ICOT Technical Memorandom: TM-0981

TM-0981

制約グラフの構造分解および整合性解析による代数制約評価系の効率化についての検討

永井保夫（東芝）

Decembert, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

制約グラフの構造分解および整合性解析による代数制約評価系の効率化についての検討

Structural Analysis of Constraint Graph by Means of Sparse Orthogonal Factorization and Its Application for Efficient Algebraic Constraint Solver

永井 保夫

Yasuo NAGAI

(株) 東芝 情報通信システム技術研究所

Toshiba Corp.

(E-mail: nagai@aioh.icsl.toshiba.co.jp)

Abstract: This paper deals with the structural analysis of constraint graph by means of sparse orthogonal factorization and its application for an efficient constraint solver. We introduce a graph-theoretic approach to improve the efficiency of constraint processing by analyzing and using an algebraic structure of constraint representation. We describe constraints in terms of the graphical representation and represent constraint set as the structure of a matrix with a bipartite graph. First, we describe a structural decomposition method of constraint graph, DM decomposition, so that a block upper triangular matrix can be correctly computed by canonical reordering of a matrix which represents the constraint graph. Next, we consider to improve an efficiency of constraint solver, through the structural analysis of the constraint graph, which decomposes this graph into subgraphs and checks their structural solvability. In particular, we give a brief explanation of the improvement of an algebraic constraint solver, based on Buchberber algorithm, of the constraint logic programming language, CAL.

1 はじめに

制約論理型言語は，制約という概念を論理型言語に導入することで，従来の論理型言語と比較して，1) 対象となる構成要素やその属性間で成立する関係を関係表現や関数表現を用いて，より宣言的に記述でき，2) プログラムの実行制御に関する表現もより宣言的に記述できるという特徴をもつ [15,19,8,9]．制約論理型言語はこのような記述能力に優れており，宣言的プログラミングを可能にする．しかしながら，その処理系は必ずしも効率のよい処理を実現しているとはいえない [8,15,19]．

そこで制約論理型言語を用いた効率的な問題解決を実現するためには，制約論理型言語がもつ推論機構ならびに制約を取り扱う制約評価系の両者について考慮することが要求される．

本稿では後者の制約評価系が対象とする制約集合を制約グラフ [1,11,13,20] として表現し，グラフ論的手法を用いて制約評価系を効率化する手法について報告する．本手法は，制約集合がもつ代数的構造を抽出し，その情報に基づいて求められた制約間の依存関係情報から，制約評価系に対する制御情報を生成し，効率的な処理を実現するものである．

そのためには，構造情報をいかに有効に利用して，整合性を判定し，構造的に可解な部分問題として分解していくかが重要な問題になる．このような問題に対してはグラフ論的な手法を用いることが有効である [3,2,20]．

第2章ではグラフ論的な考え方の有用性をふまえて，制約グラフの代数構造を行列形式で求め，そのスパース構造を解析し、グラフにより表現された問題を構造情報に基づきブロック三角化行列となるように部分問題 (部分グラフ) に分割する手法について述べる．本手法では制約のもつ代数構造をグラフを用いて表現し，制約集合を2部グラフ $G=(V^+,V^-;E)$ であらわし，2部グラフの DM 分解 [6,2] を用いて構造分解および整合性解析をおこなう．

第3章では制約グラフの整合性解析によって，制約論理型言語の制約評価系の効率化処理を実現する手法について検討する．ここでは，数値処理における効率化手法であるスパース行列処理手法を用いて，制約論理型言語 CAL[9] における Buchberger アルゴリズム [14,16,17] に基づいた代数制約評価系を効率化する手法について述べる．Buchberger アルゴリズムは，1) 非線形制約に対する基底計算で求められる多項式の次数が，最悪の場合では多項式環における変数の個数の double exponential となり，2) 計算時間が制約の評価順序ならびに単項の優先順位に依存し，3) 解の係数成長が計算時間に影響を及ぼすという性質を有するため [14,16]，制約評価系における効率的な処理の実現が不可決であると考えられる．

第4章では線形制約および非線形制約を対象とした問題に対して，本効率化手法を適用し，処理時間の測定ならびに評価をおこない，その有効性を確認した．さらに，Buchberger アルゴリズムと関連付けながら，効率化の要因についても考察した．

2 制約グラフの構造分解および整合性解析

2.1 制約集合と制約グラフ

制約とは対象の構成要素およびその属性間で成立する関係を宣言的に記述したものである．制約は関係や関数によって表現される．関数制約では，陽関数による表現と陰関数

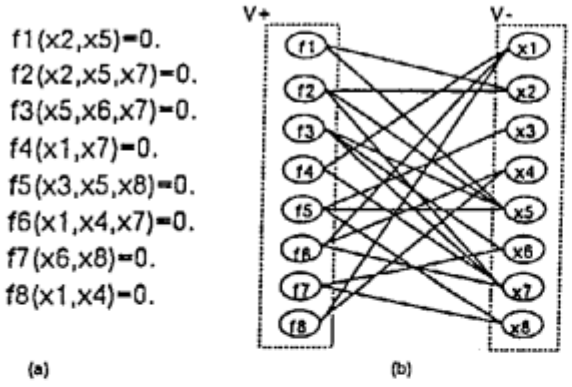

図1: 制約集合と2部グラフにより表現された制約グラフ

による表現が考えられる．陽関数により表現された関数制約は，有向グラフとみなされ，変数間の因果関係 (関数の依存関係) を表現したグラフとして取り扱われる．また，陰関数として表現された関数制約は無向グラフとみなされ，その多くは代数方程式の形式をとる．このような関数制約は集合を構成し，連立方程式系とみなされて，解かれることが多い．実際の問題を考えた場合，制約集合を統一的に扱うことが望ましい．

制約グラフはこのような関係や関数といった様々な形式をとる制約集合の代数構造をグラフ [5] により表現したものである [1,11,13].

本論文では制約集合のもつ代数的構造情報を抽出するために，制約集合を 2 部グラフ [5] を用いた制約グラフとして表現する．2 部グラフはグラフ $G=(V,E)$ の V_1, V_2 が共に空集合とならない $V=V_1 \cup V_2$, において，$V_1 \cap V_2 = \phi$ であり，かつ各 $e \in E(G)$ がノード V_1 とノード V_2 につながっていると定義される．たとえば，図 1-(a) の制約集合の 2 部グラフ表現 $G=(V^+,V^-;E)$ が，図 1-(b) に与えられたとき，制約 $f_2(x_2,x_5,x_7)$ は右端点集合 V^+ が制約 $\{f_2\}$ に，左端点集合 V^- が制約 $\{f_2\}$ の変数集合 $\{x_2,x_5,x_7\}$ にそれぞれ対応する．

2.2 制約グラフの構造分解および整合性解析

制約集合により定式化された問題に対処するためには，最初に与えられた問題を部分問題に分割し，各部分問題をどのような順序に従って解いていけば最終的に解が求まるかを決定する必要がある．そのためには，制約グラフの依存関係解析の処理において，問題を部分問題に分割し，各部分問題での処理に要する計算量の和をできるだけ少なくすることが望ましい．

さらに，大規模システムを解析するためには，その構造情報をいかにうまく利用して，制約 (方程式) の構造的な整合性を判定し，構造的に可解な部分問題に分解するかが重要な問題となる．このような問題に対しては，グラフ論的な手法を用いることが有用である．

以下では，まず 2 部グラフ G の DM (Dulmage-Mendelsohn) 分解 [6,2] について説明し，次にこれを用いた構造分解と整合性解析について述べる．

