論 文

Technical Papers

分散制約充足におけるnogood学習の効果

The Effect of Nogood Learning in Distributed Constraint Satisfaction

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/

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/

Keywords: constraint satisfaction problem, distributed CSP, multiagent system, nogood learning.

Summary

We present *resolvent-based learning* as a new nogood learning method for a distributed constraint satisfaction algorithm. This method is based on a look-back technique in the CSP literature and can efficiently make effective nogoods.

We combine the method with the *asynchronous weak-commitment search algorithm* (AWC) and evaluate the performance of the resultant algorithm on distributed 3-coloring problems and distributed 3SAT problems. As a result, we found that the resolvent-based learning works well compared to previous learning methods for distributed constraint satisfaction algorithms. We also found that the AWC with the resolvent-based learning is able to find a solution with fewer cycles than the *distributed breakout algorithm*, which was known to be the most efficient algorithm (in terms of cycles) for solving distributed CSPs.

1. はじめに

分散制約充足問題 [横尾 92, Yokoo 98a] とは，制約充足問題における変数と制約が複数のエージェントに分散された問題である．制約充足問題の定義自体は非常に単純なものであるにもかかわらず，人工知能における多くの問題が制約充足問題として定式化できることが知られている．同様に，マルチエージェントシステムにおける様々な応用問題，例えば，分散資源配分問題 [Conry 91]，分散解釈問題 [Mason 89]，マルチエージェントによるデータの真偽値管理 [Huhns 91] などは分散制約充足問題として定式化できる．そのため分散制約充足問題を解く効率的な分散アルゴリズムはマルチエージェントシステムにおける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．

筆者らは，分散制約充足問題を解くアルゴリズムの一つとして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 [横尾 96, Yokoo 98a]．非同期弱コミットメント探索アルゴリズムでは，すべてのエージェントが変数への暫定的な値を互いに交換しあい，非同期かつ並行的に変数への値を変更して，変数間の制約を満足しようとする．エージェントは別のエージェントから変数への最新の値を受け取ると，まず，*agent_view* とよばれるデータ（エージェントの識別子，変数の識別子，変数の値からなる三組のリスト）を更新し，そのagent_viewのもとで自分のもつ変数に対して無矛盾な値を探索する．しかし，あるagent_viewのもとでは，そのような値が存在しない行き詰まりの状況に陥る場合がある．

非同期弱コミットメント探索アルゴリズムでは，そのような場合，エージェントは *nogood* というデータを生成し，関連するエージェントに送信する．nogoodとはagent_viewの部分集合で，そのもとでは変数に対する無矛盾な値が存在しないというものである．このようなnogoodは，探索の途中で発見された新しい制約と見なすことができるため，nogoodを生成して関連するエージェントがそれを記録することをnogood学習，あるいは単に学習とよぶ．非同期弱コミットメント探索アルゴリズムは，生成したすべてのnogoodを記録することによりアルゴリズムの完全性が保証される [横尾 96, Yokoo 98a]．

過去の研究では，次のようなnogood学習法が提案されている．

- 非同期バックトラッキングアルゴリズム [横尾 92, Yokoo 98a] では，agent_viewそのものをnogoodとする方法が採用されている．この方法では，agent_viewの部分集合を調べる必要がないため，nogoodの生成にかかるコストは実質的にゼロである．しかし一般に，生成されるnogoodのサイズが大きいため，そ

れにより削られる探索空間のサイズは小さい．つまり，あまり効果的なnogoodは得られない．

- 非同期弱コミットメント探索アルゴリズムに関する過去の研究[横尾 96, Yokoo 98a]では，nogood学習をアルゴリズムの完全性を保証する手段と見なし，完全性を追求しない実用的立場から実際のアルゴリズムにおいては，nogoodは生成，記録されない*1．
- MammenとLesserは，agent_viewから最小矛盾集合を求め，それをnogoodとしている[Mammen 98]．最小矛盾集合とは，agent_viewの部分集合のうち，行き詰まりを引き起こすものでサイズが最小のものである．一般に，このようなnogoodはサイズが小さく，探索空間を大きく削ることができる．しかし，一般に最小矛盾集合を求めるには大きなコストがかかる．

