論 文 Technical Papers

複雑な局所問題に対応する分散制約充足アルゴリズム

Distributed Constraint Satisfaction Algorithm for Complex Local Problems

横尾 真 Makoto Yokoo
NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories.
yokoo@cslab.kecl.ntt.co.jp, http://www.kecl.ntt.co.jp/csl/ccrg/members/yokoo/

平山 勝敏 Katsutoshi Hirayama
神戸商船大学
Kobe University of Mercantile Marine.
hirayama@ti.kshosen.ac.jp, http://jos2.ti.kshosen.ac.jp/~hirayama/

Keywords: constraint satisfaction problem, multi-agent system, distributed artificial intelligence.

Summary

A distributed constraint satisfaction problem can formalize various application problems in Multi-agent systems, and several algorithms for solving this problem have been developed. One limitation of these algorithms is that they assume each agent has only one local variable. Although simple modifications enable these algorithms to handle multiple local variables, obtained algorithms are neither efficient nor scalable to larger problems.

We develop a new algorithm that can handle multiple local variables efficiently, which is based on the asynchronous weak-commitment search algorithm. In this algorithm, a bad local solution can be modified without forcing other agents to exhaustively search local problems. Also, the number of interactions among agents can be decreased since agents communicate only when they find local solutions that satisfy all of the local constraints. Experimental evaluations show that this algorithm is far more efficient than an algorithm that uses the prioritization among agents.

1. はじめに

分散人工知能とは，複数の自律的に動作するエージェントの相互作用，特にこれらのエージェント間の協調に関する人工知能の研究の一分野である．近年の計算機の小型化，低価格化，および計算機ネットワーク技術の進歩により，分散計算機環境が急速に普及しつつあり，このような自律的なエージェントに関する研究の必要性は非常に大きくなっている．このため，分散人工知能は人工知能の中でも非常に活発な分野となっている．

一方，制約充足問題 [Mackworth 92] とは，有限で離散的な領域から値を取る複数の変数に対して，制約を満足する値の割当を発見する問題であり，人工知能の様々な問題がこの問題により定式化できることが知られている．著者らは文献 [横尾 92a, Yokoo 92b, Yokoo 98a] において，制約充足問題の変数，制約が複数のエージェントに分散された問題として分散制約充足問題を定義し，分散人工知能の様々な応用問題 (分散資源割当問題 [Conry 91], マルチエージェントによるデータの真偽値管理 [Huhns 91] 等) が分散制約充足問題として定式化可能であることを示した．このように，分散人工知能，マルチエージェントシステムで生じる様々な問題が分散制約充足問題として定式化可能であるため，分散制約充足問題を解くための分散アルゴリズムは，エージェント間の協調を実現するための重要なインフラストラクチャであると考えることができる．

著者らは分散制約充足問題を解く基本的なアルゴリズムとして，各エージェントが局所的な知識に基づいて，非同期，並行に解を探索する非同期バックトラッキングアルゴリズム [横尾 92a, Yokoo 92b] を，さらに，非同期バックトラッキングアルゴリズムを拡張し高速化した非同期弱コミットメント探索アルゴリズムを提案している [Yokoo 95, 横尾 96a, Yokoo 98a]．また，著者らは分散制約充足問題を解く反復改善型の探索アルゴリズムを提案している [Hirayama 95, Yokoo 96b, 横尾 98b].

これらのアルゴリズムの欠点として，基本的には各エージェントは唯一の変数を持つことを仮定していることがある．このような仮定は，各エージェントの解く局所問

題が大きく複雑である場合には成立しないと考えられる．これらのアルゴリズムは以下のような2種類の単純な方法で，1つのエージェントが複数の変数を持つ場合に対応可能であるが，いずれも効率，局所問題が大きくなった場合のスケーラビリティの点で問題がある．

(1) 各エージェントが最初に局所問題の解をすべて求めておく．

これにより，各エージェントは唯一の変数を持ち，局所問題の解がこの唯一の変数の取り得る値だとすることにより，1エージェント1変数のアルゴリズムが適用可能である．この方法の問題点として，局所問題が大きく複雑になった場合に，局所問題のすべての解を求めることが現実的には不可能となることがある．

