

PROGRAMAÇÃO NÃO LINEAR SEM DERIVADAS[†]

Lucas Garcia Pedroso

Doutorado em Matemática Aplicada

José Mario Martínez

Orientador

Maria Aparecida Diniz Ehrhardt

Co-orientadora

Tese de doutorado pelo Instituto de Matemática, Estatística e Computação Científica,
UNICAMP.

[†]Este trabalho teve o apoio financeiro da FAPESP e CNPQ.

PROGRAMAÇÃO NÃO LINEAR SEM DERIVADAS

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por **Lucas Garcia Pedroso** e aprovada pela comissão julgadora.

Campinas, 28 de agosto de 2009



Prof. Dr. **José Mario Martínez**
Orientador



Prof. Dra. **Maria Ap. Diniz-Ehrhardt**
Co-orientadora

Banca Examinadora:

Prof. Dr. José Mario Martínez (IMECC/UNICAMP)

Prof. Dr. Roberto Andreani (IMECC/UNICAMP)

Prof. Dr. Orizon Pereira Ferreira (UFG)

Prof. Dra. Elizabeth Wegner Karas (UFPR)

Prof. Dr. Yuan Jin Yun (UFPR)

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do Título de Doutor em Matemática Aplicada.

**FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Miriam Cristina Alves CRB8a / 5094**

Pedroso, Lucas Garcia

P343p Programação não linear sem derivadas/Lucas Garcia Pedroso --
Campinas, [S.P. :s.n.], 2009.

Orientadores : José Mario Martínez; Maria Aparecida Diniz
Ehrhardt.

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de
Matemática, Estatística e Computação Científica.

1. Programação não-linear. 2. Otimização com restrições. 3.
Métodos sem derivadas. I. Martínez Pérez, José Mario. II. Ehrhardt,
Maria Aparecida Diniz. III. Universidade Estadual de Campinas.
Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Título em inglês: Derivative-free nonlinear programming

Palavras-chave em inglês (Keywords): 1. Nonlinear programming. 2. Constrained
optimization. 3. Derivative-free methods.

Área de concentração: Otimização matemática

Titulação: Doutor em Matemática Aplicada

Banca examinadora: Prof. Dr. José Mario Martínez (IMECC-UNICAMP)
Prof. Dr. Roberto Andreani (IMECC-UNICAMP)
Prof. Dr. Orizon Pereira Ferreira (UFG)
Profa. Dra. Elizabeth Wegner Karas (UFPR)
Prof. Dr. Yuan Jin Yun (UFPR)

Data da defesa: 28/08/2009

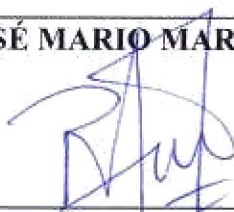
Programa de pós-graduação: Doutorado em Matemática Aplicada

Tese de Doutorado defendida em 28 de agosto de 2009 e aprovada


Pela Banca Examinadora composta pelos Profs. Drs.



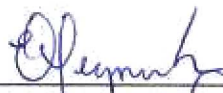
Prof(a). Dr(a). JOSÉ MARIO MARTÍNEZ PÉREZ



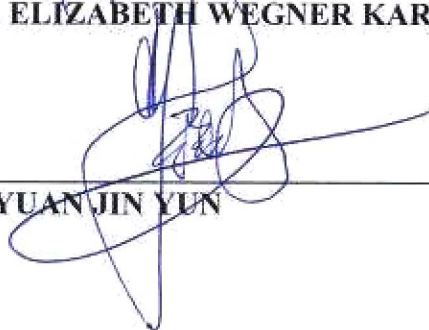
Prof(a). Dr(a). ROBERTO ANDREANI



Prof(a). Dr(a). ORIZON PEREIRA FERREIRA



Prof(a). Dr(a). ELIZABETH WEGNER KARAS



Prof(a). Dr(a). YUANJIN YUN

Agradecimentos

Agradeço a Deus pela oportunidade ímpar de obter o título de doutor em uma das melhores universidades do Brasil. Peço força e discernimento para que eu consiga converter o conhecimento até agora adquirido em benefícios para outras pessoas.

Agradeço a meus pais, meus irmãos, e toda minha família pelo amor incondicional, pelo apoio constante e sem exigências. Não há palavras que descrevam o que sinto por vocês. Só me resta dizer: eu os amo, obrigado por tudo!

Agradeço a meus amigos pelo afeto e companheirismo, pelas palavras de ânimo, por estarem do meu lado e terem me abraçado tanto nos momentos difíceis como nos alegres. Obrigado pelo ombro amigo, pelas conversas e pela torcida sincera.

Agradeço aos funcionários do IMECC, especialmente Tânia e Fátima, por terem me ajudado com prontidão sempre que precisei.

Agradeço a todos professores que já tive, dentro e fora do IMECC, que são responsáveis por tudo aquilo que hoje sei. Em especial, agradeço a todos os professores do grupo de otimização. Guardo lembranças agradáveis de cada um, dos congressos, das aulas, dos grupos de estudo.

Agradeço à Cheti pelos quase dez anos de orientação e amizade. Olhando para trás penso que não poderia ter tomado decisão melhor ao pedir para você me orientar em minha primeira iniciação científica. Obrigado pelas inúmeras reuniões, onde você com carinho me atendeu. Obrigado por ter me ensinado ética, bom senso, disciplina e generosidade.

Agradeço ao Mario por ter me dado a imensa honra de ser seu aluno, por ter me mostrado que o conhecimento e a genialidade podem sim estar associados à humildade e à generosidade. Obrigado pela paciência, pelos bons conselhos e pelas críticas sempre construtivas.

Agradeço, por fim, à Fapesp e ao CNPQ pelo imprescindível amparo financeiro.

Resumo

Neste trabalho propomos um algoritmo Lagrangiano Aumentado sem derivadas para o problema geral de otimização. Consideramos o método introduzido por Andreani, Birgin, Martínez e Schuverdt, eliminando os cálculos de derivadas inerentes ao algoritmo através de modificações adequadas no critério de parada. Foram mantidos os bons resultados teóricos do método, como convergência sob a condição de qualificação CPLD e a limitação do parâmetro de penalidade. Experimentos numéricos são apresentados, entre os quais destacamos um exemplo de problema sem derivadas baseado na simulação de áreas de figuras no plano.

Palavras-chave: programação não linear, otimização com restrições, métodos sem derivadas.

Abstract

We propose in this work a derivative-free Augmented Lagrangian algorithm for the general problem of optimization. We consider the method due to Andreani, Birgin, Martínez and Schuverdt, eliminating the derivative computations in the algorithm by making suitable modifications on the stopping criterion. The good theoretical results of the method were maintained, as convergence under the CPLD constraint qualification and the limitation of the penalty parameter. Numerical experiments are presented, and the most relevant of them is an example of derivative-free problem based on the simulation of areas of figures on the plane.

Key words: nonlinear programming, constrained optimization, derivative-free methods.

Sumário

Introdução	1
1 Métodos sem Derivadas	4
1.1 Algoritmos de Busca Padrão e Correlatos	5
1.2 Algoritmo Nelder-Mead	10
1.3 Algoritmo NEWUOA	12
2 Lagrangiano Aumentado Sem Derivadas	14
2.1 Algoritmo Lagrangiano Aumentado	14
2.2 Lagrangiano Aumentado sem Derivadas	20
2.2.1 Problemas com Caixas no Nível Inferior	20
2.2.2 Problemas com Restrições Gerais no Nível Inferior	27
2.2.3 Gradiente Projetado Contínuo sem Derivadas	31
3 Experimentos Numéricos	34
3.1 Exemplos de Hock-Schittkowski	36
3.2 Problema da Área	39
3.3 Problema MDO	48
Conclusões	50
Referências Bibliográficas	52

Introdução

O presente trabalho se dedica ao problema de otimização

$$\text{minimizar } f(x) \text{ sujeita a } x \in \Gamma,$$

onde $f : \mathbb{R}^n \rightarrow \mathbb{R}$ e $\Gamma = \{x \in \mathbb{R}^n \mid h(x) = 0, g(x) \leq 0\}$, para funções $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$ e $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$. Chamamos as condições $h_i(x) = 0$, $i = 1, \dots, m$ de restrições de igualdade, $g_i(x) \leq 0$, $i = 1, \dots, p$ de restrições de desigualdade, e se $x \in \Gamma$, dizemos que é um ponto factível. Um caso particular do problema acima que nos interessa de modo especial ocorre quando há canalizações entre as restrições de desigualdade, ou seja, o conjunto Γ abriga restrições do tipo $l_i \leq x_i \leq u_i$, $l_i \leq u_i$, $i = 1, \dots, n$. Tais restrições são conhecidas como restrições de caixa.

Um diferencial de nosso estudo em relação aos consagrados algoritmos propostos na literatura para o problema acima reside no fato de estarmos abrindo mão do cômputo de derivadas. Propomos um método para a resolução do problema geral de otimização que não calcula gradientes ou matrizes hessianas em nenhum de seus passos, o que faz que nosso trabalho se enquadre na categoria de métodos sem derivadas, classe que tem sido objeto de estudo de vários pesquisadores ao redor do mundo. Longe de querer substituir os métodos clássicos, os trabalhos em otimização sem derivadas visam a resolução de problemas onde as derivadas estão indisponíveis, o que ocorre, por exemplo, quando as funções envolvidas no problema são calculadas através de simulações, de executáveis cujos códigos não estão disponíveis (caixas-pretas), ou simplesmente quando o usuário preferiu se abster de implementar derivadas. De fato, o desempenho prático de métodos sem derivadas dificilmente supera o de um bom algoritmo baseado em derivadas, especialmente no que tange ao tempo computacional e número de avaliações de função.

Para que atinjamos nosso objetivo, modificamos o algoritmo Lagrangiano Aumentado proposto por Andreani, Birgin, Martínez e Schuverdt em [3] para torná-lo sem derivadas. Evidentemente, tal alteração foi realizada com o cuidado de se preservar ao máximo os bons resultados teóricos obtidos pelos autores. Isso felizmente foi conseguido, através de

um resultado de equivalência que diz que toda sequência gerada pelo algoritmo Lagrangiano Aumentado sem derivadas poderia ter sido gerada pelo algoritmo original. Nosso trabalho não é o primeiro a tratar de problemas com restrições sem utilizar derivadas, tampouco pioneiro em se beneficiar de um algoritmo Lagrangiano Aumentado já existente [20]. No entanto, até onde sabemos, não há outro método sem derivadas criado sob a condição de qualificação CPLD [4], característica herdada do algoritmo em [3], que torna nossa teoria mais abrangente do que a encontrada em trabalhos similares.

Dizemos que uma propriedade a ser verificada por pontos factíveis é uma *condição de qualificação* se todos os minimizadores locais do problema que a satisfazem verificarem a condição KKT (Karush-Kuhn-Tucker). Três condições de qualificação aparecerão em vários momentos deste texto: a regularidade, Mangasarian-Fromovitz e a já citada condição CPLD. Adiantamo-nos em defini-las. Consideremos $I(x)$ o conjunto dos índices referentes às condições de desigualdades ativas em x , ou seja, $I(x) = \{i \in \{1, \dots, p\} \mid g_i(x) = 0\}$, e $J = \{1, \dots, m\}$. A regularidade, ou condição de qualificação de independência linear, é satisfeita pelos pontos $x \in \Gamma$ tais que os gradientes $\{\nabla g_i(x)\}_{i \in I(x)} \cup \{\nabla h_j(x)\}_{j \in J}$ são linearmente independentes. Essa é provavelmente a condição de qualificação mais usada na literatura. Já a condição de Mangasarian-Fromovitz é satisfeita para $x \in \Gamma$ se os gradientes das restrições de igualdade são linearmente independentes em x e se existe uma direção $d \in \mathbb{R}^n$ tal que $\nabla h_j(x)^T d = 0$ para $j \in J$ e $\nabla g_i(x)^T d < 0$ para $i \in I(x)$.

Para definirmos a condição CPLD (Condição de Dependência Linear Positiva Constante), precisamos da definição de dependência linear positiva. Seja $x \in \Gamma$, $I_0 \subset I(x)$ e $J_0 \subset J$. Os gradientes $\{\nabla g_i(x)\}_{i \in I_0} \cup \{\nabla h_j(x)\}_{j \in J_0}$ são positivo-linearmente dependentes se existem escalares $\{\alpha_j\}_{j \in J_0}$ e $\{\beta_i\}_{i \in I_0}$ tais que $\beta_i \geq 0$ para todo $i \in I_0$, ao menos um dos escalares é diferente de zero e

$$\sum_{j \in J_0} \alpha_j \nabla h_j(x) + \sum_{i \in I_0} \beta_i \nabla g_i(x) = 0.$$

A condição CPLD diz que se $x \in \Gamma$ a satisfaz, então para quaisquer $I_0 \subset I(x)$ e $J_0 \subset J$ tais que os gradientes $\{\nabla g_i(x)\}_{i \in I_0} \cup \{\nabla h_j(x)\}_{j \in J_0}$ são positivo-linearmente dependentes, existe uma vizinhança $V(x)$ de x tal que para todo $y \in V(x)$, o conjunto $\{\nabla g_i(y)\}_{i \in I_0} \cup \{\nabla h_j(y)\}_{j \in J_0}$ é linearmente dependente.

A condição CPLD é implicada pela condição de Mangasarian-Fromovitz, que por sua vez é implicada pela regularidade. Por isso dizemos que a condição CPLD é a mais fraca entre as três. No entanto, condições mais fracas são satisfeitas por uma quantidade maior de pontos, de modo que resultados com estas são mais expressivos do que os obtidos sob condições mais fortes. A teoria que permite caracterizar a condição CPLD como condição de qualificação pode ser encontrada em [4].

A apresentação de nossos resultados está dividida em três capítulos. No primeiro, discutimos brevemente alguns algoritmos sem derivadas. Muitos trabalhos interessantes não são sequer citados, pois a lista de métodos sem derivadas relevantes é vasta, e preferimos nos ater àqueles que de alguma forma nos influenciaram. No segundo capítulo, apresentamos nossos algoritmos Lagrangiano Aumentado sem derivadas, seguidos pelos teoremas associados. Por fim, no Capítulo 3 apresentamos os resultados numéricos obtidos ao testar a implementação que fizemos do nosso método. Entre os experimentos, destacamos o problema de minimizar simulações de áreas de figuras, por ser um problema de fácil compreensão e implementação ainda que não seja possível calcular suas derivadas.

Notação:

- $[v]_i$ é a i -ésima componente do vetor v , que poderá ser designada por v_i quando não houver possibilidade de confusão,
- $e^i \in \mathbb{R}^n$ é a i -ésima direção canônica, ou seja, $[e^i]_i = 1$ e $[e^i]_j = 0$ se $i \neq j$,
- se $x, y \in \mathbb{R}^n$ e $x \leq y$ então $x_i \leq y_i$, $i = 1, \dots, n$,
- $\|x\|$ é a norma infinito de x , ou seja, $\|x\| = \max_{i=1, \dots, n} \{|x_i|\}$,
- se $A \in \mathbb{R}^{m \times n}$ e $d \in A$, então d é uma coluna da matriz A .

Capítulo 1

Métodos sem Derivadas

Métodos sem derivadas, como a própria expressão explica, são aqueles que não utilizam derivadas em nenhum de seus passos. Estes vêm recebendo crescente atenção por parte da comunidade científica. Muitos procedimentos clássicos dessa categoria foram propostos na década de 60. A resistência enfrentada pelos autores na época foi grande, principalmente pela ausência de bons resultados teóricos. No entanto, já há muitos anos as publicações direcionadas ao assunto trazem teoremas tão expressivos quanto os apresentados para métodos baseados em derivadas.

Ao abandonar o uso de derivadas das funções envolvidas nos problemas, seja na forma de seus gradientes, hessianas, derivadas direcionais ou outros, somos privados de aplicar muitas estratégias corriqueiras em otimização. Por exemplo, devemos nos abster de checar se uma direção é de descida calculando seu produto escalar com o gradiente. Não nos é permitido nem ao menos calcular normas de gradientes como medida de estacionariedade nos critérios de parada. Apesar de ainda assim bons resultados teóricos serem atingidos, é evidente que o não uso de derivadas acarreta um desempenho inferior em relação aos métodos clássicos da literatura, especialmente em relação ao tempo computacional. A proposta da otimização sem derivadas não é, pois, desenvolver algoritmos que superem os tradicionais, senão resolver os problemas onde as derivadas não estão disponíveis. Esse é o caso quando a função objetivo é uma caixa-preta, ou seja, um arquivo executável cujo código não está acessível ao usuário. Ou simplesmente quando não há expressão para a função objetivo, como na situação em que esta é uma medida de desempenho computacional de um algoritmo [7]. Mas talvez o contexto mais relevante de impossibilidade de cálculo de derivadas seja o de simulações. Quando a função objetivo é uma simulação, não faz sentido calcularmos suas derivadas. Isso é muito frequente em problemas práticos, notoriamente naqueles que envolvem design de veículos aéreos [6, 11]. Neste caso, propostas de algoritmos sem derivadas eficazes se fazem necessárias. É importante observar que diferentes problemas trazem diferentes dificuldades para os métodos. As mais

comuns são o custoso esforço computacional para se avaliar a função objetivo, e a imprecisão que esta apresenta, na forma do chamado ruído. Evidentemente, não há um algoritmo sem derivadas que seja eficiente para todos os tipos de problemas, de modo que a escolha do método é uma etapa importante na abordagem de problemas práticos.

Outra razão que justifica o estudo de métodos sem derivadas é a possibilidade de aplicá-los de modo quase imediato, seja pela estrutura via de regra simples dos algoritmos sem derivadas clássicos, que torna fácil programá-los, ou pelo fato de não ser necessário implementar gradientes ou hessianas ou recorrer a pacotes de diferenciação automática. Não há dúvidas de que o tempo despendido pelo usuário para transcrever o problema para código computacional se reduz drasticamente ao não ser mais necessário implementar as derivadas, ainda que o tempo de execução do programa seja potencialmente grande.

Há uma classe especial de algoritmos sem derivadas que merece ser destacada. Dizemos que um método é de busca direta se, além de não computar derivadas, ele não utilizar os valores de função em nenhum cálculo [22]. Só é permitido a um algoritmo de busca direta questionar qual entre dois pontos tem o menor valor de função objetivo. Os valores de função nunca precisam ser explicitados de fato. Logo, um algoritmo que interpola a função objetivo por quadráticas ou que aproxima gradientes por diferenças finitas pode ser sem derivadas, porém não de busca direta. Uma consequência imediata desta definição é que tais métodos não podem impor critérios de decréscimo suficiente.

