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

計算グラフと分枝限定法を利用した大域的最適化

01701240 （株）数理システム ＊山下 浩 YAMASHITA Hiroshi
（株）数理システム 逸見 宣博 HENMI Nobuhiro

1. はじめに

本発表は，問題：

$$\text{最小化} \quad f(x), \quad x \in X_0 \subset \mathbf{R}^n \qquad (G_0)$$

の大域的最適解を数値的に求めるためのアルゴリズムについて述べる．問題の構造を特に仮定しない場合，(一定の数値誤差の範囲内で) 真の大域的最適解を確実にかつ高速に求めるためには，数値解法としては分枝限定法を利用するのが最も一般的であろう．本発表では

(i) 高速自動微分の機能を持つモデリング言語を利用することによって，元の問題の計算グラフによる表現から緩和問題の生成が一般的に可能になる．

(ii) 緩和問題の解法に高速な大規模凸計画問題ソルバーを利用することによって，分枝限定法のアルゴリズムが実用的な時間で実施できる．

という二つの目論見を基本にした大域的最適化のアルゴリズム開発について述べる．([1,2,3] でも計算グラフが利用されている．)

問題 (G_0) の緩和問題を

$$\text{最小化} \quad f_0(x), \quad x \in \overline{X}_0 \subset \mathbf{R}^n \qquad (C_0)$$

とする．緩和問題は次の性質を持っているとする．

(i) (C_0) が許容解を持たないとき，(G_0) も許容解を持たない．

(ii) (C_0) の最適解を $\overline{x}$ としたとき，$\overline{x} \in X_0$ ならば $\overline{x}$ は (G_0) の最適値の上界を与える．

(iii) (C_0) の最適値を f_0^* としたとき，(G_0) の最適値 f^* とは $f_0^* \leq f^*$ の関係がある．

上記の性質を持つ緩和問題として，以下ではいわゆる凸緩和問題を考える．問題の制約領域 X_0 が部分領域 $X_1, X_2, \ldots$ に分割されたとき，それらに対応する問題：

$$\text{最小化} \quad f(x), \quad x \in X_k \subset \mathbf{R}^n \qquad (G_k)$$

を子問題という．子問題 (G_k) の緩和問題を

$$\text{最小化} \quad f_k(x), \quad x \in \overline{X}_k \subset \mathbf{R}^n \qquad (C_k)$$

と定義する．

2. 分枝限定法

[アルゴリズム]

ステップ 0. $\overline{z} = +\infty, \mathcal{P} = \{(G_0)\}, \epsilon > 0.$

ステップ 1.

● $\mathcal{P} \neq \emptyset$ ならばステップ 2 へ．

● $\mathcal{P} = \emptyset$ ならば停止．($\overline{z} = +\infty$ ならば問題 (G_0) には許容解が存在しない．$\overline{z} < +\infty$ ならば $\overline{x}$ が最適解，$\overline{z}$ が最適値となる．)

ステップ 2. (部分問題選択) $\mathcal{P}$ の中から問題を一つ選び，それを (G_k) とする．$\mathcal{P} = \mathcal{P} \setminus \{(G_k)\}$.

ステップ 3. 緩和問題 (C_k) を定義し，それを解いて最適解 $\widehat{x}_k$，最適値 $\widehat{f}_k$ を求める．

● (分枝停止) 許容解がなければステップ 1 へもどる．

● $\widehat{x}_k \in X_k$ ならばステップ 4 へ．

● $\widehat{x}_k \notin X_k$ ならばステップ 5 へ．

ステップ 4. (分枝停止)

● $\widehat{f}_k \geq \overline{z} - \epsilon$ ならばステップ 1 へもどる．

● (上界値更新) $\widehat{f}_k < \overline{z}$ ならば $\overline{z} = \widehat{f}_k, \overline{x} = \widehat{x}_k$ としてステップ 1 へもどる．

ステップ 5.

● (分枝停止) $\widehat{f}_k \geq \overline{z} - \epsilon$ ならばステップ 1 へもどる．

● (分枝) $\widehat{f}_k < \overline{z}$ ならば (G_k) の子問題を作り，$\mathcal{P}$ に加える．ステップ 1 へもどる． □

上界値として元の問題の局所最適界が利用できるが上記アルゴリズムでは省略してある．

3. 緩和問題

問題を

$$(1) \quad \begin{array}{ll} \text{最小化} & f(x), \quad x \in \mathbf{R}^n, \\ \text{条　件} & g_{iL} \leq g_i(x) \leq g_{iU}, i = 1, \ldots, m \\ & x_L \leq x \leq x_U, \; x \in S \end{array}$$

と書く．有界な領域を扱うために，変数に対する (有界な) 上下限が常に存在することを前提とする．集合 S は整数条件などを表わす．集合 S から生成される分枝操作と緩和集合は通常の整数計画問題と同様なので以下では述べない．以下では，連続な非線形関数である目的関数 $f(x)$ と制約条件 $g_{iL} \leq g_i(x) \leq g_{iU}, i = 1, \ldots, m$ の凸緩和について考える．得られる凸緩和問題の可能性としては，(i) 一般の非線形凸計画問題，(ii)2 次錘計画問題，(iii) 凸 2 次計画問題，(iv) 線形計画問題などが考えられる．ここでは，主として (i) の可能性について考える．

