Vol. 35 No. 12 情報処理学会論文誌 Dec. 1994

制約充足問題の並列化効率に基づく分類

内野寛治† 窪田信一郎†
狩野均† 西原清一†

制約充足問題（CSP）とは複数の構成要素に関する局所的解釈の候補が与えられたときに対象物全体の矛盾のない解釈を求める問題である．このような問題は人工知能の分野における線画理解などの問題，N-クイーン問題等のパズル，さらに同型部分ネットワークの探索など極めて幅広い分野に広がっている．CSP は NP 完全な探索問題の一種であり，効率の良い汎用解法は存在しない．したがって，具体的な応用問題ごとにその特徴を活かした解法を開発する必要があると予測される．本論文ではその解法の1つである併合法と呼ばれる横型探索法について考察する．併合法とは制約条件をグラフの頂点に対応させた制約ネットワークを生成し，その頂点同士の併合操作を順次進めて最終解を求める方法である．この方法の特徴として，各頂点併合操作を独立に行うことができるため処理の並列化が容易である点が挙げられる．しかし CSP の種類によっては並列処理する頂点をどのように選んでも逐次で処理した方が処理時間が少ない場合が存在する．本論文では併合法における並列処理の効率悪化の原因を明らかにするために，併合処理が3種類に分類されることを示し，分類された各処理の性質を明らかにする．さらに並列処理可能な基本的な構成である4頂点の制約ネットワークをその位相構造に着目して分類し，実験により CSP の位相構造と並列処理の効率の関係について解析する．

Classifying CSPs on the Basis of Parallelism

Kanji Uchino,† Shin'ichirou Kubota,† Hitoshi Kanoh† and Seiichi Nishihara†

This paper deals with a parallelization of the combinatorial search problem referred to as the constraint satisfaction problem (CSP). CSP is an NP-complete problem of finding all of the totally consistent interpretations. We use a breadth-first search called a merge method to sovle CSP here. In a merge method, CSP is converted into a constraint network which nodes are constraints and edges are common variables including in constraints. A merge operation is the operation to merge two nodes in constraint network, and then we can get solution when all node has merged. A merge operation can do independently, so it is able to do by parallel processing. But it does not always work effectively. So to make this ineffectiveness of parallel processing clear, by introducing the concept of semi-parallel processing, we try to classify CSP into 3 adequate processing categories, such as 'for parallel processing', 'for sequential processing' or 'for semi-parallel processing'. In semi-parallel processing, it required a temporal memory space to keep intermediate solutions, which is not needed in sequential processing. As to 4-node constraint-networks, we performed a thorough expreriments, to show that the efficiency of parallelization is considerably affected by their topological structure of the network.

1. はじめに

制約充足問題（以下 CSP）とは複数の構成要素に関る局所的解釈の候補が与えられたときに対象物全体の矛盾のない解釈を求める問題である．このような問題は，パターン処理や人工知能の分野における画像のラベリングや線画理解などの問題，N-クイーン問題等のパズル，さらに同型部分ネットワークの探索など極めて幅広い分野に広がっている[1]．

† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

CSP は NP 完全な探索問題の一種であり，いわゆる効率の良い汎用解法は存在しないことが予想される[2]．したがって，具体的な応用問題ごとにその特徴を活かした解法を開発する必要がある．その解法は，バックトラッキングを基本とした縦型探索法[3]と併合法[4]に代表される横型探索法の2種類に大別される．後者の併合法とは制約条件をグラフの頂点に対応させた制約ネットワークを生成し，その頂点同士の併合操作を順次進めて行って最終解を求める方法である．この併合法は1つの解を求めるだけではなく，すべての解を求めなければならないような CSP に対して有効にはたらく．またそれぞれの解法の前処理として，探

索空間を縮小するために行う制約伝播[5]などの補助的な技法も存在する．本論文では併合法について議論する．

併合法においては‘併合系列’[6]すなわち併合する頂点の順序によって，処理時間が大きく異なる．また，各頂点の併合操作を独立に行うことができるため処理の並列化が容易である．しかし，並列化しても必ずしも処理時間が短くなるとは限らないことが指摘されている[7]．また Kasif[8]によって CSP に制約伝播を施して弛緩するための辺整合操作は log-space 完全な手続きであり，すなわち CSP は本質的に逐次的に解かれるべき問題である，という CSP の並列処理に対する悲観的な報告がなされている．

本論文では併合法における並列処理の効率悪化の原因を明らかにするために，併合処理手順が併合系列とプロセッサの個数によって3種類に分類されることを示し，分類された各処理の性質を明らかにする．さらに並列処理可能な基本的な構成である4頂点の制約ネットワークをその位相構造に着目して分類し，実験により CSP の位相構造と並列処理の効率の関係について解析する．

以下では，まず第2章で CSP の諸定義と CSP のネットワーク表現，併合法と併合時の処理時間の絶対的な目安となる計算コストの定義を与える．第3章ではカッコ表現を用いた併合系列の表現を示し，そして併合処理と CSP の分類とそれぞれの処理の特徴について述べる．その後，頂点数4の制約ネットワークに限定して可能なすべての位相構造と併合系列を示す．第4章では第3章で提案した制約ネットワークについて，CSP のラベル制約関係を実際に計算機を使ってランダムに発生させ本論文の提案の妥当性を実験的に検証し，併せて並列処理に向く CSP の構造的特徴についても考察する．