Este capítulo trata de alguns algoritmos sem derivadas propostos na literatura. Sem a pretensão de querer esgotar a extensa lista de algoritmos deste tipo, nosso intuito é apenas citar aqueles que motivaram nosso trabalho, sendo que alguns deles foram utilizados em nossos experimentos numéricos. A maioria dos trabalhos que discutiremos tem como foco a minimização irrestrita ou para restrições simples, não só pela grande oferta de métodos interessantes propostos para este tipo de problema, como também pelo fato de estarmos buscando algoritmos que resolvam subproblemas com este formato, como veremos no próximo capítulo. Uma análise mais profunda sobre métodos sem derivadas pode ser encontrada em [12].

1.1 Algoritmos de Busca Padrão e Correlatos

Métodos de busca padrão sintetizam o espírito dos bons métodos de busca direta, de algoritmos de forte apelo geométrico com garantias teóricas de convergência. Em [32], Torczon define o modelo geral de algoritmo de busca padrão, apresenta resultados de convergência e demonstra que alguns algoritmos de busca direta populares são casos especiais de busca padrão, de modo que as propriedades teóricas deste valem também para aqueles. Podem ser

classificados como busca padrão o algoritmo proposto por Hook e Jeeves [16], que é tido como o primeiro método de busca direta, e o algoritmo de busca coordenada, que foi proposto mais de uma vez sob diversas nomenclaturas, e teve sua convergência anteriormente comprovada em [27].

Na k -ésima iteração do algoritmo, o *padrão* é definido pela matriz $P^k \in \mathbb{R}^{n \times p}$, $P^k = [BM^k \ -BM^k \ BL^k]$, onde $B \in \mathbb{R}^{n \times n}$ é não singular, $M^k \in M$ com M sendo um conjunto finito de matrizes não singulares com elementos inteiros e L^k um conjunto de $p - 2n$ colunas cujas componentes são inteiros. A partir do ponto corrente x^k , são testados os pontos da forma $x^k + \Delta_k d^k$, onde $d^k \in P^k$ e Δ_k é o passo a ser dado em cada direção. Quando há decréscimo em ao menos alguma das direções $\pm BM^k$, então x^{k+1} é definido como um dos pontos testados com menor valor de função objetivo em relação a x^k , mas não necessariamente aquele que apresentou o maior decréscimo. Neste caso, $\Delta_{k+1} = \lambda_k \Delta_k$, sendo que o fator $\lambda_k \geq 1$ satisfaz propriedades convenientes, cuja discussão foge do escopo do presente texto. Se não há decréscimo para nenhum ponto $x^k + \Delta_k d^k$, então $x^{k+1} = x^k$ e $\Delta_{k+1} = \alpha \Delta_k$, onde $0 < \alpha < 1$ é racional.

Definindo $B = M^k = I$, obtemos o método de busca coordenada. Em [32] é estabelecida de maneira unívoca a ordem em que as direções devem ser testadas na busca coordenada, e permitindo que direções não canônicas com elementos pertencentes ao conjunto $\{-1, 0, 1\}$, que são definidas como colunas de BL^k , sejam tomadas. As colunas de BL^k aparecem por uma escolha de notação, já que seria indiferente, por exemplo, dar um passo na direção $(1, \dots, 1)^T$ ou dar n passos nas direções e^i , $i = 1, \dots, n$. É assim que o algoritmo proposto daria um passo na direção $(1, \dots, 1)^T$, mas por questões de definição, isso seria considerado apenas uma iteração, e não n iterações. De qualquer forma, contrariando o espírito de unificação do método, consideramos que busca coordenada é qualquer algoritmo de busca padrão que use apenas as direções coordenadas como direções de busca, não importando a ordem em que são testadas.

A demonstração de que uma sequência gerada por um algoritmo de busca padrão possui uma subsequência que converge a um ponto estacionário é baseada em duas propriedades dos algoritmos, e que foram exploradas por várias publicações posteriores. A primeira é de que os pontos visitados estão em treliças racionais, que podem ficar mais densas à medida que Δ_k diminui. Isso é usado em um lema auxiliar que estuda o comportamento de Δ_k , que deve ter uma subsequência tendendo a zero, o que é demonstrado utilizando o fato de que o conjunto de nível $L(x^0)$, onde estão todos os iterandos, é suposto compacto, e sua intersecção com qualquer treliça racional possui finitos pontos. A outra propriedade importante é a de que as direções de busca geram positivamente o \mathbb{R}^n , pois contém colunas das matrizes BM^k e $-BM^k$.

É possível demonstrar que a sequência de pontos gerados converge a um ponto estacionário no sentido amplo de convergência, não apenas para subsequências. Entretanto, algumas hipóteses extras são necessárias, as mais importantes são a de que a sequência de passos converge a zero e que o passo dado em uma iteração deve ser tal que $f(x^k + \Delta_k d^k) \leq \min\{f(x^k \pm \Delta_k y) \mid y \in BM^k\}$.

Vários outros trabalhos foram elaborados em sequência, tomando como base o artigo original sobre busca padrão. Em [21], os autores apresentam uma versão da busca padrão para minimização em caixas. Estamos particularmente interessados neste problema, que será abordado no capítulo seguinte como subproblema advindo de um algoritmo tipo Lagrangiano Aumentado.

O conjunto de direções de busca deve satisfazer a propriedade de que se um ponto não é estacionário, então ao menos uma das direções deve ser de descida. No caso do algoritmo de busca padrão e restrições de caixa, isso é conseguido impondo que as matrizes BM^k sejam diagonais. Tal condição é satisfeita naturalmente pela busca coordenada. Outra modificação no algoritmo de busca padrão para adequá-lo ao presente contexto está no fato de que todo iterando deve ser interior à caixa. Uma consequência indesejada é que pontos na fronteira provavelmente só serão atingidos por uma sequência de pontos interiores. No entanto, os resultados de convergência de [32] foram recuperados sob as mesmas hipóteses, tanto o de convergência usual como o relativo a subsequências. A figura abaixo ilustra o funcionamento de um algoritmo de busca coordenada em uma caixa, para minimizar a função $f(x)$ a partir do ponto inicial x^0 .

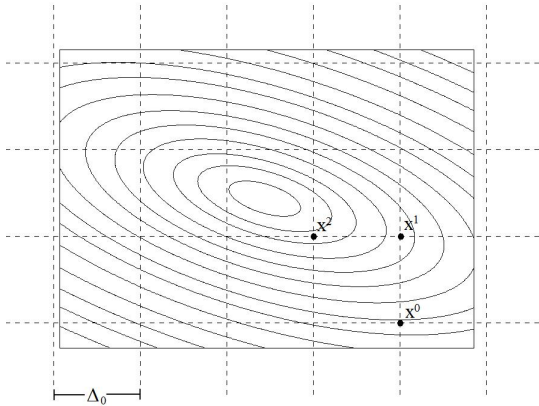


Figura 1.1: Busca coordenada em caixas, $\Delta = \Delta_0$.

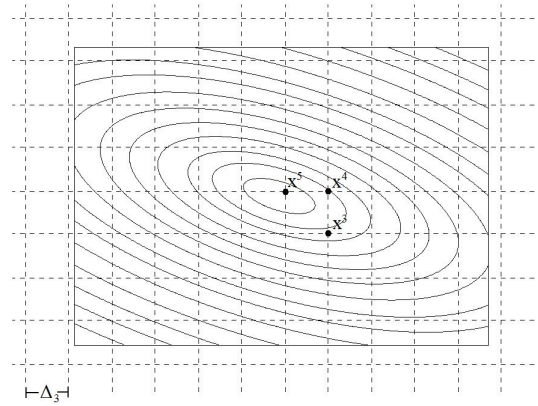


Figura 1.2: Busca coordenada em caixas, $\Delta = \Delta_0/2$.

Dado o passo inicial Δ_0 , nos movemos para o ponto x^1 com $f(x^1) < f(x^0)$, e depois para x^2 com $f(x^2) < f(x^1)$. Como estas foram iterações onde houve sucesso, o tamanho do passo foi mantido, ou seja, $\Delta_2 = \Delta_1 = \Delta_0$. Essas etapas estão representadas na figura

(1.1). Analisando o ponto x^2 , vemos que não há um ponto vizinho na malha com menor valor de função. Deste modo, declaramos fracasso na iteração e fazemos $x^3 = x^2$ e $\Delta_3 = \Delta_2/2$ (estamos usando $\alpha = 1/2$). Visitando pontos vizinhos como descrito anteriormente, chegamos ao ponto x^5 , onde novamente seríamos obrigados a reduzir o tamanho do passo, como mostrado na figura (1.2).

Em [18] e em artigos correlatos, os autores apresentam versões de busca padrão para restrições lineares. A dificuldade em manter uma direção de descida factível entre as direções de busca para pontos não estacionários é muito maior em comparação ao caso de restrições de caixa. Um estudo de cones normais e tangentes é feito nesse sentido. Há dois métodos propostos: um baseado em treliças racionais, como nos trabalhos anteriores, e outro que utiliza decréscimo suficiente em detrimento ao decréscimo simples, o que faz com que o algoritmo não seja mais de busca direta.

Vários métodos trazem as mesmas ideias apresentadas pelos trabalhos até então discutidos. Lucidi e Sciandrone propõem em [24] um algoritmo sem derivadas para minimização irrestrita que utiliza direções de busca que satisfazem condições semelhantes às empregadas em busca padrão. Em resumo, duas classes de direções são apresentadas. A primeira é formada por sequências de direções uniformemente linearmente independentes, cujos pontos limites geram positivamente o \mathbb{R}^n . A segunda utiliza n sequências de direções uniformemente linearmente independentes, e a cada iteração é testada uma direção que emula a de máxima descida; quando isso não é possível, utiliza uma que faça com que as direções da iteração dada gerem positivamente o \mathbb{R}^n . Há um procedimento de busca linear, que explora direções onde há diminuição da função objetivo. Um critério de decréscimo suficiente é apresentado. Com esses ingredientes, é criado um algoritmo que possui a propriedade de gerar sequências cujos limites são pontos estacionários. Em [23] é apresentado um algoritmo muito semelhante, porém adequado a problemas em caixas. Como em [21], as direções coordenadas desempenham papel fundamental nas demonstrações de convergência.

Os métodos que utilizam um conjunto finito de direções de busca jamais serão adequados a problemas não suaves. No caso irrestrito, se a derivada é descontínua em algum ponto, não há garantias de que existe ao menos uma direção de descida mesmo que o conjunto de direções de busca gerem positivamente o \mathbb{R}^n . Um conjunto pequeno de direções de busca pode trazer dificuldades mesmo em problemas suaves, por exemplo, através de uma lentidão intolerável na convergência. Para tentar resolver essas questões, Audet e Dennis desenvolveram um método semelhante conceitualmente à busca padrão, mas que utiliza um número infinito de direções de busca, chamado MADS (Mesh Adaptive Direct Search) [5]. O artigo aborda problemas irrestritos ou com qualquer tipo de restrições, impondo que $f(x) = \infty$ se x é infactível, o que é chamado de *barreira extrema*. Pela maneira com que as restrições são

tratadas, não é necessário saber a expressão das funções que as definem, tampouco se um ponto é mais infactível que outro de acordo com alguma medida de infactibilidade. Toda a teoria de convergência é baseada no cálculo introduzido por Clarke para funções não suaves, de modo que o único requerimento teórico para as funções envolvidas é que sejam Lipschitz. O decréscimo exigido para um novo ponto é o decréscimo simples.

Há dois passos onde um ponto factível melhor é procurado. O primeiro é opcional, e é chamado passo de busca (search step). Ele consiste em uma busca sobre a malha M_k da forma

$$M_k = \bigcup_{x \in S_k} \{x + \Delta_k^m D z \mid z \in \mathbb{N}^{n_D}\},$$

onde S_k é o conjunto de todos os pontos onde a função objetivo foi computada desde o início da iteração k , e D , apesar de vir de uma definição mais complexa, pode ser visto apenas como uma matriz real $n \times n_D$.

O diferencial do algoritmo está no passo de pesquisa (poll step), onde são investigados pontos pertencentes à estrutura (frame) P_k definida por

$$P_k = \{x^k + \Delta_k^m d \mid d \in D_k\} \subset M_k,$$

onde o conjunto D_k gera positivamente o \mathbb{R}^n , suas componentes são combinações inteiras não negativas das direções em D , a distância dos pontos investigados a x^k é limitada pelo passo da estrutura, Δ_k^p , e os limites das direções d normalizadas devem gerar positivamente o \mathbb{R}^n . Os passos devem satisfazer a propriedade $\Delta_k^m \leq \Delta_k^p$. A cada iteração, é performed um passo de busca, seguido por um passo de pesquisa se o primeiro falhar em melhorar a função objetivo. Similarmente ao que ocorre em busca padrão, o passo da malha Δ_k^m pode aumentar se um ponto melhor for encontrado, e deve ser reduzido em caso contrário, sempre multiplicando-o por escalares pertencentes a um conjunto finito convenientemente escolhido. As figuras 1.3 e 1.4 mostram exemplos de estruturas no plano. Os pontos p^1, p^2 e p^3 são escolhas possíveis para formarem a estrutura, através das direções $p^i - x^k$, $i = 1, 2, 3$. As linhas escuras representam a caixa onde os pontos da estrutura devem estar contidos, definida pelo valor Δ_k^p .

Um ponto x^k é chamado centro mínimo da estrutura (minimal frame center) se $\Delta_{k+1}^m < \Delta_k^m$. Sendo K um subconjunto dos índices dos centros mínimos da estrutura, $\{x^k\}_{k \in K}$ é chamada subsequência de refinamento (refining subsequence) se $\{\Delta_k^m\}_{k \in K}$ converge a zero. Chamemos seu limite de \hat{x} . Se d^k é uma subsequência de direções de pesquisa e se existe um subconjunto $L \subseteq K$ tal que o limite de $\{d^k / \|d^k\|\}_{k \in L}$ existe e $x^k + \Delta_k^m d^k$ é factível para infinitos $k \in L$, então esse limite é chamado direção de refinamento para \hat{x} . Os autores já haviam demonstrado em artigo prévio que há ao menos uma subsequência de refinamento.

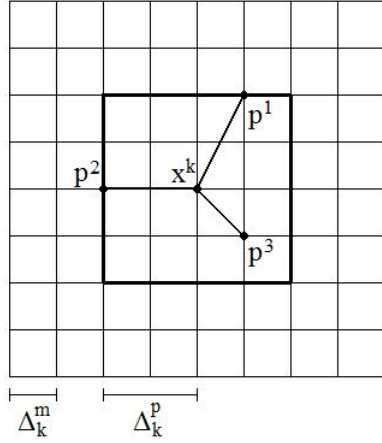


Figura 1.3: Exemplo de estrutura para $\Delta_k^m = \frac{1}{4}$ e $\Delta_k^p = \frac{1}{2}$.

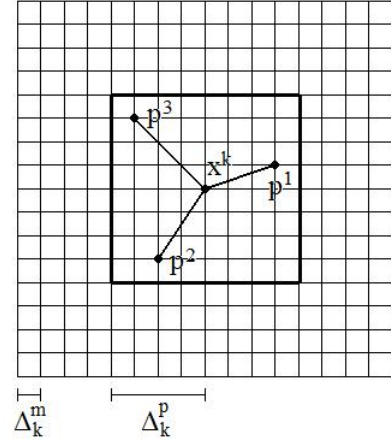


Figura 1.4: Exemplo de estrutura para $\Delta_k^m = \frac{1}{8}$ e $\Delta_k^p = \frac{1}{2}$.

Mas no trabalho que estamos agora discutindo, provam que se o conjunto de direções de refinamento para \hat{x} é denso no cone hipertangente, então \hat{x} é um ponto estacionário de Clarke para o problema.

A proposta dos autores para cumprir a exigência de que as direções sejam densas é baseada em direções aleatórias. O processo de geração dessas direções tem como ponto de partida a criação de uma matriz triangular inferior com componentes inteiras aleatórias, que depois de alguns cálculos trará as direções desejadas. O algoritmo MADS com a escolha de direções apresentada é chamado LTMADS. Como é baseado em componentes randômicas, o resultado teórico é que as direções de pesquisa são assintoticamente densas no cone hipertangente com probabilidade 1. Uma maneira determinística de se computar direções densas foi recentemente divulgada em [1], baseada em sequências quase randômicas. O algoritmo obtido com essa escolha de direções recebe o nome de ORTHOMADS. Os algoritmos LTMADS e ORTHOMADS fazem parte do software para otimização NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search) [34].

1.2 Algoritmo Nelder-Mead

No contexto de otimização sem derivadas, denominamos algoritmos simplex aqueles que trabalham com $n + 1$ pontos correntes simultaneamente. Recebem este nome pois $n + 1$ pontos no \mathbb{R}^n definem um simplex. Os simplex utilizados devem ser não degenerados, ou seja, se é definido pelos pontos $\{x^0, \dots, x^n\}$, então o conjunto $\{x^1 - x^0, \dots, x^n - x^0\}$ deve ser linearmente independente. Os algoritmos simplex tentam substituir a cada iteração o

pior vértice, ou seja, aquele que apresenta maior valor de função objetivo. Quando isso não é possível, reduzem o tamanho do simplex diminuindo a distância entre os pontos que o compõem. É válido frisar que tais métodos não são derivados do Método Simplex para Programação Linear.

O mais famoso método simplex é provavelmente o método sem derivadas mais utilizado na prática. Foi proposto em 1965 por Nelder e Mead. Ao tentar substituir o pior ponto, quatro tipos de movimento são possíveis: reflexão, expansão, contração interna e contração externa. Os candidatos a novos pontos para os simplex devem estar na reta que liga o pior vértice ao centróide da face formada pelos vértices restantes. Apresentar as razões que fazem com que o algoritmo escolha um tipo de movimento em detrimento a outro parece desnecessário nesta breve discussão, mas é relevante observar que, qualquer que seja o movimento escolhido, ele deve trazer um ponto que seja melhor do que o ponto que está sendo substituído. Quando não é possível encontrar tal ponto, o simplex é reduzido na direção do melhor ponto. A figura 1.5 mostra as possibilidades de substituição para o pior ponto, enquanto que a figura 1.6 mostra como é a redução.

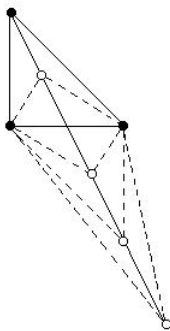


Figura 1.5: Substituição do pior vértice.