(2) 各エージェントが，それぞれの変数に対応する仮想的なエージェントの動作をシミュレートする．

すなわち，あるエージェント k が変数 x_i, x_j の2つを持っている場合，x_i に対応する仮想的なエージェントと，x_j に対応する仮想的なエージェントが存在すると仮定し，エージェント k がこれらの2つの仮想的なエージェントの並行動作をシミュレートし，1エージェント1変数のアルゴリズムを実行する．この場合は局所問題の解をあらかじめ数え上げる必要はないが，一般にエージェント間通信はエージェント内部の処理と比較して低速である．このため，複数のエージェントの並行動作を1つのエージェントでそのままシミュレートし，同じエージェント内で実行される複数の仮想的なエージェント間の通信を，他の物理的に異なるエージェントとの通信と同等に扱うのは無駄がある．

文献 [Armstrong 97] では，1エージェントが複数の変数を持つ場合に対応するため，エージェント間の優先順位を導入している．このアルゴリズムでは，各エージェントは自分より優先順位の高いエージェントの持つすべての変数の値と制約を満足するような局所解を求めようとし，そのような局所解が存在しない場合にはバックトラック，もしくは優先順位の変更が行われる．文献 [Armstrong 97] では，より良い優先順位を決定し，動的に変更するための様々なヒューリスティクスが検討されている．

この方法の問題点として，優先順位の高いエージェントの変数の値を変更するためには，局所問題の解に関する網羅的な探索が必要となることがある．局所問題が大きく複雑になった場合，そのような網羅的な探索を行うことは現実的ではない．このアプローチは前者の方法に近いが，局所問題の解をあらかじめ数え上げておくのではなく，必要に応じて作る点が異なっている．しかしながら，優先順位の高いエージェントの選択した局所解が悪い場合には，優先順位の低いエージェントでは網羅的な局所解の探索を強いられる．

本論文では，後者のアプローチをベースに，エージェント内での複数の変数に関する処理を逐次化し，かつ，エージェント内の制約を満足する局所解が得られた時点で他のエージェントと通信を行うことにより，他のエージェントとのインタラクションの回数を削減したアルゴリズムを提案する．例題を用いた評価により，本アルゴリズムが，エージェントの優先順位を用いる方法よりもはるかに効率的であることを示す．

本論文では以下，分散制約充足問題の定義と，1エージェント1変数の場合の非同期弱コミットメント探索アルゴリズムの紹介を行い (**2章**)，1エージェント複数変数の場合への拡張方法を示し (**3章**)，実験的な評価結果を示す (**4章**)．

2. 分散制約充足問題

2·1 問題の定義

制約充足問題（CSP）は一般に次のように定義される．n 個の変数 $x_1, x_2, \ldots, x_n$ と，変数のそれぞれが値をとる有限で離散的な領域 $D_1, D_2, \ldots, D_n$，および制約の集合が存在する．本論文では制約は述語によって内包的に定義されるとする．すなわち，制約 $p_k(x_{k1}, \ldots, x_{kj})$ は，直積 $D_{k1} \times \ldots \times D_{kj}$ に対して定義され，これらの変数の値が互いに整合のとれている場合に真となる．制約充足問題の解を求めることは，すべての制約を満足する変数の値の組を求めることである．

分散制約充足問題とは，制約充足問題の変数が複数のエージェントに分散された問題である．本論文では以下のエージェント間通信のモデルを仮定する．

- エージェント間通信はメッセージ通信によってなされる．
- エージェントは，他のエージェントのアドレスを知っている場合に限りそのエージェントにメッセージを送信できる．
- メッセージの遅延は有限であるが，遅延時間の上限は分かっていない．
- 任意の2つのエージェントの組み合わせに関して，送信されたメッセージの順序は保存される．

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，異なるエージェントの持つ変数間に制約がある．エージェントの目的は，エージェント間の制約をすべて満足する値の割当を見つけることである．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 p_k を知っていることを known(p_k, i) と書く．

