日本ソフトウェア科学会第 29 回大会 (2012 年度) 講演論文集

制約不足の数値制約充足問題のための区間計算に基づく射影手法

石井 大輔　アレクサンドル ゴールドシュタイン
クリストフ ジャーマン

数値制約充足問題 (numerical CSP) において制約不足 (under-constrained) の場合，一般に解集合は連続集合となる．本研究ではそのような解集合の射影について，区間計算により外部近似と内部近似を求める手法を提案する．提案手法は branch-and-prune の枠組みに基づく．等式・不等式制約からなる一般的な問題への適用可能性，内部近似の効率的な計算，逐次的に精度を改善していく求解プロセス等を特徴としている．提案手法は以下の 3 つの拡張からなる: (1) 射影空間の内部 box の高精度な同定，(2) 解集合を射影する際に生じる解の重なりを考慮した探索空間削減，(3) 射影計算を高速化する探索ヒューリスティクス．また実験結果から，応用的な問題を射影問題として定式化し，提案手法により効率よく扱えることを示す．

1 はじめに

制御工学やロボティクス分野等において，数値制約充足問題 (2.2 節，以下「NCSP」) による定式化とその求解法が有用である．当該分野の問題を記述する NCSP はしばしば制約不足 (under-constrained) となる．すなわち，変数の次元よりも制約の次元が小さくなる．制約不足の NCSP では，解が単一の値ではなく (連続) 集合として与えられるため，解集合の射影が問題の解析において役立つ．たとえば，問題の次元が大きい場合に射影を可視化したり，問題が性質の異なる変数 (例: 入力・出力変数) を持つ場合に各変数毎の射影を分析したりすることができる．

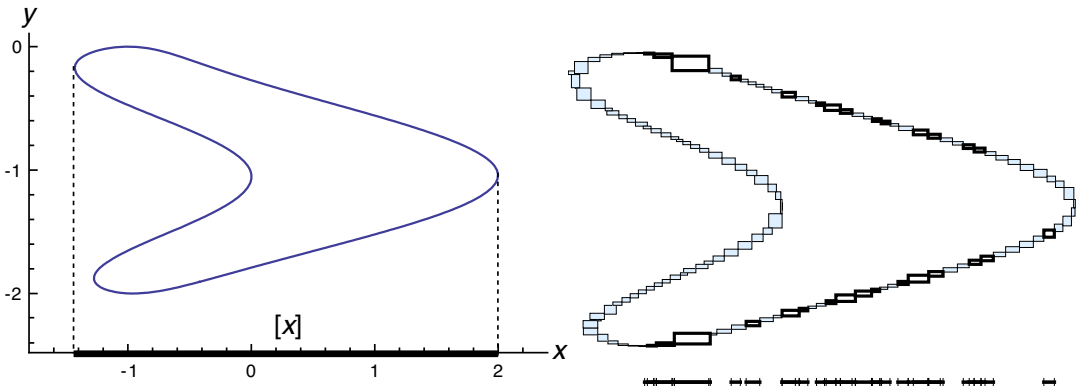
制約が多項式で記述される場合には記号計算に基づき解析的に射影計算ができる (例: [2])．しかし記号計算では扱うべき記号的表現のサイズが変数・制約の数に応じて増大するため，小規模な系にしか適用できないという問題がある．

一方で，より大きな系や非線形制約を含む系を扱うために，区間計算 (2.1 節) に基づく手法が発展してきた (例: [9] [6] [3])．これらの手法では射影を box (区間ベクトル) の集合により近似する．計算結果の box 集合は射影を数値計算誤差も含めて保守的に近似するが (外部近似)，一部の box について射影内部に含まれることが保証することにより内部近似も得ることができる．既存の区間計算に基づく手法では，等式制約のみ [3]・不等式制約のみ [9] を扱っていたり，変数の出現回数が限られていたり [6]，多くの内部 box の判定に失敗する [3] といった問題があった．

本論文では，区間計算に基づく NCSP 求解のための基本アルゴリズムである BranchAndPrune (2.2 節) を用いて解集合の射影を計算する手法を提案する (第 3 節)．提案手法は既存手法よりも広範な等式および不等式制約からなる系を扱うことができる．また提案手法では内部 box の同定に独自の方法を用いており，境界 box 数の大幅な減少を実現している．提案手法は BranchAndPrune アルゴリズムの各サブルーチンを射影計算のために拡張するというアプローチを取っているため，BranchAndPrune の利点やチューニング技法を引き継いでいる．その利点の 1 つとして，逐次的に近似結果を高精度化していく anytime 計算の

Interval-Based Projection Method for Under-Constrained Numerical CSPs.

