ICOT Technical Memorandom: TM-1026

TM-1026

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の検討

永井　保夫（東芝）

February, 1991

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

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の検討

Boolean Algebraic Approach to Constraint Satisfaction Problems

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)

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により，すべての制約を宣言的に記述でき，宣言的なプログラミングを容易にする．制約論理型言語で取り扱う制約の対象領域には，有理数領域，実数領域，ブール代数領域，複素数領域，有限領域などがある [8].

一方，制約充足は人工知能や画像理解の分野をはじめ，グラフ同形判定，グラフラベリング，巡回セールスマン問題などのグラフの問題やパズルなどの探索問題，いわゆる組み合わせ問題を対象として研究が行われている．制約充足問題の代表的な解法としては，探索法や無矛盾性保持 (consistency) 手法を用いる方法がある [4] [1] [5] [2].

本稿では，ブール代数領域を対象とした制約 (これをブール制約という) が取り扱える制約論理型言語を用いた制約充足問題の代数的アプローチについて述べる．このアプローチでは，制約充足問題をブール代数 [9] を用いてブール方程式として定式化し，これらの方程式をブール制約とみなし，ブール方程式の求解を制約論理型言語のブール制約評価系を用いておこなう．さらに，制約集合のもつ構造情報を用いたブール制約評価系の効率化手法についても報告する．そして，制約論理型言語のブール制約を用いて記述されたいくつかの問題に対する本効率化手法の適用実験について述べる．

まず，第 2 章では，制約充足問題について触れ，ブール代数による定式化 [17] を説明する．第 3 章では，ブール代数による制約充足の解法を具体的な例題を用いて説明する．

第 4 章では，制約論理型言語 CAL [12] [13] のブール制約評価系において実現されているブーリアングレブナ基底による解法について説明する．そして，ブール制約評価系の効率化についても検討する．

第 5 章では，ブール制約評価系を用いた制約充足処理の効率化について検討する．特に，ブール制約 (集合) のもつ構造情報を解析し，求められた解析情報を用いて制約評価系を効率化する方法について検討する．

第 6 章では，制約論理型言語 CAL のブール制約を用いて記述したプログラムを示し，いくつかの例題に対して同様の定式化をおこなった．第 7 章では，効率化手法の適用実験ならびに評価をおこない，さらに，効率化についての考察をおこなった．

2 制約充足問題とブール代数による定式化

2.1 制約充足問題

制約充足問題とは，次のような変数の有限集合および各変数に対してとりうる値を離散値の有限集合として与えた場合に，すべての制約を満足するように各変数に対して値を求める問題である．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．制約充足問題では，探索の効率化を目的としたバックトラック探索アルゴリズムの性能を改善する研究がおこなわれている [2] [3] [4] [1].

- 変数: $V = \{v_1, \ldots, v_n\}$
- ドメイン: 各変数 v_n は離散値の有限集合 $D_i = \{d_1, \ldots, d_m\}$, $i = 1, \ldots, n$ から値 d_m を選択する．
- 制約: 集合 $C = \{c_1, \ldots, c_l\}$ where $c_n \equiv (v_1, v_2, \ldots, v_n) \subseteq D_1 \times D_2 \times \ldots \times D_n$

制約充足問題では，制約は制約ネットワークとして静的な問題の記述に適した知識表現をとる場合が多い [2] [3]. その場合，制約ネットワークはグラフによってモデル化され，制約グラフとして表現される． ネットワークを用いたアプローチの利点は，知識の構造化が容易であり，知識の無矛盾性を効率的に管理できることである． 制約ネットワークは，n 個の変数 $v_1, v_2, \ldots, v_n$ ，各変数に対するドメイン $D_1, D_2, \ldots, D_n$ および制約 (n 項関係) 集合をネットワークとして表現したものである． n 個の変数 $v_1, v_2, \ldots, v_n$ に対する制約 $c(v_1, v_2, \ldots, v_n)$ とは， n 個の集合 $\{D_1, D_2, \ldots, D_n\}$ に対して成り立つ関係 (つまり n 項関係) を表し，無矛盾な変数値の直積 $D_1 \times D_2 \times \ldots \times D_n$ の部分集合である．例えば， 2 項制約ネットワークはすべての制約が 2 項関係である (高々 2 個の変数からなる) ネットワークである．

一般に，制約充足とは，制約ネットワークを解くことであり，すべての制約が満足されるようにネットワーク中のすべての変数に対して値を求めることに相当し，単解または全解を求めることが考えられる．

2.2 ブール代数による制約充足問題の定式化検討

ブール代数とは， 0 と 1 のみからなるブール集合 B 上で交換律，結合律，分配律，べき等律，補元律，対合律，

普遍限界の7つの性質を満足する代数系 $< B, \wedge, \vee, \neg, 0, 1 >$ である [9].

以下では，ブール代数を用いたＣＳＰの定式化について示す.

- 変数に関する制約

  変数集合を $V = \{v_1, \ldots, v_n\}$，各変数を v_i がとりうる離散値の有限集合 $D_i = \{d_1, \ldots, d_m\}, (i = 1, \ldots, n)$ とする．変数 v_i に対するドメイン D_i の要素 d_j の割り当てをブール変数 $x_{ij} = 1$ により，割り当て禁止を $x_{ij} = 0$ によりあらわす．なお，以下，とくにことわらない限り，$i = 1, \ldots, n$ とする.

$$\bigvee_{j=1}^{m} x_{ij} = 1 \quad (1)$$

  同時に，各変数 v_i はドメイン D_i からその要素である値 d_j をひとつだけとり，これをブール式として表現すると次のようになる.

$$\bigwedge_{1 \le j,k \le m, j \ne k} \neg x_{ij} \vee \neg x_{ik} = 1 \quad (2.1)$$

  または

$$\bigwedge_{1 \le j,k \le m, j \ne k} x_{ij} \wedge x_{ik} = 0 \quad (2.2)$$

- 2つの変数間において成り立つ制約 (2項制約)

  制約 C_{ij} は，変数集合 V の部分集合である変数 v_i ならびに v_j に対して成立すべき2項制約 V_{ij} として，タプル集合 T_{ij} により表現される.

$$C_{ij} = (V_{ij}, T_{ij}), \quad where \;\; V_{ij} \subseteq V \quad (3)$$

  変数 v_i にドメイン D_i の要素を割り当て変数 v_j にドメイン D_j の要素を割り当てた場合の可能な組み合わせ，もしくは2変数間に対して割り当てを許さない値の組み合わせを2項のタプル集合としてあらわす.

