離散組み合わせ問題の制約充足問題による定式化ならびに整合化手法の適用検討

永 井 保 夫*

本論文では、離散組み合わせ問題、特に、地図の彩色問題とクロスワードパズル問題を制約充足問題として定式化するとともに、整合化手法を用いた制約充足解法を適用し、その有効性を示す。

キーワード：制約充足問題，離散組み合わせ問題，定式化，整合化手法，効率性

Towards CSP Formalization of Discrete Combinatorial Problems and an Application of Consistency Techniques

Yasuo NAGAI

In this paper, we describe a formalization of discrete combinatorial problems from viewpoint of constraint satisfaction problems and the consistency methods for constraint satisfaction problems. Effectiveness of these consistency methods is shown using application of these methods to map coloring problem and crossword puzzle problem.

Keyword：constraint satisfaction problem, discrete combinatorial problem, formalization, consistency method, efficiency

1 はじめに

制約充足問題は人工知能や画像解析の分野をはじめ、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おこなわれている［1］［2］［3］［4］［5］［6］［7］［8］［10］［11］。制約充足問題とは、変数の有限集合および各変数に対応する離散値の有限集合のドメインから値を選択し、すべての制約を満足するように各変数に対して値を割り当てる問題である。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。制約充足問題における従来の解法としては、バックトラックを基本とした探索による方法が代表的である。探索の効率化を図るために、整合化手法を前処理として利用したり、探索に組み込むなどの処理がおこなわれている。

本論文では、まず、制約充足問題による定式化に代表される制約指向アプローチのメリットについて論じる。次に、制約充足問題の定義と定式化について述べる。そこでは、具体的に、離散組み合わせ問題の代表的な問題である地図の色塗り問題とクロスワードパズル問題を取り上げ、制約充足問題としての定式化を説明する。さらに、制約充足問題の解法である整合化手法について説明する。最終的に、定式化された地図の色塗り問題とクロスワードパズル問題に対して、整合化手法を適用した結果について示すことで、効率化手法としての整合化手法の有効性について明らかにする。

*東京情報大学総合情報学部情報システム学科
Tokyo University of Information Sciences, Faculty of Informatics, Department of Information Systems

2005年11月 9 日受理

2　制約指向によるアプローチのメリット

2.1　知識表現・問題解決

制約指向導入のメリットは、次のような制約知識の取り扱いにより常識に関する知識がきめ細かく記述できるので表現能力を向上させ、ユーザの知識の記述の負担を軽減できることにある。

・入出力による処理の流れを固定化しない記述による知識の維持の容易性
・制約知識を複数の方法により利用できる
・制約知識を様々方法で表現できる（特に、量についての知識の表現）
・知識のインクリメンタルな追加や修正
・知識ベースをよりコンパクト化できる

従来の問題解決では、問題領域と取り扱うべき対象を設定し、それらの関係を明確化し、さらに問題を解く手法を見い出すことが必要である。制約問題解決では、問題を解析し対象間に成立する関係を方向性を与えない形で表現し、それらの関係を満足させる問題解決はシステムがおこなう。従来型の問題解決では、このような対象間に成立する関係には方向性が与えられ、手続き的な知識として表現されるのに対して、制約による問題解決ではこれらの関係は制約知識として表現されている。手続き的な知識を用いた問題解決では、制約を用いた問題解決と比較して問題解決の制御に関する知識を過剰に記述したり、記述が冗長になったり、入出力関係が厳密に与えられている必要が生じる。これに対して、制約を利用した問題解決では、知識を制約として定式化し、これらの制約を解く（満足される）方法については問題解決する側に任せる形で解を求めていく。

このような制約を利用した問題解決のメリットとしては、以下のような処理の効率化と処理の高度化の2つの側面からなる問題解決能力の向上を期待できる。

・処理の効率化
　制約を利用した問題解決の中で特に探索処理の効率化が重要である。
　－　局所的な整合性の保持
　－　探索空間の大域的な（最適化）視点の提供
　－　局所化ならびに特化された問題解決手法の提供
　－　適用領域に依存しない問題解決（探索）ヒューリスティックスの提供
・処理の高度（多機能）化
　制約を利用した問題解決における処理の高度化では、様々な問題解決機構が提供されることが必要である。
　－　様々な適用領域向け問題解決手法（機構）の提供
　－　最適化問題の取り扱い

3　制約充足問題

3.1　制約充足問題の基本的な考え方

図1に示すような制約充足問題の基本的な考え方は、制約を明らかにすることで問題を定義し、この制約に基づいて問題解決をおこなうものである。もう少し具体的に言うと、制約充足問題では、対象とする問題を変数と変数間の制約として表現し、制約を満足するように変数への値の割り当てを求める。

制約充足問題として問題を定式化するメリットは、以下の点にある［1］［2］［3］［5］［6］。

・対象となる問題を、制約充足問題として変数集合、変数のドメイン集合（変数に対して割り当てる候補となる値の集合）、制約集合からなる共通形式の表現に変換できるため、汎用的な記述として取り扱え、利用することができる。
・対象となる問題の固有の専門知識を要求しないで、汎用のヒューリスティックスが検討できる。
・対象となる問題の構造が制約ネットワーク（グラフ）として表現できる場合には、その構造情報を用いて問題解決プロセスを単純化できる場合がある。この場合には、複雑な問題でも簡略化することが可能である。

