ICOT Technical Memorandum: TM-0928

TM-0928

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

永井保夫(東芝)、生駒憲治

July, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

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

Structural Analysis of Constraint Graph by Means of Sparse Orthogonal Factorization and Its Application to Efficient Constraint Processing

永井 保夫
Yasuo NAGAI
(株) 東芝 情報通信システム技術研究所
TOSHIBA Corp.

生駒 憲治
Kenji IKOMA
(財) 新世代コンピュータ技術開発機構
ICOT

This paper deals with the structural analysis of constraint graph by means of sparse orthogonal factorization and its application to an efficient constraint processing. We introduce a graph-theoretic approach to improve the efficiency of constraint processing by manipulating network-based knowledge representation. We describe constraints in terms of the graphical representation and represent constraint network 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 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 processing, 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 and constraint satisfaction, for the purpose of realizing the efficient problem solvers based on constraint logic programming language.

1 はじめに

制約とは適用対象の構成要素やその属性間で成立する関係を宣言的に記述し，関係や関数により表現したものであり，その代表的な問題解決機構には連続領域を対象とした代数的制約評価系ならびに離散領域を対象とした制約充足処理系がある [4]．このような制約をネットワークによって表現したものを制約ネットワークとよび，その構造を抽象化という観点からグラフ形式によって統一的に表現したものを制約グラフとよぶ．

制約ネットワーク（グラフ）の問題解決を効率的におこなうためには，その構造情報をいかに有効に利用して，制約表現の構造的な整合性を判定し，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．

そこで，まずグラフ論的な考え方の有用性をふまえて，制約ネットワークのグラフ表現を行列形式で求め，そのスパース構造を解析し，問題を構造情報に基づきブロック三角化行列形式となるように分割する手法について述べる．本手法では制約をグラフ表現し，制約ネットワークを2部グラフ $G=(V^+,V^-;E)$ とみなし，DM 分解 [6,2] を用いて構造分解し，整合性の解析をおこなう．次に，制約グラフの整合性解析によって，効率的な制約問題解決を実現する方法について検討する．ここでは，制約論理型言語を用いた制約問題解決機構の効率化の検討を中心としておこない，例として代数制約評価系および制約充足処理系を取り上げて説明する．

2 制約ネットワークの構造分解および整合性解析

2.1 制約グラフと制約ネットワーク

制約とは適用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．関係によって表現された制約，つまり N 個の変数 $X_1,X_2,\ldots,X_n$ に対する関係制約 $c(X_1,X_2,\ldots,X_n)$ は N 個のドメイン集合 $D_1,D_2,\ldots,D_n$ に対して成立する関係（つまり N 項関係）をあらわし，$D_1\times D_2\times\ldots\times D_n$ の部分集合である．一方，関数制約では，陽関数による表現と陰関数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，有効グラフとみなされ，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，その多くは方程式などの形式をとる．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，このような関係制約や陽関数および陰関数により表現された関数制約を統一的に扱うことが必要である．そこで，抽象化という観点から制約の構造をグラフ [5] により表現することとし，これを制約グラフとよぶ．制約グラフは完全グラフ，ハイパーグラフ，2部グラフなどを用いる場合が多い．

制約ネットワークは，N 個の変数 $X_1,X_2,\ldots,X_n$ および各変数に対するドメイン集合 $D_1,D_2,\ldots,D_n$ からなる関係（N 項関係）集合ならびに陽関数と陰関数からなる制約集合をネットワークとして表現したものであり，このように様々な表現形式をとる制約を満足する解を求める問題の記述に適した知識表現の一種であるとみることができる．

2.2 制約ネットワークの構造分解および整合性解析

組み合せ（最適化）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よって解いていけば最終的に要求される解が求まるかを決定する必要がある．このためには，制約ネットワーク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，ネットワークで表現された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，制約式（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 $\{G_0,G_{n+1}\}\cup\{G_i\}_{i=1}^n$ に完全正準分解することである．前者の $\{G_0,G_{n+1}\}$ を不整合部，後者の $\{G_i\}_{i=1}^n$ を整合部とよぶ．

制約ネットワークを2部グラフ $G=(V^+,V^-;E)$ として表現し，このような DM 分解をおこなうことにより，ネットワーク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 $G_i=(U_i^+,U_i^-;E_i)$ $(i=0,1,\ldots,n+1)$ に分割し，解を求めるための順序を決定できる．制約ネットワークが構造的に可解であるとは，ネットワークに対応する制約（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

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

図 1: DM 分解の概要

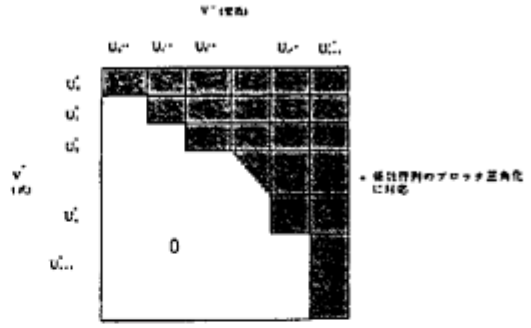

図 2: DM 分解により求められたブロック三角化行列

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.