  前者の場合は，変数 v_i と変数 v_j のそれぞれのドメイン D_i と D_j の要素である d_k と d_l の組み合わせを2項タプル集合 $T_{ij} = \{(d_k, d_l) \mid d_k \in D_i \wedge d_l \in D_j\}$ とし，以下のブール式により表現する.

$$\bigvee_{(d_k,d_l) \in T_{ij}} x_{ik} \wedge x_{jl} = 1, \quad j = 1, \ldots, m \quad (4)$$

  後者の場合は，変数 v_i と変数 v_j のそれぞれのドメイン D_i と D_j に対して値の割り当てを許さない d_k と d_l からなる組み合わせを2項のタプル集合 $\overline{T}_{ij} = \{(d_k, d_l) \mid D_i \times D_j - T_{ij}\}$ とし，以下のブール式により表現する.

$$\bigwedge_{(d_k,d_l) \in \overline{T}_{ij}} \neg x_{ik} \vee \neg x_{jl} = 1, \quad j = 1, \ldots, m \quad (5.1)$$

  または，

$$\bigwedge_{(d_k,d_l) \in \overline{T}_{ij}} x_{ik} \wedge x_{jl} = 0, \quad j = 1, \ldots, m \quad (5.2)$$

3 具体例に対するブール代数を用いた制約充足問題の定式化

我々は制約充足問題をブール代数を用いて定式化し，得られたブール方程式に対して上記の解法を適用することで充足解を求めるアプローチをとり，以下では制約充足問題の具体例を用いて説明をおこなう.

3.1 制約充足問題の具体例による説明

図1のような3つの変数 X, Y, Z とそのドメイン D_X $=\{5,2\}$, $D_Y = \{2,4\}$, $D_Z = \{5,2\}$ からなる制約ネットワークを例とした制約充足問題について説明する．制約は変数 X と Z 間の関係を示すタプル集合 $(X,Z) = \{(5,5),(2,2)\}$ と変数 Y と Z 間の関係を示すタプル集合 $(Y,Z) = \{(2,2),(4,2)\}$ からなる.

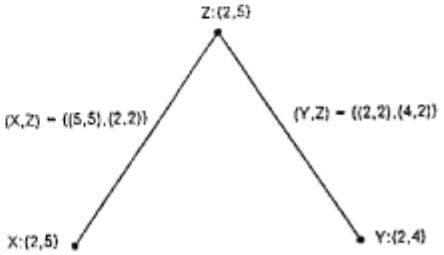

図1: 制約ネットワーク

変数 $X(= v_1)$ はそのドメインが $D_X = \{5,2\}$ であり，ドメインの要素からは値 d_j をひとつだけとるとすると，$X = v_1$, $D_X = D_1$, $d_1 = 5, d_2 = 2$ であり，これをブール式により表現すると次のようになる.

$$x_{11} \vee x_{12} = 1, \quad \neg x_{11} \vee \neg x_{12} = 1$$

同様に，変数 $Y(= v_2)$ と $Z(= v_3)$ に対してもブール式を求めると次のようになる.

$$x_{21} \vee x_{22} = 1, \quad \neg x_{21} \vee \neg x_{22} = 1$$

$$x_{31} \vee x_{32} = 1, \quad \neg x_{31} \vee \neg x_{32} = 1$$

さらに制約 $(X,Z) = \{(5,5),(2,2)\}$ と $(Y,Z) = \{(2,2),(4,2)\}$ には，割り当てる値の組み合わせと値の割り当てを許さない組み合わせの2通りの記述が考えられる．前者の場合の制約には $(X,Z) = T_{13}$, $(Y,Z) = T_{23}$ という関係があるとすると，タプル集合には $T_{13} = \{(d_1,d_1),(d_2,d_2)\}$, $T_{23} = \{(d_1,d_2),(d_2,d_2)\}$ という関係が成り立つとする.

a) 割り当てを許す値の組み合わせを考えた場合

上記の関係を用いて制約 (X,Z) と (Y,Z) のタプル T_{13} と T_{23} をブール式に変換すると次のようになる.

$$(x_{11} \wedge x_{31}) \vee (x_{12} \wedge x_{32}) = 1$$
$$(x_{21} \wedge x_{32}) \vee (x_{22} \wedge x_{32}) = 1$$

前者の式は変数 X と Z 間の制約をあらわし，後者の式は変数 Y と Z 間の制約を示している.

b) 割り当てを許さない値の組み合わせを考えた場合

変数 X と Z 間および変数 Y と Z 間において値の割り当てが許されないタプルをそれぞれ $\overline{T}_{13}$, $\overline{T}_{23}$ とすると

$\overline{T}_{13} = \{(d_1, d_2), (d_2, d_1)\}$, $\overline{T}_{23} = \{(d_1, d_1), (d_2, d_1)\}$ となる．これをブール式により表現すると次のようになる．

$$(\neg x_{11} \vee \neg x_{32}) \wedge (\neg x_{12} \vee \neg x_{31}) = 1$$
$$(\neg x_{21} \vee \neg x_{31}) \wedge (\neg x_{22} \vee \neg x_{31}) = 1$$

または，

$$x_{11} \wedge x_{32} = 0, x_{12} \wedge x_{31} = 0$$
$$x_{21} \wedge x_{31} = 0, x_{22} \wedge x_{31} = 0$$