2.2.1 2 部グラフの DM 分解

2 部グラフ $G=(V^+,V^-;E)$ における DM 分解とは既約成分への分解であり，半順序構造 $\prec$ をもつ $V(=V^+ \cup V^-)$ の部分集合の族である $\{G_-,G_+\} \cup \{G_i\}_{i=1}^n$ に完全正準分解することである．前者の $\{G_-,G_+\}$ を不整合部，後者の $\{G_i\}_{i=1}^n$ を整合部とよぶ．

制約集合を 2 部グラフ $G=(V^+,V^-;E)$ として表現し，2 部グラフの DM 分解をおこなうことにより，制約集合を表現するグラフが構造的に可解であるかどうかを判定し，可解であれば部分問題 $G_i=(W_i^+,W_i^-;E_i)$ $(i=1,\ldots,n)$ に分割し，解を求めるための順序を決定できる．制約グラフが構造的に可解であるとは，グラフに対応する制約 (関係) 集合が各々の制約が関数や変数のとりうる値に無関係に，依存関係だけに基づいて一意的な可解性が存在することであり，2 部グラフ G 上で完全マッチング [5] が存在することと同値である．可解でない場合には，グラフの不整合な部分について検出することができる．

次に 2 部グラフの DM 分解の処理概要について説明する．具体的なアルゴリズムは，図 2 に示される．3 行目は，2 部グラフ $G=(V^+,V^-;E)$ の最大マッチング M を求める [7]．4 行目は，最大マッチング M の各エッジの逆向きエッジを G に追加して得られる補助グラフ $G_M=(V^+,V^-;\tilde{E})$ $(\tilde{E}=E \cup \{(u,v) \mid (v,u) \in M\})$ を求める．5 行目では，不整合部 $\{G_-,G_+\}$ に対する処理をおこなう．まず，$S^+=V^+-\partial^+M$，$S^-=V^--\partial^-M$ としたとき (ただし，∂^+M は M の左端点，∂^-M は M の右端点をあらわす)，補助グラフ G_M において，S^+ の点から有向道によって到達可能な点全体の集合を $U_{(-)}$ とおき，S^- の点へ有向道によって到達可能な点全体の集合を $U_{(+)}$ とおき，G の部分グラフを誘導する．次に，$E_i=\{e \mid e \in E, \partial^+e \in U_{(i)}, \partial^-e \in U_{(i)}\}$, $W_i^+=V^+ \cap U_{(i)}$, $W_i^-=V^- \cap U_{(i)}$, $(i=\{-,+\})$ として，$G_i=(W_i^+,W_i^-;E_i)$, $(i=\{-,+\})$ を定める．$G'=\{G_-,G_+\}$ とする．6 行目は，$G_M-(U_{(-)} \cup U_{(+)})$ を弱連結成分 $G_i=(W_i^+,W_i^-;E_i)$, $(i=1,2,\ldots,n)$ に分解する．$G''=\{G_i\}_{i=1}^n$ とする．7 行目は，得られた $G=\{G_i\}$ を半順序関係と矛盾しないように並べ換える．このような DM 分解の全体の処理時間は，高々 $O(|V|^{5/2})+O(MAX(|V|,|E|))$ であり，十分実用的であるといえる．

```
1  procedure DM_decomposition(G:input, G':output, G'':output)
2    begin
3      find_maximum_matching(G, M);
4      make_auxiliary_graph(G, M, G_M);
5      induce_auxiliary_graph_and_handle_two_tails(G_M, G');
6      find_strongly_connected_component(G', G'');
7      permute_subgraph_merge(G'');
8    end;
```

図 2: DM 分解のアルゴリズムの概要

2.2.2 制約グラフの構造分解と整合性解析

制約グラフの整合性解析は次のような 2 部グラフの DM 分解の定理から判定することができる．

[定理]

2 部グラフ $G=(V^+,V^-;E)$ の DM 分解 $G_i=(W_i^+,W_i^-;E_i)$ $(i=\{1,\ldots,n\} \cup \{-,+\})$ に対して (a) から (c) が成り立つ．

(a) $W_-^- \neq \phi$ ならば，$|W_-^+|<|W_-^-|$ である．(G_- の場合)

(b) $W_+^+ \neq \phi$ ならば，$|W_+^+|>|W_+^-|$ である．(G_+ の場合)

(c) $|W_i^+|=|W_i^-|$ であって，G_i $(i=1,2,\ldots,n)$ は完全マッチングをもつ．

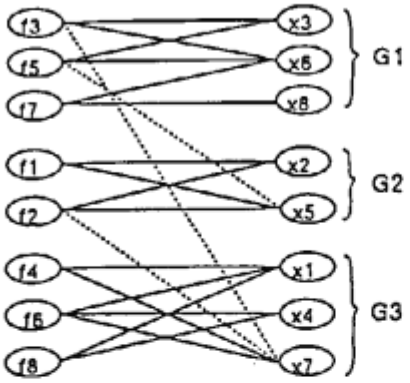

図3：図1-(b)をDM分解した結果

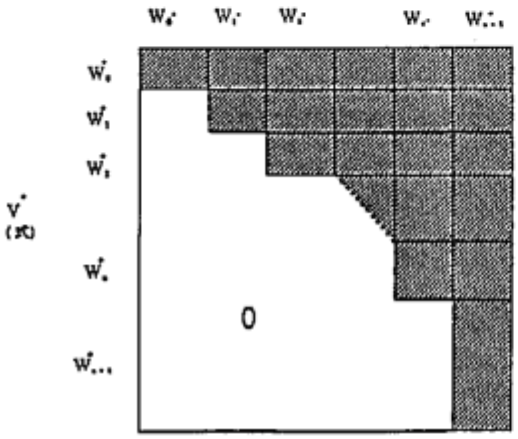

図4：DM分解により得られた半順序構造とそれに対応するブロック三角化行列

(a) は構造的に可解でなく，制約 (方程式) が不足した (under-constrained) 状態であり，W_-^+ の属する方程式 (制約) において未知数の数が方程式の数より多いため，解が一意に定まらない (不定である) ことを示している．

(b) も構造的に可解でなく，制約が矛盾または競合した (over-constrained) 状態であり，未知数の数が方程式の数より少ないため，解が求まらない (不能である) ことを示している．

(c) は分解された各 G_i $(i = 1, \ldots, n)$ が完全マッチングをもち，G も完全マッチングをもつため，構造的に可解であることを示している．

図1-(b)は図1-(a)の制約集合を示す制約グラフを2部グラフ形式で表現したものであり，図3は図1-(b)をDM分解し，整合性解析の結果，構造的に可解であることをあらわす．

この定理より制約グラフの整合性を判定することができる．また，$\mathcal{G} = \{G_i\}$ を半順序 $\prec$ に矛盾しないように並べ換え，$W_-^- = W_+^+ = \phi$ かつ $\mathcal{G}$ によって得られる有向グラフを図4に示すようなブロック三角行列の形 (スパース構造) に表現できれば，$(W_n^+, W_n^-), (W_{n-1}^+, W_{n-1}^-), \ldots, (W_2^+, W_2^-), (W_1^+, W_1^-)$ に対応するブロックごとに分割して解くことができる．

このような制約グラフの構造分解および整合性の判定は，与えられた問題を部分問題に分解し，効率的な問題解決を行う場合には，重要である．さらに，問題の定式化段階において，与えられた知識が不十分であるために，制約集合が不十分 (制約が不足している) であったり，冗長な (制約が矛盾または競合する) 状態になることがあり，このような状態の検出においても整合性判定が必要である [4].

