特　集　「制約充足問題の基礎と応用」

CSPの新しい展開：分散／動的／不完全CSP

New Directions of CSP: Distributed/Dynamic/Partial CSP

横尾　真*1　平山　勝敏*2
Makoto Yokoo　Katsutoshi Hirayama

*1 NTTコミュニケーション科学研究所
NTT Communication Science Laboratories.
*2 神戸商船大学
Kobe University of Mercantile Marine.

1997年2月17日　受理

Keywords: constraint satisfaction problem, distributed CSP, dynamic CSP, partial CSP, multiagent system, optimization.

1. まえがき

制約充足問題（Constraint Satisfaction Problem, CSP）は人工知能における様々な問題を表現できる一般的な枠組であり，様々な応用問題を表現可能であるが，より現実的な状況を表現しようとする場合に，いくつかの制限がある．

例えば，通常のCSPの定式化においては，問題に関する知識は単一の問題解決器（problem solver）が持ち，この問題解決器がCSPを解くことを前提としている．また，与えられた問題は時間的に変化しないことを仮定している．一方，現実的な問題では問題解決器と外界とのインタラクションがあるのが通例であり，閉じた世界で与えられた問題を解くというより，開いた世界で外界／他の問題解決器とインタラクションしながら問題を解いていき，かつ，インタラクションの過程で解くべき問題も動的に変化していく可能性がある．このような外界／他の問題解決器とインタラクションしながら問題解決する主体を示すためにエージェント（agent）という用語が用いられることが多い．

また，通常のCSPでは，目標は与えられた制約を完全に満足する解を求めることであり，満足されない制約が一つでも存在すれば，それは解とはならない．ただし，このように制約を満足するか否かで割り切って区分をしていること自体は欠点であるとは言えない．CSPよりも一般的なクラスの問題として最適化問題が存在するが，一般的に最適化問題で最適解を得ることは難しく，最適ではないが許容できる解を求めるという方向は，ある意味でAIの基本的な方針である．CSPの研究が最適化問題とは違った方向での発展を遂げているのは，このような単純化した仮定のためであるといっても過言ではない．しかしながら，現実の問題をCSPとして表現しようとする場合に，問題を定義することが難しいということがよく言われる．すなわち，何が本質的な制約で，何が付加的で無視しても構わない制約かを問題のデザイナが判断する必要がある．現実の問題をCSPとして表現する際に人手によって行なわれている，考えられる無数の制約から取捨選択を行ない適切な問題を切り出すという作業を，計算機に行なわせたいというのは自然な要求であろう．

本稿では以下，このような仮定を緩和して通常のCSPの枠組を拡張した，複数のエージェントの関連するCSPである分散CSP（2章），問題の変化を導入した動的CSP（3章），与えられた制約を部分的に満足する解を求める不完全CSP（4章）に関して解説を行う．以下，各章では，第1節に形式的な問題の定義，第2節で問題の具体例を挙げ，第3節で問題を解くためのアルゴリズムについて解説するという構成をとる．

2. 分散CSP

2・1 問題の定式化

分散CSPとは，CSPの変数と制約が複数のエージェントに分散された問題と見なせる．各エージェントは自分の持つ変数の値を決定しようとするが，異なるエー

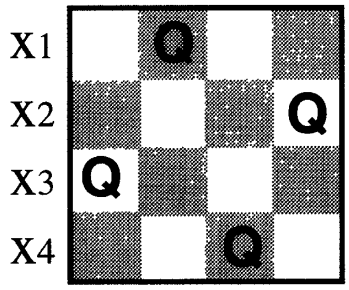


図1　CSP の例（4-queens 問題）

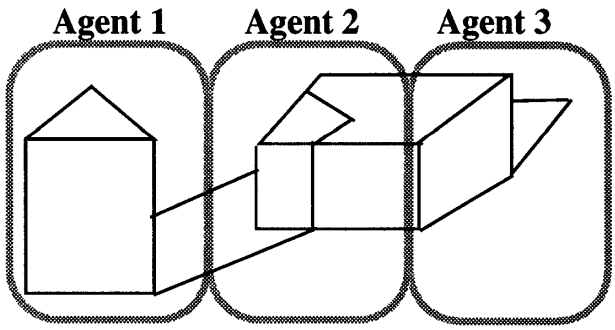


図2　分散 CSP の例（線画の解釈問題）

ジェントの持つ変数間に制約があるため，互いに値の割当を通信し整合をとる必要がある．すべてのエージェントの変数への値の割当がすべての制約を満たすことが分散 CSP の目標となる．

形式的には，エージェントの集合 $1,2,\ldots,m$ が存在し，各変数 x_j に対して，その属するエージェント i が定義される（belongs(x_j,i) と書く）．制約に関する情報も同様にエージェント間に分散される．エージェント i が制約 C_l を知っていることを known(C_l,i) と書く．

次の場合に，分散 CSP が解けたという．

- すべてのエージェント i において，
  $\forall x_j$ belongs(x_j,i) について，x_j の値が d_j に決定される．また，$\forall C_l$ known(C_l,i) なる制約が，$x_1 = d_1, x_2 = d_2, \ldots, x_n = d_n$ のもとで真となる．