Daisuke Ishii, 国立情報学研究所, National Institute of Informatics.
Alexandre Goldsztejn, Christophe Jermann, ナント大学計算機科学研究所, LINA, University of Nantes.



(a) 解集合とその射影　(b) 単純な BranchAndPrune 実装による計算結果

(c) 提案する内部 box 同定手法を用いた計算結果　(d) 提案する box 削減手法を用いた計算結果

図 1　射影問題の BranchAndPrune アルゴリズムによる計算

実現が挙げられる．第 4 節の実験結果では提案手法の応用的な問題への適用可能性と性能を評価する．

1.1　簡単な例題

変数 $(x, y) \in [-2, 3] \times [-3, 1]$ と制約

$$(x + \cos 3y)^2 + (y + 1)^2 - 1 = 0$$

を考える．この制約を満たす解の集合は図 1(a) に示すような 2 次元の曲線となる．ここではこの解集合を x 軸上に射影した区間 $[x]$ を求めることを考える．

図 1(b) は BranchAndPrune アルゴリズムの標準的実装により解集合を計算した結果を示している．計算結果は曲線を包む box の集合となり，各 box の x 成分の和集合をとることで解集合の射影 $[x]$ の外部近似を得る．太線で示した box はその射影が $[x]$ の内部であることを，単純な区間ニュートン法を用いて判定することができた box (内部 box) で，細線の box は判定できなかった box である．内部 box の x 成分の和集合をとることで $[x]$ の内部近似を得る．図の下部に示したように，得られた内部近似は所々結合しておらず精度がよくない．

図 1(c) は提案する内部 box の同定手法 (3.1 節) を組み込んだ BranchAndPrune アルゴリズムによる計算結果である．提案手法により多くの内部 box が同定できている．また同定できなかった内部 box は，射影に直交するような特異解の近くに位置している．

図 1(d) は前述の手法に加えて提案する探索空間削減手法 (3.2 節) を組み込んだ BranchAndPrune アルゴリズムによる計算結果である．この問題では射影内の各点で高々4 つの解が重なりうるが，ある解を包む内部 box を 1 つ計算できれば射影の計算には十分である．そこで先に同定された内部 box を利用して重なった解を含む探索空間を削減することにより，計算する box の数を大幅に減らすことができる．

2 基本技術

2.1 区間計算

区間計算 [8] [7] は浮動小数点数の計算を区間の計算に拡張したものである．区間計算は実数計算の保守的近似であり，結果として得られる区間は実数計算がとりうるすべての結果を計算誤差とともに包む．区間 $[x] = [l, u]$ により集合 $\{r \in \mathbb{R} \mid l \leq r \leq u\}$ を表す．$\mathbb{IR}$ ですべての区間の集合を表す．区間 $[x]$ について，$\mathrm{ub}([x])$ と $\mathrm{lb}([x])$ で上限と下限を表し，$\mathrm{wid}([x])$ で幅 $\mathrm{ub}([x]) - \mathrm{lb}([x])$ を，$\mathrm{int}([x])$ で内部 $\{\tilde{x} \in \mathbb{R} \mid \mathrm{lb}([x]) < \tilde{x} < \mathrm{ub}([x])\}$ を，$\mathrm{mid}([x])$ で中心値 $\frac{\mathrm{ub}([x])+\mathrm{lb}([x])}{2}$ を表す．区間 $[x_1]$ と $[x_2]$ について，$[x_1] \setminus [x_2]$ で差分 $\{\tilde{x} \in \mathbb{R} \mid \tilde{x} \in [x_1], \tilde{x} \notin \mathrm{int}([x_2])\}$ (高々2 区間で表される) を表す．これらの各定義は区間ベクトル用に拡張できる．

