2003年日本オペレーションズ・リサーチ学会
秋季研究発表会

信頼領域法を用いた逐次二次計画法の実装

01308410 (株) 数理システム *檀 寛成 DAN Hiroshige
01701240 (株) 数理システム 山下 浩 YAMASHITA Hiroshi

1 序論

本発表では次の最適化問題を考える.

$$\begin{array}{ll}\text{minimize} & f(x),\ x \in \Re^n \\ \text{subject to} & g_j(x) = 0\ (j \in J_E) \\ & g_j(x) \geq 0\ (j \in J_I)\end{array} \tag{1}$$

ただし, $f, g_j\ (j \in J_E \cup J_I)$ は二回連続微分可能であるものとし, $J_E \cap J_I = \emptyset$ とする.

(1) に対する解法として逐次二次計画法を用い, さらに (1) が大規模な場合にも適用可能であるように, 部分問題に信頼領域法における枠組みを導入した手法を考える. このような手法は過去にいくつか提案されているが, それらの手法では, 信頼領域制約条件を付加した二次計画部分問題が実行不可能となる可能性が少なくないという問題がある. 過去に提案された手法は, この問題を回避するために極めて複雑な操作を行っている. また, 部分問題として非凸二次計画問題を解かなくてはいけないという問題もある.

檀・山下は, これらの問題を解消した手法を提案した [2]. この手法の各反復では, 部分問題として凸二次計画問題と線形方程式系を解くだけで良い.

本発表では, [2] で提案した手法の実装の詳細と, その実装に基づく数値実験の結果紹介する.

2 準備

(1) に対するメリット関数 $F(x)$, モデル二次関数 $F_q(x,\cdot)$, ならびにそれらの差分 $\Delta F(x;\cdot), \Delta F_q(x;\cdot)$ を次のように定める.

$$F(x) = f(x) + \sum_{j\in J_E} \rho_j |g_j(x)| + \sum_{j\in J_I} \rho_j |\min\{0, g_j(x)\}|$$

$$F_q(x;d) = f(x) + \nabla f(x)^T d + \sum_{j\in J_E} \rho_j |g_j(x) + \nabla g_j(x)^T d| + \sum_{j\in J_I} \rho_j |\min\{0, g_j(x) + \nabla g_j(x)^T d\}| + \frac{1}{2} d^T G d$$

$$\Delta F(x;d) = F(x+d) - F(x)$$

$$\Delta F_q(x;d) = F_q(x;d) - F(x)$$

ただし, $\rho_j\ (j \in J_E \cup J_I)$ は正の値を取るペナルティパラメータであり, 行列 G は, アルゴリズム中に出てくる G_k に対応している.

次の二次計画問題の解を Δx_{SD_k}, ラグランジュ乗数を $y_{SD_{k+1}}$ とする:

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\Delta x^T D_k \Delta x + \nabla f(x_k)^T \Delta x,\ \Delta x \in \Re^n \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k)^T \Delta x = 0\ (j \in J_E) \\ & g_j(x_k) + \nabla g_j(x_k)^T \Delta x \geq 0\ (j \in J_I)\end{array} \tag{2}$$

ただし, D_k は要素が正の値の対角行列とする. 詳細は 4.1 節で説明する. ここで, (2) において有効な制約の添字の集合を J_{A_k} とする. すなわち,

$$J_{A_k} := \{j \in J_E \cup J_I \mid g_j(x_k) + \nabla g_j(x_k)^T \Delta x_{SD_k} = 0\}$$

である. さらに, 次の二次計画問題の解を Δx_{N_k}, ラグランジュ乗数を $y_{N_{k+1}}$ とする:

$$\begin{array}{ll}\text{minimize} & \frac{1}{2}\Delta x^T G_k \Delta x + \nabla f(x_k)^T \Delta x,\ \Delta x \in \Re^n \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k)^T \Delta x = 0\ (j \in J_{A_k})\end{array} \tag{3}$$

