論　文

Technical Papers

山登り法を用いた分散制約充足における組織化

An Organizing in Distributed Constraint Satisfaction with a Hill Climbing Method

平山　勝敏*　山田　誠二*　豊田　順一*
Katsutoshi Hirayama　Seiji Yamada　Jun'ichi Toyoda

*　大阪大学産業科学研究所
ISIR, Osaka University, Ibaraki, Osaka 567, Japan.

1993年11月11日　受理

Keywords: dynamic organizational structure, distributed constraint satisfaction problem, hill climbing.

Summary

DCSP (Distributed Constraint Satisfaction Problem) provides a formal framework for studying cooperative distributed problem solving. Several algorithms for DCSP have been proposed. This paper presents a new method, LMO (Local Minimum driven Organizing), to solve DCSP.

The basic algorithm is iterative improvement. Each agent measures the cost of its own instantiations as the number of violated constraints. If local change of instantiations reduces the cost, the agent performs hill-climbing locally. The advantage of this method is that agents solve their local problems in parallel. One drawback, however, is the possibility of getting caught in local minima.

LMO provides a technique for escaping from local minima. It is summarized as follows. When an agent (A1) gets caught in a local minimum, 1) A1 sends its local CSP to the agent (A2) that shares a violated constraint with it, 2) A2 puts their CSPs together and solve them by a brute-force search. It eliminates local minima from search spaces, alters a hill-climbing for a brute-force search, and reduces communication overhead at the cost of the parallel execution. It is triggered by a local minimum, therefore the more local minima agents get caught, the more CSPs are put together and solved with a brute-force search. This means that the algorithm is adaptable to the problem it solves. In this paper, we verify this fact experimentally.

1. はじめに

現在，分散協調問題解決における組織に関する研究は，固定的な組織構成から動的な組織構成に主眼が移りつつある．従来の組織の研究が，問題解決過程と独立に設定された組織の比較という内容が主であったのに対し[Durfee 87, 大沢 92]，近年，問題解決過程において組織を再編するというアプローチがとられている[Decker 93, Ishida 90, Steels 90]．その動機は，単一の組織が常に最適ではないという直観と，組織形成のメカニズムは何かという興味の2点にあると思われる．本研究の動機もこの2点にある．

我々は，分散制約充足問題(DCSP: Distributed Constraint Satisfaction Problem)を，大域的な情報を持たない複数エージェントが動的に組織を形成しながら解く方法を提案する．具体的には，各エージェントが制約矛盾数を評価関数として山登り法を実行し，局所最適解に陥るとほかの一つのエージェントと結合し局所的にDCSPを解く．本研究では，これをエージェントの組織化とみなしLMO (Local Minimum driven Organizing)と呼ぶ．LMOの第一の目的は，組織形成メカニズムを探ることにあり，DCSPを解く効率的なアルゴリズムを提案することではない．

本論文では，まず，LMOのアルゴリズムとその効果について説明する．次に，他の組織化との実験的な比較から，LMOがDCSPの難易度に適応することを示す．

2. 制約充足問題(CSP)

CSP (Constraint Satisfaction Problem)は，変数 v_i $(i=1, \cdots, n)$，値域 D_i $(i=1, \cdots, n)$，制約 C_j $(j=1, \cdots, m)$ の3要素より構成される[Kumar 92]．変数 v_i は，そのとり得る値の集合として値域 D_i を持つ．制約 C_j は特定の値域の集合 $D_{j1}, \cdots, D_{jk}$ の直積 $D_{j1} \times \cdots D_{jk}$ 上の部分集合で，変数に許されている制約条件を表す．CSPを解くとは，すべての C_j を満たす各変数 v_i の値 d_i を求めることである．以下，制約は特定の2変数間のみに存在すると仮定する．このようなCSPは変数を節点，制約を枝とする無向グラフで表現でき，これを制約ネットワークと呼ぶ．

3. 分散制約充足問題(DCSP)

DCSP (Distributed Constraint Satisfaction Problem)は，変数と制約が複数エージェントに分散されたCSPであり，分散協調問題解決を形成的に記述できる．その定式化および通信モデルは以下のとおりである[Yokoo 92]．

〈定式化〉

エージェントの集合 $\{1, \cdots, n\}$ が存在する．各変数 v_i に対して，それが属するエージェント k が定義され，これをbelongs(v_i, k) で表す．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 k が制約 C_j を知っていることをknown(C_j, k) と書く．次のときDCSPが解けたという．

