ESTUDO E AVALIAÇÃO DE PROBLEMAS ASSOCIADOS A CONES DE SEGUNDA ORDEM

Denise dos Santos Trevisoli Maria Aparecida Diniz Ehrhardt

DMA-IMECC-UNICAMP

Resumo

Este trabalho tem como foco o estudo de problemas SOCP, tanto nos seus aspectos teóricos quanto nos seus aspectos práticos. Problemas SOCP são problemas convexos de otimização nos quais uma função linear é minimizada sobre restrições lineares e restrições de cone quadrático. Temos dois objetivos principais: estudar o conceito, as aplicações e os métodos de resolução de problemas SOCP, permitindo avaliar a viabilidade de trabalhar com tais problemas; e verificar na prática o benefício de se utilizar uma ferramenta específica de SOCP para a resolução de problemas que se enquadram nessa classe. Para tal utilizamos um software de otimização genérica (`fmincon`) e outro específico de SOCP (CVXOPT). A análise ficou concentrada nos requisitos: robustez, número de iterações e variação do tempo com o aumento da dimensão dos problemas.

Abstract

This work focuses on the study of SOCP problems, both in its theoretical, and in its practical aspects. SOCP problems are convex optimization problems in which a linear function is minimized over linear constraints and second-order cone constraints. We have two main objectives: to study the concept, applications and methods for solving the SOCP problem, making it possible to evaluate the feasibility of working with such problems; and to verify the practical benefits of using a SOCP specific tool for the resolution of problems of this class. So we used a generic optimization software

(`fmincon`) and other SOCP specific software (CVXOPT). The analysis was concentrated on the robustness, number of iterations and time variation with the increasing scale of the problems.

1 Introdução

Programação não linear (PNL) é um campo de pesquisa bastante ativo, com aplicação direta em várias áreas do conhecimento humano. Em problemas de PNL minimizamos ou maximizamos uma função, conhecida por função objetivo, sujeita a restrições de igualdade e desigualdade. Tais problemas se caracterizam por envolverem ao menos uma função não linear. Isto faz com que sua resolução tenda a ser complexa, o que se traduz na prática em processos de otimização computacionalmente caros.

Dentro deste cenário, torna-se importante explorar as propriedades matemáticas de cada classe de problemas, permitindo direcionar o processo de otimização, e tornando-o mais eficiente. Este processo nem sempre é simples, mas pode representar a diferença entre uma otimização ser ou não computacionalmente viável.

Na década de 90, identificou-se que vários tipos de problemas podiam ser formulados como problemas de programação cônica de segunda ordem (SOCP - *Second-Order Cone Programming*). Problemas SOCP são problemas convexos de otimização nos quais uma função linear é minimizada sobre restrições lineares e restrições de cone quadrático.

Tratados como SOCP, muitos problemas, como por exemplo, programação linear robusta, programação quadrática, programação quadrática com restrições quadráticas, entre outros, podem ser otimizados com maior eficiência, estabelecendo um campo de pesquisa bastante frutífero.

Exemplos específicos de otimização de problemas SOCP são abundantes na literatura científica. Por exemplo, o problema clássico de *Fermat-Weber* (descrito em 4.1), vide [23] e Witzgall apud [1]. Lobo et al [9], Boyd et al [3] e Srirangarajan et al [16] expõem diversas aplicações de problemas SOCP em engenharia. Os artigos de Nesterov e Nemirovski [13], Lobo et al [9], Nemirovski e Ben-Tal [12], Vandenberghe e Boyd [19] e Boyd et al [3] mostram que diversos tipos de problemas, entre eles programação linear robusta, quadrados mínimos

robustos, problemas com restrições hiperbólicas, mínimo de soma de normas e problemas de controle também podem ser formulados como SOCP.

Devido à importância e aplicabilidade do problema SOCP, podem ser encontrados não apenas estudos científicos, mas também softwares específicos para a sua resolução. São exemplos destes softwares o CVXOPT [18], o SeDuMi [17] e o CVX [6], mantidos principalmente por pesquisadores e disponibilizados de forma livre.

Este trabalho tem como foco problemas SOCP, tanto nos seus aspectos teóricos quanto nos seus aspectos práticos. Como principais objetivos, temos: (i) estudar o conceito, as aplicações e os métodos de resolução de problemas SOCP, permitindo verificar a viabilidade de trabalhar com estes; e (ii) avaliar na prática o benefício de se utilizar uma ferramenta específica de SOCP para a resolução de problemas que se enquadram nessa classe.

O estudo teórico foi conduzido tanto com base em material específico de SOCP, quanto em material relativo a áreas relevantes ao estudo destes problemas. As principais áreas estudadas foram programação não linear, análise convexa, problemas SOCP, métodos de pontos interiores, método de barreira (muito comum na resolução de problemas de PNL), método de redução potencial e método seguidor de caminho com escalamento de Nesterov-Todd (bastante utilizado na resolução de problemas SOCP).

A avaliação prática utilizou um software de otimização genérica (`fmincon` [10]) e outro específico de SOCP (CVXOPT). Durante o estudo, comparou-se a eficiência de cada ferramenta na otimização tanto de problemas teóricos (QCQP) quanto de problemas práticos (arranjo de antenas e garra com múltiplos dedos).

2 Cones Quadráticos

Definição 2.1. *Um cone é um conjunto não vazio $C \subset \mathbb{R}^n$ tal que para qualquer $x \in C$, temos $\alpha x \in C$, $\forall \alpha \geq 0$. Além disso, se C é convexo, então C é chamado cone convexo [2].*

Definição 2.2. *Considere uma norma $\|.\|$ arbitrária em $\mathbb{R}^{k-1}$. Um cone normado associado à norma $\|.\|$ é o conjunto $C_k = \left\{(x,t) \mid x \in \mathbb{R}^{k-1}, t \in \mathbb{R}, \|x\| \leq t\right\} \subseteq \mathbb{R}^k$ [19]. Para $k = 1$ definimos $C_1 = \{t \mid 0 \leq \|t\|\}$.*

Definição 2.3. *Um cone quadrático é um cone normado, na norma Euclidiana [19], isto é,* $C_k = \left\{(x,t) \mid x \in \mathbb{R}^{k-1}, t \in \mathbb{R}, \|x\|_2 \leq t\right\} \subseteq \mathbb{R}^k$.

O cone quadrático é também chamado de cone de segunda ordem e recebe a abreviação SOC (*second-order cone*).

3 Problemas Associados a Cones de Segunda Ordem

Problemas de programação cônica de segunda ordem (SOCP - *Second-Order Cone Programming*) são problemas convexos de otimização nos quais uma função linear é minimizada sobre restrições lineares e restrições de cone quadrático. Estes problemas são um caso especial de programação semidefinida (SDP - *semidefinite programming*).

Problemas de programação linear, problemas de programação quadrática e problemas de programação quadrática com restrições quadráticas podem ser formulados como problemas SOCP, assim como outros problemas que não se encaixam em nenhuma destas três categorias [1].

A norma Euclidiana será representada por $\|.\|$.

Um problema SOCP é escrito na forma:

$$\begin{array}{ll} \text{Minimizar} & f^T x \\ \text{sujeita a} & \|A_i x + b_i\| \leq {c_i}^T x + d_i, \quad i = 1, ...N, \\ & Fx = g, \end{array} \tag{1}$$

onde $x \in \mathbb{R}^n$ é a variável de otimização, e os parâmetros do problema são $f \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{(n_i-1)\times n}$, $b_i \in \mathbb{R}^{n_i-1}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $F \in \mathbb{R}^{m\times n}$, $g \in \mathbb{R}^m$.

Chamamos a restrição $\|A_i x + b_i\| \leq {c_i}^T x + d_i$ de restrição de cone quadrático, ou restrição SOC, já que o conjunto de pontos que satisfazem essa desigualdade é a imagem inversa de um cone quadrático sobre a função afim $f(x) = (Ax + b, c^T x + d)$. Observe que essa restrição é convexa.

Temos então que o problema SOCP (1) é convexo, já que sua função objetivo é convexa e suas restrições definem um conjunto convexo.

4 Problemas que podem ser expressos como SOCP

Diversos problemas podem ser escritos como problemas SOCP. Nesta seção serão apresentados alguns exemplos, extraídos de [19], [9] e [12].

4.1 Soma de normas

Considere o problema irrestrito:

$$\text{Minimizar} \sum_{i=1}^{p} \|A_i x + b_i\|, \tag{2}$$

onde $A_i \in \mathbb{R}^{(n_i-1)\times n}$ e $b_i \in \mathbb{R}^{(n_i-1)}$, $i = 1, ..., p$.

Introduzindo novas variáveis auxiliares, $t_1, ..., t_p$, podemos escrever o problema acima como um problema SOCP:

$$\begin{aligned} \text{Minimizar} \quad & \sum_{i=1}^{p} t_i \\ \text{sujeita a} \quad & \|A_i x + b_i\| \leq t_i, \qquad i = 1, ..., p, \end{aligned}$$

com variáveis de otimização $x \in \mathbb{R}^n$ e $t_i \in \mathbb{R}$.

Observe que não há alteração nos cálculos se inserirmos qualquer restrição SOC no problema (4.1).

O problema clássico de *Fermat-Weber* é um caso especial de soma de normas. Ele identifica onde posicionar uma instalação de modo a minimizar a soma das distâncias desta em relação a um conjunto de locais fixos. A formulação deste problema é:

$$\text{Minimizar} \sum_{i=1}^{p} \|d_i - x\|$$

onde d_i, $i = 1, ..., p$ são as posições dos locais fixos e x é a posição desconhecida da instalação.

4.2 Máximo de normas

De forma similar ao problema de soma de normas, podemos escrever como SOCP problemas cujo objetivo é minimizar o máximo dentre algumas normas.