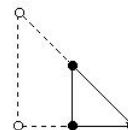


Figura 1.6: Redução do simplex.

Há liberdade na escolha dos coeficientes que definem os movimentos. Nas figuras acima, foi escolhido o valor 1 para o coeficiente de reflexão $\rho > 0$; 2 para o coeficiente de expansão $\chi > \max\{1, \rho\}$; 0.5 para o coeficiente de contração $0 < \gamma < 1$ e 0.5 para o coeficiente de redução $0 < \sigma < 1$. Esses valores são canônicos, e nunca nos deparamos com implementações que utilizem outros valores.

Em [19], o algoritmo Nelder-Mead é minuciosamente analisado. Os autores preenchem algumas lacunas sobre como proceder em caso de empate entre dois vértices, além de discutir a não-degenerescência dos simplex gerados e o comportamento do método para funções estritamente convexas. Seus principais resultados, no entanto, são a demonstração de convergência a minimizadores para $n = 1$ e a de que para $n = 2$ os diâmetros dos simplex tendem a zero, com vértices que no limite têm o mesmo valor de função. Apesar de toda a popularidade da

qual o algoritmo Nelder-Mead goza, não há garantias de convergência para $n > 1$. De fato, muitos de seus entusiastas classificavam a dificuldade que o método experimenta para alguns problemas como simples lentidão. Mas em [25], McKinnon apresenta uma família de funções em \mathbb{R}^2 para as quais o algoritmo converge a pontos não estacionários. Dentre as funções pertencentes a esta família, estão funções estritamente convexas com derivadas contínuas até a ordem 3. Esse comportamento inadequado do método pode ser evitado com a imposição de restrições sobre os ângulos internos dos simplex e implementação de decréscimo suficiente [33].

1.3 Algoritmo NEWUOA

Uma técnica bastante conhecida para evitarmos o cálculo das derivadas de f é aproximá-la por quadráticas a serem minimizadas nas sucessivas iterações. É dessa forma que trabalha o algoritmo para minimização irrestrita NEWUOA (New unconstrained optimization algorithm), proposto por Powell em 2004. Ele foi desenvolvido através de uma pesquisa de vários anos, que culminou no algoritmo apresentado em [30]. Todo o processo de aperfeiçoamento do método pode ser observado durante a leitura dos diversos artigos citados em [30].

O principal parâmetro algorítmico é o número de pontos m para se construir a interpolação quadrática Q que aproxima f localmente. Se $m = \frac{1}{2}(n+1)(n+2)$, a quadrática é determinada univocamente, através da resolução de um sistema linear. O algoritmo exige que $n+2 \leq m \leq \frac{1}{2}(n+1)(n+2)$, sendo que o valor sugerido é $m = 2n+1$. Para $m < \frac{1}{2}(n+1)(n+2)$, as quadráticas também são determinadas de maneira única, impondo que a diferença entre duas quadráticas consecutivas seja a menor possível. Mais especificamente, se Q^{old} é a quadrática corrente que deve ser substituída por Q^{new} , então além das condições de interpolação serem satisfeitas, a quantidade $\|\nabla^2 Q^{new} - \nabla^2 Q^{old}\|_F$ deve ser mínima, onde $\|\cdot\|_F$ é a norma de Frobenius. Vale observar que, de uma iteração a outra, apenas um ponto de interpolação pode ser modificado. Nos artigos relacionados a NEWUOA, é estudado como atualizar a quadrática de modo a satisfazer as condições de interpolação e de minimização da norma de Frobenius. O sistema linear a ser resolvido para encontrar os parâmetros de Q^{new} foi apresentado nos diversos trabalhos, e em [30] o autor concluiu que a melhor maneira de manipular esse sistema é armazenando e atualizando a inversa da matriz que o define, sendo que parte dessa matriz é fatorada antes de ser armazenada, contornando assim o problema da instabilidade numérica observada em versões anteriores do algoritmo. Dada uma quadrática, o problema de região de confiança

$$\text{minimizar } Q(x^{opt} + d) \text{ sujeita a } \|d\| \leq \Delta$$

é resolvido através de uma versão do algoritmo de gradientes conjugados truncado, onde Δ é o raio de confiança limitado inferiormente pelo parâmetro ρ e x^{opt} é o ponto de interpolação com menor valor de f na iteração atual. A razão

$$\frac{f(x^{opt}) - f(x^{opt} + d)}{Q(x^{opt}) - Q(x^{opt} + d)}$$

é analisada, de modo a decidir se a interpolação está sendo bem sucedida em representar localmente a função f , o que pode resultar na substituição de um dos pontos de interpolação. Essa substituição também pode ocorrer se d retornado pelo algoritmo de região de confiança for muito pequeno. Dependendo do contexto, o novo ponto de interpolação pode ser $x^{opt} + d$ ou algum ponto que vise o bom condicionamento do sistema linear que define Q .

A maneira como os parâmetros Δ e ρ são atualizados, bem como muitas outras questões técnicas são cuidadosamente explicadas no artigo. Nenhum algoritmo geral é apresentado, sendo o intuito do autor esmiuçar todos os detalhes da implementação de NEWUOA, e apresentar toda a base teórica que justifica cada etapa do algoritmo. Mesmo se apoiando em vários teoremas, ao que consta não há demonstrações de convergência. Isso não é de forma alguma surpreendente, dada a complexidade do algoritmo, que foi todo desenhado para responder questões de fundo prático. Ainda assim, experimentos mostram o bom desempenho do algoritmo, cujo uso é defendido inclusive para n grande, já que o trabalho por iteração é apenas da ordem de $(m + n)^2$. O software BOBYQA (Bound optimization by quadratic approximation) traz algumas modificações em NEWUOA para adequá-lo à minimização em caixas. Recentemente o autor apresentou um relatório de pesquisa contendo detalhes da implementação de BOBYQA [29].

Capítulo 2

Lagrangiano Aumentado Sem Derivadas

2.1 Algoritmo Lagrangiano Aumentado

Consideremos o problema geral de minimização

$$\text{minimizar } f(x) \text{ sujeita a } x \in \Gamma, \quad (2.1)$$

onde $\Gamma \subset \mathbb{R}^n$ e $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Resolver este problema é o foco de grande parte da pesquisa em otimização. A complexidade do problema reside principalmente no conjunto de restrições Γ . Existem diversos algoritmos que resolvem com eficácia o problema acima para conjuntos particulares de restrições, como caixas e bolas, por exemplo. Isso ocorre porque a simplicidade destes conjuntos se traduz em propriedades algébricas e geométricas interessantes, que são oportunamente exploradas na elaboração dos algoritmos. Já para conjuntos gerais de restrições, a construção de um algoritmo que aproveite a estrutura do problema torna-se difícil, uma vez que pouco sabemos ou exigimos sobre o conjunto Γ . Uma abordagem interessante para esse problema é convertê-lo numa sequência de problemas de minimização mais fáceis, ou seja, que possuam restrições mais simples. Essa é a filosofia de métodos bastante conhecidos como Penalidade Externa, Barreira e Lagrangiano Aumentado.

Separemos o conjunto de restrições em dois, Ω e $\tilde{\Omega}$, de modo que $\Gamma = \Omega \cap \tilde{\Omega}$. Chamemos a restrição $x \in \Omega$ de restrição do nível superior e $x \in \tilde{\Omega}$ de restrição do nível inferior. Desse modo, (2.1) pode ser reescrito como

$$\text{minimizar } f(x) \text{ sujeita a } x \in \Omega \cap \tilde{\Omega}. \quad (2.2)$$

A divisão do conjunto Γ é baseada no critério de simplicidade das restrições. Estamos supondo que o conjunto $\tilde{\Omega}$ é tal que dispomos de um algoritmo eficiente para a sequência de problemas

$$\text{minimizar } F^k(x) \text{ sujeita a } x \in \tilde{\Omega}, \quad k = 1, 2, \dots \quad (2.3)$$

As funções $F^k : \mathbb{R}^n \rightarrow \mathbb{R}$ devem ser tais que seus minimizadores restritos a $\tilde{\Omega}$ sejam preferencialmente mais factíveis em relação a Ω e mais ótimos em relação a $f(x)$ à medida que k cresce. Nesta seção, discutiremos o algoritmo proposto em [3], que lida com conjuntos Ω e $\tilde{\Omega}$ que podem conter restrições de igualdade e desigualdade, definindo assim problemas da forma

$$\text{minimizar } f(x) \text{ sujeita a } h(x) = 0, \quad g(x) \leq 0, \quad \tilde{h}(x) = 0, \quad \tilde{g}(x) \leq 0, \quad (2.4)$$

onde $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $h : x \mapsto [h_1(x) \dots h_m(x)]^T$; $\tilde{h} : \mathbb{R}^n \rightarrow \mathbb{R}^{\tilde{m}}$, $\tilde{h} : x \mapsto [\tilde{h}_1(x) \dots \tilde{h}_{\tilde{m}}(x)]^T$; $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$, $g : x \mapsto [g_1(x) \dots g_p(x)]^T$ e $\tilde{g} : \mathbb{R}^n \rightarrow \mathbb{R}^{\tilde{p}}$, $\tilde{g} : x \mapsto [\tilde{g}_1(x) \dots \tilde{g}_{\tilde{p}}(x)]^T$.

O problema acima é atacado através de uma sequência de problemas (2.3), com $F^k(x)$ sendo sucessivas funções Lagrangiano Aumentado $L(x, \lambda, \mu, \rho)$ propostas por Hestenes [14], Powell [28] e Rockafellar [31]:

$$L(x, \lambda, \mu, \rho) = f(x) + \frac{\rho}{2} \sum_{i=1}^m \left(h_i(x) + \frac{\lambda_i}{\rho} \right)^2 + \frac{\rho}{2} \sum_{i=1}^p \max \left\{ 0, g_i(x) + \frac{\mu_i}{\rho} \right\}^2,$$

com $\rho \in \mathbb{R}_+$, $\lambda \in \mathbb{R}^m$ e $\mu \in \mathbb{R}_+^p$. As funções envolvidas devem ter derivadas contínuas em um aberto grande o suficiente. Os vetores λ e μ são estimativas para os multiplicadores de Lagrange associados, respectivamente, às restrições de igualdade e desigualdade do nível superior. Já ρ é o parâmetro de penalidade. O preço pago pela escolha da definição acima para a função Lagrangiano Aumentado é a não existência de derivadas segundas no caso em que há restrições do tipo $g(x) \leq 0$. Contudo, os autores defendem que esse não é um ônus tão grande quanto o que seria trazido pela transformação das restrições de desigualdade em igualdade através da introdução de variáveis de folga. Na k -ésima vez em que o problema (2.3) é resolvido, os valores ρ_k , $\bar{\lambda}^k$ e $\bar{\mu}^k$ para ρ , λ e μ permanecem constantes, de modo que a minimização é apenas em x , sendo posteriormente atualizados. Dessa forma, temos que $F^k(x) = L(x, \bar{\lambda}^k, \bar{\mu}^k, \rho_k)$. Quando conveniente, denotaremos $L(x, \bar{\lambda}^k, \bar{\mu}^k, \rho_k)$ por $L^k(x)$. Reprodiremos agora o algoritmo apresentado em [3], e iremos chamá-lo Algoritmo 1.

Algoritmo 1 (Lagrangiano Aumentado para restrições gerais)

Seja $x^0 \in \mathbb{R}^n$ um ponto inicial arbitrário. Os parâmetros algorítmicos a serem definidos são: $\tau \in [0, 1)$, $\gamma > 1$, $\rho_1 > 0$, $-\infty < \bar{\lambda}_i^{min} \leq \bar{\lambda}_i^{max} < \infty \forall i = 1, \dots, m$, $0 \leq \bar{\mu}_i^{max} < \infty \forall i = 1, \dots, p$, $\bar{\lambda}_i^1 \in [\bar{\lambda}_i^{min}, \bar{\lambda}_i^{max}] \forall i = 1, \dots, m$, $\bar{\mu}_i^1 \in [0, \bar{\mu}_i^{max}] \forall i = 1, \dots, p$, $\{\varepsilon_k\} \subset \mathbb{R}_+$ tal que $\lim_{k \rightarrow \infty} \varepsilon_k = 0$.

Passo 1: *Inicialização*

Fazer $k \leftarrow 1$. Para $i = 1, \dots, p$, calcular $\sigma_i^0 = \max\{0, g_i(x^0)\}$.

Passo 2: *Resolução do subproblema*

Encontrar $x^k \in \mathbb{R}^n$ tal que existam $\tilde{\lambda}^k \in \mathbb{R}^{\tilde{m}}$ e $\tilde{\mu}^k \in \mathbb{R}^{\tilde{p}}$ satisfazendo

$$\|\nabla L^k(x^k) + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k \nabla \tilde{h}_i(x^k) + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k \nabla \tilde{g}_i(x^k)\| \leq \varepsilon_k, \quad (2.5)$$

$$\tilde{\mu}_i^k \geq 0 \text{ e } \tilde{g}_i(x^k) \leq \varepsilon_k \text{ para } i = 1, \dots, \tilde{p}, \quad (2.6)$$

$$\tilde{g}_i(x^k) < -\varepsilon_k \Rightarrow \tilde{\mu}_i^k = 0, \text{ para } i = 1, \dots, \tilde{p}, \quad (2.7)$$

$$\|\tilde{h}(x^k)\| \leq \varepsilon_k. \quad (2.8)$$

Se não for possível encontrar x^k satisfazendo (2.5)-(2.8), parar a execução do algoritmo.

Passo 3: *Atualização das estimativas para os multiplicadores*

Para $i = 1, \dots, m$, calcular

$$\begin{aligned} \lambda_i^{k+1} &= \bar{\lambda}_i^k + \rho_k h_i(x^k), \\ \bar{\lambda}_i^{k+1} &= \min\{\max\{\lambda_i^{k+1}, \bar{\lambda}_i^{min}\}, \bar{\lambda}_i^{max}\}. \end{aligned} \quad (2.9)$$

Para $i = 1, \dots, p$, calcular

$$\begin{aligned} \mu_i^{k+1} &= \max\{0, \bar{\mu}_i^k + \rho_k g_i(x^k)\}, \\ \sigma_i^k &= \max\left\{g_i(x^k), -\frac{\bar{\mu}_i^k}{\rho_k}\right\}, \\ \bar{\mu}_i^{k+1} &= \min\{\mu_i^{k+1}, \bar{\mu}_i^{max}\}. \end{aligned} \quad (2.10)$$

Passo 4: *Atualização do parâmetro de penalidade*

Se $\max\{\|h(x^k)\|, \|\sigma^k\|\} \leq \tau \max\{\|h(x^{k-1})\|, \|\sigma^{k-1}\|\}$, definir $\rho_{k+1} = \rho_k$. Caso contrário, definir $\rho_{k+1} = \gamma \rho_k$.

Passo 5: *Nova iteração externa*

Fazer $k \leftarrow k + 1$. Voltar ao Passo 2.

A atualização das estimativas para os multiplicadores de Lagrange são correções clássicas

de primeira ordem propostas em [14, 28, 31]. Além disso, os multiplicadores devem obrigatoriamente estar contidos em uma caixa. Segundo os autores, essa exigência visa a propriedade de que minimizadores globais dos subproblemas convirjam em k a minimizadores globais do problema original, propriedade esta herdada dos métodos de Penalidade Externa. Em [3], parte dos resultados teóricos é atingida mesmo se $\bar{\lambda}^k$ e $\bar{\mu}^k$ assumirem quaisquer valores dentro das respectivas caixas, e não necessariamente as projeções nestas como em (2.9) e (2.10). Estamos adotando desde já as projeções por questão de simplicidade. Há também uma liberdade maior na escolha das tolerâncias em (2.5)-(2.8), que foi pelo mesmo motivo omitida. As condições (2.5)-(2.8) definem um critério de parada para o algoritmo interno que resolve os subproblemas. Elas representam um relaxamento das condições KKT para o problema de minimizar $L^k(x)$ sujeita a $\tilde{\Omega}$.

A condição CPLD desempenhou um papel importantíssimo na elaboração do método. Por essa razão, os resultados para o Algoritmo 1 são mais abrangentes do que os encontrados em [9, 10], onde a regularidade é requerida. Será sempre suposto que a sequência $\{x^k\}$ gerada pelo Algoritmo 1 possui um ponto limite x^* . Isso ocorre, por exemplo, quando há um $\varepsilon > 0$ tal que o conjunto $\{x \in \mathbb{R}^n \mid \tilde{g}(x) \leq \varepsilon, \|\tilde{h}(x)\| \leq \varepsilon\}$ é limitado. Essa condição é naturalmente atingida quando há restrições de caixa no nível inferior.

O primeiro teorema trata do status dos pontos limites em relação à factibilidade. Ele nos diz que um ponto limite que satisfaz a condição CPLD em relação a $\tilde{\Omega}$ é factível ou ao menos ponto estacionário de uma medida de infactibilidade.

Teorema 1 *Seja x^* ponto limite de uma sequência $\{x^k\}$ gerada pelo Algoritmo 1. Se a sequência de parâmetros de penalidade $\{\rho_k\}$ é limitada, então x^* é factível. Caso contrário, uma das duas possibilidades é verificada:*

- x^* é um ponto KKT do problema

$$\text{minimizar } \sum_{i=1}^m h_i(x)^2 + \sum_{i=1}^p \max\{0, g_i(x)\}^2 \text{ sujeita a } x \in \tilde{\Omega};$$

- x^* não satisfaz a condição CPLD associada a $\tilde{\Omega}$.

Demonstração: Conferir a demonstração do Teorema 4.1 de [3]. □

O segundo teorema prova que, sob a condição CPLD, pontos limites factíveis são estacionários.

Teorema 2 *Seja x^* um ponto limite factível de uma sequência $\{x^k\}$ gerada pelo Algoritmo 1. Se x^* satisfaz a condição CPLD em relação a $\Omega \cap \tilde{\Omega}$, então x^* é ponto KKT do problema*

(2.2). Além disso, se x^* satisfaz a condição de Mangasarian-Fromovitz e $\{x^k\}_{k \in K}$ é uma subsequência que converge a x^* então o conjunto $\{\|\lambda^{k+1}\|, \|\mu^{k+1}\|, \|\tilde{\lambda}^k\|, \|\tilde{\mu}^k\|\}_{k \in K}$ é limitado.