3.2 生成されるブール方程式と求められるべき充足解

割り当てを許す値の組み合わせと割り当てを許さない値の組み合わせという2つの場合が考えられ，それぞれに対して次のようなブール式の集合 $BE1$ と $BE2$ が生成される．

$$\left.\begin{array}{c} x_{11} \vee x_{12} = 1, \neg x_{11} \vee \neg x_{12} = 1 \\ x_{21} \vee x_{22} = 1, \neg x_{21} \vee \neg x_{22} = 1 \\ x_{31} \vee x_{32} = 1, \neg x_{31} \vee \neg x_{32} = 1 \\ (x_{11} \wedge x_{31}) \vee (x_{12} \wedge x_{32}) = 1 \\ (x_{21} \wedge x_{32}) \vee (x_{22} \wedge x_{32}) = 1 \end{array}\right\} = BE1$$

$$\left.\begin{array}{c} x_{11} \vee x_{12} = 1, \neg x_{11} \vee \neg x_{12} = 1 \\ x_{21} \vee x_{22} = 1, \neg x_{21} \vee \neg x_{22} = 1 \\ x_{31} \vee x_{32} = 1, \neg x_{31} \vee \neg x_{32} = 1 \\ (\neg x_{11} \vee \neg x_{32}) \wedge (\neg x_{12} \vee \neg x_{31}) = 1 \\ (\neg x_{21} \vee \neg x_{31}) \wedge (\neg x_{22} \vee \neg x_{31}) = 1 \end{array}\right\} = BE2$$

一般，制約充足問題をブール代数を用いてブール式に変換すると，ブール式の数が非常に増えるため，規模の大きな方程式を解くことが必要となる．例えば，後者 ($BE2$) の場合には，mn 個の変数 x_{ij} からなる $2n+\mid \overline{T} \mid$ 個のブール式が生成される．

最終的には，上記のブール式の集合 ($BE1$ または $BE2$) を連立方程式として解くことにより，次のような充足解を求めることができる．その結果から解を列挙すると，

$(x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32}) = (0, 1, 0, 1, 0, 1)$ と
$(x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32}) = (0, 1, 1, 0, 0, 1)$
が得られる．前者は充足解が変数 X はドメイン D_X の2番目の要素2，Y は変数 Y のドメイン D_Y の2番目の要素4，Z はドメイン D_Z の2番目の要素2であることを示しており，同様に後者は $X = 2, Y = 2, Z = 2$ であることを示している．

4 制約論理型言語のブール制約評価系

制約論理型言語のブール制約評価系は，数値処理や記号(数式)処理の分野において用いられている代数方程式の解法に基づいている．前者の数値処理分野においては，ガウス消去法に代表される直接法や繰り返し法に基づいた解法が提案されている [7]．一方，後者の記号処理分野においてはコンピュータ環論に基づいた解法が提案されている [6]．

以下では，後者の記号処理分野におけるコンピュータ環論に基づいた解法のひとつであるブーリアングレブナ基底を用いた方法について述べる．

4.1 ブーリアングレブナ基底を用いたブール制約評価系

制約論理型言語 CAL ではブール代数領域においてグレブナ基底を求める制約評価系が提供されている [12] [13]．ブール制約評価系では，代数多項式環により表現された方程式の解法であるグレブナ基底によるアプローチをブール代数の領域に対して適用したものである．つまり，評価系は代数領域多項式のグレブナ基底を求める Buchberger アルゴリズムを拡張し，ブール領域においてグレブナ基底 (これをブーリアングレブナ基底とよぶ) を求めることにより，制約の可解性を判定する．

ブーリアングレブナ基底を求めるアルゴリズムは，ブール多項式に対して，グレブナ基底のアルゴリズムを拡張したものであり，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操作が中心となる．この操作は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ものである．このアルゴリズムは，制約集合に対して，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基底 (正規化された制約) を修正して解を求めるインクリメンタルな性質をもっている．CAL のブール制約評価系は SLD 導出に基づいてインクリメンタルなブーリアングレブナ基底を求めるアルゴリズムを呼び出し，ブール制約集合の可解性を判定し，これらの制約の正規形式 (基底) を求める．

このようなブーリアングレブナ基底によるアプローチの特徴は，1) プログラムやゴール中で陽に記述されていない補助変数は導入する必要がなく，2) すべての制約は効率的な計算がおこなえる正規形をもつということである．

4.2 ブーリアングレブナ基底を用いたブール制約評価系の効率化検討

CAL のブール制約評価系を取り上げ，その効率化について検討する．

CAL のブール制約評価系は，代数制約評価系と同様に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．さらに，制約の並べ変えにより，評価順序を制御できる [13]．

ブール制約評価系では，ブール代数 $< B, \wedge, \vee, \neg, 0, 1 >$ により記述されたブール制約は，ブール多項式環により表現され，これらのブール多項式環に対して，グレブナ基底による解法が適用される．ブール代数上の演算 $\wedge, \vee$ は単位元1をもつブール多項式環 $< B, +, \cdot, 0, 1 >$ によって，$x \vee y =_{def} x + y + x \cdot y$, $x \wedge y =_{def} x \cdot y$, $\neg x =_{def} 1 + x$ により表現される [9]．したがって，$\wedge, \vee, \neg$ によって定義されたブール制約が複雑であれば，それにより表現される多項式環もより複雑になる．そこで，ブーリアングレブナ基底を求める処理の効率化をはかるためには，ブール多項式が複雑にならないように，ブール制約を簡単な形式で与えることが望ましい．つまり，ブール制約を用いた定式化において，より簡単な表現をとることができれば，効率化が期待できる．

また，制約の評価順序であるが，制約が簡単な形式をとるものほど，その評価順序をはじめに設定し，複雑な形式の制約ほど評価順序をあとのほうに設定するほうが効率化が期待できる．そこで，効率化をはかるために，より簡単な形式にできる制約は，できるかぎり簡単化することにする．したがって，変数に関する制約では，(2.1) 式を (2.2) 式に変換し，2 項制約では (4) 式を (5.2) 式に変換する．このような点を考慮してブール式の集合 $BE1$ を簡単化した例を以下の $BE1n$ に示す．

$$\left.\begin{array}{c} x_{11} \wedge x_{12} = 0, x_{21} \wedge x_{22} = 0 \\ x_{31} \wedge x_{32} = 0, x_{11} \wedge x_{32} = 0 \\ x_{12} \wedge x_{31} = 0, x_{21} \wedge x_{31} = 0 \\ x_{22} \wedge x_{31} = 0 \\ x_{11} \vee x_{12} = 1, x_{21} \vee x_{22} = 1 \\ x_{31} \vee x_{32} = 1 \end{array}\right\} = BE1n$$