図1は，CSP の例としてよく用いられる n-queens 問題で $n = 4$ とおいた場合であるが，各クイーンに対応する独立なエージェントが存在し，それらのエージェントが自分のクイーンの位置を決定しようとしていると考えれば，この問題は分散 CSP として定式化される．

分散 CSP を解くアルゴリズムには，通常の CSP の場合と同様，なるべく少ない探索量で，より速く解を発見することが要求される．さらに，分散 CSP では，問題に関する知識が複数のエージェントに分散され，各エージェントは全体の問題に関する部分的な知識しか持っていない．このため，各エージェントがどのような手順で，どのような情報を交換し合えば，全体として制約を満たす解が発見できるかが問題となる．

2・2　例題／応用問題

解釈型の問題 [Lesser 83] とは，各エージェントがセンサ情報の一部分を担当し，エージェント全体としてセンサデータの矛盾のない解釈を構築する問題である．入力データの可能な解釈の候補を変数の取り得る値として表すことができれば，解釈型の問題は分散 CSP として表現できる．例えば，線画を解釈する問題では，各頂点の可能な解釈方法は有限であり，線分で結ばれた頂点の解釈の間には制約があるため，CSP として表現できることが知られている [Waltz 75]．図2に示すように，複数のエージェントが線画の一部分を担当して解釈を行う問題は，分散 CSP としてマッピング可能である．

一方，割当型の問題とは，複数のエージェントにタスク，資源を割り当てる問題である．このような問題でエージェント間の制約が存在する場合には，各タスク，資源を変数，その可能な割当方法を変数の値とおくことにより，分散 CSP として定式化が可能である．例えば，マルチステージネゴシエーションプロトコル [Conry 91] は，1つのゴールの達成方法に複数の可能性（プラン）がある場合，ある時刻で存在する複数のゴールのすべてを達成可能とするようなプランの組合せを求めることを目的としている．この問題は，ゴールを変数，プランを変数の取り得る値として表現することにより分散 CSP として表現可能である．

2・3　アルゴリズム

〔1〕 集中型／同期型のアルゴリズム

分散 CSP を解く極めて単純な方法として以下のような手順が考えられる．すなわち，あるリーダとなるエージェントを選択し，そのエージェントにすべての情報を集中することにより，通常の制約充足アルゴリズムを適用する．しかしながらこの方法では，リーダの選択/情報を集中するための通信量，リーダエージェントの処理の負荷等が問題となる．また，セキュリティ等の理由から，各エージェントが持つ情報をすべて明らかにし，他のエージェントに通信することが許されない場合も存在する．

また，別の方法として，通常の CSP を解くバックトラック型の探索アルゴリズムを，複数のエージェン

トでシミュレートすることが考えられる．具体的には，各エージェントには動作する順序があらかじめ決められており，最初のエージェントが自分の変数の値を一時的に決定し，その値を次のエージェントに送信する．値が送信されたエージェントは，送信された値と制約を満たすように自分の値を決定し，その結果（部分解）を次のエージェントに送信する．もし，送信された部分解と制約を満たす値が存在しない場合には，1つ前のエージェントに対して値の変更要求（バックトラック）メッセージを送信する．この解法では，エージェントが逐次的な順序で部分解を通信することにより，エージェントの持つ部分的な知識が統合されて制約を満たす解が得られる．

この方法の明らかな欠点は，処理の並列性が生かせない，すなわちある瞬間には1つのエージェントしか動作しないこと，また，あらかじめ動作の順序を決定しておかなければならないことである．文献[Collin 91]では，エージェント間の制約が木構造で表現される場合に，独立な部分に関しては並列に処理を行うバックトラッキングアルゴリズムが提案されている．

〔2〕 **非同期バックトラッキング**

非同期バックトラッキングアルゴリズム[Yokoo 92]では，同期型のバックトラッキングアルゴリズムとは対照的に，エージェントは非同期，並行に，各自の局所的な情報に基づいて動作し，新しい制約条件（nogood）を互いに通信し合うことにより制約を満足する値の組合せを得る．各エージェントは並行して自分の持つ変数の値を一時的に決定し，その値を関連するエージェントに送信する．送信された値と制約条件違反が生じる場合には，エージェント間の優先順位を用いて，優先順位の低いエージェントから値の変更が行われる．優先順位の低いエージェントにおいて，優先順位の高いエージェントの変数の値と制約を満たす値が存在しない場合には，逐次的なバックトラックが行われるのではなく，新しい制約条件（nogood）を導出し，優先順位の高いエージェントに送信することにより，非同期，並行的なバックトラックが実現される．このアルゴリズムは完全性が証明されている（解が存在するなら必ず解を発見し，解が存在しない場合は，解が存在しないことを発見して有限時間で終了する）．

〔3〕 **非同期弱コミットメント探索アルゴリズム**

非同期バックトラッキングでは，優先順位の高いエージェントが解になり得ないような値の選択を行なった場合，nogoodにより優先順位の低いエージェントからその誤りを指摘されるまで，その値に強くコミットする．その間，優先順位の低いエージェントはnogoodを導出するための網羅的な探索を行なう必要があり，優先順位の高いエージェントの値の選択が不適切な場合に効率が著しく悪くなる．