3 制約グラフの構造情報を利用した制約評価系の効率化

制約論理型言語の制約評価系の効率化に対処するために，数値計算分野における大規模な問題に対する効率化手法であるスパース行列処理法 [3,10] の適用について検討する．

スパース行列処理法という一種の制約グラフの構造解析処理によって，いままで制約評価系が大域的に平板な制約集合として取り扱っていた制約集合を分割することができ，部分集合のみを考慮して制約評価をおこなうことができる．そのため，制約評価において，制約の全体集合を考慮する必要がなくなり，制約評価系の効率化とともに大規模な制約集合の取り扱いが可能となる．したがって，制約集合全体を対象とした制約評価と制約グラフの構造解析により分解された部分集合に対する制約評価を比較すると，後者のほうが問題解決機構の負荷をより軽減でき，効率的な処理が期待できる．

3.1 制約グラフの構造情報に基づいた依存関係と実行制御情報の生成

制約グラフの整合性解析によって，その構造情報に基づいた制約間の依存関係情報が生成される．

制約グラフの構造分解 (DM 分解) によってブロック三角行列 (スパース構造) が表現できれば，依存関係情報は各部分グラフ $G_i = (W_i^+, W_i^-; E_i)$ 間の半順序関係ならびに部分グラフ内の半順序関係に基づいて生成することができる．効率的に制約を評価するためには，この依存関係情報にしたがって問題を部分問題に分解し，解くべき制約の順序ならびにそれぞれの部分問題内での制約の評価順序を実行制御情報としてあらわすことが必要である．

このような点を考慮して制約評価系の実行制御情報を生成し，効率的な制約評価を実現することが要求される．

3.2 制約論理型言語における代数制約評価系の効率化検討

以下では制約グラフの構造情報を利用した制約論理型言語 [8,9,15] の代数制約評価系の効率化について説明する．

制約論理型言語は，論理型言語に対して制約という概念を明確に導入することで，より宣言的な記述能力を向上することに成功した．しかしながら，その半面，その処理効率が問題となっている．制約論理型言語処理系の基本機能は図5のように制約を集める処理 (推論エンジン) と集められた制約集合を解く処理 (制約評価系) からなると考えられる．処理効率を向上させるためには，両者の処理をどのように制御するかが非常に重要になる．たとえば，制約を集める処理では与えられた問題の定式化において制約をどのような順序で与えるかということが全体の処理効率に影響を与える．一方，制約を解くという処理は，集められた制約を制約評価系が解くので，その処理効率は制約評価系に依存したものとなる．このような処理を向上させるためには，プログラムの大域的な情報の解析を用いた制約論理型言語の処理系 (制約を集める処理と制約を解く処理) の効率化手法が必要である．今回は，制約論理型言語を用いた制

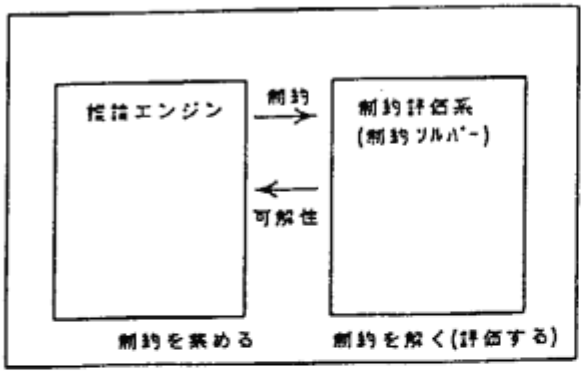

図5:制約論理型言語の構成

約問題解決機構の効率化という視点から，特に制約論理型言語において制約を解くという処理の効率化について制約論理型言語 CAL[9] を用いて検討をおこなう．

3.2.1 代数制約評価系の効率化検討

制約集合がもつ代数構造に関する情報を用いて，制約問題解決機構の一例である代数的制約評価系を効率化する方法について考察する．代数制約評価系には数値処理解法に基づいた評価系と記号 (数式) 処理解法に基づいた評価系が存在する．このような制約評価系の効率化をおこなうために，第2章で述べた制約グラフの構造分解によって求められたブロック三角化行列から制約 (間) の依存関係を生成し，依存関係情報から制約を解く順序を決定する方法を説明し，その適用結果について考察する．ここでは，代数制約を解く例として，図6の線形代数制約を対象とした例題1と図7の非線形代数制約を対象とした例題2を取り上げる．

```
:-   f1(X2,X5), f2(X2,X5,X7), f3(X3,X6,X7), f4(X1,X7),
     f5(X3,X5,X8), f6(X1,X4,X7), f7(X6,X8), f8(X1,X4).

f1(X2,X5) :- 2*X2 + X5 =2.
f2(X2,X5,X7) :- X2 + X5 +X7 =1.
f3(X3,X6,X7) :- X3 + X6 + 2*X7 =4.
f4(X1,X7) :- X1 + X7 =3.
f5(X3,X5,X8) :- X3 + 2*X5 + X8 =2.
f6(X1,X4,X7) :- X1 + X4 + 3*X7 =1.
f7(X6,X8) :- X6 + X8 =2.
f8(X1,X4) :- X1 + X4 =4.
```

図6: 線形代数制約を対象とした例題 (例題1)

```
:-   f1(X2,X5), f2(X2,X5,X7), f3(X3,X6,X7), f4(X1,X7),
     f5(X3,X5,X8), f6(X1,X4,X7), f7(X6,X8), f8(X1,X4).
```

$f1(X2, X5) : -2 * X2^2 + X5 = 2.$
$f2(X2, X5, X7) : -X2 + X5^2 + X7 = 1.$
$f3(X3, X6, X7) : -X3 + X6 + 2 * X7^3 = 4.$
$f4(X1, X7) : -X1 + X7 = 3.$
$f5(X3, X5, X8) : -X3^2 + 2 * X5 + X8 = 2.$
$f6(X1, X4, X7) : -X1^3 + X4 + 3 * X7 = 1.$
$f7(X6, X8) : -X6 + X8^3 = 2.$
$f8(X1, X4) : -X1 + X4 = 4.$

図7: 非線形制約を対象とした例題 (例題2)

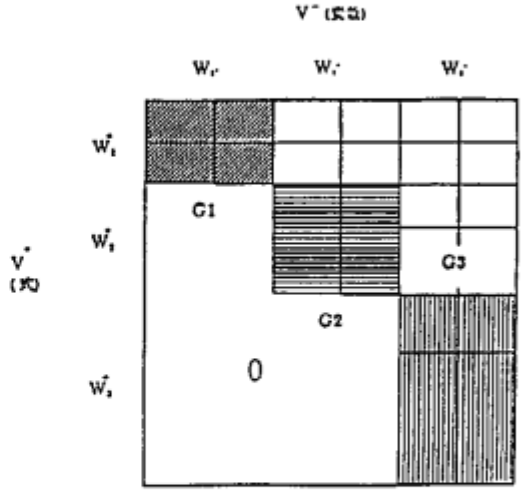

図8: DM分解により得られた半順序構造とそれに対応するブロック三角化行列

例題1では，まずゴール f1(X2,X5), f2(X2,X5,X7), f3(X3,X6,X7), f4(X1,X7), f5(X3,X5,X8), f6(X1,X4,X7), f7(X6,X8), f8(X1,X4) が実行され[1]，次にクローズ f1(X2,X5), ..., f8(X1,X4) の呼び出しにより制約が順次集められ，制約集合 $C = \{$ 2*X2 + X5 =2, X2 + X5 +X7 =1, ..., X1 + X4 =4 $\}$ が制約評価系により解かれる．制約論理型プログラミング言語 CAL の推論機構は SLD-導出に基づいており，詳細については 4.2.2 節に示される．図1-(b) は，この制約集合 C を2部グラフを用いて表現したものであり，図3は図1-(b) で表現された2部グラフを DM 分解した結果，G_1, G_2, G_3 という3つの部分グラフに構造分解されたことを示す．図8は図3のブロック三角化行列を表現し，3つの部分問題 G_1, G_2, G_3 に分割されたことを示す．このような制約集合の構造情報を制約グラフを用いて解析する手法は，数値処理の分野における連立一次方程式解法の効率化手法である LU 分解 [10] において上三角化行列を求めることに相当する．この結果から，G_3 に関する制約 (代数方程式) の集合，次に G_2 に関する制約の集合，最後に G_1 という順序で求解 (制約の評価) をおこなっていけば，その処理が簡単化され，効率的な問題解決がおこなわれる．実際には，図8で求められたブロック三角化行列から依存関係情報を抽出し，CAL の代数制約評価系に対して，制約の評価順序ならびに変数の優先順位を与え，処理を効率化する．