以下では、制約充足問題として定式化を考えていくために、2つの離散組み合わせ問題を対象として取り上げる。

［例題1：地図の色塗り問題］

地図の色塗り問題［5］とは、地図といくつかの色が与えられた時、隣接した領域では同じ色にならないように、それぞれの領域に色を割り当てていく問題である（図2）。

［例題2：クロスワードパズル］

図3に示されるクロスワードパズル問題は、以下の単語リストから選んだ単語を空欄の縦方向と横方向に

埋めるものである。但し、縦方向と横方向の単語が交差する欄には同じ文字が入るものとする[10]。

・単語リスト
- HOSES, LASER, SAILS, SHEET, STEER
- HEEL, HIKE, KEEL, KNOT, LINE
- AFT, ALE, EEL, LEE, TIE

3.2 制約充足問題の定義と問題の定式化

以下では、制約充足問題の定義について説明し、この定義に基づいて前述した問題を定式化する。

3.2.1 制約充足問題の定義

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応する離散値の有限集合のドメインから値を選択し、すべての制約を満足するように各変数に対して値を割り当てる問題である。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方向性を与えない形で記述したものである。

・変数集合：$V=\{v_1,\ldots,v_n\}$, 各 v_i は互いに独立な変数。

・ドメイン：離散値の有限集合　$D_i=\{d_{i_1},\ldots,d_{i_m}\}$, $i=1,\ldots,n$, d_{i_j}（$j=1,\ldots,m$）は数値または記号値をあらわす。各変数 v_i には D_i の要素である値 d_{i_m} が割り当てられる。

・制約集合：$C=\{c_1,\ldots,c_t\}$. ここで、各 c_i は制約で、それは次のような等式（$v_1, v_2,\ldots,v_n$）=〈直積 $D_1\times D_2\times\ldots\times D_n$ の部分集合〉
の表現形式をとる。

制約充足問題では、変数を表すノードと制約によりラベル付けされたアーク（エッジ）からなるグラフとして表現される。このようなグラフ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とみなされる。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$\mathcal{R}$は、n 個の変数 $v_1, v_2,\ldots,v_n$ を要素とする集合V 、各変数に対するドメイン $D_1, D_2,\ldots, D_n$ を要素とする集合 D および $c_1, c_2,\ldots,c_n$ を要素とする制約（ n 項関係）集合から構成され、三つ組（V, D, C）により表現される。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、二項制約

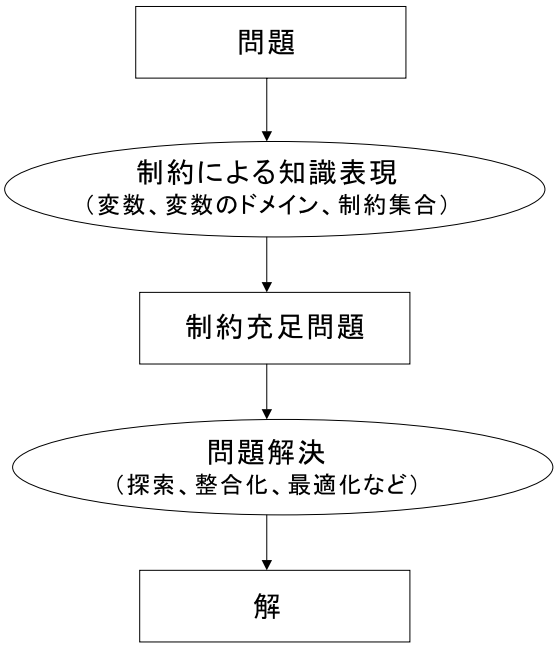

図１　制約充足問題の基本的な考え方

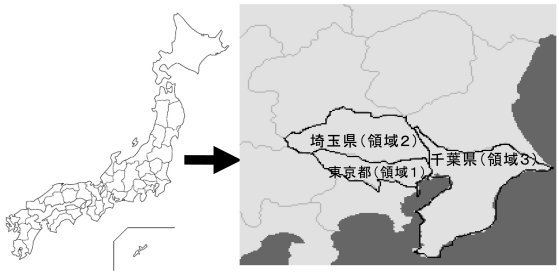

図２　地図の色塗り問題

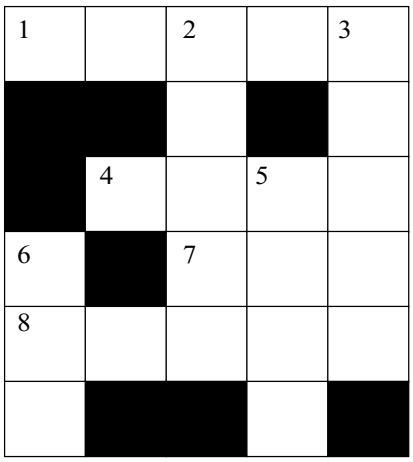

図３　クロスワードパズル

ネットワークはすべての制約が二項関係である（高々2個の変数からなる）ネットワークである。制約充足問題では、n 個の変数に対する制約、つまり n 項制約は二項制約として表現できるので、今後は、二項制約のみを対象とした制約充足問題を考える。

それから、変数に対してドメインから値が割り当てられるとき、変数が具体化されたという。変数集合の具体化とは、各変数のドメインからの値の割り当てを意味する。つまり、変数集合 $\{v_{i_1}, \ldots, v_{i_k}\}$ の具体化とは各変数への値の割り当てを示し、変数と変数に割り当てられる値のペアのタプル（$\langle v_{i_1}, a_{i_1}\rangle, \ldots, \langle v_{i_k}, a_{i_k}\rangle$）として表される。但し、各ペア（$\langle v, a\rangle$）は変数 v への値 a の割り当てを、a は v のドメインを示す。このような変数集合の具体化は（$v_1 = a_1, \ldots, v_i = a_i$）と表現する。たとえば、（$\langle v_1, a_1\rangle, \ldots, \langle v_i, a_i\rangle$）は $\bar{a} = (a_1, \ldots, a_i)$ と表す。

制約ネットワーク $\mathcal{R} = (V, D, C)$ の解とは、すべての制約を満足するようなすべての変数の具体化を意味する。一般に、制約充足（制約を満足させる）とは、このような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を意味する。