すべてのエージェント k において，$\forall v_i$belongs(v_i, k) なる変数 v_i の値が d_i に決定され，かつ，$\forall C_j$known(C_j, k) なる制約が $v_1=d_1, \cdots, v_n=d_n$ のもとで真となる．

〈通信モデル〉

- エージェント間通信はメッセージ通信である．
- エージェントは，アドレスを知るエージェントにメッセージを送信できる．
- メッセージ遅延時間は有限だが上限はわからない．
- 任意の2エージェント間では，送信メッセージの順序は保存される．

本研究では，変数が，初期状態において異なる1エージェントに属し，かつ，エージェントは自分に属する変数に関する制約のみを知っているとする．また，自分と制約を共有するすべてのエージェント(近傍)のアドレスを知っているとする．このモデルにより，制約ネットワークの節点(変数)をエージェント，枝(制約)を通信回線と見た論理的な通信ネットワークを想定することができる．図1にDCSPの例を示す．

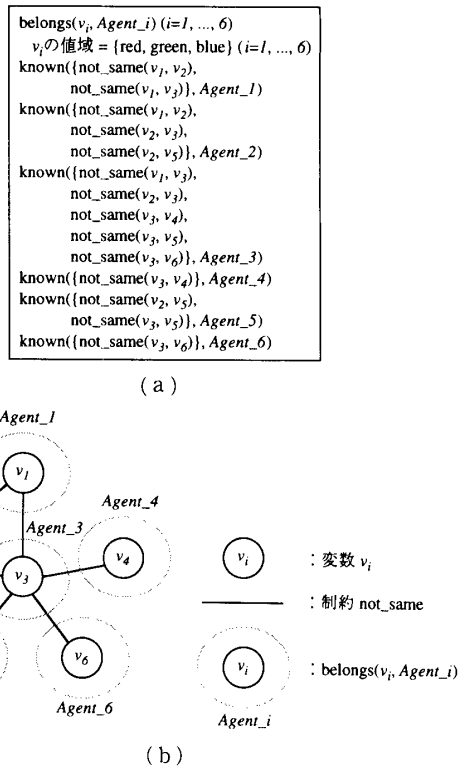

図1 DCSPの例

4. 山登り法を用いた分散制約充足

まず，山登り法を用いた分散制約充足について説明する．

4・1 山登り法

まず，各エージェントは自分が持つ変数に任意の値を代入し，近傍に送信する．以降，変数の値を具体化と呼ぶ．エージェントは近傍の具体化をもとに，自分の現在の具体化の制約矛盾数，具体化を変更した場合の制約矛盾数の最小値を求める．ここで，{具体化，現在の制約矛盾数，制約矛盾数の最小値}の3つ組のデータを *state* と呼ぶ．以降，エージェントは *state* を *state* メッセージとして交換しあい，近傍の *state* の集合を *state_view* として蓄える．山登り法の基本的な動作は，各エージェントが，自分の制約矛盾数を下げることができるならば具体化を変更する，という操作を繰り返すことによって進行する．具体化の変更に際しては，最小矛盾ヒューリスティクス[Minton 90]を用いて最も制約矛盾数が減少する値に変更する．

この方法は，エージェント内部の処理が単純，かつ，

大域的な情報がいらないという利点があるが*1，局所最適解という問題がある．詳細は5・1節で述べる．

4・2 交　　渉

全エージェントが非同期に山登り法を実行すると，無限ループに入ることがある[平山 93]．これを避けるためにエージェントは，具体化を変更する前に近傍と交渉を行う．エージェントが具体化を変更できるのは，交渉を行って，近傍の全エージェントから承認のメッセージを受信した場合である．エージェントが近傍から承認されるのは，自分の制約矛盾数の減少幅が，近傍 ∪{自分}で唯一最大か，または，最大のエージェントのなかで自分の識別子が最も小さい場合である．交渉は次の手順で実行される．

〈交渉アルゴリズム〉

Step 1：エージェントは *state_view* を見て，自分が矛盾する制約を持ち，かつ，自分の制約矛盾数の減少幅が近傍 ∪{自分}で最大だと判断したときに交渉を開始する．交渉は，{交渉識別子，現在の制約矛盾数，制約矛盾数の最小値}を近傍に送信することで始まる(*negotiate* メッセージ)．

Step 2：*negotiate* メッセージを受信したエージェントは，以下の条件が一つでも成立するなら，相手に承認のメッセージを送信し，どれも成立しなければ，不承認のメッセージを送信する(*reply* メッセージ)．