非同期弱コミットメント探索アルゴリズム[Yokoo 95]は，非同期バックトラッキングを改良したアルゴリズムであり，その主な特徴は，特定のエージェントでの値の選択の誤りが探索の効率に致命的な影響を与えないよう，探索の途中でエージェント間の優先順位を動的に変更することである．具体的には，あるエージェントにおいて，優先順位の高いエージェントとの制約を満たす値がないとき，nogoodを導出して送信すると同時に，自分の優先順位を近傍（同じ制約を共有するエージェントの集合）よりも大きくする．また，もう一つの特徴は，変数の値選択において制約条件違反最少化ヒューリスティックス[Minton 92]を採用していることである．この2つにより非同期バックトラッキングと比較して効率が劇的に改善される．また，アルゴリズムの完全性も保証される．

〔4〕 **山登り法＋提携形成**

非同期バックトラッキングや非同期弱コミットメント探索アルゴリズムなどの非同期型のアルゴリズムでは，全エージェントが非同期並行に動作するため，エージェントがある意図をもって値を変更したとしても，必ずしも意図した効果が得られるとは限らない．もし，このようなことが頻発すれば効率に悪影響があると予想される．

これに対し，[Hirayama 95]では，局所同期型アルゴリズムである山登り法＋提携形成が提案されている．このアルゴリズムでは，エージェントが自分の制約条件違反数を減らすように変数の値を変更するが，実際に変更する前に近傍と交渉を行なって値変更動作を相互排除する．これにより，動作の並列性をある程度残しながら，エージェントが意図した効果を確実に得ることができる．さらに，このアルゴリズムではエージェントが局所解（疑似局所解）に陥ったときに，そこから脱出する手続きとして，近傍のエージェント間で問題に関する情報を集中し，集団として行動を決定する提携形成という手法がとられている．これによりアルゴリズムの完全性が保証される．

〔5〕 **分散ブレイクアウトアルゴリズム**

山登り法＋提携形成は，提携形成のコストが提携サイズとともに指数的に増加するため，提携サイズが大きくなると非常に効率が悪くなる．分散ブレイクアウトアルゴリズム[Yokoo 96]は，CSPにおけるブレイクアウト法[Morris 93]にヒントを得て，山登り法＋提携形成を改良した局所同期型アルゴリズムであり，疑

似局所解からの新しい脱出法を採用していることが特徴である．このアルゴリズムでは制約に重みが導入されていて，エージェントは違反する制約の重み付き和を減らすように変数の値を変更するが，疑似局所解に陥った場合に違反している制約の重みを上げる．これにより効率良く疑似局所解から脱出できる．また，従来のアルゴリズムでは，解を発見した場合のアルゴリズムの終了判定に大域スナップショット [Chandy 85] などの他のアルゴリズムを起動する必要があったが，分散ブレイクアウトではその必要はなく，単独で終了判定可能である．ただし，このアルゴリズムは効率が良い反面，アルゴリズムの完全性が保証されないという欠点をもつ．

〔6〕 **分散 consistency アルゴリズム**

consistency アルゴリズム [Mackworth 92] は探索に先立つ前処理であり，問題で陽に与えられた制約から間接的に導かれる新しい制約をあらかじめ導き出しておくことにより，最終的な解になり得ないような変数の値を取り除き，無駄なバックトラックを避けようとするものである．文献 [Yokoo 90] で提案されている分散 consistency アルゴリズムでは，各エージェントは仮説に基づくデータ整合性管理システム（Assumption-based Truth Maintenance System，ATMS）[de Kleer 89] を用いて，あらかじめすべての取り得る値に関して nogood を生成し，これらの nogood を交換し，さらに新しい nogood を生成することにより，無駄な探索を排除している．文献 [Prosser 92] で示されている分散 consistency アルゴリズムは，新しく導かれた制約条件と，その前提となる変数の領域等との依存関係を管理しており，変数の領域の制約条件の変化に対応することが可能となっている（3 章の動的 CSP に対応する）．文献 [Conry 91] で提案されているマルチステージネゴシエーションプロトコルでは，分散 consistency アルゴリズムと同様，nogood を交換し合うことにより解となり得ないプランを排除している．

3. 動　的　CSP

3・1 問題の定式化

動的 CSP は，形式的には通常の CSP の列 $\mathcal{P}_{(0)}$, $\mathcal{P}_{(1)}$,... として記述される．$\mathcal{P}_{(i)}$ は通常の CSP であり，$\mathcal{P}_{(i-1)}$ に対して，新しい制約を追加する（restriction），もしくは制約を取り除く（relaxation）といった変更を加えることにより得られる．これらの変化は外界から与えられるものであり，ユーザからの入力，他のエージェントの動作，外乱等に起因する．

動的 CSP の目標は，基本的には，各 CSP の解をなるべく少ない探索量で，より速く解を発見することである．また，ある種のアプリケーションでは，$\mathcal{P}_{(i)}$ の解として，$\mathcal{P}_{(i-1)}$ の解との違いがなるべく少ないものが望まれる．例えば，設計問題をユーザがインタラクティブに解いている場合，制約をわずかに変更しただけで，以前のものと大きく異なる解が得られることは望ましくない．この条件は解の安定性（stability）と呼ばれる．

