人　工　知　能 90－4
（１９９３．９．２８）

山登り法を用いた分散制約充足における組織化

平山勝敏　山田誠二　豊田順一

(大阪大学産業科学研究所)

E-mail：hirayama@ai.sanken.osaka-u.ac.jp

分散人工知能の組織に関する研究の１つとして，分散制約充足問題(DCSP)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．DCSPは，分散人工知能の問題を形式的に記述できる枠組みであり，その上での組織形成に関する議論には，かなりの一般性が期待できる．本稿では，まず，組織形成方法として，LMO(Local Minimum driven Organization)について説明する．これは，エージェントが局所最適解に陥ったときに組織を形成するという方法である．また，個々のエージェントの処理から導かれるマクロな挙動の特徴として，健全性と完全性を証明する．最後に，エージェント全体が，問題の難易度に応じて組織を形成し，集団としての適応性があることを実験的に示す．

The Organizing in Distributed Constraint Satisfaction with a Hill Climbing Method

Katsutoshi Hirayama, Seiji Yamada, Jun'ichi Toyoda

ISIR, Osaka University

We propose a method to solve Distributed Constraint Satisfaction Problem(DCSP) in which agents solve their own problems by organizing. DCSP gives us a framework for Distributed Artificial Intelligence. Thus, implementing the organizing in DCSP makes it possible to discuss the problems of organization independent of specific domains. We present LMO(Local Minimum driven Organizing) in which the agents organize when they get caught in local minima. This paper describes agent's behaviors and shows emergent properties resulting from individual agents' behaviors. One property is completeness and soundness. We prove it analytically. The other is that the more difficult DCSP agents solve, the larger groups they organize, i.e. they adapt themselves to the degree of difficulty. For verifying this property, we compare several societies which organize differently. As a result, the society with LMO better than the others.

1. はじめに

現在，分散人工知能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組織を設定し，様々な組織を比較するという内容が主であったのに対し[1][2]，近年，問題解決過程において組織を動的に再編するというアプローチがとられている[3][4]．その動機は，単一の組織があらゆる場合に他の組織に勝るわけではないという直観と，なぜ組織が形成されるのか，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，他の1つのエージェントと結合し，局所的にDCSPを解く．この組織形成を，本研究ではLMO(Local Minimum driven Organizing)と呼ぶ．

DCSPは，分散人工知能の問題を形式的に記述できる枠組みである．DCSPを解くアルゴリズムにLMOを組み込むことは，アルゴリズムの効率化のみならず，分散人工知能における組織の問題を，一般的に議論できるという点で意義がある．

本稿では，まず，個々のエージェントの処理の詳細を示し，その結果得られるエージェント全体のマクロな挙動の特徴として，健全性と完全性ををを証明する．さらに，LMOを実行するエージェントの社会(ある組織形成方法をとるエージェントの集団)を，異なる組織形成方法をとる社会と実験的に比較し，前者が，DCSPの難易度に適応すること，すなわち，難しいDCSPに対しては，組織を形成し，易しいDCSPに対しては，組織を形成しないことを示す．

2. 制約充足問題(CSP: Constraint Satisfaction Problem)

CSP[5]は，変数v_i(i=1, ..., n)，値域D_i(i=1, ..., n)，制約C_j(j=1, ..., m)の3要素より構成される．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}, ..., D_{jk}の直積$D_{j1}\times\cdots\times D_{jk}$上の部分集合で，変数に与えられる制約条件を表わす．CSPを解くとは，任意のC_jを満たす各変数v_iの値d_i(具体化)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: Distributed Constraint Satisfaction Problem)

DCSP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，分散人工知能の問題を形式的に記述できる枠組みとして近年注目されている．その定式化および通信モデルは以下の通りである[6].

●定式化

エージェントの集合{1, ..., n}が存在し，各変数v_iに対して，その属するエージェントkが定義され，これをbelongs(v_i, k)で表わす．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．次の場合にDCSPが解けたと言う．

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

●通信モデル

・エージェント間通信は，メッセージ通信によりなされる．
・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，初期状態において，異なる変数がそれぞれ異なる1エージェントに属し，かつ，エージェントは自身に属する変数に関する制約のみを知っているとする．

4. 山登り法を用いた分散制約充足

まず，LMOがその上で実現される，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

山登り法を用いた分散制約充足では，各エージェントは適当な初期値から出発して，自身のもつ制約のうち矛盾している制約の数(制約矛盾数)を評価関数とし，それが減少するように具体化を変更する．このとき各エージェントは，近傍がもつ変数の具体化をメッセージ通信(*Instantiation*メッセージ)により知り，それをAgent_viewとして蓄えているとする．具体化の変更に際しては，最小矛盾ヒューリスティックス[7]により最も制約矛盾数が減少する値に変更する．なお，近傍とは自身と制約を共有する全てのエージェントの集合