次の場合に，分散制約充足問題が解けたという．

- 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．そして，すべてのエージェント k について，$\forall p_l$ known(p_l, k)

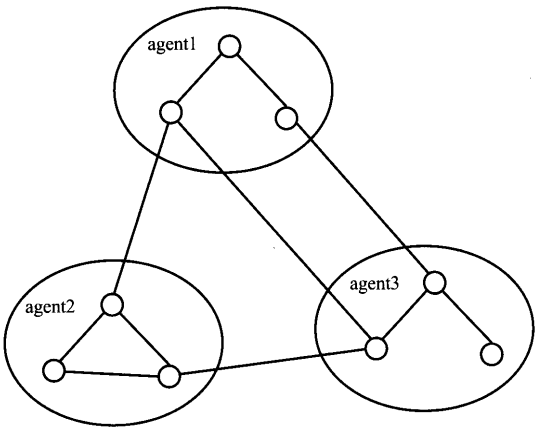

図1 制約グラフの例

なる制約が，$x_1 = d_1, x_2 = d_2, \ldots, x_n = d_n$ のもとで満足される．

以下，アルゴリズムの説明を行う際に，簡単のため次の仮定をおく．これらの仮定を緩和し，アルゴリズムを一般の場合に拡張することは容易である．

- 各エージェントは自分に属する変数が関係する制約をすべて知っている．
- エージェント間の制約はすべて二項関係（binary）である．

すべての制約がbinaryである分散制約充足問題はグラフを用いて表現できる (図1)．すなわち，変数がグラフのノードに対応し，ノード間のリンクが制約に対応する．また，エージェントは変数の集合，図中の大きな円として表される．

2·2 非同期弱コミットメント探索アルゴリズム（1エージェント1変数の場合）

1エージェント1変数の非同期弱コミットメントアルゴリズムでは，エージェントは非同期，並行的に自分の変数の値を決定し，その値を他のエージェントに送信した後，メッセージ待ちの状態になり，以降，到着したメッセージに関する処理を行う．エージェント間で交換されるメッセージは，設定された値を送信する *ok?* メッセージと，制約条件違反が生じたことを伝える *nogood* メッセージの二種類である．各エージェントでメッセージの受信によって起動される手続きの内容，および手続き中で用いられる変数の意味は次の通りである．

- 各変数に対して，その変数の優先順位を表す非負の整数値を定義する．これを変数の優先度と呼ぶ．
- 変数の優先順位は，優先度の大小関係によって決定される (優先度が大きいほど順序関係で上位となる)．
- 優先度が同じ変数の順序関係は，変数の識別子の辞書式順序で決定される．
- 変数の優先度の初期値はすべて0である．
- エージェントは他のエージェントから *ok?* メッセージによって送信された値を *agent_view* に記録する．*agent_view* はこのエージェントから見た他のエージェントの値の割当の状態である．
- 現在の自分の変数の値が，*agent_view* 中の，自分の変数より優先順位がより高いものとの間の制約を満足するとき，現在の値と *agent_view* は整合が取れている (consistent である) という[*1]．現在の値と *agent_view* が整合が取れていない場合，エージェントは *agent_view* と整合するように値の変更を行い，他のエージェントに通信を行う．
- エージェントが，*agent_view* と整合する値を発見できない場合には，エージェントは，自分の優先順位を，関連する他のエージェントの優先順位の最大値より大きく設定する．また，*agent_view* 中のエージェントに対して *nogood* メッセージを送信する[*2]．

このアルゴリズムでは，解が存在する場合は，すべてのエージェントの値が制約を満たす安定状態に到達し，解が存在しない場合は空集合からなるnogoodが発見されてアルゴリズムは終了する．

3. 複数変数非同期弱コミットメントアルゴリズム

3·1 基本的なアイデア

本論文では，1エージェント1変数の場合の非同期弱コミットメント探索アルゴリズムをベースに，以下のような変更を行う．

- エージェントは，自分の持っている変数のうち，より優先順位の高い変数 (他のエージェントの持つ変数と自分の持つ変数の両方) との制約を満足しないもので，最も優先順位の高いものから順に値の変更を行う．
- より優先順位の高い変数との制約を満足する値が存在しない場合は，その変数に関して優先順位を高くする．
- 上記の処理を繰り返し，エージェント内の制約をすべて満足する局所解が得られた時点で，他のエージェントに対して変更分を送信する．