2. CSP における併合解法と計算コスト

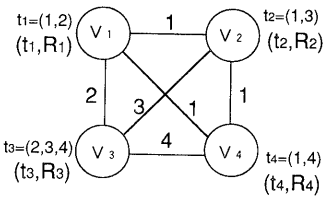
2.1 CSP の定義とネットワーク表現

CSP は4つ組 (U, L, T, R) で定義される[2]．$U=\{1,\cdots,M\}$ はユニットの集合で，各要素は問題対象の構成要素に対応する．Lはラベル集合で，各要素はユニットに与えるべき解釈や値の候補を表す．Tはユニットの多項組の集合で‘ユニット制約関係’と言う．それぞれのユニット組に対して可能な局所的解釈が $R=\{R_1,\cdots,R_{|T|}\}$ で与えられ，各 R_i を’ラベル制約関係’と言い，(t_i, R_i) を‘制約条件ペア’と言う．制約条件ペアによってあるユニット間に存在する制約を表わす．CSP を解くとは全ユニット $(1,\cdots,M)$ に対し，制約条件ペアから成るすべての制約を満たすラベル組 $(l_1,\cdots,l_M)$ を求める操作である．本論文で扱う CSP の例を**図1**に示す．

```
U = {1,...,4},
L = {a,b,c,d,e},
T = {t1,...,t4},
    t1=(1,2), t2=(1,3), t3=(2,3,4), t4=(1,4),
R = {R1,...,R4},
    R1 = {(a,b),(a,e)},
    R2 = {(a,c),(a,e),(e,b)},
    R3 = {(b,a,e),(b,c,d),(e,b,e)},
    R4 = {(a,d),(a,e),(b,e)}.

        全ユニット：(1,2,3,4)
        解　　：(a,b,c,d)
```

図1 CSP の例
Fig. 1 Example of CSP.



（辺に書かれている数字は共通ユニットを表わす）

図2 制約ネットワーク
Fig. 2 Constraint network.

次に CSP をネットワークによって等価表現する方法を示す．本論文では，制約条件ペアをネットワークの‘頂点’，2つの制約条件ペアに共通して含まれるユニット（共通ユニット）を‘辺’に対応させた‘制約ネットワーク’を考える．**図2**は図1の例を制約ネットワークを用いて表現したものである．

CSP をネットワークで表現する他の方法として，頂点をユニットに，辺をラベル制約関係に対応させる方法も考えられるがそれは2項制約までしか表現できないという欠点が存在する．しかし本論文で考察する制約ネットワークでは3項以上の多項関係も表現可能となっている[2]．

2.2 併合法と計算コスト

CSP の解法として，制約ネットワーク中の2つの頂点をまとめて新たに1つの頂点で置き換える操作，すなわち頂点併合操作を順次繰り返していき最終的に1つの頂点に縮退させる方法がある．これを併合法[5]と呼ぶ．頂点 V_i と V_j を併合する操作を本論文ではカッコ表現を用いて

$(V_i\ V_j)$

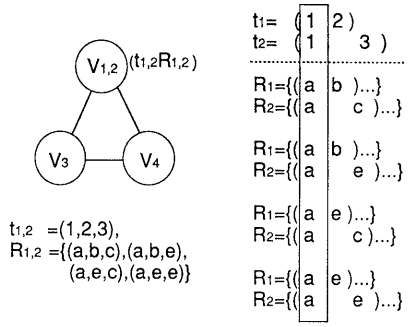


図 3 併合処理
Fig. 3 Merge operation.

と表記することとし，新たに生成される頂点を $V_{i,j}$ と表わす．図2の例の場合，V_1 と V_2 とを併合させて新たに頂点 $V_{1,2}$ を得る併合操作は $(V_1\ V_2)$ と表記され，新たに生成された頂点 $V_{1,2}$ の制約条件ペアは，$t_{1,2}=(1, 2, 3)$，$R_{2,3}=\{$(a, e, e), (a, e, c), (a, b, e), (a, b, c)$\}$ となる．以下に併合操作の過程を説明する．

step 1. 2頂点間に共通に含まれるユニット（共通ユニット）を求める

step 2. 各頂点に含まれるラベル組のすべての組み合わせを走査し，共通ユニットに対応するラベルが等しいラベル組を選ぶ

step 3. 選ばれたラベル組を自然結合する

図3の例では共通ユニットは1なので，それに対応するラベルが等しい場合，つまりラベル a のときそれを含むラベル組の自然結合が頂点 $V_{1,2}$ の新たなラベル組となる．図3は図2の制約ネットワーク中の V_1 と V_2 を併合した結果を示している．