また，例題2についても同様な処理がおこなわれ，この場合は例題1と同様な結果となる．つまり，制約集合 C は図1-(b) の2部グラフ表現と同様であり，構造分解された結果は図3，ブロック三角化行列は図8と同様になる．

[1] CAL では，トップレベルからゴール列を与える場合には，変数名を小文字にする必要がある．

3.3 制約論理型言語 CAL における代数制約評価系の実行制御

3.3.1 グレブナ基底を求める代数制約評価系

制約論理型言語 CAL の代数制約評価系は，多項式イデアル計算手法として B. Buchberger によって提案された概念であるグレブナ基底計算アルゴリズム (Buchberger アルゴリズム) [14,16,17] に基づいており，グレブナ基底を解として求める．

このような手法によって計算される多項式イデアルのグレブナ基底は，イデアルのメンバシップ決定問題を解くために利用されることが多い．また，代数方程式系とみなされる多変数多項式の有限集合に対して厳密解の計算や多変数多項式の有限集合によって生成されるイデアルによる剰余環の計算にも用いられる．Buchberger アルゴリズムはこのような数式処理の分野においては有効な手法にひとつであるが，一方基底計算には非常に処理時間がかかることが多い．一般には，その計算量については以下の項目について知られている [16].

- 基底の計算には非常に時間がかかる (非線形の場合)
- 計算時間は，以下の 2 つの項目に依存する．
  - 制約の評価順序
  - 変数の優先順位
- 係数成長が計算時間に影響をおよぼす

3.3.2 CAL を用いたスパース行列処理手法の実現

本節では，CAL の制約評価系において数値処理における効率化手法であるスパース行列化手法をいかに実現するかついて論じる．これは制約評価系が取り扱う制約集合の代数構造を抽出し，抽出された情報に基づいて決定された変数の優先順位ならびに制約の評価順序を CAL の制約評価系に与えることにより実現される，以下では，その詳細について説明する．

CAL の代数制約評価系が導入した Buchberger アルゴリズムはグレブナ基底を求めるアルゴリズムであり，その詳細については第 4 章に示すが，その中心となる操作は等式 (制約) 集合の各等式を簡約化するための，項書き換え系の適用であるといえる．つまり，制約集合が与えられ，さらにその要素である各制約に対し単項間にある順序がつけられたとき，その順序のもとで最大の単項をそれ以外の多項式へ書き換える規則であるとみなし，書き換えをおこない基底を求める．

CAL では，変数に対して優先度を指定する述語が組み込みで提供されている．変数に対する優先度を指定することにより，単項間の順序を決定できるので，これに基づいて多項式に対する書き換え規則を設定できる．Buchberger アルゴリズムは合流性を保証した項書き換え系とみなせるので，CAL の制約評価系では，変数に対する優先順位の付与により書き換え規則の変更が可能であり，制約の評価順序を制御できる．

一方，線形連立方程式 $A \times \mathrm{x} = \mathrm{b}$ の数値解法における効率化手法として知られているスパース行列処理手法は，行列 A のスパース性 (行列が係数要素のうちで，非 0 要素と比較して 0 要素が著しく多いような構造を有しているという性質) に着目し，非 0 要素の値だけを変化させて方程式を解くことにより，無駄な計算を省略し，処理速度を向上させることを目的とした手法である．

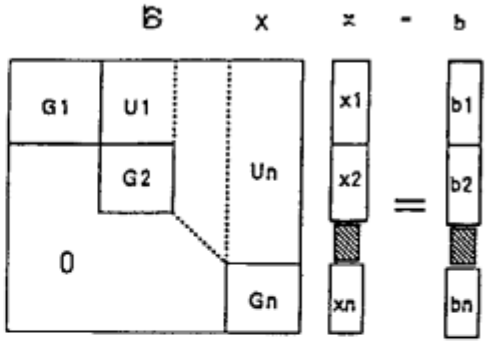

図 9：スパース行列処理手法

連立方程式の代表的な求解法であるガウスの消去法におけるスパース行列処理手法では，LU 分解が用いられる場合が多い．これに対して，本論文では DM 分解により求められた上三角化行列 (LU 分解における U に相当する) を用いたスパース行列処理の実現について説明する．

まず，スパース行列を構成する線形連立方程式 $A \times \mathrm{x} = \mathrm{b}$ に対して，図 9 に示されるような行列 A のブロック上三角化行列 B を求め，$b = (b_1, b_2, \ldots, b_n)^T$ とする．次に，Gn に関する部分問題について解を求める，すなわち，連立方程式 $G_n \times x_n = b_n$ を解く．さらに求められた x_n を用いて，G_{n-1} に関する部分問題を解く．つまり，$b \leftarrow b - U_n \times b_n$ とし，連立方程式 $G_{n-1} \times x_{n-1} = b_{n-1}$ を解くことにより x_{n-1} が求まる．このようにして順番に連立方程式 $G_i \times x_i = b_i\ (i = n-1, \ldots, 1)$ を解くことにより，$x_{n-1}, \ldots, x_2, x_1$ の値を求めることができる．

次に，このようなスパース行列効率化手法を CAL を用いて実現する方法について論じる．制約の評価順序指定および変数の優先順位指定によりスパース行列処理手法を実現し，代数制約評価系の効率的な処理手法について考える．ここでは，前述の線形連立方程式の数値解法におけるスパース行列処理手法と同様に DM 分解による方法に基づいた実現法について説明する．

数値解法の場合と同様に DM 分解によってブロック三角化をおこない，図 8 のようなブロック上三角化行列を求める．そして，部分問題 G_3 からはじめて，G_2, G_1 という順序に従って，部分問題ごとに制約評価ができるように ゴール列を並べ変え，グレブナ基底を求める．加えて，部分問題ごとに制約評価系によって基底を求めることを pre 述語を用いておこなう．具体的には変数間の優先順位を部分問題 G_n の変数集合 $\ll$ 部分問題 G_{n-1} の変数集合 $\ll \cdots$ 部分問題 G_1 の変数集合となるように pre 述語を指定する．

まず，部分問題 G_n に関する制約多項式のグレブナ基底 GB_n を求める．次に G_{n-1} に関する部分問題に関する制約集合 C_{n-1} を等式集合とみなし，今までに求められたグレブナ基底 (ここでは GB_n) を規則集合とみなす．これらの等式集合と規則集合から，等式の簡約化ならびに等式集合の更新をおこない，グレブナ基底 GB_{n-1} を求める．このような手順を部分問題 G_1 に関する制約多項式に関するグレブナ基底 GB_1 が求められるまで繰り返し，最終的には全体の制約集合 C に対する基底集合 GB が得られる．