Considere o problema:

$$\text{Minimizar} \max_{i=1,\ldots,p} \|A_i x + b_i\| .$$

Introduzindo uma única variável t, escrevemos o problema acima como SOCP:

$$\begin{array}{ll} \text{Minimizar} & t \\ \text{sujeita a} & \|A_i x + b_i\| \leq t, \quad i = 1, ..., p. \end{array}$$

As variáveis deste problema são: $x \in \mathbb{R}^n$ e $t \in \mathbb{R}$.

4.3 QCQP - *Quadratically Constrained Quadratic Programming*

Um problema de otimização convexa é chamado de problema de programação quadrática (PQ) se a função objetivo é quadrática e as funções de restrição são afins. Um problema de programação quadrática pode ser expresso na forma:

$$\begin{array}{ll} \text{Minimizar} & \frac{1}{2}x^t P x \ + \ q^t x + r \\ \text{sujeita a} & Bx \ \leq \ v \\ & Fx = g, \end{array} \tag{3}$$

onde $P \in \mathbb{R}^{n\times n}$ é simétrica semidefinida positiva, $B \in \mathbb{R}^{N\times n}$, $x,\ q \in \mathbb{R}^n,\ \ r \in \mathbb{R}$, $v \in \mathbb{R}^N$, $F \in \mathbb{R}^{m\times n}$ e $g \in \mathbb{R}^m$.

Problemas de PQ incluem problemas de PL como um caso especial, basta tomar $P = 0$.

Um problema de otimização convexa é chamado de QCQP (*Quadratically Constrained Quadratic Programming*) se a função objetivo e as restrições de desigualdade são quadráticas e as restrições de igualdade são lineares. Um problema QCQP pode ser expresso na forma:

$$\begin{array}{ll} \text{Minimizar} & x^t P x \ + \ 2q^t x + r \\ \text{sujeita a} & x^t A_i x \ + \ 2b_i^T x + c_i \leq 0, \quad i = 1, ..., N \\ & Fx = g, \end{array} \tag{4}$$

onde $P \in \mathbb{R}^{n\times n},\ A_i \in \mathbb{R}^{n\times n},\ i = 1, ..., N$, são simétricas e semidefinidas positivas, $F \in \mathbb{R}^{m\times n}$, $x,\ q,\ b_i \in \mathbb{R}^n$, $g \in \mathbb{R}^m$ e $r,\ c_i \in \mathbb{R}$.

Tomando $A_i = 0$ para todo i, o problema QCQP acima transforma-se em um problema PQ. Logo, problemas QCQP incluem problemas PQ, e consequentemente PL, como casos especiais.

Para mostrar que problemas PL, PQ e QCQP podem ser expressos como SOCP, mostraremos apenas a transformação de QCQP em SOCP, já que PL e PQ são casos especiais de QCQP.

Para simplificar os cálculos trabalharemos apenas com problemas cujas matrizes P e A_i são definidas positivas. Assim, o problema (4) pode ser escrito da seguinte forma:

$$\begin{array}{ll} \text{Minimizar} & \left\| P^{\frac{1}{2}}x \ + \ P^{-\frac{1}{2}}q \right\|^2 + r - q^t P^{-1} q \\ \text{sujeita a} & \left\| A_i^{\frac{1}{2}}x \ + \ A_i^{-\frac{1}{2}}b_i \right\|^2 + c_i - b_i^t A_i^{-1} b_i \leq 0, \quad i = 1, ...N, \\ & Fx = g. \end{array}$$

Introduzindo uma nova variável t, podemos escrever o problema acima como:

$$\begin{array}{ll} \text{Minimizar} & t \\ \text{sujeita a} & \left\| P^{\frac{1}{2}}x \ + \ P^{-\frac{1}{2}}q \right\|^2 + r - q^t P^{-1} q \leq t \\ & \left\| A_i^{\frac{1}{2}}x \ + \ A_i^{-\frac{1}{2}}b_i \right\|^2 + c_i - b_i^t A_i^{-1} b_i \leq 0, \quad i = 1, ...N, \\ & Fx = g. \end{array}$$

Tomando $d_i = b_i^t A_i^{-1} b_i - c_i$, temos:

$$\begin{array}{ll} \text{Minimizar} & t \\ \text{sujeita a} & \left\| P^{\frac{1}{2}}x \ + \ P^{-\frac{1}{2}}q \right\| \leq t \\ & \left\| A_i^{\frac{1}{2}}x \ + \ A_i^{-\frac{1}{2}}b_i \right\| \leq d_i, \quad i = 1, ...N, \\ & Fx = g. \end{array}$$

O problema acima é um SOCP com variável de otimização $(t, \ x)$.

5 Método de Pontos Interiores Primal-Dual Para Problemas SOCP

5.1 Dual do SOCP

Para os cálculos desta seção, associaremos ao cone C a ordenação parcial em $\mathbb{R}^n$ definida por:

$$x \geq_C y \Leftrightarrow x - y \in C. \tag{5}$$

Desta forma, temos que $x \geq_C 0$ representa $x \in C$.

Considere o problema SOCP (1) reescrito da seguinte forma:

$$\begin{array}{ll} \text{Minimizar} & f^T x \\ \text{sujeita a} & A_i x + b_i = u_i, \\ & c_i^t x + d_i = t_i, \\ & \|u_i\| \leq t_i \quad i = 1, ...N. \end{array} \tag{6}$$

Observe que, pela definição 2.2, as restrições de desigualdade $\|u_i\| \leq t_i$ implicam que, para todo i, o vetor $(u_i; t_i)$ pertence ao cone C_i. Desta forma, temos por (5) que $(u_i; t_i) \geq_{C_i} 0$ e podemos reescrever o problema primal por:

$$\begin{array}{ll} \text{Minimizar} & f^T x \\ \text{sujeita a} & A_i x + b_i = u_i, \\ & c_i^t x + d_i = t_i, \\ & (u_i; t_i) \geq_{C_i} 0 \quad i = 1, ...N. \end{array}$$

Nesta seção, abordaremos as formulações primal e dual associadas ao SOCP pois descreveremos, em seguida, métodos do tipo primal-dual.

Para trabalharmos com o dual Lagrangiano, $y = (x, u, t)$, e a função Lagrangiana associada ao problema primal:

$$L(y, \lambda, \mu) = f^T x + \sum_{i=1}^{N} \lambda_{1_i}^T (A_i x + b_i - u_i) + \lambda_{2_i} (c_i^T x + d_i - t_i) - \mu_i^T (u_i; t_i),$$

com $\mu_i \geq_{C_i} 0$, ou seja, $\mu_i \in C_i$.

Considerando $\mu_i = (z_i; w_i)$ temos novamente pela definição 2.2 que $\|z_i\| \leq w_i$. Assim, podemos reescrever a Lagrangiana como,

$$\begin{aligned} L(y,\lambda,\mu) &= f^T x + \sum_{i=1}^{N} \lambda_{1_i}^T (A_i x + b_i - u_i) + \lambda_{2_i}(c_i^T x + d_i - t_i) - (z_i; w_i)^T (u_i; t_i) \\ &= f^T x + \sum_{i=1}^{N} \lambda_{1_i}^T (A_i x + b_i - u_i) + \lambda_{2_i}(c_i^T x + d_i - t_i) - z_i^T u_i - w_i t_i. \end{aligned}$$

Reorganizando, temos:

$$L(y,\lambda,\mu) = \left(f + \sum_{i=1}^{N} (A_i^T \lambda_{1_i} + c_i \lambda_{2_i}) \right)^T x - \sum_{i=1}^{N} (\lambda_{1_i} + z_i)^T u_i - (\lambda_{2_i} + w_i) t_i.$$

Desta forma, temos:

$$\theta(\lambda,\mu) = \inf_y \; L(y,\lambda,\mu) = \begin{cases} \sum_{i=1}^{N} b_i^T \lambda_{1_i} + d_i^T \lambda_{2_i} & \text{se} \begin{cases} f + \sum_{i=1}^{N} A_i^T \lambda_{1_i} + c_i \lambda_{2_i} = 0, \\ \sum_{i=1}^{N} \lambda_{1_i} + z_i = 0, \\ \sum_{i=1}^{N} \lambda_{2_i} + w_i = 0. \end{cases} \\ -\infty & \text{caso contrário.} \end{cases}$$

De acordo com a equação acima, utilizamos $\lambda_{1_i} = -z_i$ e $\lambda_{2_i} = -w_i$ temos o SOCP dual:

$$\begin{aligned} \text{Maximizar} \quad & -\sum_{i=1}^{N} (b_i^T z_i + d_i w_i) \\ \text{sujeita a} \quad & \sum_{i=1}^{N} (A_i^T z_i + c_i w_i) = f \\ & \|z_i\| \leq w_i \quad i = 1, ...N. \end{aligned} \tag{7}$$

As variáveis duais são os vetores $z_i \in \mathbb{R}^{n_i - 1}$ e $w \in \mathbb{R}^N$. Como visto anteriormente, o SOCP dual também é convexo. De fato, (7) e (6) têm a mesma forma. Além disso, se eliminarmos as restrições de igualdade, podemos reescrever o SOCP dual na forma do SOCP original (1).

5.2 Métodos de Pontos Interiores do tipo Primal-Dual

Os métodos do tipo primal-dual aplicados a problemas de programação não linear ,

$$\begin{aligned} \text{Minimizar} \quad & f(x) \\ \text{sujeita a} \quad & h(x) = 0, \\ & g(x) \leq 0, \end{aligned} \tag{8}$$

geram iterações (x^k, λ^k, μ^k) que satisfazem as desigualdades $g(x) \geq 0,\ \mu \geq 0$ estritamente. Esta propriedade originou o termo *pontos interiores*. Respeitando essas desigualdades, os métodos evitam soluções falsas, pontos que satisfazem

$$\nabla L(x,\ \lambda,\ \mu) = 0,\ h(x) = 0,\ \mu g(x) = 0,$$

mas não satisfazem

$$g(x) \geq 0,\ \mu \geq 0.$$

Como soluções falsas surgem de forma abundante e nenhuma delas gera informação útil sobre o problema original, o melhor a se fazer é excluí-las da região de busca. Estas são as estratégias usadas nos métodos que descrevemos a seguir.