1)　自分の現在の制約矛盾数が0である．
2)　*negotiate* メッセージを送信したエージェントの制約矛盾数の減少幅が，自分の減少幅よりも大きい．
3)　互いの制約矛盾数の減少幅が同じで，送信エージェントの識別子が自分の識別子よりも小さい．

交渉相手すべてから承認されればエージェントは具体化を変更できるが，交渉中に，他エージェントから *state* メッセージを受信するか，他エージェントの交渉に対し自分が承認のメッセージを送信した場合には，その交渉を打ち切る．

交渉により，あるエージェントが具体化を変更するとき，近傍の具体化変更が禁止される．しかし，近傍以外のエージェントはその限りでないため，全体として具体化変更は並列に実行される．

*1　ただし，4・2節で示す交渉時の非決定性をなくすため，エージェントに固有の識別子(番号)が付けられている．

5. 組　織　化

複数エージェントで分散して難しいDCSPを解く場合，具体化の変更回数が増え，それに伴って通信量が増大する．この場合，分散しているエージェントが集まって解くことにより組織化すれば，具体化の変更とそれに伴う通信量が節約できると考えられる．しかし，やさしいDCSPを解く場合，集まるのにかかるコストがその節約分よりも大きくなると考えられる．よって，問題の難易度に応じた組織化が必要である．また，ここで重要なことは，DCSPでは，個々のエージェントは大域的な情報を持たず，局所的な情報に基づいて組織化しなければならないことである．

本研究では，DCSPの難易度に応じた組織化としてLMO (Local Minimum driven Organizing)を提案する．これは，あるエージェントが局所最適解に陥ると他エージェントと結合し局所的にDCSPを解く方法である．局所最適解は個々のエージェントが識別可能であり，大域的な情報を必要としない．また，潜在的な局所最適解の数は山登り法にとっての問題の難易度に対応すると考えられるため，局所最適解に陥るごとに組織を形成することは，問題の難易度に応じた組織化を実現しているといえる．なお，本論文では，「問題の難易度」をすべてこの「山登り法にとっての問題の難易度」の意味で用いる．

5・1 処 理 の 概 要

まず，局所最適解を定義する．

【局所最適解】

次の3条件がすべて成立することを知るとき，そのエージェントは局所最適解にいる．

1)　自分の現在の制約矛盾数が0でない．
2)　自分が具体化を変更しても，自分の制約矛盾数が減少しない．
3)　自分の近傍が具体化を変更しても，近傍の制約矛盾数が減少しない．

本研究では，各エージェントは自分と近傍の情報しか知らないため，自分と近傍の範囲で局所最適解を定義する．エージェントは，上の3条件が成立することを4・2節の交渉の結果知ることができる．つまり，条件1)と2)が成立するエージェントが交渉を行って，近傍から承認されれば条件3)も成立する．

次に，組織化のアルゴリズムを説明する．

〈組織化〉

Step 1：局所最適解にいるエージェントが，自分の

持つCSP(変数，値域，制約)と近傍に関する情報を，自分と矛盾制約を共有する任意の1エージェントに送信する(*organize* メッセージ).

step 2：*organize* メッセージを受け取ったエージェントは，自分のCSPと近傍に関する情報を次のように更新する．ここで，*organize* メッセージを送信したエージェントを依頼エージェント，受信したエージェントを更新エージェントと呼ぶ．

- **変数**：依頼エージェントと更新エージェントのそれぞれの変数の和集合を新たな変数とする．
- **値域**：依頼エージェントと更新エージェントのそれぞれの変数と値域，および，両者が共有する制約で構成されるCSPを単純バックトラッキング[Kumar 92]で全解探索し，得られた解すべてを新たな値域とする．
- **制約**：依頼エージェントと更新エージェントのそれぞれの制約の和集合から，両者が共有する制約を除いた集合を新たな制約とする．
- **近傍に関する情報**：依頼エージェントと更新エージェントの近傍，*state_view* それぞれの和集合から，依頼エージェントと更新エージェントに関する部分を除いた集合を，新たな近傍，*state_view* とする．

また，依頼エージェントは，*organize* メッセージを送信すると同時に，*address* メッセージを更新エージェント以外の近傍に送信し，自分のアドレスが変更されることを知らせる．