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 1 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 [8]. 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 $G_M = (V^+, V^-; \bar{E})$ $(\bar{E} = E \cup \{(u,v) \mid (v,u) \in M\})$ を求める. 5 行目では, $S^+ = V^+ - \partial^+ M$, $S^- = V^- - \partial^- M$ としたとき, 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. 6 行目は, $G_M - (U_{(-)} \cup U_{(+)})$ を独連結成分 $G'' = \{G''_i \mid i = 1, 2, \ldots, n\}$ に分解する. 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 $O(\mid V \mid^{5/2}) + O(MAX(\mid V \mid, \mid E \mid))$ であり, 十分実用的であるといえる.

2.2.2 制約ネットワークの構造分解と整合性解析

制約ネットワーク (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 $G_i = (U_i^+, U_i^-; E_i)$ $(i = 0, 1, \ldots, n+1)$ に対して (a) から (c) が成り立つ.

(a) $U_0^- \neq \phi$ ならば, $\mid U_0^+ \mid < \mid U_0^- \mid$ である.

(b) $U_{n+1}^+ \neq \phi$ ならば, $\mid U_{n+1}^+ \mid > \mid U_{n+1}^- \mid$ である.

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

(a) は構造的に可解でなく, 制約 (方程式) が不足している (under-constrained) 状態であり, U_0^+ 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ている (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 $(i = 0, \ldots, n+1)$ 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.

この定理より制約ネットワーク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, $U_0^- = U_{n+1}^+ = \phi$ かつ $\mathcal{G}$ によって得られる有向グラフを図 2 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, $(U_n^+, U_n^-), (U_{n-1}^+, U_{n-1}^-), \ldots, (U_2^+, U_2^-), (U_1^+, U_1^-)$ に対応するブロックごとに分割して解くことができる.

特に連続量を対象としたシステムの解析によって, 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, このような制約ネットワークの構造分解および整合性の判定は, 非常に重要である. 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, 生成された制約ネットワークが不十分 (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.

3 制約ネットワークの構造情報を利用した制約問題解決機構の効率化

問題解決機構の効率化に対処するために, 連立方程式系の求解を制約問題解決のひとつと捉え, 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 [3,11] の適用について検討する.

このようなスパース行列処理法という一種の制約ネットワークの構造解析処理によって, 従来制約問題解決機構が取り扱うべき平坦な制約集合が分割され, 分割された部分集合のみを考慮して制約問題解決をおこなうことができる. そのため, 問題解決においても制約集合全体を大域的に考慮する必要がなくなり, 問題解決機構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. また, 制約集合全体を対象とした問題解決と制約ネットワークの構造解析により分解された部分集合に対する問題解決を比較すると, 後者の方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 制約ネットワークの構造情報に基づいた依存関係と実行制御情報の生成

制約ネットワーク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グラフが生成される.

依存関係グラフは, 制約ネットワークの構造分解 (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, 各部分グラフ $G_i = (U_i^+, U_i^-; E_i)$ 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される. 効率的な問題解決をおこなうためには, この依存関係グラフに従って, 問題を部分問題に分割し解いていく順序ならびにそれぞれの部分問題内での問題解決の順序を実行制御情報としてあらわすことが必要である.

依存関係グラフに対して方向を与えることにより求められる実行制御情報が有向非循環 (閉路を含まなければ) であれば, ある変数の値から他の全ての変数の値を局所的な制約伝播 (変数代入) によって求めることができる. 一方, 閉路を含んでいれば, 前者の場合の問題解決に加えて, 依存関係グラフにおいて閉路を構成している部分に対応する方程式を解く必要があり, ガウスの消去法や反復法などのように大域的な制約 (方程式) の取り扱いが必要となる. このような点を考慮して問題解決についての実行制御情報を生成し, 制約問題解決機構の効率化を実現することが要求される.

3.2 制約論理型言語を用いた制約問題解決機構の効率化検討

制約ネットワークの構造情報を利用した制約問題解決機構の効率化を検討しているが, ここでは制約論理型言語 [9,10] を用いた問題解決機構の効率化について, 代数制約評価系および制約充足処理 (ブール代数制約評価系) を取り上げて説明する.

