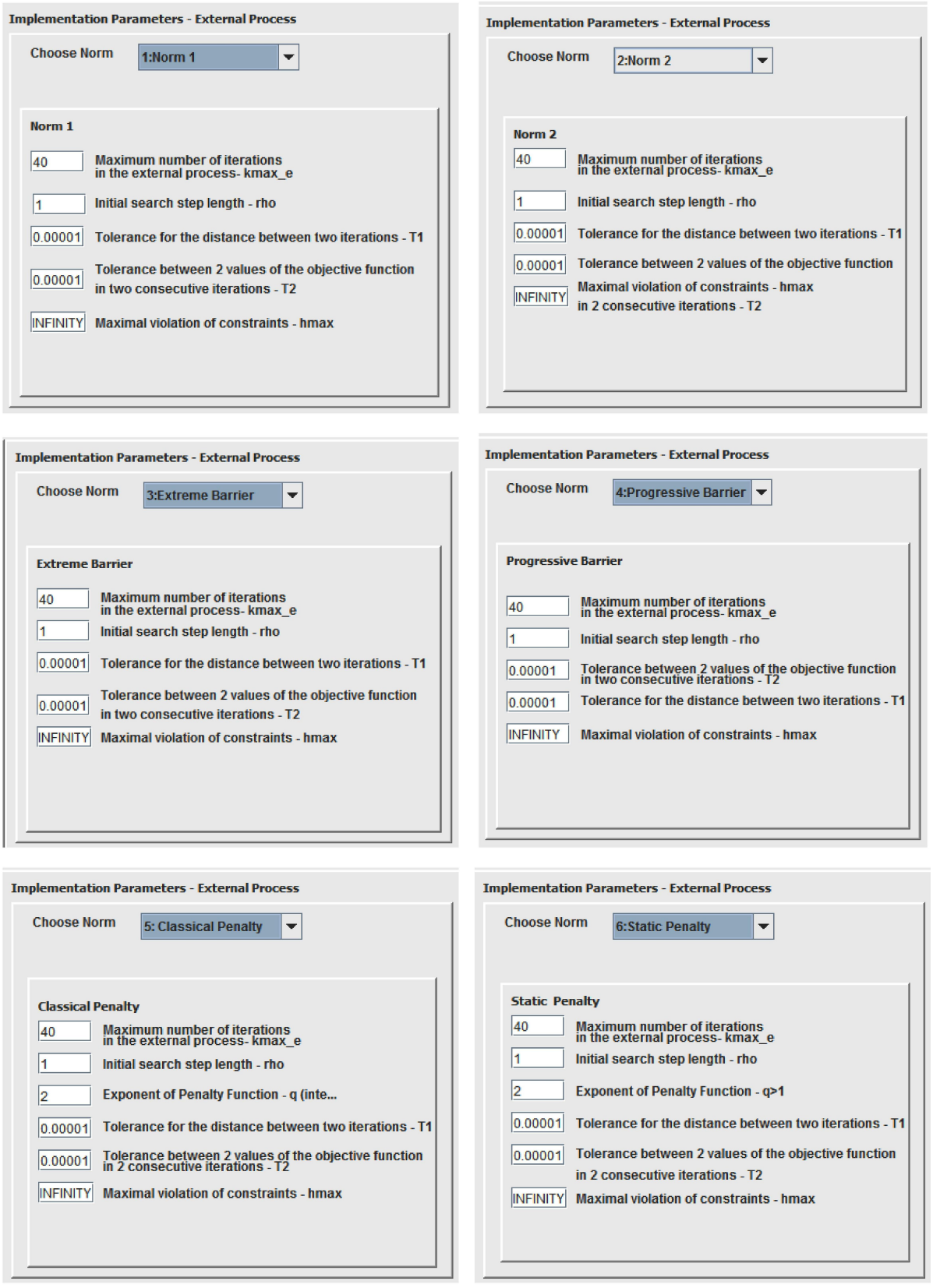

FIGURA 8.17: API/GUI - Painéis Correspondentes as medidas no Método dos Filtros

A execução deste método ainda usa objectos da classe `Filtro`, para testar pontos no Filtro, da classe `NormaFiltro`, para calcular o valor da medida, escolhida pelo utilizador, num determinado ponto e da classe `Criterios`, para calcular os valores das tolerâncias do processo iterativo.

Os resultados obtidos também são apresentados do lado direito, na caixa *Iterative Process and Results* sendo também permitida a gravação dos resultados da execução num ficheiro.

8.1.2 Problemas sem Restrições

Se o problema é um Problema sem Restrições pode, como já foi dito anteriormente, escolher-se 1 de entre os 5 métodos implementados (ver FIGURA 8.18).

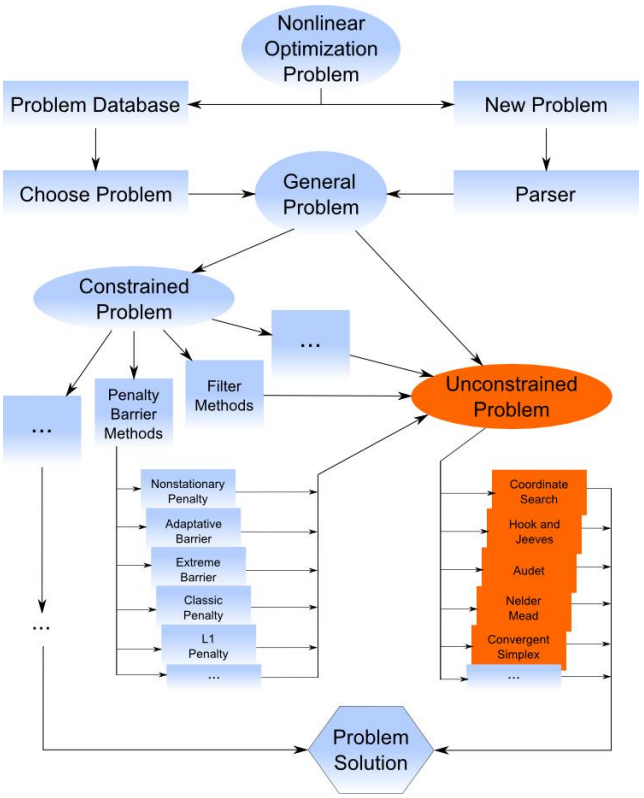

FIGURA 8.18: Métodos para Optimização sem Restrições.
(Fonte: Correia et. al. em [43, 44])

Dependendo do método escolhido, usando a componente gráfica da aplicação aparece no rectângulo inferior esquerdo o correspondente painel de parâmetros, que podem ser alterados pelo utilizador (tanto na componente da GUI para problemas com restrições como para problemas sem restrições).

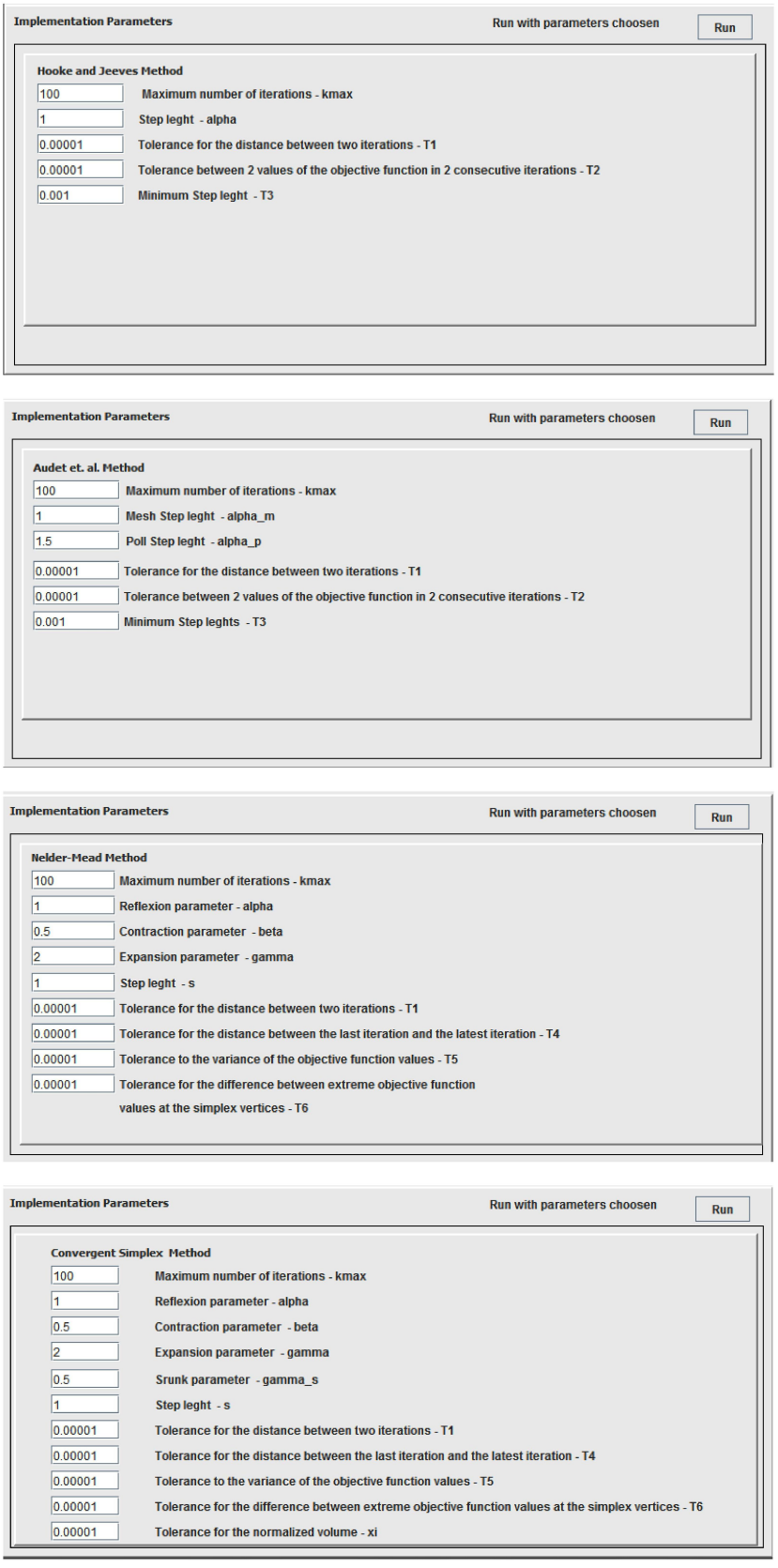

FIGURA 8.19: API/GUI - Painéis Correspondentes aos Métodos de Pesquisa Directa

Pode ver-se, por exemplo o painel correspondentes ao Método CS (integrado num painel do algoritmo MF) na FIGURA 8.15, na qual se destacou com um rectângulo verde o painel correspondente.

Os restantes painéis, correspondentes aos restantes 4 métodos implementadas, podem ser vistos na FIGURA 8.19. Neste painéis estão por defeito os parâmetros que foram apresentados no Capítulo 6, no entanto podem ser alterados, dado que cada um destes parâmetros corresponde a uma variável na API.