本論文では，非同期弱コミットメント探索アルゴリズムにおける新しいnogood学習法として，resolvent-based learningを提案する．これは制約充足アルゴリズムでのlook-backテクニック[Cha 96, Frost 94, Ginsberg 93, Richards 98]をベースにしており，効率良く効果的なnogoodを生成することができる．また，この新しいnogood学習法の性能を分散3色問題，分散3SATを用いて評価する．

以下，本論文の構成は次の通りである．**2章**では，研究の背景として，分散制約充足問題と非同期弱コミットメント探索アルゴリズムの概要を述べる．**3章**では，新しいnogood学習法であるresolvent-based learningを説明し，**4章**で，その性能を評価する．最後に，**5章**にまとめと今後の課題を示す．

2. 背　　景

2·1 分散制約充足問題

制約充足問題は，有限かつ離散的な値域をもつ変数の集合と制約の集合からなる．制約とは，特定の変数集合上で定義され，それらの変数集合への値の割り当てを規定するものである．本論文では，制約を変数集合に対して禁止されている値の集合（nogood）として表現する．制約充足問題を解くとは，すべての制約を満たすように，すべての変数の値を決めることである．

分散制約充足問題とは，制約充足問題における変数と制約が複数のエージェントに分散された問題である．エージェントの目的は，自分のもつすべての制約を満たすように変数の値を決めることだが，異なるエージェントがもつ変数との間に定義された制約が存在するため，関連するエージェント間で通信を行なって，値を決める必要がある．分散制約充足問題の形式的な定義は次の通りである．

- エージェントの集合$\{1,2,\ldots,l\}$が存在する．
- 各変数に対し，それが属するエージェントが1つ定義される．また，各エージェントは自分に属している変数に関するすべての制約を知っている．
- すべてのエージェントiにおいて，iに属する任意の変数x_jの値がd_jに決定され，かつ，それらの値がiの知っているすべての制約を満たすとき，分散制約充足問題が解けたという．

また，アルゴリズムの設計の際には，エージェント間通信に関して次のようなメッセージ通信システムをモデルとして想定する．これは一般の分散アルゴリズムの設計において利用されるもので十分妥当なものである．

- 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らない．
- 任意の2エージェント間では，送信されたメッセージの順序は保存される．

2·2 非同期弱コミットメント探索アルゴリズム

非同期弱コミットメント探索アルゴリズムは，基本的には1エージェントが1変数しかもたない分散制約充足問題に対して設計された分散制約充足アルゴリズムである*2．

非同期弱コミットメント探索アルゴリズムでは，各変数（エージェント）に対して優先度が定義されている．エージェントは，まず，自分がもつ変数に対して適当な初期値を割り当て，*ok?*メッセージを使って，変数とその値，変数の優先度（初期値はゼロ）を関連するエージェントに送る．

*ok?*メッセージを受け取ると，エージェントiはagent_viewを更新し，そのagent_viewのもとで自分がもつ変数x_iの現在の値が制約を満たす（nogoodに違反しない）かどうか調べる．ただし，調べるのは変数x_iの優先度よりも高い優先度をもつnogoodだけである．ここでnogoodの優先度とは，そのnogood内の変数のうちx_i以外の変数で最も優先度が低い変数の優先度とする．例えば，エージェント5が変数x_5を，nogoodとして$((x_1, \mathrm{red})(x_2, \mathrm{green})(x_5, \mathrm{yellow}))$をもち，変数$x_1, x_2, x_5$の優先度がそれぞれ2, 1, 0だとすると，この場合，このnogoodの優先度は1であり，変数x_5の優先度よりも大きいので，エージェント5は変数x_5の値がこのnogoodに違反しないかどうかを調べる必要がある．なお，変数の優先度が等しい場合，変数の識別子の辞書式順序によ

*1 非同期弱コミットメント探索アルゴリズムでは，変数の優先度を上げることによって行き詰まりの状況を打開することができるため，nogood学習を行なわなくてもアルゴリズムの進行上問題はない．