本稿では $\mathbf{x}$ と $[\mathbf{x}]$ で実ベクトルと区間ベクトルを表す．d 次元 **box** (区間ベクトル) $[\mathbf{x}]$ は d 区間の組 $([x_1], \ldots, [x_d])$ である．$\mathbb{IR}^d$ ですべての d 次元 box の集合を表す．実ベクトル $\mathbf{x} \in \mathbb{R}^d$ と box $[\mathbf{x}] \in \mathbb{IR}^d$ について，$\mathbf{x} \in [\mathbf{x}]$ という表記を $\forall i \in \{1, \ldots, d\}$ $(x_i \in [x_i])$ と解釈する．

関数 $f : \mathbb{R}^d \to \mathbb{R}$ について，関数 $[f] : \mathbb{IR}^d \to \mathbb{IR}$ が

$$\forall [\mathbf{x}] \in \mathbb{IR}^d \ \forall \mathbf{x} \in [\mathbf{x}] \ (f(\mathbf{x}) \in [f]([\mathbf{x}]))$$

を満たすときかつそのときに限り，f の区間拡張という．この定義は関数ベクトル $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^e$ 用に拡張できる．

2.2 数値制約プログラミング

数値制約充足問題 (numerical constraint satisfaction problem, NCSP) を下記の要素からなる 3 つ組 $(\mathbf{v}, [\mathbf{v}_0], c)$ で表す:

- d 変数の組 $\mathbf{v} = (v_1, \ldots, v_d)$.
- d 次元 box として与えられる初期ドメイン $[\mathbf{v}_0]$.
- 制約 $c \equiv \mathbf{f}(\mathbf{v}) = 0 \wedge \mathbf{g}(\mathbf{v}) \geq 0$ (ただし $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^p$, $\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^q$).

本稿では d, p, q で問題中の変数，等式，不等式の数を表す．制約を満たす変数 $\mathbf{v}$ への値 $\tilde{\mathbf{v}} \in [\mathbf{v}_0]$ の付値を NCSP の解という．また集合 $\Sigma := \{\tilde{\mathbf{v}} \in [\mathbf{v}_0] \mid c(\tilde{\mathbf{v}})\}$ を解集合という．NCSP は，$d < p$ がなりたつとき制約過多 (over-constrained)，$d = p$ がなりたつとき制約の過不足がない (well-constrained)，$d > p$ がなりたつとき制約不足 (under-constrained) であるという．一般に制約不足の NCSP は連続な解集合を持つ．

```
Input: NCSP (v, [v0], c)
Output: box 集合の組 (L, S)
    L ← {[v0]}
    S ← ∅
    while ¬Stop() do
      [v] ← Extract(L)
      [v] ← Prune([v])
      if [v] ≠ ∅ then
        if Prove([v]) then
          S ← S ∪ {[v]}
        else
          L ← L ∪ Branch([v])
        end if
      end if
    end while
    return (L, S)
```

図 2 BranchAndPrune アルゴリズム

図 2 の BranchAndPrune アルゴリズムは NCSP の標準的な求解手法である．アルゴリズムは NCSP を入力として受け取り，解集合を包む box の集合を出力する．出力される box 集合は，アルゴリズムが解の存在保証に成功した box の集合 S と，保証することができなかった box の集合 L からなる．BranchAndPrune は 5 つのサブルーチンを持ち，問題や求解アプローチに応じて実装を用意する．

- Stop は繰り返しを終了するために，L 中の box の精度チェックやタイマーとして実装する．
- Extract は各繰り返しにおいて，解の探索空間である L から box $[\mathbf{v}]$ を選択する．この Extract と後述する Branch の実装によって，深さ優先探索，幅優先探索，box 幅優先探索等の探索法を設定することができる．
- Prune は探索空間から制約と矛盾する付値を除去する処理である．NCSP の求解では無矛盾性 (consistency) に基づき，ある区間の端部分を取り除く．本研究では数値制約のための局所的な無矛盾性である Hull 無矛盾性に基づく実装 [1] を用いる．Prune の計算は提案手法の計算とは独立しているため任意の実装を利用できるが，次節で

提案する Prove と Extract の実装でも探索の打ち切りや探索空間の削減を行うことから (3.2 節参照)，高速な Hull 無矛盾性を用いることとした．