次に、制約充足問題として前述した2つの問題を定式化する。

3.2.2　問題の定式化

［例題1：地図の色塗り問題の定式化］

図2に示す地図の色塗り問題の制約充足問題としての定式化は以下のとおりである。対応する制約ネットワークは図4に示される。

- 変数集合：$V = \{v_1, \ldots, v_n\}$，各 v_i は互いに独立な変数。
  $V = \{v_1$（＝領域1），v_2（＝領域2），v_3（＝領域3）$\}$
- ドメイン：離散値の有限集合　$D_i = \{d_{i_1}, \ldots, d_{i_m}\}$，$i = 1, \ldots, n$, d_{i_j}（$j = 1, \ldots, m$）は数値または記号値をあらわす。
  $D_1 = \{red, green, blue\}$, $D_2 = \{red, green\}$, $D_3 = \{green\}$
- 制約集合：$C = \{c_1, \ldots, c_l\}$．ここで、各 c_i は、v_i と v_j と隣接していれば、v_i と v_j は同じ色を塗れない（$v_i \neq v_j$）という制約をあらわす。
  $c_1 = C_{12}$[1] $= \{(red, green), (green, red), (blue, red), (blue, green)\}$
  $c_2 = C_{23} = \{(red, green)\}$
  $c_1 = C_{13} = \{(red, green), (blue, green)\}$

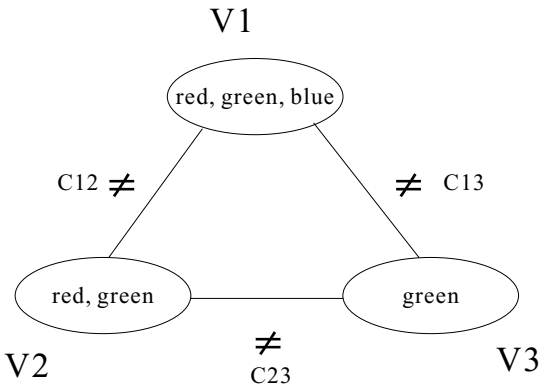

図4　地図の色塗り問題の制約ネットワーク表現

［例題2：クロスワードパズル問題の定式化］

図3に示されるクロスワードパズル問題の制約充足問題としての定式化は、以下のとおりである。対応する制約ネットワークは、図5に示される。

- 変数集合：$V = \{v_1, \ldots, v_n\}$，各 v_i は互いに独立な変数。ここでは、単語を埋める縦方向と横方向の並びに対応する。
  $V = \{v_1$（＝1の横方向の並び），v_2（＝2の縦方向の並び），v_3（＝3の縦方向の並び），v_4（＝4の横方向の並び），v_5（＝5の縦方向の並び），v_6（＝6の縦方向の並び），v_7（＝7の横方向の並び），v_8（＝8の横方向の並び）$\}$
- ドメイン：離散値の有限集合　$D_i = \{d_{i_1}, \ldots, d_{i_m}\}$，$i = 1, \ldots, n$, d_{i_j}（$j = 1, \ldots, m$）は数値または記号値をあらわす。ここでは、単語リスト中の単語の集合が対応する。
  $-D_1 = D_2 = D_3 = D_4 = D_5 = D_6 = D_7 = D_8 = \{AFT, ALE, EEL, HEEL, HIKE, HOSES, KEEL, KNOT, LASER, LEE, LINE, SAILS, SHEET, STEER, TIE\}$
- 制約集合：$C = \{c_1, \ldots, c_l\}$．ここでは、c_i には、v_i に関する制約と、v_i と v_j 間に成り立つ制約の2種類を考える。

[1] 大文字で表されている C_{12} は、図4の制約ネットワークの二項制約を表す。なお、以下に示される C_{23} と C_{13} も同様に制約ネットワークの二項制約を表す。

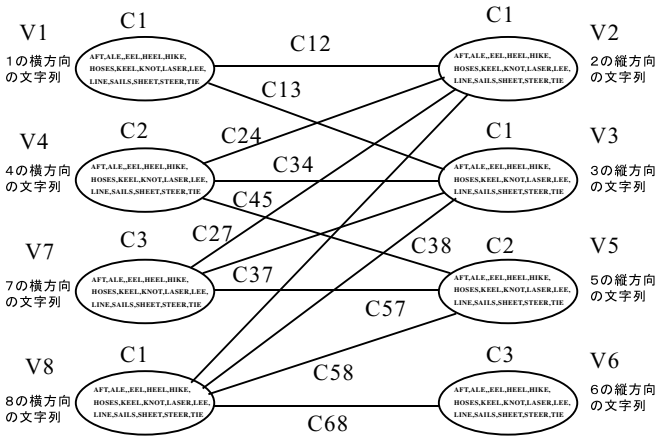

図5　クロスワードパズル問題の制約ネットワーク表現

前者の v_i に関する制約は以下の通りである。

- 1の横方向の並び、2の縦方向の並び、3の縦方向の並び、および8の横方向の並びは単語リストから長さ5の文字列が選択される（$=c_1=C_1$[2]）。
- 4の縦方向の並びと5の縦方向の並びには単語リストから長さ4の文字列が選択される（$=c_2=C_2$）。
- 7の横方向の並びと6の縦方向の並びには単語リストから長さ3の文字列が選択される（$=c_3=C_3$）。