*2 しかしながら，非同期弱コミットメント探索アルゴリズムは，1エージェントが複数の変数をもつ問題にも容易に拡張できる[Yokoo 98b]．

りタイブレークする．

以上のように変数の現在の値が nogood に違反するかどうかを調べ，その結果に応じてエージェント i は次の処理を行なう．

- 優先度の高い nogood に違反していなければ，何もしない．
- 優先度の高い nogood に違反していて，かつ，それが x_i の値を変更することによって修復可能であれば，エージェント i は値を変更し，*ok?*メッセージを送る．値の候補が複数個あれば，優先度の低い nogood の違反数がもっとも少ない値を選ぶ．
- 優先度の高い nogood に違反していて，x_i の値をどう変更してもそれが修復不可能なら，エージェント i は現在の agent_view から新しい nogood をつくり，関連するエージェントに *nogood* メッセージを使って送る*3．さらに，変数 x_i の優先度を上げ，x_i の値を違反となる nogood の数が最小となる値に変更し，それらの値を *ok?*メッセージを使って関連するエージェントに送る．

また，エージェントは *nogood* メッセージを受け取ると，その nogood を自分が管理する nogood の集合に追加し，変数の現在の値がその nogood に違反するかどうかをチェックする．なお，受け取った nogood が未知の変数を含む場合には，それをもつエージェントに今後その値を送るよう依頼することになる．

3. Nogood 学習

制約充足アルゴリズムにおける look-back テクニックでは，過去になされた探索により得られた情報を利用して，以降の探索を高速化する [Bayardo 96, Bayardo 97, Cha 96, Dechter 90, deKleer 89, Frost 94, Ginsberg 93, Richards 98]．本論文では，同様の look-back テクニックを使った，非同期弱コミットメント探索アルゴリズムにおける新しい nogood 学習法を提案する．本手法の基本的なアイデアは，文献 [Cha 96, Frost 94, Ginsberg 93, Richards 98] の手法に基づいている．これらの手法の特徴は，ある変数において値域内のどの値も nogood に違反するとき，各値についてそれを禁止する nogood を一つずつ選び，それらを合わせて一つの新しい nogood をつくることである．この新しい nogood は，命題論理における resolvent と実質的に同じものと見なすことができるため，本論文ではこのような nogood 学習を，*resolvent-based learning* とよぶ．

非同期弱コミットメント探索アルゴリズムでは，次のようにして resolvent-based learning を実現する．

まず，エージェント i が値域 D_i をもつ変数 x_i をもち，D_i 内のどの値も現在の agent_view のもとで x_i よりも優先度の高い nogood に違反するものとする．このときエージェント i は，D_i 内の各値 d について次のように一つの nogood を選択する．

(1) x_i よりも優先度が高い nogood のうち，現在の agent_view のもとで $x_i = d$ としたときに違反となるものをすべて挙げる．

(2) これらの nogood のうち，サイズが最小のものを 1 つ選択する．サイズ最小のものが複数あれば，そのうち最も優先度が高いものを選択する．

次に，エージェント i は，選択されたすべての nogood の和集合を求め，それから x_i に関する値を除いたものを新しい nogood とする．

なお，各値 d についてサイズが最小の nogood を選択するのは，新しい nogood のサイズをできるだけ小さくするためである．また，サイズ最小のものが複数あるときに，優先度が最も高いものを選択するのは，一般にそのような nogood が優先度の高い変数を含んでいるからである．非同期弱コミットメント探索アルゴリズムでは，優先度の高い変数ほど現在の値に強くコミットしているため，もしその値が誤っている場合には，nogood によりできるだけ早くその誤りを指摘した方が得策だと考えられる．

