人 工 知 能 77−12
（1991. 7. 18）

分散制約充足アルゴリズムの一般化と評価

西部 喜康　　桑原 和宏　　石田 亨
NTT 情報通信処理研究所

分散制約充足アルゴリズムは分散人工知能の基礎的なアルゴリズムの一つであり，最近注目され始めている．我々は分散人工知能の研究分野で議論されてきた多くの問題がこの分散制約充足問題として定式化できることを示してきた．

本論文ではこれまで報告されてきた種々の分散制約充足アルゴリズムを基に一般化を行い，分散制約充足アルゴリズムの枠組を与える．さらに各々の分散制約充足アルゴリズムを，その枠組の一部を特殊化したものとして説明する．また各々の分散制約充足アルゴリズムの有効性を調べるために評価を行い，その特徴を明らかにする．

Generalization and Evaluation of Distributed Constraint Satisfaction Algorithms

Yoshiyasu Nishibe　　Kazuhiro Kuwabara　　Toru Ishida
nishibe@nttkb.ntt.jp　　kuwabara@nttkb.ntt.jp　　ishida@nttkb.ntt.jp

NTT Communications and Information Processing Laboratories
1-2356 Take, Yokosuka-Shi, Kanagawa 238-03, Japan

A distributed constraint satisfaction problem is one of basic problems in distributed artificial intelligence (DAI) and has attracted a lot of attention recently. It was shown that most of discussed problems in DAI can be formalized as a distributed constraint satisfaction problem.

In this paper, we present the framework of distributed constraint satisfaction algorithms by generalizing reported algorithms. These algorithms can be considered specializing some parts of the proposed framework.

Three distributed constraint satisfaction algorithms in this framework are evaluated and characteristics of these algorithms are discussed.

1 はじめに

分散制約充足問題は複数の自律的に動作するエージェントにより行われる分散協調問題解決 [1] の一つであり, 近年注目を集めている.

従来より議論されてきた制約充足問題とは, 有限領域から値をとる複数の変数に対して制約を満たす値の割当を求める問題である [2]. これに対して分散制約充足問題とは, 制約充足問題を分散計算環境でとらえ直した問題であり, 制約充足問題の変数, 領域, 制約などの情報が複数のエージェントに分散された問題である. 我々は 分散 AI の問題の多くがこの分散制約充足問題として定式化できることを示してきた [3].

分散制約充足問題におけるこれまでの研究は, 通信ネットワークにおける回線設定問題などの資源割当問題を題材として進められてきた [4][5]. しかし現在の所, 少数の分散制約充足アルゴリズムが提案されているに過ぎず, 体系的にまとめられるに至っていない.

本論文では, 分散充足アルゴリズムの一般的な枠組を与えると共に, これまで提案されているアルゴリズムがその枠組を特殊化して得られるものであることを示す. また評価を通じて各アルゴリズムの特徴を明らかにする.

2 問題の定義

2.1 制約充足問題

制約充足問題は m 個の変数 x_1, x_2, ..., x_m と, その変数のとり得る値の集合 D_1, D_2, ..., D_m および制約の集合によって定義される. 制約は各変数のとり得る値の組で表す. 以下では変数の組を引数とする述語 $P_j(x_1, x_2, ...)$ で制約を表す.

制約充足問題を解くことは上記の変数, 領域, 制約が与えられた場合, 全ての制約を満たす値の組合せを見つけ出すことである. すなわち, 述語論理で記述すると次の整式が真となる値を各変数に割当ることである [2].

$$(\exists x_1)(\exists x_2)\ldots(\exists x_m)$$
$$(x_1 \in D_1)\quad (x_2 \in D_2)$$
$$\ldots (x_m \in D_m) \bigwedge_j P_j(x_1, x_2, ...)$$

2.2 分散制約充足問題

分散制約充足問題は分散計算環境での制約充足問題であり, 問題中の変数, 領域, 制約などが自律的に動作する複数のエージェントに分散され, 管理されている.

この条件の下で分散制約充足問題が解けたとは, 全てのエージェントにおいて変数 x_i の値が $d_i(d_i \in D_i)$ に決定されていると共に, 全ての制約が
$x_1 = d_1$, $x_2 = d_2$, $x_3 = d_3$, ..., $x_n = d_n$ のもとで真となることをいう.