- Prove は区間ニュートン法等を用い，box $[\mathbf{v}]$ 内の解の存在保証を行う．保証に成功した場合は S に追加し，探索を打ち切る．本研究における解の存在保証とは，box が解集合を射影した領域の内部に含まれるかどうかの判定である．
- Branch は box $[\mathbf{v}]$ を 2 分割する．分割された box は L に戻され，以降各 box について探索を行う．分割の際，box のどの辺を選ぶか (変数選択) が探索の効率化のために重要となる．

3 射影問題のための BranchAndPrune 実装

射影問題 (projection problem) とは制約不足の NCSP の解集合の射影を計算する問題である．ある NCSP$(\mathbf{v}, [\mathbf{v}_0], c)$ について，変数 $\mathbf{v}$ を $\mathbf{x} = (x_1, \ldots, x_n)$ と $\mathbf{y} = (y_1, \ldots, y_m)$ のように 2 分して与え，解集合の $\mathbf{x}$ 上の射影

$$\Sigma_{\mathbf{x}} := \{\mathbf{x} \in [\mathbf{x}_0] \mid \exists \mathbf{y} \in [\mathbf{y}_0]\ (c(\mathbf{x}, \mathbf{y}))\}$$

を求めることを考える．ただし $[\mathbf{x}_0]$ は初期ドメイン $[\mathbf{v}_0]$ の $\mathbf{x}$ 成分，$[\mathbf{y}_0]$ は $\mathbf{y}$ 成分を表す．本稿では制約 c が $p = m$ であるような p 個の等式制約と任意個の不等式制約からなる射影問題を扱う．†1 等式制約の個数を制限することにより射影 $\Sigma_{\mathbf{x}}$ の次元が常に n となり，射影が非 0 の容積を持つことが保証される．

提案する射影問題のための BranchAndPrune 実装は，射影 $\Sigma_{\mathbf{x}}$ の内部近似と外部近似を計算する．すなわち計算結果の L と S について

$$\cup S_{\mathbf{x}} \subseteq \Sigma_{\mathbf{x}} \subseteq \cup(L_{\mathbf{x}} \cup S_{\mathbf{x}})$$

がなりたつ．ここで $S_{\mathbf{x}} := \{[\mathbf{x}] \mid ([\mathbf{x}], [\mathbf{y}]) \in S\}$ は S 中の box の $\mathbf{x}$ 射影，$L_{\mathbf{x}}$ は同様に L 中の box の $\mathbf{x}$ 射影である．以下，BranchAndPrune アルゴリズムのサブルーチン群 Prove (3.1 節, 3.2 節)，Extract (3.2 節)，Branch (3.3 節) の実装について説明する．

3.1 内部 box の同定

Prove は与えられた box $([\mathbf{x}], [\mathbf{y}])$ が射影の内部 box であるかどうか，すなわち

$$\forall \mathbf{x} \in [\mathbf{x}]\ \exists \mathbf{y} \in [\mathbf{y}]\ (\mathbf{f}(\mathbf{x}, \mathbf{y}) = 0) \tag{1}$$

を証明できるかどうかを判定する． 証明にはパラメトリック区間ニュートン法を用いる．ここでは区間ニュートン法の一種である Hansen-Sengupta 法 [5] のパラメトリック版を用い，以下のような漸化式を計算する:

$$[\mathbf{y}^k] := [H]_{[\mathbf{x}]}([\mathbf{y}^{k-1}]) \cap [\mathbf{y}^{k-1}]. \tag{2}$$

ただし，初期値は $[\mathbf{y}^0] := [\mathbf{y}]$ とし，$[H]_{[\mathbf{x}]}$ は系 $[\mathbf{f}]([\mathbf{x}], \cdot) = 0$ に対して適用した Hansen-Sengupta 作用素とする．$\emptyset \neq [\mathbf{y}^k] \subseteq \mathrm{int}([\mathbf{y}^{k-1}])$ がなりたつとき，box $([\mathbf{x}], [\mathbf{y}^k])$ は条件 (1) をみたすことが保証される．図 3 の左図はこのときの box の計算例を示している (実線の box が $[\mathbf{y}^{k-1}]$，破線の box が $[H]_{[\mathbf{x}]}([\mathbf{y}^{k-1}])$)．Hansen-Sengupta 作用素が box の $\mathbf{y}$ 成分の両端を削減していることが確認できる．また漸化式 (2) は $\mathbf{y}$ 成分を削減しているだけなので元の box $([\mathbf{x}], [\mathbf{y}])$ も条件をみたす．box が複数個の解を含む場合にこの判定はうまくいかないが，そのときは解を 1 つだけ含むように BranchAndPrune アルゴリズムの区分け処理を適用する必要がある．