である．
この方法は，エージェント内部の処理が単純，かつ，大域的な情報が要らないという利点がある[1]．しかし，局所最適解に陥るという問題がある．詳細は§5.3に示す．

4.2. 交渉

●交渉の目的

各エージェントは具体化を変更するに先だってその権利を近傍と交渉する．これは次のような無限ループを避けるためである．「Fig.1(a)において，Agent_1，Agent_2のそれぞれの制約矛盾数は1で，どちらかが値をblueに変えると，それぞれ0になるとする．ここで両方が同時に値をblueに変えるとFig.1(b)のように再び両方の制約矛盾数が1となる．Fig.1(b)においても同様で，以下(a)(b)間を無限に循環する」．交渉の結果，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抑制される．この結果，1エージェントが自身の制約矛盾数を減少させると全エージェントの制約矛盾数の総和が必ず減少する．詳しくは，§6の補題1で述べる．

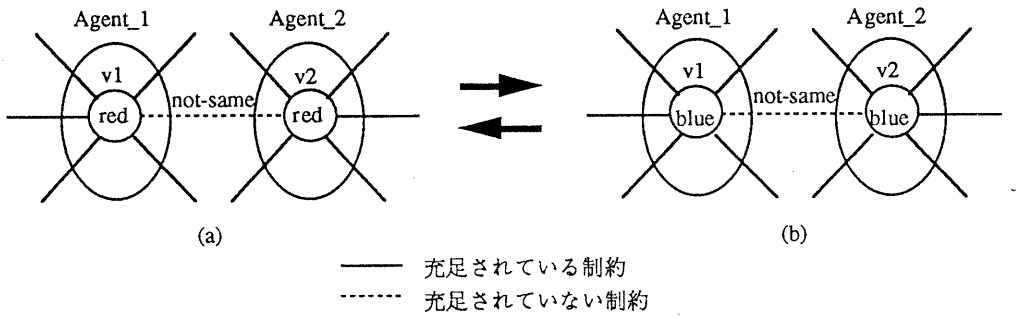

Fig. 1　無限ループ

●交渉の手順

交渉において，エージェントは近傍に，現在の制約矛盾数，具体化を変更した場合の制約矛盾数の最小値，識別子(エージェント固有の番号)を送信する．(*Negotiate*メッセージ)　近傍は，それぞれ送られてきた内容を自身のそれと比較して，次の3条件のうち，いずれかが成立するとき，そのエージェントに具体化変更の権利を与える．(Ans=Trueの*Reply*メッセージを送信する)

(1)自身の現在の制約矛盾数が零である．
(2)相手の現在の制約矛盾数とその最小値の差が，自身のそれより大きい．
(3)相手の現在の制約矛盾数とその最小値の差が，自身のそれと同じで，かつ，相手の識別子が，自身のそれより小さい．

エージェントが具体化を変更できるのは，近傍内の全エージェントから，Ans=Trueの*Reply*メッセージを受信したときに限る．なお，メッセージ数を減少させるため，各エージェントは近傍の現在の制約矛盾数，制約矛盾数の最小値，識別子をメッセージ通信(*Improve*メッセージ)により知り，それをImprove_viewとして蓄えているとし，次の2条件が共に成立するとき，交渉権を得て，交渉することができる．

(1)自身の現在の制約矛盾数が零でない
(2)自身の現在の制約矛盾数とその最小値の差が，近傍∪{自身}で最大

●交渉の解除

交渉は，次の場合に解除される．

(1)Ans=Falseの*Reply*メッセージを受信した場合
(2)*Instantiation*メッセージを受信した場合
(3)*Improve*メッセージを受信した場合
(4)他エージェントによる交渉に対して，Ans=Trueの*Reply*メッセージを送信した場合

交渉が解除されると，新たに交渉権を得て交渉を行わなければ，具体化を変更することはできない．

5. Local Minimum driven Organizing: LMO

5.1. 自律的な組織形成の必要性

---

[1]但し，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．

我々は，エージェントは動的に組織を形成すべきだと考える．DCSPにおいては，１つの組織が他の組織より常に優れているとは限らない．例えば，難しいDCSPを解くときは，できるだけ組織を形成して解いた方が効率的であり，また，易しいDCSPの場合は，組織形成にコストがかかるため，むしろ組織を形成せずに解決した方が効率的であると予想される．ゆえに，DCSPが難しい場合には，エージェントは組織を形成し，易しい場合には，組織を形成しないという枠組みが必要である．