3・2 例題／応用問題

スケジューリング問題とは，複数のジョブ／タスクに対して資源を割り当てる問題であり，資源割当問題の一種であるが，時間が関係することが特徴的である．すなわち，各ジョブに対しては開始時間，終了時間等の時間的な制約がある．また，工場での機械等の資源は再利用可能であり，使用する時間が重ならなければ複数のジョブで共有可能である．スケジューリング問題を CSP として定式化する試みが [Prosser 89, Zweben 89] 等でなされている．

以下に非常に簡単な例を示す．A,B,C,D の 4 つのジョブがあり，それぞれ共通の資源を一単位時間使用する．この共通の資源は同じ時刻には一つのジョブのみが使用可能である．各ジョブに関して，実行可能な時刻は図 3 のとおりであり，例えばジョブ A は時刻 2，もしくは時刻 3 のいずれかに実行可能である．

この例題は，単純にジョブを変数，実行可能な時刻を変数の取り得る値として CSP として定式化可能である．スケジューリング問題における制約の変化は，新しいジョブの追加／取消，資源の増加／減少，デッドラインの延期／繰り上げ等によって生じ得る．文献 [Verfaillie 94] では，このような制約の変化が存在するスケジューリング問題の実例として，French Space Agency での探査衛星の遠隔操作のスケジューリング問題を挙げている．この問題では，新しい探査目標の追加，以前の探査の実行の結果等により，日々新しいジョブが加わり，そのつどスケジュールをし直す必要が生じる．その際に，以前のスケジュール予定をなる

Job	Domain
A	2, 3
B	1, 3, 5
C	2, 4
D	1, 2, 4

図 3　スケジューリング問題の例

べく変更しないで新しいジョブを実行可能にするスケジュールを得ることが必要である．

3・3 アルゴリズム

動的CSPは，単なるCSPの列に過ぎないので，基本的には任意のCSPを解くアルゴリズムは動的CSPを解くために利用可能である．すなわち，各CSPを，通常のCSPを解くアルゴリズムを用いて最初から解き直せばよい．しかしながら，$\mathcal{P}_{(i)}$ は直前の問題 $\mathcal{P}_{(i-1)}$ とあまり大きくは変わらないことが通例であり，以前の計算結果を有効に再利用し，かつ，解の変化が小さいこと（解の安定性）が望まれる．このような性質を得るために以下の2つの方法が考えられる．

初期値利用法：以前の問題の解を各変数の暫定的な初期値として，段階的にこの初期値を改善することにより，新しい問題の解を得る [Verfaillie 94].

制約記録法：以前の問題の解を求める過程で導かれた新しい制約（問題に陽に記述された制約から導かれる付加的な制約），およびその制約が成立する前提条件（justification）を，新しい問題を解く際に利用するために記録しておく．(問題に陽に記述された）制約が変化した場合に，これらの付加的な制約の前提条件をチェックし，制約の変化によって無効となるものは取り除く．文献 [Bessière 91, Prosser 92] では動的CSPにおいて制約を記録しarc-consistencyを達成するアルゴリズムが，文献 [Schiex 93] ではバックトラック型の探索により解を求めるアルゴリズムが示されている．

各変数に暫定的な初期値を与え，その初期値を段階的に改善していく方法（反復改善法）は，動的CSPに限らず通常のCSPを解くために広く用いられている方法であり [Minton 92, Morris 93, Selman 92, Yokoo 94]，改善のためのヒューリスティックとして，違反している制約の個数を減少させるように変数の値を変更する制約違反最少化ヒューリスティック [Minton 92] が用いられる．これらのアルゴリズムはそのまま動的CSPに適用可能であり，初期値として以前の問題の解を用いることにより解の安定性（stability）が得られることが期待される．

一方，以前の問題の解を求める過程で導かれた新しい制約を記録しておくことは，通常のCSPを解く際にも無駄な再計算を避けるために用いられるが，制約の変化に伴い，どの制約が再利用可能かの判断が必要となる点が動的CSPに特徴的である．前提条件を管理し，前提条件の変更に対してデータの整合性（consistency）を管理するシステムはTruth Maintenance Systemと呼ばれ，様々な研究がなされており [de Kleer 86, Doyle 79]，動的CSPでの制約の記録に関しても同様な手法が用いられている．

一般に，新しい制約を記録する場合，探索の削減が期待されるものの，新しい制約を記録し制約のチェックを行う処理のオーバーヘッドが問題となる．動的CSPの場合には，さらに新しい制約の前提条件を管理するコストが加わるため，将来的に利用価値の高いと思われる，強い制約のみを記録する等の工夫が必要となる．

4. 不完全CSP

4・1 問題の定式化

不完全CSPの一般的な定式化は次のとおりである [Freuder 89, Freuder 92].

$$\langle (P,U),(PS,\leq),(M,(N_0,S_0))\rangle$$

where

P： CSP

U： “universes” の集合，universes：P の各変数に対する潜在的な値の集合

$(PS,\leq)$： 問題空間，PS：CSPの集合，$\leq$：半順序

M： 問題空間上の距離

(N_0,S_0)： P との距離の必要値と十分値

直観的には，P が最初に与えられる，制約が強すぎて解が存在しない問題であり，PS が P の制約を何からの方法で緩めた問題の集合である．目的はなるべく P に近い（P との距離が小さい），解を持つ問題を見つけ，その解を求めることである．