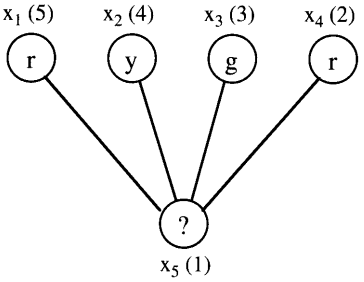

図 1 分散 3 色問題

以下，本手法を図 1 の具体例を用いて説明する．図 1 は，分散 3 色問題の例の一部を表しており，各エージェント i が節点 x_i を隣接する節点どうしが異なる色になるように全体を赤 (r)，黄 (y)，緑 (g) の 3 色に塗り分ける．節点名の横の括弧内の数字は，その節点（変数）の優先度を表している．今，エージェント 5 が節点 x_5 の色を決めようとする状況を考える．なお，図に描いていない節点はすべて x_5 よりも低い優先度をもつものとする．また，エージェント 5 は，図の枝に対応する nogood として $\{((x_1,\mathrm{r})(x_5,\mathrm{r})),\ ((x_1,\mathrm{y})(x_5,\mathrm{y})),\ ((x_1,\mathrm{g})(x_5,\mathrm{g})),\ \cdots,\ ((x_4,\mathrm{g})(x_5,\mathrm{g}))\}$ を，他のエージェントから受け取った nogood として，$((x_3,\mathrm{g})(x_4,\mathrm{r})(x_5,\mathrm{y}))$ をもつものとする．容易にわかる通り，この状況では節点 x_5 に対する無矛盾な色は存在せず，いずれの色も x_5 より優先度の高い nogood のどれかに違反してしまう．resolvent-based learning では，エージェント 5 はこの状況で次のようにして新しい nogood を生成する．

x_5 を赤 (r) にした場合，$((x_1,\mathrm{r})(x_5,\mathrm{r}))$ と $((x_4,\mathrm{r})(x_5,\mathrm{r}))$

*3 同じ nogood をすでに送っていれば処理を終える．これはアルゴリズムの完全性を保証するために必要となる．

に違反する．両方ともサイズは2で同じだが，優先度はそれぞれ5と2なので，赤に対しては優先度の高い前者を選択する．x_5 を黄（y）にした場合，$((x_2,\mathrm{y})(x_5,\mathrm{y}))$ と $((x_3,\mathrm{g})(x_4,\mathrm{r})(x_5,\mathrm{y}))$ に違反する．この場合は，サイズが小さい前者を選択する．x_5 を緑（g）にした場合，$((x_3,\mathrm{g})(x_5,\mathrm{g}))$ に違反するだけなので，これを選択する．以上，選択した3つのnogoodから，新しいnogoodとして $((x_1,\mathrm{r})(x_2,\mathrm{y})(x_3,\mathrm{g}))$ を生成する．

4. 評　価

resolvent-based learningの性能を分散3色問題と分散3SATを用いて評価する．

分散3色問題は，3色問題におけるグラフの n 個の節点（変数）と m 本の枝（制約）が複数エージェントに分散された問題である．この実験では，$m=2.7n$ の解がある3色問題を文献[Minton 92]の方法で生成し，各エージェントが1変数とそれに関連する制約をもつように，それぞれを分散させる．なお，この設定の問題は，3色問題の中でも特に困難な部類に入ることが知られている[Cheeseman 91]．この方法により，$n\in\{60,90,120,150\}$ の各 n について，10個の問題例を作成し，また各問題例について変数に対する初期値を10通り用意して，計100回の試行を行なう．

一方，分散3SATは，3SATにおける n 個の論理変数（変数）と m 個の節（制約）が複数エージェントに分散された問題である．この実験では，文献[Cha 95]の3SAT-GENと3ONESAT-GENという手続きで生成された3SATについて，各エージェントが1つの変数とそれに関連する制約をもつように分散させる．3SAT-GENは，特定の制約の数/変数の数の比の値に対して，解のある3SATの例を生成する．これを用いて，$m=4.3n, n\in\{50,100,150\}$ となる例を各 n につき25個生成し，また各例につき変数に対して4通りの異なる初期値を用意して，計100試行を行なう．また，3ONESAT-GENは，特定の制約の数/変数の数の比の値に対して，解が一つしかないような3SATの例を生成する．この実験では，$m=3.4n, n\in\{50,100,200\}$ の各 n について，DIMACSベンチマークサイト*4で公開されている4例を利用し，また各例について変数に対する初期値を25通り用意して，計100試行を行なう．

