大域的最適化の実用化に向けて

山下浩　　逸見宣博
（株）数理システム

東京都新宿区新宿 2-4-3
hy@msi.co.jp　henmi@msi.co.jp

Abstract

This paper describes our recent effort for constructing efficient deterministic global optimization algorithm and software for solving minimization of a nonlinear objective function subejct to nonlinear equality/inequality constraints including integer variables. The method is based on a branch and bound type algorithm. In this method, generation of tight convex relaxation problems of the original problems in subregions is essential. For this purpose we utilize computational graph representation of the problem using information stored by our modeling language SIMPLE. Details of various implementation issues and results of computational experiment are described.

Keywords: global optimization, branch and bound, computational graph

1　はじめに

近年，大域的最適化に対する様々なアプローチが活発に行われている．対象とする問題とアルゴリズムのカテゴリー分けをすると，
・特定の問題に対する専用アルゴリズム
・特に問題の構造を仮定しない汎用アルゴリズム
がある．また，アルゴリズムの基本的立場として，
・確実に大域的最適解を得ることを目指す方法
・局所最適解を多く発見して確率的に大域的最適解を求めようとする方法
がある．これらのアルゴリズムはそれぞれの長所・短所を持っていて，どの立場も研究に値するが，本発表では近年我々が研究している「問題の構造を特に仮定しないで，真の大域的最適解を確実にかつ高速に求めることを（最終的）目標とするアルゴリズム」について述べる．

大域的最適化には組み合わせ的側面が存在するので，アルゴリズムの基本的枠組みとして分枝限定法を利用する．本方法は
(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凸緩和問題の効率的な自動的生成が一般的に可能になる．
(ii) 凸緩和問題の解法に高速な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（希望的）観測に基づいている．

問題を

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

と書く．制約領域 X_0 は通常の等式・不等式条件や整数条件などを表している．そして，問題 (G_0) の凸緩和問題を

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

とする．すなわち，問題 (C_0) は凸計画問題で，$f_0(x) \leq f(x), \forall x \in \mathbb{R}^n, X_0 \subset \overline{X}_0$ をみたす．このとき，以下の事実が成り立つ．

(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．

問題の制約領域 X_0 が部 分領域 $X_1, X_2, ...$ に分割されたとき，それらに対応する問題：

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

を子問題という．子問題 (G_k) の緩和問題を

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

と定義する．

我々のアルゴリズムは，以下で述べるように「モデリング言語 → 計算グラフ → 中間変数の導入 → 基本演算に対応する等式条件の凸緩和 → 分枝限定法の適用」という一連のアイディアに基づくものであるが，開発を行っている途中でこのようなアイディアの各部分は古くからあったことが徐々に分ってきた．1976 年の G.P.McCormick の論文 [8](分解可能な関数に対する凸緩和）から，1990 年代に入って中間変数の導入や計算グラフの概念も含めて幾つか論文が発表されている（[6],[10],[12] など）．また，単項演算あるいは 2 項演算で定義される “任意の” 問題に適用可能なアルゴリズムには，本論文で述べるものと類似の方法が使用されていることが多いと想像する．その他，多くの関連した文献は参考文献の項を参照されたい．

2　分枝限定法

分枝限定法のプロトタイプアルゴリズムを以下のように定義する．

[**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.（子問題選択）$\mathcal{P}$ 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. 子問題 (G_k) の緩和問題 (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）許容解がないか，$\widehat{f_k} \geq \overline{z} - \epsilon$ ならばステップ 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）$f_0(\widehat{x}_k) \geq \overline{z} - \epsilon$ ならばステップ 1 へもどる．
●（上界値更新）$f(\widehat{x}_k) < \overline{z}$ ならば $\overline{z} = f(\widehat{x}_k), \overline{x} = \widehat{x}_k$ とする．
ステップ 5.　（分枝）(G_k) の子問題を作り，$\mathcal{P}$ に加える．ステップ 1 へもどる．　□

問題が整数変数を含まない場合は元の最適化問題の局所最適解あるいは許容解の情報を上界値として利用できる．すなわち，元の問題 (G_0) の局所最適解をステップ 2 の最初の反復で求める．あるいはその後ステップ 2 で選択された子問題 (G_k) の局所最適解を適宜解いて上界値として利用する．すべての子問題に対して局所最適解を求めるのは計算時間の点からはあまり望ましくないので，適当なタイミングで実施するのが良い．我々のコードでは，局所最適化のために内点法あるいは SQP 法のアルゴリズム（[16],[19], [17]）を使用している．