組織化後は，更新エージェントが山登り法を継続する．山登り法と組織化のアルゴリズムをまとめて**付録**に掲載する．

5・2 実　行　例

図1(a)のようなDCSPを考える．制約not_same(v_1, v_2)は変数v_1とv_2の値が異なることを表す．制約ネットワークおよび論理的な通信ネットワークは図1(b)のように表現される．以降，図2をもとにLMOを説明する．

各エージェントが図2(a)のように初期化して近傍に具体化を送信し，*state* および *state_view* を構築したとする．このとき各エージェントの *state* は，図2(a)中の表のようになり，交渉の結果，*Agent_3* の制約矛盾数の減少幅が2であり，これは近傍∪{自分}で唯一最大なので，*Agent_3* がv_3の具体化をgreenに変更し，近傍に *state* メッセージを送信する．その結果，図2(b)のようになる．

このとき *Agent_2* と *Agent_3* が局所最適解にい

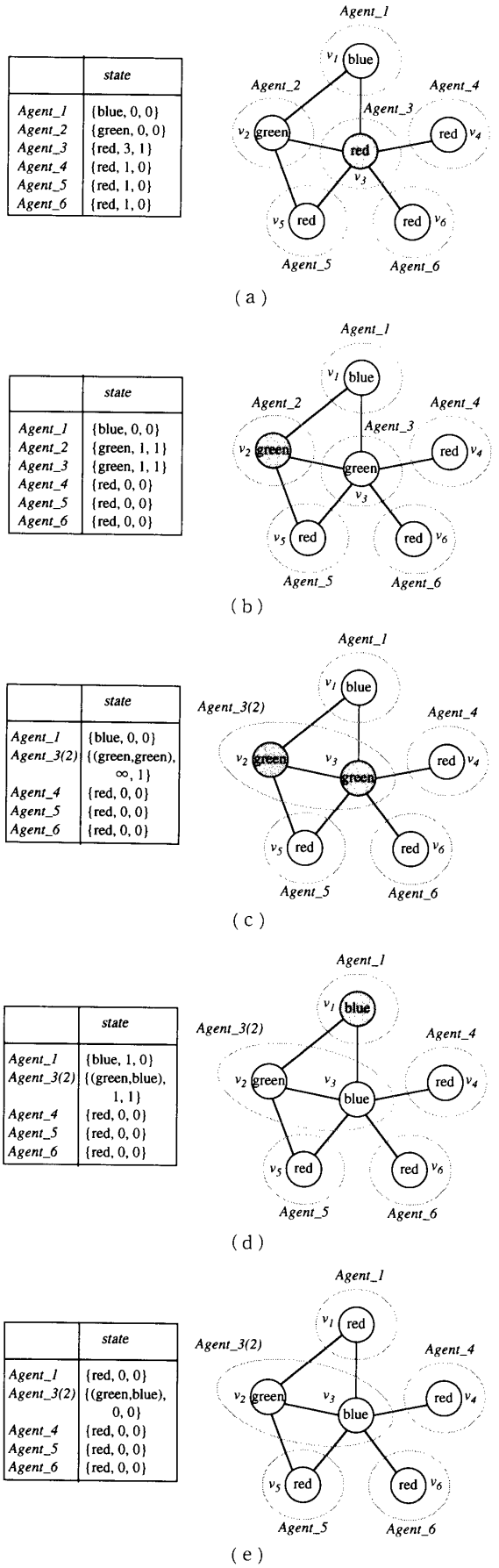

図2 具体例(注：*state*＝{具体化，現在の制約矛盾数，制約矛盾数の最小値})

るが，交渉の結果，*Agent_2* が依頼エージェント，*Agent_3* が更新エージェントとなって組織化する．このとき，*Agent_3* は変数 (v_2, v_3) の値域を単純バックトラッキングにより求める．その結果，図2(c)のようになる．表中の ∞ とは，現在の具体化が値域にない場合の制約矛盾数を表す．

交渉の結果，*Agent_3* が (v_2, v_3) の具体化を (green, blue) に変更し，近傍に *state* メッセージを送信する．その結果，図2(d)のようになる．次に，*Agent_1* が v_1 の具体化を red に変更し，近傍に *state* メッセージを送信して，図2(e)のようになる．以上で全エージェントの制約が充足され，DCSP が解かれた．