図 1に本論文で扱う分散制約充足問題のモデルを示す.

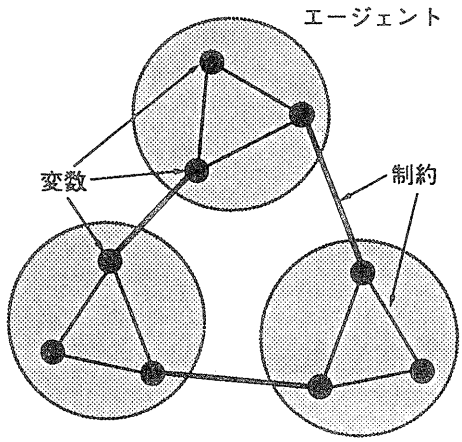

図 1: 分散制約充足問題

このモデルでは変数, 領域, 制約が複数のエージェントに対して以下のように分散されているものとする.

- 一変数は一つのエージェントに属する.
- 各エージェントは自身に属する変数の値の領域 D を知っている.

- 各エージェントは自身に属する変数を引数として含む制約を知っている.

また, エージェント間通信については以下のように仮定する.

- エージェント間の通信はメッセージの送信によって行われる (共有メモリは存在しない).
- エージェントの id が通信のために用いられる.id を知っているエージェントに対してはメッセージを送信できる.
- メッセージは失われることはなく, またメッセージの送信順序は着信時にも保存される.

2.3 分散制約充足問題の特徴

例えばエージェント i が, 変数 x_i に領域 D_i より値を割当て, 他のエージェント j が変数 x_j に領域 D_j より値を割当てるとする. また, 変数 x_i と x_j の間に制約 $P(x_i, x_j)$ が存在しているとする.

このような場合, 制約充足問題では, まず変数 x_i に値が割当られ, その値と制約違反を起こさない値を変数 x_j に割当るという逐次的な手法が用いられる.

しかしながら分散制約充足では変数, 領域, 制約が複数のエージェントによって分散管理されている. このため, 複数のエージェントが非同期並行的に変数に値を割当てる. また相互に制約のあるエージェントに, 割当てた値を通信することが必要となる.

上記の例では制約 $P(x_i, x_j)$ を管理しているエージェントが エージェント i から変数 x_i に対する値を, エージェント j から変数 x_j に対する値を通信によって得, 制約チェックを行う. また制約違反が存在した場合はエージェント i,j に対して制約違反の存在を報告して再割当を要請する.

この各エージェント間の通信が分散制約充足の特徴であり, アルゴリズムを考えるうえで問題となる部分である.

3 分散制約充足アルゴリズムの枠組

3.1 アルゴリズムの方針

従来, ある任意のエージェント (以後, ルートエージェントと呼ぶ) に全ての情報を集中させて問題を解くアルゴリズムが提案されてきた [6]. このアルゴリズムは, ルートエージェントに情報を集中し, ルートエージェントが問題を解き, 解を全てのエージェントに対して放送するというものである. この手法は, ルートエージェントでバックトラック法などこれまでに提案されてきた一エージェントでの制約充足アルゴリズムが適用可能であるという利点を持つ. しかしながらこのアルゴリズムには以下のような問題点が存在する.

- ルートエージェントに処理が集中し, 並列性が活かされない.
- エージェントが全ての情報をルートエージェントに報告しなければならない. 従って, ルートエージェントへの通信がボトルネックになる.

これに対し本論文で扱う分散制約充足アルゴリズムでは, エージェント間でできる限り通信をせずに解を導き出す. 基本的な処理は, 複数のエージェントが非同期並行的に自エージェント内の変数に対して値の割当を行い (局所解の選択), 値を割当てた変数と制約がある変数を持つエージェントに, その値 (局所解) を送信するというものである.

3.2 分散制約充足アルゴリズムの一般的な枠組

これまで提案された分散制約充足アルゴリズムでの各エージェント内の処理 [3][7] を一般化すると図2のように表すことができる.

この処理は通信駆動型であり, 通信メッセージは次の三種類である.