エージェントの持つ各変数は，より優先順位の高い変数 (他のエージェントの持つ変数および自分の持つ変数) との制約を満足する必要がある．このため，自分の持つ変数に関しては，優先順位の高い変数の値が決定されるより前に，優先順位の低い変数の値を決定しても無駄になることが多い．よって，優先順位の高い変数から逐次的に値の決定を行う．また，エージェント内の制約をすべて満足する局所解が得られた時点で初めて他のエージェ

*1 正確には，与えられた制約を満たすだけではなく，他のエージェントから通信された新しい制約条件 (nogood) にも違反しない必要がある．

*2 この処理はアルゴリズムの完全性を保証するために必要であるが，完全性を犠牲にすれば省いても良い．実際，大規模な問題では，アルゴリズムの理論的な完全性はあまり意味を持たない．

ントに対して変更分を送信することにより，エージェント間の余分なインタラクションの削減が可能となる．

この方法を用いる場合，あるエージェントの選択した局所解が悪い (他のエージェントで，その局所解と制約を満す局所解が存在しない) 場合，他のエージェントは局所解の網羅的な探索を行う必要はなく，単にその局所解との制約を満足しない，いくつかの変数の優先順位を増加させるのみである．

3·2 アルゴリズムの詳細

本アルゴリズムでは，基本となる1エージェント1変数の場合の非同期バックトラッキングアルゴリズムと同様，各変数に対して優先度が定義され，エージェントは非同期に並行して値を決定し，その値を関連するエージェントに送信した後，メッセージ待ちの状態になり，以降，到着したメッセージに関する処理を行う．図2に *ok?* メッセージを受けた時のエージェント x_i で起動される手続きを示す*3．この例では，1つの *ok?* メッセージを受ける毎に check_agent_view が呼ばれるように記述しているが，実際には複数の *ok?* メッセージを受け，agent_view を更新してから一度だけ check_agent_view を実行することが可能である．

3·3 アルゴリズムの実行例

具体例を図3を用いて示す．例題は分散グラフ色塗り問題であり，各変数に，リンクで結ばれた変数と異なる色になるように，色を割り当てる問題である．変数の取りうる色は白，黒，灰色の三種類である．図ではエージェント1，エージェント2が存在し，それぞれ変数を3つ持っている．

初期値は図3 (a) のようになっていると仮定する．各エージェントはこの初期値を *ok?* メッセージにより通信しあう．初期状態では変数の優先度はすべて0である．各エージェントは，通信された値を用いて，現在の変数への値の割当てが，より優先順位の高い変数との制約を満足しているかをチェックする．変数の優先度はすべて0であるため，変数の優先順位は変数の名前のアルファベット順で決定される．

初期状態ではエージェント1の持つ変数はすべてエージェント2の持つ変数より優先順位が高く，かつエージェント1内の制約を満足しているため，エージェント1は値の変更を行わない．一方，エージェント2では，エージェント2内で最も優先順位の高い変数 x_4 は制約を満足しているが，x_5 はエージェント1の持つ変数 x_2 との制約を満足していない．ここでエージェント2は x_5 の値を，より優先順位の高い変数 x_2 と x_4 との制約を満す値である灰色に変更する．この変更により，x_5 と x_6 の制約が満足されないため，次に x_6 の値を変更しようとするが，x_3 が黒，x_4 が白，x_5 が灰色であるため，制約を満す値が存在しない．このため，エージェント2は，変数 x_6 の優先度を1に変更する．また，同時に x_6 の値を，より優先順位の低い変数との制約条件違反の個数を最少化する値に設定する．この場合はどの色を選択しても制約条件違反の個数は1であるため，ランダムに値を選択する (この場合は黒が選択されたとする)．nogood の処理を行う場合は，{(x_3, black), (x_4, white), (x_5, gray)} を nogood として記録し，関連するエージェントへの送信を行う．この結果，エージェント2の変数は，すべてより優先順位の高い変数との制約を満足しており，変更分をエージェント1に通信する (図3 (b))．

