記　号　処　理 78－5
（１９９５．　３．　２４）

大規模分散制約充足問題における協調と自己組織化

佐々木 崇郎　中西 正和
慶應義塾大学大学院理工学研究科数理科学専攻

分散制約充足問題において情報は複数のエージェントに分散しており、各エージェントは大域的な情報を持っていない。分散制約充足問題の解法として、情報を一つに集約することにより制約充足問題として解くというアプローチも取られている。しかし、現実の問題ではネットワークの経路制御など規模が大きくなる、あるいは保安上の問題で情報の集約が難しい問題も少なくない。本稿では、大域的な情報を持たない複数のエージェントが個々に独立したまま、協調によって問題解決に当たるアルゴリズムを提案する。また、シミュレータを作成し、3色問題を解かせることによりその効率を測定し、有効性を示す。

Cooperating and Organizing Algorithm of Large Scale Distributed Constraint Satisfaction Problem

SASAKI Takao and NAKANISHI Masakazu
Department of Mathematics
Graduate School of Science and Technology
KEIO University

On solving DCSP, where no agent has global information, some approaches are proposed to gather all information to one agent. In fact, however, there are not a few problems that it is difficult to concentrate all information on one agent, such as managing the network routing, because of the scale of the problem or on account of some security problems. In this paper, we propose an algorithm that every agent, which has no global information, cooperates with each other to solve DCSP, and also simulate the algorithm on problem of 3-colorability to show the effectiveness of our method.

1 はじめに

制約充足問題はラベリングやN-クィーン問題などの人工知能分野のおける様々な問題、あるいは負荷分散、ネットワークのルーティングなどの計算資源の分割問題など、計算機における多くの問題を定式化できるモデルである。

分散制約充足問題において情報は複数のエージェントに分散しており、各エージェントは大域的な情報を持っていない。分散制約充足問題の解法として、大きく分けて二つある。一つは、個々のエージェントが自立協調して全体として問題を解くアプローチ方法であり、もう一方は、一つのエージェントに情報を集約することにより制約充足問題として解くというアプローチである。しかし、現実の問題ではネットワークの経路制御などのように規模が大きくなる、あるいは保安上の理由などで情報の集約が難しいあるいは制限されている問題も少なくない。

本稿では、大域的な情報を持たない複数のエージェントが個々に独立したまま、協調によって問題解決に当たるアルゴリズムを提案する。また、シミュレータを作成し、典型的制約充足問題である3色問題を解くことによりその効率を測定し、本稿で提案するアルゴリズムの有効性を示す。

2 問題の定式化

まず、制約充足問題の定式化および分散制約問題の定式化を行なう [1]。

2.1 制約充足問題

制約充足問題は次のように定義される。

n個の変数 $x_1, x_2, \cdots, x_n$ と、変数のそれぞれが値を取る有限で離散的な値域 $D_1, D_2, \cdots, D_n$、および制約の集合から成る。制約は述語によって内包的に定義される。本稿では、モデルの簡略化のために制約は特定の2変数間に定義されるものとする。述語 $P_{ij}(x_i, x_j)$ はそれぞれの変数の値域の直積 $D_i \times D_j$ の部分集合に対して定義され、x_i, x_j に割り当てられた値 $d_i, d_j (d_i \in D_i, d_j \in D_j)$ が制約条件を満たすとき真となる。

制約充足問題を解くとは、すべての制約を満たす変数の割り当てを求めることである。

2.2 制約ネットワーク

制約充足問題は、変数を頂点、制約を辺に対応させた無向グラフ (図1-a) で表現することができる。

また、制約充足問題は図1-bのような制約を頂点、制約間で共通する変数を辺に対応させた無向グラフで表現することもできる。制約充足問題を解くアルゴリズムの一つとして頂点併合法があるが、これは共通する変数を持つ、すなわち辺で結ばれた頂点を順次繰り返し併合していくことによって最終的に一つの頂点とすることで制約充足問題を解く方法である。