しかし，1.1 節の図 1(b) に示したように，上記計算は多くの内部 box の同定に失敗する．図 3 の右図は同定が失敗する例を示しており，Hansen-Sengupta 作用素が box の $\mathbf{y}$ 成分をずらすため，片端の削減しか観察できない．そこで提案する実装では，[4] で提案された方法を応用し，漸化式 (2) を以下のように変更する:

$$[\mathbf{y}^k] := [H]_{[\mathbf{x}]}([\tilde{\mathbf{y}}^{k-1}]), \text{ただし} \tag{3}$$

$$[\tilde{\mathbf{y}}^{k-1}] := \mathrm{mid}([\mathbf{y}^{k-1}]) + \tau([\mathbf{y}^{k-1}] - \mathrm{mid}([\mathbf{y}^{k-1}])). \tag{4}$$

$\emptyset \neq [\mathbf{y}^k] \subseteq \mathrm{int}([\tilde{\mathbf{y}}^{k-1}])$ がなりたつとき，box $([\mathbf{x}], [\mathbf{y}^k])$ は条件 (1) をみたす．式 (3) では，Hansen-Sengupta 作用素の結果と前項との積集合をとらないことにより，$\mathbf{y}$ 成分の両端から解を包むように box $[\tilde{\mathbf{y}}^{k-1}]$ を移動あるいは拡大することを許している．

さらに，条件 (1) の証明のために box $[\mathbf{y}^k]$ の強い削減を観察する必要があるため，式 (4) において box を微小に拡大している．本研究では $\tau = 1.01$ とした．漸化式 (3) による box の削減は，box の両端が解集合を包んだ後は 2 次収束することが期待される．そ

†1 提案手法の簡単な拡張により $p < m$ の場合も扱えることを確認している．

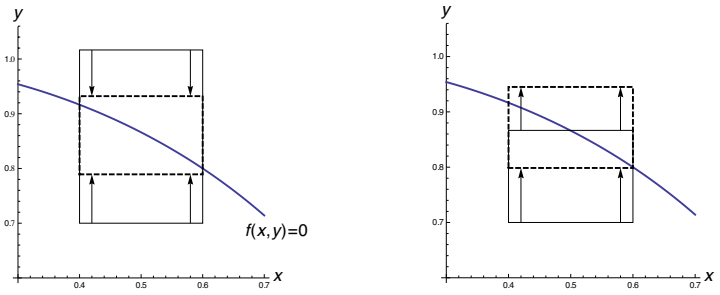


図 3 パラメトリック **Hansen-Sengupta** 法による内部 **box** の同定 (左: 成功例，右: 失敗例)

こで提案する Prune 実装では最近 2 回の計算による box の削減幅を記録し，それらの比が定数 0.9 以下である限り計算を続けるようにしている．

Prune では制約 $\mathbf{g}(\mathbf{x},\mathbf{y}) \geq 0$ の評価も行う．入力された box $([\mathbf{x}],[\mathbf{y}])$ について $[\mathbf{g}]([\mathbf{x}],[\mathbf{y}])$ を計算し，強く 0 より大きいかどうかを評価する．

3.2 探索空間 (box) の削減

BranchAndPrune アルゴリズムでは Prune 手続きが無矛盾性に基づき探索空間 (box) の削減を行うが，提案する実装ではさらに，Prove と Extract の各手続きにおいても探索空間を削減することができる．