1) 局所解メッセージ
   複数のエージェントが非同期に自エージェント内の変数に制約を満足するように値を割当てる (この割当を以後局所

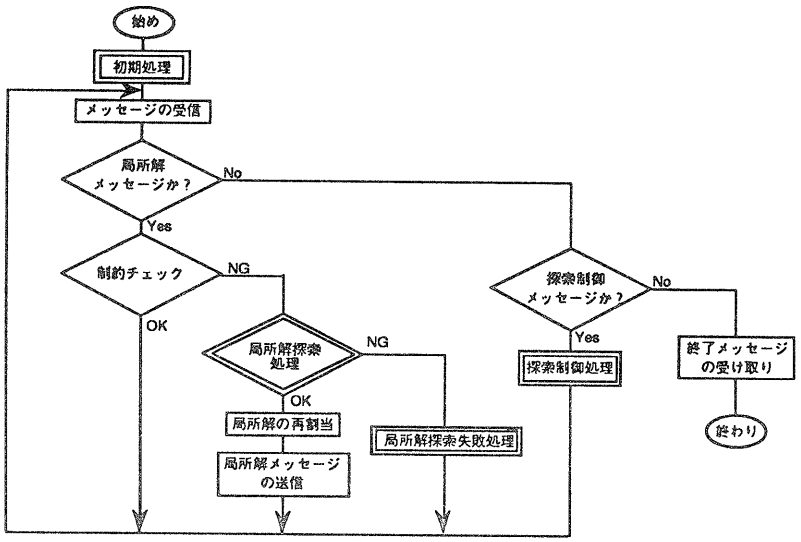

図 2: 分散制約充足アルゴリズムの枠組

解と呼ぶ). その後, この局所解の割当を行った変数と制約がある変数を持つエージェントに局所解を送信する. このメッセージが局所解メッセージである.

2) 探索制御メッセージ
他のエージェントの探索制御に有効な情報を与えるためのメッセージである.

3) 終了メッセージ
使用しているアルゴリズムにおいて解を導き出せないことを伝えるメッセージである.

次に各エージェント内で行われる処理について説明する.

エージェントの処理は二つの段階からなる.

1) 初期処理
基本的には初期局所解の探索および局所解メッセージの送信が行われる. コンシステンシアルゴリズムの採用 [3] などで, この処理中に特殊なコントロールメッセージの送信が行われる場合もある.

2) メッセージ駆動処理ループ
この処理では送られてきたメッセージによって三種の処理から一つを選択し, 実行する.

2-1) 制約チェック処理
局所解メッセージを受けとった場合に行われる処理である. 送られてきた他エージェントの局所解と自身が選択している局所解の間で制約チェックを行う. 制約違反が存在しない場合, エージェントは何も行わない. しかし制約違反が存在する場合, 制約を満たすよう局所解の更新を行う. ここでは各アルゴリズムで決められた局所解探索処理により局所解の更新を行い, 更新結果を局所解メッセージとして他エージェントに送信する. 局所解の更新が不可能な場合は局所解探索失敗処理を行う. ここでは, 各アルゴリズムで定義した探索制御メッセージ, あるいは終了メッセージが送信される.

2-2) 探索制御処理
探索制御メッセージを受けとった場合

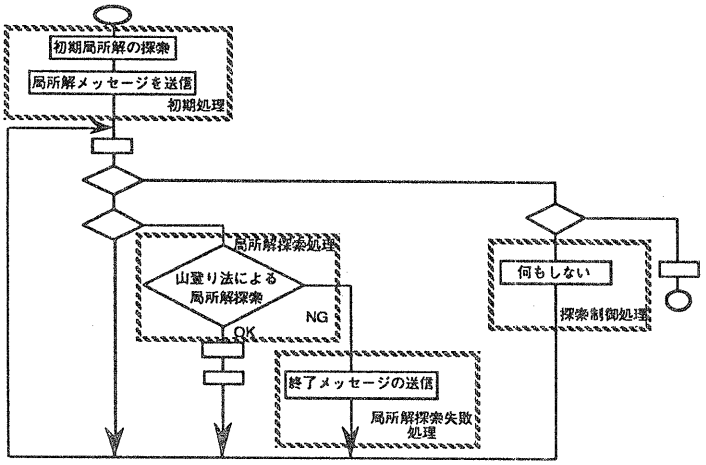

図 3: 山登り法

に行われる処理である. この処理は各アルゴリズムで定義される. 局所解候補に対して付加情報を与えることで, 続いて行われる局所解探索を助ける働きをする.