5 ブール制約評価系を用いた制約充足処理の効率化検討

ブール制約 (集合) がもつ構造的特徴を用いて，ブール制約評価系を効率化し，制約充足処理をおこなう方法について検討する．特に，制約ネットワークの構造情報を用いて，制約間の依存関係を解析し，制約の評価順序を決定することにより，制約充足処理を効率化する手法について述べる．

5.1 代数制約評価系の効率化手法の適用検討

われわれは，制約論理型言語における代数制約評価系が対象とする制約集合を制約グラフとして表現し，グラフ論的手法を用いて代数制約評価系を効率化する手法について提案した．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 [18].

このような効率化手法は制約の数と変数の数を比較して，どちらか一方が他方をおおきく上回る場合には，有効ではない．ブール代数による定式化では，ブール制約は効率化という観点から簡潔な表現形式に変換されるため，一般にブール制約が増加する傾向となる．つまり，取り扱う変数の数と比較して，制約の数が非常に多くなる場合が多い．

そこで，ブール制約に対して上記の代数制約の依存関係解析をおこなうかわりに，ハイパーグラフの概念を取り入れる．ここでは，関連したブール変数を集合とみなし，その要素からなる制約集合をノードとみなし，さらにブール変数の集合間で成立する関係 (制約) 集合をエッジとみなすことにより，構造解析をおこなう．制約充足問題では，対応する制約ネットワークとして表現される構造情報がハイパーグラフの構造に対応する．そこで，以下では，制約充足問題がもつネットワーク形式の構造情報を用いて，ブール制約処理を効率化する手法について説明する．

5.2 制約ネットワークの構造情報を用いたブール制約評価系の効率化

制約ネットワークの構造情報を用いて制約評価系を効率化する手法として，連立方程式解法の一種であるガウス消去法の枢軸 (ピボット) 選択において，グラフ論的なアプローチを用いる方法が考えられる [11]．もうひとつの方法として，制約ネットワークの構造情報からあらかじめ予測できる矛盾を除去し，バックトラックに基づく探索処理の効率化をはかる方法が考えられる [3]．両方の手法ともフィルインという概念を用いて制約充足処理の効率化をはかるものである．フィルインとはもともとガウス消去法の前進消去操作においてもともとゼロだった部分に新たに生成される非ゼロ成分をあらわす．そこで，前進消去において消去すべき変数の順序 (枢軸の選択) を考慮することにより，ガウス消去法において生成されるフィルインの数を制限できる．

[ガウス消去法におけるグラフ論的なアプローチ]

ガウス消去法におけるグラフ論的なアプローチでは，このようなフィルインの数をなるべく少なくするように，変数の消去順序を決定する [11].

n × n の対称行列 $M = (m_{ij})$ からなる連立方程式 M × x = b において，M の対角項である変数 x_i をノード v_i $(i = 1, \ldots, n)$ に対応させ，非ゼロ要素 M_{ij} をエッジ $(i,j) \in E$ に対応させた無向グラフ $G = (V, E)$ を生成する．変数の消去順序は，グラフ $G(V,E)$ に対して，1 対 1 対応の関数 $\alpha : \{1, 2, \cdots, n\} \leftrightarrow V$ を適用することにより決定される．そのとき，$G_\alpha = (V, E, \alpha)$ を順序付きグラフという．さきほど構築されたグラフ $G(V,E)$ はフィルイン数ができるだけ少なくなるような変数消去の順序を選択するために用いられる．フィルインは変数消去によってグラフ G に対して，あらたにエッジが追加されることを示す．行列 M の対角要素の選択がグラフから頂点 v を選択することに相当する．ノード v の欠損 $D(v)$ とは，$D(v) = \{(u,v) \mid (u,v) \in E, (v,w) \in E, (u,w) \notin E\}$ により定義されたエッジ集合であり，ノード v の消去により生成されるフィルインエッジをあらわす．グラフ $G_v = (V - \{v\}, E(V - \{v\}) \cup D(v)$ を G のノード除去グラフという．α によって示された順序にしたがって，ノードを除去した結果，部分グラフの列 $P(G_\alpha) = [G = G_0, G_1, \cdots, G_{n-1}]$ が得られる．この消去順序によって生成されるフィルイン数は $F(G_\alpha) = \bigcup_{i=1}^{n-1} D(v_{\alpha_i})$ により与えられる．

[三角化グラフによるアプローチ]

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 [3]．本方法は制約ネットワークのスパース構造に依存したアプローチであり，制約ネットワークを三角化グラフとして表現し，さらにこれを木構造化されたグラフに変換するものである．制約グラフを三角化することができれば，はじめに与えられた制約充足問題を，非循環グラフ構造をもつ制約充足問題に定式化できるので，探索処理の効率化をはかることができる．三角化処理では，グラフ表現された制約ネットワークに対して，変数の消去順序を決定し，その順序に基づき生成すべきフィルインエッジを加え，コーダル (弦状) グラフを求める．

[制約ネットワークの構造情報を用いた効率化]

制約ネットワークの構造情報を用いた効率化処理では，与えられたグラフに対してフィルインエッジ数をできるだけ少なくするような，変数の消去順序を決定することが必要である．しかしながら，フィルインエッジ数の最小値を求める問題は NP-完全問題である [11]．このような問題に対してはヒューリスティックを用いるアプローチが盛ん

に研究されている．よく知られている手法には最小次数アルゴリズムと最小欠損アルゴリズムがある．最小次数アルゴリズムは変数の将来の消去に関しては考慮しないで，消去すべき変数を選択するグリーディーアルゴリズムである．また，最小欠損アルゴリズムは $D(v)$ が最小となるように，消去すべき変数を選択するアルゴリズムである．今回は前者の最小の次数をもつ変数を選択し，消去順序を決定する最小次数アルゴリズムを適用する．