本研究では，DCSPの難易度が，潜在的な局所最適解の数に反映されると考える．そこで，潜在的な局所最適解の数と組織形成を結ぶ手続きとして，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，他エージェントと組織を形成し，局所的にDCSPを解く方法である．ここでは，潜在的な局所最適解の数と，エージェントが実際に陥る局所最適解の数に相関があると仮定している．局所的にDCSPを解くことにより，以降，同一組織内のエージェント間では，無矛盾な具体化を通信コスト0で求めることができる．

5. 2. 処理の概要

山登り法を用いた分散制約充足には局所最適解という問題がある．局所最適解は，次のように定義される．

【定義】　局所最適解

次の３条件が共に成立するとき，そのエージェントは局所最適解にいる．

(a)自身の現在の制約矛盾数が零でない．

(b)具体化を変更しても自身の制約矛盾数が減少しない．(現在の制約矛盾数=制約矛盾数の最小値)

(c)近傍のエージェントが具体化を変更しても，その制約矛盾数が減少しない．　□

(c)が成立していない場合，近傍内のエージェントの具体化変更により，自身の現在の制約矛盾数が変更される可能性がある．よって，この場合は，局所最適解ではないと定義する．本研究では，各エージェントは，自身以外に近傍に関する情報のみを知ると仮定しているため，エージェントが局所最適解にいるかどうかの判定も近傍の範囲で行う．

次に，LMOの処理を示す．

【LMOの処理】

局所最適解にいるエージェントが，自身のもつCSP(変数，値域，制約)と近傍に関する情報を，自身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)．*Organize*メッセージを受け取ったエージェントは，自身のCSPと近傍に関する情報を次のように更新する．ここで，*Organize*メッセージを送信するエージェントを依頼エージェント，*Organize*メッセージを受信するエージェントを更新エージェントと呼ぶ．

●変数：依頼エージェントの変数と更新エージェントの変数の和集合を新たな変数とする．

●値域：依頼エージェントの変数と値域，更新エージェントの変数と値域，依頼エージェントと更新エージェントが共有する制約で構成されるCSPを，盲目的探索により全解探索し，得られた解を新たな値域とする．

●制約：依頼エージェントの制約と更新エージェントの制約の和集合から，両者が共有する制約を引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，Agent_view，Improve_viewそれぞれの和集合をとり，その共通部分を引いたそれぞれの集合を，新たな近傍，Agent_view，Improve_viewとする．

なお，エージェントは，自身が局所最適解にいることを，交渉により知ることができる．LMO後は山登り法を継続する．以上の処理の具体例は，[8]に示されている．

5. 3. LMOの効果

LMOにより，複数エージェントがもつCSPが結合され，それらが全解探索されてDCSPが局所的に解かれる．これにより，今後，更新エージェントは，解かれた制約に矛盾する値に具体化を変更することはない．現在，局所最適解に陥ったときに矛盾している制約を解いているため，少なくとも同じ局所最適解に二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にも陥ることはない．つまり，LMOには，探索空間から局所最適解を除去する効果がある．

また，問題に多くの局所最適解がある場合，LMOの回数が増え，多くのエージェントがもつCSPが結合される．これは，DCSPにおいて１箇所で盲目的探索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ことを表している．このことは，直観的に次のように解釈できる．

”問題が難しいほど，多くのエージェントが集まって確実な方法で解く”

これは，エージェントの集団が，問題の難易度に適応する性質をもつことを表わす．この性質は，陽に記述されたものではなく，個々のエージェントの振舞いから得られるものである．

6. 特徴

各エージェントは，次の場合にアルゴリズムを起動する．

(1)Agent_viewが十分でない，つまり，近傍のいずれかのエージェントの具体化を知らない場合

(2)他エージェントからのメッセージを受信した場合
(1)の場合，具体化を知らないエージェントに対し，*Request*メッセージを送信する．(2)の場合は，各メッセージについて，それぞれのアルゴリズムを起動する．詳細は付録に掲載する．以上から，エージェント全体のマクロな挙動に関する２つの定理が導かれる．まず，２つの補題を証明する．