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制約を集める処理と集められた制約の集合を解く処理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，論理プログラミングにおいて述語に対する入出力(モード)情報と述語や関数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研究[13]が有効である．将来的には，このような大域的な情報の解析を用いて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)の効率化をはかることを考えている．今回は，制約論理型言語を用いた制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 [10] 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

まず，ネットワークの構造情報を用いて，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くものと記号(数式)処理解法に基づくものが存在する．このような制約評価系の効率化をおこなうために，2で述べた制約ネットワークの構造分解によって求められたスパース構造から制約(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，線形代数制約を対象とした例題1と非線形代数制約を対象とした例題2を取り上げる．その場合の代数制約評価系は CAL の数式処理解法(Buchberger アルゴリズム)に基づいた制約評価系を用いており，得られる結果は Gröbner 基底[15]である．

(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 + 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.
```

(例題2)

```
:-   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.
```

例題1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，次にクローズ f(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 $C1 = \{2*X2 + X5 =2, X2 + X5 +X7 =1, \ldots, X1 + X4 =4\}$ が制約評価系により解かれるプログラムをあらわす．図3は，この制約集合 $C1$ を2部グラフを用いて表現したものであり，図4は図3で表現された2部グラフをDM分解した結果，$G1, G2, G3$ という3つの部分グラフに構造分解されたことを示している．図5は図4のブロック三角化行列を表現したものであり，3つの部分問題 $G1, G2, G3$ に分割されたことを示している．このような制約ネットワークの構造情報を解析する手法は，数値処理の分野における連立一次方程式

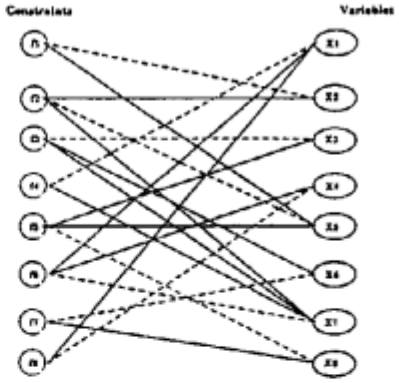

図3: 例題1および2の2部グラフ表現(点線は最大マッチングをあらわす)

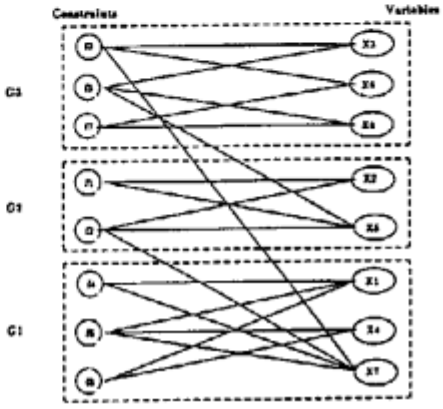

図4: 図3で表現された2部グラフをDM分解した結果($G1, G2, G3$ は分解された部分グラフをあらわす)

解法のひとつであるガウス消去法の LU 分解 [11] 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，$G1$ に関する制約式(方程式)の集合，次に $G2$ に関する制約式に関する制約式の集合，最後に $G3$ 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5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，解法の順序を解くべき変数の優先順位という形で与え，効率化をはかっている．

また，例題2のような非線形制約を対象とした制約評価系，すなわち非線形連立方程式解法の効率化においても，制約ネットワークの構造分解によって制約(方程式)と変数間の依存関係をブロック三角化行列形式で求め，この情報を実行制御情報として利用できる．なお，例題2の解析結果は例題1と同様なものとなる．

本効率化手法を用いた結果，例題1の線形制約を対象とした場合には本手法を用いない場合とさほど処理時間に差がなかったが，例題2の非線形制約を対象とした場合には本手法を用いない場合の約1/2の処理時間で解が得られた．さらに，線形制約ならびに非線形制約の両者において制約の数が増化すると，本効率化手法が有効性であることがわかった．たとえば，非線形制約をを対象とした例(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数時間かけても解が得られなかったものが，本手法を用いることにより数秒で解を求めることができた．つまり，従来線形制約評価系の効率化において有効であった大規模スパース行列のブロック三角化法が，非線形制約評価系の効率化に対しても適用できることがわかった．

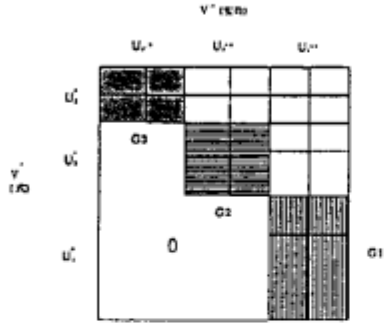

図 5: 例題 1 の制約集合のブロック三角化行列

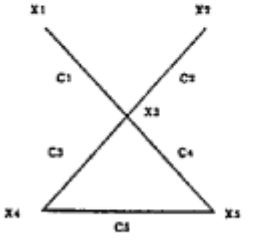

図 6: 制約ネットワーク

3.2.2 制約充足問題向け問題解決機構の効率化検討