エージェント1では，x_1,x_2 はより優先順位の高い変数との制約を満足しているが，x_3 はより優先順位の高い変数 x_6 (優先度が1) との制約を満足していない．よって x_3 の値を制約を満す値，灰色に決定し，この結果エージェント全体として制約を満足する解が得られる (図3 (c))．

実際には，初期状態でのエージェント1の局所解と制約を満足するエージェント2の局所解は存在しない．エージェント毎に優先順位を決定する方法では，エージェント2で局所解の網羅的な探索が必要となる．一方，本ア

```
when received (ok?, (sender_id, variable_id,
        variable_value, priority)) do
  add (sender_id, variable_id, variable_value, priority)
        to agent_view;
  when agent_view and current_assignments
        are not consistent do
    check_agent_view; end do;

procedure check_agent_view
  if agent_view and current_assignments
        are consistent then
    communicate changes to related agents;
  else select x_k, which has the highest priority and
      violating some constraint with
      higher priority variables;
    if no value in D_k is consistent with
        agent_view and current_assignments then
      record and communicate a nogood, i.e., the subset
          of agent_view and current_assignments,
        where x_k has no consistent value;
      when the obtained nogood is new do
        set x_k's priority value to the highest priority
            value of related variables + 1;
        select d ∈ D_k where d minimizes the number of
          constraint violations with
              lower priority variables;
        set the value of x_k to d;
        check_agent_view; end do;
    else select d ∈ D_k where d is consistent
        with agent_view and current_assignments,
        and minimizes the number of constraint
        violations with lower priority variables;
      set the value of x_k to d;
      check_agent_view; end if; end if;
```

図 2 メッセージに対する処理 (複数変数非同期弱コミットメント探索)

*3 簡単のため，nogood メッセージの受信に関する処理の記述は省略する．完全性が必要とされない場合は，nogood メッセージの送受信は実行しなくても良い．

ルゴリズムでは，変数毎に優先順位を設定し，それらを動的に変更することにより，局所解の網羅的な探索なしに，他のエージェントの悪い選択を変更することが可能となっている．

4. 評　価

本論文では以下のような計算モデルを用いて評価を行う．すなわち，各エージェントは自分自身のクロックを管理し，送信された複数のメッセージを受理し，局所的な計算を行い，複数のメッセージを送信するごとにクロックの値を1つ増加させる（これを1サイクルと呼ぶ）．送信されたメッセージは次のサイクルで他のエージェントで受信されるとする．シミュレータを用いて，解を得るのに必要とされるサイクル数を測定する．1サイクルはエージェントが外界の状況を認識し，状況への対応方法を決定し，通信を行うという一連の行動を意味する．

本アルゴリズム (multiple-variables asynchronous weak-commitment search, multi-AWC) の比較の対象として，文献 [Armstrong 97] と同様，エージェントの優先順位を用い，あるエージェントでより優先順位の高いエージェントの局所解と制約を満す局所解が得られないことが判明した時点で，優先順位の変更を行うアルゴリズム (asynchronous weak-commitment search with agent priority, AWC+AP), および各エージェントが，各変数に対応する複数のエージェントの動作をシミュレートするアルゴリズム (single-variable asynchronous weak-commitment search, single-AWC) を用いる．AWC+APは，文献 [Armstrong 97] で示されている decaying nogoods heuristic を用いたアルゴリズムに対応するが，文献 [Armstrong 97] で示されているアルゴリズムでは，各エージェントは逐次的に動作することを前提としている．一方，本論文の AWC+AP では，他のアルゴリズムとの公正な比較を行うため，エージェントが並行に動作することを許している．

以下，前述の分散グラフの色塗り問題を用いて評価を行う．この問題は，移動体通信での周波数帯割当問題（隣接する区域では干渉を避けるため同じ周波数帯を使用できない）等を表現していると考えることができる．グラフの色塗り問題は変数の個数 n，各変数の取りうる色の数 k，エージェント間のリンクの個数 l の3つのパラメータで表現される．リンクの個数を変数の個数で割った値 (リンクの密度) が問題の難しさに大きく影響し，$k=3$ の場合，リンクの密度が2.7付近の場合が，最も難しい問題が得られることが知られている [Cheeseman 91]．