2-3) 終了処理
終了メッセージを受けとった場合に行われる. エージェント内の処理を終了する.

この枠組において分散制約充足問題が解けている状態は, 全エージェントがメッセージ受信待ちになっている状態であり, その時の全エージェントの局所解を集めたものが解となる.

上記処理中の下線を引いてある部分が (図2では2重枠でかこってある部分) が, 各アルゴリズムによって変化する部分である.

4 分散制約充足の各種のアルゴリズム

本章ではこれまで報告された各種の分散制約充足アルゴリズムが前章で定義した分散制約アルゴリズムの枠組を特殊化したものであることを示す.

本論文では, 提案した枠組における最も簡単なアルゴリズムである山登り法, 全探索を保障した分散バックトラック法をとりあげる. また, ヒューリスティックの効果を考えるために山登り法に対しヒューリスティックの適用を行ったアルゴリズムもとりあげる.

4.1 山登り法

まず分散制約充足アルゴリズムの枠組を用いた最も簡単なアルゴリズムを考える. 図3にこの方式を示す (以下このアルゴリズムを山登り法と呼ぶ). この方式では各エージェントは局所解の選択以外の処理を行わない. そのため, 探索制御メッセージは存在せず, 探索制御処理が省かれる. 以下アルゴリズムでの可変部分の処理をより詳細に説明する.

- 初期処理
  任意の局所解候補を初期局所解として選択し, 他のエージェントに局所解メッセージを送信する.
- 局所解探索処理
  ここで局所解の探索はもっとも簡単な山

登り法を使用する．局所解は局所解候補として格納されているものから任意の一つが選択される．一度選択された局所解候補は再び選択されることはない．

- 局所解探索失敗処理
  局所解探索において選択する局所解候補がエージェント内に存在しなくなった場合には，終了メッセージを送信する．
- 探索制御処理
  探索制御メッセージが送信されないため行われない．

4.2 分散バックトラック [3]

この手法の基本的な戦略は，制約違反が生じた場合にその制約違反を起こした局所解の組合せを記憶しておき，これを利用することで探索を効率化し，完全性を保障することである．図4に処理の概要を示す．

この分散バックトラック法ではエージェントに対して全順序が設けられる．これはエージェント間で制約違反が起こった場合，両方のエージェントで局所解の更新を行わず低順位のエージェントのみが局所解の更新処理を行うようにするためである．また，このアルゴリズムでは制約違反を起こした局所解の組合せを NoGood として記憶し，エージェント間で通信することによって制約違反を起こす原因となった局所解を再び選択しないように制御する．

このアルゴリズムでの可変部分の処理は次の通りである．

- 初期処理
  任意の局所解候補を初期局所解として選択し，他のエージェントに局所解メッセージを送信する．
- 局所解探索処理
  局所解の探索処理では，他のエージェントから送られてきた局所解メッセージおよび，エージェント内に蓄積した NoGood と矛盾しない局所解候補が任意に選択される．この処理は制約違反を起こしている局所解を持つエージェントの中で最低順位のエージェントでのみ行われる．
- 局所解探索失敗処理
  選ばれるべき局所解候補がなくなった場合は，一つ上位のエージェントに対して探索制御メッセージを送信し，局所解の変更を要請する．この場合の探索制御メッセージは制約違反を起こした局所解の組合せ (NoGood) である．この処理を行おうとしたエージェントが最高順位のエージェントの場合，解は存在しないことになり終了メッセージを全エージェントに対して送信する．
- 探索制御処理
  まず送られてきた NoGood を蓄積し，局所解探索処理にそれを反映する．続いて送られてきた NoGood と自分の選択している局所解とを比較し，選択している局所解が制約違反を起こしている場合には局所解の更新処理を行う．

4.3 ヒューリスティックの適用

分散制約充足アルゴリズムに対するヒューリスティックの適用は [7] で提案されている．この手法の特徴は各エージェントで選択できる局所解候補に対して評価値を付加することで局所解候補の並び替えを行い，制約違反の出現確率を下げようとするものである．