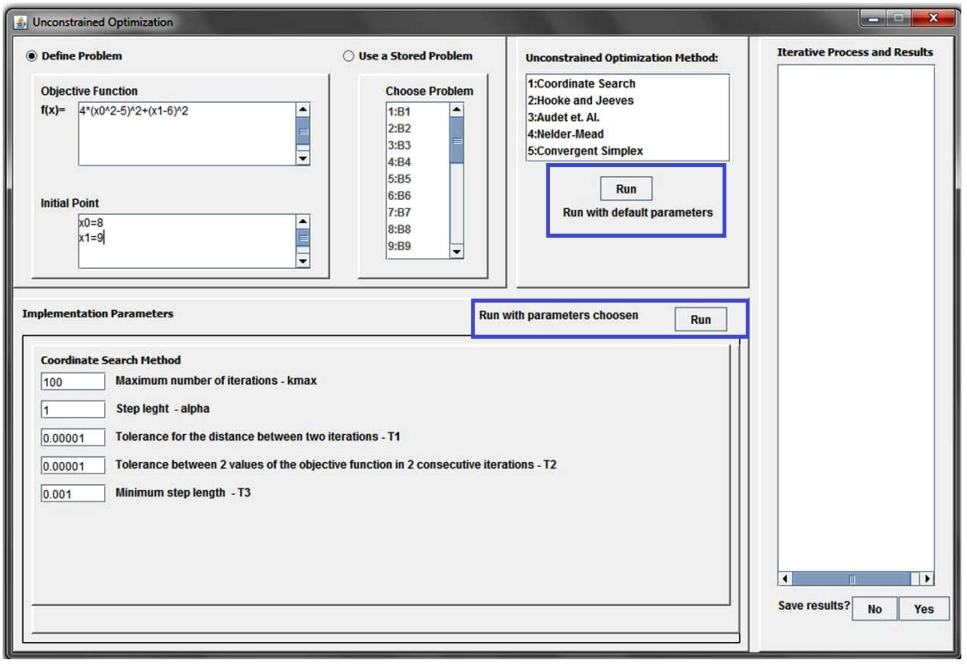

FIGURA 8.20: API/GUI - Opções de execução do algoritmo de Pesquisa Directa

Para executar o algoritmo de Pesquisa Directa, com a GUI e as respectivas opções escolhidas, estão disponíveis duas possibilidades, destacadas na FIGURA 8.20 com rectângulos azuis.

Premindo o botão *Run* - *Run with default parameters* não há leitura dos dados introduzidos nos painéis inferiores e o algoritmo é executado com os parâmetros que estão definidos por defeito, criando um problema ou usando um problema escolhido da lista disponível. Para tal são usadas as instruções:

```
pr = new ProblemaSemRestricoes(
jTextArea_eq.getText(), jTextArea_initialPoint.getText());
xk = MaquinasEstado.generico.SemRestricoes.metodo(MET, pr, this);
```

ou

```
PR = jList_PR.getSelectedIndex() + 1;
pr = SemRestricoes.prob(PR);
xk = MaquinasEstado.generico.SemRestricoes.metodo(MET, pr, this);
```

As primeiras são usadas se o utilizador escolhe a opção de definir o problema. Neste caso é usado o construtor da interface `ProblemaSemRestricoes` para criar um problema (em que os dados de entrada são as *Strings* função objectivo e ponto inicial) e depois é executado o método `metodo` da classe `SemRestricoes` com os dados de entrada `MET` (inteiro que define o método de Pesquisa Directa a usar), `pr` (problema sem restrições criado na instrução anterior a partir dos dados introduzidos) e `this` (correspondente à aplicação que vai mostrar os resultados do método).

A classe `SemRestricoes` é uma classe que, dependendo dos dados de entrada faz executar o método de Pesquisa Directa correspondente. O método `metodo` dessa classe faz com que seja executado o método que foi identificado como o método com o número MET (pode tomar valores de 1 a 5 correspondendo aos 5 Métodos de Pesquisa Directa: CS, HJ, AA, NM ou SC).

As segundas são usadas se o utilizador escolhe a opção de resolver um problema da base de dados. Neste caso é usado o método `prob` da classe `SemRestricoes`, que permite identificar o `ProblemaSemRestricoes pr`, da lista guardada, e depois é executado o método `metodo` da classe `SemRestricoes` com os dados de entrada `MET`, `pr` e `this`.

Premindo o botão *Run - Run with parameters choosen* há leitura dos dados introduzidos no painel correspondente ao método escolhido e o algoritmo é executado com os parâmetros que foram introduzidos.

O procedimento para criar ou escolher o problema a resolver, neste caso, é semelhante ao que foi apresentado anteriormente para o *Run* com os parâmetros por defeito e a execução é feita usando o método `metodo_com_parametros_sem_perg` da classe `SemRestricoes` com todos os dados de entrada. Para tal é usada a instrução:

```
xk = MaquinasEstado.generico.SemRestricoes.
 metodo_com_parametros_sem_perg(
```

```
MET, pr, kmax, alpha,T1, T2, T3, alpha_m, alpha_p,
alfa, beta, gama, s, T4, T5, T6, gama_s, xi, this);
```

A classe `SemRestricoes` foi criada inicialmente na API para não ter que se executar cada uma das classes correspondentes a cada um dos algoritmos de Pesquisa Directa individualmente. Assim, esta classe quando executada directamente (sem ser a partir do GUI) solicita através de mensagens no ecrã, na janela de saída, os vários parâmetros necessários à sua execução. Os dados podem ser escritos na mesma janela e o correspondente algoritmo é executado com esses dados de entrada. Para esta escrita de mensagens e entrada de dados foram usados os procedimentos apresentados por Mendes e Marcelino em [114].

Esta classe tem vários construtores e tem implementados 5 métodos: o usual `main` cuja execução produz o resultado descrito acima, o `metodo` que permite a execução dos algoritmos com os parâmetros por defeito, o `metodo_com_parametros` que permite a execução dos algoritmos com os parâmetros introduzidos na janela de Output em resposta aos pedidos que lá vão sendo apresentados, o `metodo_com_parametros_sem_perg` que permite fazer a mesma execução com os parâmetros como dados de entrada e o `prob` que permite identificar o `ProblemaSemRestricoes pr` da lista guardada, dado um número inteiro `PR` (que pode assumir valores de 1 a 25).

Nos métodos `metodo`, `metodo_com_parametros` e `metodo_com_parametros_sem_perg` é criado um novo objecto de uma das classes `ProgramaCoordenada, Programa Padrão, ProgramaAudet, ProgramaNelder` ou `ProgramaSimplexConv`, dependendo do valor de `MET` ou `MET_i` (no caso da sua execução nos algoritmos PenBar ou MF) através de uma instrução do tipo:

```
ProgramaCoordenada p1 = new ProgramaCoordenada(pr);
```

ou

```
ProgramaCoordenada p1 = new ProgramaCoordenada(pr, kmax, alpha, T1, T2, T3);
```

Neste caso é criado um novo objecto da classe `ProgramaCoordenada`, que corresponde à classe implementada para execução do Método de Pesquisa Coordenada para resolver o `ProblemaGeral pr` ou `ProblemaSemRestrições pr`. Para a criação de objectos

das classes correspondentes aos restantes métodos implementados são usadas instruções semelhantes substituindo a descrição *ProgramaCoordenada* pelos respectivos nomes das classes e adaptando os respectivos dados de entrada.

Depois de criado o objecto é executado o algoritmo, através duma instrução do tipo `p1.run();` que executa o procedimento implementado com os dados de entrada introduzidos quando da criação do objecto.

Assim, a cada Método de Pesquisa Directa corresponde uma classe que, à semelhança da classe `SemRestricoes`, pode ser executada directamente ou ser invocada de outras classes.

A execução destas classes ainda usa objectos da classe `Criterios` para calcular os valores das tolerâncias do processo iterativo.

Na componente gráfica para Optimização sem Restrições os resultados obtidos também são apresentados do lado direito, na caixa *Iterative Process and Results*, tal como acontece para a Optimização com Restrições, e também é possível gravar os resultados da execução num ficheiro seleccionando a opção *Yes*, escolhendo o nome que ser quer dar ao ficheiro e fazendo *OK* (ver FIGURA 8.21).

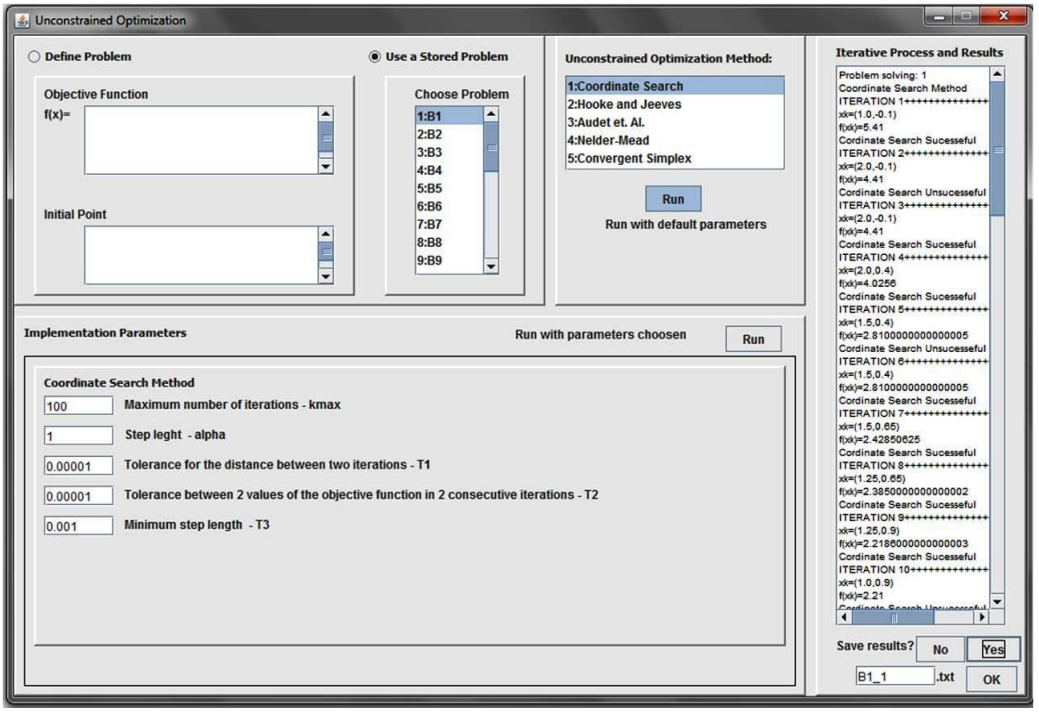

FIGURA 8.21: API/GUI - Apresentação dos resultados do Processo iterativo Optimização sem Restrições

8.2 Utilização da API

Nesta secção descreve-se de que forma se pode utilizar a API e baseia-se essencialmente no apresentado em [112].

A API implementada pode ser usada instalando-a localmente no computador onde está a ser utilizada, nesse caso pode ser usada como qualquer outra API em Java, instalando o ficheiro `.jar` que contém todas as classes implementadas.

Para usar, por exemplo, o algoritmo de Audet para minimizar a função `expr` pode usar-se o código Java:

```
String expr = ‘‘(x0-2)^2 + (x1+2)^2’’;
String initPoint = ‘‘x0=0.0 x1=0.0’’;
ProgramaAudet audet =
       new AudetProgram(expr,initPoit);
audet.run();
double[] result = audet.getLastResult();
```

Os parâmetros de entrada (função objectivo e ponto inicial), neste exemplo, são enviados para o algoritmo usando o construtor da classe ProgramaAudet. Para resolver o problema é chamado o método `run()` e os resultados podem ser obtidos invocando o método `getLastResult()`, que devolve a aproximação obtida como *array* com a dimensão 2. Também podem ser invocados os métodos `setInitialPoint()` e `setExpression()` para alterar o ponto inicial ou a função objectivo.

Este acesso só pode ser feito se a API estiver instalada no computador onde forem necessários os métodos implementados e utilizando Java, uma vez que é a tecnologia em que a API está implementada.

Tendo em conta esta limitação pretende-se no futuro possibilitar o seu acesso remoto podendo tal ser feito através de LAN (Local Area Network) ou de internet, usando Web Services. Esta opção tem como objectivo permitir que se aceda sempre à última versão implementada da API. Como se pretendem implementar mais métodos do que os que neste trabalho foram implementados o acesso desta forma poderá ser uma mais valia no trabalho futuro. Além disso, da forma referida é possível aceder à aplicação não apenas

usando Java, mas também outras linguagens de programação, tais como FORTRAN, C, C++ ou plataformas .NET (C# por exemplo), segundo van Engelen em [150].