制約ネットワークの各頂点に繰り返し併合操作を施し，最終的に1つの頂点になったときの制約条件ペアが最終解（全体解）を与える．

次に併合操作の処理時間の相対的な目安となる‘計算コスト’の定義を与える．頂点併合操作における処理時間は，step 2 での「走査しなければならない各頂点のラベル制約関係の要素の全組み合わせ数」にほぼ比例すると仮定できるので $(V_i\ V_j)$ の‘計算コスト’$C_{i,j}$ を次のように定義する．ただし $|R_i|$ は R_i 中のラベル組のサイズ（全要素数）を表わすものとする．

$$C_{i,j}=|R_i|\times|R_j|$$

その結果，前述した頂点 V_1 と V_2 の計算コストは

$$C_{1,2}=|R_1|\times|R_2|$$
$$=2\times3$$
$$=6$$

となる．以後，本論文では処理時間（効率）を計算コストを用いて議論し，実際に計算機に実装して解く場合のプロセッサ間の通信量等は全く考慮しないものとする．

上記の定義から最終解までの計算コストはそれぞれの併合操作での計算コストの和で与えられる．例えば n 個の頂点から成る制約ネットワークを

$$(V_1\ V_2), (V_{1,2}\ V_3), \cdots, (V_{123\cdots(n-1)}V_n)$$

の順に併合操作を行って最終解を得る場合の計算コストは次のように表わされる．（注：3つ以上の頂点を併合して得られる頂点，すなわち $(V_{1,2}\ V_3)$ の結果得られる頂点等は $V_{1,2,3}$ ではなく $V_{12,3}$ と表記する）

$$C_{1,2}+C_{12,3}+\cdots+C_{12\ldots(n-1),n}$$
$$(C_{12\ldots(n-1),n}=|R_{12\ldots,(n-1)}|\times|R_n|)$$

図2の例で説明すると，$(V_1\ V_2)$, $(V_{1,2}\ V_3)$ の順に併合するときの計算コストは以下のようになる．

$$C_{1,2}+C_{12,3}$$
$$=|R_1|\times|R_2|+|R_{1,2}|\times|R_3|$$
$$=2\times3+4\times3$$
$$=18$$

ここで注意しなければならないのは3つ以上の頂点を併合する場合，頂点の併合順序によって計算コストも異なるという事実である．これは併合する頂点の組み合わせにより，新しく得られる頂点の R のサイズが異なるためである．そのため3つ以上の頂点からなる制約ネットワークの最終解を求める場合，併合順序は処理時間を大きく左右する．本論文ではこの併合順序を‘併合系列’[6]と呼び，次章において詳しく考察する．

3. 併合系列と処理の分類

3.1 併合系列とその表現

図2中の頂点を $(V_1\ V_2)$, $(V_{1,2}\ V_3)$, $(V_{12,3}\ V_4)$ の順に併合処理を行うような併合系列を本論文ではカッコ表現を再帰的に用いて次のように表記する．

$$(((V_1\ V_2)V_3)V_4) \qquad (1)$$

すなわちカッコの深さのレベルが併合順序を表している．そして，このような表現を省略して

$$(V_1\ V_2\ V_3\ V_4) \qquad (1)'$$

と表記できるものとする．ここで注意したいのは $(V_i\ V_j)$ と $(V_j\ V_i)$ の結果が同じであるため(1)′のような併合系列と最初の2つの頂点の順序を入れ替えた $(V_2\ V_1\ V_3\ V_4)$ という併合系列の結果は等しいという事実である．

(1)′のような併合系列に対して $(V_1\ V_2)$, $(V_3$

V_4), ($V_{1,2}$ $V_{3,4}$) の順に併合する場合が考えられる．このような併合系列を次のように表記する．

$$((V_1\ V_2)\ (V_3\ V_4)) \qquad (2)$$

この表記は頂点の併合順序のほかに，与えられたCSPを部分問題に分割することについても表わしている．すなわち，頂点を $\{V_1, V_2\}$, $\{V_3, V_4\}$ の2グループに分け，それぞれが1つの頂点に縮退した後それらを併合することを表わす．このような系列を本論文では'分割型併合系列'と呼ぶ．逆に(1)'のような併合系列を'非分割型併合系列'と呼ぶ．

上記の2種類の併合系列の違いを計算領域の観点から述べると次のようになる．計算機を用いて非分割型併合系列を処理する場合，一連の併合過程において併合して新たに作られる頂点数は常に1であり，これは計算機中に常に一定の作業領域のみを必要とすることを意味する．これに対し分割型併合系列を処理する場合，新たに作られる頂点は同レベルの深さのカッコ表現の数だけ存在し，計算機中の作業領域もその数に比例して多く必要となる．

3.2 併合処理の分類

3.1節で示した2種類の併合系列（非分割型，分割型）を実際に計算機を用いて処理する場合の併合処理がどのように分類されるかについて考察する．併合系列を処理する処理系（アーキテクチャ）は2種類（シングルプロセッサ，マルチプロセッサ）が考えられる．つまり，2種類の併合系列と2種類の処理系によって併合処理を**表1**のように分類することができる．ここで準並列処理と呼ばれる処理を導入することにより，逐次処理向き，並列処理向きという議論をプロセッサ数に関係なく行えるようになった．

ここで表1で分類した併合処理の計算コストについて3.1節で用いた併合系列の例を使って説明する．非分割型併合系列(1)をシングルプロセッサで処理したとき（逐次処理）の計算コストを C_s，分割型併合系列(3)をシングルプロセッサで処理したとき（準並列処理）の計算コストを $C_{p'}$，(3)をマルチプロセッサ（2プロセッサ）で処理したとき（並列処理）の計算コストを C_p とするとそれぞれの処理の計算コストは次のようになる．

表1 併合処理の分類
Table 1 Classifying merge process.