5.3 O Caminho Central

No processo iterativo de métodos de pontos interiores, são propostas duas modificações nas condições de Karush-Kuhn-Tucker(KKT): conversão das desigualdades em igualdades, usando variáveis de folga $s > 0$, e a perturbação das equações de complementariedade através do parâmetro τ [5]. Desta forma, o novo conjunto de equações é:

$$\nabla_x L(x, \lambda, \mu) = 0, \tag{9a}$$

$$h(x) = 0, \tag{9b}$$

$$g(x) + s = 0, \tag{9c}$$

$$diag(\mu)s - \tau e = 0, \tag{9d}$$

$$(s, \mu, \tau) > 0, \tag{9e}$$

onde $\tau \in \mathbb{R}$, $s \in \mathbb{R}^p$, $e \in \mathbb{R}^p$ e $e = [1, ..., 1]^T$.

O caminho central $\mathcal{C}$ é um arco de pontos estritamente factíveis que desempenha um papel vital na teoria dos algoritmos do tipo primal-dual. Cada ponto $(x, \lambda, \mu) \in \mathcal{C}$ é solução do sistema (9).

Para resolver o sistema (9), utiliza-se o Método de Newton, que fornece um modelo linear para as equações deste sistema em torno do ponto atual e obtém uma direção de busca $(\Delta x, \Delta\lambda, \Delta\mu)$ em dois passos.

Primeiro as direções Δx e $\Delta\lambda$ são encontradas através do sistema:

$$\begin{bmatrix} T & \nabla h(x) \\ \nabla^T h(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} t \\ -h(x) \end{bmatrix}, \tag{10}$$

onde

$$T = \nabla^2_{xx} L + \nabla g \ \mathrm{diag}\left(\tfrac{\mu}{s}\right) \nabla^T g$$
$$t = -\nabla_x L + \nabla g \ \mathrm{diag}(s^{-1})(\tau e + \mathrm{diag}(\mu) g).$$

No segundo passo, as direções Δs e $\Delta\mu$ são obtidas por:

$$\begin{aligned} \Delta s &= -g(x) - s - \nabla^T g(x) \Delta x, \\ \Delta\mu &= -\mu + \ \mathrm{diag}(s^{-1})(\tau e - \ \mathrm{diag}(\mu)\Delta s). \end{aligned} \tag{11}$$

Para assegurar a positividade de s e μ, são utilizados dois parâmentros, $\alpha_p \in (0, 1]$ e $\alpha_d \in (0, 1]$. Desta forma, as variáveis primal e dual para a iteração seguinte são:

$$\begin{aligned} x^{k+1} &= x + \alpha_p \Delta x, \\ s^{k+1} &= s + \alpha_p \Delta s, \\ \lambda^{k+1} &= \lambda + \alpha_d \Delta\lambda, \\ \mu^{k+1} &= \mu + \alpha_d \Delta\mu. \end{aligned}$$

A cada iteração, o parâmetro de perturbação τ é gradualmente reduzido a zero, de forma que as equações (9) se aproximam das condições KKT.

5.4 Método Seguidor de Caminho (*Path-Following Method*)

Um algoritmo *path-following* restringe as iterações para uma vizinhança do caminho central $\mathcal{C}$ e segue para uma solução do problema original. A vizinhança exclui pontos (x, λ, μ) que estão

muito próximos da fronteira da região não negativa. Assim, direções de busca calculadas a partir de qualquer ponto desta vizinhança fazem pelo menos progressos mínimos em direção ao conjunto solução [22].

Ao manter todas as iterações dentro de uma vizinhança, os métodos do tipo seguidor de caminho reduzem todos os produtos $\mu_i g_i(x)$ para zero aproximadamente na mesma taxa.

Alguns métodos do tipo seguidor de caminho escolhem valores conservadores para o parâmetro central σ (isto é, σ apenas ligeiramente inferior a 1). Estes métodos, que são conhecidos como métodos *short-step path-following*, fazem apenas progressos lentos em direção à solução porque uma vizinhança restritiva é necessária para fazê-lo funcionar.

Os métodos *long-step path-following* fazem escolhas menos conservadoras de σ do que os métodos *short-steps*. Como uma consequência, eles precisam realizar uma busca linear ao longo de $(\Delta x,\ \Delta\lambda,\ \Delta\mu)$ para evitar deixar a vizinhança escolhida. Fazendo escolhas convenientes para σ, todavia, métodos long-step podem fazer progressos muito mais rápido que métodos short-step.

5.5 O Método de Barreira

Um método seguidor de caminho muito utilizado é o método de barreira. Este é utilizado para a resolução de problemas com restrições de desigualdade, cujo interior é não vazio, pois trabalha no interior da região factível, utilizando uma função auxiliar que cresce indefinidamente próxima à fronteira e uma sequência decrescente de fatores de barreira.

Os métodos de barreira introduzem as restrições na função objetivo através de um parâmetro de barreira, que impede a aproximação de um ponto factível à fronteira da região factível. Trabalhando no interior dessa região, tais parâmetros geram barreiras que impedem as variáveis de violarem suas fronteiras. Logo, parte-se de um ponto factível e geram-se novos pontos factíveis.

O objetivo é transformar um problema com restrições de desigualdade (8),

$$\begin{aligned} \text{Minimizar} \quad & f(x) \\ \text{sujeita a} \quad & h(x) = 0 \\ & g(x) \leq 0, \end{aligned}$$

em um problema com restrições de igualdade, no qual o método de Newton pode ser aplicado [19].

O primeiro passo é reescrever o problema (8), colocando as restrições de desigualdade implicitamente na função objetivo:

$$\begin{aligned} \text{Minimizar} \quad & f(x) + \sum_{i=1}^{p} -(\tfrac{1}{t}) \log(-g_i(x)) \\ \text{sujeita a} \quad & h(x) = 0, \end{aligned} \tag{12}$$

onde $t > 0$ é um parâmetro que define a precisão da aproximação.

A função $\phi(x) = \sum_{i=1}^{p} -\log(-g_i(x))$ é chamada de *barreira logarítmica* do problema (8). Seu domínio

$$dom\phi = \{x \in \mathbb{R}^n \mid g_i(x) < 0, i = 1, ..., p\},$$

é o conjunto de pontos que satisfazem as restrições de desigualdade de (8) estritamente. Independentemente do valor de t, a função barreira cresce ilimitada se $g_i(x) \to 0$, para qualquer i, isto é,

$$\left(\frac{1}{t}\right) \phi(x) \overset{t\to\infty}{\longrightarrow} \begin{cases} 0, & \text{se } g(x) < 0, \\ \infty & \text{caso contrário.} \end{cases} \tag{13}$$

Quando aumentamos t, a solução $x(t)$ do subproblema apresentado em (12) aproxima-se da solução x^* do problema original.

Considerando $t = \frac{1}{\mu}$, temos a função Lagrangiana associada ao subproblema (12):

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{p} -\mu \log(-g_i(x)) + \sum_{i=1}^{m} \lambda_i h_i(x).$$

Desta forma, podemos escrever o sistema que define o caminho central (9) como [21]:

$$\begin{gathered} f(x) + \sum_{i=1}^{p} -\mu \log(-g_i(x)) + \sum_{i=1}^{m} \lambda_i h_i(x) = 0, \\ h(x) = 0, \\ \text{diag}(\mu)s - \tau e = 0, \\ (s, \mu, \tau) > 0. \end{gathered} \tag{14}$$

Para resolver o sistema (14) utiliza-se o Método de Newton. O vetor direção de busca $[\Delta x, \Delta\lambda, \Delta\mu]$ é utilizado para atualizar as variáveis $x,\ \lambda,\ \mu$ [7] como segue:

$$(x, \lambda, \mu) + \alpha(\Delta x, \Delta\lambda, \Delta\mu),$$

em que o tamanho de passo $\alpha \in (0, 1]$ é escolhido para preservar a positividade de t. Para mais detalhes ver [4], [21].

5.6 Barreira para Cone Quadrático

Definimos a função barreira para cone quadrátido de forma semelhante à função barreira original:

$$\phi(u,t) = \begin{cases} -\log(t^2 - \|u\|^2), & \text{se } \|u\| < t, \\ \infty \quad \text{caso contrário.} \end{cases} \tag{15}$$

Observe que a função ϕ é a função barreira para cone de segunda ordem. De fato, sendo C um cone de segunda ordem, $\phi(u,t)$ é finito se e só se $(u,t) \in C$ (i.e., $\|u\| < t$), e $\phi(u,t)$ converge para ∞ quando (u,t) se aproxima do fronteira de C.