5・3 組織化の効果

組織化は以下のような効果を持つ．

(1) 局所最適解の除去

局所最適解に陥ったときに矛盾している制約を解くため，同じ局所最適解に2度と陥ることはない．さらに，解いた制約を矛盾する制約として含む別の局所最適解を除去する．

(2) 探索方法の部分的切換え

問題に多くの局所最適解がある場合，組織化の回数が増え，多くのエージェントが持つ CSP が結合される．その結果，DCSP において局所的に単純バックトラッキングによって解かれる部分が局所最適解の数につれて増えていく．すなわち，DCSP を解く探索方法が部分的に切り換わり，かつ，切り換わる部分が問題の難易度に応じて増減する．

(3) 最悪時通信量，並列性の減少

組織化により制約ネットワークは縮退する．また，*address* メッセージでアドレスの変更を知らせることにより論理的な通信ネットワークも縮退する．我々の解析では，山登り法の最悪時の通信量は通信ネットワークの節点数と枝数の多項式オーダになるので，通信ネットワークの縮退とともに山登り法の最悪時の通信量は減少する．

しかし，エージェントの近傍が拡大するため，**4・2 節**から明らかなように処理の並列性は減少する．通信量と並列性は，互いにトレードオフの関係にあると考えられるが，組織化には，並列性を犠牲にして通信量を減少させる効果があると考えられる．

また，DCSP を解くアルゴリズムとして次の二つの特長がある．証明は[平山 93]を参照．

(4) 健全性

安定状態(十分時間が経過しても，具体化を変更するエージェントも組織化するエージェントも現れない状態)に達したとき，全エージェントは充足状態にある*2．

(5) 完全性

DCSP が有限(エージェントの総数が有限，かつ，各エージェントが持つ CSP が有限)である限り，有限時間内に全エージェントが充足状態になるか，あるいは，全エージェントの制約を充足する解がないことがわかって停止する．

6. 実　　験

エージェントの並行動作を実現するためのシミュレータを構成し，そのうえで実験を行った．シミュレータでは，各エージェントが独自の時計を持ち，処理を行うたびにその時間を進める．また，タイムマネージャという仮想エージェントが全体の時間の進行およびメッセージの送受信を管理する．

各エージェントは制約チェックを10回行うたびに時計を1単位時間(1ステップ)だけ進め，その時計が示す時刻を見てタイムマネージャがそのエージェントを起動するか否かを決定する．また，エージェント間のメッセージの送受信はすべてタイムマネージャを通じて行われ，タイムマネージャが一時的にメッセージを保持することにより通信遅れが導入される．今回の実験では，通信遅れはすべて1ステップとした．

6・1 実 験 方 法

LMO を評価するために，解を得るまでのステップ数に関して次の四つの組織化を比較する．

- O-a：LMO
- O-b：組織化なし
- O-c：集中型の組織化(Forward checking)
- O-d：集中型の組織化(単純バックトラッキング)

O-a は本研究で提案した手法である．なお，この実験では一つのエージェントに多くの CSP が集まることを防ぐために，組織化後に更新エージェントの近傍が，それと共有する制約の重みを上げるというヒューリスティクスを用いた．O-b は，少なくとも一つのエージェントが局所最適解に陥ると全エージェントが初期値をランダムに変えて，山登り法をやり直す．O-c と O-d は，初期値が解でない場合，全エージェントの CSP をリーダと呼ばれる一つのエージェントに集めて解く．リーダの選択については，分散アルゴリズム

*2 ただし，現在のアルゴリズムは，個々のエージェントは安定状態を判別する手段を持たない．

の分野でいくつかの方法が提案されているが[萩原90],各組織化が使用するメッセージのオーダを等しくするために，LMOにおいて更新エージェントの値域の更新をすべてのCSPが集まるまで遅延する方法を用いる．また，リーダがCSPを解く手法として，O-cはForward checking，O-dは単純バックトラッキングを想定した．二つともCSPの一般的解法であり，経験的にはForward checkingのほうが効率が良い．

対象とする問題は3色問題と弱3色問題である．3色問題とは，互いに隣りあう領域は異なる色で塗るという制約のもとで各領域を3色(ここでは，赤，青，緑)で塗り分けるという問題である．また，弱3色問題とは，やさしい問題として用意したもので，互いに隣りあう領域どうしを赤で塗らないという制約のもとで各領域を3色で塗り分ける問題である．これら2種類の問題それぞれにつき，次の18の問題例を作成した．ただし，mは制約の数，nは変数の数とする．密な問題，疎な問題の定義は[Adorf 90]に基づく．

(a) 密な問題 ($m=n(n-1)/4$)
  (1) $m=23$, $n=10$, (2) $m=95$, $n=20$, (3) $m=218$, $n=30$ をそれぞれ3例(計9例)