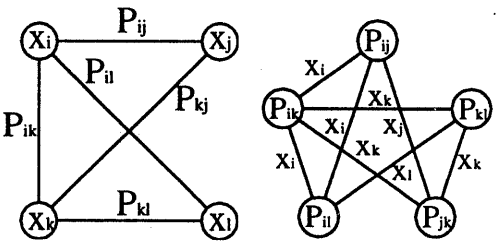

図 1-a: 変数が頂点　　図 1-b: 制約が頂点

図 1: 制約ネットワーク

2.3 分散制約充足問題

分散制約充足問題は分散環境下での制約充足問題であり、問題中の変数、制約は複数のエージェントに分散している。

さらに、本稿では以下のことを仮定する。

- 変数は各エージェントに分散して存在する。
- 各エージェントは自らに属する変数に関する制約はすべて知っている。

2.4 エージェントネットワークモデル

図1-aのように考えると異なるエージェントに属する変数間の制約はエージェント間の通信路と考えることができる。

制約充足問題を分散環境下で並列に解く研究としては図1-bのような制約が頂点であるネットワークモデルにおいて頂点併合法を用いた研究もあるが、本稿では制約充足問題を並列に解くことではなく、分散環境下で大域的な情報の取得が制限された場合の制約充足問題に主眼を置いているため、図1-aのようなネットワークモデルを採用した。

現実の世界としてネットワークの経路制御などを考えた場合、それぞれのエージェントに変数が分散しており、その変数間に制約が成り立っているとするモデルが妥当であると考えられる。

前章での定式化以外に本研究で用いるモデルではエージェント間通信に以下のような仮定をする。

- 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ることができる。
- メッセージの遅延は有限時間であるが、遅延時間の上限はわかっていない。
- 任意の二つのエージェントの組合せに関して、送信されたメッセージの順序は保存される。

また、各エージェントは自らに属する変数との間で制約が存在する変数を持つエージェントのアドレスだけを知っているものとする。

3 アルゴリズム

各エージェントは以下のようなアルゴリズムで独立に動作させる。各エージェントは固有の番号を持ち全順序関係があるものとする。各エージェントは状態として所属する変数の現在の値、制約矛盾数を隣接エージェント間で以下のアルゴリズムの中で交換する。また、隣接エージェントから送られてきた状態は状態表として保持する。

```
procedure メインループ
  while 真 do                    ▷ 無限ループ
    if 受信メッセージがある then
      受信メッセージの処理
    else if
    (制約矛盾数が 0 ではない and
    交渉中ではない and
    制約矛盾数が隣接エージェント中で最大)
    then
      交渉メッセージを隣接エージェントに
      送信
    end if
  end do
```

各エージェントは受信メッセージがある限り、その処理を行なう。隣接エージェント中最大の制約矛盾数を持つエージェントは、変数の値を変更するべく隣接エージェントとの交渉に入る。変数の値は交渉を行ない、すべての隣接エージェントから承諾が得られて初めて更新できる。

```
procedure 状態メッセージを受信
  if 送信元の状態に変化 then
    状態表を更新
    現在、自分が行なっている交渉を中断
    制約矛盾数 ← 制約矛盾数を再計算
    if 制約矛盾数に変化 then
      状態を隣接エージェントに送信
    end if
  end if
```

状態メッセージを受信した場合、送信元の状態が自分の持っている隣接エージェントの状態表と異なっていれば、表を更新し、新たな制約矛盾数を算出する。制約矛盾数に変化があれば、現在の状態を隣接エージェントの送信する。

```
procedure 交渉メッセージを受信
  if (送信元の状態を知らない or
  自分の制約矛盾数が 0 or
  送信元の制約矛盾数のほうが大きい or
```

表 1: 制約の数

変数の数	10	20	30	40	50	60	70	80	100
密である場合	23	95	218	390	613	885	1208	1580	2475
疎である場合	20	40	60	80	100	120	140	160	200

```
(制約矛盾数が等しい and
            送信元の ID の方が小さい))
then
    現在、自分が行なっている交渉を中断
    送信元に承諾のメッセージを返信
else
    送信元に不承諾のメッセージを返信
end if
```

交渉メッセージはシリアル番号を持つことで、管理されている。したがって、すでに中断してしまった交渉に対する返信は無視することができる。

```
procedure 不承諾メッセージを受信
  if 現在交渉中 and 交渉シリアル番号が最新
  then
    現在、自分が行なっている交渉を中断
  end if

procedure 承諾メッセージを受信
  if 現在交渉中 and 交渉シリアル番号が最新
  then
    if すべて隣接エージェントから承諾
      更新
    end if
  end if

procedure 更新
  if 値域からどの値を割り当てても現在の制約矛
  盾数を減少できない
    現在の値では矛盾する制約を共有する任意
    のエージェントと組織化
  else
    値域の中で制約矛盾数を最小にするものを
    割り当てる
    隣接エージェントに新状態を送信
  end if
```