Também se pretende desenvolver uma aplicação Web, que já foi iniciada, mas ainda se encontra numa fase muito inicial. A versão que já está implementada permite escolher o tipo de problema que se pretende resolver (com ou sem restrições), definir um problema novo ou escolher um dos problemas já implementados e escolher um método para o resolver. Esta aplicação foi desenvolvida usando JSP (Java Server Pages) e interage directamente com a API. O objectivo desta aplicação é permitir que seja utilizada para resolver problemas particulares, ou seja, quando não há a intenção de integrar os métodos implementados noutras aplicações. Nas FIGURAS 8.22 e 8.23 apresenta-se a interface da aplicação que está neste momento implementada, respectivamente, para Problemas sem e com restrições.

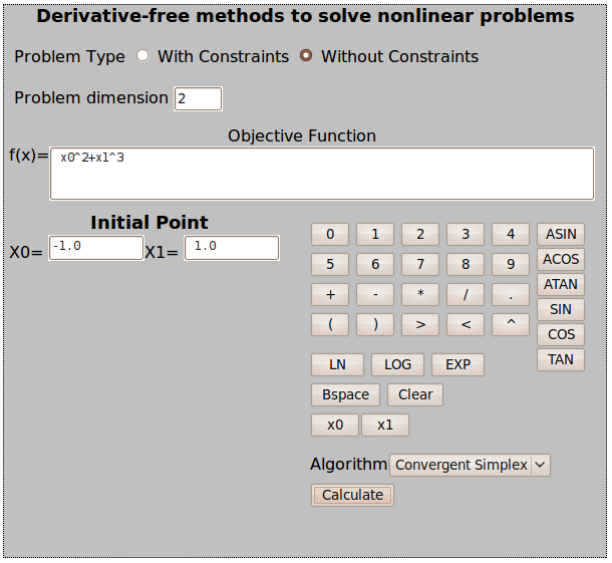

FIGURA 8.22: Interface da aplicação Web em desenvolvimento, para Optimização sem Restrições. (Fonte: Correia et. al. em [112])

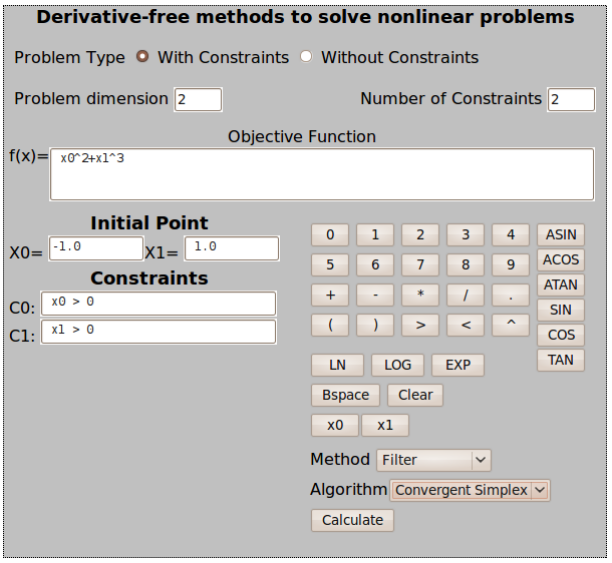

FIGURA 8.23: Interface da aplicação Web em desenvolvimento, para Optimização com Restrições. (Fonte: Correia et. al. em [112])

Parte IV

Conclusões, Principais Contribuições e Trabalhos Futuros

Capítulo 9

Conclusões, Principais Contribuições e Trabalhos Futuros

9.1 Conclusões e Considerações Finais

Neste trabalho estudaram-se, implementaram-se e compararam-se algoritmos para resolver Problemas de Optimização não Linear com e sem Restrições, sem recorrer ao uso de derivadas das funções envolvidas ou suas aproximações, nem à aproximação das funções através de modelos.

Pare se cumprir este objectivo geral, foi feita uma pesquisa geral sobre Métodos de Optimização, apresentando-se no Capítulo 1 os resultados considerados como essenciais à introdução do tema tratado.

Os Métodos de Pesquisa Directa (para Optimização sem Restrições) foram abordados no Capítulo 2 onde se incluem os aspectos essenciais dos métodos clássicos e se apresentam algumas das suas variantes que têm sido objecto de investigação de diversos autores. São, assim, incluídas referências a recentes desenvolvimentos baseados nestes métodos, que se verificou serem bastante incisivos em questões de prova de convergência.

Com o objectivo de fazer uma comparação dos recentes desenvolvimentos apresentados com os Métodos de Pesquisa Directa Clássicos, foram estudados (com resultados apresentados no Capítulo 2), implementados e comparados (com resultados apresentados no Capítulo 6) cinco Métodos de Pesquisa Directa: três Métodos de Pesquisa em Padrão (o

clássico Método de Pesquisa Coordenada e duas variantes deste método, correspondentes ao Método de Hooke-Jeeves e a uma versão dos Métodos de Audet et. al.) e dois Métodos Simplex (o clássico Método de Nelder-Mead e uma variante, denominada por Método Simplex Convergente).

No Capítulo 2 foram apresentados os resultados do estudo destes Métodos. Estes estudo permitiu, no Capítulo 6, apresentar os pormenores da implementação que foi realizada neste trabalho, bem como testar os correspondentes algoritmos usando para tal um conjunto de problemas teste. Estas implementações constituem implementações de métodos referidos e apresentados na literatura, mas as versões de implementação neste trabalho diferem em diversos aspectos específicos de implementação dos métodos originais, conforme foi apresentado no Capítulo 6.

A realização destes testes permitiu verificar que os algoritmos de Hooke-Jeeves e Nelder-Mead foram os que possibilitaram obter melhores resultados para um maior número dos problemas teste escolhidos, com a utilização dos parâmetros por defeito. No entanto, existirão problemas para os quais outros métodos terão melhor desempenho. Assim, na impossibilidade de se testarem todos os algoritmos implementados, sugerem-se os métodos anteriormente referidos, que se mostraram mais eficientes e que, por isso, dão mais garantia de êxito. Também se verificou que em alguns problemas a alteração de parâmetros de entrada, nomeadamente do tamanho do passo inicial, produz, em muitos dos resultados obtidos, uma melhoria da qualidade das soluções, pelo que esta também poderá ser uma possibilidade a implementar no caso de não se obter uma aproximação adequada à situação em questão.

Assim, considerando o algoritmo de Hooke-Jeeves como um método clássico de Pesquisa Directa, conclui-se que as versões implementadas, que incluem algumas das sugestões apresentadas nos trabalhos mais recentes nesta área, não se mostraram mais eficientes do que os métodos clássicos, considerando os problemas testados.

Note-se que estes métodos para além dos valores da função objectivo não usam mais informação pelo que se pode considerar que o seu desempenho se mostrou bastante satisfatório uma vez que possibilitaram encontrar aproximações *Boas* para a maioria dos problemas, mesmo sem usar a informação que outros métodos precisam para o fazer.

No Capítulo 8 foi apresentada genericamente a estrutura da API implementada, dando destaque às classes correspondentes a cada um dos algoritmos de Pesquisa Directa, bem como as linhas gerais de como se podem ser usadas para resolver um problema sem restrições. Foi ainda apresentada a interface gráfica de uma aplicação teste implementada e baseada na API desenvolvida. Esta aplicação não sendo objectivo deste trabalho foi considerada como a melhor forma para dar acesso à API mesmo para quem não esteja familiarizado com a Tecnologia Java.

No que respeita aos Métodos para Optimização com Restrições, foram estudados alguns dos Métodos de Penalidade e Barreira, Métodos de Lagrangeana Aumentada e Métodos dos Filtros (com resultados apresentados no Capítulo 3).

O Método de Lagrangeana Aumentada não foi implementado por trabalhar com Métodos de Optimização sem Restrições generalizados, ou seja, com técnicas específicas para lidar com restrições do tipo limite simples. Como os Métodos de Pesquisa Directa implementados não possuíam estas características optou-se por não implementar este método, não se colocando no entanto de parte a possibilidade de o fazer em trabalhos futuros.

Dos Métodos de Penalidade e Barreira foram seleccionados cinco para implementar (com resultados apresentados no Capítulo 7), sendo dois deles Métodos de Barreira e três de Penalidade. Alguns destes métodos nunca foram testados em associação com alguns dos Métodos de Pesquisa Directa implementados. Esta escolha permitiu assim disponibilizar funções desde as mais clássicas às mais recentes na área.

No Capítulo 7 apresentam-se os pormenores da implementação que foi realizada neste trabalho, no que respeita a Métodos de Pesquisa Directa para Optimização com Restrições, bem como a estrutura e parâmetros usados nos correspondentes algoritmos. Os resultados numéricos obtidos para um conjunto de problemas teste, são também aqui apresentados e analisados.

O algoritmo PenBar apresentado, difere dos usuais Métodos de Penalidade e Barreira, nomeadamente no tipo de dados de saída que apresenta. Com a motivação de o Método dos Filtros devolver um conjunto de possíveis soluções para um dado problema (pontos do conjunto Filtro), foram implementados procedimentos para que o algoritmo PenBar também guardasse e devolvesse informação não só da última aproximação encontrada

no processo iterativo (com melhor valor da função Penalidade /Barreira), mas que também guardasse e devolvesse a melhor aproximação admissível à solução (a aproximação admissível com menor valor da função objectivo) e a melhor aproximação não admissível à solução (a aproximação não admissível com menor valor da violação das restrições) encontradas durante o processo iterativo. Estes procedimentos podem ser importantes se a aproximação usualmente apresentada por algoritmos de Penalidade / Barreira, apesar de ter o melhor valor da Função de Penalidade/Barreira, não tiver os melhores valores encontrados para a função objectivo e para a violação das restrições, o que pode ser muito importante, principalmente no caso de problemas relaxáveis.

Este procedimento já tinha sido implementado neste trabalho, na versão implementada do Métodos dos Filtros por se ter verificado que o Filtro final é constituído, genericamente, por um conjunto de vários pontos e se ter considerado que a selecção dessas duas aproximações poderia ser uma mais valia na apresentação de resultados.

A implementação do Método dos Filtros que foi efectuada teve que ser idealizada, uma vez que as indicações sobre os algoritmos apresentados na literatura para este tipo de Métodos em conjugação com Métodos de Pesquisa Directa (apresentadas no Capítulo 3) eram demasiado genéricos e não permitiam uma clara linha de orientação e de construção do método. Assim, para esta implementação foram testadas diversas estratégias e acrescentadas sucessivamente alterações à primeira versão do algoritmo, no sentido da melhoria dos resultados obtidos, conforme foi apresentado no capítulo 7. Neste Capítulo ainda se apresentaram os pormenores da implementação que foi realizada neste trabalho, bem como os parâmetros usados nos correspondentes algoritmos e os resultados numéricos para um conjunto de problemas teste. Deste modo verificou-se ser viável a aplicação do Método dos Filtros associado com Pesquisa Directa, para Optimização com Restrições, mostrando-se ser eficiente em comparação com outros métodos propostos na literatura.

O algoritmo PenBar ainda serviu de motivação para que fossem implementadas medidas alternativas à usual norma 2 (N2) para medir a violação das restrições no Método dos Filtros. Neste trabalho foram testadas, além da N2, mais seis medidas dessa violação. Estas medidas incluem a usual norma 1 (N1), bem como outras medidas que foram adaptadas das definições das Funções Barreira e Penalidade estudadas.

Os resultados apresentados no capítulo 7 permitiram ainda observar a obtenção de melhores resultados para os problemas teste seleccionados, com os parâmetros por defeito,

para Função de Penalidade ℓ_1 em combinação com a maioria dos Métodos de Pesquisa Directa, nomeadamente quando é usado o Método de Nelder-Mead no processo interno. No entanto este método não permitiu encontrar aproximações à solução de alguns problemas, que para outros métodos se tornaram problemas simples de resolver.