5.7 Método Seguidor de Caminho para SOCP com Escalamento de Nesterov-Todd

Considere um problema SOCP simplificado:

$$\begin{array}{ll} \text{Minimizar} & f^{T'}x \\ \text{sujeita a} & \|Ax\| \leq d, \\ & F'x = g. \end{array} \tag{16}$$

Então novamente associando ao cone C a ordenação parcial em $\mathbb{R}^n$ definida por $x \geq_C y \Leftrightarrow x - y \in C$, vamos reformular o problema (16),

$$\begin{array}{ll} \text{Minimizar} & f^T x \\ \text{sujeita a} & Ax \leq_C d, \\ & Fx = g. \end{array} \tag{17}$$

O dual do problema SOCP (17) é:

$$\begin{array}{rl} \text{Maximizar} & -d^T\mu - g^T\lambda \\ \text{sujeita a} & A^T\mu + F^T\lambda + f = 0, \\ & \mu \geq_C 0. \end{array} \tag{18}$$

As condições KKT do problema (17) são:

$$\begin{array}{l} f + \lambda F + \mu A = 0, \\ Fx = g, \\ Ax \leq_C d, \\ diag(\mu)(Ax - d) = 0. \\ \mu \geq_C 0. \end{array} \tag{19}$$

O caminho central do problema (17) é definido pelo conjunto de pontos que satisfazem o sistema KKT modificado,

$$\begin{array}{c} f + \lambda F + \mu A = 0, \\ Fx = g, \\ Ax - d + s = 0, \\ diag(\mu)s - \tau e = 0, \\ (s, \mu) >_C 0, \\ \tau > 0. \end{array} \tag{20}$$

Reescrevendo as equações (20) na forma matricial, temos:

$$\begin{array}{c} \begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} + \begin{bmatrix} 0 & F^T & A^T \\ F & 0 & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} -f \\ g \\ d \end{bmatrix}, \\ diag(\mu)s = \tau e, \\ (s, \mu) >_C 0, \\ \tau > 0. \end{array} \tag{21}$$

Definição 5.1. *Um escalamento primal-dual W é uma transformação linear*

$$\tilde{s} = W^{-T}s, \quad \tilde{\mu} = W\mu,$$

que mantém o cone e o caminho central invariantes, isto é,

$$s >_C 0 \Leftrightarrow \tilde{s} >_C 0, \qquad \mu >_C 0 \Leftrightarrow \tilde{\mu} >_C 0,$$
$$diag(\mu)s = \tau e \Leftrightarrow diag(\tilde{\mu})\tilde{s} = \tau e.$$

Se W é um escalamento, podemos escrever as equações do caminho central (21) equivalentemente como:

$$\begin{gathered}\begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} + \begin{bmatrix} 0 & F^T & A^T \\ F & 0 & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} -f \\ g \\ d \end{bmatrix}, \\ diag(w_\mu)w_s = \tau e, \\ (s, \mu) >_C 0, \\ \tau > 0.\end{gathered} \tag{22}$$

onde $w_\mu = W\mu$ e $w_s = W^{-T}s$.

Propriedades da função barreira logarítmica (15) fornecem um procedimento para construir um escalamento primal-dual .

O algoritmo de Nesterov-Todd, a seguir, é baseado nas equações de caminho central (22), depois de ser aplicado ao problema um escalamento com uma matriz W obtida a partir da Hessiana da função barreira (15). Para mais detalhes vide [15], [14]

5.7.1 Algoritmo Seguidor de Caminho para SOCP

Dados (s, x, λ, μ).

1. Compute

$$\begin{bmatrix} r_x \\ r_\lambda \\ r_\mu \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} - \begin{bmatrix} 0 & F^T & A^T \\ F & 0 & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix}$$

Pare se $(s,\ x,\ \lambda,\ \mu)$ satisfizer (aproximadamente) as equações KKT (21).

2. Resolva o sistema linear

$$\begin{bmatrix} 0 \\ 0 \\ \Delta s_a \end{bmatrix} - \begin{bmatrix} 0 & F^T & A^T \\ F & 0 & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_a \\ \Delta \lambda_a \\ \Delta \mu_a \end{bmatrix} = - \begin{bmatrix} r_x \\ r_\lambda \\ r_\mu \end{bmatrix}$$

3. Compute

$$\alpha = sup\left\{\alpha \in [0,\ 1] \,|\, (s, \mu) + \alpha(s_a, \mu_a)\right\}$$
$$\sigma = (1 - \alpha)^3.$$

4. Resolva o sistema linear:

$$\begin{bmatrix} 0 \\ 0 \\ \Delta s \end{bmatrix} - \begin{bmatrix} 0 & F^T & A^T \\ F & 0 & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \end{bmatrix} = -(1 - \sigma) - \begin{bmatrix} r_x \\ r_\lambda \\ r_\mu \end{bmatrix}$$

5. Atualize.

$$(s, x, \lambda, \mu) = (s, x, \lambda, \mu) + \alpha(\Delta s, \Delta x, \Delta \lambda, \Delta \mu)$$

onde $\alpha = sup\left\{\alpha \in [0,\ 1] \,|\, (s, \mu) + \alpha(s_a, \mu_a)\right\}$

Compute a matriz de escalameto W para s e μ.

5.7.2 Inicialização

Quando os pontos iniciais $(x_0,\ s_0,\ \lambda_0,\ \mu_0)$ não são especificados pelo usuário, então eles são selecionados da seguinte forma. A variável primal incial será a solução do problema de quadrados mínimos:

$$\begin{array}{rl} \text{Minimizar} & \|Ax - b\| \\ \text{sujeita a} & c^T x = d. \end{array}$$

O valor inicial de s_0 é calculado do resíduo $\tilde{s} = Ax_0 - b$, como segue

$$s_0 = \begin{cases} \tilde{s} & \text{se } \alpha_p < 0, \\ \tilde{s} + (1 + \alpha_p)\, e & \text{caso contrário,} \end{cases}$$

onde $\alpha_p = \inf\{\alpha \mid \tilde{s} + \alpha e \geq 0\}$. Os valores x_0 e $\tilde{s}$ podem ser calculados através do sistema linear:

$$\begin{bmatrix} 0 & F^T & A^T \\ F & 0 & 0 \\ A & 0 & -I \end{bmatrix} \begin{bmatrix} x_0 \\ \lambda \\ -\tilde{s} \end{bmatrix} = \begin{bmatrix} 0 \\ d \\ b \end{bmatrix}.$$

As variáveis duais iniciais λ_0 e μ_0 são calculadas resolvendo o problema de norma mínima:

$$\begin{aligned} &\text{Minimizar} && \|\mu\| \\ &\text{sujeita a} && A^T\mu + c^T\lambda + f = 0. \end{aligned} \tag{23}$$

Se a solução é λ_0 e $\tilde{\mu}$, então usamos λ_0 como valor inicial de λ, e

$$\mu_0 = \begin{cases} \tilde{\mu} & \text{se } \alpha_p < 0, \\ \tilde{\mu} + (1 + \alpha_p)\, e & \text{caso contrário,} \end{cases}$$

onde $\alpha_p = \inf\{\alpha \mid \tilde{\mu} + \alpha e \geq 0\}$, como valor inicial de μ. O problema de norma mínima (23) é equivalente ao sistema linear

$$\begin{bmatrix} 0 & F^T & A^T \\ F & 0 & 0 \\ A & 0 & -I \end{bmatrix} \begin{bmatrix} x_0 \\ \lambda_0 \\ -\tilde{\mu} \end{bmatrix} = \begin{bmatrix} -c \\ 0 \\ 0 \end{bmatrix}.$$

Desta forma, podemos encontrar os valores λ_0 e $\tilde{\mu}$ resolvendo este sistema linear.

6 Testes Numéricos

Apresentamos aqui experimentos numéricos realizados com o objetivo de avaliar a eficiência do uso de um software específico para problemas SOCP, quando comparado a um software genérico para resolução de problemas de otimização.

A avaliação teve como foco um problema teórico, com dimensões variáveis, e duas aplicações reais. O software específico utilizado para execução de problemas SOCP foi o CVXOPT *Python Software for Convex Optimization* [18], enquanto a função `fmincon` *Find Minimum of Constrained Nonlinear Multivariable Function* [10], nativa do Matlab [11], foi utilizada como o software genérico.

Os problemas preliminares e o problema teórico são mostrados de duas formas diferentes, primeiro em seus formatos originais, em seguida no seu formato SOCP. Estes problemas são solucionados de três maneiras distintas: formato original em `fmincon`, formato SOCP em `fmincon` e formato SOCP em CVXOPT. As aplicações reais são executadas sempre em seu formato SOCP, tanto em `fmincon` quanto em CVXOPT.

Nestas simulações, avaliamos a quantidade de testes bem sucedidos, o valor da função objetivo e a quantidade de iterações. Como se tratam de dois softwares distintos - Matlab e Python, para avaliar o tempo utilizamos uma padronização. Todos os valores de tempo foram divididos pelo menor valor de tempo de cada software, sendo assim, o menor tempo, tanto do `fmincon` como do CVXOPT, estão representados por 1.

Foram utilizados o mesmo ponto inicial, vetor nulo, e os mesmos critérios de parada nos dois softwares: gap relativo: 10^{-6}; tolerância das restrições não lineares: 10^{-6}; máximo de iterações: 5.000, no problema QCQP, 25.000 na aplicação de arranjo de antenas e 25.000 na aplicação garra de múltiplos dedos (GRASP). Em CVXOPT zeramos o critério de gap absoluto.

Os experimentos foram realizados em um computador com processador Intel Core i5 750 (2.67GHz) com 4GB de RAM.

7 Problemas Teóricos

7.1 QCQP

Nesta seção apresentamos problemas QCQP com variações de dimensão de matrizes e de quantidade de restrições. Geramos problemas QCQP teóricos, com objetivo de comparar os problemas em seus formatos originais e no formato SOCP.

Foram avaliados 1080 problemas, com 2, 4, 6, 8, 10 ou 12 restrições. Para cada restição variamos as dimensões de x em 2, 4, 6, 8, 10 e 12. Utilizamos 30 sementes diferentes para gerar todos os dados aleatoriamente. Mostramos o desempenho dos softwares através de gráficos.