組織化する場合は、隣接エージェントでの現在の自分の状態を不定に変えるために更新メッセージを送信する。

組織化後は周囲からのメッセージを組織化したエージェントに中継する。ただし、状態表は保持しておき、必要のない (変化のない状態メッセージ等) メッセージは抑制し、交渉メッセージには自ら答える。

```
procedure 更新メッセージを受信
    状態表の送付元エージェントの状態を不定にする

procedure 組織化メッセージを受信
    送付されてきた変数、その値域、制約を併合する
```

4 実験

効率解析のためにシミュレータを作成し、実験を行なった。各エージェントは自らが持つ制約を参照した回数を単位として全体で同期した時計を持っている。その時計を基準にメッセージのやりとりが行なわれる。また、メッセージ送付の際、到着予定時刻を 10 単位時間進めておくことにより通信の際の遅延を実現している。

対象とした制約充足問題は 3 色問題である。これは、互いに隣合う領域は異なる色で塗り分けるという制約のもとで、全体を 3 色に塗り分ける問題である。

制約充足問題の例題には、この 3 色問題の他に N-クィーン問題が良く用いられるが、N-クィーン問題の場合、制約が他のすべての変数との間に存在しているので、隣接するエージェントの情報しか知り得ない状況下を想定している本稿にはそぐわないため、3 色問題のみ実験を行なった。

実験は Adorf[2] の分類に基づき、制約が密である問題と疎である問題の二通りについて行

なった。密である問題と疎である問題の定義については以下の通りである。

変数の数を m とするとき、制約の数 n を以下の通りとする (表 1)。

- 密である問題　　$m = n(n-1)/4$
- 疎である問題　　$m = 2n$

変数の数が 10,20,30,40,50,60,70,80,100 個の場合について、それぞれに解が存在することを確認した例題を 5 題づつ作り、それぞれに 200 組の初期値を乱数によって作成し、解が得られるまでの時間を求めた。

各エージェントはただ一つの変数を持つものとした。すなわち、変数の数だけエージェントを用意した。

比較対象としていずれかのエージェント間でデッドロックに陥った場合に隣接エージェントとの組織化によって解決するのではなく、すべての初期値を乱数で与え直し、計算をやり直すアルゴリズム (組織化なし) による実験を行なった。

また、制約が疎である問題に関しては、デッドロックの時に矛盾している制約を共有している任意のエージェントとの間で組織化を行なう (組織化 A) のではなく、矛盾している制約を共有しているエージェントの中で所属する変数がもっとも少ないエージェントと組織化を行なうというヒューリスティックを使って組織化を行なうようにしたエージェントでも同様に実験を行なった (組織化 B)。

5 実験結果

それぞれの例題に対して、200 個の処理時間の平均を取ると、どの例題の場合も同じ変数の場合には、ほぼ同じ値を示した。そこで、さらに 5 題の平均を取り、変数の数に対する経過時間をグラフにしたものが図 2および図 3である。

6 考察

図 4は制約が密である場合の例題で組織化なしアルゴリズムの実験で起きた変数の数に対する再初期化回数の平均を取ったグラフである。

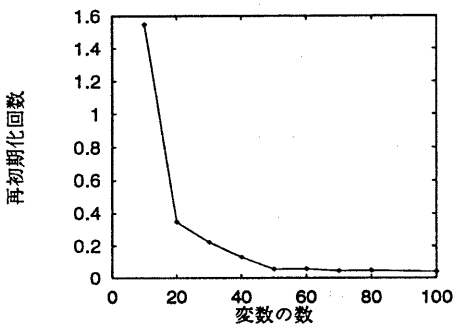

図 4: 再初期化 (密な制約)

このグラフからもわかるように制約が密である場合、デッドロックはほとんど起きない (故に、組織化が起きない)。したがって、デッドロックに対して再初期化した場合と組織化により矛盾の解決をはかった場合では有意な差は見られない。