表1に，エージェントの数を10とし，まず，各エージェントの取り得る色の数 k を3，リンクの個数を $l=n\times 2.7$ 本にした場合の，各エージェントの持つ変数の個数を変化させた場合の結果を示す．表に示す値は，100個の異なる問題に関して試行を行った場合の平均である．

問題を生成する際に，単純にランダムにリンクを生成した場合，エージェントの解く局所問題内の制約の個数が非常に小さくなってしまう．エージェントの個数が10，各エージェントの持つ変数の個数10の場合は，全体の制約の個数は270であるが，その内同じエージェントの持つ変数間のものはその10分の1以下であり，各エージェント毎では2〜3個程度にしかならない．エージェントの持つ局所問題は意味のあるまとまりであり，現実的な問題設定では，局所問題内の制約はエージェント間の制約よりも強いと予想される．このため，本論文での評価では，制約の総数の半分をエージェント内の変数間の制約に割り当て，残りを異なるエージェント間の制約とすることとする．問題はこれらのパラメータ設定に従い，文献 [Minton 92] で示されている方法を用いて，解が存在し，グラフが連結されている問題のみを乱数を用いて生成する．各変数の初期値はランダムに決定されている．

実験を妥当な時間内で終了させるため，サイクル数の上限を10,000に設定し，この上限を越えた試行は打ち切り，サイクル数を10,000とカウントして平均を求める．また，制限時間内で解が得られた試行の割合を表中に示す．また，各サイクルでのエージェントの局所的な計算時間の目安として，制約チェックの回数を併せて調べる．すなわち，各サイクルで最も制約チェックの回数の多かったエージェント (各サイクルで処理のボトルネックとなっているエージェント) を選択し，それらのエージェントの制約チェックの回数の合計を示す．

また，表2に，各エージェントの持つ変数の個数を10に固定し，エージェントの個数を変化させた場合の結果を示す．

これらの結果から以下の知見が得られる．

- 本実験の範囲においては，提案した multi-AWC の性能は single-AWC, AWC+AP の性能を上回っており，また，各エージェントの持つ局所問題が大きくなり，エージェント数が多くなるほど，他のアルゴリズムとの差が大きくなる．
- multi-AWC は single-AWC よりサイクル数が少ない．この理由として，multi-AWC ではエージェント内で整合の取れた局所解が得られた時点で初めて他のエージェントに対して変更部分を通信するのに対して，single-AWC では，エージェントは各変数に対応する仮想的なエージェントの処理を並行して行うのみであり，これらの変数間の制約が満足されていない場合でも，他のエージェントとのインタラクションが行われるためであると考えられる．single-AWC での各サイクルにおける制約チェックの回数は multi-AWC よりわずかに小さいが，サイクル数の増加を埋め合わせるには不十分である．
- multi-AWC は AWC-AP よりサイクル数が少なくなっている．この結果は一見，直感に反するように