3.3.3 例題に対するスパース行列処理手法の適用および CAL による実現について

非線形制約からなる例題 2 に対して，CAL の代数制約評価系の効率化手法を適用し，代数制約評価系に対して，制約の評価順序ならびに変数の優先順位を指定したプログラムを以下に示す．このプログラムではスパース行列処理手法を用いた制約評価系の効率的な処理がおこなわれる．

```
pre(x3, 100),
pre(x6, 100),
pre(x8, 100),
pre(x2, 99),
pre(x5, 99),
pre(x1, 98),
pre(x4, 98),
pre(x7, 98).
```

図 10: 述語 pre による変数の優先順序の指定

```
:- f4(X1, X7),
   f6(X1, X4, X7),
   f8(X1, X4),          } G3
   f1(X2, X5),
   f2(X2, X5, X7),      } G2
   f3(X3, X6, X7),
   f5(X3, X5, X8),
   f7(X6, X8).          } G1
```

$f1(X2, X5) : -2 * X2^2 + X5 = 2.$
$f2(X2, X5, X7) : -X2 + X5^2 + X7 = 1.$
$f3(X3, X6, X7) : -X3 + X6 + 2 * X7^3 = 4.$
$f4(X1, X7) : -X1 + X7 = 3.$
$f5(X3, X5, X8) : -X3^2 + 2 * X5 + X8 = 2.$
$f6(X1, X4, X7) : -X1^3 + X4 + 3 * X7 = 1.$
$f7(X6, X8) : -X6 + X8^3 = 2.$
$f8(X1, X4) : -X1 + X4 = 4.$

図 11: 制約の評価順序の指定

図 10 に示された pre 述語[2]によって与えられた変数間の優先順位は，[x3, x6, x8] ≫ [x2, x5] ≫ [x1, x4, x7] となる．また，ゴールは図 13 のように $G_3 \rightarrow G_2 \rightarrow G_1$ という順番を指定することにより，部分問題ごとに制約の評価順序が与えられる．以上のことから，まずゴール列 f4(X1,X7)，f6(X1,X4,X7)，f8(X1,X4) (部分問題 G_3 に対応) が実行され，次に f1(X2,X5)，f2(X2,X5,X7) (部分問題 G_2 に対応) が実行され，最後に f3(X3,X6,X7)，f5(X3,X5,X8)，f7(X6,X8) (部分問題 G_1 に対応) が実行される．対応するクローズが呼びだされることにより制約が順次集められ，制約評価系がインクリメンタルに制約を解く．まず，制約評価系はインクリメンタルに加えられた制約集合 $C_3 = \{X1 + X7 = 3, X1^3 + X4 + 3 * X7 = 1, X1 + X4 = 4\}$ (G_3 に対応) からグレブナ基底を計算し，基底集合 $GB_3 = \{X7 = 3 - X1, X4 = 4 - X1, X1^3 = -12 + 4 * X1\}$ を求める．

このようにして求められた基底集合に対して，制約集合 $C_2 = \{2 * X2^2 + X5 = 2, X2 + X5^2 + X7 = 1\}$ (G_2 に対応) 基底集合 $GB_2 = \{X2^2 = 1 - 1/2 * X5, X5^2 = 1 - X7 - X2\}$ を求める．

最後に，制約評価系は今までに求められた基底集合 $GB_3 \cup GB_2 = \{X7 = 3 - X1, X4 = 4 - X1, X1^3 = -12 + 4 * X1\} \cup \{X2^2 = 1 - 1/2 * X5, X5^2 = 1 - X7 - X2\}$ に対して，制約集合 $C_1 = \{X3 + X6 + 2 * X7^3 = 4, X3^2 + 2 * X5 + X8 = 2, X6 + X8^3 = 2\}$ (G_1 に対応) をインクリメンタルに加えることにより，グレブナ基底を計算し，基底集合 $GB_1 = \{X6 = 4 - 2 * X7^3 - X3, X3^2 = 2 - 2 * X5 - X8, X8^3 = -2 + 2 * X7^3 + X3\}$ を求める．最終的には図 12 のような基底集合 GB が得られる．

$x4 = 4 - x1.$
$x7 = 3 - x1.$
$x1^3 = -12 + 4 * x1.$
$x2^2 = 1 - 1/2 * x5.$
$x5^2 = -2 + x1 - x2.$
$x6 = -74 + 62 * x1 - 18 * x1^2 - x3.$
$x3^2 = 2 - 2 * x5 - x8.$
$x8^3 = 76 - 62 * x1 + 18 * x1^2 + x3.$

図 12: 求められたグレブナ基底 (例題 2)

4 実験結果と考察

本章では，例題 1 を含む線形制約を対象とした 7 つの問題および例題 2 を含む非線形制約を対象とした 9 つの問題に対して，本効率化手法の実験をおこない，それぞれの処理時間を求め，その有効性について考察する．

4.1 実験結果

グレブナ基底の計算は制約の評価順序および変数の優先順位に依存する．そこで，本実験では，制約評価系の内部については立ち入らないで，制約評価系を制御し，処理効率を向上させるために，図 10 のような変数の優先順位ならびに図 11 のような制約の評価順序の両者を指定する方法について実験をおこなった．その結果，本方法は表 1 のように有効性が示されたので，表 2 および表 3 の実験に対して適用された．

対象とした問題	適用前	適用後
問題 1 (例題 1: 線形，8 制約，8 変数)	66	51
問題 2 (線形，8 制約，8 変数)	766	688
問題 3 (例題 2: 非線形，最高次数 3，8 制約，8 変数)	610	68

表 1: 制約の評価順序および変数の優先順位指定による代数制約処理系の処理時間 (単位:msec)

表 2 は線形制約を対象とした問題について，提案した効率化手法の評価結果を示し，表 3 は非線形制約を対象とした問題に対する実験結果を示す．実験は逐次推論マシン PSI-II 上で，制約論理型言語 CAL バージョン 1.4 を用いておこなった．なお，制約グラフの構造分解および整合性解析は，SUN3 上の Sicstus PROLOG バージョン 0.6 を用いておこなった．

[2]CAL バージョン 1.4 では，変数の優先順位は 32768 以上の値の付与により指定される．

4.1.1 線形制約の実験結果

線形制約を対象とした場合は，本効率化手法により7パーセントから23パーセントまでの範囲で処理効率が改善された．なお，問題2は前節3.2.1の例題1の結果を示しており，問題3は制約の数と変数の数が問題2と等しく，定数係数が非常に複雑な分数形式をとる問題である．

対象とした問題	改善(適用)前	改善(適用)後
問題1 (6制約, 8変数)	54	43
問題2 (例題1, 8制約, 8変数)	66	51
問題3 (8制約, 8変数)	766	688
問題4 (17制約, 19変数)	215	185
問題5 (21制約, 21変数)	318	278
問題6 (18制約, 19変数)	244	227
問題7 (21制約, 21変数)	302	258

表2: 線形制約を対象とした問題の実験結果(単位:msec)

4.1.2 非線形制約の実験結果

非線形制約を対象とした場合には，その結果は与えられた問題に依存するが，おおよその場合については表3の適用後1に示されるように処理時間がおおきく改善されており，本効率化手法の有効性が示されたといえる．例えば，問題1は前節3.2.1の例題2の結果を示しており，約9倍の処理時間の改善がなされている．その場合の実行制御情報は図10の変数の優先順序ならびに図11の制約の評価順序に関する情報を用いた．但し，問題9はその例外であり，制約グラフの構造情報を解析した結果，部分グラフに分解できない，すなわちスパース構造ではなく，密構造なグラフの例である．このような場合には本手法によってわずかしか処理時間が改善されなかったことがわかる．

