Controle Dinâmico da Inviabilidade para Programação Não Linear

Abel Soares Siqueira
Supervisor: Francisco A. M. Gomes

Resumo

O método de Controle Dinâmico da Inviabilidade, recentemente estendido no doutorado em progresso de Abel Soares Siqueira para incluir restrições de desigualdade, mostrou-se um competidor promissor entre os métodos para programação não linear. A implementação do novo método, denominada **DCICPP**, foi comparada com o método IPOPT e obteve um bom desempenho. O DCICPP já consegue encontrar eficientemente a solução de 698 problemas do CUTEr em 2 horas. Neste projeto pretendemos estudar maneiras de aumentar ainda mais a robustez do DCICPP, avaliando outras estratégias para resolver os subproblemas. Também queremos melhorar aspectos práticos do algoritmo, por exemplo, implementar uma maneira de lidar com as variáveis fixas no problema e avaliar possibilidades de lidar com Jacobianas mal condicionadas. Esse projeto é uma extensão da tese de doutorado de Abel Soares Siqueira, da Matemática Aplicada, pelo IMECC, com previsão de defesa para Novembro de 2013.

Keywords: Nonlinear Programming, Constrained Opmitization, Composite-step Methods.

Dynamic Control of Infeasibility for Nonlinear Programming

Abel Soares Siqueira
Supervisor: Francisco A. M. Gomes

Abstract

The Dynamic Control of Infeasibility method, recently expanded in the Ph.D. in progress of Abel Soares Siqueira to include inequality constraints, proved to be a good choice amongst the methods for nonlinear programming. The method's implementation, called DCICPP, was compared to the IPOPT method and achieve a good performance. The DCICPP method can already find efficiently the solution of 698 problem of CUTEr in 2 hours. In this project, we intend to study ways of increasing even more the strength of the DCICPP method, considering other strategies to solve the subproblems. We also want to improve the practical aspects of the algorithm, for instance, accepting fixed variables in the problem, and pondering possibilities to handle ill-conditioned Jacobian matrices. This project is an extension of the Ph.D. of Abel Soares Siqueira, in Applied Mathematics, from the IMECC, with defense expected in November, 2013.

Keywords: Nonlinear Programming, Constrained Opmitization, Composite-step Methods.

1 Introdução

Muitos métodos desenvolvidos para resolver o problema de programação não-linear são baseados na ideia de passos compostos. Em geral, um tipo de passo, chamado de passo normal, investe na factibilidade primal enquanto o outro, chamado de passo tangente, na factibilidade dual. Os trabalhos [3, 4, 5, 14, 16, 18, 22] também seguem essa linha. Nesse tipo de método é necessário alguma maneira de limitar o tamanho ou aceitação desses passos, para que o objetivo de um não destrua o resultado do outro. Uma estratégia utilizada tradicionalmente é a de função de mérito, onde uma função é criada utilizando uma combinação da função objetivo e de alguma penalização sobre o valor das restrições.

O método de Controle Dinâmico da Inviabilidade proposto por Gomes and Bielschowsky [19] define uma nova estratégia para a aceitação dos passos. Essa estratégia, chamada de Cilindros de Confiança, permite que os passos tangentes tenham grande magnitude longe da solução, e também permitem fazer mais de um passo normal para encontrar um ponto suficientemente próximo ao conjunto factível. Esse método, originalmente proposto para problemas de igualdade, foi estendido para problemas gerais de programação não linear na tese de Abel Soares Siqueira, permitindo que o problema também tenha restrições não lineares de desigualdade e limitantes para as variáveis.

2 Motivação e Objetivos

Considere o problema de programação não linear na forma

$$\begin{array}{rlrcl} \min & f(x) & & & \\ \text{suj. a} & & c_E(x) & = & 0, \\ & c_L & \leq \quad c_I(x) & \leq & c_U, \\ & b_L & \leq \qquad x & \leq & b_U, \end{array}$$

onde $f : \mathbb{R}^n \to \mathbb{R}, c_E : \mathbb{R}^n \to \mathbb{R}^{m_E}, c_I : \mathbb{R}^n \to \mathbb{R}^{m_I} \in \mathcal{C}^2$, $c_L, c_U \in \mathbb{R}^{m_I}$ e $b_L, b_U \in \mathbb{R}^n$. Essa forma é a considerada pelo repositório CUTEr ([20]) A partir desse problema, criamos o

problema de igualdade e caixas

$$\begin{array}{rrrcl} \min & f(x) & & & \\ \text{suj. a} & & c_E(x) & = & 0, \\ & & c_I(x) & = & s, \\ & b_L & \leq \quad x & \leq & b_U, \\ & c_L & \leq \quad s & \leq & c_U. \end{array}$$