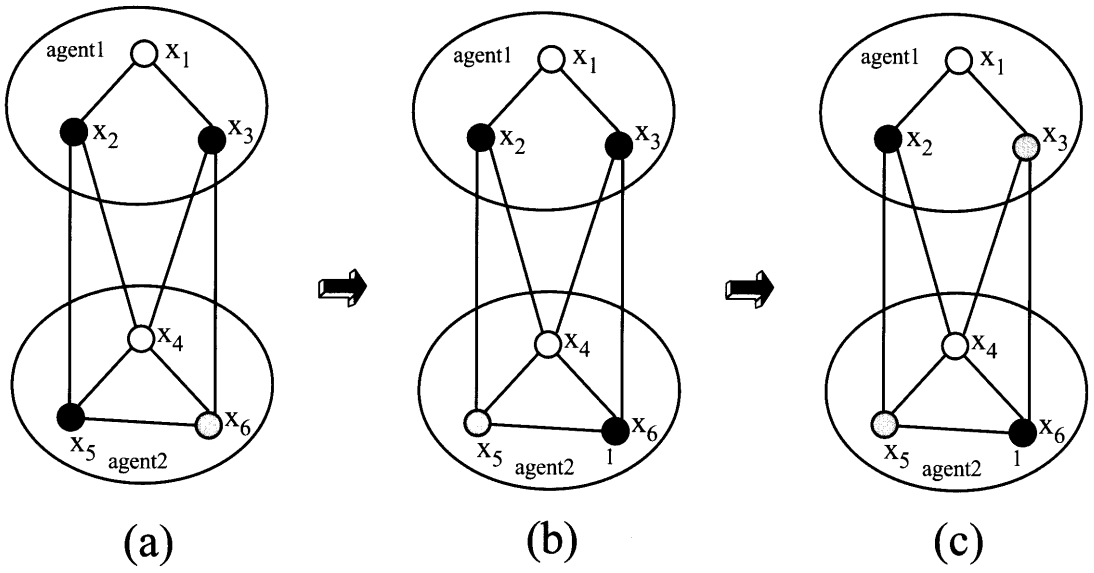

図 3 アルゴリズムの実行例

表 1 変数の個数を変化させた場合の評価結果 ($k = 3, l = 2.7 \times n, m = 10$)

	multi-AWC			AWC+AP			single-AWC		
n/m	ratio	cycles	checks	ratio	cycles	checks	ratio	cycles	checks
5	100%	26.9	2989.6	100%	35.9	3617.6	100%	323.0	13630.7
10	100%	89.5	22481.2	100%	577.7	155026.1	79%	4713.0	369195.0
15	100%	189.5	87688.8	79%	3951.8	1978801.9	14%	9083.4	1031475.1
20	100%	488.1	320312.6	37%	7529.6	6691615.5	0%	—	—

表 2 エージェントの個数を変化させた場合の評価結果
($k = 3, l = 2.7 \times n$, 各エージェントの持つ変数の個数 $= 10$)

	multi-AWC			AWC+AP			single-AWC		
m	ratio	cycles	checks	ratio	cycles	checks	ratio	cycles	checks
10	100%	89.5	22481.2	100%	577.7	155026.1	79%	4713.0	369195.0
15	100%	214.7	62049.3	90%	3039.2	888422.6	10%	9573.6	779022.9
20	100%	615.6	190718.2	54%	6568.1	2083577.1	0%	—	—

思える．AWC-AP では各サイクルにおいてエージェントは優先順位の高いエージェントとの制約を満足する局所解を網羅的に探索するのに対して，multi-AWC はそのような努力を行わず，単純に変数の優先順位を増加するのみである．しかしながら，現実には AWC-AP のような網羅的な探索を行うことは必ずしも良い結果をもたらさない．AWC-AP では，各エージェントは，より優先順位の高いエージェントの局所解と整合する局所解を選択する必要があり，結果的に，より優先順位の低いエージェントの持つ変数との制約違反の個数が多くなるような，エージェント全体としては，あまり望ましくない局所解を選んでしまうことが考えられる．multi-AWC では，各変数に関して，より優先順位の高い変数と制約を満す値が存在しない場合は，その変数の優先順位を増加させ，エージェント全体との制約条件違反の個数を減少させる値が選択される．結果として，得られた局所解は，エージェント全体としてより望ましい (制約違反の個数を減少させる) 解になっていると考えられる．

図 4 に，10 エージェント，10 変数の場合の 1 つのインスタンスに関する，制約条件違反の個数の変化のトレースを示す．各エージェントが優先順位の高いエージェントとの制約を満足するために労力を費やしすぎると，エージェント全体として制約条件違反の個数を減少させることは困難となることが示されている．