次に、後者の v_i と v_j 間の制約は以下の通りである。

- $c_4=C_{12}$[3]$=\{$(*HOSES*, *SAILS*)，(*HOSES*, *SHEET*)，(*HOSES*, *STEER*)，(*LASER*, *SAILS*)，(*LASER*, *SHEET*)，(*LASER*, *STEER*)$\}$
  v_1 の第3文字目と v_2 の第1文字目が等しい制約を表す。
- $c_5=C_{13}=\{$(*HOSES*, *SAILS*)，(*HOSES*, *SHEET*)，(*HOSES*, *STEER*)，(*SAILS*, *SAILS*)，(*SAILS*, *SHEET*)，(*SAILS*, *STEER*)$\}$
  v_1 の第5文字目と v_3 の第1文字目が等しい制約を表す。
- $c_6=C_{24}=\{$(*SAILS*, *HEEL*)，(*SAILS*, *HIKE*)，(*SAILS*, *KEEL*)，(*SAILS*, *LINE*)，(*SHEET*, *HEEL*)，(*SHEET*, *HIKE*)，(*SHEET*, *KEEL*)，(*SHEET*, *LINE*)，(*STEER*, *HEEL*)，(*STEER*, *HIKE*)，(*STEER*, *KEEL*)，(*STEER*, *LINE*)$\}$
  v_2 の第3文字目と v_4 の第2文字目が等しい制約を表す。
- $c_7=C_{27}=\{$(*HOSES*, *EEL*)，(*HOSES*, *LEE*)，(*LASER*, *EEL*)，(*LASER*, *LEE*)，(*SAILS*, *EEL*)，(*SAILS*, *LEE*)，(*SHEET*, *EEL*)，(*SHEET*, *LEE*)，(*STEER*, *EEL*)，(*STEER*, *LEE*)$\}$
  v_2 の第4文字目と v_7 の第1文字目が等しい制約を表す。
- $c_8=C_{28}=\{$(*HOSES*, *HOSES*)，(*HOSES*, *LASER*)，(*SAILS*, *HOSES*)，(*SAILS*, *LASER*)$\}$
  v_2 の第5文字目と v_7 の第3文字目が等しい制約を表す。
- $c_9=C_{34}=\{$(*SHEET*, *HIKE*)，(*SHEET*, *LINE*)，(*STEER*, *HIKE*)，(*STEER*, *LINE*)$\}$
  v_3 の第3文字目と v_4 の第4文字目が等しい制約を表す。
- $c_{10}=C_{37}=\{$(*SHEET*, *ALE*)，(*SHEET*, *LEE*)，(*SHEET*, *TIE*)，(*STEER*, *ALE*)，(*STEER*,

[2] 大文字で表されている C_1 は、図5の制約ネットワークにおける単項制約を表す。なお、以下に示される C_2 と C_3 も同様に制約ネットワークの単項制約を表す。

[3] 大文字で表されている C_{12} は、図5の制約ネットワークにおける二項制約を表す。なお、以下に示される C_{13}, C_{24}, C_{27}, C_{28}, C_{34}, C_{37}, C_{38}, C_{45}, C_{57}, C_{58}, C_{68} も同様に制約ネットワークの二項制約を表す。

LEE), (*STEER, TIE*)}

v_3 の第 4 文字目と v_7 の第 3 文字目が等しい制約を表す。

- $c_{11} = C_{38}$ = {(*HOSES, HOSES*), (*HOSES, SAILS*), (*LASER, LASER*), (*LASER, STEER*), (*SAILS, SAILS*), (*SAILS, HOSES*), (*STEER, LASER*), (*SHEET, SHEET*), (*STEER, STEER*)}

  v_3 の第 5 文字目と v_7 の第 5 文字目が等しい制約を表す。
- $c_{12} = C_{45}$ = {(*HIKE, KEEL*), (*HIKE, KNOT*)}

  v_4 の第 3 文字目と v_5 の第 1 文字目が等しい制約を表す。
- $c_{13} = C_{57}$ = {(*HEEL, EEL*), (*HEEL, LEE*), (*HIKE, TIE*), (*KEEL, EEL*), (*KEEL, LEE*), (*LINE, TIE*),}

  v_5 の第 2 文字目と v_8 の第 2 文字目が等しい制約を表す。
- $c_{14} = C_{58}$ = {(*HEEL, HOSES*), (*HEEL, LASER*), (*HEEL, SHEET*), (*HEEL, STEER*), (*KEEL, HOSES*), (*KEEL, LASER*), (*KEEL, SHEET*), (*KEEL, STEER*)}

  v_5 の第 3 文字目と v_8 の第 4 文字目が等しい制約を表す。
- $c_{15} = C_{68}$ = {(*ALE, LASER*)}

  v_6 の第 2 文字目と v_8 の第 1 文字目が等しい制約を表す。

3.3　制約充足問題の解法

制約充足問題は、一般的には探索問題として表現される。制約充足問題の解法は、大別すると整合化手法を用いるものと探索手法を用いるものに分けられる。前者の整合化手法は解を求めるまでに変数のドメインから矛盾した値を除去する手法であり、後者の探索手法はバックトラックを基本とした手法であり、探索枝の刈り込みをおこなうフォーワードチェッキングや効率性を考慮した値割り当ての先読みなどがある。また、大規模な問題を対象とした単解を求める高速制約充足解法として、ヒューリスティックスを用いた局所探索法などがある［1］［2］［5］［8］。

3.3.1　整合化手法を用いる解法

制約ネットワークの整合化手法では、制約ネットワークにおいて変数に対する無効な値をできるだけ取り除くことにより探索空間をできるだけ小さくしておき、探索の効率化を支援する手法である。

整合化手法では、解くべき制約充足問題のドメインや制約を簡略化することにより、より単純な問題へ変換する。ただし、変換された問題は、もとの問題と同じ解集合をもつことが必要である。この整合化手法を用いることで、もとの問題のもつ探索空間を狭めたり、制約を一層明示的な形で表現できるようになるので、もとの問題と比較してより容易に問題を解くことが期待できる。このような整合化手法の大部分は完全なアルゴリズムではないため、制約充足問題を完全に解くためには、単独ではほとんど利用されないで、探索手法と組み合わせて用いられる。