一方、制約が疎である場合には再初期化の回数は図 5のようになる。

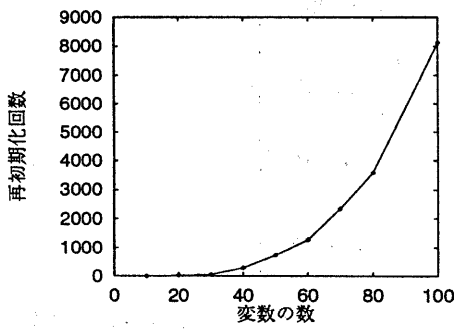

図 5: 再初期化 (疎な制約)

制約が疎である場合はデッドロックを起こす確率が変数の数とともに非常に大きくなってい

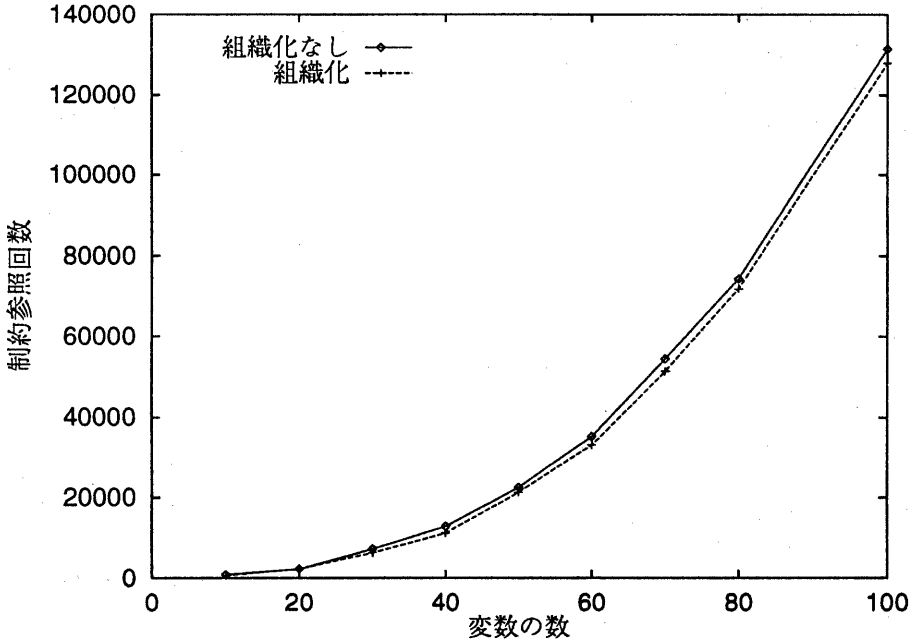

図 2: 三色問題 (密な制約)

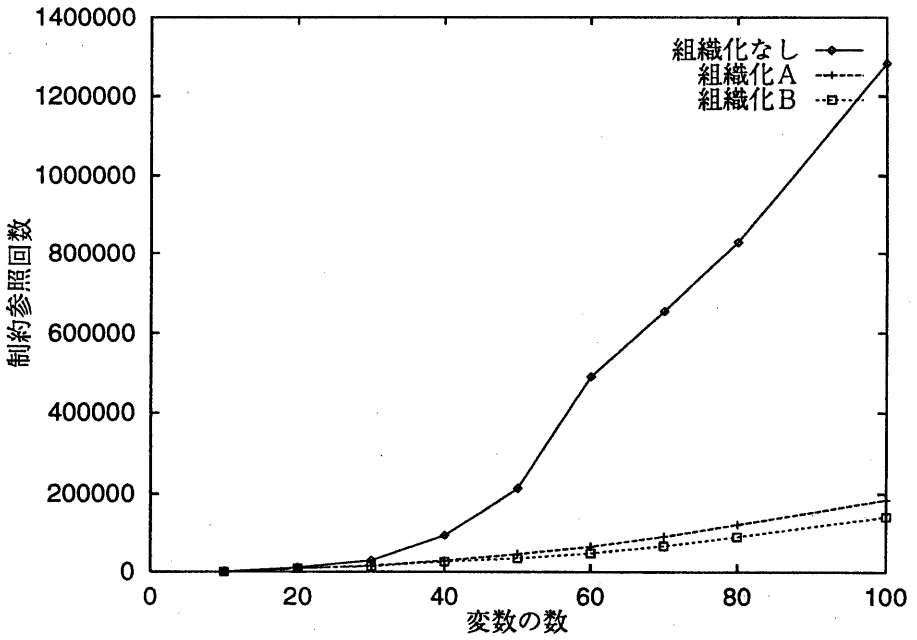

図 3: 三色問題 (疎な制約)