併合系列 / アーキテクチャ	非分割型	分割型
シングルプロセッサ	逐次処理	準並列処理
マルチプロセッサ	—	並列処理

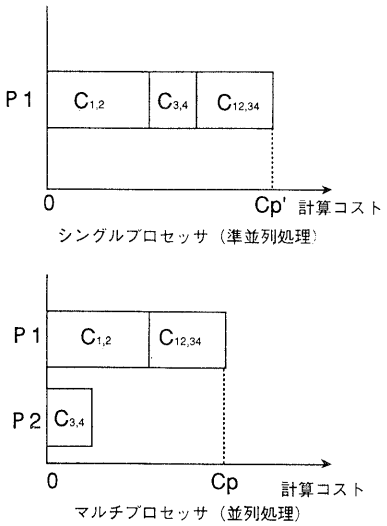


図4 並列処理と準並列処理の計算コスト
Fig. 4 Merge costs of parallel and semi-parallel processing.

$$C_s = C_{1,2} + C_{12,3} + C_{123,4}$$
$$= 6 + 12 + 3 = 21$$
$$C_{p'} = C_{1,2} + C_{3,4} + C_{12,34}$$
$$= 6 + 9 + 20 = 35$$
$$C_p = \max\{C_{1,2}, C_{3,4}\} + C_{12,34}$$
$$= \max\{6, 9\} + 20 = 9 + 20 = 29$$

シングルプロセッサにおける最終解までの計算コストは各段階の計算コストの総和の形で表わしたが，マルチプロセッサの場合は並列実行部分の計算コストをその中の最大値で代表する．つまり並列処理を行った部分で一番処理時間を要した処理に同期をとる．

例から分かるように並列処理の効果は，並列実行部分の頂点併合操作の計算コストがそのうちの最大値で代表される部分にある．そのため同じ分割型併合系列をシングルプロセッサかマルチプロセッサで処理する場合は常にマルチプロセッサの計算コストの方がシングルプロセッサでの計算コストよりも小さい．**図4**で説明すると準並列処理での計算コスト $C_{p'}$ と並列処理での計算コスト C_p の関係は常に $C_p < C_{p'}$ である．

3.3 CSPの分類

本論文では与えられたCSPを3つの処理で解き，それらの計算コストの大小を用いて本質的に逐次処理向き，並列処理向き，準並列処理向きというような分類を行う．ただし，2.2節でも述べたように併合系列によって計算コストは大きく異なるため，CSPの本

質的な性質から分類するためには最適な併合列を求める必要がある．しかし，最小の計算コストを与える併合系列を求める高速な解法は現在のところ存在しない[2]．本論文では頂点数4の制約ネットワークを扱っているので12種類の非分割型併合系列と，3種類の分割型併合系列の中から最小の計算コストを与える併合系列を求めることが可能である．以後の議論では最小の計算コストを与える併合系列の処理を最適処理と呼ぶことにする．

各併合処理の最適処理とは次のような意味である．

最適逐次処理……最適な計算コストを与える非分割型併合系列のシングルプロセッサによる処理

最適準並列処理……最適な計算コストを与える分割型併合系列のシングルプロセッサによる処理

最適並列処理……最適な計算コストを与える分割型併合系列のマルチプロセッサによる処理

ここで上記の各最適処理の計算コストをそれぞれ C^0_s, $C^0_{p'}$, C^0_p と表わすものとする．

3.3.1 単一プロセッサの場合の CSP の分類

単一プロセッサの場合は表1の分類から逐次処理と準並列処理のみ考えられる．最適逐次処理と最適準並列処理の計算コストから次のような性質が存在する．

性質(1)．$C^0_s \leqq C^0_{p'}$ となる CSP が存在する

これは本質的に部分問題に分割して解かれるべき CSP が存在することを意味している．この事実は本論文で準並列処理という分類を導入して初めて分かった結果である．このような性質の CSP は積極的に部分問題に分割して解かれるべき問題だと言える．

さらに計算コストの大小から CSP を分類すると次のようになる．ここで分割処理，非分割処理という言葉を用いているがそれぞれ準並列処理，逐次処理と同じ処理を意味している．以下に計算コストを用いた分類の枠組みを示す．

シングルプロセッサでの処理

$C^0_s > C^0_{p'}$ ……分割処理向き

$C^0_s = C^0_{p'}$ ……不定

$C^0_s < C^0_{p'}$ ……非分割処理向き

3.3.2 複数プロセッサの場合の CSP の分類

複数プロセッサの場合は並列処理のみ考えられる．最適並列処理と最適準並列処理の関係は3.2節で示したように常に $C^0_p < C^0_{p'}$ であるため，ここでは最適逐次処理と最適並列処理の議論を行う．