前節で述べたように Prove は box $([\mathbf{x}],[\mathbf{y}])$ の $\mathbf{y}$ 成分に Hansen-Sengupta 作用素を適用する．この計算結果 $([\mathbf{x}],[H]_{[\mathbf{x}]}([\tilde{\mathbf{y}}^k]))$ は証明の成否にかかわらず元の box $([\mathbf{x}],[\mathbf{y}])$ 内の解集合を包んでいる．そこで，Prove の計算の最後に計算結果と元の box との積集合を求め，元の box をこれで置き換えることにより box の $\mathbf{y}$ 成分を削減することができる．

Extract は前述のように L から box を取り出す．射影において複数解が重なっている場合，すなわち射影内の 1 点が複数解に対応している場合，1 つ内部 box が求まれば重なっている他 box に対する処理を省略することができる．この考察に基づき実装では，L から box $([\mathbf{x}],[\mathbf{y}])$ を取り出した際，S 中の内部 box $([\mathbf{x}'],[\mathbf{y}'])$ との差分 $([\mathbf{x}] \setminus [\mathbf{x}'],[\mathbf{y}])$ を計算し，元の box をこれで置き換える．ただし一般に box 同士の差分結果は複数 box になるが，効率のために 2 個以上の box になる場合は本処理を適用しないことにする．また提案する BranchAndPrune 実装では，ある box について S 中の $\mathbf{x}$ 成分が重なった box を効率よく見つけるため，各 box が射影上で重なっている他 box へのリンクのリストを持つようにしている．

3.3 探索手法

Branch は L 中の各 box について変数リストを管理し，ラウンドロビン方式で変数を 1 つ選択し，選択した変数に対応する辺で box の分割を行う．射影の計算においては，box の $\mathbf{x}$ 成分が最終的に出力され，$\mathbf{y}$ 成分は Prove の処理でのみ用いられるため，$\mathbf{y}$ 成分は 3.1 節で述べた複数解を含む場合にのみ分割を行い，その他の場合には $\mathbf{x}$ 成分を分割して精度を改善するのが望ましい．しかし，$\mathbf{y}$ 成分を分割すべきかどうかの判定は困難なため，提案する Branch 実装では，変数リストを $\mathbf{x}$ と $\mathbf{y}$ とで別々に管理し，$\mathbf{x}$ 成分を $\mathbf{y}$ 成分よりも頻繁に選択するというヒューリスティックを用いた．さらに，解集合の射影の端や特異解の近傍では Prove 処理が成功しなくなり，$\mathbf{x}$ 成分の精度を相対的に改善する必要があることから，$\mathbf{x}$ と $\mathbf{y}$ 成分の選択頻度を 3.3 節で述べた重なった box 間のリンク数に応じて決めるようにした．つまり射影の端において $\mathbf{y}$ 成分の分割が進むに従い $\mathbf{y}$ 成分の選択頻度が落ち，その分 $\mathbf{x}$ 成分の分割が進む．

4 実験結果

提案手法を区間演算ライブラリ Gaol[†2] と BranchAndPrune を実装したライブラリ RealPaver[†3]

†2 `http://sourceforge.net/projects/gaol/`.

†3 `http://pagesperso.lina.univ-nantes.fr/~granvilliers-l/realpaver/`.

を利用して C++で実装した.

以下，応用的な問題を NCSP として記述し上記実装で求解した例を紹介し (4. 1 節)，計算結果から提案手法の性能評価を行う (4. 2 節)．実験は Intel Xeon 3.4GHz と RAM 16GB を備えたマシンで行った.

4. 1 例題

制約不足の NCSP の例を 2 つ紹介し，その求解結果を示す．求解では Stop の実装として，L 中の最大 box 幅が 0.01 以下かどうかを判定している．図 4, 5 では，内部 box と判定されたものは内側を白く，そうでないものは黒く塗りつぶしている.