前述したように、制約充足問題は通常変数をあらわすノード、制約によりラベル付けされたエッジからなるグラフとして表現され、このグラフは制約ネットワークと呼ばれている。以下で取り上げる基本的な整合化手法は、このような制約ネットワークのグラフ表記（ノード、アーク、パスなど）から命名されている。任意の制約充足問題は等価な二項制約充足問題に変換可能であるが、本論文では二項制約充足問題（すなわち、単項制約と二項制約のみから構成される制約充足問題）のみを取り上げる。以下では、二項制約充足問題の整合化アルゴリズムを対象として説明する。なお、任意の（非二項）制約充足問題を取り扱うためには、二項制約充足問題のアルゴリズムの改良が必要となる。

1）ノード整合

最も単純な整合化手法は、ノード整合（node-consistency：NC）と呼ばれるものである。この手法は、問題をグラフ表現した制約ネットワークにおけるノードの整合化を行うもので、ノードに対応する各変数に関する単項制約と矛盾する変数のドメインから値を除去する。

［定義 1（ノード整合）］

すべての変数に対して、変数のドメインの値が変数上の制約を満足するとき、制約充足問題はノード整合である（node-consistent）という。

ノード整合では、該当するノード（たとえば、ノード i）に対応するドメイン D_i が単項制約 C_i を満たしているかをチェックする。ドメイン D_i のすべての値が

ノードに与えられた単項制約 C_i を満たしていれば、ノード i はノード整合であるといえる。

なお、ノード整合の詳しいアルゴリズムは、図 6 に示される。

```
procedure NC-1
入力：制約ネットワーク R = (V, D, C)
出力：ノード整合な Di
1  begin
2    for all vi∈V do
3      for all a∈Di do
4        if 変数 vi への値 a の割り当てが制約 Ci を満足しない
         then Di から { a } を削除;
5  end;
```

図 6　ノード整合アルゴリズム

2）アーク整合

最も幅広く利用されている整合化手法がアーク整合(arc-consistency：AC）である。この手法では、制約ネットワークにおけるアークの整合化を行い、アークに対応する二項制約と矛盾する変数のドメインから値を除去する。つまり、アーク v_i と v_j がアーク整合であるとは、v_i に関する制約を満足する v_i のドメインの各値に対して、変数 v_i と v_j 間での二項制約により規定された $v_i=x$ と $v_j = y$ が成り立つような v_j のドメインの値が存在することをいう。

[定義 2 （アーク整合)]

制約ネットワーク $\mathcal{R} = (V, D, C)$（ただし、$C_{ij}\in C$）が与えられた時、すべての値 a_i（$\in D_i$）に対して、$(a_i, a_j)\in C_{ij}$ となるような値 a_j（$\in D_j$）が存在するならば、その時に限り変数 v_i は v_j に対してアーク整合である（arc-consistent）という。

アーク整合では、アークの両端であるノード i とノード j に対応するドメイン D_i と D_j のそれぞれの要素に対する操作をおこなう。具体的には、アークの存在する 2 つのノード間の関係（これを 2 項制約という）に基づき、この情報を満たさないドメインの要素の組み合わせを局所的に除去していく。アーク整合アルゴリズムはAC-1 から始まり、近年はAC-7 まで研究がおこなわれてきた。これらのアルゴリズムは、矛盾のない状態に到達するか、ドメインの値がなくなるまでアークに対する整合化が繰り返しおこなわれる。

なお、アーク整合の詳しいアルゴリズムは、以下のとおりである。

```
procedure AC-1
入力：制約ネットワーク R = (V, D, C)
出力：Rと等価なアーク整合なネットワーク R´
1  begin
2    repeat
3      for 制約に関連するすべての変数に対して割り当てられた値のペア（vi, vj）
4        REVISE（(vi), vj）
5        REVISE（(vj), vi）
6    until ドメインが変化しなくなる
7  end;
```

図 7　アーク整合アルゴリズム

```
procedure REVISE（(vi), vj）
入力：2 変数 X = { xi, xj } により定義されるネットワーク,
        変数 xi, xj, ドメイン Di, Dj, 制約 Cij
出力：変数 xi が xj に対してアーク整合であるようなドメイン
      Di
1  begin
2    for each ai∈Di
3      if（ai, aj）∈Rij となるような aj∈Dj が存在しない
4        then Di から { ai } を削除
5  end;
```

図 8　REVISE 手続き

図 7 のアーク整合アルゴリズムで呼び出している REVISE 手続きでは、図 8 に示される処理をおこなう。

処理概要は以下の通りである。

1. 2 つのノード v_i, v_j 間の 2 項制約をチェックして、D_i のすべての要素から 2 項制約を満たさない要素を除去する。
2. ネットワークのすべてのアークに対して、同様な操作を行い、すべてのノードのドメインに変化がなくなったときに処理が終了する。

3）パス整合

パス整合（path-consistemcy：PC）では、 2 つの変数 x と y の値からなる組のすべてに対して、パス間のすべての二項制約が満足されるように x と y 間の経路に沿った各変数に対して値が存在することが要求される。パス整合アルゴリズムでは、上記のノード整合アルゴリズムやアーク整合アルゴリズムと比べて、より多くの矛盾した値の除去が可能である。 なお、長さが 2 のすべての経路がパス整合であれば、そのときに限り制約充足問題はパス整合であることが示されている。

[定義 3 （パス整合)]

制約ネットワーク $\mathcal{R} = (V, D, C)$ が与えられたとき、

すべての無矛盾な変数への値の割り当て（$\langle v_i, a_i\rangle, \langle v_j, a_j\rangle$）に対して、変数への値の割り当て（$\langle v_i, a_i\rangle, \langle v_k, a_k\rangle$）と（$\langle v_k, a_k\rangle, \langle v_j, a_j\rangle$）が無矛盾になるような値 $a_k \in D_k$ が存在するならば、その時に限り、2つの変数集合 $\{v_i, v_j\}$ はパス整合（path-consistent）であるという。