より詳細には，問題空間上の半順序は，P_1 と P_2 をそれぞれ PS の要素とすると，

$$P_1 \leq P_2 \textit{ iff }（P_1 \text{ の解集合}）\supseteq（P_2 \text{ の解集合}）$$

で定義される．なお，P_1 と P_2 の解集合が等しいときは $P_1 = P_2$ と表し，$P_1 \leq P_2$ だが $P_1 = P_2$ でないときは $P_1 < P_2$ と表す．$P_1 < P_2$ のとき「P_1 は P_2 より弱い」という．PS は一般には $Q \leq P$ を満たすすべての Q からなる．問題空間には，この半順序関係と整合するように距離 M が定義される．距離の定義には様々な可能性があるが，例えば後述する最大CSPでは P_1 と P_2 の距離 $M(P_1,P_2)$ として，P_1 と P_2 で共有していない制約要素の数を用いている．

不完全CSPを解くとは，PS 内の問題から，P との距離が与えられた上限 N_0 より小さい問題，すなわち $M(P',P) < N_0$ となる問題 P' とその解を得るこ

とである．このような P' とその解のうち，P との距離が最小のものを最適解という．

不完全 CSP の重要なサブクラスとして，最大 CSP（Maximal Constraint Satisfaction Problem）[Freuder 92] がある．最大 CSP の目的は，できるだけ多くの制約を満足する解，言い換えると，もとの問題からできるだけ少ない制約を取り除くことによって充足可能となる問題とその解を探すことである．

4・2 例題／応用問題

現実の問題を CSP として記述した場合に過制約となることは多く，これらの CSP として記述したときに過制約となる可能性の高い問題は，不完全 CSP の応用問題と見なすことができる．

文献 [Verfaillie 96] では，地球観測衛星からの写真撮影のスケジューリングが不完全 CSP として記述されている*1．この場合，撮影したい写真を変数，撮影手段（どの機材をいつ使うか等）を値域，機材のハードウエア的な制約による撮影手段に関する制約を制約として初期の問題を作成し，撮影可能な写真とそれを撮る手段を解として求める．

また，文献 [Bakker 93] では，オランダのサッカーリーグの試合日程計画が不完全 CSP として記述されている*2．そこでは，HAP（Home-Away Patterns）と呼ばれるチームごとの試合日程計画のスキームが与えられ，リーグのクラブチームを変数，可能なスキームを値域とし，自治体や警察，鉄道，マスコミなどの要求や希望を制約とする．重要な要求や希望をできるだけ多く満たす計画を立てることが目標となる．

4・3 アルゴリズム

最大 CSP を解くアルゴリズムをいくつか紹介する．

〔1〕 分枝限定法（P-BB）

P-BB[Freuder 92] は，最大 CSP を解くもっとも素朴な深さ優先探索アルゴリズムであり，通常の CSP におけるバックトラック探索に相当する．基本的な動作は，まず，(N, S) に初期値 (N_0, S_0) を与え，深さ優先探索により部分解を伸ばしていく．途中，部分解の制約条件違反数，すなわち，その部分解を含む解の制約条件違反数の下界値が N 以上になるとバックトラックする．また，下界値が N 以上になることなく部分解を最後まで伸ばすことができれば，解を記憶し，その制約条件違反数を新たに N とする．なお，このとき，制約条件違反数が S 以下であれば終了するが，そうでなければバックトラックする．

P-BB は制約条件違反数が S_0 以下の解があれば，必ずそれを見つけることができる．また，それがない場合でも N_0 未満の解があれば，もっとも S_0 に近い解を見つけることができる．N_0 未満の解がない場合には，その事実がわかる．容易にわかるとおり，$(N_0, S_0) = (1, 0)$ であれば CSP の通常のバックトラック探索と等価である．

〔2〕 バックマーキング探索（P-BMK）

CSP におけるバックマーキング探索 [Gaschnig 77] は，無駄な制約チェックを行なわないようにバックトラック探索を改良したものである．基本的な動作は，変数順序を固定したバックトラック探索と同じで，部分解を伸ばすために，現在考慮中の変数 v の各値 d について，部分解との無矛盾性を調べる．バックトラック探索と異なるのは，このとき矛盾があれば，矛盾の原因となった部分解の値のうち変数順序がもっとも小さいものを選び，その順位を $mark(v, d)$ として記録しておく．また，ある変数に対する無矛盾な値が見つけられず，バックトラックして過去に割り当てた変数の値を変更する場合，遡って変更した変数の順位の最小値を $backto$ として記録する．これらの値をもとにバックマーキング探索では，変数 v への値を再考する際に，例えば $mark(v, d) < backto$ ならば，過去に $v = d$ と矛盾した変数の値がまだ変更されていないことになるため，制約をチェックしなくても d と現在の部分解とが矛盾すると推論できる．

P-BMK[Freuder 92] は，バックマーキング探索と同様のアイデアを用いて，無駄な制約チェックをしなくてすむように P-BB を改良した最大 CSP アルゴリズムである．CSP のバックマーキング探索と P-BMK の主な違いは，前者が，変数 v の値 d に対し矛盾の原因となる値の最小順位を記録するだけで十分なのに対し，後者では，変数順序中で最初に矛盾した値の順位，最後に矛盾した（制約条件違反数の下界値が N 以上になった）値の順位の両方を記録する必要がある．当然，それに合わせて矛盾を判定する推論規則も変更される．