図5にヒューリスティックを適用した分散制約充足アルゴリズムを示す (以下このアルゴリズムをヒューリスティック法と呼ぶ)．

この手法では，初期処理として自身が選択できる全ての局所解候補を送りあい，各エージェント内で局所解候補が制約違反を起こす確率を計算する．その確率を基に局所解候補を制約違反を起こす確率の少ない順に並び替える．この処理を先の山登り法に加えることで，制約を満足する局所解をより早く見つけ出すよう制御しようとするものである．

以下，アルゴリズムの可変部分の詳細を説

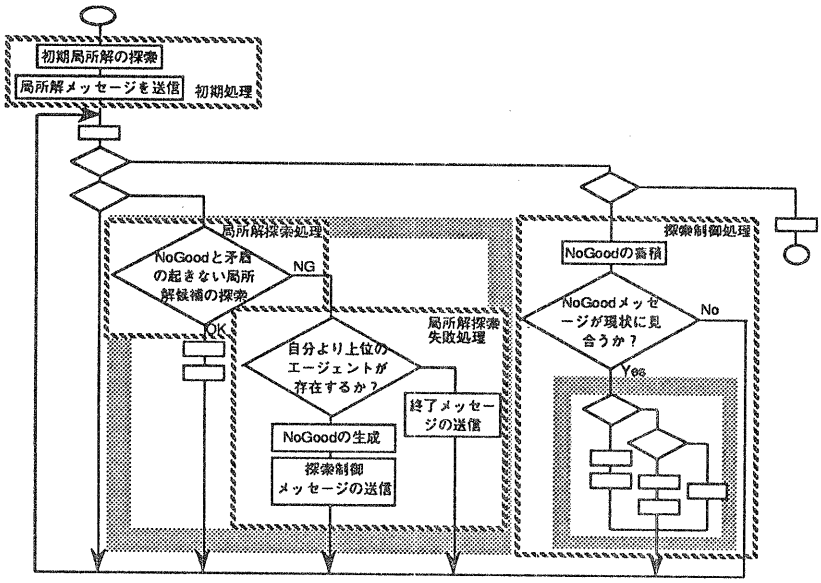

図 4: 分散バックトラック法

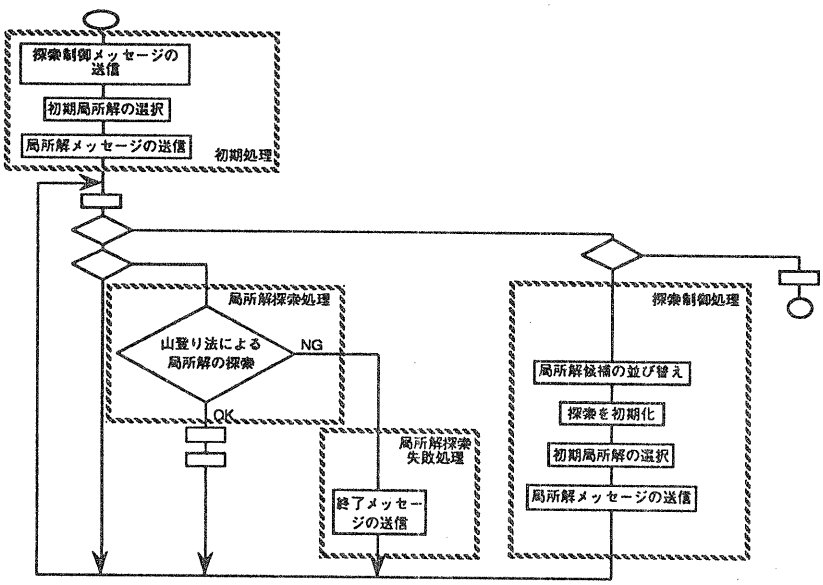

図 5: ヒューリスティック法

明する.