また，本手法における制約グラフの構造解析においては，変数の共有関係のみを考慮して，構造分解をおこない，最終的には制約の評価順序ならびに変数の優先順位を決定するが，変数の次数に関しても考慮して効率化することが望ましい．そこで，問題7および問題8ではこのような変数の次数に関する情報も考慮して効率化手法を適用した結果について示しており(表3の適用後2)，大幅に効率効率が改善されたことがわかる．たとえば，問題8(制約式の数21，変数の数21，最高次数7)では，本手法を用いないで制約評価をおこなうと約12時間かけても解が得られなかった(∞によりあらわす)が，本手法により数秒で解を求めることができた．

対象とした問題	適用前	適用後1	適用後2
問題1 (最高次数3, 8制約, 8変数)	610	68	-
問題2 (最高次数7, 17制約, 19変数)	618	264	-
問題3 (最高次数7, 21制約, 21変数)	8554	1778	-
問題4 (最高次数7, 21制約, 21変数)	470294	6914	-
問題5 (最高次数7, 21制約, 21変数)	1087822	288	-
問題6 (最高次数2, 12制約, 10変数)	281279	1322	-
問題7 (最高次数8, 6制約, 8変数)	3882	18581	57
問題8 (最高次数7, 21制約, 21変数)	∞	∞	2738
問題9 (最高次数2, 11制約, 11変数)	2633	2542	-

表3: 非線形制約を対象とした問題の実験結果(単位:msec)

4.2 Buchbergerアルゴリズムとその効率化に関する考察

4.2.1 Buchbergerアルゴリズム

以下では制約論理型言語CALの代数制約評価系に取り入れられている多項式イデアルのグレブナ基底を求めるBuchbergerアルゴリズムについて示す[14,16,17].

図13にBuchbergerアルゴリズムの主要な処理アルゴリズムを示す．入力である等式集合Eを多項式の有限集合$F = \{f_1, \ldots, f_r\}$，出力であるEをFによって生成されるイデアルのグレブナ基底$G(F)$とする．

まず，$E \leftarrow F, R \leftarrow 0$とする．$E$のすべての要素について，規則集合$R$を用いて簡約化をおこない，規則集合$R$を更新する(図15のReduceAll)．その結果から図16のNewBasisによって新しい基底を求める．Eのなかから互いに等しくない2個の要素からなるペア$\{f_1, f_2\}$をすべて求め，その集合をBとする．

次に，集合Bから任意のひとつの要素$\{f_1, f_2\}$を取りだし，S-多項式hを求め，Bから$\{f_1, f_2\}$を消去する．さらに，Eに関するhの正規表現h'を計算する．もし，$h' = 0$でなければ，新たなペア$\{\{g, h'\} \mid g \in G\}$を$B$に加え，さらに$E$に$h'$を加え，$E$のすべての要素について，規則集合$R$を用いて簡約化をおこない，基底を更新する．このような操作をBの要素がなくなるまで繰り返す．図17には正規表現を求める正規化アルゴリズムを，図18にはS-多項式を求めるアルゴリズムをそれぞれ示す．

```
1    E ← F;
2    R ← 0;
3    ReduceAll(E, R);
4    NewBasis(E, R);
5    B ← {{f1, f2} | f1, f2 ∈ E, f1 ≠ f2};
6      while B ≠ 0 do
7        choose a pair {f1, f2} ∈ B;
8        B ← B − {f1, f2};
9        h ← SPolynomial(f1, f2);
10       h' ← NormalForm(h, E ∪ R);
11       if h' ≠ 0 then
12         B ← B ∪ {{e, h'} | e ∈ E};
13         R ← R ∪ {h'};
14         ReduceAll(E, R);
15         NewBasis(E, R);
16       endif
17     endwhile
```

図13: Buchbergerアルゴリズムの処理概要

4.2.2 代数制約評価系の効率化に関する検討項目および考察

次に，上記のBuchbergerアルゴリズムに基づいた代数制約評価系の効率化について考察する．Buchbergerアルゴリズムの効率化については以下の検討項目について考察をおこなった．

[単項の順序付け]

- 変数の優先順位はアルゴリズムの処理時間に非常に影響を与える．

CALでは，変数に対して優先度を指定することにより，次の手順によって単項間の順序を決定する [17].

1. pre 述語によって与えられた優先度に基づいて，変数間の全順序関係 $\gg$ を求める.
2. 求められた変数間の全順序関係にしたがって，単項を変数の順序が大きいものから並べ，グループ化する.
3. グループ化された単項を比較し，最大の変数を含む単項がいくつか存在するならば，単項どうしを全次数式 (total-degree) 順序で比較する．もし，等しければそれをもとの単項から取り除いてできる単項に対して上記の処理を繰り返し，どちらか一方が空になったときは，空でないほうの単項を大とする.

Buchberger アルゴリズムはこのように求められた単項間の順序に基づき，与えられた制約集合の要素である等式 (制約) の最大単項をそれ以外の多項式へ書き換えていくことにより，グレブナ基底を求める．pre 述語を用いることにより，制約評価系の書き換え規則を指定し，制約の評価を制御できる.

制約グラフの構造情報を解析し，解析された情報に基づいて制約間の依存関係を求め，この依存関係情報からなるべく少ない書き換えにより既約 (正規) 形が計算されるように，等式の書き換え規則を決定する．図 13 に示される Buchberger アルゴリズムにおいては，3 行目の ReduceAll，4 行目の NewBasis，さらに 9 行目の SPolynomial や 10 行目の NormalForm に対してこのような効率化がおこなわれていると考えられる．その結果，制約評価系に対してスパース行列手法を適用し，効率的な制約の書き換えが実現できる.

[生成されるクリティカルペアの数と選択すべき順序]

- 生成されるクリティカルペア $\{f_1, f_2\}$ の数はアルゴリズムの処理時間に非常に影響を与える.
- 集合 B からクリティカルペア $\{f_1, f_2\}$ を選択する順序はアルゴリズムの性能に影響を与える.
- 求められる基底はクリティカルペアから S- 多項式を生成する順序に依存する.

図 13 の Buchberger アルゴリズムの 5 行目では，n 個の要素の等式集合 E のなかから互いに等しくない 2 個の要素からなるペア $\{f_1, f_2\}$ を要素とする集合 B を求めており，その要素数は ${}_nC_2 = n(n-1)/2$ である．6 行目から 16 行目では B の要素がなくなるまで，B からクリティカルペアを選択し，これより S- 多項式を求め，さらに項の書き換えによって S- 多項式の簡約化を繰り返す．この部分ではクリティカルペアの選択は非決定的であり，B の要素が急激に増加する可能性が考えられる．したがって，その部分を効率的に処理するためには，生成されるクリティカルペアの数をできるだけ少なくすることが必要である.

われわれは制約グラフの構造分解によって，与えられた問題を部分問題に分解し，後述する制約評価系のもつインクリメンタリティという性質を用いて，部分問題ごとにグレブナ基底を計算し，最終的に全体の基底を求める方法[3]を選択した．これにより，生成されるクリティカルペアの増加をなるべく抑制し，効率的な処理が可能になると考えられる.

たとえば，図 7 に示される例題 2 を考えてみると，制約の数が 8(n=8) であるため，生成されるクリティカルペアの数は ${}_8C_2 = 8(8-1)/2 = 28$ である．一方，提案した効率化手法では部分問題ごとに生成されるクリティカルペアの数は部分問題 G_3 では ${}_3C_2 = 3(3-1)/2 = 3$, G_2 では ${}_8C_2 = 2(2-1)/2 = 1$, G_1 では ${}_3C_2 = 3(3-1)/2 = 3$ となる．したがって，部分問題間のインタラクション (変数の共有) が少なければ，部分問題ごとに生成されるクリティカルペアの個数 (合計数 7) に対して，さらにインタラクションのある部分のクリティカルペアの個数を考慮すればよいことになり，前者の ${}_8C_2 = 8(8-1)/2 = 28$ の場合と比較して，その数を減少させることができる.