〔3〕 Arc Consistency Count の計算（P-ACC）

Arc Consistency Count (ACC) とは，変数の各値ごとに定義され，その値より生じる制約条件違反の数を表す．前もってこの ACC を知ることができれば，例えば P-BB などのアルゴリズムにおいて，現在の部分解から得られる制約条件違反数の下界値に次の変数の

*1 正確には，不完全 CSP を拡張した重み付き CSP として記述される．

*2 同じく，重み付き CSP として記述される．

可能な値のACCを足すことで，より正確な下界値を得ることができるため，通常より早めに枝刈りしてバックトラックすることができる．P-ACC[Freuder 92]は，探索に先立つ前処理としてACCを計算するアルゴリズムであり，その計算量は，cを制約の数，dを値域サイズの最大値として$O(cd^2)$である．

また，文献[Larrosa 96, Wallace 95, Wallace 96]では，ACCの代わりにDirected Arc Consistency Count（DACC）を用いることにより，さらに正確な下界値が得られ，探索の効率が上がることが報告されている．

〔4〕 **フォワードチェッキング（P-FC）**

フォワードチェッキング[Haralick 80]とはCSPの解法の一種であり，制約伝播の手法を組み込んだバックトラック探索である．特徴は，変数に値を代入したときに制約伝播すること，すなわち，まだ値を代入していない残りの変数の値域からその値と矛盾するものを削除することである．制約伝播により，ある変数の値域が空になれば，先の代入を取り消し，かつ，代入に伴って削除した値を元に戻してバックトラックする．

同様にP-FC[Freuder 92]は，制約伝播の手法を組み込んだ分枝限定法と見なすことができる．P-FCでは，変数の各値ごとにInconsistency Count（IC）が定義され，初期値として0が与えられる．基本的な動作は，変数に値を代入するとき，(1) その時点での部分解とその値を含む解の制約条件違反数の下界値がNより小さいこと，(2) この値と矛盾する値のICを1つ増やし，先の下界値とICの和がN以上となる値を値域から削除した後，空となる値域がないこと，の2点を確認して，変数にその値を代入し，次の変数，値を考慮する．(1)，(2) のどちらかが満たされない時はバックトラックする．この手続きで，部分解を最後まで伸ばすことができれば，P-BBと同様，解の記憶，Nの更新，Sとの比較を行なう．

P-FCには，下界値の計算方法，Nとの比較方法，ICの計算方法に関して自由度があり，いくつかのバリエーションが提案されている．

〔5〕 **拡張およびその他の話題**

以上の基本的なアルゴリズムに対する拡張手段をいくつか示す．

最大CSPを解くアルゴリズムでは，値を代入していく変数の順序（変数順序），ある変数に対して代入する値の順序（値順序）が効率に大きく影響する．文献[Freuder 92]では，ACCを用いた変数順序，値順序が，また，文献[Wallace 93]では，順序付き制約グラフにおけるノードの幅[Dechter 92]を利用した変数順序が提案されている．

CSPを解くアルゴリズムとして近年，反復改善法と呼ばれる，完全ではないが特定の問題に対して非常に効率のよいアルゴリズムが提案されている．その基本的なアイデアは，制約条件違反を含む不完全な解を近傍探索により順次改善するというものである．文献[Wallace 96]では，最大CSPのアルゴリズムの前処理として反復改善法を一定時間実行し，得られた解の制約条件違反数をNの初期値とするというアイデアが示されている．

Russian Doll Search[Verfaillie 96]は，変数順序を固定して（$v_1, v_2, \ldots, v_n$とする），v_nから順番に，$\{v_n\}$，$\{v_n, v_{n-1}\}, \{v_n, v_{n-1}, \ldots, v_1\}$という各変数集合を含む部分問題系列を作り，この順番に分枝限定法で解いていく．特徴は，i番目の部分問題を解いて得られた最適値を利用して，$i+1$番目の問題の制約条件違反数の下界値を効率良く計算する点である．

最後に，不完全CSPに関連するその他の話題をいくつか取り上げる．

4・2節の脚注でも触れたとおり，不完全CSPの拡張として重み付きCSP（weighted CSP）がある．重み付きCSPでは，各制約にその制約が違反した場合のコストを表す自然数（重み）が定義され，違反する制約の重み和を最小化することが目標となる．**4・3節**であげた最大CSPを解くアルゴリズムを重み付きCSP用に拡張することは容易である．また，文献[Lau 96]では重み付きCSPの近似解を求める多項式時間アルゴリズムが提案されている．

制約論理プログラミングを拡張した階層型制約論理プログラミングでは，制約階層[Borning 92]というアイデアを用いて過制約な問題に対処している．制約階層では，制約がhardとsoftの2種類にわけられ，hardは必ず満たさなければならない制約，softは必ずしも満たす必要のない制約を表す．また，softな制約はその選好の度合に応じてさらに区分される．文献[Jampel 96]では，不完全CSPと制約階層の関係が明らかにされ，両者間の変換方法が示されている．さらに，文献[Bistarelli 95, Schiex 95]では，不完全CSP，重み付きCSP，制約階層などを統一的に扱える一般的な枠組が提案されている．