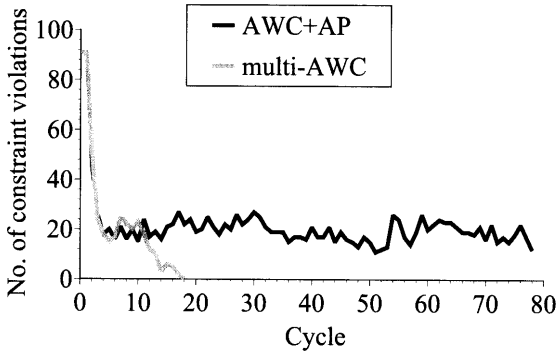

図 4 制約条件違反の個数の変化

5. おわりに

本論文では非同期弱コミットメントアルゴリズムをベースとして，1エージェントが複数変数を持つ場合に効率的に対応する方法を示した．本アルゴリズムでは，各エージェントは各変数に付属する優先度に基づいて変数の優先順位を決定し，優先度を変化させることにより，変数の優先順位を動的に変更する．この方法により，エージェント内の局所解に関する網羅的な探索なしに，他のエージェントの局所解を変更可能である．また，各エージェントは制約を満す局所解が得られた時点で，変更分を通信することにより，他のエージェントとのインタラクションの回数を削減することが可能となった．実験的な評価により，本方式がエージェント毎に優先順位を設定する方法よりも効率的であることを示した．

今後の研究課題として，反復改善型のアルゴリズム[Yokoo 96b, 横尾 98b]に関しても同様に1エージェントが複数変数を持つ場合への拡張方法を検討すること，また，分散制約充足問題において，エージェント内とエージェント間の制約の比率が問題の難しさに与える影響に関して解析することが挙げられる．

◇参考文献◇

[Armstrong 97] Armstrong, A. and Durfee, E. H.: Dynamic Prioritization of Complex Agents in Distributed Constraint Satisfaction Problems, in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pp. 620–625 (1997).

[Cheeseman 91] Cheeseman, P., Kanefsky, B., and Taylor, W.: Where the really hard problems are, in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pp. 331–337 (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).

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

[Huhns 91] Huhns, M. N. and Bridgeland, D. M.: Multiagent Truth Maintenance, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 6, pp. 1437–1445 (1991).

[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).

[横尾 92a] 横尾 真, E. H. Durfee, 石田 亨, 桑原 和宏：分散制約充足による分散協調問題解決の定式化とその解法, 電子情報通信学会論文誌, Vol. J-75 D-I, No. 8, pp. 704–713 (1992).

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

[Yokoo 95] Yokoo, M.: Asynchronous Weak-commitment Search for Solving Distributed Constraint Satisfaction Problems, in *Proceedings of the First International Conference on Principles and Practice of Constraint Programming (CP-95)*, pp. 88–102, Springer-Verlag (1995), Lecture Notes in Computer Science 976.

[横尾 96a] 横尾 真：柔軟で動的なエージェントの組織構造を用いた分散制約充足アルゴリズム, 人工知能学会誌, Vol. 11, No. 6, pp. 933–940 (1996).

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

[Yokoo 98a] Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K.: The Distributed constraint satisfaction problem: formalization and algorithms, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 5, pp. 673–685 (1998).

[横尾 98b] 横尾 真, 平山 勝敏：分散 breakout：反復改善型分散制約充足アルゴリズム, 情報処理学会論文誌, Vol. 31, No. 1, pp. 106–114 (1998).

〔担当委員：新田克己〕

1999年7月8日 受理

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年～1991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基礎研究所に勤務．制約充足問題，マルチエージェントシステム，分散人工知能に関する研究に従事．制約充足/分散制約充足，エージェントの合意形成メカニズム等に興味を持つ．博士(工学)．1992年度人工知能学会論文賞，1995年度情報処理学会坂井記念特別賞，1999年度人工知能学会全国大会優秀論文賞受賞．情報処理学会，日本ソフトウェア科学会，AAAI各会員．

平山 勝敏(正会員)

1967年生まれ．1990年大阪大学基礎工学部制御工学科卒業．1992年同大学院基礎工学研究科博士前期課程修了．1995年同博士後期課程修了．神戸商船大学商船学部助手を経て，1997年神戸商船大学商船学部講師．1999年10月よりカーネギーメロン大学客員研究員．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．博士(工学)．情報処理学会，日本ソフトウェア科学会，AAAI各会員．