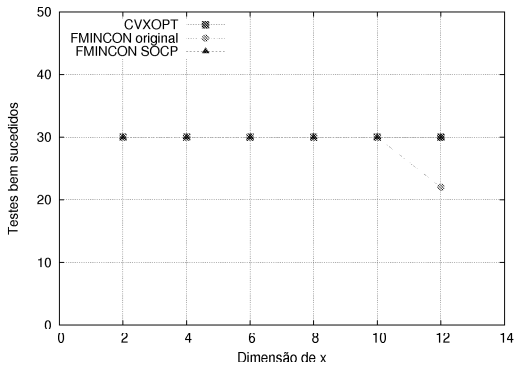

Figura 1: *Número de testes bem sucedidos X dimensão n de x para os problemas QCQP com número de iterações = n.*

7.1.1 Convergência

Começamos com a análise da convergência. A Figura 1 nos mostra problemas que possuem quantidade de restrições igual a dimensão de x. Para estes casos, CVXOPT e `fmincon` com problema no formato SOCP realizaram todas as execuções com sucesso, já `fmincon` usando o formato original não obteve convergência em alguns problemas com 12 restrições. A Figura 2 apresenta o número de testes bem sucedidos de todos os problemas com variações de restrições e dimensões. Podemos observar pela Figura 2(b) que CVXOPT convergiu em todos os problemas; `fmincon` usando o formato SOCP falhou em alguns problemas com dimensões maiores ou iguais a 8 e `fmincon`, usando o formato original falhou em alguns problemas de todas as dimensões. Concluímos que CVXOPT foi o mais robusto.

7.1.2 Iterações

Para análise da quantidade de iterações, utilizamos apenas os dados dos problemas que obtiveram testes bem sucedidos. Primeiro analisamos problemas que possuem quantidade de restrições igual a dimensão de x. Através da Figura 3 verificamos que para estes casos `fmincon` usando o formato original, apresenta um desempenho muito abaixo dos demais, os quais, por sua vez, têm um comportamento parecido. Já analisando a Figura 4, que nos mostra todas as variações de restrições e dimensões, podemos perceber que existe uma diferença significativa na quantidade de iterações realizadas por `fmincon`, formato SOCP, e

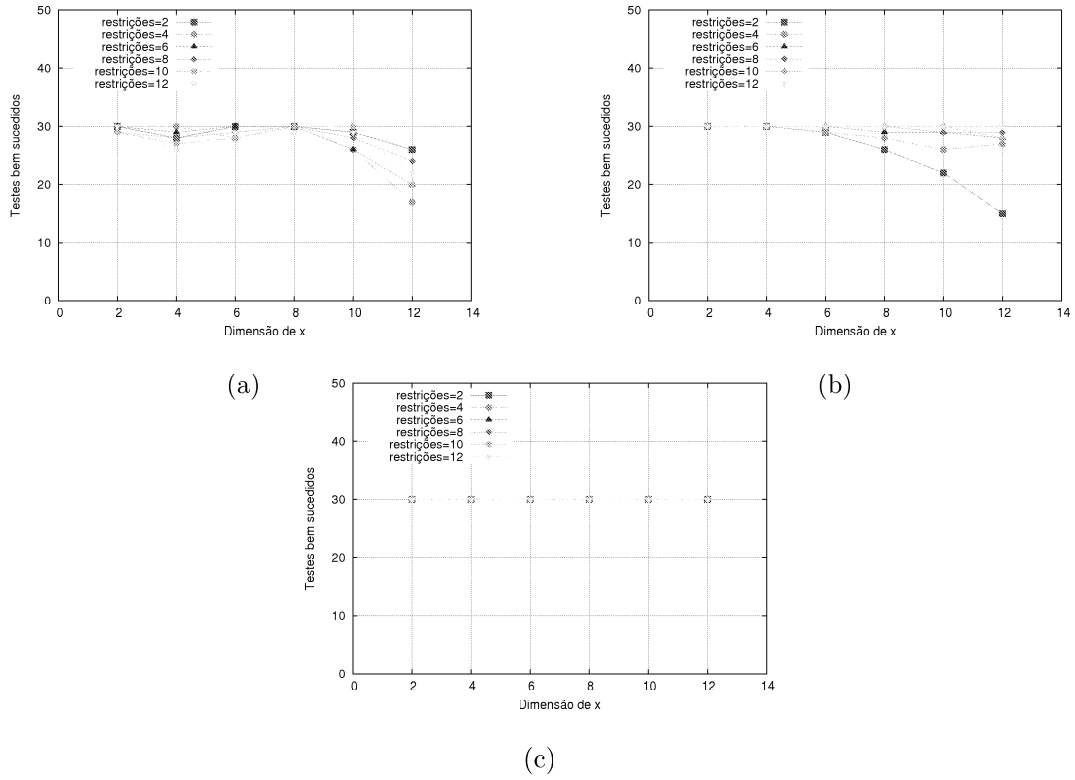

Figura 2: *Variação da dimensão de x e da quantidade de restrições em (a) problema original em* `fmincon`*, (b)problema SOCP em* `fmincon`*, (c) CVXOPT*

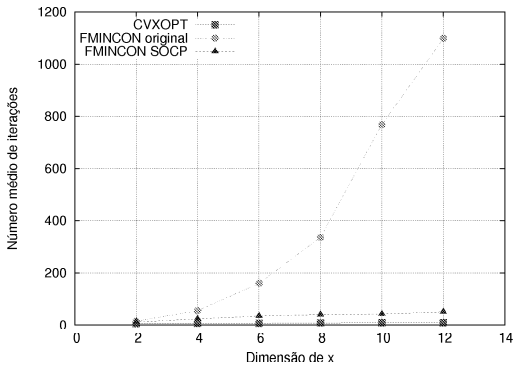

Figura 3: *Número de iterações X dimensão n de x para os problemas QCQP com número de iterações = n.*

por CVXOPT. Este último tem o melhor desempenho.

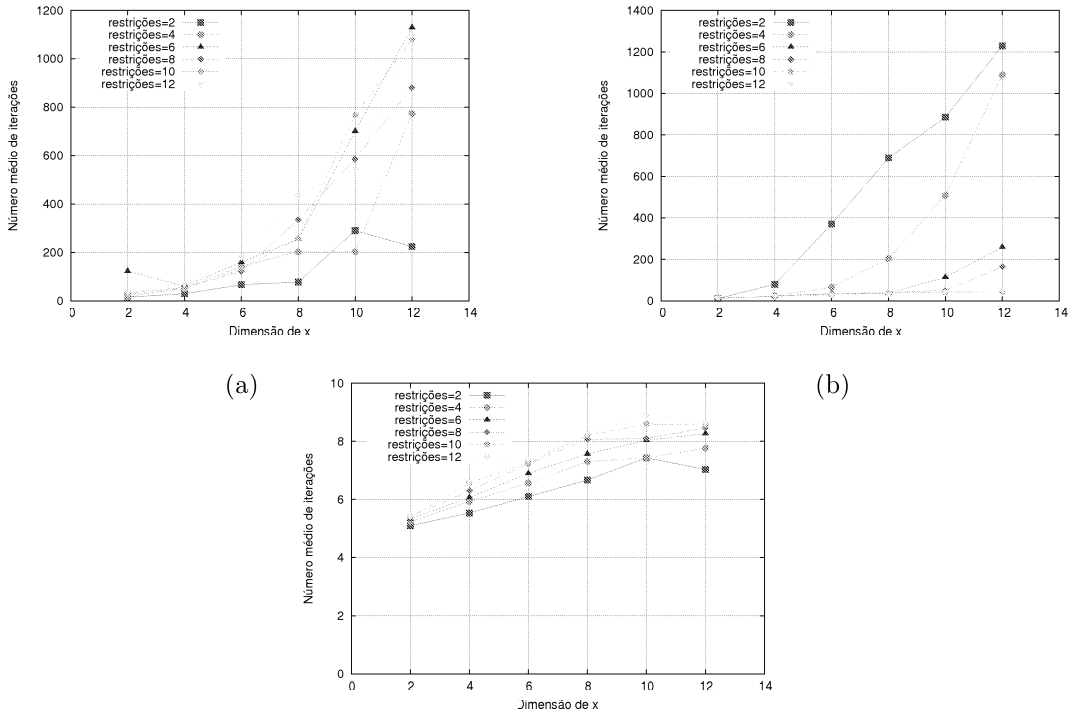

Figura 4: *Variação da dimensão de x e com variação das restrições em (a) problema original em* `fmincon`*, (b) problema SOCP em* `fmincon`*, (c) CVXOPT*

7.1.3 Tempo

Para fazer a análise do tempo, usamos a padronização descrita no início deste capítulo e novamente apenas os dados dos problemas que obtiveram testes bem sucedidos. Através da Figura 5 analisamos os problemas que possuem quantidade de restrições igual a dimensão de x e assim verificamos que para estes problemas `fmincon` usando o formato original apresenta um desempenho muito abaixo dos demais, que apresentam um comportamento parecido. Já analisando o gráfico 6, podemos perceber que ao variarmos a dimensão nas diversas restrições, `fmincon` usando o formato SOCP apresenta uma diferença significativa na quantidade de iterações quando comparado a CVXOPT. Este último tem o melhor desempenho.