5. むすび

本稿ではCSPの拡張である分散CSP，動的CSP，不完全CSPについて解説を行った．これらの3つの拡張は排他的なものではなく，互いに組み合わされ，よ

り現実的な状況を表現できるように拡張されるのが自然である*3．しかしながら，むやみに問題の定義を拡張し一般化しすぎると，そのような一般的な問題を解く効率的なアルゴリズムを得ることが難しくなる．問題を解くアルゴリズムには健全性，完全性，アルゴリズムの効率等の明確な評価基準があるが，問題の定式化にはそのような明確な評価基準は存在しない．本稿で取り上げたこれらの拡張が妥当で有益であるかどうかは，これらの定式化により，どのような現実的な問題が表現されるか，効率的なアルゴリズムが得られるかどうかにかかっている．これらのCSPの拡張に関する研究は通常のCSPに関する研究と比較すると未だ事例も少なく，今後の研究の発展を期待したい．

◇参考文献◇

[Bakker 93] Bakker, R.R., Dikker, F., Tempelman, F. and Wognum, P.M.: Diagnosing and Solving Over-determined Constraint Satisfaction Problems, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp.276–281 (1993).

[Bessière 91] Bessière, C.: Arc-consistency in Dynamic Constraint Satisfaction Problem, *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp.221–226 (1991).

[Bistarelli 95] Bistarelli, S., Montanari, U. and Rossi, F.: Constraint Solving over Semirings, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp.624–630 (1995).

[Borning 92] Borning, A., Freeman-Benson, B. and Wilson, M.: Constraint Hierarchies, *Lisp and Symbolic Computation*, Vol.5, pp.223–270 (1992).

[Chandy 85] Chandy, K. and Lamport, L.: Distributed Snapshots: Determining Global States of Distributed Systems, *ACM Transaction on Computer Systems*, Vol.3, No.1, pp.63–75 (1985).

[Collin 91] Collin, Z., Dechter, R. and Katz, S.: On the Feasibility of Distributed Constraint Satisfaction, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp.318–324 (1991).

[Conry 91] Conry, S.E., Kuwabara, K., Lesser, V.R. and Meyer, R.A.: Multistage Negotiation for Distributed Constraint Satisfaction, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.21, No.6, pp.1462–1477 (1991).

[de Kleer 86] de Kleer, J.: An Assumption-based TMS, *Artificial Intelligence*, Vol.28, pp.127–162 (1986).

[de Kleer 89] de Kleer, J.: A Comparison of ATMS and CSP Techniques, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 290–296 (1989).

[Dechter 92] Dechter, R.: Constraint Networks, Shapiro, S.C., ed., *Encyclopedia of Artificial Intelligence*, pp.276–285, Wiley-Interscience (1992).

[Doyle 79] Doyle, J.: A Truth Maintenance System, *Artificial Intelligence*, Vol.12, pp.231–272 (1979).

[Freuder 89] Freuder, E.C.: Partial Constraint Satisfaction, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp.278–283 (1989).

[Freuder 92] Freuder, E.C. and Wallace, R.J.: Partial Constraint Satisfaction, *Artificial Intelligence*, Vol.58, No.1–3, pp.21–70 (1992).

[Gaschnig 77] Gaschnig, J.: A General Backtrack algorithm that eliminates most redundant checks, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, p.457 (1977).

[Haralick 80] Haralick, R.M. and Elliot, G.L.: Increasing Tree Search Efficiency for Constraint Satisfaction Problems, *Artificial Intelligence*, Vol.14, pp.263–313 (1980).

[Hirayama 95] Hirayama, K. and Toyoda, J.: Forming Coalitions for Breaking Deadlocks, *Proceedings of First International Conference on Multi-Agent Systems*, pp.155–162 (1995).

[Jampel 96] Jampel, M., Jacquet, J., Gilbert, D. and Hunt, S.: Transformations Between HCLP and PCSP, Freuder, E.C., ed., *Principles and Practice of Constraint Programming –CP96*, Vol.1118 of *Lecture Notes in Computer Science*, pp.252–266, Springer-Verlag (1996).

[Larrosa 96] Larrosa, J. and Meseguer, P.: Exploiting the Use of DAC in MAX-CSP, Freuder, E.C., ed., *Principles and Practice of Constraint Programming – CP96*, Vol.1118 of *Lecture Notes in Computer Science*, pp.308–322, Springer-Verlag (1996).

[Lau 96] Lau, H.C.: A New Approach for Weighted Constraint Satisfaction: Theoretical and Computational Results, Freuder, E.C., ed., *Principles and Practice of Constraint Programming –CP96*, Vol.1118 of *Lecture Notes in Computer Science*, pp.323–337, Springer-Verlag (1996).

[Lesser 83] Lesser, V.R. and Corkill, D.D.: The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks, *AI Magazine*, Vol.4, No.3, pp.15–33 (1983).

[Mackworth 92] Mackworth, A.K.: Constraint Satisfaction, in Shapiro, S.C., ed., *Encyclopedia of Artificial Intelligence*, pp.285–293, Wiley-Interscience Publication, New York (1992).

