2M－7

プレスブルガー文真偽判定手続きにおける多元連立1次合同式の求解処理の高速化

柴田 直樹　森岡 澄夫　東野 輝夫　谷口 健一

大阪大学 大学院基礎工学研究科 情報数理系専攻

1. はじめに

加算を持つ整数の理論 (整数の集合 Z 上の変数, 定数, $+, -, =, <, \wedge, \vee, \forall, \exists$からなる理論) はプレスブルガー (Presburger) 算術と呼ばれ, その上の閉論理式をプレスブルガー文 (P 文) と呼ぶ [1]. 全ての変数が同一の限定記号 ($\exists$または$\forall$) で束縛された冠頭形の P 文 (EPP 文) はプログラムや回路の正当性証明などに利用されている [2]. EPP 文の真偽判定アルゴリズム [3] では, 掃き出し法に類似した手法を用い, 多元連立1次合同式の解をバックトラック探索する処理を多用する. 本稿では, この処理を高速化する手法を考案し, いくつかの例題に対して評価を行った. 本手法は, 解の個数が少ないと予想される変数から解を探索することにより, 探索の総試行回数を削減するものである.

2. EPP 文の真偽判定アルゴリズム

本稿では, これまでに知られている最も速い直井らの EPP 文の真偽判定アルゴリズム [4] を用いる. このアルゴリズムでは, EPP 文の判定を以下の式の真偽判定に帰着する.

$$\begin{array}{l}\exists i_1 \exists i_2 \cdots \exists i_n \\ F(i_1, i_2, \cdots, i_n) \wedge \lambda_1 \mid a_1 \wedge \lambda_2 \mid a_2 \wedge \cdots \wedge \lambda_n \mid a_n\end{array} \quad (1)$$

$$\text{where } 1 \le i_1 \le d_1, \cdots, 1 \le i_n \le d_n$$

ここで, F は限定記号を含まない論理式, $\lambda_1, ..., \lambda_n$ は正の整数定数, $\mu_{11}, ..., \mu_{nn}, \nu_1, ..., \nu_n$ は整数定数, $a_j \equiv \mu_{j1} i_1 + \mu_{j2} i_2 + ... + \mu_{jn} i_n + \nu_j$ である. また, $d \mid A$ は項 A が整数 d で割り切れる関係を表す.

式 (1) の真偽判定は, 以下のようにして行う.

まず, 式 (1) の部分式

$$\lambda_1 \mid a_1 \wedge \lambda_2 \mid a_2 \wedge \cdots \wedge \lambda_n \mid a_n \quad (2)$$

$$\text{where } 1 \le i_1 \le d_1, \cdots, 1 \le i_n \le d_n$$

を以下の多元連立一次合同式

$$\left\{ \begin{array}{l|l} \lambda_1 & \mu_{11} i_1 + \mu_{12} i_2 + \cdots + \mu_{1n} i_n + \nu_1 \\ \lambda_2 & \mu_{21} i_1 + \mu_{22} i_2 + \cdots + \mu_{2n} i_n + \nu_2 \\ \vdots & \vdots \\ \lambda_n & \mu_{n1} i_1 + \mu_{n2} i_2 + \cdots + \mu_{nn} i_n + \nu_n \end{array} \right. \quad (3)$$

とみなし, $1 \le i_j \le d_j (1 \le j \le n)$ の範囲指定のもとでそれを満たす全ての $i_1 \cdots i_n$ の組を求める.

次に, 求めた各解を式 (1) の部分式 $F(i_1, i_2, \cdots, i_n)$ に代入して, その真偽を判定する. 代入した結果が一つでも真になれば式 (1) は真であり, 全て偽であれば式 (1) は偽である.

以下, 合同式 (3) の解を求める方法について, 簡単に説明する (詳細は [3] 参照).