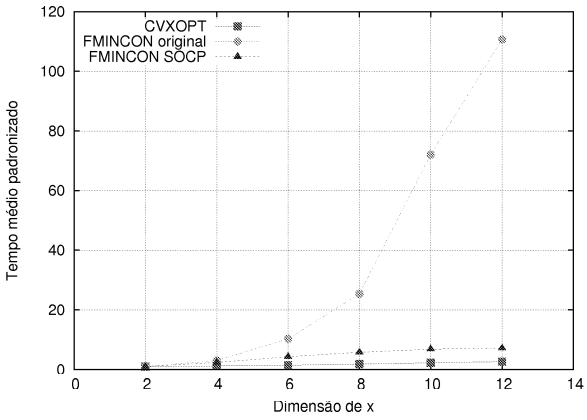

Figura 5: *Tempo de processamento X dimensão n de x para os problemas QCQP com número de iterações = n.*

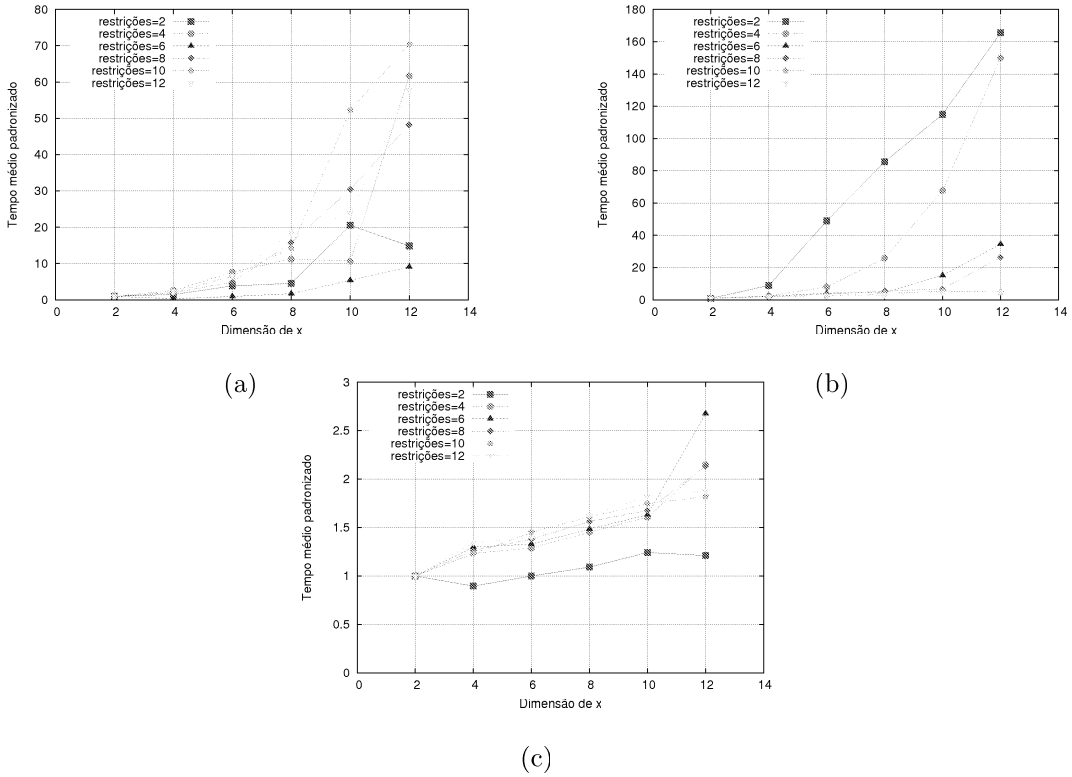

Figura 6: *Variação da dimensão de x e com variação da quantidade de restrições em (a) problema original em* `fmincon`*, (b) problema SOCP em* `fmincon`*, (c) CVXOPT*

8 Problemas Reais

8.1 Arranjo de Antenas

8.1.1 Descrição do problema

Em um arranjo de antenas, as emissões de vários sensores são linearmente combinadas para produzir uma emissão mais apurada. A saída do arranjo tem um padrão direcional, que depende dos pesos relativos ou fatores escalares utilizados no processo combinado. O objetivo é estabalecer os pesos que satisfaçam os requisitos do feixe de ondas específico a ser emitido de maneira ótima.

Consideremos o modelo mais simples, um arranjo onidirecional de antenas em um plano, nas posições $(x_j, y_j),\ j = 1, ...n$. Uma única onda, de freqüência ω, incide com ângulo θ. Assumimos que o comprimento de onda é $\lambda = 2\pi$. Esta onda incidente induz no j-ésimo sensor um sinal $exp(i(x_j cos\theta + y_j sen\theta - \omega t))$, onde $i = \sqrt{-1}$. Este sinal é demodulado, isto é, multiplicado por $e^{i\omega t}$ e então multiplicado por um fator complexo $w_j \in \mathbb{C}$. Desta forma temos,

$$\begin{aligned} y_j &= w_j exp(i((x_j cos\theta + y_j sen\theta)) \\ &= (w_{re,j} cos\gamma_j(\theta) - w_{im,j} sen\gamma_j(\theta)) + i(w_{re,j} sen\gamma_j(\theta) + w_{im,j} cos\gamma_j(\theta)), \end{aligned}$$

onde $\gamma_j(\theta) = x_j cos\theta + y_j sen\theta$. A saída do arranjo de antenas é a soma das saídas ponderadas dos sensores:

$$y(\theta) = \sum_{j=1}^{n} y_j(\theta),$$

Para um dado conjunto de pesos, estas saídas combinadas são uma função do ângulo de chegada θ da onda. O objetivo do problema é selecionar os pesos w_i que alcancem uma direção padrão $y(\theta)$ desejável.

A propriedade fundamental é que, para qualquer θ, $y(\theta)$ é uma função linear do vetor peso w. Esta propriedade é verdadeira para uma classe muito ampla de problemas de arranjos e utilizaremos o caso $y(\theta) = a(\theta)w$ para algum vetor linha complexo $a(\theta)$.

Como exemplo de problema de um projeto simples, podemos utilizar a normalização $y(\theta_t) = 1$, onde θ_t é chamado de direção de destino. Se tivermos como objetivo fazer um arranjo relativamente insensível a ondas provenientes de outras direções, teremos $|\theta - \theta_t| \geq \Delta$, onde 2Δ é chamado de largura de feixe do padrão.

Para minimizar a maior sensibilidade de ondas fora do feixe, resolvemos o problema:

$$\begin{array}{ll} \text{Minimizar} & \max\limits_{|\theta-\theta_t|>\Delta} |y(\theta)| \\ \text{sujeita a} & y(\theta_t) = 1. \end{array} \tag{24}$$

Este problema pode ser aproximado por um problema SOCP discretizando o ângulo θ, por exemplo, considerando $\theta_1, ..., \theta_m$, onde $m >> n$. Assumimos que a direção de destino é um dos ângulos, tomemos $\theta_t = \theta_k$. Podemos expressar o aranjo padrão por:

$$\tilde{y} = Aw$$

onde $\tilde{y} \in \mathbb{C}^{m \times n}$, e

$$\tilde{y} = \begin{bmatrix} y(\theta_1) \\ \vdots \\ y(\theta_m) \end{bmatrix}, \; A = \begin{bmatrix} a(\theta_1) \\ \vdots \\ a(\theta_m) \end{bmatrix}.$$

O problema (24) pode ser expresso da seguinte forma:

$$\begin{array}{ll} \text{Minimizar} & t \\ \text{sujeita a} & |y(\theta_i)| \leq t \\ & |\theta_i - \theta_k| > \Delta \\ & y(\theta_k) = 1, \end{array} \tag{25}$$

o qual torna-se um SOCP quando é expresso em termos das partes reais e imaginárias das variáveis e dos dados [9].

8.1.2 Resultados experimentais

Geramos 870 problemas, com $n = 4, ..., 32$ sensores e utilizamos 30 sementes diferentes para gerar aleatoriamente os dados. Mostramos o desempenho dos softwares através de gráficos.

8.1.3 Convergência

A Figura 7 nos mostra que CVXOPT convergiu em todos os problemas testados, mostrando robustez. Já `fmincon` convergiu para uma quantidade menor de problemas com mais de 15 sensores.

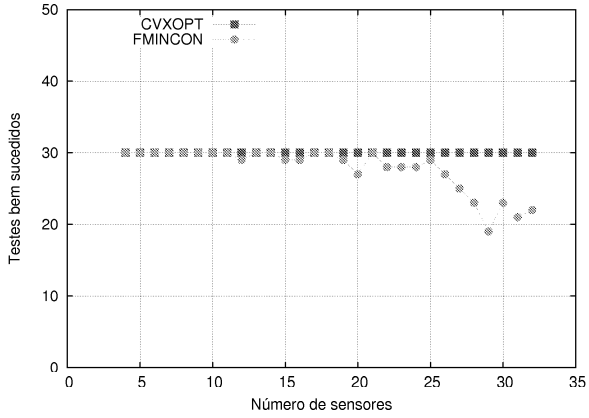

Figura 7: *Testes bem sucedidos para os problemas de arranjo de antenas, com variação do número de sensores.*

8.1.4 Iterações

Para análise da quantidade de iterações, utilizamos apenas os dados dos problemas que obtiveram testes bem sucedidos. Pela Figura 8, podemos perceber que o número de iterações de CVXOPT mantem-se aproximadamente constante e de `fmincon` varia de forma desordenada, sendo sempre maior que no primeiro caso.

Na Figura 9, podemos fazer uma leitura mais apurada do número de iterações de CVXOPT. Percebemos que o número de iterações aumenta, mas não ultrapassa 20 iterações.