Definindo a variável

$$z = \begin{bmatrix} x \\ s \end{bmatrix},$$

os vetores

$$l = \begin{bmatrix} b_L \\ c_L \end{bmatrix} \qquad \text{e} \qquad u = \begin{bmatrix} b_U \\ c_U \end{bmatrix},$$

a as funções

$$\begin{aligned} \beta(z) &= -\sum_{i:l_i>-\infty} \log(z_i - l_i) - \sum_{i:u_i<\infty} \log(u_i - z_i), \\ \varphi(z) &= f(x) + \mu\beta(z), \\ h(z) &= \begin{bmatrix} c_E(x) \\ c_I(x) - s \end{bmatrix}, \end{aligned}$$

criamos o problema de pontos interiores

$$\begin{array}{rl} \min & \varphi(z, \mu) \\ \text{suj. a} & h(z) = 0. \end{array} \tag{2.1}$$

Note que $z \in \mathbb{R}^N$, com $N = n + m_I$. Para resolver esse problema, utilizamos a estratégia de Cilindros de Confiança, onde os iterados ficam limitados a "cilindros" em torno do conjunto $\{z \in \mathbb{R}^N : h(z) = 0\}$, definidos como

$$\mathcal{C}(\rho) = \{z \in \mathbb{R}^N : \|h(z)\| \leq \rho\}.$$

O passo normal é um passo que tenta resolver aproximadamente o problema

$$\begin{aligned} \min \quad & \frac{1}{2}\|h(z)\|^2 \\ \text{suj. a} \quad & l \leq z \leq u. \end{aligned}$$

A partir do ponto x^{k-1}, fazemos iterações de uma modificação do método proposto por Francisco, Krejić, and Martínez [17], até que calculamos um ponto z_c^k e um raio $\rho^k = \mathcal{O}(\|g_p^k\|)$ tais que $\|h(z_c^k)\| \leq \rho^k$, onde g_p^k é o gradiente projetado escalado no ponto z_c^k.

O passo tangente é encontrado resolvendo uma aproximação quadrática do Lagrangeano do problema (2.1).

$$\begin{aligned} \min \quad & q(\delta) = \frac{1}{2}\delta^T B\delta + \delta^T g \\ \text{suj. a} \quad & A\delta = 0, \\ & \tilde{l} \leq \delta \leq \tilde{u}, \end{aligned}$$

onde $\tilde{l}$ e $\tilde{u}$ definem a intersecção de uma região de confiança e os limites l e u. Esse problema é resolvido com uma modificação do método de Steihaug [25].

Implementamos o DCICPP utilizando a biblioteca CHOLMOD ([6, 8, 9, 10, 11, 12, 13, 2, 1, 7]) como base, mas criando "embrulhos" em `C++` para as estruturas. Decidimos pelo CHOLMOD porque nossos sistemas lineares são da forma $AA^Tx = b$. Nossa biblioteca de embrulhos ficou separada do algoritmo, e pode ser usada para outras finalidades. Ela está disponível gratuitamente em [23]. Utilizamos o Goto BLAS ([26]) para as operações de Álgebra Linear, e também usamos o Metis ([21]) para procurar permutações adequadas de linhas do CHOLMOD. Nosso algoritmo já está disponível na internet ([24]) sob licença GPL, apesar de ainda estar em fase de desenvolvimento.

Utilizamos o repositório de testes CUTEr para comparar o nosso método com o método IPOPT([27]). Fizemos a comparação utilizando uma técnica chamada "perfil de desempenho" (*performance profile* [15]). Para fazer o gráfico do perfil de desempenho para um conjunto S de algoritmos em um conjunto P de problemas, precisamos calcular, para cada problema $p \in P$ e cada algoritmo $s \in S$, a razão de desempenho, definida por

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}},$$

onde $t_{p,s}$ é o tempo gasto pelo algoritmo s para resolver o problema p. O desempenho geral do algoritmo s é representado pela função

$$\mathcal{P}_s(t) = \frac{1}{N_p}\text{size}\{p \in P : r_{p,s} \leq t\},$$

onde N_p é o número de problemas considerado. Se um algoritmo s não consegue resolver o problema p, definimos $r_{p,s} = \infty$. Também definimos o maior valor finito para o qual os algoritmos convergem como $r_f = \max_{p,s}\{r_{p,s} : r_{p,s} < \infty\}$. A função $\mathcal{P}_s(t)$ mede a fração dos problemas resolvidos pelo algoritmo s dentro do tempo t escalado pelo tempo do algoritmo mais rápido. O valor $\mathcal{P}_s(1)$ indica a eficiência do algoritmo s, enquanto que o valor $\mathcal{P}_s(r_f)$ indica a robustez.