《補題1》
1エージェントが具体化を変更すると，全エージェントの制約矛盾数の総和が必ず減少する．
〈証明〉
複数エージェントが，同時に同じ制約を修正する場合，厳密には，互いに相手の*Instantiation*メッセージを受信する前に具体化を変更し，同じ制約を修正する場合，かつその場合に限り，全エージェントの制約矛盾数の総和が減少しない．よって，付録のアルゴリズムのもとで，以下を仮定して矛盾を導く．
互いに他の近傍に属するエージェントAとBが，それぞれ具体化を変更し，*Instantiation*メッセージ(それぞれ*Ins_a*，*Ins_b*とする)を送信するとき，次の(1)かつ(2)が成り立つと仮定する．
(1)*Ins_a*がBに到着するのは，Bが*Ins_b*を送信した後である．
(2)*Ins_b*がAに到着するのは，Aが*Ins_a*を送信した後である．
AとBは具体化を変更し，それぞれ*Ins_a*と*Ins_b*を送信する．付録のアルゴリズムによれば，AとBでは，これ以前に以下のことが起きている．
(エージェントA)
(i)*Ins_a*の交渉中に，Bから，*Ins_a*に対するAns=True の*Reply*メッセージ(*Rep_a*とする)を受信した．(付録(D)の(2))
(ii)交渉を解除して，BにAns=True の*Reply*メッセージ(*Rep_b*とする)を送信した．(付録(C)の(4)(5))
(エージェントB)
(iii)*Ins_b*の交渉中に，Aから，*Rep_b*を受信した．(付録(D)の(2))
(iv)交渉を解除して，A に*Rep_a*を送信した．(付録(C)の(4)(5))
Aに注目すると，１エージェント内では，処理は逐次的になされるため，(i)と(ii)には，時間的な前後関係がある．ここでは，(i)(ii)の順に処理されたとき，(ii)(i)の順に処理されたときのそれぞれについて，矛盾を導く．
○Aにおいて，(i)(ii)の順に処理されたとき
Aは，(i)の後，Bに*Ins_a*を送信する．(付録(G)の(1)) *Ins_a*の送信は，(a)(ii)よりも前か，(b)(ii)よりも後である．
(a)の場合，Aのメッセージの送信順序は，*Ins_a*が先で，*Rep_b*が後になる．仮定(1)より，Bが，*Ins_a*を受信するのは，*Ins_b*を送信した後である．しかし，このためにはAから，*Rep_b*を*Ins_a*よりも先に受信しなければならない．これは，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ことに反する．よって矛盾．
(b)の場合，(ii)で*Ins_a*の交渉が解除されるため，*Ins_a*が送信されない．よって矛盾．
○Aにおいて，(ii)(i)の順に処理されたとき
この場合，(ii)で解除する交渉は，明らかに*Ins_a*の交渉ではない．よって，Aは(ii)の後，近傍に*Negotiate*メッセージを送信して*Ins_a*の交渉を行う．
この*Negotiate*メッセージをBが受信したとき，Aが*Rep_b*を先に送信していることから，Bは，(a)*Ins_b*の交渉中か，(b)交渉を終えて*Ins_b*を送信した後である．
(a)の場合，Bは，この*Negotiate*メッセージに対して，(iv)より，交渉を解除するため，*Ins_b*が送信されない．よって矛盾．
(b)の場合，Bのメッセージの送信順序は，*Ins_b*が先で，*Rep_a*が後になる．仮定(2)より，Aが，*Ins_b*を受信するのは，*Ins_a*を送信した後である．しかし，このためにはBから，*Rep_a*を*Ins_b*よりも先に受信しなければならない．これは，任意の２エージェント間では，送信メッセージの順序が保存されることに反する．よって矛盾．
よって仮定は否定される．以上で補題1が証明された． □

《補題2》
矛盾する制約をもつエージェントが存在する限り，有限時間内に具体化を変更するエージェントか，あるいは，LMOを実行するエージェントが必ず存在する．
〈証明〉
ある時刻Tにおける全エージェントの具体化の集合に対して，各エージェントの現在の制約矛盾数Cと制約矛盾数の最小値Mが一意に決まる．このとき，CとMの差が全エージェント内で最大，かつ，C≠0であるようなエージェントが必ず存在し，そのうち識別子が最小のエージェントをAとする．
時刻T以降，どのエージェントも具体化を変更しない場合，メッセージ遅延時間は有限なので，十分時間が経過した後，Aは近傍の全エージェントの具体化，C，およびMを知る．このときAは，自身のCとMの差が近傍∪{自身}で最大，かつ，C≠0であるため，自身に交渉権があることを知り，近傍に*Negotiate*メッセージを送信する．この*Negotiate*メッセージを受信した近傍内のエージェントのうち，制約が全て満たされてい

るエージェントは，AにAns=Trueの*Reply*メッセージを返す．(付録(C)の(1)) また，制約が全て満たされていないエージェントは，自身のC，Mの差をAのそれと比較するが，Aの定義と付録(C)の(2)(3)から，AにAns=Trueの*Reply*メッセージを返すことは明らかである．

よって，Aは近傍の全エージェントからAns=Trueの*Reply*メッセージを受信するため，具体化を変更するか，あるいは，(依頼エージェントとして)LMOを実行する．(付録(G)の(1)(2))