A função de Barreira Extrema conjugada com o Método de Nelder-Mead também permitiu obter resultados satisfatórios quando é imperativo que a aproximação à solução obtida seja admissível. Existindo alguma flexibilidade na violação das restrições, pode considerar-se o Método de Penalidade Clássica com o qual foi encontrado um número bastante satisfatório de *Boas* aproximações não admissíveis à solução, isto tendo em atenção os critérios definidos.

O Método dos Filtros não se mostrou tão eficiente como os Métodos de Penalidade/Barreira a encontrar *Boas* aproximações admissíveis. No entanto foi essencial para motivar as alterações que foram feitas ao algoritmo PenBar e que permitiram encontrar muitas das *Boas* aproximações, segundo os critérios definidos, obtidas por estes métodos.

Considerando os totais de aproximações admissíveis à solução, os resultados obtidos com o algoritmo MF são similares aos obtidos com o algoritmo PenBar, sendo que usando os métodos CS, HJ e NM no processo interno foi possível encontrar mais aproximações à solução dos problemas teste, associando-os com o algoritmo PenBar. Por outro lado, com os métodos AA e SC encontraram-se mais aproximações à solução, associando-os com o algoritmo MF.

Tendo em conta os resultados obtidos com o algoritmo MF verificou-se que a adopção de medidas alternativas para a violação das restrições, nomeadamente da N1 e NP conduz a resultados muito similares aos que são obtidos com a usual N2, pelo que estas medidas podem ser consideradas como alternativas válidas para a medição dessa violação. A medida NEB, à semelhança do que acontece com a função Barreira Extrema, não permitiu encontrar aproximações não admissíveis e não se mostrou muito eficaz a encontrar aproximações admissíveis, tendo sido a medida com piores prestações, pelo que não será uma boa alternativa de medição da violação das restrições.

Assim, não sendo possível testar todos os métodos/algoritmos para um novo problema, o que seria desejável pois nesse caso poder-se-ia escolher a aproximação que mais se

adequasse à situação em questão, ficam aqui as indicações dos que se mostraram mais eficientes para os problemas teste seleccionados.

Note-se, mais uma vez, que estes métodos não usam mais informação, para além dos valores da função objectivo e das restrições. Por este facto considerou-se o desempenho dos algoritmos bastante satisfatório uma vez que possibilitaram encontrar *Boas* aproximações para a maioria dos problemas, mesmo sem usar a informação necessária a outros métodos. Além disso pode considerar-se que os critérios definidos para classificar as aproximações, tanto no caso de Problemas sem Restrições como no caso de Problemas com Restrições, tiveram um grau de exigência considerável, pois ao fixar os graus de tolerância para a classificação das aproximações em 10^{-5} exigiu-se que as aproximações tivessem pelo menos quatro casas decimais significativas correctas para serem consideradas *Boas* aproximações.

Deste modo, o conhecimento das características dos métodos já implementados e estudados, permitiu verificar a possibilidade de desenvolver novas metodologias de resolução de problemas, adoptando procedimentos de uns, para a melhoria e alargamento das opções de resolução disponíveis de outros.

A implementação de todos estes algoritmos conduziu, assim, ao desenvolvimento de uma API usando a Tecnologia Java, apresentada mais em pormenor no Capítulo 8. Esta API permite uma interacção com todos os métodos e metodologias desenvolvidas, sendo constituída por classes correspondentes a cada um dos métodos e algoritmos.

Para testar estas e outras metodologias a desenvolver ainda se implementaram alguns problemas teste, estando estes disponíveis para o utilizador da aplicação.

Com a aplicação gráfica desenvolvida é possível visualizar o decorrer de todo o processo iterativo, bem como posterior registo da informação gerada e utilizada pelos métodos durante a execução. Esta execução pode ser efectuada, pela forma referida, utilizando o GUI, mas também pela chamada directa das classes correspondentes ao algoritmo desejado.

De uma maneira geral, a interface gráfica permite que o utilizador possa testar os algoritmos implementados necessitando apenas de introduzir os dados de entrada, definir ou escolher um problema a resolver, manter ou alterar parâmetros definidos por defeito,

executar os algoritmos, visualizar os resultados ao longo do processo iterativo e os resultados finais e ainda guardar esses resultados num ficheiro `.txt` com o nome à sua escolha.

9.2 Sumário das Principais Contribuições

As principais contribuições deste trabalho são consequência directa da tentativa de cumprimento dos objectivos inicialmente propostos e cuja ideia geral já se apresentou na secção anterior. No entanto, podem destacar-se, sumariamente, aquelas que se consideram ser as contribuições chave para a questão central deste trabalho.

As primeiras contribuições, que resultaram do estudo dos algoritmos apresentados na literatura para Optimização através de Pesquisa Directa (com e sem Restrições), foram a realização de sínteses sobre os temas tratados. Algumas delas, nomeadamente a introdução geral sobre Optimização não Linear, no Capítulo 1, a síntese sobre Métodos de Pesquisa Directa clássicos e versões recentes, no Capítulo 2, a síntese sobre métodos de Penalidade e Barreira e sua classificação (que foi baseada nos trabalhos [36] e [40]) e sobre o Métodos dos Filtros, no Capítulo 3.

As primeiras implementações dos Métodos de Pesquisa Directa clássicos permitiram, ainda numa fase inicial deste trabalho, a apresentação dos primeiros resultados numéricos em [35]. No Capítulo 6 expõem-se os pormenores de implementação destes métodos e de novas versões recentemente propostas na literatura, dando ênfase às especificidades das versões adoptadas e implementadas neste trabalho e à comparação de resultados obtidos. Nesta exposição apresentam-se contribuições que se podem considerar pertinentes uma vez que os desempenhos de alguns dos métodos ainda não tinham sido comparados, nomeadamente as versões das novas abordagens com os métodos clássicos e entre si.

As implementações de Métodos de Penalidade e Barreira tiveram como base as pesquisas realizadas, apresentadas no Capítulo 3, no entanto, o algoritmo PenBar final implementado possui diversas alterações, tal como foi apresentado no Capítulo 7. Uma das alterações adoptadas foi o procedimento de guardar e devolver no final do processo as aproximações com melhores valores das funções envolvidas. Este procedimento, embora simples de implementar, nunca tinha sido adoptado por nenhum autor. O mais usual é que um Método de Penalidade ou Barreira devolva apenas a iteração final obtida no

processo, ou seja, a que tem menor valor da Função Penalidade/Barreira. Outro ponto importante neste aspecto é que tenham sido usados no processo interno Métodos de Pesquisa Directa. Grande parte dos resultados das implementações de Métodos de Penalidade e Barreira publicados, usam nos seus processos internos, métodos baseados em derivadas, pelo que não era conhecido o seu desempenho quando implementados em conjugação com Métodos de Pesquisa Directa. Alguns destes resultados foram apresentados em [45].

Relativamente ao Método dos Filtros foram dadas diversas contribuições, uma vez que este método, como Método de Pesquisa Directa para Optimização com Restrições, só tinha sido implementada por Charles Audet e o seu grupo de trabalho e apenas em conjugação com o Método de Pesquisa Coordenada. Assim, houve durante a elaboração deste trabalho, diversos desenvolvimentos e versões implementadas, uma vez que não havia informação disponível sobre a melhor forma de o fazer em conjugação com outros Métodos de Pesquisa Directa, pelo que o trabalho aqui apresentado é quase integralmente contributo novo para este tema.

As primeiras sugestões de implementação e resultados numéricos desta abordagem foram apresentados em [38]. Neste trabalho compararam-se os desempenhos das implementações do Método dos Filtros em conjugação com o Método de HJ e em conjugação com o Método NM, para alguns problemas teste. Além disso, já se apresentam como resultados em análise, as melhores aproximações admissíveis e não admissíveis. Esta informação não é usual no Método dos Filtros, como não o era nos Métodos de Penalidade e Barreira. O resultado usual do Método dos Filtros é o conjunto de pontos que pertencem ao Filtro final. Esta ideia também veio depois a ser implementada no algoritmo PenBar, conforme se apresentou no Capítulo 7.

Sendo essas ideias a base do algoritmo agora implementado, houve ainda algumas alterações que foram incluídas e comparações que foram feitas. Em [41, 111] foi apresentado um novo esquema de implementação com as alterações que já foram descritas no Capítulo 7, tendo-se feito uma comparação do desempenho deste novo esquema com o que tinha sido apresentado anteriormente em [38].

Em [39] foi comparado o Método dos Filtros conjugado com o Método de Nelder-Mead com o Método de Penalidade ℓ_1 conjugado com o mesmo método de Pesquisa Directa. E no ano seguinte, em [42], apresentaram-se alguns dos resultados do Método dos Filtros

conjugado com qualquer um dos Métodos de Pesquisa Directa e com as novas alternativas de medidas para a violação de restrições.

Os resultados que foram sendo obtidos e as alterações que foram sendo implementados foram objecto de mais algumas apresentações genéricas do trabalho desenvolvido até então, como é o caso de [37, 43, 44, 46].

No que respeita às várias componentes da API, estas foram sendo desenvolvidas e apresentadas na maioria dos trabalhos referidas anteriormente tendo-se dado uma visão mais geral em [112].

Deste modo, as principais contribuições de síntese são expostas nos Capítulos iniciais (1,2 e 3) e as principais contribuições práticas são descritas nos Capítulos 7 e 8.

A API, apresentada no Capítulo 8, consiste também numa contribuição nesta área, uma vez que, pela tecnologia que está na sua base ser a Tecnologia Java, com as suas características de portabilidade e eficiência quando comparada com outras tecnologias, pelas possibilidades de acesso remoto, seja a classes específicas implementadas ou a toda a aplicação e de acesso através de WebServices, permite que a aplicação adopte essas características e possa ser utilizada por um número vasto de utilizadores. Não estando todas estas formas de acesso ainda em funcionamento, são no entanto um potencial em aberto que pode ser explorado no futuro.

9.3 Trabalho Futuro

Ao longo deste trabalho foram já apresentadas algumas direcções para trabalho futuro, no âmbito do tema tratado. Dessas direcções podem destacar-se algumas, que se consideram ser pertinentes e ainda outras que não tendo sido referidas têm sido discutidas como futuras possibilidades de investigação.

As principais linhas orientadores e intenções de implementação para trabalhos futuros são, assim:

- Implementar Métodos de Pesquisa Directa generalizados, capazes de lidar com restrições do tipo limites simples, no sentido de alargar o tipo de métodos disponíveis

na API e de tornar possível implementar outros métodos, como é o caso dos Métodos de Lagrangeana Aumentada;

- Fazer a análise dos efeitos da escolha/definição de parâmetros de penalidade, na tentativa de melhorar o desempenho do algoritmo PenBar, dado que esse estudo não foi efectuado neste trabalho;

- Não tendo sido testados os algoritmos para resolução de problemas com restrições de igualdade, uma vez que se fez a sua substituição por 2 restrições de desigualdade, pretende testar-se o desempenho dos algoritmos implementados com problemas com esse tipo de restrições, no sentido de averiguar a aplicabilidade na sua resolução, bem como testar novas metodologias no caso de se verificar fraco desempenho;

- Uma situação semelhante foi verificada para o Método dos Filtros, uma vez que algumas das medidas estão desenhadas para problemas com restrições apenas de igualdade. Assim, adoptar novos procedimentos ou fazer adaptações às expressões utilizadas para a medição da violação das restrições, é outra das ideias a seguir no futuro;

- Pretende ainda fazer-se uma ligação da API com uma base de dados (por exemplo *MySQL*), para que seja possível incluir novos problemas ou eliminar outros directamente da base de dados, consultar resultados obtidos em execuções anteriores, construir gráficos de resultados anteriormente obtidos, entre outras possibilidades que permitam este tipo de ligações;

- Outro objectivo para o futuro é incluir novos métodos de optimização para problemas sem e com restrições na API, nomeadamente aqueles que, não sendo da área agora em estudo, se mostram mais eficazes na resolução de problemas com características de diferenciabilidade, convexidade, etc., para os quais os métodos agora implementados não são os mais adequados;

- Os métodos de Optimização Global, conjugados com os métodos agora implementados para o refinamento local de soluções, também são um assunto que se tem intenção de estudar;

- Pretende ainda efectivar-se a possibilidade de acesso remoto à API, disponibilizando a aplicação num servidor e melhorar a sua componente correspondente à aplicação Web.