3　緩和問題の生成

問題を，より詳しく

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

と書く．以下では，有界な領域を扱うために，変数に対する上下限が常に存在することを前提とする.（ただし，実際の実装では必ずしもこのことを仮定しない.）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である．

以下では，目的関数 $f(x)$ と制約条件 $g_{iL} \le g_i(x) \le g_{iU}, i = 1, ..., m$ 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 次錐計画問題，
(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)，(iii), (iv) の可能性を念頭に置いて考えるが，現在の実装は (iv) の場合について行われている．

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：$\sin(x)$）などの素過程の連なりとして表現するものである．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べる．

変数 $x_1, ..., x_n$ 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．計算グラフの計算過程を示す全ノ 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

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

などの演算の連鎖となる．ここで，w は中間変数のどれか，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．

すべての中間変数を最適化問題の変数に組み入れ，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 $z, \mathbb{R}^N$ とすると，問題 (1) は

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

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．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．関数 f 自身が線形の場合は，緩和する必要がないので，上記のような中間変数の導入は不要である．また，凸緩和が必要なのは (積),(商),(関数) の演算のみである．

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．以下で，上記の演算の緩和について具体例を示す．簡単のために $a = 1$ とする．

(i) 積の演算が現れた場合は新たに

$$\begin{aligned} x_L y + y_L x - x_L y_L &\leq w \\ x_U y + y_U x - x_U y_U &\leq w \\ w &\leq x_U y + y_L x - x_U y_L \\ w &\leq x_L y + y_U x - x_L y_U \end{aligned}$$

という条件を設定する．また，上記の制約条件より $w_L \leq w \leq w_U$ となる w の上下限 w_L, w_U を計算しておく．w が計算グラフのトップに位置する場合は対応する元の問題の制約条件によって $w = 0, w \leq 0$ などの制約が設定される．
(ii) 商の演算が現れた場合は，$y_L > 0$ ならば

$$\begin{aligned} \frac{x_U}{y} + \frac{x}{y_L} - \frac{x_U}{y_L} &\leq w \\ \frac{x_L}{y} + \frac{x}{y_U} - \frac{x_L}{y_U} &\leq w \\ w &\leq \frac{x_U}{y} + \frac{x}{y_U} - \frac{x_U}{y_U} \\ w &\leq \frac{x_L}{y} + \frac{x}{y_L} - \frac{x_L}{y_L} \end{aligned}$$

を得る．$y_L < 0$ の場合も同様である．あるいは，x, y の上下限の情報から w の上下限 w_L, w_U を計算し，その後に $x = w \times y$ と変換して (i) にしたがって，不等式条件を設定してもよい．
(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

$$\begin{aligned} w &\leq \frac{\exp(x_U) - \exp(x_L)}{x_U - x_L}(x - x_L) + \exp(x_L) \\ \exp(x_L)(x - x_L) + \exp(x_L) &\leq w \\ \exp(x_U)(x - x_U) + \exp(x_U) &\leq w \end{aligned} \tag{2}$$

2 変数の関数の例として，$w = x^y$ の $x > 0$ の領域での凸緩和を考える．この場合は

$$\log w = y \log x$$

と変換して

$$w' = \log w, x' = \log x, w' = yx'$$

の 3 条件を凸緩和する．その他の関数も同様である．

SIMPLE で扱うことの可能な関数には以下のようなものがある．これらのそれぞれについて，個別の処理が必要になる．

```
sin    cos    tan    asin   acos   atan   sec    csc    cot
asec   acsc   acot   sinh   cosh   tanh   asinh  acosh  atanh
sech   coth   asech  acsch  acoth  atan2  hypot
exp    log    log10  pow    sqrt
ifelse min    max    fabs   fmod
ceil   floor  cbrt
```

4　アルゴリズムの細部

4.1　数値的問題

大域的最適化では，基本的に対象となる全領域にわたって “探索” を行う必要がある．通常の局所最適化アルゴリズムでは，ある初期点から出発して目的関数あるいは適当に設定されたメリット関数がより小さくなるような点列を生成するので，制約領域の “端” の部分まで探索することはあまり起こらない．ところが，大域的最適化では，そのような部分領域には大域的最適解が存在しない，あるいは実行可能点が存在しないことを確認する必要がある．このような際に，しばしば数値的悪条件に遭遇することになる．

