数理解析研究所講究録
第1629巻 2009年 194-202

多項式記憶量による非線形大域的最適化

筑波大学大学院　システム情報工学研究科
対馬伊織，久野誉人

概要　連続な実数値関数を目的関数とし，閉集合を実行可能集合とする最適化問題を考える．この問題の大域的最適解を求めるアルゴリズムの1つに矩形分枝限定法がある．しかし，矩形分枝限定法はいくつもの矩形データを記憶するため，多くのメモリが必要となる．本研究では，矩形分割木がもつ性質を利用して使用メモリ量を問題規模の多項式に抑えるアルゴリズムを提案する．

1. はじめに

ユークリッド空間 $\mathbb{R}^n$ 上で定義された連続な実数値関数 f を目的関数とし，$\mathbb{R}^n$ の閉部分集合 D を実行可能集合とする最適化問題

$$\begin{aligned} &\text{最小化} \quad f(x) \\ &\text{条　件} \quad x=(x_1,\ldots,x_n)\in D \end{aligned}$$

を考える．目的関数 f と実行可能集合 D が凸である凸計画問題ならば，任意の局所解が大域的最適解となるので局所探索によって容易に答を求めることができる．一方，目的関数 f や実行可能集合 D が凸でない非凸計画問題では，局所解が複数存在し，大域的な最適解を求めるのが困難であることが多い [1, 3, 4]．その例を図1の目的関数 f と図2の実行可能集合 D で示す．この目的関数に等高線を引くと，原点からのユークリッド距離が大きいほど解が改善され，D の5つの端点で f は局所的に最適化されることが分かる．したがって，局所探索では初期解によって得られる解が変わり，この局所解が最適解であることを保証できない．

しかし，図1，2の例のように実行可能集合 D が凸多面体の場合，最適解が D の端点に現れるので，このことを求解に利用することができる．特に，目的関数 f が複数の1変数関数の和に分離可能な場合，現実的な計算時間で最適解を求めることが可能である．本研究では，このような最小化問題を問題規模の多項式の記憶量で解くアルゴリズムを提案する．2節で既存のアルゴリズムである矩形分枝限定法を説明し，3節で提案するアルゴリズムに必要な矩形分割の性質を挙げる．4節で提案するアルゴリズムを説明し，5節で計算実験により既存アルゴリズムと計算効率を比較する．

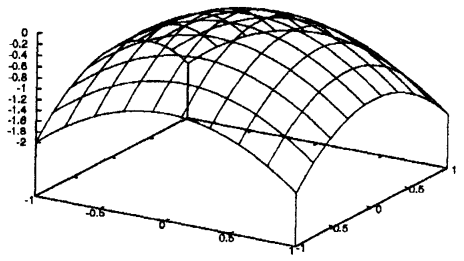

図 1: 凹関数の例 ($f(x) = -x_1^2 - x_2^2$)

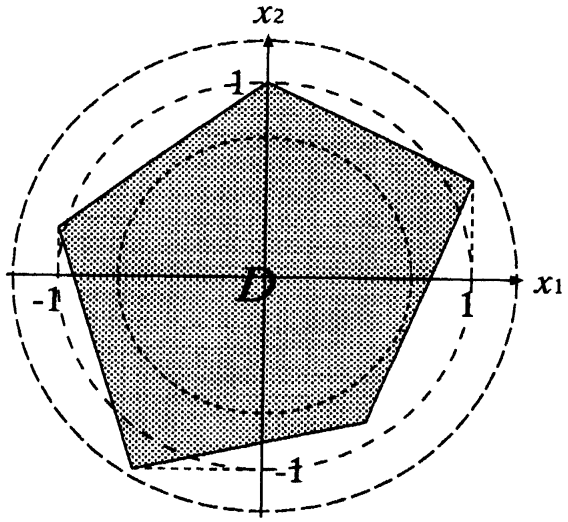

図 2: 凸多面体の例

2. 矩形分枝限定法

2.1 既存アルゴリズム

非凸計画問題の最適解を求めるアルゴリズムの 1 つに分枝限定法がある．分枝限定法は一種の列挙法で，実行可能領域を分割する操作の反復によって部分問題を生成し (分枝操作)，最適値の上下界値の情報を用いて最適解の出現する見込みのない部分問題を省く (限定操作) ことで，組合せ的爆発を克服するアルゴリズムである．凹最小化問題で分枝限定法を使う場合，目的関数 f が $f(x) = \sum_{i=1}^{n} f_i(x_i)$ のように 1 変数凹関数 f_i に分離可能なときには，矩形分枝限定法 [2] が最も効力を発揮する．実行可能集合 D を含むような矩形 M を，

$$M = \{x \in \mathbb{R}^n \mid p_i \leq x_i \leq q_i, \quad i = 1, \ldots, n\}$$