性質(2)．$C^0_s \leqq C^0_p$ となる CSP が存在する

これは CSP をどのように並列処理しても逐次処理で解いた方が効率が良い問題が存在することを意味している[7]．このような問題は分割せずに逐次的に解かれるべき問題である．図1の CSP はその例になっている．

以下に計算コストを用いた分類の枠組みを示す．

マルチプロセッサでの処理

$C^0_s = C^0_p$ ……並列処理向き

$C^0_s = C^0_p$ ……不定

$C^0_s < C^0_p$ ……逐次処理向き

これらの性質をもつ CSP が実際に存在することは4章の実験で確認する．

3.4 制約条件の構造

制約ネットワークの位相構造と CSP の分類の関係について実験で調べるために，ここではネットワークの構造と可能な併合系列の種類について述べる．本論文では並列処理が可能な最小構成である頂点数4の制約ネットワークについて考察する．

(1) 4頂点制約ネットワークの種類

実験に使用する4頂点の制約ネットワークの位相構造の種類について述べる．頂点数4の制約ネットワークをその位相構造で分類すると，**図5**に示す G1〜G6 の6種類となる．

本実験ではネットワークの各辺に対応している共通ユニットの個数を1に限定する．一般に制約ネットワークの位相構造が完全グラフをその内部に含む場合は共通ユニットの種類の組み合わせによって複数の場

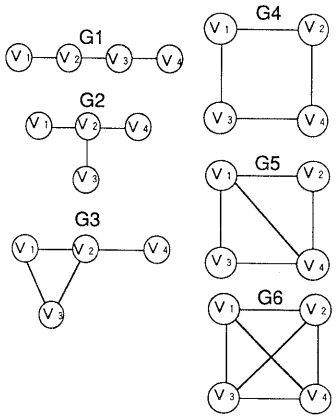
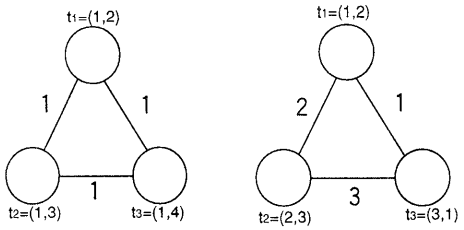


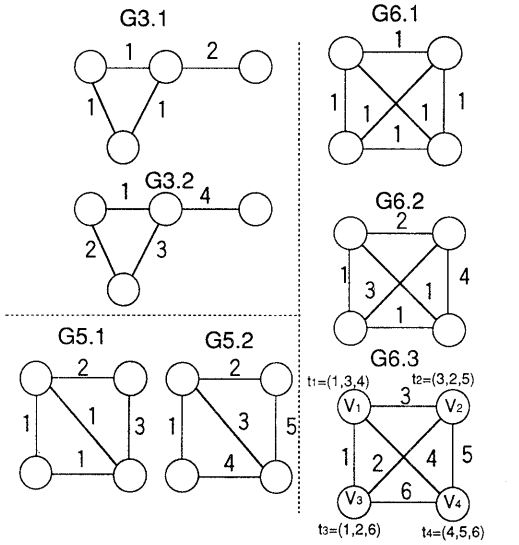
図5 制約ネットワークの位相構造の種類（頂点数=4）
Fig. 5 Topological structures of constraint network (nodes=4).



共通ユニットの種類＝1　　共通ユニットの種類＝3

（各辺に書かれた数字は共通ユニットを表わす）

図 6 K3の2種類の表現
Fig. 6 Expressions of K3.



（具体的な制約ネットワークの例）

（各辺に書かれた数字は共通ユニットを表わす）

図 7 位相以外の構造による分類
Fig. 7 Non-topological structures.

合が存在する．最小の完全グラフの構成である頂点数が3の完全グラフ（K3）では，辺に対応している共通ユニットがすべて等しい場合とすべて異なる場合の2通りが考えられる（本質的にこの2通り以外は考えられない）．（**図6**参照）

ここで図5の制約ネットワークG3, G5, G6はその内部にK3の構造を持つため，K3部分の可能な共通ユニットの組み合わせによって2種類の構造が考えられる．特にG6はそれ自身も完全グラフ構造（K4）になっているため，すべての辺に対応する共通ユニットがすべて等しい場合が加わり，合わせて3種類が考えられる．その典型的な例を**図7**に示す．

非分割型併合系列の種類
（V1 V2 V3 V4）（V1 V2 V4 V3）
（V1 V3 V2 V4）（V1 V3 V4 V2）
（V1 V4 V2 V3）（V1 V4 V3 V2）
（V2 V3 V1 V4）（V2 V3 V4 V1）
（V2 V4 V1 V3）（V2 V4 V3 V1）
（V3 V4 V1 V2）（V3 V4 V2 V1）

分割型併合系列の種類
（（V1 V2）（V3 V4））（（V1 V3）（V2 V4））
（（V1 V4）（V2 V3））

図 8 頂点数4の併合系列の種類
Fig. 8 Merge series of 4-node constraint network.