以上より，矛盾する制約をもつエージェントが存在する限り，どのエージェントも具体化を変更しない場合は，有限時間内にAが具体化を変更するか，LMOを実行する． □

《定理1》 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LMOを実行するエージェントも現われない状態)に達したとき，全エージェントは充足状態にある．

〈証明〉

補題2の対偶から，安定状態に達したとき，全エージェントが充足状態にあることは明らかである． □

《定理2》 完全性

エージェントの総数と各エージェントがもつ制約の総数が有限である限り，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分かって停止する．

〈証明〉

初期状態におけるエージェントの総数をn，各エージェントがもつ制約の総数をmとする．このとき，n−1回のLMOにより，初期状態において各エージェントがもっていたCSPが一箇所に集まる．一箇所に集まったCSPは，1エージェントの盲目的探索により解かれる．盲目的探索は完全であるから，初期状態における全エージェントのCSPに解があればそれを得るし，なければそれが存在しないことがわかる．ゆえに，最悪の場合，有限時間内にn−1回のLMOが起こることを示せばよい．これは，次の2つを示せば十分である．

(1)矛盾する制約をもつエージェントが存在する限り，有限時間内に具体化を変更するエージェントか，あるいは，LMOを実行するエージェントが必ず存在する．

(2)あるLMOとその次に実行されるLMOの間になされる具体化の変更回数は有限回である．

(1)は，補題2である．また，全エージェントの制約矛盾数の総和はm以下であること，全エージェントの制約矛盾数の総和が零になると具体化の変更が起こらないことと補題1から，mが有限ならば，連続して起こる具体化の変更回数は有限回であることより，(2)が導ける．

以上より，最悪の場合，有限時間内にn−1回のLMOが起こる．有限時間内にn−1回のLMOが起こらない場合は，全エージェントが充足状態にある． □

7. 実験

7.1. 実験方法

LMOを評価するために，DCSPの解を得るまでにかかる時間に関して，次の3つの社会を比較する．

社会(A)：LMOを実行する社会

社会(B)：分散型の社会

社会(C)：一極集中型の社会 (Forward checking)

社会(D)：一極集中型の社会 (Simple backtracking)

(A)は，本研究で提案した手法である．(B)は，局所最適解に陥ると初期値をランダムに変えて，山登り法をやり直すエージェントからなる社会である．ここでは，1つのエージェントでも局所最適解に陥ると，全エージェントが，山登り法をやり直すことにした．(C)と(D)は，初期値が制約矛盾を含む場合，全エージェントのCSPをリーダーと呼ばれる1つのエージェントに集めて解く社会である．リーダーの選択には，本研究の処理の一部を改良し，LMO時の更新エージェントの値域の更新を，全てのCSPが集まるまで遅延するという方法を用いた[2]．また，リーダーのCSPを解く能力として，(C)はForward checking，(D)はSimple backtrackingを想定した[5]．

対象とするDCSPは，3色問題と弱3色問題である．3色問題とは，互いに隣り合う領域は，異なる色で塗るという制約のもとで，地図上の各領域を赤，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，易しいDCSPとして用意したもので，互いに隣り合う領域どうしを赤で塗らないという制約のもとで，地図上の各領域を赤，青，緑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，それぞれの社会に与えた．但し，mは制約の数，nは変数の数とする．これらの問題例の設定は，[7]に基づく．

---

[2]リーダー選択問題に関しては，分散アルゴリズムの分野で，いくつかの方法が提案されている[9][10][11]．ここで，上の方法を用いた理由は，3つの社会が，それぞれ使用するメッセージ数のオーダーを等しくするためである．一般に，問題解決で使用するメッセージ数と問題解決にかかる時間にはトレードオフがある．

(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[3] (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，18の問題例それぞれについて，100個の異なる初期値を作成し，計1800問を各社会に与え，解が得られるまでのステップ数を求めた．各社会の評価として，同一問題例の異なる初期値100問の，解が得られるまでのステップ数の平均をとった．ここで1ステップ=制約チェック10回である．また，処理時間の上限を100nステップとし，これを越えたときには処理を打ち切った．通信遅れは，1ステップに固定した．

Fig. 2とFig. 3に結果を示す．横軸は，社会(B)が解を得るまでの平均ステップ数で，大まかに，潜在的な局所最適解の数(問題の難易度)を表わしている．縦軸は，対応する社会が，解を得るまでの平均ステップ数を，社会(B)のそれで割った値(平均ステップ数の比の値)である．グラフのプロットは，1つの問題例の異なる初期値100問の平均ステップ数を表わし，その100問を，社会(B)が平均何ステップで解いたか(横軸の値)，また，対応する社会が，その何倍のステップ数をかけて解いたか(縦軸の値)を表わしている．縦軸の値が1を下回ると，対応する社会は，その問題例を社会(B)よりも速く解いたことになる．