われわれは制約ネットワークのグラフ表現に対して，上記のヒューリスティックアルゴリズムを適用し，フィルイン数ができるだけ少なくなるようなノードの順序を決定する．そして，順番にノードを誘導し，制約ネットワークを縮退する順序にしたがってブール制約の評価をおこない，最終的に効率的な制約充足をおこなう．つまり，フィルイン概念に基づいて，制約評価をおこなう順序を決定し，ブール制約に関する処理の効率化をはかる．

6 制約論理型言語 CAL のブール制約を用いた記述ならびに効率化手法の適用

本章では，制約論理型言語 CAL のブール制約を用いて記述したプログラムを示し，いくつかの例題に対して同様の定式化をおこない，さらに，上記で示された効率化手法の適用をおこなう．

(例題: 制約ネットワークの記述)

3.1 に示されている例題を制約論理型言語 CAL[6] で記述し，ブール制約評価系を利用してブール方程式の求解をおこなう．

```
:- public c_network/6.

c_network(X11,X12,X21,X22,X31,X32) :-

        bool: X11 \/ X12 =1,
        bool: X21 \/ X22 =1,
        bool: X31 \/ X32 =1,
        bool: ~X11 \/ ~X12 =1,
        bool: ~X21 \/ ~X22 =1,
        bool: ~X31 \/ ~X32 =1,
        bool: (X11 & X31) \/ (X12 & X32) =1,
        bool: (X21 & X32) \/ (X22 & X32) =1.
```

$x_{11} = 0, x_{12} = 1, x_{22} = \bar{x}_{21}, x_{31} = 0, x_{32} = 1$

(クイーン問題の記述)

ここでは，制約充足問題の代表的な例題として N-クイーン問題を取り上げる．

N-クイーン問題は，変数 v_i を盤の $1 \leq n \leq n$ 列に対応させ，それぞれの変数 v_i のドメインを $D_i = \{1, \ldots, n\}$ とし，要素が盤の行をあらわす．クイーンを i 列の j 行目におくことは $v_i = j$ という変数への値の割り当てにより示される．これは，$x_{ij} = 1$ というブール制約により表現される．

次に，4 クイーン問題を 2 項制約を用いた制約充足問題として定式化することを考える [5]．任意の 2 つのクイーンが互いに攻撃しあわないようにする制約は次のように表現される．

$$(v_i \neq v_j) \wedge (|v_i - v_j| \neq j - i) \quad (1 \leq i < j \leq n)$$

これは図 2 のように ${}_nC_2 = n(n-1)/2$ 個の 2 項制約を用いて表現される．また，2 項制約は 2 項のタプルにより表現され，4 クイーン問題では，6 個の 2 項制約によりあらわされる [5] [2]．

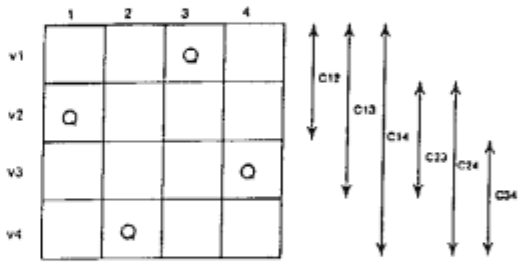

図 2: 4 クイーン問題の定式化

以下のプログラム queen4/16 は，図 2 に示された 2 項制約により表現された 4 クイーン問題を，2.2 節のブール代数を用いて定式化し，制約論理型言語のブール制約で記述したものである．

```
:- public queen4/16.
queen4(X11,X12,X13,X14,X21,X22,X23,X24,
                X31,X32,X33,X34,X41,X42,X43,X44) :-

 bool : X11 \/ X12 \/ X13 \/ X14 = 1. } C10
 bool : X21 \/ X22 \/ X23 \/ X24 = 1. } C20
 bool : X31 \/ X32 \/ X33 \/ X34 = 1. } C30
 bool : X41 \/ X42 \/ X43 \/ X44 = 1. } C40
   bool : ~X11 \/ ~X12 = 1.
   bool : ~X11 \/ ~X13 = 1.
   bool : ~X11 \/ ~X14 = 1.
   bool : ~X12 \/ ~X13 = 1.
   bool : ~X12 \/ ~X14 = 1.
   bool : ~X13 \/ ~X14 = 1.   } C1
   bool : ~X21 \/ ~X22 = 1.
   bool : ~X21 \/ ~X23 = 1.
   bool : ~X21 \/ ~X24 = 1.
   bool : ~X22 \/ ~X23 = 1.
   bool : ~X22 \/ ~X24 = 1.
   bool : ~X23 \/ ~X24 = 1.   } C2
   bool : ~X31 \/ ~X32 = 1.
   bool : ~X31 \/ ~X33 = 1.
   bool : ~X31 \/ ~X34 = 1.
   bool : ~X32 \/ ~X33 = 1.
   bool : ~X32 \/ ~X34 = 1.
   bool : ~X33 \/ ~X34 = 1.   } C3
   bool : ~X41 \/ ~X42 = 1.
   bool : ~X41 \/ ~X43 = 1.
   bool : ~X41 \/ ~X44 = 1.
   bool : ~X42 \/ ~X43 = 1.
   bool : ~X42 \/ ~X44 = 1.
   bool : ~X43 \/ ~X44 = 1.   } C4
 bool : (X11&X23) \/ (X11&X24) \/ (X12&X24) \/
        (X13&X21) \/ (X14&X21) \/ (X14&X22) = 1. } C12
 bool : (X11&X32) \/ (X11&X34) \/ (X12&X31) \/
        (X12&X33) \/ (X13&X32) \/ (X13&X34) \/
        (X14&X31) \/ (X14&X33) = 1.               } C13
```

```
bool : (X11&X42) \/ (X11&X43) \/ (X12&X41) \/
       (X12&X43) \/ (X12&X44) \/ (X13&X41) \/
       (X13&X42) \/ (X13&X44) \/ (X14&X42) \/
       (X14&X43) = 1.                                  } C14
bool : (X21&X33) \/ (X21&X34) \/ (X22&X34) \/
       (X23&X31) \/ (X24&X31) \/ (X24&X32) = 1.         } C23
bool : (X21&X42) \/ (X21&X44) \/ (X22&X41) \/
       (X22&X43) \/ (X23&X42) \/ (X23&X44) \/
       (X24&X41) \/ (X24&X43) = 1.                     } C24
bool : (X31&X43) \/ (X31&X44) \/ (X32&X44) \/
       (X33&X41) \/ (X34&X41) \/ (X34&X42) = 1.         } C34
```