Demonstração: Conferir a demonstração do Teorema 4.2 de [3]. \square

O último teorema trata da limitação do parâmetro de penalidade. As hipóteses abaixo devem ser verificadas, sendo que os vetores $\lambda^*, \mu^*, \tilde{\lambda}^*$ e $\tilde{\mu}^*$ são os multiplicadores de Lagrange em x^* associados respectivamente aos gradientes de h, g, \tilde{h} e \tilde{g} .

Hipótese 1: A sequência x^k converge a x^* .

Hipótese 2: O ponto x^* é factível, ou seja, $h(x^*) = 0, \tilde{h}(x^*) = 0, g(x^*) \leq 0$ e $\tilde{g}(x^*) \leq 0$.

Hipótese 3: O ponto x^* é regular, ou seja, os gradientes $\{\nabla h_i(x^*)\}_{i=1}^m, \{\nabla g_i(x^*)\}_{g_i(x^*)=0}, \{\nabla \tilde{h}_i(x^*)\}_{i=1}^{\tilde{m}}, \{\nabla \tilde{g}_i(x^*)\}_{\tilde{g}_i(x^*)=0}$ são linearmente independentes.

Hipótese 4: As funções $f, h, \tilde{h}, g, \tilde{g}$ admitem derivadas segundas contínuas numa vizinhança de x^* .

Hipótese 5: Seja o subespaço tangente T o conjunto de todos $z \in \mathbb{R}^n$ tais que $\nabla h_i(x^*)^T z = 0, i = 1, \dots, m, \nabla \tilde{h}_i(x^*)^T z = 0, i = 1, \dots, \tilde{m}, \nabla g_i(x^*)^T z = 0, \forall i \mid g_i(x^*) = 0$ e $\nabla \tilde{g}_i(x^*)^T z = 0, \forall i \mid \tilde{g}_i(x^*) = 0$. Então, para todo $z \in T, z \neq 0$, temos que $z^T (\nabla^2 f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla^2 h_i(x^*) + \sum_{i=1}^p \mu_i^* \nabla^2 g_i(x^*) + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^* \nabla^2 \tilde{h}_i(x^*) + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^* \nabla^2 \tilde{g}_i(x^*)) z > 0$.

Hipótese 6: Para todo $i = 1, \dots, m, j = 1, \dots, p, \lambda_i^* \in (\bar{\lambda}_i^{min}, \bar{\lambda}_i^{max}), \mu_i^* \in [0, \bar{\mu}_j^{max})$.

Hipótese 7: Para todo i tal que $g_i(x^*) = 0, \mu_i^* > 0$.

É importante notar que a Hipótese 7 de complementaridade estrita não envolve as restrições de desigualdade do nível inferior. A Hipótese 5 é um pouco mais fraca do que a condição suficiente de segunda ordem, já que nesta o subespaço tangente deve ser ortogonal a um conjunto menor por envolver, entre os gradientes associados às restrições de desigualdade ativas, apenas aqueles com multiplicadores não nulos. O teorema da limitação do parâmetro de penalidade é enunciado abaixo.

Teorema 3 *Suponha que as Hipóteses 1-7 são satisfeitas. Suponha também que existe uma sequência $\eta_k \rightarrow 0$ tal que $\varepsilon_k \leq \eta_k \max\{\|h(x^k)\|, \|\sigma^k\|\}$ para todo $k \in \mathbb{N}$. Então a sequência de parâmetros de penalidade é limitada.*

Demonstração: Conferir a demonstração do Teorema 5.5 de [3]. \square

A demonstração do teorema acima foi feita em duas etapas. Primeiramente, foi provada apenas para o caso de restrições de igualdade nos níveis inferior e superior. Então, um

teorema auxiliar demonstrou que, para um problema geral, as sequências de x^k, ρ_k, λ^k e μ^k podem ser geradas pelo Algoritmo 1 aplicado a um problema convenientemente escolhido envolvendo apenas igualdades, o que implica o Teorema 3.

Quando $\tilde{\Omega}$ é uma caixa, temos provavelmente o exemplo mais importante de aplicação do Algoritmo 1, tanto pela quantidade de problemas práticos que se encaixam nesse perfil como pelos bons algoritmos vistos na literatura que podem ser aplicados na resolução do subproblema. Deste modo, o problema (2.4) toma a forma

$$\text{minimizar } f(x) \text{ sujeita a } h(x) = 0, g(x) \leq 0, l \leq x \leq u, \quad (2.11)$$

onde $l, u \in \mathbb{R}^n, l \leq u$. Como não há igualdades em $\tilde{\Omega} = \{x \in \mathbb{R}^n \mid l \leq x \leq u\}$, as condições (2.5)-(2.8) são agora mais convenientemente representadas do seguinte modo: para todo $j = 1, \dots, n$,

$$|[\nabla L^k(x^k)]_j - \alpha_j^k + \beta_j^k| \leq \varepsilon_k, \quad (2.12)$$

$$-x_j^k + l_j \leq \varepsilon_k, \quad (2.13)$$

$$x_j^k - u_j \leq \varepsilon_k, \quad (2.14)$$

$$\alpha_j^k, \beta_j^k \geq 0, \quad (2.15)$$

$$-x_j^k + l_j < -\varepsilon_k \Rightarrow \alpha_j^k = 0, \quad (2.16)$$

$$x_j^k - u_j < -\varepsilon_k \Rightarrow \beta_j^k = 0. \quad (2.17)$$

Consideremos a função $P : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, tal que para $x, y \in \mathbb{R}^n$, $P(x, y)$ tem sua j -ésima componente dada por

$$[P(x, y)]_j = \max\{l_j, \min\{x_j - y_j, u_j\}\} - x_j, \quad j = 1, \dots, n.$$

Para $l \leq x^k \leq u$, o vetor $P(x^k, \nabla L^k(x^k))$ é denominado *gradiente projetado contínuo* (negativo) de $L^k(x)$ no ponto x^k . Se $P(x^k, \nabla L^k(x^k)) = 0$, temos que x^k é um ponto KKT do problema de minimizar $L^k(x)$ sujeita a $\tilde{\Omega}$. É possível demonstrar que a condição

$$\|P(x^k, \nabla L^k(x^k))\| \leq \varepsilon_k \quad (2.18)$$

implica (2.12)-(2.17). Por essa razão, a própria implementação do Algoritmo 1 para o problema (2.11) utiliza o critério (2.18) ao invés de (2.12)-(2.17) [35]. Retornaremos a discutir este critério no final do capítulo, ao sugerir uma variante sem derivadas do mesmo.

2.2 Lagrangiano Aumentado sem Derivadas

O objetivo deste trabalho é introduzir um algoritmo Lagrangiano Aumentado para o problema (2.2) baseado no que foi apresentado na seção anterior. E é nesta seção que discutiremos toda a estrutura desse novo algoritmo, e demonstraremos os motivos pelos quais todos os bons resultados teóricos de [3] se conservam.

2.2.1 Problemas com Caixas no Nível Inferior

Inicialmente, focaremos nossa atenção no problema (2.11), onde

$$\Omega = \{x \in \mathbb{R}^n \mid h(x) = 0, g(x) \leq 0\} \text{ e } \tilde{\Omega} = \{x \in \mathbb{R}^n \mid l \leq x \leq u\}.$$

Analisando o Algoritmo 1, notamos que o único momento onde derivadas são utilizadas é no critério de parada para o subproblema, que no caso do problema (2.11) é representado pelas condições (2.12)-(2.17). Fica claro que as modificações no Algoritmo 1 para torná-lo sem derivadas devem atingir (ao menos) tais condições. Tendo isso em mente, nos propusemos a desenvolver um critério de parada sem derivadas para o subproblema.

O critério de parada ideal é sem derivadas. Se possível, testaríamos para um candidato a minimizador se todos os pontos factíveis em uma vizinhança sua têm valor não menor de função objetivo, o que é a própria função de minimizador local. Por razões óbvias, esse critério de parada é impraticável. O critério de parada que propomos é, sob certos aspectos, uma versão mais fraca do critério ideal. Suponhamos que vamos testar se um ponto x^k o satisfaz. Ao invés de exigir que $L^k(x)$ não seja menor que $L^k(x^k)$ para todo x dentro de uma bola de raio δ_k , checaremos se para uma tolerância $\delta_k > 0$ escolhida

$$\begin{aligned} L^k(x^k) &\leq L^k(x^k + \delta_k d), \\ \forall d \in D \text{ tal que } l &\leq x^k + \delta_k d \leq u, \end{aligned} \tag{2.19}$$

onde $D = \{e^1, -e^1, \dots, e^n, -e^n\}$. Em palavras, o critério acima é satisfeito por um ponto x^k para o qual L^k não decresce em pontos y que distam δ_k de x^k e estão posicionados de modo que $(y - x^k)/\delta_k$ ou $(x^k - y)/\delta_k$ seja uma direção canônica. Notemos que a condição vale apenas para os pontos $x^k + \delta_k d$ factíveis em relação ao conjunto $\tilde{\Omega}$. A figura abaixo nos ajuda a compreender o que acontece no caso $n = 2$.

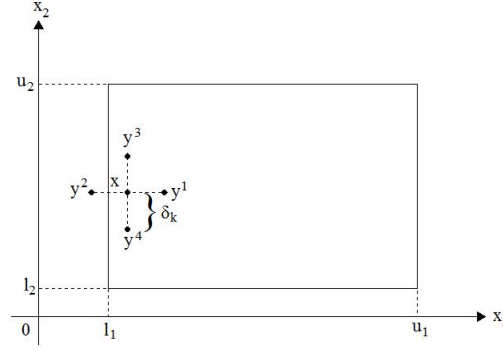


Figura 2.1: Critério de parada em \mathbb{R}^2 .

Os pontos y^1, y^2, y^3 e y^4 são os pontos do tipo $x + \delta d$. O ponto x satisfaz (2.19) se $L^k(x) \leq L^k(y^1)$, $L^k(x) \leq L^k(y^3)$ e $L^k(x) \leq L^k(y^4)$. O ponto y^2 não precisa ser analisado por não pertencer à caixa.

Este critério de parada possui a conveniente propriedade de ser sempre satisfeito por algum ponto no interior da caixa, que pode ser encontrado em um número finito de iterações internas. Suponha que um determinado ponto \bar{x}^k não satisfaz o critério (2.19). Como $\tilde{\Omega}$ é limitado, existem finitos pontos interiores à caixa posicionados sobre a malha de tamanho δ_k que têm \bar{x}^k como nó. Entre esses pontos, ao menos o que possui o menor valor de L^k satisfaz o critério de parada. Assim sendo, podemos proceder visitando pontos na malha até encontrar o ponto desejado. Isso é muito similar ao que é feito nos métodos de busca coordenada [21, 27]. Estar associado a algoritmos existentes é inclusive um pré-requisito para candidatos a critério de parada, pois é o que garante sua aplicabilidade. Evidentemente, não apenas algoritmos baseados em busca coordenada podem ser usados na resolução dos subproblemas. Qualquer outro método capaz de encontrar um ponto satisfazendo (2.19) pode ser utilizado. Se o usuário optar por um algoritmo onde não há garantias de cumprimento do critério, este pode ser sucedido, por exemplo, por passos de busca coordenada. Se o algoritmo escolhido retornar um ponto que satisfaz um critério de parada que lhe é mais natural, é esperado que esse ponto não esteja muito longe de satisfazer (2.19), uma vez que ambos critérios medem, em última instância, estacionaridade. Outro ponto positivo do critério apresentado é sua fácil compreensão, oriunda de uma motivação geométrica bastante intuitiva.

Munidos de um critério de parada que não utiliza derivadas, estamos em condições de propor um algoritmo Lagrangiano Aumentado sem derivadas. A única modificação feita em relação ao Algoritmo 1 foi no Passo 2 deste.

Algoritmo 2 (Lagrangiano Aumentado sem derivadas para restrições de caixa no nível inferior)

Performar os mesmos passos do Algoritmo 1, exceto pelo passo 2, que toma a forma:

Passo 2: *Resolução do subproblema*

Encontrar $x^k \in \tilde{\Omega}$ tal que

$$L^k(x^k) \leq L^k(x^k + \delta_k d), \forall d \in D \mid l \leq x^k + \delta_k d \leq u,$$

para algum $\delta_k > 0$ tal que $\delta_k \rho_k \leq \varepsilon_k$.

É importante destacar que, para que atinjamos os resultados teóricos pretendidos, necessitamos que a sequência de tolerâncias $\{\delta_k\}$ satisfaça a propriedade $\delta_k \rho_k \rightarrow 0$, o que é atingido no algoritmo impondo a condição $\delta_k \leq \varepsilon_k / \rho_k$. Uma análise superficial poderia levar à equivocada conclusão de que essa condição é muito forte se comparada à condição baseada em derivadas, onde é apenas requerido que $\varepsilon_k \rightarrow 0$. No entanto, naquele contexto, o gradiente projetado contínuo da função Lagrangiano Aumentado deve ter módulo menor que ε_k , e tal quantidade depende fortemente de ρ_k , o que torna aquela condição igualmente severa.

Apesar das modificações no algoritmo, somos capazes de recuperar todos os resultados teóricos de [3]. Para atingir esse objetivo de maneira bastante direta, podemos verificar que qualquer sequência $\{x^k, \lambda^k, \mu^k, \rho_k\}$ gerada pelo Algoritmo 2 pode ser gerada pelo Algoritmo 1 mediante uma escolha adequada de dados iniciais. Uma maneira razoável de demonstrar isso é verificando que, sob certas condições, o critério de parada (2.19) implica as condições (2.12)-(2.17). Os lemas abaixo têm papel fundamental nessa empreitada.

Lema 1 *Seja $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ uma função diferenciável. Se para algum inteiro $i \in [1, n]$ $\phi(x) \leq \phi(x + \Delta e^i)$ com $\Delta > 0$, então existe $c = x + \xi e^i$, $\xi \in [0, \Delta]$ tal que $\nabla \phi(c)^T e^i \geq 0$.*

Demonstração: Pelo Teorema do Valor Médio, existe $\xi \in [0, \Delta]$ tal que

$$\nabla \phi(x + \xi e^i)^T e^i = \frac{\phi(x + \Delta e^i) - \phi(x)}{\Delta}.$$

Mas, por hipótese, $\phi(x + \Delta e^i) - \phi(x) \geq 0$, de onde obtemos a tese. \square

Lema 2 *Seja $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ uma função diferenciável. Se para algum $i \in [1, n]$ $\phi(x) \leq \phi(x - \Delta e^i)$ com $\Delta > 0$, então existe $c = x + \xi e^i$, $\xi \in [-\Delta, 0]$ tal que $\nabla \phi(c)^T e^i \leq 0$.*

Demonstração: Seguir a demonstração do lema anterior, atento às trocas de sinal. \square

Lema 3 *Seja $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ uma função continuamente diferenciável. Se para algum inteiro $i \in [1, n]$ $\phi(x) \leq \phi(x - \Delta^- e^i)$ e $\phi(x) \leq \phi(x + \Delta^+ e^i)$ com $\Delta^-, \Delta^+ > 0$, então existe $c = x + \xi e^i$, $\xi \in [-\Delta^-, \Delta^+]$ tal que $\nabla \phi(c)^T e^i = 0$.*

Demonstração: Pelos lemas 1 e 2, existem $c^1 = x + \xi_1 e^i$, $\xi_1 \in [0, \Delta^+]$ e $c^2 = x + \xi_2 e^i$, $\xi_2 \in [-\Delta^-, 0]$ tais que $\nabla \phi(c^1)^T e^i \geq 0$ e $\nabla \phi(c^2)^T e^i \leq 0$. O Teorema do Valor Intermediário aplicado a c^1 e c^2 trazem a tese do lema. \square

Para atingir os resultados teóricos que desejamos, algumas hipóteses sobre as funções que definem o problema e seus gradientes precisam ser feitas. Isso é conseguido impondo que os pontos envolvidos estejam dentro de um conjunto com propriedades convenientes. Tais propriedades são sintetizadas pela *Condição 1*. Um conjunto S a satisfaz se:

Condição 1:

- ∇f é Lipschitz contínua em S ,
- Para todos os índices $i = 1, \dots, m$, h_i é limitada e ∇h_i é limitada e Lipschitz contínua em S ,
- Para todos os índices $i = 1, \dots, p$, g_i é limitada e ∇g_i é limitada e Lipschitz contínua em S .

Uma condição suficiente para que a Condição 1 seja satisfeita para o conjunto $\tilde{\Omega}$ é representada pela Hipótese 8, que assumiremos válida por toda esta seção:

Hipótese 8: As funções f, g_i $i = 1, \dots, p$ e h_i , $i = 1, \dots, m$ são diferenciáveis e possuem derivadas Lipschitz no conjunto $\tilde{\Omega}$.

O lema abaixo mostra que se dois pontos estão próximos, os gradientes $\nabla L^k(x)$ nesses pontos devem estar proporcionalmente próximos.

Lema 4 *Suponha que $\|x^k - c^k\| \leq \delta_k$, com $\delta_k \leq \varepsilon_k / \rho_k$, para algum c^k e que as sequências $\{x^k\}$ e $\{c^k\}$ estão dentro de um conjunto que satisfaz a Condição 1. Então, para qualquer índice $j = 1, \dots, n$, existe um escalar M_j independente de k tal que*

$$|[\nabla L^k(x^k)]_j - [\nabla L^k(c^k)]_j| \leq M_j \varepsilon_k.$$

Demonstração: Escrevendo o lado esquerdo da expressão acima em termos das funções f, g e h e usando a desigualdade triangular obtemos, após alguma manipulação,

$$\begin{aligned}
& |[\nabla L(x^k)]_j - [\nabla L(c^k)]_j| \leq |[\nabla f(x^k)]_j - [\nabla f(c^k)]_j| + \\
& \sum_{i=1}^m |\bar{\lambda}_i^k + \rho_k h_i(x^k)| |[\nabla h_i(x^k)]_j - [\nabla h_i(c^k)]_j| + \rho_k \sum_{i=1}^m |h_i(x^k) - h_i(c^k)| |[\nabla h_i(c^k)]_j| + \\
& \sum_{i=1}^p \max\{0, \bar{\mu}_i^k + \rho_k g_i(x^k)\} |[\nabla g_i(x^k)]_j - [\nabla g_i(c^k)]_j| + \\
& \sum_{i=1}^p |\max\{0, \bar{\mu}_i^k + \rho_k g_i(x^k)\} - \max\{0, \bar{\mu}_i^k + \rho_k g_i(c^k)\}| |[\nabla g_i(c^k)]_j|.
\end{aligned}$$

Sabemos que para $a, b \in \mathbb{R}$, $|\max\{0, a\} - \max\{0, b\}| \leq |a - b|$. Como os vetores $\bar{\lambda}^k$ e $\bar{\mu}^k$ têm suas normas limitadas por definição, x^k e c^k pertencem a um conjunto satisfazendo a Condição 1 e $\rho_k^{-1} \leq \rho_1^{-1}$, concluímos que há um $\bar{M} \in \mathbb{R}$ tal que

$$\begin{aligned}
& |[\nabla L(x^k)]_j - [\nabla L(c^k)]_j| \leq |[\nabla f(x^k)]_j - [\nabla f(c^k)]_j| + \\
& \rho_k \bar{M} \sum_{i=1}^m |[\nabla h_i(x^k)]_j - [\nabla h_i(c^k)]_j| + \rho_k \bar{M} \sum_{i=1}^m |h_i(x^k) - h_i(c^k)| + \\
& \rho_k \bar{M} \sum_{i=1}^p |[\nabla g_i(x^k)]_j - [\nabla g_i(c^k)]_j| + \rho_k \bar{M} \sum_{i=1}^p |g_i(x^k) - g_i(c^k)|.
\end{aligned}$$

Como todas as funções são Lipschitz em um conjunto contendo c^k e x^k , vemos facilmente que existe um escalar M_j tal que

$$|[\nabla L(x^k)]_j - [\nabla L(c^k)]_j| \leq M_j \rho_k \|x^k - c^k\| \leq M_j \rho_k \delta_k \leq M_j \varepsilon_k,$$

o que conclui a demonstração. □

A partir deste ponto, suporemos que a hipótese abaixo é satisfeita:

Hipótese 9: Para todo $k \in \mathbb{N}$, $\delta_k < \frac{1}{2} \min_{j=1, \dots, n} \{u_j - l_j\}$.