7.2. 考察

●社会(A)と社会(B)の比較

Fig. 2より，3色問題に関して，社会(A)は，社会(B)よりも良い結果を得ている．社会(A)のグラフが右下がりであることは，社会(B)でステップ数のかかる問題に対して，LMOの効果が大きいことを表わしている．社会(B)でステップ数のかかる問題とは，多くの局所最適解が存在する問題であり，このことから，LMOが，局所最適解が多い難しい問題に対して有効であることがわかる．また，Fig. 3によれば，弱3色問題に関しては，両社会の差異は見られない．局所最適解が，ほとんどない問題に対して，社会(A)は，社会(B)とほとんど同じ振舞いを見せる．

●社会(A)と社会(C)(D)の比較

3色問題に関して，社会(A)は，社会(C)よりも，やや効率が悪い．しかし，Fig. 2(a)のグラフ横軸の0付近では，社会(A)は社会(C)(D)よりも良くなっている．また，弱3色問題に関しては，社会(A)は，社会(C)(D)よりもかなり効率が良い．両者の違いは，組織形成の方法である．社会(A)は制約充足の結果，局所最適解に陥ったときに組織を形成するのに対し，社会(C)(D)は，制約充足の結果に関係なく，前もって組織を形成する．つまり，社会(C)(D)は，問題が，組織を形成することなく，局所的に解決できる可能性を無視しており，問題の難易度に適応できていない．これが，Fig. 2(a)の横軸の0付近での逆転および弱3色問題での社会(C)(D)の効率の悪さの原因と考えられる．

また，Fig. 2(b)では，社会(D)の効率が非常に悪くなっている．これは，Simple Backtrackingの問題点であるスラッシング[5](無駄なバックトラック)が原因だと考えられる．つまり，一極集中型の社会では，個々のエージェントのCSPを解く能力，特にリーダーの能力が問題になる．一方，社会(A)では，LMOにより集められるCSPが部分的で小さく，また，矛盾制約を共有する場合のみ集めるため，スラッシングの影響は小さい．つまり，全エージェントがSimple backtrackingしかできなくても，あまり効率に影響しない．

以上より，社会(A)は，問題の難易度に適応できる，個々のエージェントのCSPを解く能力に影響されないという点で，社会(C)(D)よりも優れているといえる．

●まとめ

分散型の社会は，問題が難しい場合に不利であり，一極集中型の社会は，問題が易しい場合に不利である．その点，LMOを実行する社会は，問題が難しいときは，集中型の問題解決，易しいときは，分散型の問題解決を行うため，両者の長所を保持できている．

8. 関連研究

まず，DCSPを解く他のアルゴリズムとの比較を行う．

横尾ら[6]は，非同期バックトラックを提案している．これは，あらかじめエージェントの機能を分散し，組織を静的に形成したものと解釈できる．機能分散により，各エージェントの局所計算時間は少なくなるが，メッセージ数が指数的に増加するという問題がある．それに対し，LMOによる解法では，メッセージ数が，問題のサイズの多項式で押さえられることが期待される．

また，DCSPを情報収集型分散アルゴリズム[9]で解くことが可能である．これは，実験での社会(C)(D)に相当する．§7.2でも示したように，LMOによる解法の方が，問題の難易度に適応できる，個々のエージェントの能力に影響されないという点で優れている．

次に，組織形成の研究と比較する．

石田ら[3]は，分散プロダクションシステムに組織の概念を導入し，実時間問題解決に従事するエージェ

---

[3]制約ネットワークの位相が異なる．

ントが，動的に組織を再編するというアプローチを取っている．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．組織再編の契機を問題解決の効率に求めた点で本研究との関連が深い．ただ，彼らの手法では，効率が悪くなると，並列性を向上させるために，タスクを分割するのに対し，本研究では，効率が悪くなる(解けなくなる)と，並列性を犠牲にして，タスクを集めるという違いがある．DCSPに関しては，分散という状況で問題解決を行う意義は，並列性よりも，集めることの不経済性にあると思われる．

9. まとめと検討

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形成について報告した．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を形成するというアプローチであり，問題の難易度に適応するという性質を持つ．以下に，検討事項を列挙する．

●LMOのコスト

LMOでは，更新エージェントが値域を更新するときに盲目的探索により全解探索するため，計算コストが大きくなる．そこで，LMOに適当なヒューリスティックスを導入して，これを抑える必要がある．１つの方法として，エージェントの能力を考慮した組織形成等が考えられる．

●通信遅れと組織形成

本研究では，問題の難易度と組織形成方法の関係を調べるため，通信遅れは，適当な値(1ステップ)に固定して実験を行った．しかし，通信遅れと組織形成方法との関係についても，今後検討する必要がある．

<謝辞>

日頃，お世話になっている豊田研FAIグループの皆さんに感謝します．また，分散アルゴリズムについてコメントを頂いたNTTソフトウエア研究所の寺内敦氏に感謝致します．

<参考文献>

[1] 大沢英一，沼岡千里，石田亨：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).