Apêndice A

Problemas

Este Apêndice contém os problemas utilizados nos testes computacionais realizados para comparar os resultados numéricos obtidos com os diversos algoritmos apresentados.

Assim, na secção A.1 apresentam-se 25 Problemas Sem Restrições, sendo que:

- 10 foram seleccionados da colecção de Bagirov em [15]: B1, B2, B3, B4, B5, B6, B7, B8, B9 e B10 (secção A.1.1);
- 13 pertencem à colecção de Schittkowski em [138]: S201, S202, S205, S206, S207, S208, S211, S212, S213, S240, S246, S256 e S257 (secção A.1.2);
- os dois últimos são problemas de Maratos: M500 e M501 (secção A.1.3).

Na secção A.2 apresentam-se 18 Problemas Com Restrições:

- os dois primeiros são problemas da colecção CUTE: C801; C802 (secção A.2.1);
- o problema PA - que foi utilizado nos testes numéricos do trabalho de Correia et. al. [38] para comparar o desempenho do algoritmo filtro simplex com o algoritmo de Audet et. al. [8] (secção A.2.2);
- 15 foram seleccionados da colecção de Schittkowski em [138]: S224; S225; S226; S227; S228; S231; S233; S234; S249; S264; S270; S323; S324; S325; S326 (secção A.2.3).

A.1 Problemas sem Restrições

A.1.1 Problemas de Bagirov

Problema	Dimensão	Ponto Inicial	Solução
B1	$n = 2$	$x = (1, -0.1)$ $f(x) = 5.41$	$x^* = (1.1390, 0.8996)$ $f(x^*) = 1.952224$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv \max\left[x_1^2 + x_2^4, (2 - x_1)^2 + (2 - x_2)^2, 2e^{-x_1 + x_2}\right]$			

Problema	Dimensão	Ponto Inicial	Solução
B2	$n = 2$	$x = (-1, 5)$ $f(x) = 56$	$x^* = (1.2, 2.4)$ $f(x^*) = 7.2$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv \max[x_1^2 + x_2^2, x_1^2 + x_2^2 + 10(-x_1 - x_2 + 4), x_1^2 + x_2 + 10(-x_1 - 2x_2 + 6)]$			

Problema	Dimensão	Ponto Inicial	Solução
B3	$n = 4$	$x = (1, 2, 1, 1)$ $f(x) = 13$	$x^* = (0, 0, 0, 0)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^4} \quad f(x) \equiv 4\lvert x_1 - x_2\rvert + \lvert x_1 + x_2\rvert + \lvert x_2 - x_3\rvert + \lvert x_2 + x_3\rvert + \lvert x_3 - x_4\rvert + \lvert x_3 + x_4\rvert$			

Problema	Dimensão	Ponto Inicial	Solução
B4	$n = 5$	$x = (1, 2, -3, -4, -5)$ $f(x) = 25$	$x^* = (0, 0, 0, 0, 0)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^5} \quad f(x) \equiv \max\left[x_1^2, x_2^2, x_3^2, x_4^2, x_5^2\right]$			

Problema	Dimensão	Ponto Inicial	Solução
B5	$n = 5$	$x = (1, 2, -3, -4, -5)$ $f(x) = 5$	$x^* = (0, 0, 0, 0, 0)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^5} \quad f(x) \equiv \max\left[\lvert x_1\rvert, \lvert x_2\rvert, \lvert x_3\rvert, \lvert x_4\rvert, \lvert x_5\rvert\right]$			

Problema	Dimensão	Ponto Inicial	Solução
B6	$n = 5$	$x = (0, 0, 0, 0, 0)$ $f(x) = 46.06816666$	$x^* = \left(\frac{1}{n}, \frac{1}{n}, \frac{1}{n}, \frac{1}{n}, \frac{1}{n}\right)$ $f(x) = 0$
$f(x) \equiv \sum_{j=1}^{100} \left\lvert \sum_{i=1}^{n} (x_i - x_i^*)(0.01j)^{i-1} \right\rvert$			

Problema	Dimensão	Ponto Inicial	Solução
B7	$n = 2$	$x = (-1.2, 1)$ $f(x) = 22.2$	$x^* = (1, 1)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv \lvert x_1 - 1 \rvert + 100 \lvert x_2 - \lvert x_1 \rvert \rvert$			

Problema	Dimensão	Ponto Inicial	Solução
B8	$n = 4$	$x = (1, 3, 3, 1)$ $f(x) = 402.2$	$x^* = (1, 1, 1, 1)$ $f(x^*) = 0$
$\underset{x \in \mathbb{R}^4}{minimize}$	$f(x) \equiv \lvert x_1 - 1 \rvert + 100 \lvert x_2 - \lvert x_1 \rvert \rvert + 90 \lvert x_4 - \lvert x_3 \rvert \rvert + \lvert x_3 - 1 \rvert +$ $+10.1 (\lvert x_2 - 1 \rvert + \lvert x_4 - 1 \rvert) + 4.95 (\lvert x_2 + x_4 - 2 \rvert - \lvert x_2 - x_4 \rvert)$		

Problema	Dimensão	Ponto Inicial	Solução
B9	$n = 5$	$x = (1, 2, -3, -4, -5)$ $f(x) = 26$	$x^* = (0, 0, 0, 0, 0)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^5} \quad f(x) \equiv \max\left[x_1^2, x_2^2, x_3^2, x_4^2, x_5^2\right] + \min\left[x_1^2, x_2^2, x_3^2, x_4^2, x_5^2\right]$			

Problema	Dimensão	Ponto Inicial	Solução
B10	$n = 5$	$x = (1, 1, 1, 1, 1)$ $f(x) = 34.16333$	$x^* = \left(\frac{1}{n}, \frac{1}{n}, \frac{1}{n}, \frac{1}{n}, \frac{1}{n}\right)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^5} \quad f(x) \equiv \sum_{j=1}^{20} \left\lvert \sum_{i=1}^{n} (x_i - x_i^*)(0.05j)^{i-1} \right\rvert - \max_{j=1,\ldots,20} \left\{ \left\lvert \sum_{i=1}^{n} (x_i - x_i^*)(0.05j)^{i-1} \right\rvert \right\}$			

A.1.2 Problemas de Schittkowski

Problema	Dimensão	Ponto Inicial	Solução
S201	$n = 2$	$x = (8, 9)$ $f(x) = 45$	$x^* = (5, 6)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv 4(x_1 - 5)^2 + (x_2 - 6)^2$			

Problema	Dimensão	Ponto Inicial	Solução
S202	$n = 2$	$x = (6, 3)$ $f(x) = 866$	$x^* = (5, 4)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv \left(-13 + x_1 - 2x_2 + 5x_2^2 - x_2^3\right)^2 + \left(-29 + x_1 - 14x_2 + x_2^2 + x_2^3\right)^2$			

Problema	Dimensão	Ponto Inicial	Solução
S205	$n = 2$	$x = (2, 0.2)$ $f(x) = 0.529781$	$x^* = (3, 0.5)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv [1.5 - x_1(1 - x_2)]^2 + \left[2.25 - x_1\left(1 - x_2^2\right)\right]^2 + \left[2.625 - x_1\left(1 - x_2^3\right)\right]^2$			

Problema	Dimensão	Ponto Inicial	Solução
S206	$n = 2$	$x = (-1.2, 1)$ $f(x) = 484.1936$	$x^* = (1, 1)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv \left(x_2 - x_1^2\right)^2 + 100(1 - x_1)^2$			

Problema	Dimensão	Ponto Inicial	Solução
S207	$n = 2$	$x = (-1.2, 1)$ $f(x) = 5.0336$	$x^* = (1, 1)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv \left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$ Função Banana			

Problema	Dimensão	Ponto Inicial	Solução
S208	$n = 2$	$x = (-1.2, 1)$ $f(x) = 24.2$	$x^* = (1, 1)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$ Função de Rosenbrock			

Problema	Dimensão	Ponto Inicial	Solução
S211	$n = 2$	$x = (-1.2, 1)$ $f(x) = 749.0384$	$x^* = (1, 1)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv 100\left(x_2 - x_1^3\right)^2 + (1 - x_1)^2$			

Problema	Dimensão	Ponto Inicial	Solução
S212	$n = 2$	$x = (2, 0)$ $f(x) = 100$	$x^* = (0, 0)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv [4(x_1 + x_2)]^2 + \left[4(x_1 + x_2) + (x_1 - x_2)\left((x_1 - 2)^2 + x_2^2 - 1\right)\right]^2$			

Problema	Dimensão	Ponto Inicial	Solução
S213	$n = 2$	$x = (3, 1)$ $f(x) = 3748096$	$x^* = (1, 1)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv \left[10(x_1 - x_2)^2 + (x_1 - 1)^2\right]^4$			

Problema	Dimensão	Ponto Inicial	Solução
S240	$n = 3$	$x = (100, -1, 2.5)$ $f(x) = 29726.75$	$x^* = (0, 0, 0)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^3} \quad f(x) \equiv (x_1 - x_2 + x_3)^2 + (-x_1 + x_2 + x_3)^2 + (x_1 + x_2 - x_3)^2$ Função de Zangwill			

Problema	Dimensão	Ponto Inicial	Solução
S246	$n = 3$	$x = (-1, 2, 0)$ $f(x) = 8.4$	$x^* = (1, 1, 1)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^3} \quad f(x) \equiv 100\left[x_3 - \left(\frac{x_1 + x_2}{2}\right)^2\right]^2 + (1 - x_1)^2 + (1 - x_2)^2$			

Problema	Dimensão	Ponto Inicial	Solução
S256	$n = 4$	$x = (3, -1, 0, 1)$ $f(x) = 215$	$x^* = (0, 0, 0, 0)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^4} \quad f(x) \equiv (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$ Função de Powell			

Problema	Dimensão	Ponto Inicial	Solução
S257	$n = 4$	$x = (-3, -1, -3, -1)$ $f(x) = 19271.2$	$x^* = (1, 1, 1, 1)$ $f(x^*) = 0$
$\min_{x \in \mathbb{R}^4} \quad f(x) \equiv 100\left(x_1^2 - x_2\right)^2 + (x_1 - 1)^2 + 90\left(x_3^2 - x_4\right)^2 + (x_3 - 1)^2 + +10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_1 - 1)(x_4 - 1)$			

A.1.3 Problemas de Maratos

Problema	Dimensão	Ponto Inicial	Solução
M500	$n = 2$	$x = (1.1, 0.1)$ $f(x) = 5.94$	$x^* = (-1, 0)$ $f(x^*) = -1$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv x_1 + 100\left(x_1^2 + x_2^2 - 1\right)^2$			

Problema	Dimensão	Ponto Inicial	Solução
M501	$n = 2$	$x = (1.1, 0.1)$ $f(x) = 485.1$	$x^* = (-1, 0)$ $f(x^*) = -1$
$\min_{x \in \mathbb{R}^2} \quad f(x) \equiv x_1 + 10000\left(x_1^2 + x_2^2 - 1\right)^2$			