実験は，複数エージェントの並行動作をシミュレートするシミュレータ上で行なう．このシミュレータは同期型分散システムの動作を模擬するもので，すべてのエージェントは同期して，一連の行動サイクル（受信メッセージの読み取り，内部計算，関連エージェントへのメッセージ送信）を繰り返す．このシミュレータ上にアルゴリズムを実現し，各試行ごとに，解を見つけるまでに費やしたサイクル数（*cycle*）と最大nogoodチェック回数の総和（*maxcck*）を測定する．最大nogoodチェック回数の総和とは，各サイクルごとに最も多くのnogoodチェックを行なったボトルネックとなるエージェントを特定し，そのチェック回数の全サイクルでの総和のことである．アルゴリズムの評価は，各 n についてそれらの測定値の100試行平均により行なう．また，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10000とし，これを越えた試行については実行を打ち切り，その時点での値を測定値とする．

*4 ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/

4·1 他のnogood学習法との比較

まず，resolvent-based learningを次の2つのnogood学習法と比較する．

mcs-based learning 文献[Mammen 98]の方法と同様に最小矛盾集合（minimum conflict set）を新しいnogoodとして記録する．最小矛盾集合を見つける手順は，まずresolvent-based learningによりnogoodを生成し，その部分集合について大きいものから順に矛盾集合かどうかを調べ，最小の部分集合を求めるというものである．

no learning 文献[横尾 96, Yokoo 98a]の方法と同じで，行き詰まりにおいてnogoodを生成しない．

それぞれのnogood学習法を非同期弱コミットメント探索アルゴリズムに組み込み，実験を行なった．結果を表1～表3に示す．表中の%は設定した上限サイクル以内で解けた試行の割合を示している．

まず，resolvent-based learningとmcs-based learningを比較する．両手法は分散3色問題と3SAT-GENによる分散3SATにおいては，サイクル数（*cycle*）についてはほぼ同等，最大nogoodチェック回数の総和（*maxcck*）については，すべての場合においてresolvent-based learningの方が良くなっている．このことからresolvent-based learningは，これらの問題に対して有効なnogoodを少ない計算コストで生成できていることがわかる．

一方，3ONESAT-GENによる分散3SATにおいては，resolvent-based learningは，最大nogoodチェック回数の総和については常に良いが，サイクル数については，mcs-based learningよりも若干悪くなっている．この問題は，制約の数が変数の数に対して比較的少ない（$m=3.4n$）にもかかわらず解が1つしか存在しないという性質をもつことから，一般にサイズの小さい暗黙的なnogoodを多く含んでいると考えられる．そのようなnogoodは，mcs-based learningでは，多くのnogoodチェックを費しはするが，比較的早いサイクルで発見されるため，このような結果が得られたと考えられる．

次に，resolvent-based learningとno learningを比較すると，サイクル数（*cycle*）に関してはresolvent-based learningがno learningよりも圧倒的に良い．この理由を探るために，それぞれの手法において，あるエージェント

表 1 分散 3 色問題での他の nogood 学習との比較

n	*learn*	*cycle*	*maxcck*	%
60	Rslv	83.2	58084.4	100
	Mcs	88.8	119019.2	100
	No	458.2	52601.6	100
90	Rslv	125.4	135569.8	100
	Mcs	133.2	275099.1	100
	No	2923.9	358486.1	91
120	Rslv	178.5	263115.1	100
	Mcs	172.3	494266.7	100
	No	6121.9	793280.3	60
150	Rslv	173.9	273823.3	100
	Mcs	177.1	512657.0	100
	No	8800.5	1188345.1	21

表 2 3SAT-GEN による分散 3SAT における他の nogood 学習との比較

n	*learn*	*cycle*	*maxcck*	%
50	Rslv	125.0	76256.2	100
	Mcs	120.7	180122.0	100
	No	360.0	15959.3	100
100	Rslv	215.3	233003.8	100
	Mcs	238.9	830660.5	100
	No	3949.8	188182.3	80
150	Rslv	275.3	399146.6	100
	Mcs	286.0	1146204.1	100
	No	7793.8	382634.7	41

表 3 3ONESAT-GEN による分散 3SAT における他の nogood 学習との比較

n	*learn*	*cycle*	*maxcck*	%
50	Rslv	140.4	64011.0	100
	Mcs	120.3	90813.5	100
	No	1378.1	47784.3	62
100	Rslv	155.4	81086.1	100
	Mcs	138.2	132518.7	100
	No	9179.5	340172.3	14
200	Rslv	263.8	294334.5	100
	Mcs	237.4	544732.6	100
	No	-	-	0