4クイーン問題に対応する制約ネットワークは，図3のように4個のノードからなる完全グラフ K_4 により表現される．各ノードは変数に関する制約に対応する．たとえば，ノード N_1 は変数 v_1 に関する制約集合 $\{C_{10}, C_1\}$ に対応し，ノード N_1 と N_2 の間のエッジ C_{12} は2つの変数 v_1 と v_2 間において成立する2項制約 C_{12} に対応する．

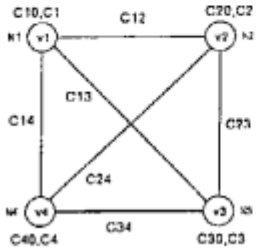

図3：制約ネットワーク（4クイーン問題）

完全グラフは弦状グラフとみなせるなので，変数の消去において誘導されるグラフにおいて新たに生成されるフィルインの数はゼロである．ここでは，変数の消去順序を $v_1 \Rightarrow v_2 \Rightarrow v_3 \Rightarrow v_4$ とし，生成されるノード除去グラフ (v-elimination graph) は図4のようになる．制約の評価順序は制約ネットワークのグラフ表現である K_4 において，変数に対応するノードの除去によるグラフの縮退順序にしたがって決定される [10].

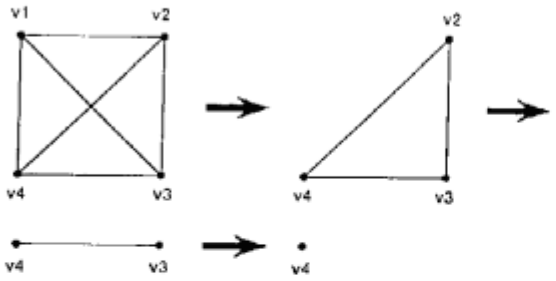

図4：変数消去により求められるグラフ

次のプログラム queen4a/16 は，queen4/16 におけるブール制約を 4.2 節のブール制約集合 $BE1n$ と同様に簡単化したものである．

```
:- public queen4a/16.
queen4a(X11,X12,X13,X14,X21,X22,X23,X24,
        X31,X32,X33,X34,X41,X42,X43,X44) :-

  c10(X11,X12,X13,X14),                          ;C10
  c20(X21,X22,X23,X24),                          ;C20
  c30(X31,X32,X33,X34),                          ;C30
  c40(X41,X42,X43,X44),                          ;C40
  c1(X11,X12,X13,X14,X15),                       ;C1
  c2(X21,X22,X23,X24),                           ;C2
  c3(X31,X32,X33,X34),                           ;C3
  c4(X41,X42,X43,X44),                           ;C4
  c12(X11,X12,X13,X14,X21,X22,X23,X24),          ;C12
  c13(X11,X12,X13,X14,X31,X32,X33,X34),          ;C13
  c14(X11,X12,X13,X14,X41,X42,X43,X44),          ;C14
  c23(X21,X22,X23,X24,X31,X32,X33,X34),          ;C23
  c24(X21,X22,X23,X24,X41,X42,X43,X44),          ;C24
  c34(X31,X32,X33,X34,X41,X42,X43,X44).          ;C34
```

以下に示すプログラム queen4b/16 は，ブール制約が簡約化された queen4a/16 に対して，さらに，図4に示されるような制約集合の構造情報を用いて決定されたノード消去順序に基づいて制約の評価順序を変更したものである．

```
:- public queen4b/16.

queen4b(X11,X12,X13,X14,X21,X22,X23,X24,
        X31,X32,X33,X34,X41,X42,X43,X44) :-

  c1(X11,X12,X13,X14,X15),                       ;C1
  c12(X11,X12,X13,X14,X21,X22,X23,X24),          ;C12
  c13(X11,X12,X13,X14,X31,X32,X33,X34),          ;C13
  c14(X11,X12,X13,X14,X41,X42,X43,X44),          ;C14
  c10(X11,X12,X13,X14),                          ;C10
  c2(X21,X22,X23,X24),                           ;C2
  c23(X21,X22,X23,X24,X31,X32,X33,X34),          ;C23
  c24(X21,X22,X23,X24,X41,X42,X43,X44),          ;C24
  c20(X21,X22,X23,X24),                          ;C20
  c3(X31,X32,X33,X34),                           ;C3
  c34(X31,X32,X33,X34,X41,X42,X43,X44),          ;C34
  c4(X41,X42,X43,X44),                           ;C4
  c30(X31,X32,X33,X34),                          ;C30
  c40(X41,X42,X43,X44).                          ;C40
```