[制約評価系のインクリメンタリティ]

新しい多項式が基底に追加されるたびに，その多項式を用いて基底中の要素である多項式を簡約化できれば，基底の簡約化が可能である．これにより，基底計算時に生成される基底の数を減らせるので，アルゴリズムの処理効率を向上させることができる．CAL や CLP(R)[8] に代表される制約論理型プログラミング言語の大きな特徴として，制約評価系に対して与えられた制約が充足しているかどうかを調べるために，制約をインクリメンタルに評価し，冗長な計算を減らして，計算量をなるべく少なくする機能がある．これは制約論理型言語の推論機構が Prolog などの論理型言語の特徴である操作 (手続き) 的意味論に基づいた計算モデルである SLD- 導出 [18] に制約という概念を拡張したモデルに基づいているためである [8,9]．以下ではこのように拡張された SLD- 導出の定義を示す.

[定義] P をプログラム，R を計算規則，C を制約，G_i をゴール $\leftarrow A_1, \ldots, A_k, \ldots, A_m; C (1 \leq k \leq m)$ とする．D_{i+1} を P 内の確定節 $B \leftarrow E_1, \ldots, E_n; C_1 (n \geq 0)$ とし，D_{i+1} のヘッド B と G_i のサブゴール A_k は最汎単一化子 (mgu) θ_{i+1} をもつとき，計算規則 R と θ を用いて，G_i と D_{i+1} よりゴール G_{i+1} が次の条件で導かれる.

- A_k は計算規則 R により選択された原子式である.
- $C_{i+1} = C \cup C_1 \cup \{B = G_i\}$ が可解 (solvable) ならば，G_{i+1} は G_i と D_{i+1} から
  ゴール $\leftarrow (A_1, \ldots, A_{k-1}, D_1, \ldots, D_n, A_{k+1}, \ldots, A_m; C_{i+1})\theta_{i+1}$
  を導出する.

[定義] P をプログラム，G をゴール，R を計算規則とする．$P \cup \{G\}$ の R による SLD- 導出列とはゴールの列 $G_0 = G, G_1, \ldots, G_n$ と P のプログラム節の変種の列 $D_1, D_2, \ldots$ と最汎単一化子 $\theta_1, \theta_2, \ldots$ の列から構成されており，各 G_{i+1} は R より θ_{i+1} を使って G_i と C_{i+1} から導かれ，n を SLD- 導出列の長さという．SLD- 導出列が successful であるとは，その最後の要素が原子式でないゴールをとり，successful な導出における最後の要素である制約を解制約という.

[定義] P をプログラムとする．P の成功集合 S_p は次のように定義される.

$S_p = \{(p \leftarrow A; C) \mid$ ゴール $\leftarrow A; C$ が解制約 C をもつ successful な SLD- 導出列を有する $\}$

制約は図 14 のように制約論理型言語の推論機構すなわち，上述の拡張された SLD- 導出に従って評価される．CAL プログラムの実行では，制約が得られるたびに制約評価系が呼び出される．制約評価時に既に得られている制約と矛盾

[3] CAL の実際の制約評価系では，制約の順次追加により制約集合が与えられた場合，あらたな制約が追加されるたびに，最初から解くのではなく，いままでに求められた制約の正規形を修正して解を求めている．これを制約評価系のもつインクリメンタリティという性質ということにする.

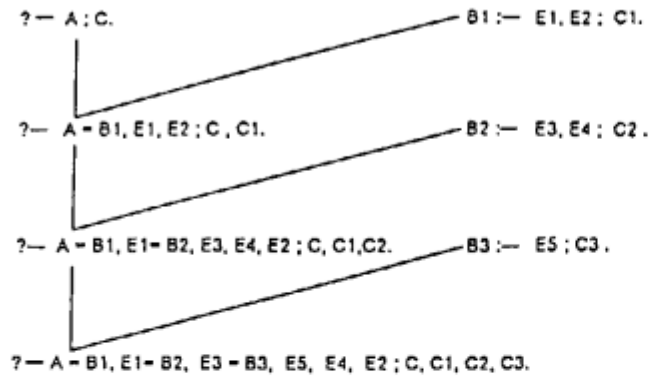

図14:拡張SLD導出に基づいた操作モデル

を生じるときにはバックトラックが起こり，あらたな制約が評価される．制約評価系を含む制約論理型言語を用いた処理を効率化するためには，このような制約のバックトラックをなるべく少なくし，冗長な処理を減らすことが必要とされる．そのためには，制約評価系において評価すべき制約集合のもつ代数構造の解析および構造情報の抽出によって，制約間の依存関係情報を生成し，これを用いて無駄な計算 (S-多項式の生成) を避け，解の係数成長をできるだけ抑制するように，バックトラックを制御することが考えられる．これは3.3節で述べた制約の評価順序を求め，代数制約評価系を効率化する手法で実現されている．

4.3 実験結果に対する考察

4.3.1 線形制約の実験結果に対する考察

線形制約を対象とした代数制約評価系は解制約であるグレブナ基底を一意に求めることができた．線形制約を対象にした問題では，Buchberger アルゴリズムではS-多項式は生成されない．これは，線形制約ではクリティカルペアの要素である最大単項が互いに素である (coprime) ため，S-多項式がゼロとなり，S-多項式の計算が不要になるからである．

本効率化手法では，制約の評価順序を指定することにより，無駄な計算をおこなわないで，処理を効率化することができた．特に，図13の Buchberger アルゴリズムでいうと1行目から4行目までの制約評価の部分がこれに相当し，その主要部である3行目の ReduceAll(E, R) と4行目の NewBasis (E, R) の処理が効率化されたと考えられる．しかしながら，その処理効率に非常に影響を与えるS-多項式に関する計算がないため，非線形制約の問題ほど急激には処理時間は改善されなかった．

4.3.2 非線形制約の実験結果に対する考察

非線形制約を対象とした代数制約評価系では，特に，クリティカルペア $\{f_1, f_2\}$ に対するS-多項式の計算が Buchberger アルゴリズムにおけるグレブナ基底の計算時間に大きな影響を与える．本手法では，ブロック三角化手法によって問題を部分問題に分解し，それぞれの部分問題に対して，順番にグレブナ基底を解として求めていく．このような三角化手法の適用によって，クリティカルペアに関する計算を減少させることができる．

その理由は，与えられた問題全体に対してクリティカルペアに関して計算するのではなく，分解された部分問題に対応する多項式の部分集合に対してのみ計算するためである．つまり，問題を部分問題に分割することによって，生成されるS-多項式を減少でき，依存関係情報を用いることにより集合 B から冗長なクリティカルペアをできるだけ取り除けるので，クリティカルペアに関する計算を減少できるためである．さらに，基底計算において部分問題に対して求められているグレブナ基底を規則集合 R とみなし，分割された部分問題間の相互依存関係がなるべく少なくできれば，クリティカルペアに関する計算，つまり生成されるS-多項式の数の減少ならびに解の係数成長の抑制により，効率的な処理が実現できると考えられる．

4.4 問題点と今後の課題

本手法では制約集合があらわす制約グラフのスパース性 (疎行列) という代数的特徴を用いて，処理を効率化するものであるが，次の項目については十分対処されておらず，今後の課題として検討していく予定である．