[Minton 92] Minton, S., Johnston, M.D., Philips, A.B. and Laird, P.: Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems, *Artificial Intelligence*, Vol.58, No.1–3, pp.161–205 (1992).

[Morris 93] Morris, P.: The Breakout Method for Escaping From Local Minima, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp.40–45 (1993).

[Prosser 89] Prosser, P.: A Reactive Scheduling Agent, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp.1004–1009 (1989).

[Prosser 92] Prosser, P., Conway, C. and Muller, C.: A Constraint Maintenance System for the Distributed Allocation Problem, *Intelligent Systems Engineering*, Vol.1, pp.76–83 (1992).

[Schiex 93] Schiex, T. and Verfaillie, G.: Nogood Recording for Static and Dynamic Constraint Satisfaction Problems, *5th International Conference on Tools with Artificial Intelligence*, pp.48–55 (1993).

[Schiex 95] Schiex, T., Fargier, H. and Verfaillie, G.: Valued Constraint Satisfaction Problems: Hard and Easy

*3 文献[Prosser 92]は複数エージェントが動的な変化に対応する方法を，文献[Yokoo 93]は複数エージェントで不完全な解を求める方法を示している．

Problems, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp.631–637 (1995).

[Selman 92] Selman, B., Levesque, H. and Mitchell, D.: A New Method for Solving Hard Satisfiability Problems, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp.440–446 (1992).

[Verfaillie 94] Verfaillie, G. and Schiex, T.: Solution Reuse in Dynamic Constraint Satisfaction Problems, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp.307–312 (1994).

[Verfaillie 96] Verfaillie, G., Lemaitre, M. and Schiex, T.: Russian Doll Search for Solving Constraint Optimization Problems, *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp.181–187 (1996).

[Wallace 93] Wallace, R.J. and Freuder, E.C.: Conjunctive Width Heuristics for Maximal Constraint Satisfaction, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp.762–768 (1993).

[Wallace 95] Wallace, R.J.: Directed Arc Consistency Preprocessing as a Strategy for Maximal Constraint Satisfaction, Meyer, M. ed., *Constraint Processing*, volume 923 of *Lecture Notes in Computer Science*, pp.121–138, Springer-Verlag (1995).

[Wallace 96] Wallace, R.J.: Enhancements of Branch and Bound Methods for the Maximal Constraint Satisfaction Problems, *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp.188–195 (1996).

[Waltz 75] Waltz, D.: Understanding Line Drawing of Scenes with Shadows, Winston, P.(ed.), *The Psychology of Computer Vision*, pp.19–91, McGraw-Hill (1975).

[Yokoo 90] Yokoo, M., Ishida, T. and Kuwabara, K.: Distributed Constraint Satisfaction for DAI Problems, *10th International Workshop on Distributed Artificial Intelligence* (1990).

[Yokoo 92] Yokoo, M., Durfee, E.H., Ishida, T. and Kuwabara, K.: Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving, *Proceedings of the Twelfth IEEE International Conference on Distributed Computing Systems*, pp.614–621 (1992).

[Yokoo 93] Yokoo, M.: Constraint Relaxation in Distributed Constraint Satisfaction Problem, *5th International Conference on Tools with Artificial Intelligence*, pp.56–63 (1993).

[Yokoo 94] Yokoo, M.: Weak-commitment Search for Solving Constraint Satisfaction Problems, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp.313–318 (1994).

[Yokoo 95] Yokoo, M.: Asynchronous Weak-commitment Search for Solving Distributed Constraint Satisfaction Problems, Montanari, U. and Rossi, F., eds., *Principles and Practice of Constraint Programming –CP95*, volume 976 of *Lecture Notes in Computer Science*, pp.407–422, Springer-Verlag (1995).

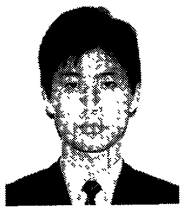
[Yokoo 96] Yokoo, M. and Hirayama, K.: Distributed Breakout Algorithm for Solving Distributed Constraint Satisfaction Problems, *Proceedings of Second International Conference on Multi-Agent Systems*, pp.401–408 (1996).

[Zweben 89] Zweben, M. and Eskey, M.: Constraint Satisfaction with Delayed Evaluation, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp.875–880 (1989).

著 者 紹 介

横尾 真(正会員)

1962年生まれ．1984年東京大学工学部電子工学科卒業．1986年同大学院修士課程修了．同年NTTに入社．1990～91年ミシガン大学客員研究員．現在NTTコミュニケーション科学研究所に勤務．分散人工知能，制約充足問題に関する研究に従事．分散探索，分散制約充足問題等に興味を持つ．博士(工学)．1992年 人工知能学会論文賞，1995年 情報処理学会坂井記念特別賞受賞．情報処理学会，日本ソフトウェア科学会，AAAI各会員．〈yokoo@cslab.kecl.ntt.co.jp〉



平山 勝敏(正会員)

1990年3月大阪大学基礎工学部制御工学科卒業．1992年3月同大学院基礎工学研究科博士前期課程修了．1995年3月同博士後期課程修了．同年4月 神戸商船大学情報システム工学講座助手．現在に至る．博士(工学)．制約充足，マルチエージェントシステムに関する研究に従事．情報処理学会，AAAI各会員．〈hirayama@ti.kshosen.ac.jp〉