4. 1. 1 マニピュレータの可動範囲

最初の問題は図 4 に示すような簡単なマニピュレータの可動範囲を計算する問題である [4]．マニピュレータは点 A を中心に回転可能な長さ 2 のバー AB からなり，点 A は線分 DE 上を移動できる．また不等式制約によりマニピュレータのバーと，点 C を中心とし半径 1 の円との衝突回避を表現する．マニピュレータのモデル化にあたり，可動範囲計算の対象となる点 B の位置を変数 $\mathbf{x} \in [-10, 10]^2$ で表し，マニピュレータの制御パラメタである点 A の位置とバーの水平に対する角度をそれぞれ変数 $y_1 \in [0, 4]$ と $y_2 \in [-2, 2]$ で表す (A の位置は (y_1, y_1) となる)．モデルを以下の制約で記述する:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{x} - \begin{pmatrix} y_1 + 2\cos y_2 \\ y_1 + 2\sin y_2 \end{pmatrix} = 0$$
$$\wedge\ \mathbf{g}(\mathbf{x}, \mathbf{y}) = d^2(C, A, \mathbf{x}) - 1 \geq 0.$$

関数 $d(\mathbf{p}, \mathbf{a}, \mathbf{b})$ は点 $\mathbf{p}$ と線分 $\mathbf{ab}$ 間の距離を内積を用いて計算する関数である．提案手法により計算した解集合の $\mathbf{x}$ 射影を図 4 に示す．計算には 13 秒を要した.

4. 1. 2 帆船の速度ダイアグラム

2 番目の問題は，制御パラメタのドメイン内での帆船の振る舞いをモデル化した問題である [6]．モデルは以下のような制約で記述される:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} f_s \sin y_1 - f_r \sin y_2 - 60x_1 \\ f_s(1 - \cos y_1) - f_r 2\cos y_2 \end{pmatrix} = 0.$$

ただし，$f_s := 100(10\cos(x_1 + y_1) - x_2 \sin y_1)$, $f_r := 300x_2 \sin y_2$ とする．変数 $x_1 \in [0, 2\pi]$ と $x_2 \in [0, 20]$ は帆船の進行方向と速度を表し，変数

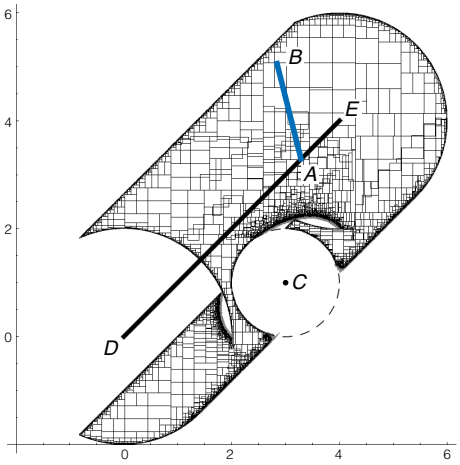


図 4 マニピュレータの計算結果

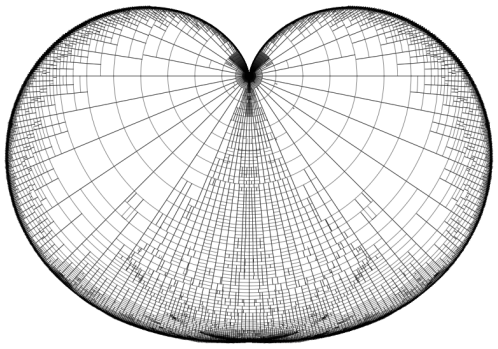


図 5 帆船の計算結果

$y_1 \in [-\pi/2, \pi/2]$ と $y_2 \in [-\pi/2, \pi/2]$ は制御パラメタである．提案手法により計算した解集合の $\mathbf{x}$ 射影を図 5 に示す．図では x_1 の値を角度で示している．計算には 1.1 分を要した.

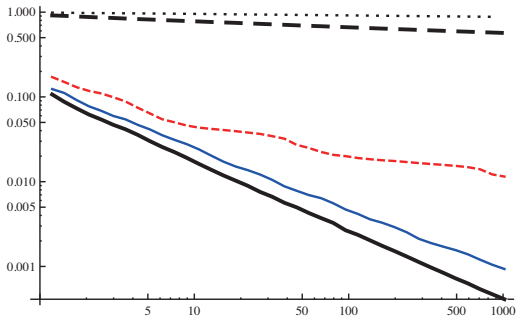
4. 2 性能評価

3 章で述べた BranchAndPrune 実装の性能を評価するために，提案した実装方法を下記の 3 要素に分け，すべて適用した場合と各提案方法を適用しなかった場合との比較を行った:

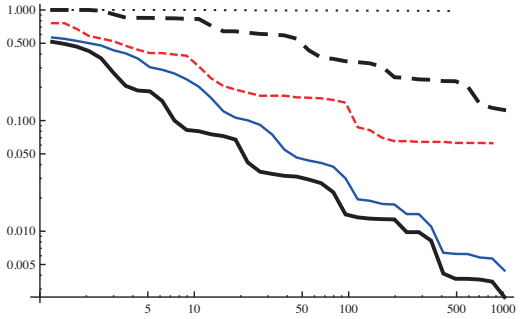
提案 1 3. 1 節で述べた内部 box 同定処理の改良.