が過去に生成した nogood と全く同じ nogood を生成した回数を測定し，その全エージェントでの合計を求めた*5．なお，no learning は nogood を生成しないが，この測定のために，行き詰まりにおいて resolvent-based learning により nogood を生成するが，それを関連するエージェントに送信しないように修正されている．結果を表 4 に示す．これを見ると，no learning では，エージェントが何度も同じ行き詰まり（厳密には，同じ nogood を生成する行き詰まり）に陥っているのに対して，resolvent-based learning では，その回数が劇的に減少している．このことから，nogood 学習によりエージェントが変数に対して適切な値を選べるようになり，その結果，早いサイクルで解を発見できたと考えられる．

一方，最大 nogood チェック回数の総和（*maxcck*）については，no learning は，エージェントの各サイクルでの負荷が軽いため，問題のサイズが小さい場合には resolvent-based learning よりも良い．しかし，問題サイズが大き

*5 過去に生成した nogood が逐一記録されていたとしても，エージェントの非同期な並行動作のため，このようなことが起こり得る．

表 4 同じ nogood を生成した回数の全エージェントでの合計 (100 試行平均)

problem	n	Rslv	No
分散 3 色問題	60	69.1	1612.3
	90	208.1	24399.3
	120	432.5	69784.6
	150	565.3	135502.5
分散 3SAT (3SAT-GEN)	50	195.3	1105.3
	100	908.0	42998.7
	150	1947.2	133162.6
分散 3SAT (3ONESAT-GEN)	50	276.6	5523.3
	100	651.9	86595.8
	200	2683.4	190501.8

くなるとサイクル数が増え，その結果，resolvent-based learning よりも悪くなっている．

4·2 サイズ限定の nogood 学習

nogood 学習には，新たに生成された nogood の数が多くなると，以降の探索においてそれらをチェックするコストが増大するという問題が存在する．ここでは，この問題に対処するため，生成した nogood すべてを記録するのではなく，有効と期待されるものだけを記録する．具体的には，k というパラメータを導入し，生成された nogood のうちサイズが k 以下のものだけを記録する [Dechter 90, Frost 94]．

非同期弱コミットメント探索アルゴリズムにおいて，このようなサイズ限定の resolvent-based learning を行なった場合*6の実験結果を表 5～表 7 に示す．表中の k はサイズ k 以下の nogood だけを記録することを表し，$k=\infty$ とはすべての nogood を記録することを示している．

分散 3 色問題では，$k=3$ の場合がサイズ未限定（$k=\infty$）の場合に比べて，サイクル数（*cycle*）ではほぼ同等，最大 nogood チェック回数の総和（*maxcck*）では良い結果が得られている．また，同様のことが，3SAT-GEN による分散 3SAT では $k=5$ の場合，3ONESAT-GEN による分散 3SAT では $k=4$ の場合について言える．この結果を見る限り，k についての最適な値の設定は問題依存であり，問題の特徴に応じて適宜設定する必要があるといえる．一般的な傾向としては，サイズ限定の resolvent-based learning は，k の値を小さくすると no learning に近くなり，各サイクルでのエージェントの負荷は小さくなるが，サイクル数は増加する．逆に，k を大きくすると，各サイクルでのエージェントの負荷は大きくなるが，サイクル数は減る傾向にある．

4·3 分散ブレイクアウト法との比較

筆者らは，分散制約充足問題を解く反復改善型のアルゴリズムである分散ブレイクアウト法を提案している [横尾 98c]．分散ブレイクアウト法では，各エージェントが違反制約の重み和をコスト関数とした山登り法の要領で

*6 すべての nogood は記録されないため，アルゴリズムの完全性は保証されなくなる．

表 5 分散3色問題でのサイズ限定 resolvent-based learning

n	k	*cycle*	*maxcck*	%
60	∞	83.2	58084.4	100
	3	85.6	40594.2	100
	4	90.6	66622.4	100
90	∞	125.4	135569.8	100
	3	126.4	76923.5	100
	4	136.0	151973.7	100
120	∞	178.5	263115.1	100
	3	171.8	124226.1	100
	4	167.3	217033.4	100
150	∞	173.9	273823.3	100
	3	186.1	153139.2	100
	4	180.4	249459.3	100