[2] Durfee, E.H., Lesser, V.R. : Using Partial Global Plans to Coordinate Distributed Problem Solvers, *IJCAI-87*, pp.875-883 (1987).

[3] Ishida, T., Yokoo, M., Gasser, L. : An Organizational Approach to Adaptive Production Systems, *AAAI-90*, pp.52-58 (1990).

[4] Steels, L. : Cooperation Between Distributed Agents Through Self-Organization, *Decentralized AI*, North-Holland Pub, Amsterdam (1990).

[5] Kumar, V. : Algorithms for Constraint Satisfaction Problems: A Survey, *AI magazine*, 13(1), pp.32-pp.44 (1992).

[6] Yokoo, M., Durfee, E.H., Ishida, T., Kuwabara, K. : Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving, *12th IEEE International Conference on Distributed Computing Systems*, pp.614-621 (1992).

[7] Minton, S., Johnston, M.D., Philips, A.B., Laird, P. : Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method, *AAAI-90*, pp.17-24 (1990).

[8] 平山勝敏，山田誠二，豊田順一：山登り法を用いた分散制約充足における組織化，人工知能学会全国大会(第7回)論文集，pp. 269-272 (1993).

[9] 萩原兼一：分散アルゴリズム，人工知能学会誌，Vol. 5，No. 4，pp.430-440 (1990).

[10] 萩原兼一：分散アルゴリズム入門，bit, Vol. 20, No. 4，pp. 378-400(1988).

[11] 大戸豊，茨木俊秀：グラフ構築およびリーダ選択問題における時間最小分散アルゴリズムについて，信学論(DI)，Vol. J72-D-I，No.10，pp.726-733 (1989).