と定義し，これを小さな矩形

$$M^k = \{x \in \mathbb{R}^n \mid p_i^k \leq x_i \leq q_i^k\}, \quad k = 1, \ldots, K,$$

に分割していく．

基本となる操作は，まず M^k を選んで，
限定操作：$D \cap M^k$ 上における f の下界値 f' を計算し，その値がそれまでに得られた最も小さな目的関数値 f^o 以上ならば，M^k を考察の対象から外す；
分枝操作：$f' < f^o$ ならば，M^k を 2 つの矩形 M^{k_1}，M^{k_2} に分割する
を繰り返し，その部分問題を解く [2, 3, 4].

2.2 既存アルゴリズムの短所

しかし，このようなアルゴリズムを計算機で実装する場合，いくつもの矩形のデータを分割木として記憶するため，非常に多くのメモリが必要となる．しかし，分割した矩形 M^{k_1} あるいは M^{k_2} から分割前の矩形 M^k を再現できれば，M^k のデータをメモリ上に記憶する必要はない．

そこで，本研究では矩形の分割木が持つ性質をを利用し，分枝限定法で使用するメモリを問題規模の多項式に抑えるアルゴリズムを提案する．

3. 矩形の分割と性質

分枝限定法で分割の対象となる矩形を，

$$M = \{x \in \mathbb{R}^n \mid p_i \leq x_i \leq q_i, \quad i = 1, \ldots, n\}$$

として，その表し方と分割方法を定める．

3.1 矩形の表現

n 次元矩形を，
x：矩形の中で各成分が最小となる頂点の座標（以後，ベースと呼ぶ）．$x \in \mathbb{R}^n$
l：各方向の辺の長さ．$l \in \mathbb{R}^n$
となる x，l で定義する．

また，分割前の初期矩形 M（以後，ルートと呼ぶ）を x^0, l^0 とする．ただし，ルートについて，どの辺も他の辺の２倍未満，すなわち，

$$\forall i, j (1 \leq i, j \leq n); \quad l_i^0 < 2l_j^0$$

であることを仮定する．

3.2 矩形分割手続き

矩形の分割方法を以下のように定義する．

- 最も長い辺を２等分して２つの矩形に分割する．
- 最も長い辺が複数存在する場合は，添字が小さいものを分割する．
- 最も長い辺がしきい値 $\epsilon > 0$ 以下になるまでこれを繰り返す．

分割でできる２つの矩形を子として，２分木を作る．以下に，この２分木のノードの列挙を，少ない記憶量で行う方法を説明する．

3.3 矩形分割木の性質

次に，矩形の分割木が持つ性質を挙げる．

命題1 木に含まれる任意の矩形に対して，

$$\forall i,j(1 \leq i < j \leq n); \quad \frac{1}{2}l_j \leq l_i < 2l_j$$

が成り立つ．

（証明）矩形 x, l の子の矩形を x', l' とする．

i) 前提条件より，

$$\forall i,j(1 \leq i < j \leq n); \quad \frac{1}{2}l_j^0 \leq l_i^0 < 2l_j^0$$

ii) $\frac{1}{2}l_j \leq l_i < 2l_j$ のとき，

・$l_i \geq l_j$ の場合，

$$l_i' = l_i \quad \text{or} \quad \frac{1}{2}l_i$$

$$l_j' = l_j$$

$$\text{よって，} \quad \frac{1}{2}l_j' \leq l_i' < 2l_j'$$

・$l_i < l_j$ の場合，

$$l_i' = l_i$$

$$l_j' = l_j \quad \text{or} \quad \frac{1}{2}l_j$$

$$\text{よって，} \quad \frac{1}{2}l_j' \leq l_i' < 2l_j'$$

□

命題2 木に含まれる任意の矩形に対して，

$$\forall i(1 \leq i \leq n); \quad x_i - x_i^0 \text{は } l_i \text{の整数倍}$$

が成り立つ．

（証明）

i) 初期矩形に対し，

$$\forall i(1 \leq i \leq n); \quad x_i^0 - x_i^0 = 0 = 0 \cdot l_i^0$$

ii) $\forall i(1 \leq i \leq n); \quad x_i - x_i^0$ が l_i の整数倍のとき，

・i が分割する方向の場合，

$$l_i' = l_i$$

$$x_i' = x_i$$

よって，$x'_i - x^0_i$ は l'_i の整数倍．
・i が分割する方向でない場合，

$$l'_i = \frac{1}{2} l_i$$

$$x'_i = x_i, \quad x_i + \frac{1}{2} l_i$$

$$x'_i - x^0_i = x_i - x^0_i, \quad x_i - x^0_i + \frac{1}{2} l_i$$

$x'_i - x^0_i$ は $l_i = 2l'_i$ の整数倍，　$\frac{1}{2} l_i = l'_i$ であるから，
$x'_i - x^0_i$ は l'_i の整数倍．