例として，$w = \exp(x)$ の凸緩和を考える．変数 x が領域 $[x_L, x_U]$ を動くとき，通常は (2) の様に緩和するが，x_U の値が大きいと係数 $\exp(x_U)$ が巨大となり，緩和問題の LP の係数行列に巨大な値が現れ，単体法による求解に悪影響を及ぼすことになる． この様な場合は，$w = \exp(x), x \geq 0$ を $x = \log(w)$ と書き直して凸緩和してやる．このような操作を，必要ならばそれぞれの関数に対して行う必要がある．

また，有限の変数値に対して無限大の値を取るような関数（たとえば $1/x$,$\tan(x)$) が計算過程で現れる場合は，関数値を適当な有限の値以下になるような制約を加える必要がある．

4.2　緩和問題の求解の実際的手続き

分枝限定法のステップ 3 において緩和問題を解く事になるが，その具体的手続きは以下のようになる．

１．緩和問題の初期化
- 親問題を初期化する．
- 親問題の領域に子問題の制約を追加する．
- 目的関数値に関する制約（目的関数値が暫定大域的最適解以下になる）を追加する．
- 子問題の局所最適解を求めて，その解が暫定大域的最適解を下回ったら， その局所最適解を暫定的大域的最適解として更新して目的関数値制約を再度付加する．

２．緩和問題の制約領域を縮小
- 各非線形変数をチェックして，可能な限り領域を縮小する．
- 緩和線形計画問題を生成する．
- 各線形制約をチェックして，可能な限り領域を縮小する．
- 領域縮小回数が 0 となるまで，上記を反復する．

３．緩和線形計画問題を単体法で求解

4.3　分枝変数の選択

子問題 (G_k) の緩和問題 (C_k) を解いて，最適解 $\hat{z}_k$ が得られたとする．このとき，変数 $z_i, i = 1, \ldots, N$ の中から以下のような基準により分枝変数を選択する．

(i) いつまでも領域分割されない変数があっては困るので，適当なタイミングで変数を選択してその領域を 2 分する．たとえば，元の領域巾に比べて現在の領域巾の減少度合いが少ないものを選択．

(ii) 「制約条件の侵犯」に最も寄与していると思われる変数：たとえば，$w = xy$ という制約式の緩和条件の場合は，$\|\hat{w} - \hat{x}\hat{y}\|$ という量を変数 w, x, y の与える侵犯量とする．すべての緩和制約条件の侵犯量を変数毎に足し合わせてスコア化する．

分枝ポイントは，領域の中点，緩和問題の最適解の位置，元の問題の局所最適解の位置などが候補となる

4.4 前処理による変数減少法

本方法では，中間変数の導入によって凸緩和問題の変数の数が元の問題に比べて多くなる．そこで変数の数を減らすような前処理を工夫する必要がある．そのような方法を例によって示す．

(i) 以下のような問題が与えられたとする．

$$\begin{array}{ll} \text{最小化} & -x_1 \\ \text{条　件} & x_2 - x_1^3 - x_3^2 = 0 \\ & -x_2 + x_1^2 - x_4^2 = 0 \end{array}$$

この問題を中間変数を使って書き直すと，以下のようになる．

$$\begin{array}{ll} \text{最小化} & -x_1 \\ \text{条　件} & x_2 - w_1 - w_2 = 0 \\ & -x_2 + w_3 - w_4 = 0 \\ & w_1 = x_1^3, w_3 = x_1^2 \\ & w_2 = x_3^2, w_4 = x_4^2 \end{array}$$

この問題では，変数 x_3 と x_4 は中間変数定義のための等式条件 $w_2 = x_3^2, w_4 = x_4^2$ にしか現れないので，この等式条件は緩和することなく $w_2 \geq 0, w_4 \geq 0$ の条件を加えるだけでよい．したがって，以下のような問題を緩和すればよい．

$$\begin{array}{ll} \text{最小化} & -x_1 \\ \text{条　件} & x_2 - w_1 - w_2 = 0 \\ & -x_2 + w_3 - w_4 = 0 \\ & w_1 = x_1^3, w_3 = x_1^2 \\ & w_2 \geq 0, w_4 \geq 0 \end{array}$$