（2） 頂点数4の可能な併合系列の種類

頂点数4の制約ネットワークの場合，**図8**に示すように非分割型併合系列は12種類，分割型併合系列は3種類の組み合わせが考えられる．ここでは各CSPに対してこれら15種類の併合系列をすべて実施し，最適な計算コストを与える併合系列が分割型か非分割型かによってそのCSPを分類することとした．なお，与えられたCSPに対して最適な計算コストを与える併合系列を決定することは応用上重要な問題である[5]がここでは扱わない．

また，ネットワークによって15種類の併合系列の中には辺によって連結されていない頂点（非隣接頂点）の併合も含まれているが，4.3節（2）の考察でも述べるようにこのような併合は最適とはならないので実験結果には影響しない．

4. 実　　験

4.1 実験方法

ここでは計算機を用いてランダムに生成したCSPに対して3章で示した分類法に従ってCSPを分類し，位相構造によって最適な併合処理がどのように変化するかを調べる．

CSPの生成はG1〜G6.3に対応するユニット制約関係T，ラベル総数$|L|$，各ラベル制約関係の要素数$|R|$を与えその条件にあった具体的なラベル組をランダムに生成する方法で行った．$|L|$が大きいと併合時に対応ラベルが一致する可能性が低くなり，中間解の個数は抑制される．逆に$|R|$が大きいと対応ラベルが一致する可能性が高くなり，中間解の個数は増大する傾向にある．なお位相構造の影響のみを調べるために，ラベル制約関係の要素数$|R|$はすべて等しくする．

そして生成されたCSPに対して，実験1では3.4節の（2）で示した非分割型12種類と分割型3種類の併合系列を用いて解き，2つの型の最小計算コストを

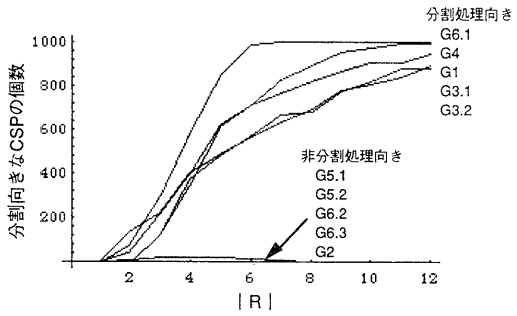


図 9 分割/非分割の分類結果（実験１）
Fig. 9 Result of divide/non-divide classification (exp. 1).

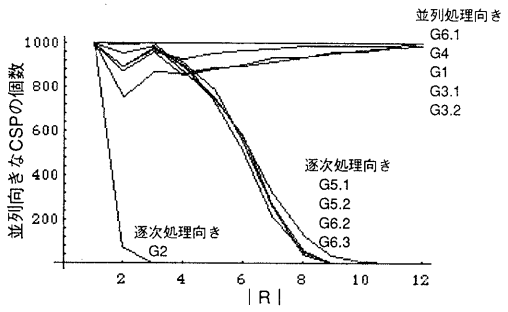


図 10 並列/逐次の分類結果（実験２）
Fig. 10 Result of parallel/sequential classification (exp. 2).

比較して分割型/非分割型の分類を行う．これをランダムに生成した 1000 個の CSP に対して調べる．実験２では分割型併合系列の計算コストを算出する過程で 3.2 節で説明したように分割部分を最大値で代表することにより並列処理の計算コストを求め並列型/逐次型の分類を行う．実験の詳細を以下に示す．

［実験１］ 分割型/非分割型の分類

$|L|=5$, $|R|=1$〜12 の条件で CSP を 1000 個生成し分割型/非分割型/不定の３種類に分類する．その結果，G1〜G6.3 について分割向きと分類された CSP の個数を**図 9** のグラフに表わす．（縦軸＝分割向きな CSP の個数，横軸＝$|R|$）

［実験２］ 並列型/逐次型の分類

$|L|$ と $|R|$ は実験１と同条件，CSP を 1000 個生成し並列型/逐次型/不定の３種類に分類する．その結果，G1〜G6.3 について並列向きと分類された CSP の個数を**図 10** のグラフに表わす．（縦軸＝並列向きな CSP の個数，横軸＝$|R|$）

4.2 実験結果と考察

（1） 実 験 結 果

図９から $|R|>2$ のところで G1〜G6.3 の各構造を持つ CSP において分割/非分割処理向きの分類が明確に行えることがわかる．さらに図 10 から $|R|>5$ のところで並列/逐次処理向きの分類が明確に行えることがわかる．これは G2, 5.1, 5.2, 6.2, 6.3 のような構造を持つ CSP は積極的に逐次処理して解くべきであることを表わしている．また，図９，10 で示した CSP の分類結果はすべて等しくなっている．

（2） 考 察

以上の実験結果から並列処理向きと分類される CSP には制約ネットワークの構造上次のような特徴があることがわかる．

（1） ネットワークを分割する際に切断しなければならない辺の数が２本以下でかつ分割した部分も連結グラフとなっている（G1, 3, 4）

（2） 辺に対応しているユニット（共通ユニット）の種類が１種類である（G6.1：付録A参照）

まず（1）について説明する．制約の分断を意味する辺の切断数が多いほど中間解の個数は大きくなり，その結果計算コストが増大する．G2 のように切断しなければならない辺の数が２本以下であっても分割後の部分グラフの一方が必ず非連結となると非隣接頂点の併合を行わなければならない．非隣接頂点間の併合の場合は２頂点の $|R|$ の積の大きさの中間解が生成されてしまうので計算コストの増大を引き起こす．