A Hipótese 9 implica que $x^k + \delta_k d^i \in \tilde{\Omega}$ para algum $d^i \in \{-e^i, e^i\}$, $i = 1, \dots, n$. Em outras palavras, garante que para qualquer $x^k \in \tilde{\Omega}$, os pontos $x^k + \delta_k e^i$ e $x^k - \delta_k e^i$ não podem estar simultaneamente fora da caixa, de forma que ao menos um deles é testado no critério de

parada (2.19). A condição acima não é de forma alguma restritiva, pois é automaticamente satisfeita para k suficientemente grande, uma vez que $\delta_k \rightarrow 0$. O lema abaixo estabelece a relação entre os critérios de parada sem derivadas e original.

Lema 5 *Existe um escalar M independente de k tal que se x^k verifica a condição (2.19) com tolerância δ_k então também verifica as condições (2.12)-(2.17) com tolerância $M\varepsilon_k$.*

Demonstração: Consideremos a j -ésima componente de x^k . Como qualquer ponto gerado pelo Algoritmo 2 é factível com relação às restrições de caixa, as condições (2.13) e (2.14) são sempre satisfeitas. Definamos $M = \max\{\rho_1^{-1}, M_1, \dots, M_n\}$, com M_1, \dots, M_n dados como no Lema 4. Uma vez que a Hipótese 9 é satisfeita, dividimos nossa análise nas três situações possíveis:

1. $l_j > x_j^k - \delta_k$ e $x_j^k + \delta_k \leq u_j$.

Neste caso temos, pelo critério de parada (2.19), que $L^k(x^k) \leq L^k(x^k + \delta_k e^j)$. Então, pelo Lema 1, existe um ponto $c^{jk} \in \tilde{\Omega}$ tal que $[\nabla L^k(c^{jk})]_j \geq 0$, com $\|x^k - c^{jk}\| \leq \delta_k$. Definamos $\alpha_j^k = [\nabla L^k(c^{jk})]_j$ e $\beta_j^k = 0$, de modo que (2.15) e (2.17) são satisfeitos. Como $l_j > x_j^k - \delta_k$, vemos que $l_j - x_j^k > -\delta_k \geq -\varepsilon_k/\rho_k \geq -M\varepsilon_k$, de onde obtemos (2.16).

Temos que

$$|[\nabla L^k(x^k)]_j - \alpha_j^k + \beta_j^k| = |[\nabla L^k(x^k)]_j - [\nabla L^k(c^{jk})]_j|.$$

O Lema 4 nos permite concluir que

$$|[\nabla L^k(x^k)]_j - \alpha_j^k + \beta_j^k| \leq M\varepsilon_k,$$

por onde (2.12) é verificado.

2. $l_j \leq x_j^k - \delta_k$ e $x_j^k + \delta_k \leq u_j$.

Neste caso temos, pelo critério de parada (2.19), que $L^k(x^k) \leq L^k(x^k + \delta_k e^j)$ e $L^k(x^k) \leq L^k(x^k - \delta_k e^j)$. Então, pelo Lema 3, existe um ponto $c^{jk} \in \tilde{\Omega}$ tal que $[\nabla L^k(c^{jk})]_j = 0$, com $\|x^k - c^{jk}\| \leq \delta_k$. Definamos $\alpha_j^k = \beta_j^k = 0$, então (2.15)-(2.17) são satisfeitos. A condição (2.12) é verificada de maneira idêntica à do caso 1.

3. $l_j \leq x_j^k - \delta_k$ e $x_j^k + \delta_k > u_j$.

Neste caso temos, pelo critério de parada (2.19), que $L^k(x^k) \leq L^k(x^k - \delta_k e^j)$. Então, pelo Lema 2, existe um ponto $c^{jk} \in \tilde{\Omega}$ tal que $[\nabla L^k(c^{jk})]_j \leq 0$, com $\|x^k - c^{jk}\| \leq \delta_k$. Definamos $\alpha_j^k = 0$ e $\beta_j^k = -[\nabla L^k(c^{jk})]_j$, de modo que (2.15) e (2.16) são satisfeitos. Como $x_j^k + \delta_k > u_j$, vemos que $x_j^k - u_j > -\delta_k \geq -\varepsilon_k/\rho_k \geq -M\varepsilon_k$, de onde obtemos (2.17). A condição (2.12) é

verificada de maneira idêntica à do caso 1.

Analizando todos os índices $j = 1, \dots, n$ obtemos a tese do lema. \square

O Lema 5 tem uma consequência muito importante. Uma sequência $\{x^k\}$ gerada pelo Algoritmo 2 com sequência de tolerâncias $\{\delta_k\}$ poderia ser gerada pelo Algoritmo 1 com sequência de tolerâncias $\{M\varepsilon_k\}$. Isso é verdade se escolhermos os mesmos dados e parâmetros iniciais para ambos algoritmos, dado que as regras para atualização de λ^k, μ^k e ρ_k são as mesmas. Isso permite que os resultados teóricos de [3], ou seja, os Teoremas 1, 2 e 3, possam ser recuperados para o Algoritmo 2. Reescrevemo-los abaixo, lembrando que algumas simplificações nos enunciados advêm do fato de, por ora, termos apenas caixas no nível inferior.

Teorema 4 *Seja x^* ponto limite de uma sequência $\{x^k\}$ gerada pelo Algoritmo 2. Se a sequência de parâmetros de penalidade $\{\rho_k\}$ é limitada, então x^* é factível. Caso contrário, x^* é um ponto KKT do problema*

$$\text{minimizar } \sum_{i=1}^m [h(x)]_i^2 + \sum_{i=1}^p \max\{0, [g(x)]_i\}^2 \text{ sujeita a } x \in \tilde{\Omega}.$$

Teorema 5 *Seja x^* um ponto limite factível de uma sequência $\{x^k\}$ gerada pelo Algoritmo 2. Se x^* satisfaz a condição CPLD em relação a $\Omega \cap \tilde{\Omega}$, então x^* é ponto KKT do problema (2.2). Além disso, se x^* satisfaz a condição de Mangasarian-Fromovitz e $\{x^k\}_{k \in K}$ é uma subsequência que converge a x^* então o conjunto $\{\|\lambda^{k+1}\|, \|\mu^{k+1}\|, \|\alpha^k\|, \|\beta^k\|\}_{k \in K}$ é limitado.*

Teorema 6 *Suponha que as Hipóteses 1-7 são satisfeitas. Suponha também que existe uma sequência $\eta_k \rightarrow 0$ tal que $\delta_k \leq \rho_k^{-1} \eta_k \max\{\|h(x^k)\|, \|\sigma^k\|\}$ para todo $k \in \mathbb{N}$. Então a sequência de parâmetros de penalidade é limitada.*

Em relação ao Teorema 4, devemos lembrar que o caso em que x^* não satisfaz a condição CPLD em relação a $\tilde{\Omega}$ não precisa ser considerado como no Teorema 1, pois restrições de caixa sempre verificam a CPLD. No Teorema 6, substituímos a condição $\varepsilon_k \leq \eta_k \max\{\|h(x^k)\|, \|\sigma^k\|\}$ por $\delta_k \leq \rho_k^{-1} \eta_k \max\{\|h(x^k)\|, \|\sigma^k\|\}$. Isso se deve ao fato de a sequência $\{\varepsilon_k\}$ possuir significados diferentes para os Algoritmos 1 e 2. No Algoritmo 1, $\{\varepsilon_k\}$ é usada diretamente no critério de parada, enquanto que no Algoritmo 2, é apenas uma sequência que controla a convergência da sequência $\{\delta_k\}$ através da relação $\delta_k \rho_k \leq \varepsilon_k$. Vejamos que a nova condição sobre δ_k no Teorema 6 implica a condição sobre ε_k no Teorema 3. Uma sequência $\{\delta_k\}$ gerada pelo Algoritmo 2 poderia ter sido obtida a partir de uma sequência $\{\varepsilon_k\}$ tal que seus

termos satisfazem $\varepsilon_k = \delta_k \rho_k$. Segundo nossos resultados, a sequência $\{x^k\}$ correspondente poderia ter sido gerada pelo Algoritmo 1 com sequência de tolerâncias $\{\bar{\varepsilon}_k\} = \{M\varepsilon_k\}$. Mas se $\delta_k \leq \rho_k^{-1} \eta_k \max\{\|h(x^k)\|, \|\sigma^k\|\}$, então $\bar{\varepsilon}_k \leq \bar{\eta}_k \max\{\|h(x^k)\|, \|\sigma^k\|\}$, com $\bar{\eta}_k = M\eta_k \rightarrow 0$, o que implica que a hipótese sobre ε_k do Teorema 3 é satisfeita.

Para concluir a teoria para restrições de caixa no nível inferior, é relevante observar que há um trabalho semelhante, onde Lewis e Torczon também propõem um algoritmo tipo Lagrangiano Aumentado sem derivadas para o problema (2.11) [20]. Apesar de nossos resultados terem sido alcançados de maneira independente, há algumas semelhanças, especialmente quanto à motivação. O algoritmo proposto também é baseado em um método Lagrangiano Aumentado já existente, devido a Conn, Gould e Toint [10]. As diferenças existentes entre [3] e [10] são também observadas se compararmos os trabalhos correspondentes em otimização sem derivadas. Elas residem especialmente no fato de que, em [20], as restrições de desigualdade não são tratadas diretamente, mas pela introdução de variáveis de folga. Além disso, a condição de qualificação utilizada nas demonstrações é a regularidade, que é mais forte que a condição CPLD por nós usada. Na resolução dos subproblemas no algoritmo em questão, é sempre aplicada busca padrão com direções coordenadas (a menos de escalamento das direções), sendo que nós defendemos a utilização de qualquer algoritmo sem derivadas que satisfaça o critério de parada. Por fim, as funções envolvidas precisam ser duas vezes continuamente diferenciáveis, e não apenas ter gradientes Lipschitz como em nossa teoria. Vale também citar o trabalho para restrições lineares de desigualdade no nível inferior [17], que é baseado no correspondente com derivadas [9]. Ao que consta, os algoritmos encontrados em [17, 20] nunca foram implementados, de modo que nunca foram testados na prática.

2.2.2 Problemas com Restrições Gerais no Nível Inferior

Nesta seção, apresentamos um algoritmo sem derivadas para o problema (2.4), com dois níveis de restrições de igualdade e desigualdade. Até onde sabemos, é o primeiro trabalho com esse intuito. Novamente, modificamos o Algoritmo 1, a fim de que a condição de parada para os subproblemas não mais utilizem gradientes.

Seja $\phi^k(x)$ um vetor que aproxima $\nabla^k L(x)$, com j -ésima componente dada por

$$[\phi^k(x)]_j = [\phi^k(x, \delta^{+k}, \delta^{-k})]_j = \frac{L^k(x + \delta_j^{+k} e^j) - L^k(x + \delta_j^{-k} e^j)}{\delta_j^{+k} - \delta_j^{-k}},$$

onde $-\delta_k \leq \delta_j^{-k} \leq 0 \leq \delta_j^{+k} \leq \delta_k$ e $\delta_j^{-k} \neq \delta_j^{+k}$, $i = 1, \dots, n$. Nesta seção, δ_k é um limitante para o passo da aproximação, admitindo uma função diferente em relação ao Algoritmo 2,

porém semelhante do ponto de vista da teoria como veremos posteriormente. É fácil verificar que escolhas adequadas para δ^{+k} e δ^{-k} podem fazer com que $\phi^k(x)$ seja, por exemplo, uma aproximação por diferenças finitas centradas, avançadas ou atrasadas de $\nabla L^k(x)$. Propomos um critério de parada sem derivadas para o subproblema, definido pelas condições abaixo:

$$\|\phi^k(x^k) + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k \nabla \tilde{h}_i(x^k) + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k \nabla \tilde{g}_i(x^k)\| \leq \varepsilon_k, \quad (2.20)$$

$$\tilde{\mu}_i^k \geq 0 \text{ e } \tilde{g}_i(x^k) \leq \varepsilon_k \text{ para } i = 1, \dots, \tilde{p}, \quad (2.21)$$

$$\tilde{g}_i(x^k) < -\varepsilon_k \Rightarrow \tilde{\mu}_i^k = 0, \text{ para } i = 1, \dots, \tilde{p}, \quad (2.22)$$

$$\|\tilde{h}(x^k)\| \leq \varepsilon_k. \quad (2.23)$$

As condições acima são exatamente iguais ao critério de parada do Algoritmo 1, mas com a aproximação $\phi^k(x)$ substituindo $\nabla L^k(x)$ em (2.5). Com essa simples modificação, estamos em condição de propor um algoritmo Lagrangiano Aumentado sem derivadas para o problema (2.4).

Algoritmo 3 (Lagrangiano Aumentado sem derivadas para restrições gerais)

Performar os mesmos passos do Algoritmo 1, exceto pelo passo 2, que toma a forma:

Passo 2: *Resolução do subproblema*

Encontrar $x^k \in \mathbb{R}^n$ tal que as condições (2.20)-(2.23) sejam satisfeitas, para algum $\delta_k > 0$ tal que $\delta_k \rho_k \leq \varepsilon_k$.

Para provar resultados de convergência para o Algoritmo 3, a Condição 1 deve ser satisfeita para um conjunto que contenha tanto x^k como pontos $c^{jk} \in [x^k + \delta_j^{-k} e^j, x^k + \delta_j^{+k} e^j]$, $j = 1, \dots, n$, sendo que esses últimos pontos têm importância apenas teórica. Cumpriremos essa exigência supondo que a Hipótese 10 abaixo é sempre satisfeita.

Hipótese 10: Para todos os pontos x^k gerados pelo Algoritmo 3, os segmentos de reta $[x^k + \delta_j^{-k} e^j, x^k + \delta_j^{+k} e^j]$ estão contidos em um conjunto que satisfaz a Condição 1, onde $\{x^k + \delta_j^{-k} e^j\}$ e $\{x^k + \delta_j^{+k} e^j\}$ são os pontos utilizados para que os gradientes sejam aproximados.

Uma pergunta razoável a se fazer é se existem pontos que satisfazem (2.20)-(2.23). O lema a seguir garante que este é o caso, desde que sejamos capazes de encontrar pontos satisfazendo (2.5)-(2.8). Consideraremos o escalar $\overline{M} = \max\{1, M_1, \dots, M_n\}$, onde M_j é

dado pelo Lema 4.

Lema 6 *Suponha que exista um ponto x^k que satisfaz as condições (2.5)-(2.8) com tolerância $\bar{\varepsilon}_k = \varepsilon_k/2\bar{M}$. Então se $\delta_k \rho_k \leq \bar{\varepsilon}_k$, x^k verifica as condições (2.20)-(2.23) com tolerância ε_k e mesmas estimativas de multiplicadores.*

Demonstração: As condições (2.21) e (2.23) são facilmente verificadas com tolerância ε_k , posto que $\bar{M} \geq 1$ e os multiplicadores são os mesmos por definição. Se $\tilde{g}_j(x^k) < -\varepsilon_k$ para algum índice j , então $\tilde{g}_j(x^k) < -\bar{\varepsilon}_k$, logo por hipótese $\tilde{\mu}_j^k = 0$, o que prova que (2.22) é atingida.

Para provar que (2.20) é verificada, consideremos para cada índice $j = 1, \dots, n$ a quantidade abaixo

$$|[\phi^k(x^k)]_j + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k [\nabla \tilde{h}_i(x^k)]_j + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k [\nabla \tilde{g}_i(x^k)]_j|.$$

Após alguma manipulação, vemos que

$$|[\phi^k(x^k)]_j + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k [\nabla \tilde{h}_i(x^k)]_j + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k [\nabla \tilde{g}_i(x^k)]_j| \leq$$

$$|[\phi^k(x^k)]_j - [\nabla L^k(x^k)]_j| + |[\nabla L^k(x^k)]_j + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k [\nabla \tilde{h}_i(x^k)]_j + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k [\nabla \tilde{g}_i(x^k)]_j|.$$

Pela definição de $\phi^k(x^k)$ podemos ver, usando o Teorema do Valor Médio, que $[\phi^k(x^k)]_j = [\nabla L^k(c^{jk})]_j$ para algum $c^{jk} \in [x^k + \delta_j^{-k} e^j, x^k + \delta_j^{+k} e^j]$. Como $\|x^k - c^{jk}\| \leq \delta_k$, considerando o Lema 4 e as hipóteses deste lema concluímos que

$$|[\phi^k(x^k)]_j + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k [\nabla \tilde{h}_i(x^k)]_j + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k [\nabla \tilde{g}_i(x^k)]_j| \leq M_j \bar{\varepsilon}_k + \bar{\varepsilon}_k \leq \varepsilon_k,$$

o que prova a tese. \square

O lema 6 estabelece que é sempre possível encontrar um ponto satisfazendo (2.20)-(2.23), para uma escolha convenientemente pequena de passos nas aproximações dos gradientes e desde que (2.5)-(2.8) possa ser atingido com uma tolerância suficientemente pequena, o que não é uma hipótese impossível de ser atingida. De fato, qualquer ponto KKT do subproblema a satisfaz para qualquer tolerância.