表 6 3SAT-GEN による分散 3SAT でのサイズ限定 resolvent-based learning

n	k	*cycle*	*maxcck*	%
50	∞	125.0	76256.2	100
	4	124.7	37717.9	100
	5	113.0	49770.3	100
100	∞	215.3	233003.8	100
	4	387.9	311048.8	100
	5	216.0	171115.7	100
150	∞	275.3	399146.6	100
	4	595.7	522191.2	100
	5	255.5	246534.5	100

表 7 3ONESAT-GEN による分散 3SAT でのサイズ限定 resolvent-based learning

n	k	*cycle*	*maxcck*	%
50	∞	140.4	64011.0	100
	4	130.8	38892.5	100
	5	128.9	46611.6	100
100	∞	155.4	81086.1	100
	4	167.8	68777.9	100
	5	162.8	84404.4	100
200	∞	263.8	294334.5	100
	4	265.7	181491.7	100
	5	272.6	290999.9	100

変数の値を変更する．処理に関する主な特徴は，値変更による効果を確かなものとするために，文献 [Hirayama 95] の方法と同様に各エージェントが関連するエージェント間でメッセージ通信を行なって値変更動作を相互排除すること，山登り法において疑似局所最適解 [横尾 98c] に陥った場合に各エージェントがブレイクアウト法 [Morris 93] を実行する（違反制約の重みを上げてコスト関数を変更する）ことの2点である．その結果，性能面の特徴として，アルゴリズムの完全性は保証されないが，解が存在する困難な問題に対して特に有効であることが知られている [横尾 98c]．

ここでは，サイズ限定の resolvent-based learning を用いた非同期弱コミットメント探索アルゴリズム（AWC+Rslv）と分散ブレイクアウト法（DB）の性能を比較する．表8～表10に結果を示す*7．表からわかる通り，非

表 8 分散3色問題における分散ブレイクアウト法との比較

n	*algorithm*	*cycle*	*maxcck*	%
60	AWC+Rslv($k=3$)	85.6	40594.2	100
	DB	164.9	7730.0	100
90	AWC+Rslv($k=3$)	126.4	76923.5	100
	DB	282.1	14228.5	100
120	AWC+Rslv($k=3$)	171.8	124226.1	100
	DB	522.4	26931.5	100
150	AWC+Rslv($k=3$)	186.1	153139.2	100
	DB	523.7	29207.0	100

表 9 3SAT-GEN による分散 3SAT における分散ブレイクアウト法との比較

n	*algorithm*	*cycle*	*maxcck*	%
50	AWC+Rslv($k=5$)	113.0	49770.3	100
	DB	322.6	6461.3	100
100	AWC+Rslv($k=5$)	216.0	171115.7	100
	DB	847.2	19870.8	100
150	AWC+Rslv($k=5$)	255.5	246534.5	100
	DB	1257.2	31717.2	100

表 10 3ONESAT-GEN による分散 3SAT における分散ブレイクアウト法との比較

n	*algorithm*	*cycle*	*maxcck*	%
50	AWC+Rslv($k=4$)	130.8	38892.5	100
	DB	690.1	11691.1	100
100	AWC+Rslv($k=4$)	167.8	68777.9	100
	DB	1917.4	38210.5	97
200	AWC+Rslv($k=4$)	265.7	181491.7	100
	DB	5246.5	117277.4	69

同期弱コミットメント探索アルゴリズムは，最大 nogood チェック回数の総和（*maxcck*）に関しては分散ブレイクアウト法よりも劣るが，サイクル数（*cycle*）に関しては優れているといえる．

実際にはどちらのアルゴリズムが効率的なのだろうか?