Utilizamos como comparação os problemas considerados pequenos pelo CUTEr, excluindo aqueles com variáveis fixas. A Figura 1 compara os método nesses problemas. Definimos um máximo de 200000 iterações, 200000 restaurações, e 2 horas. Essas com-

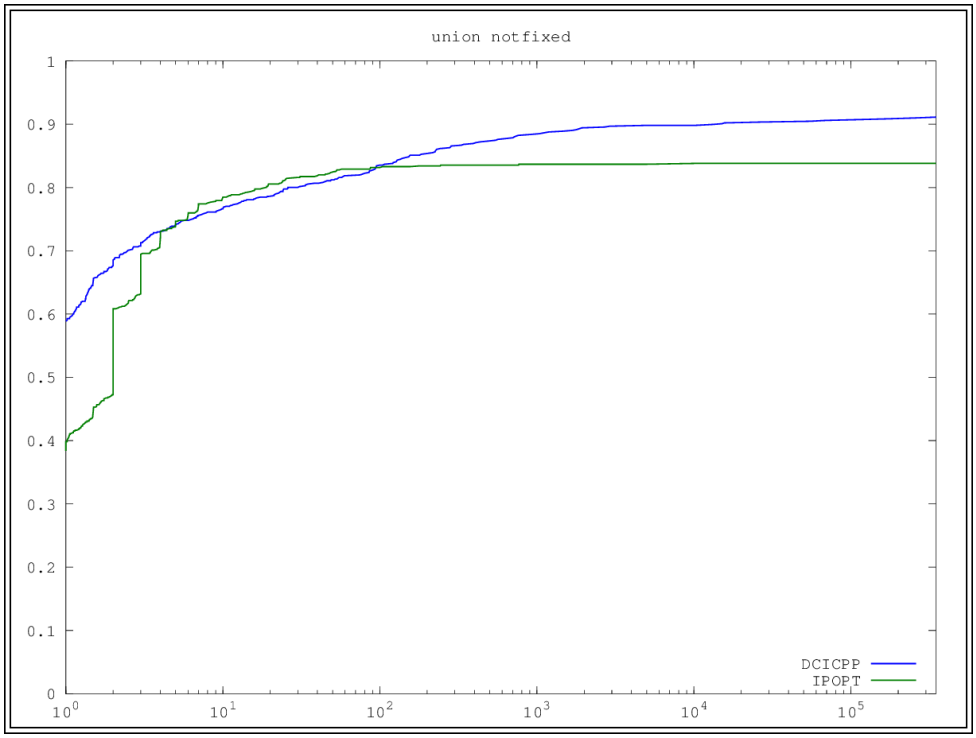

Figure 1: 767 problemas pequenos do CUTEr

parações indicam que o algoritmo é mais eficiente que o IPOPT, numa boa quantidade dos

problemas. No entanto, o IPOPT é mais robusto, conseguindo resolver uma quantidade maior que problemas. Acreditamos que existe espaço para melhoria no algoritmo, e que é possível alcançar uma boa porcentagem de convergência.

A Tabela 1 mostra os testes pequenos do CUTEr. O resultado **Convergiu** indica que o algoritmo convergiu para um ponto estacionário. O resultado **Máximo de Iterações** indica que o algoritmo não convergiu no máximo de iterações. O resultado **Máximo de Restaurações** indica que o algoritmo normal não conseguiu encontrar um ponto dentro do cilindro menor no número dado de iterações. O resultado $\rho_{\max}$ **pequeno** indica que o cilindro ficou muito pequeno. O resultado **Máximo de tempo** indica que o algoritmo não convergiu no tempo de execução permitido. O resultado **Estacionário da Inviabilidade** indica que o algoritmo normal parou em um ponto estacionário para o problema normal. O resultado **Ilimitado** indica que a norma do iterando ficou muito grande. O resultado **Outras falhas** é um conjunto de falhas específicas para cada problema. Os outros resultados são auto-explicativos.

Saída do Algoritmo	**Total**	
	N°	**%**
Convergiu	698	91.00
Máximo de Iterações	5	0.65
Máximo de Restaurações	12	1.56
$\rho_{\max}$ **pequeno**	21	2.74
Máximo de tempo	17	2.22
Estacionário da Inviabilidade	7	0.91
Ilimitado	5	0.65
Outras Falhas	2	0.26
Total	767	100.00

Table 1: Resultados do algoritmo

Planejamos estudar em detalhes os motivos pelos quais o DCICPP não converge para os outros problemas, separando-os em classes diferentes e analisando separadamente cada classe. Também queremos procurar outros métodos para resolver os subproblemas e avaliar se as Jacobianas mal condicionadas são responsáveis pela falha de alguns problemas. Aproveitaremos para poder melhorar a usabilidade do algoritmo, e incluir a possibilidade de variáveis fixas. Dessa maneira, teremos um algoritmo mais competitivo.