(ii) 次の問題を考える．

$$\begin{array}{ll} \text{最小化} & -x_1 x_2 x_3 \\ \text{条　件} & x_j - 4.2 \sin^2(x_{j+3}) = 0, j = 1, 2, 3 \\ & x_1 + 2x_2 + 2x_3 - 7.2 \sin^2(x_7) = 0 \end{array}$$

この問題を中間変数を使って書き直すと

$$\begin{array}{ll} \text{最小化} & -x_1 x_2 x_3 \\ \text{条　件} & x_j - 4.2 w_{4+j} = 0, j = 1, 2, 3 \\ & x_1 + 2x_2 + 2x_3 - 7.2 w_8 = 0 \\ & w_j = \sin(x_{j+3}), j = 1, 2, 3 \\ & w_4 = \sin(x_7) \\ & w_{4+j} = w_j^2, w_8 = w_4^2 \end{array}$$

となる．この問題から $x_j, j = 4, 5, 6$ を消去できて，以下の問題を緩和すればよい．

$$\begin{array}{ll} \text{最小化} & -x_1 x_2 x_3 \\ \text{条　件} & x_j - 4.2 w_{4+j} = 0, j = 1, 2, 3 \\ & x_1 + 2x_2 + 2x_3 - 7.2 w_8 = 0 \\ & 0 \leq w_{4+j} \leq 1, j = 1, 2, 3, 4 \end{array}$$

5　数値実験結果

以下では，現状の実装での簡単な数値実験結果を述べる．数値実験はPentium IV 1.5GHz のCPU，1GBのメモリをもつLinuxマシンで行った．数多くのテスト用問題が [4], [2], [5] などから得られるが，まず特定の問題に対する結果を述べる．以下の表には最適値（optimal value），計算時間，解かれた緩和問題（線形計画問題）の数（#LP）単体法の全反復数（#pivot）が示されている．

(i) Murtagh and Saunders:

$$\begin{aligned} \text{minimize} \quad & (x_1-1)^2+(x_1-x_2)^2+(x_2-x_3)^3+(x_3-x_4)^4+(x_4-x_5)^4 \\ & x_1+x_2^2+x_3^3=3\sqrt{2}+2 \\ \text{subject to} \quad & x_2-x_3^2+x_4=2\sqrt{2}-2 \\ & x_1x_5=2 \\ & -5\le x_i\le 5, i=1,...,5 \end{aligned}$$

optimal value	time(sec)	#LP	#pivot
0.0293108	0.76	325	8433

(ii) Reactor Network Design:

$$\begin{aligned} \text{minimize} \quad & -x_4 \\ & x_1+k_1x_1x_5=1 \\ & x_2-x_1+k_2x_2x_6=0 \\ & x_3+x_1+k_3x_3x_5=1 \\ & x_4-x_3+x_2-x_1+k_4x_4x_6=0 \\ \text{subject to} \quad & \sqrt{x_5}+\sqrt{x_6}\le 4 \\ & 0\le x_1\le 1, 0\le x_2\le 1, 0\le x_3\le 1 \\ & 10^{-5}\le x_4\le 16, 10^{-5}\le x_5\le 16 \\ & k_1=0.09755988, k_2=0.00k_1, k_3=0.0391908 \\ & k_4=0.9 \end{aligned}$$

optimal value	time(sec)	#LP	#pivot
-0.37461046	0.39	281	5815

(iii) Heat Exchange Network:

$$\begin{aligned} \text{minimize} \quad & x_1+x_2+x_3 \\ & 0.0025(x_4+x_6)-1\le 0 \\ & 0.0025(-x_4+x_5+x_7)-1\le 0 \\ & 0.01(-x_5+x_8)-1\le 0 \\ \text{subject to} \quad & 100x_1-x_1x_6+833.33252x_4-83333.333\le 0 \\ & x_2x_4-x_2x_7-1250x_4+1250x_5\le 0 \\ & x_3x_5-x_3x_8-2500x_5+1250000\le 0 \\ & 100\le x_1\le 10000,\ 1000\le x_2,x_3\le 10000 \\ & 10\le x_4,x_5,x_6,x_7,x_8\le 1000 \end{aligned}$$