以下では，queen4/16 のブール制約（集合）$C_1, C_2, C_3, C_4, C_{12}, C_{13}, C_{14}, C_{23}, C_{24}, C_{34}$ を簡約化したプログラム $c_1, c_2, c_3, c_4, c_{12}, c_{13}, c_{14}, c_{23}, c_{24}, c_{34}$ を示す．

```
c1(X11,X12,X13,X14) :-

   bool : X11 & X12 = 0,
   bool : X11 & X13 = 0,
   bool : X11 & X14 = 0,
   bool : X12 & X13 = 0,
   bool : X12 & X14 = 0,
   bool : X13 & X14 = 0.

c12(X11,X12,X13,X14,X21,X22,X23,X24) :-

   bool : X11 & X21 = 0,  bool : X11 & X22 = 0,
   bool : X12 & X21 = 0,  bool : X12 & X22 = 0,
   bool : X12 & X23 = 0,  bool : X13 & X22 = 0,
   bool : X13 & X23 = 0,  bool : X13 & X24 = 0,
   bool : X14 & X23 = 0,  bool : X14 & X24 = 0.

c13(X11,X12,X13,X14,X31,X32,X33,X34) :-

   bool : X11 & X31 = 0,  bool : X11 & X31 = 0,
   bool : X11 & X33 = 0,  bool : X12 & X32 = 0,
   bool : X12 & X34 = 0,  bool : X13 & X31 = 0,
   bool : X13 & X33 = 0,  bool : X14 & X32 = 0,
   bool : X14 & X34 = 0
```

```
c14(X11,X12,X13,X14,X41,X42,X43,X44) :-

   bool: X11 & X41 = 0,  bool: X11 & X41 = 0,
   bool: X11 & X44 = 0,  bool: X12 & X42 = 0,
   bool: X13 & X43 = 0,  bool: X14 & X41 = 0,
   bool: X14 & X44 = 0.

c10(X11,X12,X13,X14) :-

   bool: X11 \/ X12 \/ X13 \/ X14 = 1.

c2(X21,X22,X23,X24) :-

   bool: X21 & X22 = 0,  bool: X21 & X22 = 0,
   bool: X21 & X23 = 0,  bool: X21 & X24 = 0,
   bool: X22 & X23 = 0,  bool: X22 & X24 = 0,
   bool: X23 & X24 = 0.

c23(X21,X22,X23,X24,X31,X32,X33,X34) :-

   bool: X21 & X31 = 0,  bool: X21 & X32 = 0,
   bool: X22 & X31 = 0,  bool: X22 & X32 = 0,
   bool: X22 & X33 = 0,  bool: X23 & X32 = 0,
   bool: X23 & X33 = 0,  bool: X23 & X34 = 0,
   bool: X24 & X33 = 0,  bool: X24 & X34 = 0

c24(X21,X22,X23,X24,X41,X42,X43,X44) :-

   bool: X21 & X41 = 0,  bool: X21 & X41 = 0,
   bool: X21 & X43 = 0,  bool: X22 & X42 = 0,
   bool: X22 & X44 = 0,  bool: X23 & X41 = 0,
   bool: X23 & X43 = 0,  bool: X24 & X42 = 0,
   bool: X24 & X44 = 0.

c20(X21,X22,X23,X24) :-

   bool: X21 \/ X22 \/ X23 \/ X24 = 1.

c3(X31,X32,X33,X34) :-

   bool: X31 & X32 = 0,  bool: X31 & X32 = 0,
   bool: X31 & X33 = 0,  bool: X31 & X34 = 0,
   bool: X32 & X33 = 0,  bool: X32 & X34 = 0,
   bool: X33 & X34 = 0

c34(X31,X32,X33,X34,X41,X42,X43,X44) :-

   bool: X31 & X41 = 0,  bool: X31 & X41 = 0,
   bool: X31 & X42 = 0,  bool: X32 & X41 = 0,
   bool: X32 & X42 = 0,  bool: X32 & X43 = 0,
   bool: X33 & X42 = 0,  bool: X33 & X43 = 0,
   bool: X33 & X44 = 0,  bool: X34 & X43 = 0,
   bool: X34 & X44 = 0.

c4(X41,X42,X43,X44) :-

   bool: X41 & X42 = 0,  bool: X41 & X42 = 0,
   bool: X41 & X43 = 0,  bool: X41 & X44 = 0,
   bool: X42 & X43 = 0,  bool: X42 & X44 = 0,
   bool: X43 & X44 = 0,

c30(X31,X32,X33,X34) :-

   bool: X31 \/ X32 \/ X33 \/ X34 = 1.

c40(X41,X42,X43,X44) :-

   bool: X41 \/ X42 \/ X43 \/ X44 = 1.
```

7 実験結果および考察

7.1 実験結果

2項制約を用いて記述された制約充足問題をブール制約を用いて定式化し，制約論理型言語 CAL のブール制約評価系を用いて実験をおこなった．実験結果は表1に示す．実験は逐次マシン 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．

ここでは，3種類の制約ネットワークにより表現された制約充足問題，4クイーン問題ならびに5クイーン問題を対象とした．記号による解法では，ブール制約を制約評価系によって解き，ブール(制約)方程式の正規形を求めている．一方，数値による解法では，ブール変数に対して0ならびに1の値を仮定し，割り当てることにより，制約を簡単化し，解を求めている．

適用前では，2.2節に示された定式化を CAL を用いて記述したプログラムを実行し，適用後1では，適用前に対してブール制約の簡単化をおこなったプログラムを実行した．さらに，適用後2では，適用後1に対して，さらに制約の評価順序を修正したプログラムを実行した．

表1: ブール制約を対象とした問題の評価結果 (単位:msec)

対象とした問題	適用前		適用後1	
	記号	数値	記号	数値
3変数，ドメインの合計要素数6(例題)	51	-	42	-
3変数，ドメインの合計要素数7	105	-	64	-
4変数，ドメインの合計要素数8	95	-	67	-
4クイーン全解(4変数，各ドメインの要素数4)	8862	9724	5844	1400
4クイーン単解	-	2414	-	326
5クイーン全解(5変数，各ドメインの要素数5)	1564474	2426679	135471	11127
5クイーン部分解	-	452154	-	2217

対象とした問題	適用後2	
	記号	数値
3変数，ドメインの合計要素数6(例題)	32	-
3変数，ドメインの合計要素数7	50	-
4変数，ドメインの合計要素数8	50	-
4クイーン全解(4変数，各ドメインの要素数4)	8570	850
4クイーン単解	-	177
5クイーン全解(5変数，各ドメインの要素数5)	734856	7436
5クイーン部分解	-	1427

はじめの3種類の制約ネットワーク問題では，適用後1では，約30パーセントから40パーセントの処理時間の改善がみられた．適用後2では，約49パーセントから63パーセントの処理時間の改善がみられた．

また，4クイーン全解問題の記号による解法では，適用後1の場合で35パーセントの処理時間の改善がみられたが，適用後2の場合にはほとんど改善がみられなかった．一方，数値による解法では，適用後1で約6.4倍の処理時間の改善がみられ，適用後2の場合には約10.5倍の処理時間の改善がみられた．