ただし, $y_{N_{k+1}j} = 0\ (j \notin J_{A_k})$ とする. また, G_k は $\nabla_x^2 L(x_k, y_k)$ とするが, アルゴリズム中で変更される場合がある. 詳細は 4.2 節で説明する. また, (3) の KKT 条件を整理すると, 次の線形方程式系となることに注意する.

$$\begin{pmatrix} G_k & -\nabla g_J(x_k) \\ \nabla g_J(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta x_k^N \\ y_{k+1,J}^N \end{pmatrix} = \begin{pmatrix} -\nabla f(x_k) \\ -g_J(x_k) \end{pmatrix} \tag{4}$$

ここで $g_J(x_k), y_{k+1,J}^N$ は, それぞれ $g_j(x_k), y_{k+1,j}^N (j \in J_{A_k})$ からなるベクトルであるものとする.

3 アルゴリズム

[2] で提案したアルゴリズムは次のとおりである.

アルゴリズム TRSQP

Step 0. 初期点 $x_0 \in \Re^n$ と, 対称行列 $G_0 \in \Re^{n\times n}$ を定める. また, パラメータ $\rho > 0, M > 0, \delta_0 > 0$ を選ぶ. $k = 0$ とする.

Step 1. (2), (3) を解き $\Delta x_{SD_k}, \Delta x_{N_k}, y_{SD_{k+1}}, y_{N_{k+1}}$ を求める. このとき,

$$\|\Delta x_{N_k}\| \leq M \|\Delta x_{SD_k}\| \tag{5}$$

が満たされない場合は, (5) を満たすように G_k を変更する.

Step 2. y_{k+1} を

$$y_{k+1} := \begin{cases} y_{k+1}^N, & y_{k+1,j}^N \geq 0\ (j \in J_{A_k} \cap J_I) \text{ の場合} \\ y_{k+1}^{SD}, & \text{それ以外の場合} \end{cases}$$

とする. (x_k, y_{k+1}) が終了条件を満たしていれば計算終了.

Step 3. 次の条件を満たすような $s_k \in \Re^n$ を求める.

$$\begin{cases} \|s_k\| \le \delta_k,\ \|s_k\| \le M\|\Delta x_{SD_k}\| \\ \Delta F_q(x_k; s_k) \le \frac{1}{2}\Delta F_q(x_k; \alpha^*(x_k, \Delta x_{SD_k})\Delta x_{SD_k}) \end{cases} \quad (6)$$

ここで, $\alpha^*(x_k, d)$ は次のように定められる.

$$\alpha^*(x_k, d) = \arg\min_{\alpha}\{F_q(x; \alpha d) \mid \alpha \in (0,1],\ \|\alpha d\| \le \delta_k\}$$

Step 4. δ_{k+1} を次のように定める.

$$\begin{array}{rcl} \Delta F(x_k; s_k) > \frac{1}{4}\Delta F_q(x_k; s_k) & \Rightarrow & \delta_{k+1} = \frac{1}{2}\delta_k \\ \Delta F(x_k; s_k) \le \frac{3}{4}\Delta F_q(x_k; s_k) & \Rightarrow & \delta_{k+1} = 2\delta_k \\ \text{それ以外} & \Rightarrow & \delta_{k+1} = \delta_k \end{array}$$

Step 5. もし $\Delta F(x_k; s_k) \le 0$ であれば $x_{k+1} = x_k + s_k$ とする. さもなくば $x_{k+1} = x_k$ とする. $k := k+1$ とし Step 1 へ. ∎

アルゴリズム TRSQP は適当な仮定の下で大域的収束する [2]. なお, Step 3. における s_k の構成方法については 4.3 節で説明する.

4 アルゴリズムの実装詳細

4.1 D_k の構成

D_k は, 理論的には要素が正の値の対角行列とすればよいが, 実装上は $\nabla_x^2 L(x_k, y_k)$ のスケールを反映するように設定するのがよいと考えられる. そこで, 以下の二つの方法を考えた.

(i) $\nabla_x^2 L(x_k, y_k)$ の対角要素の絶対値で D_k を構成する. ただし, その対角要素が 0 の場合は, 適当な正の微小量を設定する.