くために組織化によって同じデッドロックを二度と起こさない本アルゴリズムが有効に働いていると思われる。

組織化の際にヒューリスティックを導入した組織化 B の結果を見ると組織化に関してはまだ工夫の余地があると見られる。

7 逐次型制約充足問題の解法との比較

既存の逐次型制約充足問題の解法である弱コミットメント探索 [1] および制約違反最小化バックトラッキング [3] との比較を行なった [1]。

3 色問題は NP-完全問題であり、CSP は並列化による高速化は本質的にできないとの報告 [4] もあることから、一概に比較はできないが、組織化を行なわない場合でも制約違反最小化バックトラッキングと同等、組織化を行なった場合は、弱コミットメント探索には及ばないもののかなり、有効なアルゴリズムであるといえる。

8 今後の課題

- 組織化に際し、ヒューリスティックを用いることは有望性があるため既存の逐次型アルゴリズムで用いられているものなどを検討してみる。
- さらに大きな規模になった場合にも継続して有効であるか検討する。
- アルゴリズムとしては厳密な検証を行なっていないので完全性の証明など検討する必要がある。
- 現実では、制約が動的に変化する場合も多いのでそのような場合にも絶え得るアルゴリズムについても検討を行なう。

参考文献

[1] 横尾 真. 弱コミットメント戦略を用いた制約充足問題の解法. 情報処理学会論文誌, 35(8):1540–1548, Aug. 1994.

[2] Hans-Martin Adorf. A DISCRETE STOCHASTIC NEURAL NETWORK ALGORITHM FOR CONSTRAINT SATISFACTION PROBREMS. In *IEEE International Joint Conference on Neural Networks III*, pages 917–924, 1990.

[3] Steven Minton et al. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[4] S Kasif. On the parallel complexity of discrete relaxation i constraint satisfaction networks. *Artificial Intelligence*, 45:275–286, 1990.

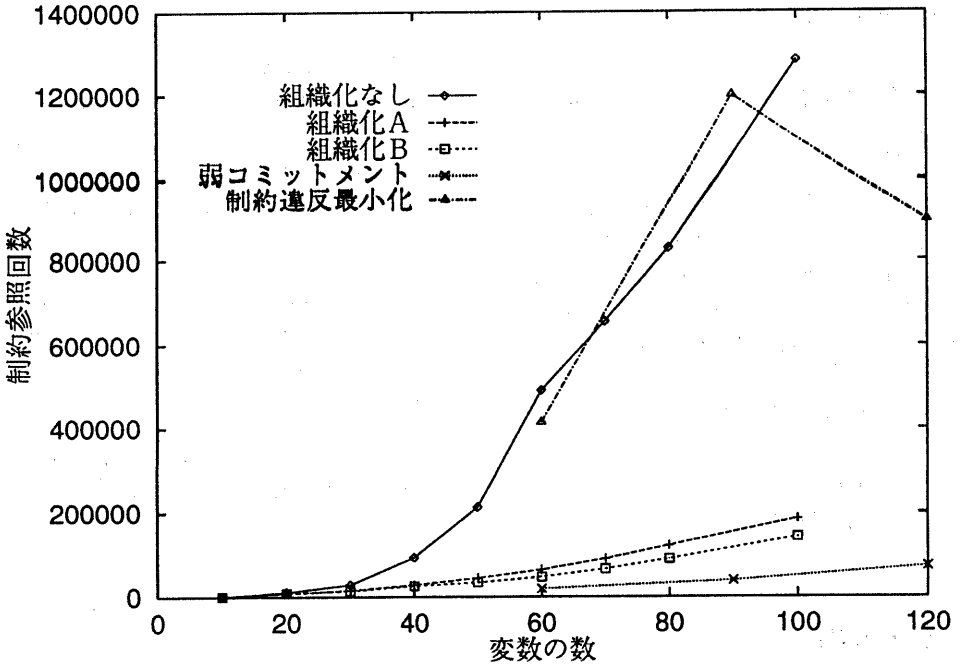
1400000
1200000
1000000
800000
600000
400000
200000
0
制約参照回数
0
20
40
60
80
100
120
変数の数
組織化なし
組織化A
組織化B
弱コミットメント
制約違反最小化

図 6: 逐次型との比較