A.2 Problemas com Restrições

A.2.1 Problemas CUTE

Problema	Dimensão	Ponto Inicial	Solução
C801	$n = 2$	$x = (0.1, -0.1)$ $f(x) = 160.87$	$x^* = (?, ?)$ $f(x^*) = 7.563$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv 6x_1^2 + x_2^2 - 60x_1 - 8x_2 + 166 \\ \text{sujeito a} \quad & 0 \leq x_1 \leq 10 \\ & 0 \leq x_2 \leq 10 \\ & -x_1 - x_2 + x_1 x_2 \leq 0 \\ & -x_1 - x_2 + 3 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
C802	$n = 2$	$x = (0.1, -0.1)$ $f(x) = 295.1$	$x^* = (?, ?)$ $f(x^*) = -97.30952$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv 7x_1^2 + 3x_2^2 - 84x_1 - 34x_2 + 300 \\ \text{sujeito a} \quad & 0 \leq x_1 \leq 10 \\ & 0 \leq x_2 \leq 0 \\ & -x_1 x_2 + 1 \leq 0 \\ & -9 + x_1^2 + x_2^2 \leq 0 \end{aligned}$$

A.2.2 Problema PA

Problema	Dimensão	Ponto Inicial	Solução
PA	$n = 2$	$x = (0, 0)$ $f(x) = 0$	$x^* = (1.0, 0.0)$ $f(x^*) = -1$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv -x_1 - 2x_2 \\ \text{sujeito a} \quad & 0 \leq x_1 \leq 1 \\ & x_2 \leq 0 \end{aligned}$$

A.2.3 Problemas de Schittkowski

Problema	Dimensão	Ponto Inicial	Solução
S224	$n = 2$	$x = (0.1, 0.1)$ $f(x) = -8.77$	$x^* = (4, 4)$ $f(x^*) = -304$

$$
\begin{aligned}
\min_{x \in \mathbb{R}^2} \quad & f(x) \equiv 2x_1^2 + x_2^2 - 48x_1 - 40x_2 \\
\text{sujeito a} \quad & 0 \leq x_1 \leq 2 \\
& 0 \leq x_2 \leq 6 \\
& -x_1 - 3x_2 \leq 0 \\
& -18 + x_1 + 3x_2 \leq 0 \\
& -x_1 - x_2 \leq 0 \\
& -8 + x_1 + x_2 \leq 0
\end{aligned}
$$

Problema	Dimensão	Ponto Inicial	Solução
S225	$n = 2$	$x = (3, 1.8)$ $f(x_0) = 12.24$	$x^* = (1, 1)$ $f(x^*) = 2$

$$
\begin{aligned}
\min_{x \in \mathbb{R}^2} \quad & f(x) \equiv x_1^2 + x_2^2 \\
\text{sujeito a} \quad & -x_1 - x_2 + 1 \leq 0 \\
& -x_1^2 - x_2^2 + 1 \leq 0 \\
& -9x_1^2 - x_2^2 + 9 \leq 0 \\
& -x_1^2 + x_2 \leq 0 \\
& -x_2^2 + x_1 \leq 0
\end{aligned}
$$

Problema	Dimensão	Ponto Inicial	Solução
S226	$n = 2$	$x = (0.8, 0.05)$ $f(x) = -0.04$	$x^* = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ $f(x^*) = -0.5$

$$
\begin{aligned}
\min_{x \in \mathbb{R}^2} \quad & f(x) \equiv -x_1 x_2 \\
\text{sujeito a} \quad & -x_1 \leq 0 \\
& -x_2 \leq 0 \\
& -x_1^2 - x_2^2 \leq 0 \\
& -1 + x_1^2 + x_2^2 \leq 0
\end{aligned}
$$

Problema	Dimensão	Ponto Inicial	Solução
S227	$n = 2$	$x = (0.8, 0.05)$ $f(x) = -0.04$	$x^* = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$ $f(x^*) = -0.5$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv (x_1 - 2)^2 + (x_2 - 1)^2 \\ \text{sujeito a} \quad & x_1^2 - x_2 \leq 0 \\ & -x_1 + x_2^2 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S228	$n = 2$	$x = (0, 0)$ $f(x) = 0$	$x^* = (0, -3)$ $f(x^*) = -3$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv x_1^2 + x_2 \\ \text{sujeito a} \quad & x_1 + x_2 - 1 \leq 0 \\ & \left(x_1^2 + x_2^2\right) \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S231	$n = 2$	$x = (-1.2, 1)$ $f(x) = 24.2$	$x^* = (1, 1)$ $f(x^*) = 0$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 \\ \text{sujeito a} \quad & -\tfrac{1}{3}x_1 - x_2 - 0.1 \leq 0 \\ & \tfrac{1}{3}x_1 - x_2 - 0.1 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S233	$n = 2$	$x = (1.2, 1)$ $f(x) = 19.4$	$x^* = (1, 1)$ $f(x^*) = 0$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 \\ \text{sujeito a} \quad & -x_1^2 - x_2^2 + 0.25 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S234	$n = 2$	$x = (0.5, 0.5)$ $f(x) = -0.5$	$x^* = (0.2, 0.2)$ $f(x^*) = -0.8$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv (x_2 - x_1)^4 - (1 - x_1) \\ \text{sujeito a} \quad & 0.2 \leq x_1 \leq 2 \\ & 0.2 \leq x_2 \leq 2 \\ & x_1^2 + x_2^2 - 1 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S249	$n = 3$	$x = (1, 1, 1)$ $f(x) = 3$	$x^* = (1, 0, 0)$ $f(x^*) = 1$

$$\begin{aligned} \min_{x \in \mathbb{R}^3} \quad & f(x) \equiv x_1^2 + x_2^2 + x_3^2 \\ \text{sujeito a} \quad & 1 - x_1 \leq 0 \\ & -x1^2 - x_2^2 + 1 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S264	$n = 4$	$x = (0, 0, 0, 0)$ $f(x) = 0$	$x^* = (0, 1, 2, -1)$ $f(x^*) = -44$

$$\begin{aligned} \min_{x \in \mathbb{R}^4} \quad & f(x) \equiv x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4 \\ \text{sujeito a} \quad & x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 - x_3 - x_4 - 8 \leq 0 \\ & x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 9 \leq 0 \\ & 2x_1^2 + x_2^2 + x_3^2 + 2x_4^2 - x_2 - x_4 - 5 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S270	$n = 5$	$x = (1.1, 2.1, 3.1, 4.1, 1)$ $f(x) = 1.0001$	$x^* = (1, 2, 3, 4, 2)$ $f(x^*) = -1$

$$\begin{aligned} \min_{x \in \mathbb{R}^5} \quad & f(x) \equiv x_1x_2x_3x_4 - 3x_1x_2x_4 - 4x_1x_2x_3 + \\ & +12x_1x_2 - x_2x_3x_4 + 3x_2x_4 + 4x_2x_3 - 12x_2 + \\ & +2x_1x_3x_4 + 6x_1x_4 + 8x_1x_3 - 24x_1 + 2x_3x_4 + \\ & -6x_4 - 8x_3 + 24 + 1.5x_5^4 - 5.75x_5^3 + 5.25x_5^2 \\ \text{sujeito a} \quad & 1 - x_1 \leq 0 \\ & 2 - x_2 \leq 0 \\ & 3 - x_3 \leq 0 \\ & 4 - x_4 \leq 0 \\ & -34 + x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S323	$n = 2$	$x = (0, 1)$ $f(x) = 5$	$x^* = (0.5536, 1.306)$ $f(x^*) = 3.79894$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv x_1^2 + x_2^2 - 4x_1 + 4 \\ \text{sujeito a} \quad & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & -x_1 + x_2 - 2 \leq 0 \\ & -x_1^2 - x_2 + 1 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S324	$n = 2$	$x = (15, 3)$ $f(x) = 11.25$	$x^* = (15.81, 1.581)$ $f(x^*) = 5$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv 0.01x_1^2 + x_2^2 \\ \text{sujeito a} \quad & 2 - x_1 \leq 0 \\ & -x_1 x_2 + 25 \leq 0 \\ & -x_1^2 - x_2 + 25 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S325	$n = 2$	$x = (-3, 0)$ $f(x) = 9$	$x^* = (-2.372, -1.836)$ $f(x^*) = 3.79184$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv x_1^2 + x_2 \\ \text{sujeito a} \quad & x_1 + x_2 - 1 \leq 0 \\ & x_1 + x_2^2 - 1 \leq 0 \\ & -x_1^2 - x_2^2 + 9 \leq 0 \end{aligned}$$

Problema	Dimensão	Ponto Inicial	Solução
S326	$n = 2$	$x = (4, 3)$ $f(x) = -69$	$x^* = (5.240, 3.746)$ $f(x^*) = -79.8078$

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & f(x) \equiv x_1^2 + x_2^2 - 16x_1 - 10x_2 \\ \text{sujeito a} \quad & -x_1 \leq 0 \\ & -x_2 \leq 0 \\ & -11 + x_1^2 - 6x_1 + 4x_2 \leq 0 \\ & -x_1 x_2 + 3x_2 + e^{x_1 - 3} - 1 \leq 0 \end{aligned}$$

Bibliografia

[1] M. Abramson, C. Audet, and J. Dennis Jr. Filter pattern search algorithms for mixed variable constrained optimization problems. *Pacific Journal of Optimization*, (3):477–500, (2004).

[2] M. Abramson, C. Audet, J. Dennis Jr., and S. Le Digabel. Orthomads: A deterministic mads instance with orthogonal directions. Technical Report G-2008-15, Les Cahiers du GERAD, École Polytechnique de Montréal, (2008).

[3] P. Alberto, F. Nogueira, H. Rocha, and L. Vicente. Pattern search methods for user-provided points: Application to molecular geometry problems. *SIAM Journal on Optimization*, 14(4):1216–1236, (2004).

[4] R. Andreani, E. Birgin, J. Martínez, and M. Schuverdt. On augmented lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, (18): 1286–1309, (2007).

[5] R. Andreani, G. Birgin, J. Martínez, and M. Schuverdt. Augmented lagrangian methods under the constant positive linear dependence constraint qualification. *Mathematical Programming*, (111):5–32, (2008).

[6] A. Antunes and M. Monteiro. A filter algorithm and other nlp solvers: Performance comparative analysis. *Lecture Notes in Economics and Mathematical Systems - Recent Advances in Optimization*, 563:425–434, (2006).

[7] C. Audet and J. Dennis Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, (2002).

[8] C. Audet and J. Dennis. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization*, 5(14):980–1010, (2004).

[9] C. Audet. Convergence results for pattern search algorithms are tight. *Optimization and Engineering*, 2(5):101–122, (2004).

[10] C. Audet and J. Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, (17):188–217, (2006).

[11] C. Audet and J. Dennis Jr. A mads algorithm with a progressive barrier for derivative-free nonlinear programming. Technical Report G-2007-37, Les Cahiers du GERAD, École Polytechnique de Montréal, (2007).

[12] C. Audet, J. Dennis Jr., and S. Le Digabel. Globalization strategies for mesh adaptative direct search. Technical Report G-2008-74, Les Cahiers du GERAD, École Polytechnique de Montréal, (2008).

[13] C. Audet, V. Béchard, and S. Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *J. Global Opt.*, (41): 299–318, (2008).

[14] M. Bartholomew-Biggs. *Nonlinear Optimization with Engineering Applications*. Springer, Series: Springer Optimization and Its Applications, Vol. 19, (2008).

[15] A. Bagirov. Derivative-free methods for unconstrained nonsmooth optimization and its numerical analysis. *Investigação Operacional*, 1(19):75–93, (1999).

[16] M. Bazaraa, H. Sherali, and C. Shetty. *Nonlinear Programming: Theory and Applications*. J. Wiley and Sons, Chichester, England, 3rd edition, (2006).

[17] H. Benson, A. Sen, D. Shanno, and R. Vanderbei. Interior-point algorithms, penalty methods and equilibrium problems. *Comput. Optim. Appl.*, 34(2):155–182, (2006).

[18] D. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, New York, (1982).

[19] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, (1999).

[20] H. Beyer and H. Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing: an international journal*, 1(1):3–52, (2002).

[21] E. Birgin, R. Castillo, and J. Martínez. Numerical comparison of augmented lagrangian algorithms for nonconvex problems. *Comput. Optim. Appl.*, (31):31–55, (2005).