提案 2 3. 2 節で述べた box 間の差分計算による探索空間の削減.

提案 3 3. 3 節で述べた $\mathbf{x}$ 変数と $\mathbf{y}$ 変数の表を別に管理した 2 重ラウンドロビン変数選択.



(a) マニピュレータ ($\tilde{v} \approx 22.57$)



(b) 帆船 ($\tilde{v} \approx 25.98$)

図 6 各計算の収束度 (両対数グラフ, 横軸は時間, 縦軸は実際の容積 $\tilde{v}$ に対する内部近似の割合)

実験では，Stop を 1000 秒までの実行時間タイマーとして実装し (anytime 計算)，計算の収束度を計測，比較した．ここで収束度とは，各時刻 t までに計算した内部 box の全容積 $v(t)$ が，解集合の射影の実際の容積 $\tilde{v}$ に対して占めている割合

$$1 - \frac{v(t)}{\tilde{v}}$$

をいう．計算結果を下記の要領で図 6 に示した:

- 太い実線: 全提案方法を用いた計算.
- 太い破線: 提案 1 を適用しなかった計算.
- 細い実線: 提案 2 を適用しなかった計算.
- 細い破線: 提案 3 を適用しなかった計算.
- ドット線: 未改良の BranchAndPrune による計算.

計算結果から，各提案方法が性能向上に寄与しており，全提案方法を用いた計算が最も高効率になっていることが確認できる．また全提案方法を用いて計算される内部近似の割合がほぼ一様に収束していることが確認できる．提案 2 による性能向上が他の提案方法に較べて少ないのは，重なった box 間のリンク管理に計算コストを要しているためだと考えられる．また 1000 秒に達していない計算はメモリ不足により途中終了している．

5 まとめ

制約不足の NCSP のための数値求解手法を提案した．提案手法は BranchAndPrune アルゴリズムを拡張したものであり，既存手法よりも広範な問題を扱うことができる．制御やロボティクス等の多くの問題を制約不足の NCSP として定式化し，提案する射影手法がそれらの問題の解析に役立つことが期待される．

謝辞 本研究の一部は科研費 23-3810 と ANR grant PSIROB06_174445 の補助を得て行った．

参 考 文 献

[1] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget, “Revising hull and box consistency,” Proc. of ICLP, pp.230–244, 1999.

[2] G. E. Collins, “Quantifier Elimination by Cylindrical Algebraic Decomposition – Twenty Years of Progress,” Quantifier Elimination and Cylindrical Algebraic Decomposition, pp.8–23, 1998.

[3] A. Goldsztejn and L. Jaulin, “Inner and outer approximations of existentially quantified equality constraints,” Proc. of CP’06, pp.198–212, *LNCS*4204, 2006.

[4] A. Goldsztejn and Jaulin, L., Inner approximation of the range of vector-valued functions. Reliable Computing 14, 1–23 (2010)

[5] E. Hansen and S. Sengupta, “Bounding solutions of systems of equations using interval analysis,” BIT, vol.21, pp.203–211, 1981.

[6] P. Herrero, M.A. Sainz, J. Veh, and L. Jaulin, “Quantified set inversion algorithm with applications to control,” Reliable Computing, vol.11, no.5, pp.369–382, 2005.

[7] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, Applied Interval Analysis, Springer-Verlag, 2001.

[8] A. Neumaier, Interval Methods for Systems of Equations, Cambridge University Press, 1990.

[9] S. Ratschan, “Uncertainty propagation in heterogeneous algebras for approximate quantified constraint solving,” Journal of Universal Computer Science, vol.6, no.9, pp.861–880, 2000.