つまり、二項制約 C_{ij} のすべての無矛盾な要素の組である（a_i, a_j）（ただし、a_i は D_i の要素、a_j は D_j の要素）に対して、（a_i, a_k）が R_{ik} の要素となり、（a_k, a_j）が R_{kj} の要素となるような、D_k の要素である a_k が存在すれば、そのときに限り、二項制約 C_{ij} は v_k に関してパス整合であるという。

なお、パス整合の詳しいアルゴリズムは、図9に示される。

図10のREVISE-3 手続きでは、3つの変数からなるネットワーク、3つの二項制約 C_{xy}, C_{yz}, C_{xz} が入力として与えられる。C_{xy} の要素である無矛盾な値のペアのすべてが、値 z を与えることで矛盾を生じないかをチェックし、矛盾を発生する場合にはそのペアを除去する。最終的には、z に対してパス整合となるような二項制約 C_{xy} を求める。

```
procedure PC-1
入力：ネットワーク𝓡 = (V, D, C)
出力：ネットワーク𝓡と等価でパス整合な関係をもつネットワーク𝓡′
1  begin
2    repeat
3      for k ← to n
4        for i, j ← 1 to n
5          REVISE-3 ((i, j), k))
6    until 制約が変化しない
7  end;
```

図9　パス整合アルゴリズム

```
procedure REVISE-3 ((x, y), z)
入力：変数 x, y, z から構成されるネットワーク, 制約 Cxy, Cyz, Cxz
出力：変数 z とパス整合関係を維持するように修正された制約 Cxy
1  begin
2    for each pair (a, b) ∈ Cxy
3      if (a, c) ∈ Cxz and (b, c) ∈ Cyz となるような c ∈ Dz が存在しない
4        then Cxy から (a, b) を削除する
5  end;
```

図10　REVISE-3 手続き

今まで説明してきたように、パス整合はアーク整合と比較してより強力な整合化の考え方であるが、実際には以下の点から余り利用されていないのが現状である。

- パス整合の適用はアーク整合の適用と比較してより多くの不整合を除去できる。しかしながら、アルゴリズムの性能はアーク整合と比較して劣っている。
- パス整合の適用により新たな制約が導出され、制約ネットワークの接続関係が変化する。そのために、ネットワーク構造を利用した解法が利用できない。
- パス整合の適用では、すべての不整合を除去できない。

4）i-整合（i-consistency）

i-整合は，今まで説明した整合化手法（ノード整合、アーク整合、パス整合）を一般化したものである。

［定義4-1（i-整合）］

制約ネットワーク $\mathcal{R} = (V, D, C)$ において、$i-1$ 個の変数に対して制約を満足する値の割り当てが与えられたとき、i 個の変数間の制約をすべて満たす i 番目の変数への割り当てが存在するならば、かつそのときに限り制約ネットワークは i-整合である（i-consistent）という。

つまり、これは $i-1$ 個の任意の変数について、これらの変数間の制約をすべて満足するような変数への値の割り当てに対して、i 個の変数が制約条件をすべて満たすような i 番目の変数への値の割り当てを見つけることができるとき、かつそのときに限り、ネットワークは i-整合であることを示している。

［定義4-2（強 i-整合）］

もしネットワークが i 以下のすべての j に対して j-consistent であるならば、その時に限りネットワークは強 i-整合（strongly i-consistent）であるという。

［定義4-3（大域的整合）］

変数の数が n 個で、強n-整合なネットワークは、大域的整合（global consistent）であるという。

大域整合なネットワークは、行き止まりに出会うことなく、つまりバックトラックなしに変数への割り当てをおこない解を求めることができるという性質をもつ。また、今までに説明した整合化手法（ノード整合、アーク整合、パス整合）と強i-整合には、次のような関

係がある。

・ノード整合は強 1-整合（strong 1-consistency）と等価である。

・アーク整合は強 2-整合（strong 2-consistency）と等価である。

・パス整合は強 3-整合（strong 3-consistency）と等価である。

これまで i-整合手法について説明してきたが、$i \geq 3$ では、パス整合よりも計算に時間を要するので、実際には利用されない場合が多い。さらに、n 個のノードからなる制約ネットワーク（$i < n$）に対して i-整合の考え方を適用した場合には、すべての不整合な値を除去できないことにも注意が必要である。

以上のような基本整合化手法では、ノード整合→アーク整合→パス整合→ i-整合という順番で厳しくなっていく制約条件に対して矛盾する解候補を除去していく。すなわち、これらの整合化手法では、ノード整合からアーク整合、パス整合、さらに i-整合といくに従ってより厳しい制約条件を満足する解を求めることになる。それから、整合化手法を適用した結果、整合関係が成立しない場合には、制約を満足しない（制約に違反した）変数への値の割り当てが解候補に含まれていることを意味している。

3.3.2 離散組み合わせ問題への整合化手法の適用

以下では、整合化手法として、地図の色塗り問題に対しては、ノード整合、アーク整合およびパス整合を適用し、クロスワードパズル問題に対してはノード整合とアーク整合を適用する。

a）例題1（地図の色塗り問題）への適用

・ノード整合の適用

以下では、図 2 の地図の色塗り問題を用いてノード整合の考え方を説明する。制約ネットワークのノードに対応する変数 v_1 には、制約C_1：$v_1 \neq$ *blue* が、変数 v_2 には、制約C_2：$v_2 \neq$ *green* が、変数 v_3 には、制約C_3：$v_3 \neq$ *red* がそれぞれ与えられているとする。図11の（a）の制約ネットワークに対してノード整合の考え方を適用した結果が、図11の（b）に示される。