(b) 疎な問題 ($m=2n$)
  (1) $m=20$, $n=10$, (2) $m=40$, $n=20$, (3) $m=60$, $n=30$ をそれぞれ3例(計9例)

実験では，36の問題例それぞれについて500個の異なる初期値を作成し，計18 000問を各組織化を行うエージェントの集団に与え，解が得られるまでのステップ数を求めた．組織化の評価は，異なる初期値500問ステップ数の平均(平均ステップ数)で行う．また，(O-a)以外については，処理時間の上限を100nステップとし，これを超えたときには打ち切った．打ち切った場合のステップ数は100nステップとして計算する．

図3～図5に結果を示す．グラフのプロットは，各組織化で解いた一つの問題例Pに対応している．プロットのx座標は，問題例Pに対するO-bの平均ステップ数であり，大まかに潜在的な局所最適解の数(問題の難易度)を表す．プロットのy座標は，問題例Pに対する各組織化の平均ステップ数をO-bのそれで割った値(平均ステップ数の比の値)である．すなわち，y座標の値が1を下回ると，対応する組織化は，その問題例をO-bよりも速く解いたことになる．

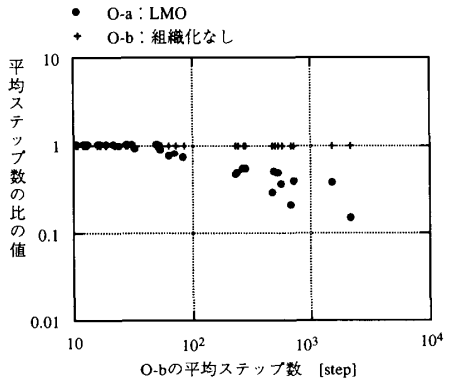

図3 O-aとO-bの比較

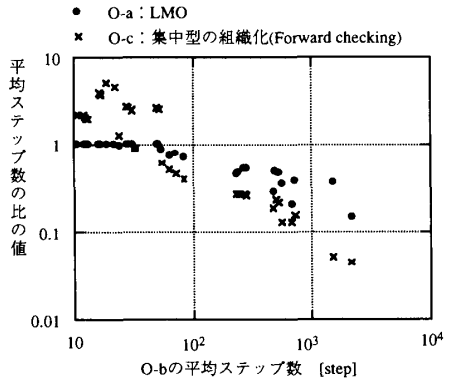

図4 O-aとO-cの比較

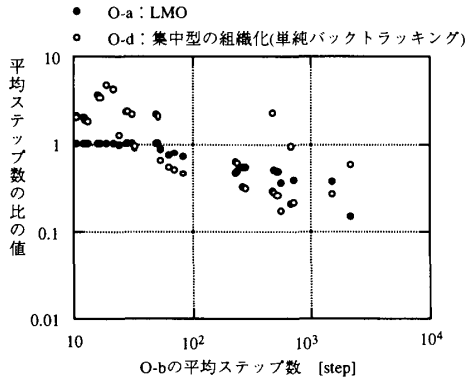

図5 O-aとO-dの比較

6・2 考　　察

〔1〕 O-aとO-bの比較

図3より，グラフの右部分では，O-aはO-bよりもかなり良く，それは右にいくほど顕著である．つまり，局所最適解が多い難しい問題に対して，LMOの局所最適解を取り除く効果がでていると考えられる．

また，グラフ左部分の局所最適解の少ない問題に対しては両者の優劣は見られない．

〔2〕 O-aとO-cの比較

図4より，グラフの右部分の局所最適解が多い問題に対しては，O-aはO-cよりも若干効率が悪い．しかし，グラフの左部分の局所最適解の少ない問題に対しては，O-aはO-cよりも効率が良い．

O-aは制約充足の結果，局所最適解に陥ったときに組織化するのに対し，O-cは，制約充足の結果に関係なく，前もって組織化する．つまり，O-cは，問題が局所的かつ並列的に解決できる可能性を無視して組織

化する．これが，局所最適解が少ない場合のO-cの効率の悪さの原因と考えられる．

〔3〕 **O-aとO-dの比較**

図5より，O-aとO-dにおいても，局所最適解が少ない問題に対してO-aのほうが効率が良い．その原因は〔2〕項で述べたことと同じと考えられる．また，別の特徴として，O-dはO-aに比べばらつきが大きい．これは，大きなCSPに対する単純バックトラッキングの効率が変数順序(variable ordering)の影響を受けやすいことが原因だと考えられる．