□

4. アルゴリズムの改訂

4.1 子ノードの生成

矩形 x, l の子を求める．
分割手続きにより，分割する辺の添字 i は，

$$i \in \arg_k \max l_k$$

$\arg_k \max l_k$ が複数存在する場合はその中で最も小さいものである．
同じく，分割手続きにより，

$$l'_i = \frac{1}{2} l_i$$

$$x'_i = x_i, \quad x_i + l'_i$$

となる．また，$j \neq i$，$1 \leq j \leq n$ をみたす j に対し，

$$l'_j = l_j$$

$$x'_j = x_j$$

となる．

4.2 親ノードの再現

木に含まれるルートでない矩形 x, l の親を求める．
矩形 x, l の親の矩形を $pr(x)$，$pr(l)$ とする．

(a) 前節より，ある $i(1 \leq i \leq n)$ に対して，

$$\begin{array}{l} pr(l_i) = 2l_i \\ pr(x_i) = x_i \quad \text{or} \quad x_i - l_i \\ pr(l_j) = l_j \\ pr(x_j) = x_j \end{array} \qquad (j \neq i, \quad 1 \leq j \leq n)$$

となる矩形以外は親の可能性はない．
(b) 命題１と分割手順より，

$$i = \max \arg_k \min l_k$$

である．
(c) 命題２より，

$(x_i - x_i^0)$ が l_iの偶数倍ならば，

$$pr(x_i) = x_i$$

$(x_i - x_i^0)$ が l_iの奇数倍ならば，

$$pr(x_i) = x_i - l_i$$

(a)，(b)，(c) より，親の矩形を一意に求めることができる．
しかし，２分木を列挙するには，親を求めた矩形がその親の左の子，右の子のどちらであるかを明らかにする必要がある．そこで，$x_i' = x_i, x_i + \frac{1}{2}l_i$ から，$x_i' = x_i$ となる矩形を左の子と定めれば，ベースの一致/不一致により，どちらの子であるかがわかる．

4.3 矩形列挙アルゴリズム

子ノードから親のノードを求めることができるようになり，左右どちらの子ノードから親ノードが再現されたかも明らかになったので，ここで記憶量を抑えた列挙アルゴリズムを提案する．縦型探索と同様の経路をたどるようにすると，

- 最初はルートから左の子ノードへ移動する．
- 子へ移動した次は，子が存在すれば左の子へ移動する．
- 子が存在しなければ親ノードへ移動する．
- 左の子から親へ移動した次は，右の子へ移動する．
- 右の子から親へ移動した次は，親が存在すれば親へ移動する．
- 右の子から親へ移動した次に，親が存在しなければ終了する．

という規則に従うことで，深さ１以上の完全２分木をたどることができる．また，最初に訪れた子ノードでその矩形情報を表示すれば，分割矩形の列挙も可能である．

5. 数値実験

初期矩形としきい値を与えて，矩形分割木のノードの列挙問題を，前章のアルゴリズムを用いた縦型探索，一般の縦型探索，横型探索で行い，実行時間と使用メモリを比較した．

また，分離可能な凹最小化問題を，提案するアルゴリズムと縦型探索での既存アルゴリズムで解かせて，実行時間と使用メモリを比較した．以下では n を次元とする．

5.1 実験結果（矩形分割木の列挙）

初期矩形 $l^0 = (4, 4, \ldots, 4)$，$\epsilon = 1$ として実験を行った．その結果，実行時間は表 1，使用メモリは表 2 のようになった．CP は本研究で提案するアルゴリズム，DFS は縦型探索，BFS は横型探索である．この表から，本研究のアルゴリズムが縦型探索，横型探索の定数倍の時間で実行できることが読み取れる．また，次元が増えるに従って使用メモリの差が顕著になっていくことが分かる．

表 1: 実行時間 (秒)

	n=2	n=4	n=6	n=8	n=10
CP	0.0100	0.0970	1.47	23.4	376
DFS	0.00700	0.0310	0.413	6.55	106
BFS	0.00700	0.0370	0.462	7.39	120