8.1.5 Tempo

Para fazer a análise do tempo, usamos a padronização descrita no início deste capítulo e novamente apenas os dados dos problemas que obtiveram testes bem sucedidos. Através da Figura 10 podemos perceber que o tempo de CPU em CVXOPT mantem-se baixo, quase

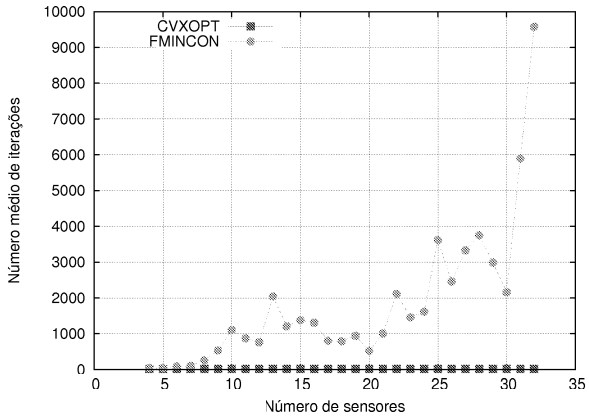

Figura 8: *Número de iterações para os problemas de arranjo de antenas com variação do número de sensores.*

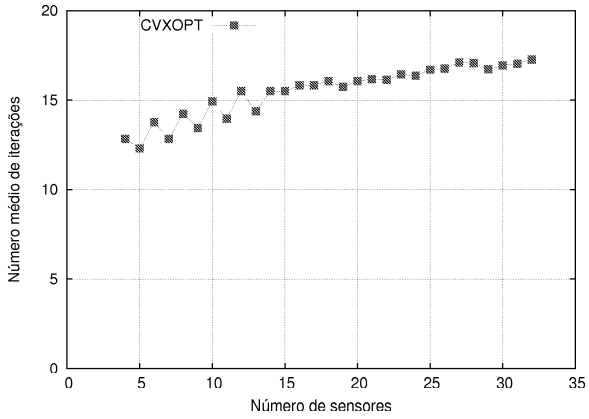

Figura 9: *Número de iterações para os problemas de arranjo de antenas com variação do número de sensores.*

constante. Já `fmincon` apresenta variação no tempo de processamento a partir de 8 sensores, sendo este tempo sempre maior que o registrado por CVXOPT.

Na Figura 11, podemos fazer uma leitura mais apurada do tempo de CPU em CVXOPT. Percebemos que o tempo aumenta, mas não ultrapassa 25 vezes o tempo da primeira iteração ao passo que na Figura 10 ultrapassa 9.000 vezes o tempo da primeira iteração.

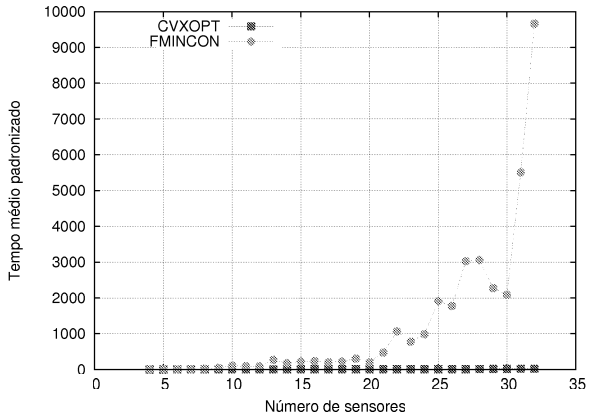

Figura 10: *Tempo de processamento para os problemas de arranjo de antenas com variação do número de sensores.*

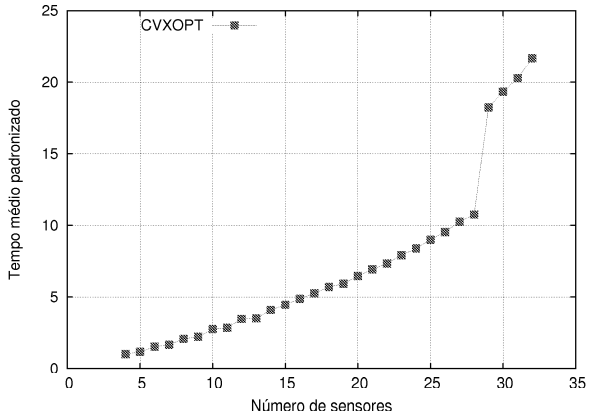

Figura 11: *Tempo de processamento para os problemas de arranjo de antenas com variação do número de sensores.*

8.2 Garras Múltiplos Dedos - *Grasp*

8.2.1 Descrição do problema

Este problema trata de uma garra múltiplos dedos, também conhecida por mão mecânica, que deve agarrar um objeto de forma estável, o mais suave possível.

Consideramos um corpo rígido segurado por N dedos de um robô. Para simplificar as fórmulas assumimos que o centro de massa do corpo está na origem. Os dedos exercem forças de contato em pontos dados: $p_1, ..., p_N \in \mathbb{R}^3$. O vetor normal à superfície no i-ésimo

ponto de contato é dado pelo vetor unitário $v_i \in \mathbb{R}^3$ e a força aplicada no ponto é dada por $F_i \in \mathbb{R}^3$.

Cada força de contato F_i é decomposta em duas partes: uma componente $v_i^T F_i v_i$ normal à superfície, e uma componente $(I - v_i v_i^T)F_i$, a qual é tangente à superfície. Assumiremos que a componente tangente ocorre pelo atrito estático e sua magnitude não pode exceder o produto da magnitude da componente normal com o coeficiente de atrito $\mu > 0$, isto é,

$$\left\| \left(I - v_i v_i^T\right) F_i \right\| \leq \mu v_i^T F_i, i = 1, ..., N. \tag{26}$$

Essas restrições são restrições de cone quadrático nas variáveis F_i [9].

Por fim, assumimos que forças e torques externos agem sobre o corpo. Isso é equivalente a uma única força externa F_{ext} agindo na origem (que é o centro de massa do corpo) e um torque externo T_{ext}. O equilíbrio estático do corpo é caracterizado pelas equações lineares

$$\begin{aligned} \sum_i^N F_i + F_{ext} &= 0, \\ \sum_i^N p_i \times F_i + T_{ext} &= 0. \end{aligned} \tag{27}$$

O objetivo do problema é encontrar forças de contato F_i que satisfaçam as restrições de atrito (26), as restrições de equilíbrio estático (27), e certos limites nas forças de contato, por exemplo, um limite superior $v_i^T F_i \leq F_{max}$ na componente normal.

É possível selecionar um conjunto particular de forças por um certo critério de otimização. Por exemplo, podemos calcular a pegada mais suave possível, isto é, o conjunto de forças F_i que realiza uma pegada estável e minimiza o máximo da força normal nos pontos de contato, resolvendo o SOCP

$$\begin{aligned} \text{Minimizar} \quad & t \\ \text{sujeita a} \quad & v_i^T F_i \leq t, \quad i = 1, ..., N, \\ & \left\| (I - v_i v_i^T F_i \right\| \leq \mu v_i^T F_i, i = 1, ..., N, \\ & \sum_i^N F_i + F_{ext} = 0, \\ & \sum_i^N p_i \times F_i + T_{ext} = 0. \end{aligned} \tag{28}$$

Para mais detalhes de otimização da força de uma mão mecânica, vide [20] e [8].

8.2.2 Resultados experimentais

Geramos 870 problemas, com $N = 4, ..., 32$ dedos e utilizamos 30 sementes diferentes para gerar aleatoriamente os dados. Mostramos o desempenho dos softwares através de gráficos.

8.2.3 Convergência

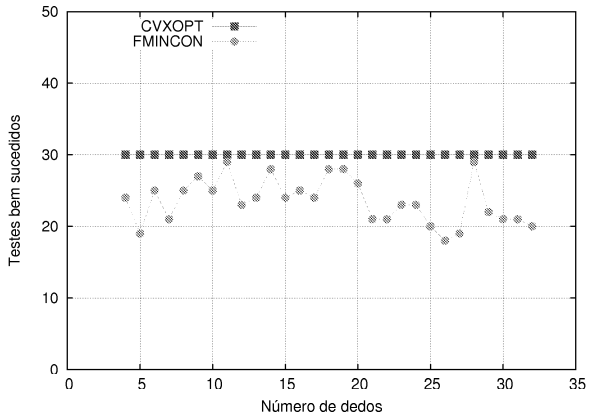

Figura 12: *Testes bem sucedidos para os problemas de garras múltiplos dedos, com variação do número de dedos.*

A Figura 12 nos mostra que CVXOPT convergiu em todos os problemas testados, mostrando robustez. Já `fmincon` falhou em alguns dos problemas.

8.2.4 Iterações

8.2.5 Tempo

Através da Figura 15, podemos perceber que o tempo de CPU registrado por CVXOPT cresce de forma ordenada até 25 dedos, e o de `fmincon` apresenta-se de forma desordenada desde o princípio. Nestes primeiros problemas o tempo de CPU de CVXOPT é menor que o tempo de `fmincon`. No entanto, nos problemas com mais de 25 dedos, o tempo de processamento de CVXOPT tornou-se maior se comparado ao de `fmincon`.

Para análise da quantidade de iterações, utilizamos apenas os dados dos problemas que obtiveram testes bem sucedidos. Pela Figura 13, podemos perceber que o número de iterações

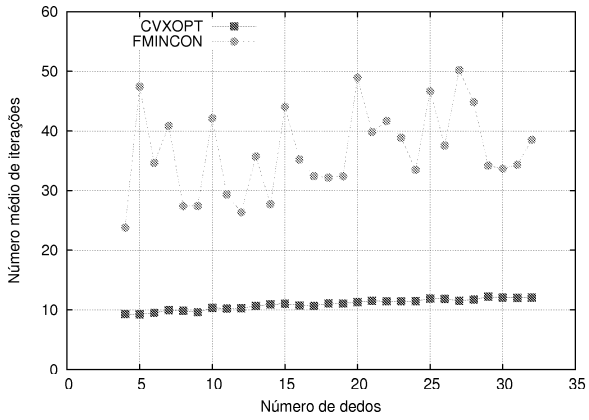

Figura 13: *Número de iterações para os problemas de garras múltiplos dedos com variação do número de dedos.*

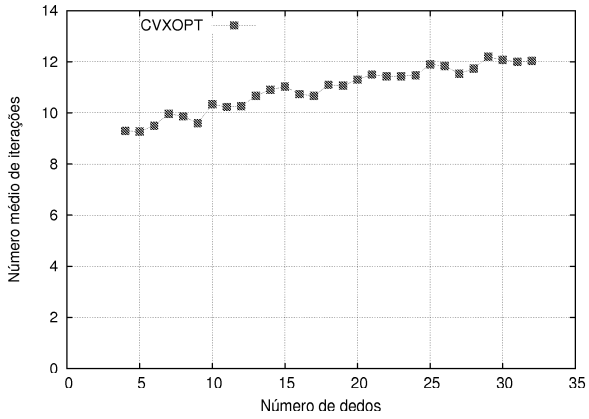

Figura 14: *Número de iterações para os problemas de garra múltiplos dedos com variação do número de dedos.*

de CVXOPT mantem-se aproximadamente constante, enquanto que em `fmincon` varia de forma desordenada. Outra informação deste gráfico é a diferença entre o número de iterações nos dois softwares: CVXOPT apresenta um número de iterações bem inferior a `fmincon`.