3 Cronograma

- **Dezembro de 2013 - Junho de 2013**
  - Separação e estudo dos problemas com falha do algoritmo.
  - Tratamento das variáveis fixas.
- **Julho de 2013 - Novembro de 2014**
  - Implementação de outras estratégias de resolução dos subproblemas.
  - Estudo das Jacobianas mal condicionadas.
  - Limpeza do código.

References

[1] P. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. **SIAM Journal on Matrix Analysis and Applications**, 17(4):886–905, 1996.

[2] P. Amestoy, T. A. Davis, and I. S. Duff. Algorithm 837: Amd, an approximate minimum degree ordering algorithm. **ACM Transactions on Mathematical Software**, 30(3):381–388, 2004.

[3] Lorenz T. Biegler, Jorge Nocedal, and Claudia Schmid. A reduced hessian method for large-scale constrained optimization. **SIAM Journal of Optimization**, 5:314–347, 1995.

[4] H. Byrd, J. C. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. **Mathematical Programming**, 89: 149–186, 2000.

[5] H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. **SIAM Journal of Optimization**, 9:877–900, 2000.

[6] Y. Chen, T. A. Davis, W. W. Hager, and S. Raamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. **ACM Transactions on Mathematical Software**, 35(3), 887.

[7] T. A. Davis. Suitesparse. http://www.cise.ufl.edu/research/sparse/SuiteSparse.

[8] T. A. Davis and W. W. Hager. Multiple-rank modifications of a sparse cholesky factorization. **SIAM Journal on Matrix Analysis and Applications**, 22(4): 997–1013, 2001.

[9] T. A. Davis and W. W. Hager. Modifying a sparse cholesky factorization. **SIAM Journal on Matrix Analysis and Applications**, 20(3):606–627, 2002.

[10] T. A. Davis and W. W. Hager. Row modifications of a sparse cholesky factorization. **SIAM Journal on Matrix Analysis and Applications**, 26(3):621–639, 2005.

[11] T. A. Davis and W. W. Hager. Dynamic supernodes in sparse cholesky update/downdate and triangular solves. **ACM Transactions on Mathematical Software**, 35(4), 2009.

[12] T. A. Davis, J. R. Gilbert, S. Larimore, and E. Ng. A column approximate minimum degree ordering algorithm. **ACM Transactions on Mathematical Software**, 30 (3):353–376, 2004.

[13] T. A. Davis, J. R. Gilbert, S. Larimore, and E. Ng. Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm. **ACM Transactions on Mathematical Software**, 30(3):377–380, 2004.

[14] J. E. Dennis and L. N. Vicente. On the convergence theory of trust-region-based algorithms for equality-constrained optimization. **SIAM Journal of Optimization**, 7:927–950, 1997.

[15] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. **Mathematical Programming**, 91(2):201–213, 2002.

[16] M. El-Alem. A global convergence theory for a general class of trust-region-based algorithms for constrained optimization without assuming regularity. **SIAM Journal of Optimization**, 9:965–990, 1999.

[17] J. B. Francisco, N. Krejić, and J. M. Martínez. An interior-point method for solving box-constrained underdetermined nonlinear system. **Journal of Computational and Applied Mathematics**, 177:67–88, 2005.

[18] F. A. Gomes, M. C. Maciel, and J. M. Martínez. Nonlinear programming algorithms using trust regions and augmented lagrangians with nonmonotone penalty parameters. **Mathematical Programming**, 84:161–200, 1999.

[19] F. A. M. Gomes and R. H. Bielschowsky. Dynamic control of infeasibility in equality constrained optimization. **SIAM Journal of Optimization**, 19:1299–1325, 2008.

[20] N. Gould, D. Orban, and Ph. L. Toint. Cuter, a constrained and unconstrained testing environment, revisited. **Transactions of the American Mathematical Society on Mathematical Software**, 29(4):373–394, 2003.

[21] Karypis Lab. Metis - serial graph partitioning and fill-reducing matrix ordering. http://www-users.cs.umn.edi/ karypis/metis.

[22] M. Lalee, J. Nocedal, and T. Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. **SIAM Journal of Optimization**, 8: 682–706, 1998.

[23] Abel S. Siqueira. Base matrices. http://www.github.com/abelsiqueira/base_matrices.

[24] Abel S. Siqueira and F. A. M. Gomes. Dcicpp. http://www.github.com/abelsiqueira/dcicpp.

[25] Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. **SIAM Journal of Numerical Analysis**, 20(3):626–637, 1983.

[26] TACC: Texas Advanced Computing Center. GotoBLAS2. http://www.tacc.utexas.edu/tacc-projects/gotoblas2/.

[27] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. **Mathematical Programming**, 106:25–57, 2006.