optimal value	time(sec)	#LP	#pivot
7049.247931	3.32	2783	66020

(iv) Hock and Schittkowski no. 104 (Optimal Reactor Design):

$$
\begin{array}{ll}
\text{minimize} & f(x) = 0.4\ x_1^{0.67} x_7^{-0.67} + 0.4\ x_2^{0.67} x_8^{-0.67} + 10 - x_1 - x_2 \\
\text{subject to} & 1 - 0.0588\ x_5 x_7 - 0.1\ x_1 \geq 0 \\
& 1 - 0.0588\ x_6 x_8 - 0.1\ x_1 - 0.1\ x_2 \geq 0 \\
& 1 - 4\ x_3 x_5^{-1} - 2\ x_3^{-0.71} x_5^{-1} - 0.0588\ x_3^{-1.3} x_7 \geq 0 \\
& 1 - 4\ x_4 x_6^{-1} - 2\ x_4^{-0.71} x_6^{-1} - 0.0588\ x_4^{-1.3} x_8 \geq 0 \\
& 0.1 \leq x_i \leq 10,\ i = 1 \cdots 8
\end{array}
$$

optimal value	time(sec)	#LP	#pivot
3.951120873	72	30699	1289105

(v) Robust stability analysis 1

$$
\begin{array}{ll}
\text{minimize} & k \\
\text{subject to} & 10q_2^2q_3^3 + 10q_2^3q_3^2 + 200q_2^2q_3^2 + 100q_2^3q_3 \\
& +100q_2q_3^3 + q_1q_2q_3^2 + q_1q_2^2q_3 + 1000q_2q_3^2 \\
& +8q_1q_3^2 + 1000q_2^2q_3 + 8q_1q_2^2 + 6q_1q_2q_3 \\
& +60q_1q_3 + 60q_1q_2 - q_1^2 - 200q_1 \leq 0 \\
& 800 - 800k \leq q_1 \leq 800 + 800k \\
& 4 - 2k \leq q_2 \leq 4 + 2k \\
& 6 - 3k \leq q_3 \leq 6 + 3k
\end{array}
$$

optimal value	time(sec)	#LP	#pivot
0.341739725	0.14	11	270

(vi) Robust stability analysis 2

$$
\begin{array}{ll}
\text{minimize} & k \\
\text{subject to} & q_1^4q_2^4 - q_1^4 - q_2^4q_3 = 0 \\
& 1.4 - 0.25k \leq q_1 \leq 1.4 + 0.25k \\
& 1.5 - 0.20k \leq q_2 \leq 1.5 + 0.20k \\
& 0.8 - 0.20k \leq q_3 \leq 0.8 + 0.20k
\end{array}
$$

optimal value	time(sec)	#LP	#pivot
1.089870022	0.14	8	82

(vii)Robust stability analysis 3

$$
\begin{array}{ll}
\text{minimize} & k \\
\text{subject to} & q_3 + 9.625q_1\omega + 16q_2\omega + 16\omega^2 + 12 - 4q_1 - q_2 - 78\omega = 0 \\
& 16q_1\omega + 44 - 19q_1 - 8q_2 - q_3 - 24\omega = 0 \\
& 2.25 - 0.25k \leq q_1 \leq 2.25 + 0.25k \\
& 1.5 - 0.50k \leq q_2 \leq 1.5 + 0.50k \\
& 1.5 - 1.50k \leq q_3 \leq 1.5 + 1.50k
\end{array}
$$

optimal value	time(sec)	#LP	#pivot
0.817529482	0.07	19	192

(viii) COCONUT ベンチマーク問題（[2]）

AMPL で記述されたテスト問題を SIMPLE に書き直してテストを実施中である．現在のところ一部のみの実験となっている．計算時間リミットは 1800 秒としている．まず，Library 1 の size 1 のグループ（変数の数が 1-9）に対しては，試みた 83 問題中 76 問に対して正解が得られた（正答率=91.6%）．Library 1 の size 2 のグループ（変数の数が 10-99）に対しては，試みた 81 問題中 48 問に対して正解が得られた（正答率=59.3%）．[9] によると，これらの問題に対して，BARON はそれぞれ 64/88=72.7%，46/77=59.7%，LINGO はそれぞれ 70/91=76.9%,42/80=52.5%などとなっている．更に広い範囲の問題に対して計算実験を試みたい．