それに対し，O-aでは，各エージェントは組織化の際に単純バックトラッキングをしても，解くべきCSPが小さいため，この影響が小さい．

以上をまとめると，組織化しないことは問題が難しい(局所最適解が多い)場合に不利であり，集中型の組織化は問題がやさしい(局所最適解が少ない)場合に不利である．一方，LMOは，問題が難しいときは集中型の問題解決，やさしいときは組織化しない局所型の問題解決を行うため，両者の長所を併せ持つといえる．

7. 関 連 研 究

DCSPを解くアルゴリズムとして，横尾らは非同期型バックトラッキングを提案し[Yokoo 92]，二つの効果的なヒューリスティクスを導入している[Yokoo 93]．アルゴリズムとしての効率の追及はLMOの本来の目的ではないが，彼らのアルゴリズムとの効率面での比較は今後の課題である．

また，DCSPを情報収集型分散アルゴリズムで解くことが可能である[萩原 90]．これは実験での集中型の組織化に相当する．6•2節でも示したように，LMOによる解法のほうが，問題の難易度に適応できる点で優れている．

Landerら[Lander 93]は，分散探索においてNegotiated searchを提案している．これは競合を探索の制御に利用する点で本研究に近いが，時間の経過とともに制約を緩和するため完全性が保証されない．

組織化の研究として，石田ら[Ishida 90]は分散プロダクションシステムに組織の概念を導入し，実時間問題解決を行うエージェントが動的に組織を再編するというアプローチをとっている．そこでは，局所的な統計情報と組織に関する統計情報を用いて組織が再編される．しかしながら後者は，エージェント全体に関する大域的な情報であり，局所的な情報のみを用いて組織化する我々のアプローチとは異なる．

8. まとめと課題

動的な組織構成の研究の一つとして，山登り法を用いた分散制約充足における組織化を提案した．これは，局所最適解に陥ったエージェントが他エージェントと組織化するというアプローチであり，問題の難易度に適応する性質を持つ．この方法の現状における問題点は，組織化の回数が増え一つのエージェントに多くのCSPが集まったときに，それを全解探索するコストが指数関数的に増大することである．これを避けるためには，①全解探索を一解探索にする，②一つのエージェントに多くのCSPが集まらないよう適当に負荷を分散する，などが考えられる．今後の課題としては，組織化時の探索方法の検討，負荷分散，個々のエージェントの能力を考慮したLMOの制御，他のアルゴリズムとの比較などがあげられる．

謝　　辞

多くの貴重なコメントをいただいたMARK(Multi-Agent Research community in Kansai)の皆様，ならびに豊田研究室の皆様に感謝します．また，論文を送っていただいたNTTコミュニケーション科学研究所の横尾　真氏に深く感謝します．さらに，的確な指摘を下さった査読者の方に感謝します．

◇参 考 文 献◇

[Adorf 90] Adorf, H. M. and Johnston, M. D.: A Discrete Stochastic Neural Network Algorithm for Constraint Satisfaction Problems, *Int. Joint Conf. on Neural Networks, III,* pp. 917-924(1990).

[Decker 93] Decker, K. and Lesser, V. R.: An Approach to Analyzing the Need for Meta-Level Communication, *IJCAI-93,* pp. 360-366(1993).

[Durfee 87] Durfee, E. H., Lesser, V. R. and Corkill, D. D.: Coherent Cooperation Among Communicating Problem Solvers, *IEEE Transactions on Computers,* Vol. 36, No. 11 pp. 1275-1291(1987).

[萩原 90] 萩原：分散アルゴリズム，人工知能学会誌，Vol. 5, No. 4, pp. 430-440(1990).

[平山 93] 平山，山田，豊田：山登り法を用いた分散制約充足における組織化，情処学会人工知能研資，93-AI-90, pp. 23-32(1993).