式 (3) から以下に示すような各変数の係数を表す行列 (係数行列)

$$\begin{array}{c|c|c} & \begin{array}{cccc} i_1 & i_2 & \cdots & i_n \end{array} & \\ \begin{array}{c} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{array} & \left(\begin{array}{cccc} \mu_{11} & \mu_{12} & \cdots & \mu_{1n} \\ \mu_{21} & \mu_{22} & \cdots & \mu_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{n1} & \mu_{n2} & \cdots & \mu_{nn} \end{array} \right) & \begin{array}{c} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_n \end{array} \end{array}$$

$$\text{where } 1 \le i_1 \le d_1, \cdots, 1 \le i_n \le d_n$$

を作り, それをガウスの消去法と同様な手法を用いて三角化する.

$$\begin{array}{c|c|c} & \begin{array}{cccc} i_1 & i_2 & \cdots & i_n \end{array} & \\ \begin{array}{c} \lambda'_1 \\ \lambda'_2 \\ \vdots \\ \lambda'_n \end{array} & \left(\begin{array}{cccc} \mu'_{11} & \cdots & \cdots & \mu'_{1n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mu'_{nn} \end{array} \right) & \begin{array}{c} \nu'_1 \\ \nu'_2 \\ \vdots \\ \nu'_n \end{array} \end{array} \quad (4)$$

$$\text{where } 1 \le i_1 \le d_1, \cdots, 1 \le i_n \le d_n$$

次に, 後退代入の要領で各変数の解を $i_n, \cdots, i_1$ の順に探索していく. まず, 行列 (4) の第 n 行が表す, $\lambda'_n \mid \mu'_{nn} i_n + \nu_n$ という関係を満たす i_n の解のうち, $1 \le i_n \le d_n$ の範囲内にあるものを全て求める (変数を一つしか含まない一次合同式はそれ単独で解くことが出来る). ただし, 解が存在しないこともある. μ'_{nn} が 0 の場合は, $\lambda'_n \mid \nu_n$ であれば範囲内の全ての数は解であり, そうでなければ解は存在しない.

次に, 求まった各解 i_n を第 $n-1$ 行目の表す $\lambda'_{n-1} \mid \mu'_{n-1\ n} i_{n-1} + \mu'_{n-1\ n} i_n + \nu_n$ に代入し, 同様に i_{n-1} の解を探索する. 以上を繰り返して, 全ての変数 $i_1, \cdots, i_n$ に対する解を探索する.

A Technique for Reducing Computation Time to Solve Simultaneous Linear Congruences in a Decision Procedure for Presburger Sentences.

Naoki Shibata, Sumio Morioka,
Teruo Higashino and Kenichi Taniguchi
Division of Informatics and Mathematical Science
Graduate School of Engineering Science,
Osaka University
Toyonaka-shi, Osaka 560, Japan

3. 多元連立 1 次合同式の求解の高速化手法

2. の係数行列に対して，$i_n,\cdots,i_1$の解を順に探索していくとき，無駄が出ることがある．それは，途中のi_{k+1}まで解の探索を進めた結果，次の変数i_kに対する解が存在しないということが数多く起こるということである．その場合，i_nからi_{k+1}まで行った解の探索が無駄になってしまう．

判定高速化のためには，無駄になる探索をなるべく減らすことが望ましい．

そこで今回，探索の途中で解が存在しなくなる場合をなるべく減らすために次の方法を考えた．

高速化の方法

行列の三角化を行う前に，左から変数のとりうる範囲 (レンジ) が広い (d_1が大きい) 順に係数行列の各行を並べ変える． □

探索は行列の右側から行うので、上のようにすることによって、レンジの狭い変数から順に探索を行える。

レンジの広い変数は、それまでの探索で求まった解に対する次の変数の解が存在する可能性が高い．したがって，レンジの広い変数を後で探索することによって，探索回数が減ると期待できる．

高速化の例

$$\begin{array}{c|c|c} & \begin{matrix} i_1 & i_2 & i_3 \end{matrix} & \\ 9 & \left(\begin{matrix} 12 & 4 & 19 \\ 8 & 16 & 17 \\ 14 & 17 & 19 \end{matrix}\right) & \begin{matrix} 30 \\ 5 \\ 28 \end{matrix} \\ 29 & & \\ 22 & & \end{array}$$

where $1 \le i_1 \le 10,\ 1 \le i_2 \le 100, 1 \le i_3 \le 1000$

例として，上の係数行列について，提案手法を用いる場合と用いない場合で探索回数を比べてみる (ここでは探索回数は，それぞれの変数毎に求まった解の個数の和とする)．まず，上の係数行列を変数の順序を変えないで三角化すると，次のようになる．

$$\begin{array}{c|c|c} & \begin{matrix} i_1 & i_2 & i_3 \end{matrix} & \\ 5742 & \left(\begin{matrix} 6 & 5435 & 1325 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{matrix}\right) & \begin{matrix} -714 \\ 0 \\ 0 \end{matrix} \\ 6 & & \\ 1 & & \end{array}$$