6　今後の課題

本論文で述べたようなスキームで，汎用的な大域的最適化問題のソルバーの構築が可能であることは明らかであろう．また，それほど大規模でなければ現在の計算機能力でかなり広範囲の問題を解くことができる．このような状況は，10〜20 年前の分枝限定法による整数計画法のソルバーの状況に似ているとも思える．その後の計算機のスピードアップとアルゴリズムの発展によって，整数計画法の数値解法は実用的なものになった．大域的最適化の分野でも，同様な経過を辿ることが十分に期待できる．ただし，そのために為すべきことも数多くある．我々の実装も，現在知られている知識・技術を十分に利用しているとは言えないし，候補となる改良法も無数にあると言える．以下に，必要なアルゴリズムの改良点（ほぼ全面的）をピックアップしてみる．

- 線形計画以外の凸緩和の方法（一般の凸計画，凸 2 次計画，２次錐計画など）の採用
- 分枝法に関連する問題（分枝変数の選択，分枝位置など）
- 本方法により適した非線形局所最適化法の開発
- 特定の関数パターンの抽出とその利用
- 中間変数の数の削減
- 探索領域の縮小
- 区間演算的方法の利用による，より厳密な求解
- 並列計算

その他にも色々考えられるし，更に新しい概念による新方法が現れる可能性もある．いずれにしてもこの分野の今後の発展は大いに期待できる．

References

[1] Adjiman C.S., S. Dallwig, C.A. Floudas, and A. Neumaier, ”A Global Optimization Method, αBB, for General Twice-Differentiable Constrained NLPs - I. Theoretical Advances”, *Computers and Chemical Engineering*, Vol. 22, pp. 1137-1158, 1998.

[2] The COCONUT Benchmark:

http://www.mat.univie.ac.at/~neum/glopt/coconut/benchmark.html

[3] C.A.Floudas, *Deterministic Global Optimization Theory, Methods and Applications*, Kluwer Academic Publishers, 2000.

[4] C.A.Floudas and P.M.Padalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Lecture Notes in Computer Science 455, Springer-Verlag, 1990.

[5] Global Optimization Test Problems:

http://www.mat.univie.ac.at/~neum/glopt/test.html

[6] R.B.Keafort, Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems, Computing, Vol.47, pp.169-191, 1991.

[7] R.B.Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer, 1997.

[8] G.P.McCormick, Computability of global solutions to factorable nonconvex programs: Part I - convex underestimating problems, *Mathematical Programming* Vol. 10, pp. 147-175, 1976.

[9] A.Neumaier, O.Shcherbina, W.Huyer and T.Vinko, A comparison of complete global optimization solvers, *Mathematical Programming*, Vol.103, pp335-356, 2005.

[10] H.S.Ryoo and N.V.Sahinidis, A branch-and-reduce approach to global optimization, Journal of Global Optimization, Vol.8,pp.107-139, 1996.

[11] N.V.Sahinidis, "BARON: A general purpose global optimization software package", *Journal of Global Optimization*, Vol. 8, No. 2, pp.201-205, 1996.

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

[13] R.G.Strongin and Y.D.Sergeyev, Global Optimization with Non-Convex Constraints, Kluwer Academic Publishers, 2000.

[14] M.Tawarmalani and N.V.Sahinidis, Covexification and Global Optimization in Continuout and Mixed-Integer Nonlinear Programming, Kluwer Academic Publishes, 2002.

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

[16] H.Yamashita, A globally convergent primal-dual interior point method for constrained optimization, *Optimization Methods and Software*, Vol. 10, pp.443-469, 1998.

[17] H.Yamashita and H.Dan, "Global convergence of a trust region sequential quadratic programming method", *J. Operations Research Soc. Japan*, Vol. 48, No. 1, pp. 41-56, 2005.

[18] 山下，田辺，逸見 " 数理計画のためのモデリング言語 SIMPLE Ｉ概要" OR 学会 1997 年春季研究発表会アブストラクト集.

[19] H.Yamashita, H.Yabe and T.Tanabe, A globally and superlinearly convergent primal-dual interior point trust region method for large scale constrained optimization, *Mathematical Programming, Ser.B*, Vol. 102, No. 1, pp.111-151, 2005.