[22] I. Bongartz, A. Conn, N. Gould, and P. Toint. Cute: Constrained and unconstrained testing environment. *ACM Transactions and Mathematical Software*, (21):123–160, (1995).

[23] G. Booch, R. Maksimchuk, M. Engel, B. Young, J. Conallen, and K. Houston. *Object-Oriented Analysis and Design with Applications.* Addison-Wesley Professional, 3rd edition, (2007).

[24] J. Bull, L. Smith, C. Ball, L. Pottage, and R. Freeman. Benchmarking Java against C and Fortran for scientific applications. *Concurrency and Computation: Practice and Experience*, 15(3-5):417–430, March-April (2003).

[25] A. Burmen, J. Puhan, and T. Tuma. Grid restrained Nelder-Mead algorithm. *Computacional Optimization and Applications*, 34(3):359–375, (2006).

[26] R. Byrd, N. Gould, J. Nocedal, and R. Waltz. On the convergence of successive linear-quadratic programming algorithms. *SIAM Journal on Optimization*, 16(2): 471–489, (2005).

[27] R. Byrd, J. Nocedal, and R. Waltz. Steering exact penalty methods for nonlinear programming. *Optimization Methods & Software*, 23(2):197–213, (2008).

[28] R. Chelouah and P. Siarry. A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global optimization of multiminima functions. *Europeam Journal of Operational Research*, 161(3):636–654, (2005).

[29] L. Chen and D. Goldfarb. Interior-point l(2)-penalty methods for nonlinear programming with strong global convergence properties. *Mathematical Programming*, 108(1):1–36, (2006).

[30] A. Conn, N. Gould, and P. Toint. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *Journal on Numerical Analysis*, 2(28):545–572, (1991).

[31] N. Gould A. Conn, P. Toint, and Ox Qx. A globally convergent lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Mathematics of Computation*, 66(217):261–288, (1997).

[32] A. Conn, K. Scheinberg, and L. Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization, SIAM, Philadelphia, USA, (2009).

[33] I. Coope and C. Price. Frame based methods for unconstrained optimization. *J. Optim. Theory Appl.*, 107(2):261–274, (2000).

[34] I. Coope and C. Price. On the convergence of grid-based methods for unconstrained optimization. *SIAM Journal on Optimization*, 11(4):859–869, (2001).

[35] A. Correia, J. Matias, and C. Serôdio. Direct search algorithms for nonlinear optimization problems and the java language. In *Proceedings of the Second International Conference on Multidisciplinary Design Optimization and Applications*, pages 362–370, Gijon, Spain, (2008).

[36] A. Correia, J. Matias, and C. Serôdio. Métodos de penalidade exacta para resolução de problemas de optimização não linear. *Revista Investigação Operacional*, 28(1): 17–30, 2008. ISSN 0874-5161.

[37] A. Correia, J. Matias, P. Mestre, and C. Serôdio. Constrained nonlinear optimization without derivatives. In *Conference Proceedings of MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, volume 2, pages 2499 – 2503, Viena, Áustria, (2009). ARGESIM Report no. 35. ISBN 10: 3-901608-35-4.

[38] A. Correia, J. Matias, P. Mestre, and C. Serôdio. Derivative-free optimization and filter methods to solve nonlinear constrained problems. *International Journal of Computer Mathematics*, 86(10):1841–1851, (2009).

[39] A. Correia, J. Matias, P. Mestre, and C. Serôdio. Penalty versus filter methods. In *Book of Abstracts of the 23nd European Conference on Operational Research*, page 189, Bonn, Alemanha, (2009).

[40] A. Correia, J. Matias, P. Mestre, and C. Serôdio. Classification of some penalty methods. In *Integral Methods in Science and Engineering*, volume 2, pages 131–141. Birkhäuser/Springer, (2010 Edition). ISBN 978-0-8176-4896-1.

[41] A. Correia, J. Matias, P. Mestre, and C. Serôdio. Derivative-free nonlinear optimization filter simplex. *International Journal of Applied Mathematics and Computer Science (AMCS)*, 20(4), (In Press, 2010).

[42] A. Correia, J. Matias, P. Mestre, and C. Serôdio. Direct search filter methods. In *Book of Abstracts of the 24nd European Conference on Operational Research*, Lisbon, Portugal, (2010).

[43] A. Correia, J. Matias, P. Mestre, C. Serôdio, and C. Fraga. Direct search methods for nonlinear optimization, 2 de Setembro (2010). Seminar - Erasmus/Socrates staff mobility between Porto and Perugia, Perugia, Itália. (2010).

[44] A. Correia, J. Matias, P. Mestre, C. Serôdio, and C. Fraga. Direct search methods for nonlinear optimization. In *Proceedings of MME10 Mathematical Methods in Engineering International Symposium - Digital Support*, Coimbra, Portugal, (2010). IPC. paper1_MME.

[45] A. Correia, J. Matias, P. Mestre, and C. Serôdio. Direct-search penalty/barrier methods. In *Lecture Notes in Engineering and Computer Science - World Congress on Engineering 2010*, volume 3, pages 1729 – 1734, London, UK, (2010). IAENG. ISBN 978-988-18210-8-9.

[46] A. Correia, J. Matias, P. Mestre, C. Serôdio, and C. Fraga. Métodos de pesquisa directa em optimização não linear, 28 de Abril (2010). Seminários do Centro de Matemática - Universidade de Trás-os-Montes e Alto Douro, Grupo de Estatística e Investigação Operacional. Vila Real. (2010).

[47] A. Custódio and L. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization*, 18(2):537–555, (2007).

[48] X. Dai. Finite element approximation of the pure neumann problem using the iterative penalty method. *Applied Mathematics and Computation*, 186(2):1367–1373, (2007).

[49] K. Deb. An introduction to genetic algorithms. 4:293–315, (1999).

[50] K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, (2001).

[51] J. Dennis and D. Woods. Optimization on microcomputers: The nelder-mead simplex algorithm. *New Computing Environments: Microcomputers in Large-Scale Computing,*, Wouk, A., ed.:116–122, (1987).

[52] J. Dennis Jr. and V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1:448–474, (1991).

[53] G. Dantzig. *Linear Programming and Extensions.* Princeton Uinversity Press, Princeton, USA, (1998).

[54] M. Doyle. *A barrier algorithm for large nonlinear optimization problems.* PhD thesis, Stanford University, Stanford, California, (2003).

[55] I. Espírito Santo. *Desenho óptimo de estações de águas residuais através da modelação de funções custo.* PhD thesis, Escola de Engenharia, Universidade do Minho, Braga, (2007).

[56] I. Espírito Santo. *Optimização sem derivadas - Método de Hooke and Jeeves, Curso de Optimização Aplicada às Ciências e Engenharia.* Universidade do Minho, Braga, (2009).

[57] S. Fan and E. Zahara. A hybrid simplex search and particle swarm optimization for unconstrained optimization. *European Journal of Operational Research*, In Press, Corrected Proof, (2006).

[58] E. Fernandes. *Computação Numérica.* Universidade do Minho, Braga, (1998).

[59] E. Fernandes. *Modelação e Optimização não Linear, Curso de Optimização Aplicada às Ciências e Engenharia.* Braga, (2009).

[60] E. Fernandes. *Optimização sem Derivadas, Curso de Optimização Aplicada às Ciências e Engenharia.* Braga, (2009).

[61] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons Inc., New York, (1987).

[62] R. Fletcher. Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–269.

[63] R. Fletcher, S. Leyffer, and P. Toint. On the global convergence of an slp-filter algorithm. Technical Report NA/183, Dundee University, Dept. of Mathematics, (1998).

[64] R. Fletcher and S. Leyffer. A bundle filter method for nonsmooth nonlinear optimization. Technical Report NA/195, Dundee University, Dept. of Mathematics, (1999).

[65] R. Fletcher, N. Gould, S. Leyffer, P. Toint, and A. Wachter. Global convergence of trust region and sqp-filter algorithms for general nonlinear programming. *SIAM Journal on Optimization*, 3(13):635–659, (2002).

[66] R. Fletcher, S. Leyffer, and P. Toint. A brief history of filter method. Technical Report ANL/MCS-P1372-0906, Argonne National Laboratory, Mathematics and Computer Science Division, (2006).

[67] R. Fourer and D. Orban. The drampl meta solver for optimization. Technical report, GERAD, (2007).

[68] R. Freund. *Penalty and Barrier Methods for Constrained Optimization.* Massachusetts Institute of Technology, Massachusetts, February (2004).

[69] R. Freund. *Optimality Conditions for Constrained Optimization Problems.* Massachusetts Institute of Technology, Massachusetts, February (2004).

[70] M. Friedlander, N. Gould, S. Leyffer, and T. Munson. A filter active-set trust-region method. Technical report, ANL/MCS-P1456-0907, (2007).

[71] P. Gill, W. Murray, and M. Wright. *Practical Optimization.* Academic Press inc., London, (1981).

[72] H. Ghiasi, D. Pasini, and L. Lessard. Constrained globalized nelder-mead method for simultaneous structural and manufacturing optimization of a composite bracket. *Journal of Composite Materials*, 42(7):717–736, (2008).

[73] C. Godfrey and B. Babu. *New Optimization Techniques in Engineering.* Springer, (2004).

[74] C. Gonzaga, E. Karas, and M. Vanti. A globally convergent filter method for nonlinear programming. *SIAM Journal on Optimization*, 14(3):646–669, (2003).

[75] N. Gould, D. Orban, and P. Toint. An interior-point l_1-penalty method for nonlinear optimization. Technical report, Rutherford Appleton Laboratory Chilton, (2003).

[76] N. Gould, S. Leyffer, and P. Toint. A multidimensional filter algorithm for nonlinear equations and nonlinear least-squares. *SIAM Journal on Optimization*, 15(1):17–38, (2005).

[77] N. Gould, C. Sainvitu, and P. Toint. A filter-trust-region method for unconstrained optimization. *SIAM Journal on Optimization*, 16(2):341–357, (2006).

[78] L. Han and M. Neumann. Effect of dimensionality on the Nelder-Mead simplex method. *Optimization Methods & Software*, 21(1):1–16, (2006).

[79] W. Hart. Locally-adaptive and memetic evolutionary pattern search algorithms. *Evolutionary Computation*, 11:29–52, (2003).

[80] A. Hedar and M. Fukushima. Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. *Optim. Methods Softw.*, 19: 291–308, (2004).

[81] M. Hestenes. Multiplier and gradient methods. *J. Optim. Theory Appl.*, 4:303–320, (1969).

[82] A. Homaifar, S. Lai, and X. Qi. Constrained optimization via generic algorithms. *Simulation*, 62(4):242–254, (1994).

[83] R. Hooke and T. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8(2):212–229, (1961).

[84] X. Hu and R. Eberhart. Solving constrained nonlinear optimization problems with particle swarm optimization. In *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002 (SCI 2002)*, pages 203–206, (2002).

[85] E. Karas. *Exemplos de trajetória central mal comportada em otimização convexa e um algoritmo de filtros para programação não linear.* PhD thesis, Universidade Federal de Santa Catarina e Universidade de Paris I - Panthéon-Sorbonne, Florianópolis, Brasil, (2002).

[86] E. Karas, A. Ribeiro, C. Sagastizábal, and M. Solodov. A bundle-filter method for nonsmooth convex constrained optimization. *Mathematical Programming*, 1(116): 297–320, (2006).

[87] J. Karlof. *Integer programming: theory and practice.* CRC Press - Operations Research Series, University of North Carolina, Wilmington, USA, (2005).

[88] C. Kelley. Detection and remediation of stagnation in the nelder-mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization*, 10:43–55, (1999).

[89] C. Kelley. *Iterative Methods for Optimization.* Number 18 in Frontiers in Applied Mathematics, SIAM, Philadelphia, USA, (1999).