- 初期処理
  まず,各エージェントで局所解候補の並び替えを行うための探索制御メッセージを送信する.その後,初期局所解を選択し,局所解メッセージを送信する. ここで送信される探索制御メッセージは自エージェントが持つ全ての局所解候補である.自エージェントに属する変数と制約を持つ変数が属する全てのエージェントに送信する.
- 局所解探索処理
  局所解の選択は局所解候補の順序の通り行われる.探索制御処理によって局所解候補は制約違反を起こす確率の少ない順に並べられているため,制約違反を起こす確率の少ない候補より選択されることになる.
- 局所解探索失敗処理
  選択するべき局所解候補がなくなった場合には,終了メッセージを送信して処理を終る.
- 探索制御処理
  送られてきた探索制御メッセージは他のエージェントの局所解候補である. この局所解候補と自エージェント内の局所解候補との制約チェックを行い,自エージェント内の局所解候補を制約違反を起こす確率の少ない順に並び替える.また,並び替えを行った後,探索を初期状態に戻し,初期局所解の選択を行い,続いて局所解メッセージの送信を行う.

5 評価

前章で説明した各アルゴリズムの特徴を明確化するために計算機上にシミュレータを試作し,処理時間に関して評価を行った.

ここでは,簡単のために一つのエージェントは一つの変数のみを持つものとして評価した.

5.1 評価方法

評価対象として用いた分散制約充足問題について説明する.問題を生成するパラメータとして,エージェントの数 n,エージェントの持つ変数に割当るべき値の数 v,および問題の難易度を定義した.問題の難易度としては次の二つのパラメータを用いた.

- 制約違反を起こす確率 c
  任意の2変数にそれぞれ任意の値を割当てた時,その組合せが制約違反を起こす確率である.
- 制約ネットの複雑さ p
  問題を制約ネットで表した場合のエッジの存在確率.制約ネットが完全グラフの時この値は1となり,全く制約が存在しない場合は0となる.

上記のパラメータを使い,問題を計算機上で自動的に生成した.また各変数間の制約は乱数を用いて生成した.

自動生成した50の問題を各アルゴリズムを使用して解き,その結果を平均して評価値とした.各アルゴリズムは制約を満足する解を一つ導き出した時点で終了とし,それまでに必要とされた処理時間を評価した.

5.2 評価結果

図6,図7にそれぞれ制約違反を起こす確率に対する処理時間特性,制約ネットの複雑さに対する処理時間特性を示す.

ここで処理時間は制約チェックの回数で近似した.これは各エージェントの処理において制約チェックに要する時間が最も大きく,その処理量が呼び出される回数に比例するからである.

また,用意した50の問題の内,正常に解を導出した問題の数が40以上の時のみ評価対象とした(図6,図7では×印で問題を解く能力の限度を表す).

問題のパラメータとしてエージェント数nを20,各変数(エージェント)のとり得る値の数vを100に固定した.

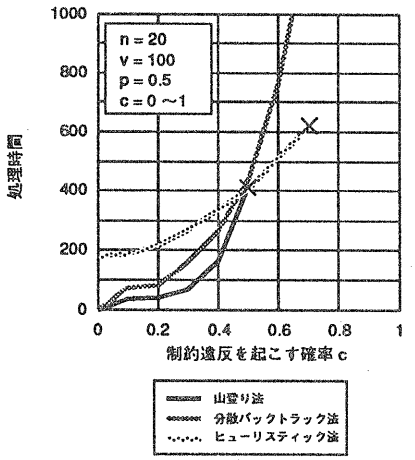

図 6: 問題の難易度 (制約違反を起こす確率 c) に対する処理時間特性

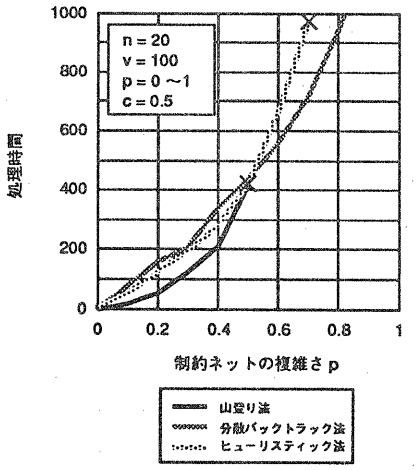

図 7: 問題の難易度 (制約ネットの複雑さ p) に対する処理時間特性

一般に問題の難易度が増すにつれて処理時間は増加する傾向を示している. また, 各アルゴリズムにおいて, 処理時間の増加する傾向に差が存在する. これより各アルゴリズムの特徴が導き出される.