- 制約グラフの構造分解において，処理の効率化という点から制約評価系に適した部分問題の大きさ (細かさ) を明確化することが必要である．
- 部分問題間の依存関係 (インタラクション) の度合いに依存するものであり，なんらかの尺度が必要である．
- 変数の共有関係だけでなく，共有する変数の次数に関しても考慮する必要がある．
- 部分問題を評価する (解く) 順序は部分問題間の半順序関係を用いて決定している．しかしながら，この順序の決定方式では不十分であり，クリティカルペアの増加を抑え，グレブナ基底の係数をできるだけ成長させないような方式を考えることが必要である．
- 制約論理型言語の処理系全体の効率化に必要となるプログラムの大域的な情報の解析 [12] はおこなわれておらず，考慮する必要がある．

5 まとめ

制約グラフのスパース構造に基づいた整合性解析とその解析により求められた情報を用いた代数制約評価系の効率化について検討した．まず，制約グラフの構造分解および整合性解析について説明し，次にこれらの解析結果である制約集合のもつ代数的な構造情報を利用した制約問題解決機構の効率化について検討をおこなった．ここでは制約論理型言語 CAL を用いた効率的な問題解決を代数制約の求解問題を対象として検討をおこなった．その結果，数値処理の効率化に有効なスパース行列処理の概念が制約問題解決の効率化，特に非線形代数制約を対象とした制約評価系の効率化においても有力な手法であることがわかった．今後はこのような効率化手法を制約評価系だけでなく，制約論理型言語処理系全体 (制約の収集および収集された制約の求解) の効率化を目的として，プログラミングにおいて述語に対する入出力 (モード) 情報と述語，関数，制約などからデータ依存関係を解析し，全体の処理の実行順序を推論し，より実行効率のよいソースレベルコードを生成するコンパイラとして利用することを考えている．

謝辞

本研究は第5世代コンピュータプロジェクトの一環として行なわれた．本研究の機会を与えてくださり，常にご指導いただいたICOTの淵一博所長，古川康一研究次長，生駒憲治部長代理，新田克己第7研究室室長ならびに，有益なコメントをいただいた長谷川隆三部長代理ならびに相場亮第4研究室長代理に深く感謝いたします．また，CALを利用するにあたり，いろいろ教えて頂いた第4研究室の皆様に感謝いたします．

参考文献

[1] A. Dechter and R. Dechter, Removing Redundancies in Constraint Networks, Procs. of AAAI 87, (1987)
[2] 伊理正夫他，講座・数理計画法 7, グラフ・ネットワーク・マトロイド，産業図書，(1986)
[3] 可児賢二，大附辰夫，設計自動化におけるグラフ理論と組み合わせ算法(2)，情報処理，Vol. 16, No. 6, (1975)
[4] Y. Nagai and K. Ikoma, Design Plan Generation Through Constraint Compilation, ICOT Technical Memorandumn, ICOT, (1989)
[5] R. J. Wilson, Introduction to Graph Theory, 3rd ed., Longman Group Limited, (邦訳: 斎藤伸自他，グラフ理論入門)，(1985)
[6] A. L. Dulmage and N. S. Mendelsohn, Two algorithms for bipartite graphs, Journal of SIAM, Vol. 11, No. 1, March, (1963)
[7] J. E. Hopcroft and R. M. Karp, An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs, SIAM Journal on Computer, Vol.2, No.4, (1973)
[8] J. Jaffar and J.-L. Lassez, Constraint Logic Programming, Procs. of POPL 87, (1987)
[9] K. Sakai and A. Aiba, CAL: A Theoretical Background of Constraint Logic Programming and its Applications, J. of Symbolic Computation 8, (1989)
[10] 淵田孝夫，岩波講座 ソフトウエア科学 9，数値処理プログラミング，岩波書店，(1988)
[11] R. Dechter and J. Pearl, Tree Clustering for Constraint Networks, Artificial Intelligence 38, (1989)
[12] S. K. Debray, Static Inference of Modes and Data Dependencies in Logic Programs, ACM trans. on Programming Languages and Systems, Vol. 11, No. 3, July, (1989)
[13] R. Dechter, Enhancement Schemes for Constraint Processing: Backjumping, Learning, and Cutset Decomposition, Artificial Intelligence 41, (1990)
[14] B. Buchberger, Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory, Publications/Reports, RISC-Linz Series no. 83-29.0, (1983)
[15] J. Cohen, Constraint Logic Programming Languages, Comm. of the ACM, Vol.33, No.7, July, (1990)
[16] C. M. Hoffmann, Gröbner Bases Techniques, Chapter 7 in Geometric & Solid Modelling - An Introduction, Morgan Kaufmann Publishers, Inc., (1989)
[17] 佐藤洋祐，制約プログラミング－処理系の具体例 (Gröbner Base)－，日本ソフトウエア科学会サマーチュートリアル，(1989)
[18] J. W. Lloyd, Foundations of Logic Programming, Springer-Verlag, (邦訳: 佐藤雅彦他，論理プログラミングの基礎)，(1984)
[19] 淵一博監修，渋口文雄他編，制約論理プログラミング，知識情報処理シリーズ 別巻2，共立出版，(1989)
[20] 永井保夫，生駒憲治，制約グラフのスパース構造に基づいた整合性解析と制約処理の効率化についての検討，人工知能学会第4回全国大会 1-14, (1990)

付録

[すべての等式 (多項式) の簡約化]

等式集合 E のすべての要素に対して，規則集合 R を用いて簡約化をおこない，既約な等式をあらたな規則集合 R に追加する．

```
ReduceAll(E, R);
1     while E ≠ 0 do
2       choose an element h ∈ E;
3       E ← E − {h};
4       h ← NormalForm(h, E ∪ R);
5         if h ≠ 0 then
6           R ← R ∪ {h};
7         endif
8     endwhile
```

図 15: 等式の簡約化アルゴリズム

[基底の更新]

入力である等式集合 E と ReduceAll において求められた規則集合 R を用いて，新しい基底集合を求める．

```
NewBasis(E, R);
1     E ← E ∪ R;
2     H ← E;
3     K ← 0;
4       while H ≠ 0 do
5         choose an element h ∈ H;
6         H ← H − {h};
7         k ← NormalForm(h, E − {h});
8         K ← K ∪ {k};
9       endwhile
10    E ← K;
```

図 16: 基底の更新アルゴリズム

[正規化]

正規化アルゴリズムは多項式 g および多項式集合 $F = \{f_1, \ldots, f_n\}$ が与えられたとき，F の要素である多項式を用いて g を書き換え，これが既約になるまで簡約化を繰り返し，正規化表現を求める．

```
NormalForm(g, F);
1     g0 ← g;
2     i ← 0;
3     while exist f ∈ F such that 最大単項 (leading power product) of f
      divides 項 (power product) p in gi
4         gi+1 ← gi − b · u · f;
5         i ← i + 1;
          where b は多項式 p を多項式 f の最大単項の係数で除算した商
                u= p/ 多項式 f の最大単項
6     endwhile
```

図 17: 正規化アルゴリズム

[S-多項式の定義]

正規化アルゴリズムでは，多項式集合 $F = \{f_1, f_2\}$ の要素である多項式の全次数が g のそれより大きい場合には書き換えおよび簡約化ができない．その解決策として，多項式集合の生成元を追加し，正規化アルゴリズムを適用することが考えられた．この生成元を多項式 f_1 と f_2 の S-多項式と呼び，定義は以下に与えられる．

```
SPolynomial(f1, f2)
1     u1 · f1 − (c1/c2) · u2 · f2;
      where ci = 多項式 fi の最大単項の係数,
            ui is such that si · ui = s1 と s2 の最小公倍数
              and
            si = 多項式 fi の最大単項 (i= 1,2)
```

図 18: S-多項式の定義