No próximo lema, utilizaremos a quantidade $\tilde{M} = 1 + \max\{M_1, \dots, M_n\}$, onde M_j é dado pelo Lema 4.

Lema 7 *Suponha que exista um ponto x^k satisfazendo as condições (2.20)-(2.23) com tolerância ε_k e com $\delta_k \rho_k \leq \varepsilon_k$. Então x^k verifica as condições (2.5)-(2.8) com tolerância $\tilde{\varepsilon}_k = \widetilde{M} \varepsilon_k$ e mesmas estimativas de multiplicadores.*

Demonstração: As condições (2.6) e (2.8) são facilmente verificadas com tolerância $\tilde{\varepsilon}_k$, posto que $\widetilde{M} > 1$ e os multiplicadores são os mesmos por definição. Se $\tilde{g}_j(x^k) < -\tilde{\varepsilon}_k$ para algum índice j , então $\tilde{g}_j(x^k) < -\varepsilon_k$, logo por hipótese $\tilde{\mu}_j^k = 0$, o que prova que (2.7) é atingida.

Para provar que (2.5) é verificada, consideremos para cada índice $j = 1, \dots, n$ a quantidade abaixo

$$|[\nabla L^k(x^k)]_j + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k [\nabla \tilde{h}_i(x^k)]_j + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k [\nabla \tilde{g}_i(x^k)]_j|.$$

Após alguma manipulação, vemos que

$$|[\nabla L^k(x^k)]_j + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k [\nabla \tilde{h}_i(x^k)]_j + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k [\nabla \tilde{g}_i(x^k)]_j| \leq$$

$$|[\nabla L^k(x^k)]_j - [\phi^k(x^k)]_j| + |[\phi^k(x^k)]_j + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k [\nabla \tilde{h}_i(x^k)]_j + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k [\nabla \tilde{g}_i(x^k)]_j|.$$

Pela definição de $\phi^k(x^k)$ podemos ver, usando o Teorema do Valor Médio, que $[\phi^k(x^k)]_j = [\nabla L^k(c^{jk})]_j$ para algum $c^{jk} \in [x^k + \delta_j^{-k} e^j, x^k + \delta_j^{+k} e^j]$. Como $\|x^k - c^{jk}\| \leq \delta_k$, considerando o Lema 4 e as hipóteses deste lema concluímos que

$$|[\nabla L^k(x^k)]_j + \sum_{i=1}^{\tilde{m}} \tilde{\lambda}_i^k [\nabla \tilde{h}_i(x^k)]_j + \sum_{i=1}^{\tilde{p}} \tilde{\mu}_i^k [\nabla \tilde{g}_i(x^k)]_j| \leq M_j \varepsilon_k + \varepsilon_k \leq \tilde{\varepsilon}_k,$$

o que prova a tese. □

Analogamente ao caso de caixas no nível inferior de restrições, o Lema 7 implica que uma sequência gerada pelo Algoritmo 3 poder ser gerada pelo Algoritmo 1, com sequência de tolerâncias $\{\tilde{\varepsilon}_k\}$ e outros parâmetros e dados iniciais inalterados. Com isso, mais uma vez recuperamos os resultados teóricos de [3], sintetizados pelos teoremas abaixo:

Teorema 7 *Seja x^* ponto limite de uma sequência $\{x^k\}$ gerada pelo Algoritmo 3. Se a sequência de parâmetros de penalidade $\{\rho_k\}$ é limitada, então x^* é factível. Caso contrário, uma das duas possibilidades é verificada:*

- x^* é um ponto KKT do problema

$$\text{minimizar } \sum_{i=1}^m h_i(x)^2 + \sum_{i=1}^p \max\{0, g_i(x)\}^2 \text{ sujeita a } x \in \tilde{\Omega};$$

- x^* não satisfaz a condição CPLD associada a $\tilde{\Omega}$.

Teorema 8 *Seja x^* um ponto limite factível de uma sequência $\{x^k\}$ gerada pelo Algoritmo 3. Se x^* satisfaz a condição CPLD em relação a $\Omega \cap \tilde{\Omega}$, então x^* é ponto KKT do problema (2.2). Além disso, se x^* satisfaz a condição de Mangasarian-Fromovitz e $\{x^k\}_{k \in K}$ é uma subsequência que converge a x^* então o conjunto $\{\|\lambda^{k+1}\|, \|\mu^{k+1}\|, \|\tilde{\lambda}^k\|, \|\tilde{\mu}^k\|\}_{k \in K}$ é limitado.*

Teorema 9 *Suponha que as Hipóteses 1-7 são satisfeitas. Suponha também que existe uma sequência $\eta_k \rightarrow 0$ tal que $\delta_k \leq \rho_k^{-1} \eta_k \max\{\|h(x^k)\|, \|\sigma^k\|\}$ para todo $k \in \mathbb{N}$. Então a sequência de parâmetros de penalidade é limitada.*

2.2.3 Gradiente Projetado Contínuo sem Derivadas

Os pacotes modernos de otimização raramente possuem subrotinas que implementem métodos sem derivadas com resultados teóricos de convergência. Usualmente, porém, é permitido aos usuários optar pela não utilização de derivadas explícitas, normalmente aproximando-as por diferenças finitas. Nesta seção abordamos uma situação hipotética onde foram escolhidas aproximações para as derivadas.

Consideremos a condição (2.18). Se substituirmos $\nabla L^k(x^k)$ por uma aproximação por diferenças finitas $\phi^k(x^k)$, o critério (2.18) se torna

$$\|P(x^k, \phi^k(x^k))\| \leq \varepsilon_k. \quad (2.24)$$

Como já discutimos, na implementação do Algoritmo 1 para problemas com restrições de caixa no nível inferior, o critério (2.18) é verificado para a parada do algoritmo que resolve o subproblema. O pacote com essa implementação é chamado ALGENCAN, e pode ser encontrado em [35]. Voltaremos a falar brevemente sobre ele no próximo capítulo. Em ALGENCAN, é possível calcular as derivadas por diferenças finitas, no caso de o usuário desconhecê-las ou simplesmente não querer implementá-las. Se assim for, o critério de parada utilizado é exatamente (2.24), com $\phi^k(x)$ sendo uma aproximação de $\nabla L^k(x)$ por diferenças finitas centradas. No lema a seguir, relacionamos o critério de parada (2.24) e as condições

(2.12)-(2.17), que no caso de derivadas aproximadas, são expressas por

$$|[\phi(x^k)]_j - \alpha_j^k + \beta_j^k| \leq \varepsilon_k, \quad (2.25)$$

$$-x_j^k + l_j \leq \varepsilon_k, \quad (2.26)$$

$$x_j^k - u_j \leq \varepsilon_k, \quad (2.27)$$

$$\alpha_j^k, \beta_j^k \geq 0, \quad (2.28)$$

$$-x_j^k + l_j < -\varepsilon_k \Rightarrow \alpha_j^k = 0 \text{ e} \quad (2.29)$$

$$x_j^k - u_j < -\varepsilon_k \Rightarrow \beta_j^k = 0. \quad (2.30)$$

Lema 8 *Seja $l \leq x^k \leq u$ tal que $\|P(x^k, \phi^k(x^k))\| \leq \varepsilon$ com $\delta_k \leq \varepsilon_k/\rho_k$. Então x^k satisfaz (2.25)-(2.30).*

Demonstração: Como $x^k \in \tilde{\Omega}$, (2.26) e (2.27) são trivialmente verificadas. Sejam α^k e β^k com componentes

$$\alpha_j^k = \begin{cases} [\phi(x^k)]_j & \text{se } x_j^k - [\phi(x^k)]_j < l_j \\ 0 & \text{caso contrário,} \end{cases} \quad (2.31)$$

$$\beta_j^k = \begin{cases} -[\phi(x^k)]_j & \text{se } x_j^k - [\phi(x^k)]_j > u_j \\ 0 & \text{caso contrário.} \end{cases} \quad (2.32)$$

Para um índice arbitrário j , analisemos os três casos possíveis:

1. $x_j^k - [\phi(x^k)]_j < l_j$. Por (2.31) e (2.32), temos que $\alpha_j^k = [\phi(x^k)]_j$ e $\beta_j^k = 0$. Como $0 \leq x_j - l_j < [\phi(x^k)]_j$, observamos que $[\phi(x^k)]_j > 0$. Então as desigualdades (2.28) e (2.30) são satisfeitas. Como $\|P(x^k, \phi(x^k))\| \leq \varepsilon_k$, concluímos que $x_j^k - l_j = |[P(x^k, \phi(x^k))]_j| \leq \varepsilon_k$, de onde a condição (2.29) é atingida. Além disso, $|[\phi(x^k)]_j - \alpha_j^k + \beta_j^k| = |[\phi(x^k)]_j - [\phi(x^k)]_j| = 0$, o que prova (2.25).

2. $l_j \leq x_j^k - [\phi(x^k)]_j \leq u_j$. Neste caso, temos por (2.31) e (2.32) que $\alpha_j = \beta_j = 0$. Então as condições (2.28), (2.29) e (2.30) são automaticamente satisfeitas. Além disso, como $\|P(x^k, \phi(x^k))\| \leq \varepsilon_k$, temos que $|[\phi(x^k)]_j| = |[P(x^k, \phi(x^k))]_j| \leq \varepsilon_k$, de onde concluímos que $|[\phi(x^k)]_j - \alpha_j + \beta_j| = |[\phi(x^k)]_j| \leq \varepsilon_k$, o que prova (2.25).

3. $x_j^k - [\phi(x^k)]_j > u_j$. Neste caso, a prova pode ser obtida de modo muito similar à discussão do caso 1.

Considerando todos os índices $j = 1, \dots, n$ obtemos a tese do lema. \square

O Lema 8 tem por efeito que se uma sequência $\{x^k\}$ é gerada por um algoritmo Lagrangiano Aumentado com a condição (2.24) como critério de parada, poderia então ter sido gerada pelo Algoritmo 3. Assim sendo, o Lema 7 pode ser aplicado e novamente podemos recuperar as garantias teóricas de [3]. Observemos que o Lema estabelece que se (2.24) é satisfeito então bons resultados podem ser alcançados, no entanto não diz sob que circunstâncias essa condição é viável, uma vez que $\phi(x^k)$ pode ser muito diferente de $\nabla L^k(x^k)$, o que poderia causar um mau comportamento do algoritmo interno. Um estudo específico de quão precisas as aproximações dos gradientes devem ser para garantir bom desempenho dos algoritmos faz-se necessário.

Capítulo 3

Experimentos Numéricos

No capítulo anterior, introduzimos um método tipo Lagrangiano Aumentado para resolver problemas de programação não-linear com restrições de igualdade e desigualdade. Demonstramos que, sob condições razoáveis, as sequências geradas pelos algoritmos propostos convergem a pontos estacionários. Neste capítulo avaliamos o Algoritmo 2 sob a ótica computacional, de modo a verificar se as expectativas promissoras criadas pelos resultados teóricos são confirmadas por um bom desempenho prático.

Os algoritmos foram implementados em Fortran e executados em um microcomputador com processador Intel Core2 Quad de 2.83Ghz e 8GB de memória RAM.

Independentemente do algoritmo escolhido para a resolução dos subproblemas, este sempre é sucedido por busca coordenada, a fim de que o critério de parada (2.25)-(2.30) se cumpra. Definidas tolerâncias δ_{opt} e ε_{fact} , o algoritmo externo encerra sua execução e declara convergência quando encontra um ponto \bar{x} que satisfaz (2.25)-(2.30) com tolerância menor ou igual a δ_{opt} e é suficientemente factível no sentido de verificar

$$\max\{\|h(\bar{x})\|, \|\sigma(\bar{x})\|\} \leq \varepsilon_{fact}.$$

A tolerância para a busca coordenada é constante para todos os subproblemas, e não uma sequência convergente a zero como nos algoritmos do capítulo anterior. Obviamente, desejamos atribuir a δ_{opt} um valor próximo de zero. Como os valores das tolerâncias foram diferentes para cada tipo de problema, serão especificados nas seções correspondentes.

O programa declara fracasso na busca pela solução em três casos:

1. quando não consegue encontrar a solução em até 50 iterações externas,
2. quando performa 9 iterações externas sem melhorar a factibilidade,

3. quando avalia a função Lagrangiano Aumentado mais de 10^6 vezes em uma única chamada do algoritmo de busca coordenada.

Cinco algoritmos internos diferentes foram testados, entretanto alguns deles apenas na abordagem de problemas específicos. Utilizamos os algoritmos de busca coordenada, BOBYQA, Nelder-Mead e NOMAD apresentados no Capítulo 1, além do algoritmo GENCAN. Falemos brevemente do critério de parada de cada um deles, e das escolhas para os parâmetros. Os valores que não forem agora citados são diferentes para cada problema, e serão por este motivo especificados em momento oportuno. O algoritmo de busca coordenada depende de três parâmetros: a tolerância δ_{opt} , o tamanho inicial da malha Δ_0 e o fator de redução α . Em BOBYQA temos o raio inicial da região de confiança ρ_{beg} , que foi fixado em $0.25 \min_i \{u_i - l_i\}$, e o limitante ρ_{end} para o raio final, sendo que a convergência é declarada quando o raio corrente é menor ou igual a ρ_{end} . Em relação ao número de pontos para a interpolação quadrática, seguimos a orientação do autor e adotamos $2n + 1$ pontos. Para o algoritmo Nelder-Mead, os valores usuais para os parâmetros foram escolhidos, e a convergência é declarada para um ponto em que a distância entre dois vértices consecutivos (segundo a ordenação corrente dos vértices) e a diferença de valores de função objetivo entre o melhor e o pior ponto não exceda ε_{NM} . Como o algoritmo é adequado apenas à minimização irrestrita, definimos que as funções envolvidas têm um valor muito grande (10^{20}) fora da caixa. GENCAN é um algoritmo para minimização em caixas baseado em gradientes espectrais [8]. É utilizado na implementação do algoritmo Lagrangiano Aumentado visto em [3], que pode ser encontrada em [35]. Seus parâmetros foram mantidos como estão em [35], apenas a tolerância ε_{gen} foi alterada. O algoritmo declara convergência quando encontra um ponto com norma infinito do gradiente projetado contínuo menor ou igual a ε_{gen} . Definimos o parâmetro $gtype = 1$ no algoritmo encontrado em [35], a fim de que os gradientes fossem computados por diferenças finitas, e o algoritmo passasse a ser sem derivadas. Por fim, todos os parâmetros de NOMAD foram mantidos como estão em [34].

É interessante notar que não há garantias de convergência para BOBYQA, GENCAN com derivadas aproximadas ou para o algoritmo de Nelder-Mead, particularmente com a modificação feita neste último para adequá-lo à minimização em caixas. Isso é irrelevante porém, considerando que as soluções geradas são corrigidas pela busca coordenada de modo a assegurar as boas propriedades teóricas. Sabendo que a garantia de convergência reside na busca coordenada, optamos por não interromper a execução do Algoritmo 1 quando os subalgoritmos fracassam em encontrar a solução por qualquer motivo. A busca coordenada é acionada normalmente nesses casos, a partir da solução (mesmo que potencialmente ruim) retornada pelos algoritmos internos.

Na primeira seção, resolvemos alguns problemas da coletânea Hock-Schittkowski [15]. Na

Seção 3.2, introduzimos um problema específico para métodos sem derivada para problemas com restrições. Tal problema tem interpretação geométrica e é de fácil implementação, em contraponto à lista de problemas extremamente complexos que são descritos na literatura para justificar o uso de métodos sem derivada. Na última seção, resolvemos um problema sem derivadas advindo da engenharia mecânica.

3.1 Exemplos de Hock-Schittkowski

Como já foi exposto no Capítulo 1, métodos sem derivada dificilmente superam em desempenho os métodos mais populares de otimização. No entanto, iniciamos os testes numéricos com problemas da coleção proposta por Hock e Schittkowski [15], que são problemas onde as derivadas estão disponíveis e por este motivo poderiam ser resolvidos com mais eficiência por métodos baseados em derivadas. A razão para tal escolha é o fato de a coletânea reunir problemas ao mesmo tempo simples, com soluções conhecidas, mas que trazem dificuldades recorrentes em otimização, como problemas mal escalados, mal condicionados ou com várias soluções locais. Não houve motivos especiais para a escolha dessa coleção em detrimento a qualquer outra que também apresentasse problemas com restrições gerais e de caixa. Essa primeira etapa de testes teve dois objetivos bem claros. O primeiro foi mostrar que o algoritmo funciona na prática. O segundo foi investigar, mesmo que de maneira muito superficial, qual seria uma escolha adequada de parâmetros algorítmicos, visando a resolução dos problemas mais complexos das seções seguintes.

Apesar de serem 119 os problemas contidos na coletânea, optamos por resolver apenas os 47 que possuem restrições simultâneas gerais e de caixa. A dimensão destes varia entre 2 e 16, enquanto o número de restrições está entre 1 e 38, excedendo 10 restrições em apenas 5 casos.

A tolerância ε_{fact} escolhida para a factibilidade foi de 10^{-5} . Iniciamos os testes aplicando apenas busca coordenada aos subproblemas. Investigamos qual valor para δ_{opt} seria adequado, de modo que a maior quantidade possível de problemas fosse resolvida com sucesso, mas efetuando um número reduzido de avaliações de função. Realizando testes para alguns valores diferentes de δ_{opt} , e analisando robustez e eficiência, optamos por adotar $\delta_{opt} = 10^{-5}$. Uma vez fixados ε_{fact} e δ_{opt} , variamos separadamente α e Δ_0 , e nos decidimos pelos valores $\alpha = 0.5$ e $\Delta_0 = 1$. Por fim, questionamos a maneira como as direções são exploradas. Implementamos cinco versões do algoritmo:

1. uma versão que testa todas as direções $\{e^1, \dots, e^n, -e^1, \dots, -e^n\}$ e dá um passo naquela que tem menor valor de função objetivo (desde que haja melhora em relação à iteração anterior);

2. uma versão que testa ciclicamente as direções na ordem $\{e^1, \dots, e^n, -e^1, \dots, -e^n\}$, dando um passo sempre que houver melhora em relação à iteração anterior. Por exemplo, testa a direção e^1 , se há melhora dá um passo, caso contrário não altera o ponto corrente, depois testa e^2 , novamente dando um passo se houver decréscimo, depois a direção e^3 , e assim procedendo até a direção $-e^n$, depois avaliando novamente o que ocorre na direção e^1 , depois e^2 e assim sucessivamente;
3. uma versão como a anterior, mas que tenta dar o máximo de passos em uma direção onde há decréscimo, só então passando para a direção seguinte;
4. uma versão que explora as direções de descida e testa ciclicamente as direções na ordem $\{e^1, -e^1, \dots, e^n, -e^n\}$, sendo que $-e^j$ só é testada se e^j não é de descida;
5. uma versão como a anterior, mas que investiga 15 pontos aleatórios na malha cada vez que um ciclo de direções é testado, na tentativa de escapar de minimizadores não globais.