（2）について G6.1 と G6.2, G6.3 を比較する．G6.2, G6.3 のように共通ユニットが２種類以上で構成されているネットワークでは併合の過程で１本の辺に対応する共通ユニットの種類が多くなっていき，対応するラベル組が一致する確率が少なくなる．その結果，中間解をより絞り込むことになる．一方 G6.1 のようなネットワークにおいては，頂点を併合していく過程で対応する共通ユニットの種類は１のままで対応するラベル組が一致する確率は G6.2, G6.3 と比較して多く，中間解の絞り込みはあまり行われない．付録において G6.1 の具体的なネットワークを示し，その併合過程における計算コストを用いて並列（分割）処理の有効性を説明する．なお，併合処理を効率的に行うために共通ユニットの種類の多い辺を優先的に併合するという方法は以前に提案している[5]．

さらに実験１と実験２の CSP の分類結果が等しくなっているという結果は並列処理向きかどうかを判断

するのに分割処理向きかどうかを判断すればよいことを意味している．つまり分割処理をして効率が悪化する（非分割処理向き）CSP はマルチプロセッサをどのように使って並列処理を試みても高速化は期待できないことがわかる．

5. おわりに

本論文では CSP の並列処理の効率悪化の原因を考察するために CSP の分類を試みた．その準備として併合系列の表現であるカッコ表現を提案し，それによって併合系列が2種類の系列に分類されることを示した．さらにそれら2種類の併合系列の処理としてシングルプロセッサとマルチプロセッサの場合が考えられるので，それぞれの組み合わせによって併合処理そのものが逐次処理，準並列処理，並列処理の3種類に分類されることを示した．そして，分割処理向き/非分割処理向き，並列処理向き/逐次処理向きという2つの枠組で CSP を分類する方法を提案した．

また，制約条件の構造が各併合処理に及ぼす影響を実験を行って調べた．その結果，辺に対応する共通ユニットの種類が1種類のネットワークや分割時に切断しなければならない辺数が2本以下であるような構造を持つネットワークは分割処理（並列処理）に向く CSP であることがわかった．さらに，分割処理をして効率が悪化する CSP は並列処理の効果は期待できないことがわかった．

これらの結果は4頂点の制約ネットワークとなる CSP について得られたものであるが，4頂点のネットワークは並列処理を行う上で基本的な位相構造であるため，4頂点以上の多頂点から成る複雑なネットワークに対しても実験で考察した構造上の特徴を利用できると推測される．すなわち，ネットワークを構成する部分グラフの位置関係が4頂点ネットワークの位相構造に対応している場合や，部分グラフが4頂点のネットワークになっている場合である．

本論文で得られた結果は CSP の並列解法について Kasif[8]のように悲観的な結論を導くものではない．むしろネットワークの構造に注目して適当な部分，すなわち分割時に切断しなければならない辺の数が2本以下で分割後の部分グラフが連結グラフになっている部分で分割を行うならば積極的に並列化を試みるべきだといえる．

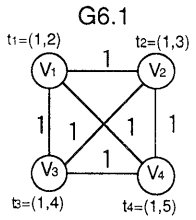
今後の課題としては，5頂点以上の複雑なネットワークに対し本論文の実験で考察した指針が適用可能かどうかの検証と大規模なネットワークを分割する場合の適当な指針を求めることが考えられる．

参考文献

1) Haralick, R. M. and Shapiro, L. G.: The Consistent Labeling Problem, Part I, *IEEE Trans. Pattern Anal. Machine Intel.*, Vol. PAMI-1, No. 2, pp. 173-184 (1979).
2) 西原清一：制約充足問題の高速解法，知識ベースシステムにおける高速推論技術チュートリアル資料，情報処理学会 (1992).
3) Knuth, D. E.: Estimating the Efficiency of Backtrack Procedure, *Mathematics of Computation*, Vol. 29, pp. 121-136 (1975).
4) 塩沢恒道，西原清一，池田克夫：拘束条件の構造を考慮した整合ラベリング問題の解法，情報処理学会論文誌，Vol. 27, No. 10, pp. 927-935 (1986).
5) Mackworth, A. K.: Consistency in Newwork of Relations, *Artif. Intell.*, Vol. 8, pp. 99-118 (1977).
6) 西原清一，松尾嘉和：概整合ラベリング問題における併合法の最適化と効率評価，人工知能学会誌，Vol. 3, No. 2, pp. 196-205 (1988).
7) 西原清一，松尾嘉和：整合ラベリング問題における併合解法の並列化について，人工知能学会誌，Vol. 6, No. 1, pp. 124-128 (1991).
8) Kasif, S.: On the Parallel Complexity of Discrete Relaxation in Constraint Satisfaction Networks, *Artif. Intell.*, Vol. 45, pp. 275-286 (1990).