この場合，まずi_3の解を探索すると全部で 1000 個求まる．それらの各解ごと探索すると，i_2の変数に対する解が延べ 16667 個，i_1の変数に対する解が延べ 179 個得られる．探索回数は 1000+16667+179=17846 回である．

次に，提案手法を使い，変数の順序を変えて三角化すると，次のようになる．

$$\begin{array}{c|c|c} & \begin{matrix} i_3 & i_2 & i_1 \end{matrix} & \\ 5742 & \left(\begin{matrix} 1 & 3991 & 5664 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}\right) & \begin{matrix} 3540 \\ 0 \\ 0 \end{matrix} \\ 1 & & \\ 1 & & \end{array}$$

i_1の変数に対する解は 10 個であり，i_2の変数に対する解が延べ 1000 個，i_3の変数に対する解が延べ 179 個得られる．解の探索回数は計 1189 回である．

この例では，提案手法により探索回数がおよそ 15 分の 1 ですむ．

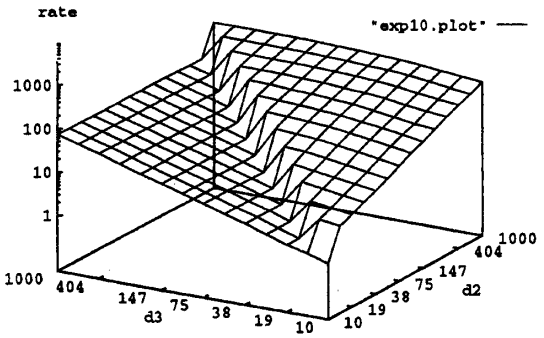
4. 評価実験

提案手法の効果を調べるため，係数行列

$$\begin{array}{c|c|c} & \begin{matrix} i_1 & i_2 & i_3 \end{matrix} & \\ 9 & \left(\begin{matrix} 12 & 4 & 19 \\ 8 & 16 & 17 \\ 14 & 17 & 19 \end{matrix}\right) & \begin{matrix} 30 \\ 5 \\ 28 \end{matrix} \\ 29 & & \\ 22 & & \end{array}$$

where $1 \le i_1 \le 10,\ 1 \le i_2 \le d_2, 1 \le i_n \le d_3$

に対し，i_1の上限を 10 に固定し，i_2とi_3のレンジを 10 から 1000 まで変化させ，提案手法を用いることで何倍高速になるかを測定した．その結果を次図に示す．



縦軸が高速化された割合である．三つの変数のレンジの差が大きくなるほど提案手法の高速化の割合が大きくなっている．

また，3. で述べた手法を EPP 文の判定ルーチンに組み込み，実行速度の比較を行った．

以下の EPP 文の真偽判定の過程で提案手法は一回使用され (提案手法を使うか使わないかという点以外においては真偽判定の過程は全く変わらない)，提案手法を使った場合が 0.1 秒 (Pentium 133MHz 使用)，使わない場合が 57 秒になり，500 倍程度高速化された．

$$\begin{array}{lrcll} \exists x \exists y \exists z[& 21x+13y+3z & < & 2031 & \wedge \\ & 32x-54y+21z & < & 2300 & \wedge \\ & -42x+21y+35z & < & 2200 & \wedge \\ & -11x-88y+41z & < & 2100 & \wedge \\ & 14x+22y-35z & < & 2200 & \wedge \\ & 25x-31y-12z & < & 2400 & \wedge \\ & -24x+55y-12z & < & 2200 & \wedge \\ & -35x-31y-33z & < & 2500 &] \end{array}$$

5. おわりに

本稿では，EPP 文の判定を高速化する手法を提案し，その評価実験を行った．その結果，多くの場合判定時間が短縮されることを確認した．

参考文献

[1] D.C.Cooper: "Theorem Proving in Arithmetic without Multiplication",Machine Intelligence, No.7,pp.91-99(1972).
[2] 東野輝夫, 北道淳司, 谷口健一:" 整数上の線形制約の処理と応用", コンピュータソフトウェア,Vol.9,No.6,pp.31-39(1992-11).
[3] 直井邦彰, 高橋直久: "プレスブルガー算術を用いた Infeasible Path 検出の高速化技法", 信学技報, SS95−19, pp.71−78(1995).