問題がモデリング言語SIMPLEによって記述されているとする．SIMPLEは，高速自動微分の実行のために問題の情報を計算グラフの形式で保持する．計算グラフは，計算の過程を2項演算 (例：$x \times y$)，単項演算 (例：x^2)，関数評価（例：$\sin(x)$）の素過程の連なりとして表現するものである．C++で書かれたユーザーのモデルを一度実行すれば，演算子と関数のオーバーローディングの機構によって計算グラフが計算機内部に貯えられる．

以下では，そのような計算グラフを利用した凸緩和問題の生成について述る．(ここで述べる方法以外にも色々な方法が存在するが省略する．以下は一例である.) $x_1,...,x_n$ の任意の関数 f の計算法が計算グラフで与えられているとする．計算グラフのノード（1項あるいは2項演算の結果）に中間変数 $x_{n+1},...,x_{n+p}$ を設定する．このとき関数 f の計算は

$$\begin{aligned} w &= a \times (x \times y) && \text{(積)} \\ w &= a \times (x \div y) && \text{(商)} \\ w &= a \times (x + y) && \text{(和)} \\ w &= a \times (x - y) && \text{(差)} \\ w &= func(x), func(x,y) && \text{(関数)} \end{aligned}$$

などの演算の連鎖となる．ここで，w は中間変数，x と y は $x_1,...,x_n$ のどれか，あるいは中間変数である．a は定数である．グラフのトップは上記の w が関数 f となった表式である．そして，すべての中間変数を最適化問題の変数として扱い，上記のような中間変数の定義式を等式条件として扱う．新たに定義された最適化の変数全体とその空間を $z, \mathbf{R}^N$ とすると，適当な仮定の下で，問題 (1) は

$$\begin{array}{ll} \text{最小化} & z_1, \quad z \in \mathbf{R}^N, \\ \text{条　件} & z_i = h_i(z), i = 1,...,N \\ & z_L \le z \le z_U,\ z \in S' \end{array}$$

と書くことが出来る．関数 h_i は上記の五つのタイプのどれかである．(線形の目的関数あるいは線形制約条件に関しては，上記のような中間変数の導入は実際は不要である.) したがって，緩和問題の生成は制約条件 $z_i = h_i$ の凸緩和となる．

x, y には上下限：$x_L \le x \le x_U$，$y_L \le y \le y_U$ が設定されているとする．

(i) 積の演算が現れた場合で $a \ge 0$ ならば新たに

$$\begin{aligned} a(x_L y + y_L x - x_L y_L) &\le w \\ a(x_U y + y_U x - x_U y_U) &\le w \\ w &\le a(x_U y + y_L x - x_U y_L) \\ w &\le a(x_L y + y_U x - x_L y_U) \end{aligned}$$

という条件を設定する．また，上記の制約条件より $w_L \le w \le w_U$ となる w の上下限 w_L, w_U を計算しておく．$a < 0$ ならば上記の不等式条件の大小関係を逆にする．w が計算グラフのトップに位置する場合は対応する制約条件の種類によって $w = 0, w \le 0$ などの制約が設定される．

積の演算で $w = a \times x^2$ となっている場合は，$a \ge 0$ ならば

$$ax^2 \le w \le a((x_U + x_L)x - x_U x_L)$$

とする．

積の演算で，新たに $s = x + y$ という中間変数を導入し，$w = a(s^2 - x^2 - y^2)/2$ として上記の不等式を利用する緩和法もあるが，ここでは省略する．

(ii) 商の演算が現れた場合は x, y の上下限の情報から w の上下限 w_L, w_U を計算する．その後に

$$x = \frac{1}{a} w \times y$$

と変換して (i) にしたがって，不等式条件を設定する．

(iii) 和と差が現れた場合は中間変数の定義式を制約条件とする．実際には，

$$w = x + y + z$$

のようにまとめることが出来て，個々の（和と差の）演算子に制約条件を対応させる必要はない．

(iv) 関数 ($func$) が現れた場合は関数ごとにあらかじめ凸制約を設定しておく．たとえば，$w = \exp(x)$ に対しては

$$\exp(x) \le w \le \frac{\exp(x_U) - \exp(x_L)}{x_U - x_L}(x - x_L) + \exp(x_L)$$

その他の関数も同様である．

4. まとめ（モデリング言語を経由した）計算グラフによる問題表現と高速な凸計画ソルバーによって，分枝限定法を利用した大域的最適化のアルゴリズムの実用化の可能性が期待できる．実用化のためには，分枝限定法の具体的なアルゴリズムの検討や様々な凸緩和問題生成の可能性のテストなど検討すべきことは多い．発表当日には簡単な数値実験結果も提示する予定である．

参考文献

[1]K.Yamamura, "An Algorithm for Representing Functions of Many Variables by Superpositions of Functions of One Variables and Addition", IEEE Trans. Circuit ans Systems-I Vol.43,No.4,1996.

[2]E.Smith and C.Pantelides,"Global Optimization of General Process Models", Grossmann ed. Global Optimization in Engineering Design, 355-386, Kluwer, 1996.

[3]B.Kearfott,"Rigorous Global Search: Continuous Problems", Kluwer, 1997.