・アーク整合の適用

図 4 に示される制約ネットワークに対してアーク整合アルゴリズムを適用した結果を以下に示す。

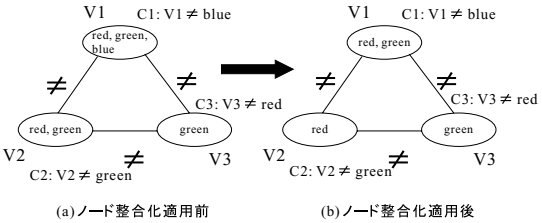

図11 地図の色塗り問題の制約ネットワークへのノード整合の適用

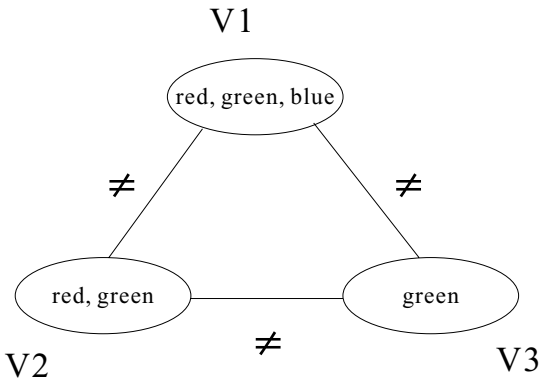

図12 アーク整合の適用（1）

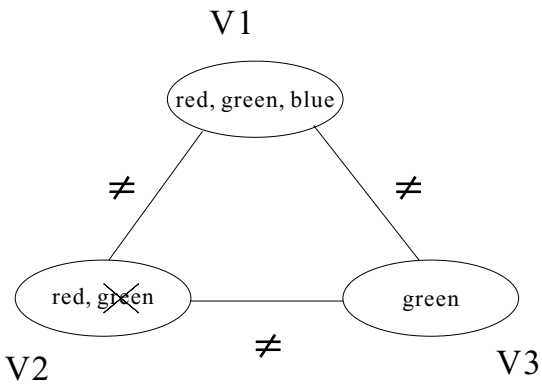

図13 アーク整合の適用（2）

図12は、図 4 のような制約ネットワークにおけるノード v_1 と v_2 間のアークに対してアーク整合を適用した結果を示している。適用結果としては、ノード v_1 と v_2 間はアーク整合なので、両者のドメインは変化していないことがわかる。

図13 は、図12の制約ネットワークに対して、さ

らにノード v_2 と v_3 間のアークに対してアーク整合を適用した結果を示している。適用結果としては、$\langle v_2, green\rangle$ に対して整合関係を満足する v_3 のドメインの値は存在しないので、v_2 のドメインから *green* が削除されていることがわかる。

以上のように、制約ネットワークの3つのアーク（v_1-v_2, v_2-v_3, v_3-v_1）のそれぞれに対して、ノードのドメインが変化しなくなるまでアーク整合を適用した結果得られた制約ネットワークが図14である。

・パス整合の適用

図4に示される制約ネットワークに対してパス整合を適用すると以下のようになる。

図15は、図4に示されるノード v_1 と v_2 間に対してパス整合を適用した結果を示している。適用結果としては、v_1 のドメイン D_1 からは *red* と *green* が、v_2 のドメイン D_2 からは *green* がそれぞれ除去されているのがわかる。

図16は、図15のノード v_2 と v_3 間に対してパス整合を適用した結果を表している。適用結果では、v_2 と v_3 間の割り当て関係は v_2 と v_1 での値の割り当て関係ならびに v_1 と v_3 ので値の割り当て関係とは無矛盾となるので、ドメインにも変化がないことわかる。