[Ishida 90] Ishida, T., Yokoo, M. and Gasser, L.: An Organizational Approach to Adaptive Production Sys-

tems, *AAAI-90,* pp. 52-58(1990).
[Kumar 92] Kumar, V.: Algorithms for Constraint Satisfaction Problems: A Survey, *AI Magazine,* Vol. 13, No. 1, pp. 32-44(1992).
[Lander 93] Lander, S. E. and Lesser, V. R.: Understanding the Role of Negotiation in Distributed Search Among Heterogeneous Agents, *IJCAI-93,* pp. 438-444(1993).
[Minton 90] Minton, S., Johnston, M. D., Philips, A. B. and Laird, P.: Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method, *AAAI-90,* pp. 17-24(1990).
[大沢 92] 大沢，沼岡，石田：分散人工知能小問題集，マルチエージェントと協調計算 I，近代科学社(1992).
[Steels 90] Steels, L.: Cooperation Between Distributed Agents Through Self-Organization, *Decentralized AI,* North-Holland Pub., Amsterdam(1990).
[Yokoo 92] Yokoo, M., Durfee, E. H., Ishida, T. and Kuwabara, K.: Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving, *12th IEEE Int. Conf. on Distributed Computing Systems,* pp. 614-621(1992).
[Yokoo 93] Yokoo, M.: Dynamic Variable/Value Ordering Heuristics for Solving Large-Scale Distributed Constraint Satisfaction Problems, *12th Int. Workshop on Distributed Artificial Intelligence*(1993).

〔担当編集委員・査読者：久野　巧〕

◇ 付　　録 ◇

メッセージを受信したときに起動されるエージェントのアルゴリズムを示す．エージェントは，*state*, *negotiate*, *reply*, *organize*, *address* の5種類のメッセージを使用する．

```
when agent_i received state(sender's id, sender's state) do
   close current_negotiation;
   update state_view;
   if sender's current_value was changed then
      new_value←the value of which cost is the lowest;
      minimum_cost←the cost of new_value;
   endif;
   send state(id, state) to neighbors;
   if (current_cost is not zero)
      and (state_view indicates that none of its neighbors reduces more cost than agent_i)
then
      send negotiate(id, negotiation_id, current_cost, minimum_cost) to neighbors;
      current_negotiation←negotiation_id;
      members_list←the ids of neighbors;
   endif;
enddo;

when agent_i received negotiate(sender's id, sender's negotiation_id, sender's current_cost,
sender's minimum_cost) do
   if (current_cost is zero)
      or (sender will reduce more cost than agent_i)
      or (sender and agent_i will reduce the same costs
         and sender's id is smaller than id) then
      send reply(id, sender's negotiation_id, True) to sender;
   else
      send reply(id, sender's negotiation_id, False) to sender;
   endif;
enddo;

when agent_i received reply(sender's id, sender's negotiation_id, Ans) do
   if current_negotiation = sender's negotiation_id then
      if Ans = True then
         remove sender's id from members_list;
         if members_list = NULL then
            do action;
         endif;

      else
         close current_negotiation;
      endif;
   endif;
enddo;

when agent_i received organize(sender's id, sender's CSP, sender's neighbors, sender's
state_view) do
   update CSP, neighbors and state_view;
   current_value←NULL;
   current_cost←∞;
   new_value←the value of which cost is the lowest;
   minimum_cost←the cost of new_value;
   send negotiate(id, negotiation_id, current_cost, minimum_cost) to neighbors;
   current_negotiation←negotiation_id;
   members_list←the ids of neighbors;
enddo;

when agent_i received address(sender's id, old_address, new_address) do
   change old_address to new_address in neighbors;
enddo;
procedure action {
   if current_cost = minimum_cost then
      who←one of the agents who share the violated constraints;
      forward←who;
      send address(id, old_address, new_address) to neighbors without who;
      send organize(id, CSP, neighbors, state_view) to who;
   else
      current_value←new_value;
      current_cost←minimum_cost;
      send state(id, state) to neighbors;
   endif;
enddo;
```

著 者 紹 介

平山　勝敏(正会員)

1990年大阪大学基礎工学部卒業，1992年同大学院修士課程修了．現在，同大学院博士課程存学中．分散AI，制約充足などに興味を持つ．

山田　誠二(正会員)

1984年大阪大学基礎工学部卒業，1989年同大学院博士課程修了．同年，基礎工学部システム工学科助手．1991年より大阪大学産業科学研究所講師，現在に至る．工学博士．人工知能，特に，効率化学習，即応プラニング，マルチエージェント系に興味を持つ．情報処理学会，日本認知科学会，日本ソフトウェア科学会，AAAI，IEEE各会員．

豊田　順一(正会員)

1961年大阪大学工学部卒業，1966年同大学院博士課程単位取得退学．同年，大阪大学基礎工学部助手．1969年助教授．1982年大阪大学産業科学研究所教授．工学博士．現在，概念の形成，Visual fidelity，マニュアルのわかりにくさに関する研究に従事．情報処理学会，電子情報通信学会，日本認知科学会各会員．