さらに，5クイーン全解問題の記号による解法では，適用後1の場合で11.5倍の処理時間の改善がみられ，適用後2の場合には約46パーセントの処理時間の改善がみられた．一方，数値による解法では，適用後1の場合で約218倍の処理時間の改善がみられ，適用後2の場合には約326倍もの処理時間の改善がみられた．

7.2 考察

われわれは，ブール制約評価系の効率化手法として，1) 制約の簡約形式を求め，さらに，2) 制約集合の構造情報を用いて制約の評価順序を決定し，その順序に基づいて制約を並び換えるという操作をおこなった．

実験結果から，4クイーンならびに5クイーン問題においては，適用後1(制約の簡約化をおこなう)ならびに適用後2(制約の簡約化をおこない，さらに制約の評価順序を制約集合の構造情報に基づいて変更する)に対する数値による解法が効率化手法として非常に有効であることがわかった．

本効率化手法における制約の簡約表現形式への変換方式は，制約の数の増加にしたがい，非常に多くの制約をあらたに生成することになる．そこで，制約評価アルゴリズムに適したコンパクトな表現形式への変換方式を考える必要がある．

われわれは，制約集合の構造解析において，フィルインという概念にしたがって，ヒューリスティックアルゴリズムを用いて変数の消去順序を決定していたが，複雑な問題に対しては適用していないため，その有効性について検討する必要がある．さらに，ハイパーグラフ表現されたブール制約集合に対して，フィルイン数が最小になるように，変数の消去順序を決定しているが，ハイパーグラフのノードにあたるブール変数の数ならびにエッジにあたる関係(タプル)の数を考慮することが必要となる．

次に，本効率化手法と forward checking 法との関連について述べる．forward checking 法 [15] は，制約を能動的に用いて，解のなかで存在してはならない値の組み合わせをあらかじめ取り除き，探索空間の絞り込み(ドメインの縮小)をおこなう効率的な探索手法の一種である．本手法は次のような処理を繰り返す: (1) できるだけたくさんの制約を伝播し，(2) 問題が解けるまで，いくつかの変数に対して値を割り当てていく．これは制約ネットワークにおける無矛盾性保持手法に基づいた手法であり，従来のバックトラック探索をより高度化して利用されている．このような方法を取り入れた論理型言語に CHIP がある [14]．

ブール制約評価系を用いた制約充足処理の効率化手法として，さきほど述べたように制約ネットワークの構造関係を解析し，ノード，つまり，変数に相当する部分の制約集合の評価順序を決定する手法，すなわち適用後2に対する数値による解法は，まさに，さきほど述べた forward checking 法を実現していると考えられる．しかしながら，本手法では，制約が静的に与えられている場合には，十分に対処できるが，制約を動的に生成しながら，制約充足をおこなう場合には対処ができない．そこで，より幅広い範囲の問題を扱うためには，制約の動的生成についてある程度予測する方法が必要になると考えられる．

8 おわりに

ブール代数領域を対象とした制約を取り扱う制約論理型言語を用いて，制約充足問題に対する従来の探索を基本とした解法とは異なるあらたな代数的な解法を提案した．本提案は制約充足問題をブール代数により定式化し，これをブール制約とみなし，制約評価系を用いて制約充足をおこなうアプローチである．具体的には制約充足の例題を取り上げて定式化をおこない，ブール制約を解くために，制約論理プログラミング言語 CAL のブール制約評価系を用いた．その結果，制約充足問題の解法として探索法がよく知られているが，それとは別にブール代数評価系を用いた解析的な方法が有効であることを示した．さらに，制約評価系の効率化をはかるために，制約集合のもつ構造情報を用いた手法について検討した．また，いくつかの制約充足問題に対して本効率化手法の適用実験をおこない，その有効性を示した．

しかしながら，先程示したように定式化で得られるブール式の数が増えるので，規模の大きな問題に対処するためにはより少ないブール式に変換できる定式化を考えることが必要である．

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，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] Johan de Kleer, A Comparison of ATMS and CSP Techniques, IJCAI-89, (1989)

[2] A. Dechter and R. Dechter, Removing Redundancies in Constraint Networks, Proc. of AAAI-87, (1987)

[3] Rina Dechter and Judea Pearl, Tree Clustering for Constraint Networks, Artificial Intelligence 38, (1989)

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

[5] B. A. Nadel, Representation Selection for Constraint Satisfaction: A Case Study Using n-Queens, IEEE Expert June, (1990)

[6] 佐々木建昭，古川昭夫，コンピュータ環論，情報処理 Vol.27 No.4, (1986)

[7] 津田孝夫，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)

[8] J. Cohen, Constraint Logic Programming Languages, Comms. of the ACM, Vol.33, No.7, (1990)

[9] R. Lidl and G. Pilz, Applied Abstract Algebra, Springer-Verlag, (1984)

[10] E. C. Freuder, Complexity of K-tree Structured Constraint Satisfaction Problems, Procs. of AAAA-90, (1990)

[11] D. J. Rose, R. E. Tarjan, G. S. Lueker, Algorithmic Aspects of Vertex Elimination on Graphs, SIAM Journal on Computing Vol. 5, No. 2, (1976)

[12] K. Sakai and A. Aiba, CAL: Theoretical Background of Constraint Logic Programming and its Application, J. of Symbolic Computation 8, (1989)

[13] K. Sakai and Y. Sato, Application of Ideal theory to Boolean constraint solving, Proc. of PRICAI-90, (1990)

[14] P. V. Hentenryck, Constraint Satisfaction in Logic Programming, The MIT Press, (1989)

[15] M. Dincbas, Constraints, Logic Programming and Deductive Databases, Proc. of France-Japan Artificial Intelligence and Computer Science Symposium 86, (1986)

[16] A. Colmerauer, An Introduction to Prolog III, Communications of The ACM, Vol.33, No.7, (1990)

[17] 永井保夫，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 1K-3，(1990)

[18] 永井保夫，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，人工知能学会研究会資料 SIG-FAI-HICG-KBS-9001-12，(1990)