(ii) BFGS 法によって $\nabla_x^2 L(x_k, y_k)$ の近似行列 B_k を構成し, その対角要素で D_k を構成する. この場合, (i) のように対角要素が 0 となることはない. しかし, 大規模問題に BFGS 法をそのまま用いるのは好ましくないため, BFGS 法を変形し, 対角要素のみを見積もるようにする.

これらの方法を比較すると, (ii) による実装の方が, より多くの問題において最適解を求めることができることが分かった.

4.2 G_k の構成

$\nabla g_j(x_k)$ $(j \in J_{A_k})$ が一次独立でありパラメータ M が十分大きいとき, G_k が正則であれば (5) は満たされる. 従って (5) が満たされないときは G_k を比較的よく反映し, かつ正則な行列を用いて (3) を構成するのが妥当である.

ここで, G_k を $\bar{G}_k := G_k + \mu_k I$ $(\mu_k > 0)$ で定義される $\bar{G}_k$ で置き換えることを考える. このとき, G_k が正則でなくとも μ_k が十分大きければ $\bar{G}_k$ は正則となる. 実際には, 最初は μ_k として正の微小量を与え, それでも (5) が満たされない場合は徐々に大きな値を設定していけばよい.

4.3 s_k の構成

Step 3. での s_k の定め方には, 例えば次のような方法がある : Δx_{SD_k} と Δx_{N_k} の凸結合

$$\bar{s}_k(\nu_k) := \nu_k \Delta x_{SD_k} + (1-\nu_k)\Delta x_{N_k},\ \nu_k \in [0,1]$$

を考える. このとき, $\alpha^*(x_k, \bar{s}_k(\nu_k))\bar{s}_k(\nu_k)$ が (6) を満たしているかどうかを調べる. このとき, $\bar{s}_k(1)$ が (6) を満たしていることは明らかである. また, $\bar{s}_k(0)$ は, 解の十分近くでは (ある条件の下で) 通常の逐次二次計画法での探索方向と同じ方向となる. 従って, 実装の際には, まず $\nu_k = 0$ を試し, (6) が満たされるまで ν_k を 0.1 ずつ増やしていくなどの方法が考えられる.

5 数値実験の結果

アルゴリズム TRSQP による数値実験の結果を紹介する. 実験環境は以下の通りである : CPU: Pentium IV 2.8 Ghz, Memory: 1Gbyte, OS: Red hat Linux 8.0. 問題は問題集 CUTE [1] 等を用いた. 実験の結果, アルゴリズム TRSQP は大規模問題にも適用できることがわかり, 十分高速な収束をすることも確認された.

表 1: 小・中規模問題で求解可能であった問題数

問題規模	問題数	TRSQP	内点法
100 変数 100 制約以下	304	274	278
500 変数 500 制約以下	82	69	62
1000 変数 1000 制約以下	36	29	20
3000 変数 3000 制約以下	24	17	19
合計	446	389	379

表 2: 大規模問題での結果

問題名	変数	制約	反復	時間 (秒)
BRAINPC7	6907	6900	42	202.16
BRAINPC8	6907	6900	9	89.97
DTOC2	5998	3996	5	6.45
DTOC5	9999	4999	3	501.33

表 3: 内点法よりも高速に求解できる問題

問題名	変数	制約	TRSQP (秒)	内点法 (秒)
HYDROELM	505	505	0.06	0.12
PRIMALC8	520	9	0.04	0.09
PT	2	502	0.04	0.21
BRIDGEND	2734	2728	0.86	1.17
GMNCASE2	175	1051	1.87	2.52
OET3	4	1003	0.11	0.23
SIPOW1M	2	2001	0.09	1.81
SIPOW3	4	2001	0.08	2.93
YAO	2002	2001	1.33	1.60

参考文献

[1] I. Bongartz, A. R. Conn, N. Gould and Ph. L. Toint, CUTE: Constrained and Unconstrained Testing Environment, ACM Transactions on Mathematical Software, 21(1995), 123-160.

[2] 檀寛成, 山下浩, 信頼領域法を用いた逐次二次計画法の大域的収束性, 2002 年日本オペレーションズ・リサーチ学会秋季研究発表会アブストラクト集, 208-209.