付録 G6.1 の処理の具体例

G6.1 の具体例として**図11**(a)に示すようなネットワークを考える．このネットワークは本実験で $|R|=8$, $|L|=5$ に設定して発生させたものの1つである．この制約ネットワークを 15 種類の併合系列で処理したときの計算コストを図11(b)に示す．このネットワークの場合，非分割，分割型併合系列の最適な併合系列と計算コストは $(V_1\ V_4\ V_2\ V_3)$, 448, $((V_1\ V_2)\ (V_3\ V_4))$, 394 である．上記の併合系列の処理過程を計算コストを用いて図11(c)に示す．$C^0{}_s=448$, $C^0{}_{p'}=394$ ($C^0{}_p=330$) となり準並列処理（並列処理）の計算コストが逐次処理の計算コストより低く抑えられ効率的であることがわかる．$C^0{}_s$ と $C^0{}_{p'}$ の各項を比較してみると第一項は問題として最初から与えられている頂点間の併合で共に等しいが第二項，第三項で差が開いていることがわかる．すなわち G6.1 のようなネットワークは中間解が急激に増大するようなネットワー



R1={(b a)(d c)(b b)(e d)(d a)(a e)(d b)(d d)}
R2={(b a)(d c)(b c)(e a)(d d)(d b)(a a)(a a)}
R3={(b b)(d b)(b d)(e e)(d a)(d d)(d e)(a a)}
R4={(b a)(d a)(e a)(a a)(d c)(a c)(c d)(e e)}

(a) G6.1の具体的な制約ネットワークの例

非分割型

併合系列	計算コスト
(v1 v2 v3 v4)	688
(v1 v2 v4 v3)	488
(v1 v3 v2 v4)	712
(v1 v3 v4 v2)	560
(v1 v4 v2 v3)	448
(v1 v4 v3 v2)	496
(v2 v3 v1 v4)	688
(v2 v3 v4 v1)	488
(v2 v4 v1 v3)	448
(v2 v4 v3 v1)	448
(v3 v4 v1 v2)	496
(v3 v4 v2 v1)	448

分割型

併合系列	計算コスト
((v1 v2) (v3 v4))	394
((v1 v3) (v2 v4))	436
((v1 v4) (v3 v2))	394

（下線は最適な計算コスト）

(b) 併合系列と計算コスト

最適逐次処理

$$C^{\circ}s = C_{1,4} + C_{14,2} + C_{142,3}$$
$$= |R_1| \times |R_4| + |R_{1,4}| \times |R_2| + |R_{14,2}| \times |R_3|$$
$$= 8\times 8 + 14\times 8 + 34 \times 8$$
$$= 64 + 112 + 272$$
$$= 448$$

最適準並列処理

$$C^{\circ}p' = C_{1,2} + C_{3,4} + C_{12,34}$$
$$= |R_1| \times |R_2| + |R_3| \times |R_4| + |R_{1,2}| \times |R_{3,4}|$$
$$= 8\times 8 + 8 \times 8 + 14 \times 19$$
$$= 64 + 64 + 266$$
$$= 394$$

並列処理

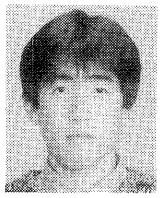
$$C^{\circ}p = \max\{C_{1,2}, C_{3,4}\} + C_{12,34}$$
$$= \max\{|R_1| \times |R_2|, |R_3| \times |R_4|\} + |R_{1,2}| \times |R_{3,4}|$$
$$= \max\{8\times 8, 8 \times 8\} + 14 \times 19$$
$$= \max\{64, 64\} + 266$$
$$= 64 + 266$$
$$= 330$$

(c) 各処理の計算コストの内訳

図 11 G6.1の処理の具体例
Fig. 11 Concrete example of merge process of G6.1.

クであり，併合のすべての段階で中間解を保持している逐次処理に比べて，問題として与えられている頂点との併合を多く行う準並列，並列処理の方が計算コストを低く抑えられる傾向にある．

（平成6年4月11日受付）
（平成6年9月 6 日採録）



内野 寛治（正会員）

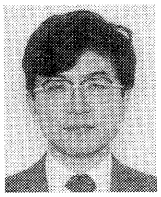
昭和43年生．平成5年筑波大学大学院理工学研究科修士課程修了．現在同大大学院工学研究科博士課程在学中．人工知能，制約充足問題の研究に従事．

窪田信一郎（正会員）

昭和 45 年生．平成6年筑波大学大学院工学研究科修士課程修了．現在(株)日立製作所勤務．人工知能，制約充足問題の研究に従事．

狩野 均（正会員）

昭和29年生．昭和55年筑波大学大学院理工学研究科物理学専攻修士課程修了．同年日立電線(株)入社．同社オプトロシステム研究所において人工知能・神経回路の応用に関する研究に従事．平成5年4月より筑波大学電子情報工学系．現在，パターン解析・知識処理の研究に従事．工学博士．平成4年電気学会論文賞受賞．人工知能学会，電気学会各会員．



西原 清一（正会員）

昭和21年生．昭和43年京都大学工学部数理工学科卒業．同年同大学大型計算機センター助手．昭和50年より筑波大学電子・情報工学系．現在，同教授．工学博士．昭和57～58年文部省在外研究員として米国ヴァージニア工科大学．図形画像処理，データ構造，組み合わせ探索，制約充足問題の研究に従事．著書に「データ構造」(オーム社) など．電子情報通信学会，ACM，IEEE 各会員．