Na Figura 14 podemos fazer uma leitura mais apurada do número de iterações de CVXOPT. Percebemos que o número de iterações aumenta, mas não ultrapassa 14 iterações.

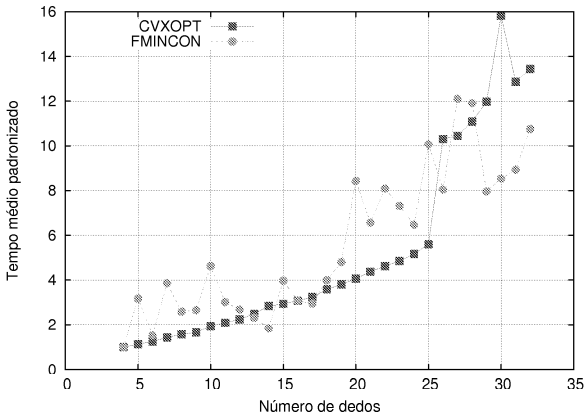

Figura 15: *Tempo de processamento para os problemas de garras múltiplos dedos com variação do número de dedos.*

9 Considerações Finais

Problemas SOCP abrangem diversos problemas de programação matemática como casos especiais e possuem uma formulação simples. Essa dissertação apresentou um estudo teórico sobre problemas de programação não linear e problemas SOCP, além de testes numéricos que possibilitaram uma análise dos algoritmos para problemas SOCP baseando-se tanto em problemas teóricos como práticos.

Foi feito um estudo teórico que analisou a transformação de diversos problemas: soma de normas, máximo de normas, programação linear, QCQP (*Quadratically Constrained Quadratic Programming*), média harmônica, superfície mínima, programação linear robusta e quadrados mínimos robusto em problemas SOCP. Esta análise mostrou que a transformação é simples, o que implica que a escolha do formato pode se basear apenas na facilidade de resolução do problema.

Foi feito também um estudo teórico sobre métodos usados em problemas gerais de programação não linear e métodos usados especificamente em problemas SOCP.

Para os testes numéricos foram analisados 1080 problemas teóricos e 1740 problemas práticos. Os problemas teóricos analisados foram problemas QCQP, que incluem programação linear e programação quadrática como casos especiais. Os práticos foram problemas sobre ar-

ranjo de antenas e garras múltiplos dedos. Um arranjo de antena se caracteriza pela utilização de vários sensores em conjunto para uma captação ou emissão mais apurada (se comparado a uma antena simples) de sinais vindos de diferentes direções. Uma garra múltiplos dedos, também conhecida por mão mecânica, deve agarrar um objeto de forma estável, satisfazendo alguns requisitos.

Nos problemas teóricos foi feita a comparação do formato original, QCQP, com o formato SOCP. Com isso pudemos avaliar experimentalmente a diferença entre otimizar um problema em seu formato original e em seu formato SOCP, utilizando uma ferramanta para problemas gerais de otimização, e a diferença na otimização de um problema SOCP usando uma ferramenta genérica e usando uma ferramenta específica para SOCP. Utilizamos os solvers `fmincon`, nativo do MatLab, como ferramenta genérica e o CVXOPT, biblioteca do Python, como ferramenta específica para SOCP. Os resultados obtidos foram analisados de uma maneira detalhada.

A análise ficou concentrada nos requisitos robustez, número de iterações e variação do tempo de CPU com o aumento da dimensão dos problemas. Observou-se que a ferramenta específica de SOCP obteve melhores resultados em todos os problemas e em todos os requisitos estudados.

Através da análise da convergência observamos que CVXOPT mostrou robustez, realizando todos os experimentos com sucesso e `fmincon` mostrou falhas de convergência em problemas dos três tipos.

Para análise da quantidade de iterações, utilizamos apenas os dados dos problemas que obtiveram testes bem sucedidos. Através de gráficos pode-se perceber que o número de iterações de CVXOPT mantém-se aproximadamente constante, enquanto que em `fmincon` varia de forma desordenada, sendo sempre maior que no primeiro caso.

Para fazer a análise do tempo de CPU, usamos a padronização descrita no início do capítulo 5 e novamente apenas os dados dos problemas que obtiveram testes bem sucedidos. Nos problemas QCQP e de arranjo de antenas o tempo de CPU em CVXOPT manteve-se baixo, quase constante, enquanto `fmincon` apresentou variação no tempo de processamento, sendo este tempo sempre maior que o registrado por CVXOPT. Nos problemas de garras múltiplos dedos o tempo padronizado de CVXOPT aumentou de forma semelhante ao tempo

de `fmincon`.

Diante dos resultados obtidos com os testes numéricos, podemos concluir que é interessante usar SOCP sempre que possível.

Como trabalhos futuros planejamos avaliar outras ferramentas específicas para SOCP tais como CVX e SeDuMi e estudar outros problemas, especialmente problemas reais que tenham duas formulações, a formulação original e a formulação SOCP.

Referências

[1] Alizadeh, F. ; Goldfarb, D. *Second-order cone programming.* Mathematical Programming, 2001.

[2] Bazaraa, M. S. Sherali; H. D.; Shetty, C. M. Nonlinear Programming - *Theory and Algorithms.* John Wiley and Sons, Inc., New Jersey, 3rd edition, 2006.

[3] Boyd, S. ; Crusius, C.; Hanson, A. *Control applications of nonlinear convex programming.* Journal of Process Control, 1998.

[4] Byrd, R. H.; Hribar, M. E.; Nocedal, J. *An interior point algorithm for large scale nonlinear programming.* SIAM Journal on Optimization, 1999.

[5] Castronuovo, E. D. ; Campagnolo, J. M. ; Salgado, R. *A largest-step central-path algorithm applied to the optimal power flow problem.* SBA Controle e Automação, 2000.

[6] Grant, M.; Boyd, S. *Matlab software for disciplined convex programming*, version 1.21. URL: http://cvxr.com/cvx/, 2011. Acessado em: 11/04/2011.

[7] Guardia, L. E. T. *Método primal-dual para otimização não linear usando algoritmos de decomposição.* X Simpósio de Pesquisa Operacional e Logística da Marinha - SPOLM, 2007.

[8] Han, L. ; Trinkle, J. C.; Li, Z. *Grasp analysis as linear matrix inequality problems.* IEEE Transactions on Robotics and Automation, 1998.

[9] Lobo, M. S. ; Vandenberghe, L. ; Boyd, S. ; Lebret, H. *Applications of second-order cone programming*. Linear Algebra and its Applications, 1998.

[10] MathWorks. *Find minimum of constrained nonlinear multivariable function*. URL: http://www.mathworks.com/help/toolbox/optim/ug/fmincon.html, 2011. Acessado em: 11/04/2011.

[11] MATLAB. *version 7.11.0.584 (R2010b)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[12] Nemirovski, A. ; Ben-Tal, A. *Lectures on Modern Convex Optimization, Analysis, Algorithms, and Engeineering Applications*. Society for Industrial and Applied Mathematics (SIAM), USA, 2001.

[13] Nesterov, Y. ; Nemirovski, A. *Interior-Point Polynomial Algorithms in Convex Programming*. Siam Studies in Applied Mathematics, Philadelphia, 1994.

[14] Nesterov,Y.E.; Todd, M.J. *Primal-dual interior-point methods for self-scaled cones*. SIAM Journal on Optimization, 1995.

[15] Nesterov,Y.E.; Todd, M.J. *Self-scaled cones and interior-point methods in nonlinear programming*. In Working Paper, CORE, Catholic University of Louvain, Louvain-laNeuve, 1997.

[16] Srirangarajan, S.; Tewfik, A.; Luo, Z.-Q. *Distributed sensor network localization using socp relaxation*. IEEE Transaction on wireless Comunication, 7, 2008.

[17] Sturm, J. F. *Sedumi*. URL: http://sedumi.ie.lehigh.edu, 2010. Acessado em: 11/04/2011.

[18] Vandenberghe, L. *Python software for convex optimization*. URL: http://abel.ee.ucla.edu/cvxopt/, 2010. Acessado em: 11/04/2011.

[19] Vandenberghe, L.; Boyd, S. *Convex Optimization*. Cambridge University Press, New York, 1st edition, 2004.

[20] Vanderberghe, L. ; Boyd, S. *Socp: Software for second-order cone programming.* Laboratory Stanford University, 1997.

[21] Waltz, R. A.; Morales, J. L.; Nocedal, J.; Orban, D. *An interior algorithm for nonlinear optimization that combines line search and trust region steps.* Mathematical Programming 107, 2006.

[22] Wright, S. J. *Primal-Dual Interior-Point Methods.* Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[23] Xue, G.; Ye, Y. *An efficient algorithm for minimizing a sum of euclidean norms with applications.* SIAM Journal on Optimization, 1997.