以上の操作により、図4に示される制約ネットワークに対してパス整合を適用した結果は、図16のようになり、解が存在することがわかる。

b）例題2（クロスワードパズル問題）への適用

・ノード整合の適用

図17は、図5の制約ネットワークに対してノード整合を適用した結果を示している。なお、それ

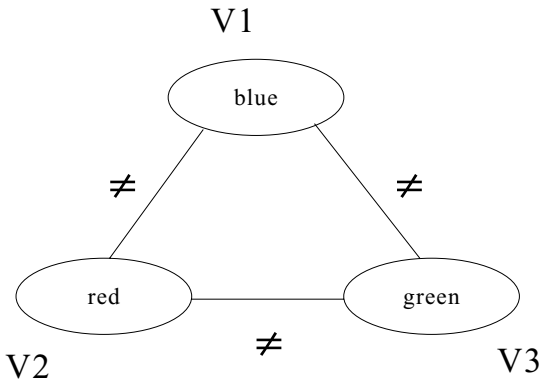

図14　アーク整合の適用（3）

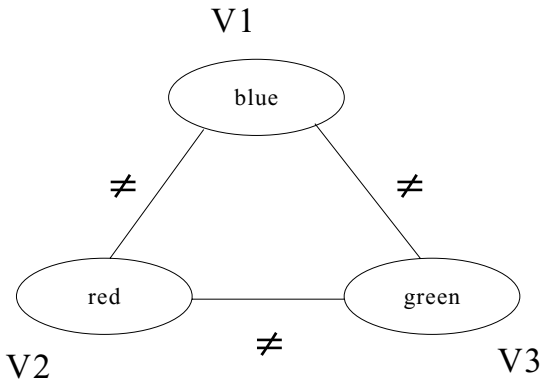

図16　パス整合アルゴリズムの適用（2）

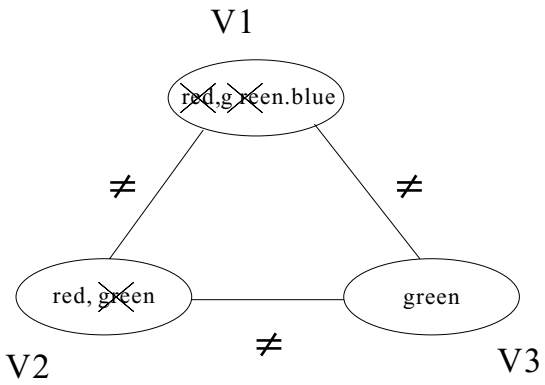

図15　パス整合アルゴリズムの適用（1）

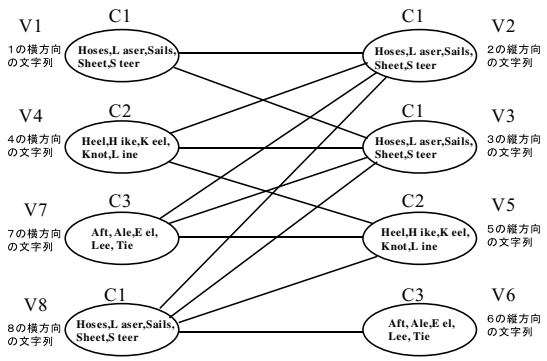

図17　ノード整合アルゴリズムの適用

ぞれのノードに関する制約 C_1, C_2, C_3 は、縦方向と横方向にそれぞれ埋められる単語の文字数を示している。

・アーク整合の適用

図18は、図17のノード v_1 と v_2 間にアーク整合を適用した結果を表している。その結果、ノード v_1 のドメイン D_1 から *Laser, Sail, Sheet, Steer* が除去されたことがわかる。さらに、v_2 のドメインD_2 からは *Hoses, Laser* が取り除かれたことがわかる。

図19は図18の結果に対して、ノード v_1 と v_3 間でアーク整合を適用した結果を表している。その結果、ノード v_3 のドメインD_3 から *Hoses, Laser* が除去されたごとがわかる。

図20は、図17のすべてのアークに対してアーク整合を適用した結果を示しており、最終的に解が得られることがわかる。

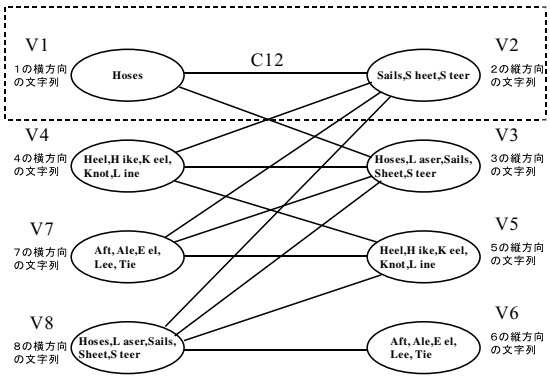

図18　アーク整合アルゴリズムの適用（1）

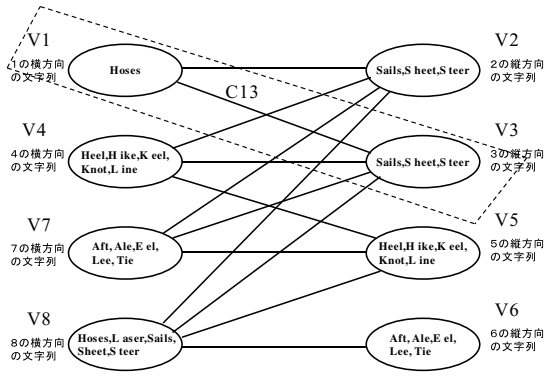

図19　アーク整合アルゴリズムの適用（2）

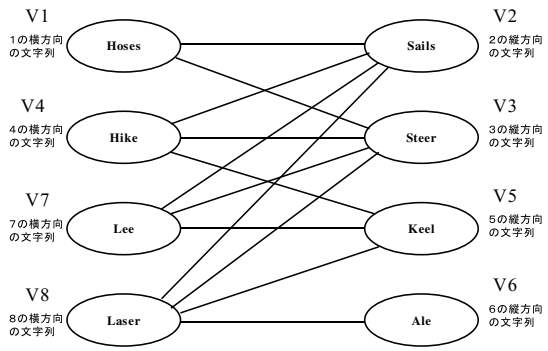

図20　アーク整合アルゴリズムの適用（3）

4　おわりに

本論文では、まず、制約充足問題による定式化に代表される制約指向アプローチのメリットについて論じた。次に、離散組み合わせ問題の代表的な問題である地図の色塗り問題とクロスワードパズル問題を取り上げて、制約充足問題の定義と定式化について説明した。そして、制約充足問題の解法のなかで、探索手法とならび代表的な手法である整合化手法を上記の組み合わせ問題に適用し、効率化手法として有効性について示した。今後は、整合化手法単独ではなく、探索手法への組み込み方式を検討していくとともに、実用的な組み合わせ問題への適用を行い、有効性を評価していく予定である。

参考文献

［1］西田豊明：人工知能の基礎、情報科学コアカリキュラム講座、丸善（1999）.

［2］石塚満：知識の表現と高速推論、情報科学コアカリキュラム講座、丸善（1996）.

［3］特集：制約充足問題の基礎と応用、人工知能学会誌、Vol.12, No.3（1997）.

［4］西原清一：整合ラベリング問題と応用、情報処理、Vol.31, No.4, pp.500-507（1990）.

［5］Edward Tsang : *Foundations of Constraint Satisfaction*, Computation in Cognitive Science, Academic Press（1993）.

［6］Ian Miguel : *Dynamic Flexible Constraint Satisfaction and its Application to AI Planning*, Distinguished Dissertations, Springer（2003）.

［7］Rina Dechter : *Constraint Processing*, Morgan Kaufmann Publishers（2003）.

［8］Stuart Russell and Peter Norvig : *Artificial Intelligence A Modern Approach, Second Edition*, Prentice Hall Series in Artificial Intelligence, Pearson Education (2003).
［9］David Pool, Alan Mackworth, and Randy Goebel : *Computational Intelligence : A Logical Approach*, Oxford Unversity Press (1998).
［10］Alan Mackworth : Constraint Satisfaction, In *Encyclopedia of Artificial Intelligence ((ed.) Shapiro, S.C.), Vol.1*, pp.205-211, John Wiley & Sons, Inc. (1987).
［11］Bernard A. Nadel : Representation Selection for Constraint Satisfaction: A Case Study Using n-Queens, In *Proc. of IEEE Expert*, pp.16-23 (1990).