1) 山登り法

このアルゴリズムは分散制約充足アルゴリズムとして最も簡単な手法である. 問題の難易度が低い領域では, 各エージェントが独立に割当てた値がそのまま解となる確率が高いために, アルゴリズムの処理の簡単さが効いて他のアルゴリズムよりも少ない処理時間で解に到達できる. しかしながら問題の難易度が高くなるにつれて, 独立に割当られた値が解となる確率が低くなり, 解に到達することなく終了する. 今回の評価では制約違反を起こす確率 c が 0.5 より大きい場合に解を導出する確率が 0.8 を下回った.

2) 分散バックトラック法

このアルゴリズムでは問題の難易度が低い部分で山登り法に比べて大きな処理時間が必要になっている. しかしながらこの手法では制約違反を起こした状況を記憶し, 局所解探索に反映することで確実に解に近付こうとする. このため山登り法で解を導くことが不可能である領域でも解を導くことが可能であり, 問題の難易度の高い領域では有効なアルゴリズムである.

3) ヒューリスティック法

このアルゴリズムでは初期オーバヘッド (図 6で制約違反を起こす確率 c が 0 の時) が存在する. これは自エージェント内の局所解候補の並び替えに必要な処理時間である.

一方, このアルゴリズムでは制約違反の起こる確率 c の増加にともなう処理時間の増加の傾きが他のアルゴリズムに比べて緩やかである. 原因としては, 局所解候補を評価し並び替えを行うことによ

り制約違反の起こる確率の少ない候補より局所解として選択していることがあげられる．また，山登り法と比較して問題の難易度の高い領域で解を導き出しているが，原因は同様である．

ところでこのアルゴリズムの制約ネットの複雑さに対する処理時間特性を調べると，他のアルゴリズムと比較してその増加傾向が大きいことがわかる．これはこのアルゴリズムが制約チェック処理を局所解候補の並び替えにおいても行っているためである．

6 まとめ

本論文では，現在までに報告されている分散制約充足アルゴリズムを一般化し，その枠組を与えると共に，各アルゴリズムの有効性を調べるために評価を行い，その特性を示した．

分散制約充足アルゴリズムの枠組としては，通信駆動型のアルゴリズムを示した．この枠組においてこれまで報告されてきた分散制約充足アルゴリズムを提案した枠組で説明した．また，三種の分散制約充足アルゴリズムを計算機上に実現し，以下の結果を得た．

- 問題の難易度の低い領域では，アルゴリズム自体が簡単な山登り法が有効である．
- 問題の難易度が高い領域では，全探索空間を調べる能力を持つ分散バックトラック法が有効に働く．
- ヒューリスティックの導入は制約ネットは単純であるが制約違反を起こす確率が大きい問題で効果的である．

参考文献

[1] A. H.Bond and L. Gasser (eds.): *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers,1988.

[2] S. C. Shapiro and D. Eckroth (eds.) *Encyclopedia of Artificial Intelligence*, pp. 205–211. Weley-Interscience Publication, 1987.

[3] M. Yokoo, T. Ishida and K. Kuwabara: Distributed Constraint Satisfaction for DAI Problems, *Proc.of 10th Workshop on Distributed Artificial Intelligence*, Chapter 18, 1990.

[4] Conry,S.E.,Meyer,R.E.and Lesser,V.R.: Multistage Negotiation in Distributed Planning, in A. H.Bond and L. Gasser (eds.) *Readings in Distributed Artificial Intelligence*, pp.367–384, Morgan Kaufmann Publishers,1988.

[5] Conry,S.E.,Kuwabara,K.,Lesser,V.R. and Meyer,R.A.: Multistage Negotiation for Distributed Constraint Satisfaction, to appear in *IEEE Transactions on Systems,Man and Cybernetics*, 1991.

[6] 山口，萩原，都倉 ： データ集中方式による分散アルゴリズムの通信計算量と理想時間計算量，電子通信学会技術研究報告，COMP86-83, pp.17–28, 1987.

[7] K.Sycara,S.Roth,N.Sadeh and M.Fox.: Decentralized Factory Scheduling: Coordinating Resources Allocation Using Constrained Heuristic Search, *Proc.of 10th Workshop on Distributed Artificial Intelligence*, Chapter 17, 1990.