A versão de melhor desempenho foi a quarta, sendo então a escolhida para todos os problemas deste capítulo, inclusive para os passos de busca coordenada dados após os outros subalgoritmos. A quinta versão, apesar de ter o pior desempenho em termos de quantidade de avaliações de função, será também utilizada nas seções posteriores. Nas tabelas e sempre que conveniente, denotaremos a quarta opção como BC (Busca Coordenada) e a quinta como BCrand.

Dando sequência à primeira etapa de experimentos, testamos mais três subalgoritmos: GENCAN, algoritmo de Nelder-Mead (que denotaremos NM) e BOBYQA. Consideremos que em uma chamada de BOBYQA um ponto x_B é retornado e será o ponto inicial para a busca coordenada. Uma pergunta perfeitamente razoável é o quão próximo x_B está de satisfazer as condições (2.25)-(2.30). Os testes desta seção tentam jogar um pouco de luz sobre esta questão, sem a menor pretensão de respondê-la de forma completa e definitiva. Desejávamos que os algoritmos de Nelder-Mead, BOBYQA e GENCAN retornassem um ponto que estivesse muito próximo de satisfazer (2.25)-(2.30), para que a importância da busca coordenada que sucedia tais algoritmos não fosse excessiva. Além disso, buscávamos o menor número de avaliações de função possível. Para tanto, testamos algumas combinações de tolerâncias para os 3 algoritmos e passo inicial Δ_0 da busca coordenada. Se Δ_0 fosse muito grande, havia a possibilidade de nos afastarmos demasiadamente do ponto retornado pelo subalgoritmo. Por outro lado, se Δ_0 fosse muito pequeno, poderiam ser necessários muitos passos de busca coordenada até que um ponto que verificasse (2.25)-(2.30) fosse atingido. Para avaliar se uma escolha de parâmetros é melhor do que outra, novamente consideramos a

quantidade de problemas resolvidos com sucesso e a de avaliações de função realizadas, ainda com $\varepsilon_{fact} = \delta_{opt} = 10^{-5}$. Para BOBYQA, escolhemos $\rho_{end} = 10^{-6}$ e $\Delta_0^{BOBYQA} = 0.01$. Para NM, $\varepsilon_{NM} = 10^{-5}$ e $\Delta_0^{NM} = 0.05$. Por fim, para GENCAN, o melhor valor de ε_{gen} foi 10^{-5} , e de Δ_0^{GENCAN} , 10^{-5} . O valor de Δ_0 para GENCAN mostra um fato interessante. Mesmo usando derivadas aproximadas, o algoritmo teve um comportamento tão satisfatório que a busca coordenada poderia iniciar imediatamente com o menor passo possível, ou seja, com $\Delta_0^{GENCAN} = \delta_{opt}$.

A tabela (3.1) dá uma idéia de quantas avaliações de função (feval) foram necessárias para se resolver cada problema com os 5 algoritmos internos testados. A primeira linha representa a porcentagem de problemas que levaram de 10 a 100 avaliações de função para serem solucionados, a segunda mostra a porcentagem de problemas que precisaram de 100 a 1000 avaliações de função e assim por diante. As três últimas linhas mostram a proporção de fracassos e quais foram eles. É importante observar que, nos casos onde houve sucesso, a solução indicada em [15] foi encontrada.

feval	BC	BCrand	BOBYQA	GENCAN	NM
10^1 a 10^2	6.4 %	0.0 %	4.3 %	48.9 %	0.0 %
10^2 a 10^3	17.0 %	14.9 %	34.0 %	36.2 %	21.3 %
10^3 a 10^4	27.7 %	31.9 %	29.8 %	6.4 %	34.0 %
10^4 a 10^5	19.1 %	14.9 %	14.9 %	8.5 %	14.9 %
10^5 a 10^6	12.8 %	14.9 %	8.5 %	0.0 %	10.6 %
10^6 a 10^7	0.0 %	6.4 %	0.0 %	0.0 %	12.8 %
fracasso 1	10.6 %	10.6 %	0.0 %	0.0 %	0.0 %
fracasso 2	0.0 %	0.0 %	2.1 %	0.0 %	0.0 %
fracasso 3	6.4 %	6.4 %	6.4 %	0.0 %	6.4 %

Tabela 3.1: Avaliações de função para 47 problemas de Hock-Schittkowski.

A tabela (3.2) mostra quais foram os métodos mais bem sucedidos para cada problema. A primeira linha mostra a porcentagem de problemas para a qual cada um dos métodos avaliou menos vezes a função Lagrangiano Aumentado, a segunda linha considera as vezes em que cada método foi o segundo melhor, e assim até a última linha, que mostra a proporção total de fracassos.

colocação	BC	BCrand	BOBYQA	GENCAN	NM
1º	2.1 %	0 %	0 %	97.9 %	0 %
2º	10.6 %	2.1 %	57.4 %	2.1 %	23.4 %
3º	36.2 %	2.1 %	29.8 %	0 %	23.4 %
4º	29.8 %	32.0 %	4.3 %	0.0 %	21.3 %
5º	4.3 %	46.8 %	0.0 %	0.0 %	25.5 %
fracassos	17.0 %	17.0 %	8.5 %	0.0 %	6.4 %

Tabela 3.2: Comparação de desempenho para 47 problemas de Hock-Schittkowski.

Com base nas tabelas anteriores, afirmamos que o subalgoritmo que deixou o Algoritmo 2 mais robusto foi GENCAN com derivadas aproximadas, seguido por BOBYQA e NM praticamente empatados, e por fim BC e BCrand. Quanto à eficiência, o melhor certamente foi GENCAN, seguido por BOBYQA, logo após BC e NM, e em último lugar BCrand.

3.2 Problema da Área

Uma das razões mais importantes para a insistência da comunidade científica em construir novos algoritmos sem derivadas é a existência de problemas onde as derivadas não estão disponíveis. De todas as situações desse tipo, talvez a mais recorrente ocorra quando as funções envolvidas advêm de simulações. A razão é simples: se a função objetivo é computada através de uma simulação, sua expressão algébrica é inexistente, logo não faz sentido algum calcular suas derivadas.

Os exemplos na literatura de simulação como função objetivo ou restrição são extremamente elaborados. Como em geral são provenientes de problemas reais, muitas vezes os leitores não têm sequer acesso aos códigos. Ou então não possuem computadores suficientemente potentes para avaliar as funções em tempo praticável. Nesse contexto de dificuldade em encontrar testes adequados para nosso algoritmo, esforçamo-nos em desenvolver um problema sem derivadas que pudéssemos modelar e implementar. Nas várias tentativas, descobrimos que muitos dos problemas que pensamos tinham derivadas disponíveis, mesmo que uma análise superficial fizesse-nos crer o contrário. Em outros casos, os problemas acabavam se mostrando irrestritos, pois percebíamos que as restrições a princípio imaginadas eram irrelevantes ou sem sentido. Ao fim desse trajeto de busca por um experimento computacional que atendesse aos nossos anseios, chegamos a um problema restrito, sem derivadas, simples em sua formulação, de forte apelo geométrico e de fácil implementação.

Consideremos o conjunto de 10 pontos no plano representados pela figura (3.1). Perguntamo-nos qual é o círculo de menor área que contém todos os pontos. A resposta é representada

pela figura (3.2).

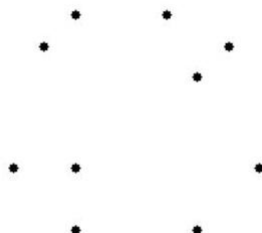


Figura 3.1: Pontos que devem estar contidos na figura ótima.

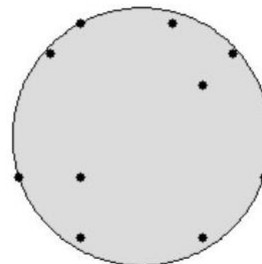


Figura 3.2: Menor círculo que contém os 10 pontos.

Sendo (\bar{x}^i, \bar{y}^i) , $i = 1, \dots, 10$ as coordenadas dos pontos que devem estar dentro do círculo ótimo, podemos encontrar a resposta à questão proposta através do problema de otimização: minimizar r^2 , sujeita a $(x - \bar{x}^i)^2 + (y - \bar{y}^i)^2 \leq r^2$, $i = 1, \dots, 10$, onde r é o raio do círculo que procuramos e (x, y) são as coordenadas de seu centro. Observamos então que o problema possui derivadas, uma vez que dispomos das expressões algébricas das funções envolvidas. Entretanto, esse não seria o caso se escolhêssemos, ao invés de um círculo, uma figura para a qual não possuímos a fórmula da área. Se não somos capazes de precisar a área de uma figura a partir dos parâmetros que a definem, podemos ao menos encontrar seu valor aproximado através de uma simulação.

Por exemplo, suponhamos que queremos encontrar a área da região em forma de gota da figura (3.3). Colocamos a figura em um quadrado de área conhecida, no caso $7 * 7 = 49$. Geramos, com distribuição uniforme de probabilidade, 105 pontos no interior do quadrado, e observamos que 15 deles estão dentro da gota. Uma aproximação sensata para a área da figura desejada é a área do quadrado que a contém vezes a proporção de pontos que caíram no interior da figura. No nosso caso, temos que a área da gota é aproximadamente $49 * (15/105) = 7$. Evidentemente, quantos mais pontos gerados aleatoriamente dentro da caixa, mais fidedigna a simulação. Este procedimento pode ser considerado um caso particular da aproximação de Monte Carlo para integrais múltiplas [13].

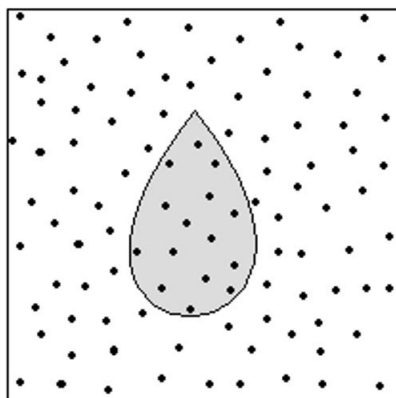


Figura 3.3: Exemplo de simulação de área.

Mas se conhecemos a função que define uma figura, podemos, mesmo que com certo trabalho, encontrar a expressão de sua área, e assim suas derivadas. Consideremos então figuras formadas pela intersecção ou união de figuras conhecidas. Nos nossos exemplos, essas figuras serão:

- retângulos, definidos por 4 parâmetros: as coordenadas (x, y) do vértice inferior esquerdo, a base e a altura. Não permitimos rotação do retângulo;
- círculos, definidos pelo raio r e as coordenadas (x, y) do centro;
- elipses, definidas por 6 parâmetros, a, b, c, d, e e f , através da inequação $ax^2 + 2xy + cy^2 + dx + ey + f \leq 0$.

Mesmo que seja sempre possível encontrar o valor exato da área intersecção entre um retângulo e um círculo, por exemplo, é extremamente difícil encontrar uma fórmula geral, principalmente pela diversidade de possibilidades. Resolvemos o problema de minimizar a área da intersecção entre duas das figuras da lista acima, sujeita a que os pontos da figura (3.1) estejam contidos nesta intersecção. Procedemos da mesma forma para a união entre duas figuras. Desse modo, obtemos 12 problemas diferentes. Alguns detalhes do problema e da implementação estão relacionados abaixo.

- as simulações foram feitas iniciando sempre com a mesma semente, de forma que o valor encontrado para a área de uma mesma figura fosse sempre a mesma em diferentes chamadas do simulador;

- a simulação das intersecções foi feita dentro do menor retângulo que contivesse uma das duas figuras envolvidas;
- os chutes iniciais foram quadrados de lado 2 centrados na origem para os retângulos, e círculos de raio unitário centrados na origem para círculos e elipses;
- no caso da união ou intersecção entre dois retângulos, o chute inicial foi um quadrado de lado 2 centrado na origem mais um retângulo de base 2.9, altura 5.7 e vértice inferior esquerdo em $(-1.0, -2.7)$;
- a densidade de pontos gerados foi de 10^5 pontos por unidade de área, desde que a quantidade de pontos não ultrapassasse o máximo de 10^7 ;
- nos problemas de união que não envolvia elipses, a área foi calculada como a soma das áreas das duas figuras menos a área simulada da intersecção;
- nos problemas de união entre elipses e círculos (o mesmo para elipses e retângulos), a área foi calculada como a área do círculo mais a área simulada da figura formada pela elipse menos sua intersecção com o círculo;
- como a inequação que define a elipse pode também definir parábolas, hipérboles e degenerescências, nos casos em que se constatou que os parâmetros não definiam uma elipse a área foi fixada em um valor alto (10^{10}) e a simulação não foi feita;
- o número de variáveis de cada problema é a soma do número de parâmetros necessários para definir cada uma das figuras;
- o número de restrições nos problemas de intersecção é 20: uma restrição para cada um dos 10 pontos pertencer a cada uma das 2 figuras cuja área da intersecção estamos minimizando;
- o número de restrições nos problemas de união é 10: uma para cada um dos 10 pontos estar contido em ao menos uma das duas figuras cuja área da união estamos minimizando.

Para entendermos as restrições, exemplifiquemos com o caso de dois círculos. Se temos intersecção entre dois círculos definidos pelas tríades (x_A, y_A, r_A) e (x_B, y_B, r_B) , temos que para cada um dos 10 pontos (\bar{x}^i, \bar{y}^i) , as relações $(x_A - \bar{x}^i)^2 + (y_A - \bar{y}^i)^2 \leq r_A^2$ e $(x_B - \bar{x}^i)^2 + (y_B - \bar{y}^i)^2 \leq r_B^2$ devem ser verificadas. No caso da união entre dois círculos, temos que apenas uma das duas restrições anteriores precisa ser satisfeita. Conseguimos isso impondo que $\min\{(x_A - \bar{x}^i)^2 + (y_A - \bar{y}^i)^2 - r_A^2, (x_B - \bar{x}^i)^2 + (y_B - \bar{y}^i)^2 - r_B^2\} \leq 0$. Notem que essa

restrição tem derivada descontínua, de modo que não estamos mais amparados pelas garantias de convergência do Capítulo 1. A própria função objetivo, a área simulada, tem derivada descontínua pois, variando gradualmente um dos parâmetros, o valor da área simulada se mantém constante e dá pequenos saltos a cada vez que um ponto gerado aleatoriamente entra ou sai da figura a ter a área calculada. Ainda assim, os problemas foram resolvidos, e os resultados obtidos foram satisfatórios.

Para os subproblemas, utilizamos BC, BCrand, BOBYQA e NM. Para os testes desta seção, decidimos aumentar ε_{fact} para 10^{-4} , devido à não continuidade da função objetivo. Como multiplicamos o valor de ε_{fact} utilizado na seção anterior por 10, o mesmo fizemos para os parâmetros diretamente relacionados a tolerâncias, na tentativa de sermos coerentes com as proporções entre os parâmetros que pareceram até então mais adequadas. Desse modo, definimos $\delta_{opt} = 10^{-4}$, $\rho_{end} = 10^{-5}$, $\Delta_0^{BOBYQA} = 0.1$, $\varepsilon_{NM} = 10^{-4}$ e $\Delta_0^{NM} = 0.5$.

Tentamos resolver os subproblemas com GENCAN, no entanto o resultado foi extremamente ruim, pois este apresentou muita dificuldade em encontrar soluções. Cremos que isso ocorreu devido ao caráter descontínuo e impreciso da função objetivo. Por exemplo, em testes que fizemos com a função objetivo sendo a área exata da intersecção entre dois retângulos (que por sinal é um outro retângulo), o desempenho do algoritmo foi muito bom. Por outro lado, em testes com a área simulada, não conseguimos encontrar a resposta. Por esse motivo, desistimos de utilizar GENCAN para o problema da área. Acreditamos que qualquer outro método baseado em aproximações de gradientes por diferenças finitas também apresentaria o mesmo desempenho insatisfatório.

Selecionamos alguns dos resultados obtidos e mostramos nas figuras que se seguem. Nas legendas, o método utilizado na resolução do subproblema, o número *feval* de avaliações de função objetivo e o valor *A* da área simulada. As quatro primeiras figuras representam a intersecção entre dois círculos, seguidas pela intersecção de um retângulo com uma elipse, união de retângulo com elipse e união de duas elipses.

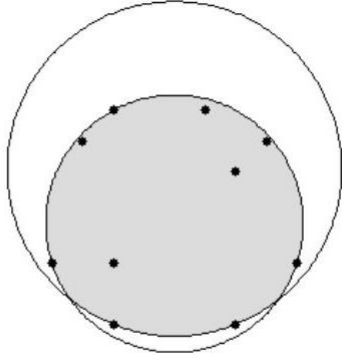


Figura 3.4: BC, $A = 13.247$, $feval = 4277$.

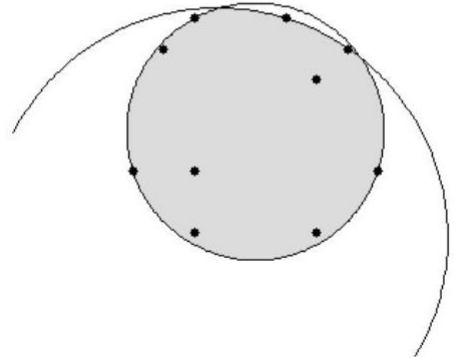


Figura 3.5: BCrand, $A = 13.582$, $feval = 8669$.

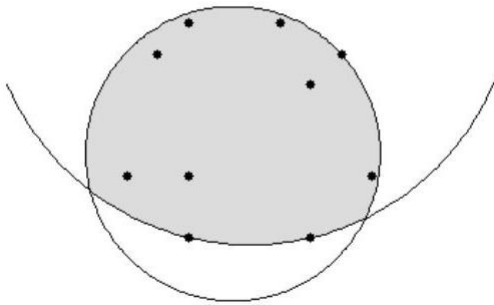


Figura 3.6: BOBYQA, $A = 15.004$, $feval = 6283$.