[90] D. Klatte and B. Kummer. Constrained minima and lipschitzian penalties in metric spaces. *SIAM Journal on Optimization*, 13(2):619–633, (2002).

[91] T. Kolda, R. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482, (2003).

[92] A. Koscianski and A. Luersen. Globalization and Parallelization of Nelder-Mead and Powell Optimization Methods. In Elleithy, K, editor, *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, pages 93–98, Netherlands, (2008). Springer.

[93] J. Lagarias, J. Reeds, M. Wright, and P. Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, (1998).

[94] R. Lewis and V. Torczon. Rank ordering and positive bases in pattern search algorithms. Technical report, (1996).

[95] R. Lewis and V. Torczon. Pattern search methods for linearly constrained minimization. Technical report, (1998).

[96] R. Lewis, V. Torczon, and M. Trosset. Why pattern search works. Technical report, (1998).

[97] R. Lewis and V. Torczon. Pattern search methods for bound constrained minimization. *SIAM Journal on Optimization*, 10(3):917–941, (1999).

[98] R. Lewis and V. Torczon. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, 10(3):917–941, (1999).

[99] R. Lewis, V. Torczon, and M. Trosset. Direct search methods: Then and now. *J. Comput. Appl. Math.*, (124):191–207, (2000).

[100] R. Lewis and V. Torczon. A globally convergent augmented lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4):1075–1089, (2002).

[101] S. Leyffer, G. Calva, and J. Nocedal. Interior methods for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 17(1):52–77, (2006).

[102] S. Lucidi and M. Sciandrone. Numerical results for unconstrained optimization without derivatives. *Nonlinear Optimization and Aplications*, pages 261–269, (1995).

[103] S. Lucidi and M. Sciandrone. On the convergence of derivative-free methods for unconstrained optimization. *SIAM Journal on Optimization*, 13(1):97–116, (2002).

[104] M. Luersen and R. Le Riche. Globalized Nelder-Mead method for engineering optimization. *Computers & Structures*, 82(23-26):2251–2260, (2004).

[105] H. Luo, X. Sun, and D. Li. On the convergence of augmented lagrangian methods for constrained global optimization. *SIAM Journal on Optimization*, (18):1209–1230, (2007).

[106] H. Luo. *Studies on the augmented Lagrangian methods and convexification methods for constrained global optimization.* PhD thesis, Department of Mathematics, Shanghai University, Shanghai, (2007).

[107] H. Luo, X. Sun, and H. Wu. Convergence properties of augmented Lagrangian methods for constrained global optimization. *Optimization Methods & Software*, 23(5):763–778, (2008).

[108] H. Maarten van der Vlerk. Stochastic programming bibliography, (1996-2007). URL http://mally.eco.rug.nl/spbib.html.

[109] F. Martins. *Java 5 e Programação por Objectos.* FCA - Editora de Informática, Lisboa, (2006).

[110] J. Matias. *Técnicas de Penalidade e Barreira Baseadas em Métodos de Pesquisa Directa e a Ferramenta PNL-Pesdir.* PhD thesis, UTAD, Vila Real, (2003).

[111] J. Matias, A. Correia, P. Mestre, and C. Serôdio. Derivative-free nonlinear optimization filter simplex. In *Book of Abstracts of the XV International Conference on Mathematics, Informatics and Related Fields*, page 47, Polónia, (2009).

[112] J. Matias, A. Correia, P. Mestre, C. Serôdio, and C. Fraga. Web-based application programming interface to solve nonlinear optimization problems. In *Lecture Notes in Engineering and Computer Science - World Congress on Engineering 2010*, volume 3, pages 1961–1966, London, UK, (2010). IAENG. "Certificate of Merit for The 2010 International Conference of Applied and Engineering Mathematics"WCE - ICAEM'10.

[113] K. McKinnon. Convergence of the nelder–mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1):148–158, (1998).

[114] A. Mendes and M. Marcelino. *Fundamentos de Programação em Java 2.* FCA - Editora de Informática, Lisboa, 3ªedição edition, (2005).

[115] P. Mestre. *Configuração, reconfiguração e gestão de sistemas distribuídos dinâmicos Jini-enabled.* PhD thesis, UTAD, Vila Real, (2006).

[116] P. Mestre, C. Serôdio, P. Oliveira, and J-P. Moura. *Conceitos fundamentais de programação em Java. Série Didáctica Ciências Aplicadas No 411.* Vila Real, (2010).

[117] M. Mongeau and A. Sartenaer. Automatic decrease of the penalty parameter in exact penalty function methods. *European Journal of Operational Research*, 3(83): 686–699, (1995).

[118] L. Nazareth and P. Tseng. Gilding the lily: A variant of the Nelder-Mead algorithm based on golden-section search. *Computationa Optimization and Applications*, 22 (1):133–144, (2002).

[119] J. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, January (1965).

[120] J. Neto. *Sistemas e Sinais - LEIC, Capítulo 2 - Máquinas de Estado.* Lisboa, (2007/08).

[121] P. Nie. Sequential penalty quadratic programming filter methods for nonlinear programming. *Nonlinear Analysis-Real World Applications*, 8(1):118–129, (2007).

[122] G. Nikishkov, Yu. G. Nikishkov, and V. Savchenko. Comparison of C and Java performance in finite element computations. *Computers & Structures*, 81(24-25): 2401–2408, September 2003.

[123] J. Nocedal and S. Wright. *Numerical Optimization.* Series in Operations Research,Springer Verlag, Heidelberg, Berlin, New York, (1999).

[124] G. Onwubolu and B. Babu. *New Optimization Techniques in Engineering.* Springer, (2004).

[125] B. Panigrahi and V. Pandi. Bacterial foraging optimisation: Nelder-Mead hybrid algorithm for economic load dispatch. *IET Generation Tranmission & Distribution*, 2(4):556–565, (2008).

[126] K. Parsopoulos and M. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1:235–306, (2002).

[127] A. Pereira. *Caracterização da função de penalidade exponencial num método de redução para programação semi-infinita.* PhD thesis, Universidade do Minho, Braga, (2006).

[128] T. Pietrzykowski. An exact potential method for constrained maxima. *SIAM Journal on Numerical Analysis*, 6(2):299–304, (1969).

[129] M. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, February (1964).

[130] C. Price, I. Coope, and D. Byatt. A convergent variant of the nelder-mead algorithm. *J. Optim. Theory Appl.*, 113(1):5–19, (2002).

[131] A. Prsa and T. Zwitter. Introducing adapted Nelder & Mead's downhill simplex method to a fully automated analysis of eclipsing binaries. In *Proceedings of the Symposium the Three-Dimensional Universe with Gaia*, volume 576 of *ESA Special Publications*, pages 611–614, Netherlands, (2005).

[132] A. Ribeiro, E. Karas, and C. Gonzaga. Global convergence of filter methods for nonlinear programming. *SIAM J. on Optimization*, 19(3):1231–1249, 2008. ISSN 1052-6234. doi: http://dx.doi.org/10.1137/060672285.

[133] A. Rykov. Simplex algorithms for unconstrained minimization. *Problems of Control and Information Theory*, 12:195–208, (1983).

[134] H. Rodrigues and M. Monteiro. Solving mathematical programs with complementarity constraints with nonlinear solvers. *Lecture Notes in Economics and Mathematical Systems - Recent Advances in Optimization*, 563(Part IV):415–424, (2006).

[135] H. Rodrigues, M. Monteiro, and A. Vaz. A mpcc approach on a stackelberg game in an electric power market: changing the leadership. *International Journal of Computer Mathematics*, 86:1921 – 1931, (2009).

[136] H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, March (1960).

[137] C. Sainvitu. *Filter-Trust-Region Methods for Nonlinear Optimization.* PhD thesis, Presses universitaires de Namur and Caroline Sainvitu, Namur, Bruxelas, (2007).

[138] K. Schittkowski. *More Test Examples for Nonlinear Programming Codes, Economics and Mathematical Systems.* Springer-Verlag, Berlin, (1987).

[139] C. Silva and M. Monteiro. A filter algorithm: comparison with nlp solvers. *International Journal of Computer Mathematics*, 85(3-4):667–689, (2008). ISSN 0020-7160.

[140] C. Silva and M. Monteiro. A filter inexact-restoration method for nonlinear programming. *TOP*, 16:126–146, (2008). ISSN 1134-5764.

[141] R. Silva, M. Ulbrich, S. Ulbrich, and L. Vicente. A globally convergent primal-dual interior-point filter method for nonlinear programming: new filter optimality measures and computational results. Technical Report Preprint 08-49, Dept. Mathematics, Univ. Coimbra, (2008).

[142] W. Spendley, G. Hext, and F. Himsworth. Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics*, 4:441–461, (1962).

[143] V. Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Department of Mathematical Sciences, Rice University, Houston, Texas, (1989).

[144] V. Torczon. On the convergence of the multidirectional search algorithm. *SIAM Journal on Optimization*, 1:123–145, (1991).

[145] V. Torczon. Pattern search methods for nonlinear optimization. *SIAG/OPT Views and News*, 6:7–11, (1995).

[146] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7:1–25, (1997).

[147] M. Trosset. I know it when i see it: Toward a definition of direct search methods. *SIAM Activity Group on Optimization*, 9:7–10, (1997).

[148] P. Tseng. Fortified-descent simplicial search method: A general approach. *SIAM Journal on Optimization*, 10(1):269–288, (2000).

[149] M. Ulbrich, S. Ulbrich, and L. Vicente. A globally convergent primal-dual interior-point filter method for nonlinear programming. *Mathematical Programming, Ser. A*, 2(100):379–410, (2004).

[150] R. van Engelen. Pushing the SOAP Envelope WithWeb Services for Scientific Computing. In *Proceedings of the International Conference on Web Services (ICWS)*, pages 346–352, 2003.

[151] A. Vaz and L. Vicente. A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, 39:197–219, (2007).

[152] A. Vaz. *Algoritmos Genéticos e Evolucionários e Optimização Sem Derivadas, Curso de Optimização Aplicada às Ciências e Engenharia*. Universidade do Minho, Braga, (2009).

[153] L. Vicente. Worst case complexity of direct search. Technical Report Preprint 10-17, Dept. of Mathematics, Univ. Coimbra, (2010).

[154] A. Wächter and L. Biegler. Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):32–48, (2005).

[155] A. Wächter and L. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, (2005).

[156] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, (2006).

[157] F. Walters, R. Lloyd, S. Morgan, and S. Deming. *Sequential Simplex Optimization: A Technique for Improving Quality and Productivity in Research, Development, and Manufacturing.* Chemometrics series. CRC Press, (1991).

[158] M. Wolfe. *Numerical Methods for Unconstrained Optimization.* Van Nostrand Reinhold Company, (1978).

[159] F. Wang and D. Liu. *Advances in Computational Intelligence: Theory And Applications (Series in Intelligent Control and Intelligent Automation).* World Scientific Publishing Co., Inc., River Edge, NJ, USA, (2006).

[160] H. Wu, H. Luo, and S. Li. The global convergence of Augmented Lagrangian methods based on NCP function in constrained nonconvex optimization. *Applied Mathematics and Computation*, 207(1, Sp. Iss. SI):124–134, (2009).

[161] C. Yang, J. Chen, and X. Tu. Algorithm of marriage in honey bees optimization based on the nelder-mead method - art. no. 1350. In *Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2007)*, Advances in Intelligent System Research, page 1350, (2007).

[162] E. Zahara and Y. Kao. Hybrid nelder-mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, 36(2, Part 2):3880 – 3886, (2009).

[163] A. Zaslavski. A sufficient condition for exact penalty in constrained optimization. *SIAM Journal on Optimization*, 16(1):250–262, (2005).

[164] S. Zhang. Constrained optimization by ϵ constrained hybrid algorithm of particle swarm optimization and genetic algoritnm. In *Proceedings of AI 2005: Advances in Artificial Intelligence*, pages 389–400, Springer, (2005). Civil-Comp press.