制約充足問題に対処するためには，代数的な解法を用いる方法と探索法や近似的解法を用いる方法の 2 つが考えられる [7]. 制約ネットワークの構造分解および整合性解析を用いた制約問題解決機構の効率化手法は，両者の方法に対しても適用可能である. 前者の代数的な解法を用いる方法は制約充足問題をブール代数により定式化をおこない，ブール代数を対象とした代数制約評価系の効率化をはかるものであり，3.2.1 に述べた代数制約評価系の効率化手法をそのまま適用できる.

一方，後者の探索法を用いる方法では，特に二項制約ネットワークにおける充足処理 (制約充足問題) に対して制約の冗長性という概念に注目し，冗長性をいかに抽出除去してバックトラック探索の効率化をはかるかということに焦点があてられている. Dechter らは，バックトラック探索の効率化をはかるために，制約ネットワークにおける冗長性を除去し，木構造化されたグラフを探索する方法について提案した [1]. この方法は，制約ネットワーク (グラフ) のスパース構造に依存したアプローチである. さらに，制約グラフを三角化し，これを木構造化されたグラフに変換し，処理の効率化をはかる木クラスタリング法を提案した [12]. 以下では，制約ネットワークの構造分解および整合性解析を用いた効率化手法を図 6 に示されるような 2 項制約からなる制約ネットワークにより表現される制約充足問題に適用し，検討をおこなう. この問題では，各エッジ (X1,X3), (X2,X3), (X3,X4), (X3,X5), (X4,X5) が 2 項制約 $C1, C2, C3, C4, C5$ にそれぞれ対応する. 図 7 はこれらの制約を 2 部グラフ表現し，DM 分解により部分グラフ $G1, G2, G3$ に分解した結果を示している. この結果から分割されたそれぞれの部分グラフ (問題) において冗長性を解析し，依存関係を生成する. 依存関係情報を用いたバックトラック処理では制約ネットワークの無矛盾性を検査するために必要となる計算が省略されるので，処理の効率化をはかることが可能となる. たとえば，依存関係に基づいたバックトラック処理であるバックジャンピング法やトリー変換により制約グラフの探索をおこなうサイクルカットセット法による効率化に対して有効である [14]. 従って，ネットワークの構造解析による効率化手法は，代数制約を対象とした制約評価系だけでなく，このような離散領域を対象にした制約充足処理にも十分対処できると考えられる.

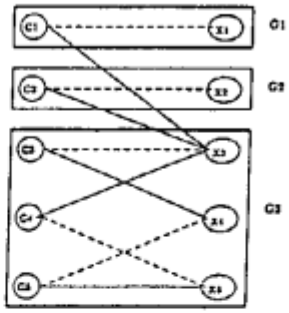

図 7: 図 6 の制約ネットワーク (グラフ) の 2 部グラフ表現を DM 分解した結果 (点線は最大マッチングをあらわす)

4 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制約処理の効率化について検討した. まず，制約ネットワーク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ネットワークの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および制約充足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た前処理 (一種のコンパイラ) 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 の渕一博所長，古川康一研究次長，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前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旧総務第 4 研究室室長代理をはじめとした第 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: 西関隆夫他, 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] 水井保夫, ブール代数を用いた制約充足問題の定式化とその解法について, 情報処理学会第 41 回全国大会, (1990)

[8] 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)

[9] J. Jaffar and J.-L. Lassez, Constraint Logic Programming, Procs. of POPL 87, (1987)

[10] A. Aiba, K. Sakai, Y. Sato, D.J. Hawley and R. Hasegawa, Constraint Logic Programming Language CAL, Procs. of Int'l Conf. on FGCS, (1988)

[11] 渕田孝夫, 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)

[12] R. Dechter and J. Pearl, Tree Clustering for Constraint Networks, Artificial Intelligence 38, (1989)

[13] 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)

[14] R. Dechter, Enhancement Schemes for Constraint Processing: Backjumping, Learning, and Cutset Decomposition, Artificial Intelligence 41, (1990)

[15] B. Buchberger, Gröbner bases: An algorithmic method in polynomial ideal theory, Publications/Reports, RISC-Linz Series no. 83-29.0