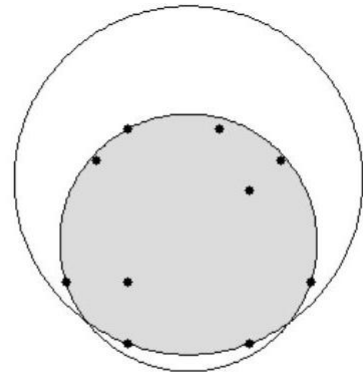


Figura 3.7: NM, $A = 13.247$, $feval = 6099$.

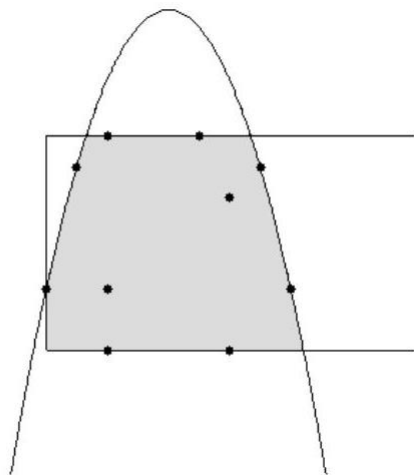


Figura 3.8: BC, $A = 12.574$, $feval = 6260$.

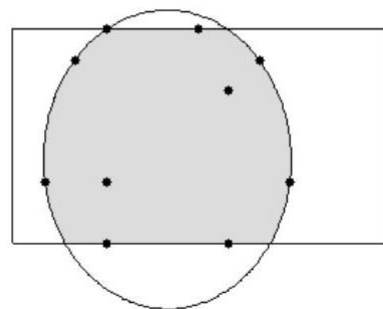


Figura 3.9: BCrand, $A = 12.589$, $feval = 11834$.

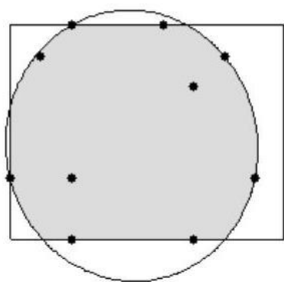


Figura 3.10: BOBYQA, $A = 12.564$, $feval = 4177$.

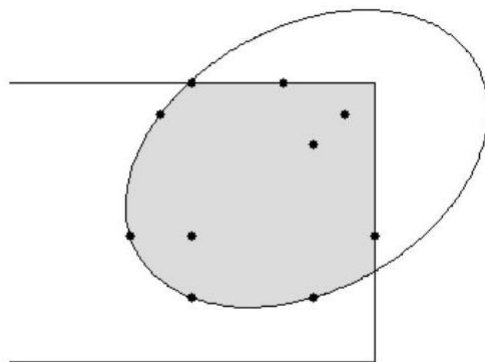


Figura 3.11: NM, $A = 13.174$, $feval = 11379$.

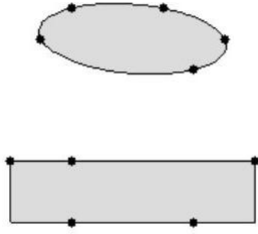


Figura 3.12: BC, $A = 6.753$, $feval = 9955$.

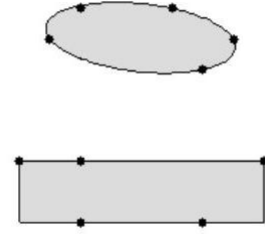


Figura 3.13: BCrand, $A = 6.734$, $feval = 13075$.

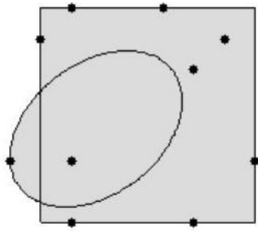


Figura 3.14: BOBYQA, $A = 12.887$, $feval = 6059$.

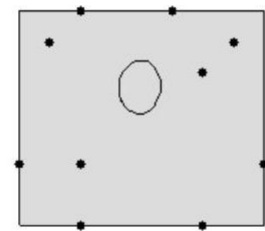


Figura 3.15: NM, $A = 14.000$, $feval = 7251$.

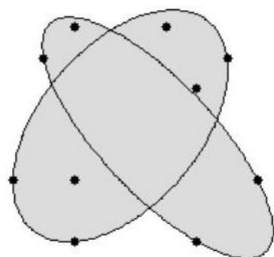


Figura 3.16: BC, $A = 12.237$, $feval = 16154$.

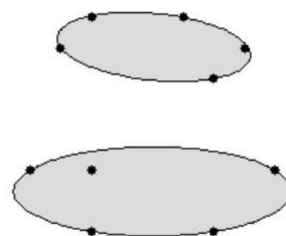


Figura 3.17: BCrand, $A = 7.941$, $feval = 13246$.

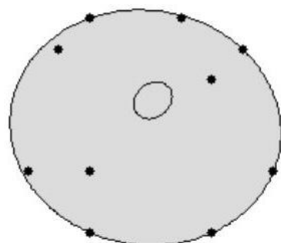


Figura 3.18: BOBYQA, $A = 13.339$, $feval = 8191$.

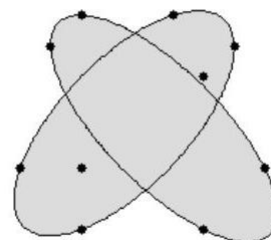


Figura 3.19: NM, $A = 11.363$, $feval = 18881$.

A tabela (3.3) mostra o número de avaliações de função e a área simulada em cada problema para cada algoritmo interno. A sigla I representa intersecção, U união, R retângulo, C círculo e E elipse.

figura	área				feval			
	BC	BOBYQA	NM	BCrand	BC	BOBYQA	NM	BCrand
I RR	14.001	13.999	14.003	14.000	2709	1535	4959	5499
I RC	12.722	13.467	12.572	13.457	6035	3629	7784	10705
I CC	13.247	15.004	13.247	13.582	4277	6283	6099	8669
I RE	12.574	12.564	13.174	12.589	6260	4177	11379	11834
I CE	13.211	12.973	13.207	13.211	5020	7201	11132	6952
I EE	12.058	13.519	13.115	12.058	6881	13156	12710	11291
U RR	11.500	12.250	11.499	11.500	4312	1745	5717	8107
U RC	13.247	13.894	13.231	13.247	2788	874	5466	6304
U CC	13.082	13.890	13.081	13.082	2636	4021	4798	6056
U RE	6.723	12.887	14.000	6.734	9955	6059	7251	13075
U CE	12.459	13.894	13.333	12.458	6715	2042	11688	13125
U EE	12.237	13.399	11.363	7.941	16154	8191	18881	13246
média	12.255	13.478	12.985	11.988	6145	4909	8989	9572

Tabela 3.3: Resultados para o problema da área mínima.

Em termos de valor de função objetivo, a melhor solução encontrada foi com os métodos de BC e BCrand, e está representada nas figuras (3.12) e (3.13). A pior solução, obtida com BOBYQA, está na figura (3.6). Comparando os métodos utilizados, BCrand encontrou soluções melhores em quase todos os casos, como podemos observar pela média de valor de área, seguida por BC, NM e BOBYQA. Quanto ao número de avaliações de função, o melhor desempenho foi de BOBYQA, seguido por BC, NM e BCrand.

3.3 Problema MDO

Abordamos aqui um problema de otimização de design multidisciplinar, que chamaremos problema MDO, do inglês Multidisciplinary Design Optimization. É um problema advindo da engenharia mecânica, retirado de [2] e resolvido em [6]. Um modelo com 10 variáveis representa a estrutura de uma aeronave, e o objetivo é maximizar a área desta, sujeita a 10 restrições de desigualdade. O código em C++ do modelo pode ser encontrado em [34].

Na resolução dos subproblemas, empregamos os algoritmos BC, BCrand, BOBYQA, NM e NOMAD. Os parâmetros são os mesmos da seção anterior, e para NOMAD foi escolhido $\Delta_0^{NOMAD} = 0.1$. Os chutes iniciais x_F^0 e x_I^0 , respectivamente factível e infactível, foram reti-

rados de [6]. São eles

$$x_F^0 = (0.4, 1, 0.872, 0.4433, 0.05, 4500, 1.728, 3.196, 62.68, 1000) \text{ e}$$

$$x_I^0 = (0.4, 0.75, 0.75, 0.189296875, 0.09, 5700, 1.4, 2.5, 70, 1500).$$

O valor de função objetivo encontrado e o número necessário de avaliações de função são mostrados na tabela (3.4). No caso do ponto inicial infactível, o Algoritmo 2 com NM para os subproblemas não conseguiu encontrar uma solução, declarando fracasso 2.

algoritmo	x_F^0		x_I^0	
	f	feval	f	feval
BC	-1436.0501	7310	-2992.7872	3608
BOBYQA	-1351.7237	4613	-3248.3072	3893
NM	-1373.2085	4805791	X	X
BCrand	-1436.0501	7774	-2992.7872	5845
NOMAD	-3955.6484	7789	-3964.0682	10244

Tabela 3.4: Comparação entre os subalgoritmos para o problema MDO.

Podemos observar que a estratégia de usar pontos aleatórios para tentar fugir dos minimizadores locais não foi eficiente neste caso. No entanto, em alguns testes, o Algoritmo 2 com BCrand encontrou a mesma solução que Algoritmo 2 com NOMAD, mas para escolhas muito particulares dos parâmetros e do algoritmo de busca coordenada, por exemplo, usando $\alpha = 2$ e ajustando o escalamento das variáveis na busca coordenada através da mudança de variáveis $\bar{x}_i = 10x_i/(u_i - l_i), i = 1, \dots, n$. É bastante evidente o melhor desempenho de NOMAD. Não podemos mensurar o quanto isso é fruto de uma superioridade inerente ao algoritmo ou consequência do fato de o problema ter sido escolhido exatamente para testar o algoritmo em questão, estando inclusive ambos os códigos escritos na mesma linguagem C++.

Conclusões

Introduzimos um método Lagrangiano Aumentado sem derivadas para o problema geral de otimização. Isso foi conseguido considerando um método Lagrangiano Aumentado preexistente e fazendo modificações no critério de parada para os subproblemas, a fim de evitar o cálculo de derivadas. Dois critérios de parada foram propostos, um para o caso de restrições de caixa no nível inferior, com inspiração no método de busca coordenada para caixas, e outro substituindo o gradiente no critério original por um gradiente aproximado por diferenças finitas, para o caso de restrições gerais no nível inferior. Incentivamos o uso de qualquer algoritmo sem derivadas na resolução dos subproblemas, desde que este seja capaz de encontrar pontos que satisfaçam os critérios de parada, mesmo que para tanto deva ser sucedido por métodos com essa característica.

Todos os resultados de convergência do trabalho original foram preservados, garantindo, sob certas condições, que os pontos limites das sequências geradas são factíveis, estacionários e que as sequências de parâmetros de penalidade são limitadas, de maneira que esperamos que subproblemas mal condicionados oriundos de parâmetros de penalidade arbitrariamente grandes não ocorram com frequência na prática. Os teoremas, com exceção do referente ao parâmetro de penalidade, foram baseados no uso da condição de qualificação CPLD que, por ser mais fraca que condições mais usuais como a regularidade, tornam os resultados mais fortes e mais abrangentes.

Experimentos numéricos foram apresentados, dentre os quais destacamos o problema da área. Uma vez que a função objetivo é calculada por simulações, o problema só pode ser resolvido por métodos sem derivadas. É o exemplo mais simples de problema sem derivadas que temos conhecimento, pois em muitos casos práticos a inexistência de derivadas explícitas reside exatamente na complexidade dos pacotes que calculam as funções envolvidas. No experimento MDO, notamos que o uso de um conjunto de direções de busca não finito, como no algoritmo MADS, pode melhorar a convergência. À finitude do conjunto de direções é atribuída a lentidão de muitos métodos sem derivadas, em particular os de busca padrão, já que as direções testadas não aproveitam de forma alguma as informações sobre a função objetivo. Isso torna o desempenho desses algoritmos especialmente ruim para problemas de

dimensão grande, uma vez que para muitos algoritmos o número de direções cresce com n . Não realizamos, no entanto, testes para problemas grandes, sendo um estudo nesse sentido uma das possibilidades de continuidades para o presente trabalho.

Referências Bibliográficas

- [1] M.A. ABRAMSON, C. AUDET, J.E. DENNIS JR. & S. LE DIGABEL, *OrthoMads: A deterministic MADS instance with orthogonal directions*, a aparecer no SIAM Journal on Optimization.
- [2] J.S. AGTE, R.R. SANDUSKY JR & J. SOBIESZCZANSKI-SOBIESKI, *Bi-level Integrated System Synthesis (BLISS)*, Technical Report NASA/TM-1998-208715, NASA, Langley Research Center, 1998.
- [3] R. ANDREANI, E.G. BIRGIN, J.M. MARTÍNEZ & M.L. SCHUVERDT, *On Augmented Lagrangian methods with general lower-level constraints*, SIAM Journal on Optimization 18, pp. 1286-1309, 2007.
- [4] R. ANDREANI, J.M. MARTÍNEZ & M.L. SCHUVERDT. *On the relation between the Constant Positive Linear Dependence condition and quasinormality constraint qualification*, Journal of Optimization Theory and Applications 125, pp. 473-485, 2005.
- [5] C. AUDET & J.E. DENNIS JR., *Mesh adaptive direct search algorithms for constrained optimization*, SIAM Journal on Optimization 17, pp. 188-217, 2006.
- [6] C. AUDET, J.E. DENNIS JR & S. LE DIGABEL, *Globalization Strategies for Mesh Adaptive Direct Search*, Technical Report, GERAD G-2008-74, GERAD, 2008.
- [7] C. AUDET & D. ORBAN, *Finding optimal algorithmic parameters using derivative-free optimization*, SIAM Journal on Optimization 17, pp. 642-664, 2006.
- [8] E.G. BIRGIN AND J.M. MARTÍNEZ, *Structured Minimal-Memory inexact quasi-Newton method and secant preconditioners for Augmented Lagrangian Optimization*, Computational Optimization and Applications 39, pp. 1-16, 2008.
- [9] A.R. CONN, N.I.M. GOULD, A. SARTENAER & PH.L. TOINT, *Convergence properties of an Augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints*, SIAM Journal on Optimization 6, pp. 674-703, 1996.

- [10] A.R. CONN, N.I.M GOULD & P.L. TOINT, *A globally convergent Augmented Lagrangian algorithm for optimization with general constraints and simple bounds*, SIAM Journal on Numerical Analysis 28, pp. 545-572, 1991.
- [11] A.R. CONN, K. SCHEINBERG & P.L. TOINT, *A derivative free optimization algorithm in practice*, Proceedings of the American Institute of Aeronautics and Astronautics Conference, St Louis, 1998.
- [12] A.R. CONN, K. SCHEINBERG & L.N. VICENTE, *Introduction to derivative-free optimization*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.
- [13] B.P. DEMIDOVICH & I.A. MARON, *Computational Mathematics*, Mir Publishers, Moscow, 1987
- [14] M.R. HESTENES, *Multiplier and Gradient Methods*, Journal of Optimization Theory and Applications 4, pp. 303-320, 1969.
- [15] W. HOCK & K. SCHITTKOWSKI, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer, 1981.
- [16] R. HOOKE & T.A. JEEVES, *Direct search solution of numerical and statistical problems*, Journal of the Association for Computing Machinery 8, pp. 212-229, 1961.
- [17] T.G. KOLDA, R.M. LEWIS & V. TORCZON, *A generating set direct search augmented Lagrangian algorithm for optimization with a combination of general and linear constraints*, Technical Report, SAND2006-5315, Sandia National Laboratories, 2006.
- [18] T.G. KOLDA, R.M. LEWIS & V. TORCZON, *Stationarity results for generating set search for linearly constrained optimization*, SIAM Journal on Optimization 17, pp 943-968, 2006.
- [19] J.C. LAGARIAS, J.A. REEDS, M.H. WRIGHT & P.E. WRIGHT, *Convergence properties of the Nelder-Mead simplex algorithm in low dimensions*, SIAM J. Opt. 9, pp. 112-147, 1998.
- [20] R.M. LEWIS & V. TORCZON, *A globally convergent Augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds*, SIAM Journal on Optimization 12, pp. 1075-1089, 2002.
- [21] R.M. LEWIS & V. TORCZON, *Pattern search algorithms for bound constrained minimization*, SIAM Journal on Optimization 9, pp. 1082-1099, 1999.

- [22] R.M. LEWIS, V. TORCZON & M.W. TROSSET, *Direct search methods: then and now*, ICASE Report 2000-26, ICASE, NASA Langley Research Center, Hampton, VA, 2000.
- [23] S. LUCIDI & M. SCIANDRONE, *A derivative-free algorithm for bound constrained optimization*, Computational Optimization and Applications 21, pp. 119-142, 2002.
- [24] S. LUCIDI & M. SCIANDRONE, *On the global convergence of derivative-free methods for unconstrained optimization*, SIAM Journal on Optimization 13, pp. 97-116, 2002.
- [25] K.I.M. MCKINNON, *Convergence of the Nelder-Mead simplex method to a nonstationary point*, SIAM Journal on Optimization 9, pp. 148-158, 1998.
- [26] J.A. NELDER & R. MEAD, *A simplex method for function minimization*, The Computer Journal 7, pp. 308-313, 1965.
- [27] E. POLAK, *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, 1971.
- [28] M.J.D. POWELL, *A Method for Nonlinear Constraints in Minimization Problems*, Optimization, R. Fletcher, Academic Press, New York, pp. 283-298, 1969.
- [29] M.J.D. POWELL, *The BOBYQA algorithm for bound constrained optimization without derivatives*, Cambridge NA Report NA2009/06, University of Cambridge, Cambridge, Reino Unido, 2009.
- [30] M.J.D. POWELL, *The NEWUOA software for unconstrained minimization without derivatives*, Large-Scale Nonlinear Optimization, pp. 255-297, 2006.
- [31] R.T. ROCKAFELLAR, *Augmented Lagrange Multiplier Functions and Duality in Non-convex Programming*, SIAM Journal on Control and Optimization 12, pp. 268-285, 1974.
- [32] V. TORCZON, *On the convergence of pattern search algorithms*, SIAM Journal on Optimization 7, pp. 1-25, 1997.
- [33] P. TSENG, *Fortified-descent simplicial search method: A general approach*, SIAM Journal on Optimization 10, pp. 269-288, 1999.
- [34] <http://www.gerad.ca/nomad/> consultado em 10/12/2008.
- [35] <http://www.ime.usp.br/~egbirgin/tango/> consultado em 10/08/2006.