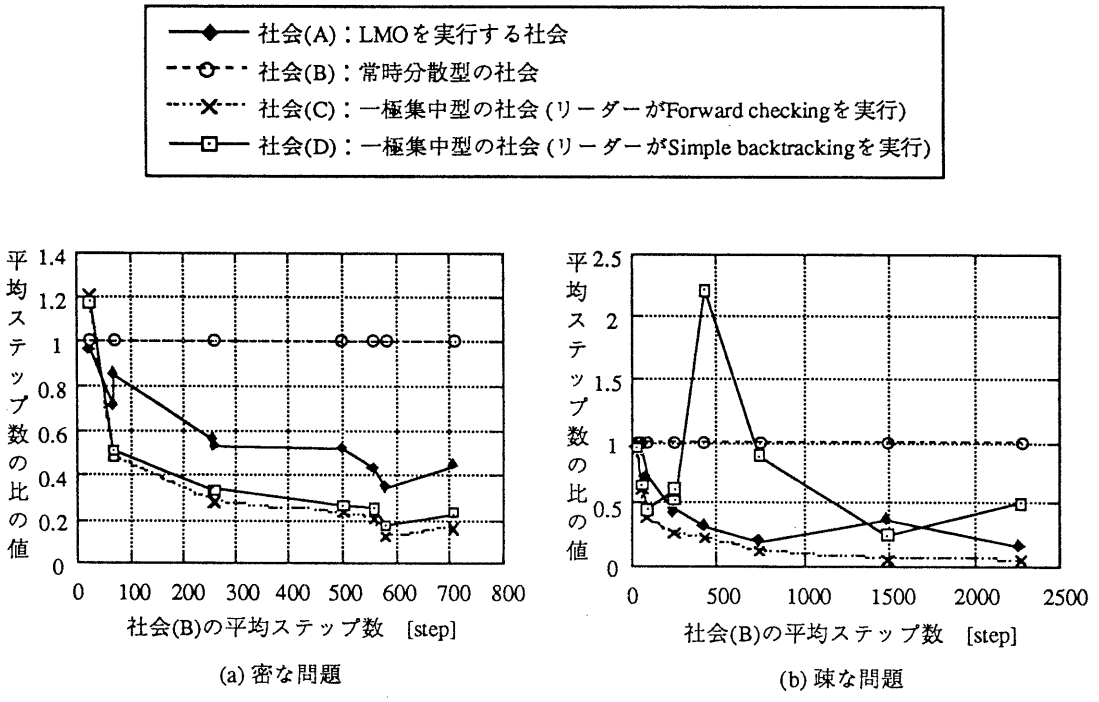

Fig. 2　3色問題の結果

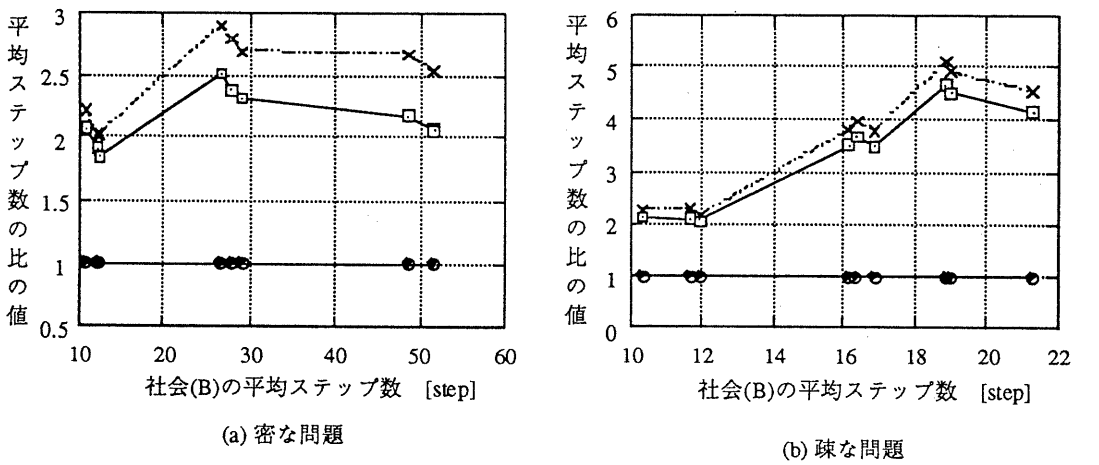

Fig. 3　弱3色問題の結果

<付録> アルゴリズム

(A)*Instantiation(Contents)*を受け取ったとき

```
begin
    Agent_viewとImprove_viewの更新;
    現在の交渉を解除;
    近傍へImproveメッセージを送信;
    if (交渉権がある) then
        近傍へNegotiateメッセージを送信;
    endif;
end
```

(B)*Improve(Contents)*を受け取ったとき

```
begin
    Improve_viewの更新;
    現在の交渉を解除;
    if (交渉権がある) then
        近傍へNegotiateメッセージを送信;
    endif;
end
```

(C)*Negotiate(Contents)*を受け取ったとき

```
begin
    C1←送信エージェントの現在の制約矛盾数;
    M1←送信エージェントの制約矛盾数の最小値;
    I1←送信エージェントの識別子;
    C2←自身の現在の制約矛盾数;
    M2←自身の制約矛盾数の最小値;
    I2←自身の識別子;
    if (自身のもつ制約が全て満たされている)  - (1)
        or (C1-M1)>(C2-M2)                  - (2)
        or {(C1-M1)=(C2-M2) and (I1 > I2)} then - (3)
        Ans←True;
    else
        Ans←False;
    endif;
    if (Ans=True) then
        現在の交渉を解除;                    - (4)
        送信エージェントへReplyメッセージを送
信;                                         - (5)
    endif;
end
```

(D)*Reply(Contents)*を受け取ったとき

```
begin
    Ans←交渉に対する返答;
    if (このReplyが現在の交渉に対するメッセージ
である) then                                - (1)
        if (Ans=True) then                  - (2)
            if (近傍の全エージェントから現在の交
渉のReplyを受信した) then
                call action;
            endif;
        else
            現在の交渉を解除;
        endif;
    endif;
end
```

(E)*Organize(Contents)*を受け取ったとき

```
begin
    CSPおよび近傍に関する情報を更新;
    if (値域=nil) then STOP;
    近傍へImproveメッセージを送信;
    if (交渉権がある) then
        近傍へNegotiateメッセージを送信;
    endif;
end
```

(F)*Request(Contents)*を受け取ったとき

```
begin
    送信エージェントへInstantiationメッセージを送
信;
end
```

(G)*procedure action*

```
begin
    C←自身の現在の制約矛盾数;
    M←自身の制約矛盾数の最小値;
    if (C > M) then
        具体化を制約矛盾数最小の値に変更;
        近傍へInstantiationメッセージを送信;   - (1)
    endif;
    if (C = M) then
        ターゲットの選択;
        メッセージのフォワード先を変更;
        ターゲットへOrganizeメッセージを送信;
                                            - (2)
    endif;
end
```