表 2: 使用メモリ (バイト)

	n=2	n=4	n=6	n=8	n=10
CP	136	216	296	376	456
DFS	336	848	1616	2472	3720
BFS	1152	32960	786688	16777536	335544704

5.2 分離可能凹最小化問題

次の凹最小化問題を提案するアルゴリズムで解いた：

$$\begin{aligned} \text{最小化} \quad & -\textstyle\sum_{i=1}^{n} c_i x_i^2, \quad c > \mathbf{0} \\ \text{条　件} \quad & Ax \leq \mathbf{1} \\ & x \geq \mathbf{0} \end{aligned}$$

$$(A \in \mathbb{R}^{m \times n}, \quad c \in \mathbb{R}^n)$$

この目的関数 f は，

$$f(x) = \sum_{i=1}^{n} f_i(x_i)$$

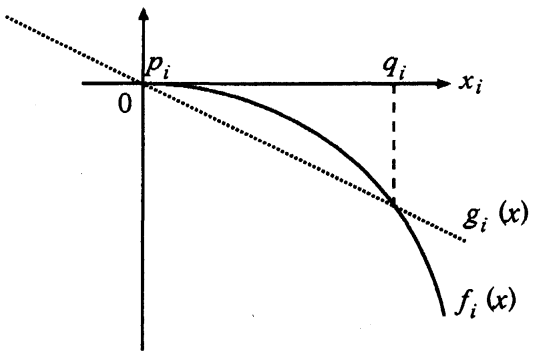

図 3: 線形緩和の例

のように $f_i(x_i) = c_i x_i^2$ の和に分離可能である．よって，M 上で f の下界値を与える関数 g は

$$g(x) = \sum_{i=1}^{n} g_i(x_i)$$

g_i：区間 $[p_i, q_i]$ 上における f_iの線形下界値関数

として簡単に求めることができる（図 3）．

矩形 M^k における部分問題は，

- M^k と目的関数 f から下界値関数 g を計算する．
- $D \cap M$ 内で g を最小化する解を求め，x^* とする．

ことで得られる x^* における f の関数値を評価する．g は線形関数，$D \cap M$ は凸集合であるから，x^* はシンプレックス法などで効率よく求めることができる．その後，以下のようにして矩形を移動する．

- 暫定最適値 $z \leq g(x^*)$ なら限定操作として，親の矩形を計算する．
- $z > f(x^*)$ なら z を更新．
- $z > g(x^*)$ なら分枝操作として，子の矩形を計算する．

目的関数の c_i を区間 $(0, 1)$ からランダムに選び，計算実験を行ったところ，以下のような結果が得られた．表 3 は実行時間，表 4 は使用メモリ量を表す．CP は提案アルゴリズム，DFS は既存アルゴリズムである．これらの表から，提案するアルゴリズムが定数倍の実行時間しか必要とせず，その一方，使用メモリは大幅に抑えられることが分かる．

表 3: 実行時間 (秒)

	n=2	n=4	n=6	n=8	n=10	n=12
CP	0.0113	0.0203	0.0593	1.51	28.7	38.8
DFS	0.00882	0.0112	0.0240	1.57	15.3	22.0

表 4: 使用メモリ (バイト)

	n=2	n=4	n=6	n=8	n=10	n=12
CP	1240	3000	5608	8920	13000	17848
DFS	1192	2904	8744	50416	74712	108456

6. まとめと今後の課題

本研究では，既存の分枝限定法の使用メモリ量が多いという短所を解消する方法として，矩形の分割木の性質を利用し，使用メモリ量を問題の次元の多項式で抑える分枝限定法を提案した．今後の課題としては，提案したアルゴリズムの高速化や，一般化した問題への応用などを考えている．

参考文献

[1] Dimitri P. Bertsekas, Angelia Nedic and Asuman E. Ozdaglar, *Convex Analysis and Optimization* (Athena Scientific, 2003).

[2] Falk,J.E. and R.M.Soland, "An algorithm for separable nonconvex programing problems." *Management Science*15(1969), 550-569.

[3] Horst,R., P.M.Pardalos and N.V.Thoai, *Introduction to Global Optimization* (Kluwer Academic Publishers, 1995).

[4] Horst,R. and H.Tuy, *Global Optimization: Deterministic Approaches*(Springer-Verlag, 1993).

[5] 宇野毅明，「列挙アルゴリズム」，オペレーションズ・リサーチ vol.52(2007) pp.531-534.