その答えは，エージェントの内部計算のコストとエージェント間の通信コストの比に依存すると考えられる．すなわち，エージェント間通信のコストがエージェントの内部計算コストに比べて十分に大きければ，サイクル数の面で優れている非同期弱コミットメント探索アルゴリズムの方が効率的であると考えられる．一方，その逆であれば，最大 nogood チェック回数の総和の面で優れている分散ブレイクアウト法の方が効率的だと考えられる．そのため，分散制約充足アルゴリズムが実行される環境に合わせて両者を使い分けることが現実的な選択であると思われる．

5. まとめと今後の課題

本論文では，非同期弱コミットメント探索アルゴリズムにおける新しい nogood 学習法である resolvent-base learning を提案し，その性能評価を行なった．その結果，この新しい nogood 学習法を用いることにより，同アルゴリズムの性能が改善されることが示された．

最後に今後の課題を述べる．今回の評価実験では，1エージェントが1変数をもつ問題だけを対象としたが，1エージェントが複数の変数をもつ問題 [Armstrong 97,

*7 分散3色問題に関して，この実験で用いた分散ブレイクアウト法と [横尾 98c] の実験で用いたものとは若干異なる．前者では，制約の重みを nogood 単位で定義しているのに対し，後者では変数ペア単位で定義している．実際，分散3色問題では前者の方が良い結果が得られる．

Yokoo 98b] に対してもこの nogood 学習法を適用することは容易である．今後，そのような一般的な問題においても十分な評価を行なう必要がある．

本論の中でも触れたように，nogood 学習では，すべての nogood を記録するとそのチェックにかかるコストが増大するという問題が存在する．今回は，記録する nogood のサイズに上限 k を設定し，サイズ k 以下の nogood だけを記録することにしたが，一般の問題においてその k をどう設定すべきかは未解決のままであり，今後検討する必要がある．また，サイズ以外の要因を考慮に入れた選択の可能性についても検討する必要がある．

◇参考文献◇

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

[Bayardo 96] Bayardo, R. J. and Miranker, D. P.: A Complexity Analysis of Space-Bounded Learning Algorithms for the Constraint Satisfaction Problem, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 298–304 (1996).

[Bayardo 97] Bayardo, R. J. and Schrag, R. C.: Using CSP Look-Back Techniques to Solve Real-World SAT Instances, in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pp. 203–208 (1997).

[Cha 95] Cha, B. and Iwama, K.: Performance Test of Local Search Algorithms Using New Types of Random CNF Formulas, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 304–310 (1995).

[Cha 96] Cha, B. and Iwama, K.: Adding New Clauses for Faster Local Search, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 332–337 (1996).

[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*, 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).

[Dechter 90] Dechter, R.: Enhancement Schemes for Constraint Processing: Backjumping, Learning, and Cutset Decomposition, *Artificial Intelligence*, Vol. 41, pp. 273–312 (1990).

[deKleer 89] deKleer, J.: A Comparison of ATMS and CSP Techniques, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 290–296 (1989).

[Frost 94] Frost, D. and Dechter, R.: Dead-end driven learning, in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 294–300 (1994).

[Ginsberg 93] Ginsberg, M. L.: Dynamic Backtracking, *Journal of Artificial Intelligence Research*, Vol. 1, pp. 25–46 (1993).

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

[Mammen 98] Mammen, D. L. and Lesser, V. R.: Problem Structure and Subproblem Sharing in Multi-Agent Systems, in *Proceedings of the Third International Conference on Multi-Agent Systems*, pp. 174–181 (1998).

[Mason 89] Mason, C. and Johnson, R.: DATMS: A Framework for Distributed Assumption based Reasoning, in Gasser, L. and Huhns, M. eds., *Distributed Artificial Intelligence*, Vol. 2, pp. 293–318, Morgan Kaufmann (1989).

[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).

[Morris 93] Morris, P.: The Breakout Method for Escaping From Local Minima, in *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 40–45 (1993).

[Richards 98] Richards, E. T. and Richards, B.: Non-systematic Search and Learning: An Empirical Study, in Maher, M. and Puget, J.-F. eds., *Principles and Practice of Constraint Programming –CP98*, Vol. 1520 of *Lecture Notes in Computer Science*, pp. 370–384, Springer-Verlag (1998).

[横尾 92] 横尾 真, Durfee, E. H., 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).

[横尾 96] 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 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).

[Yokoo 98b] Yokoo, M. and Hirayama, K.: Distributed Constraint Satisfaction Algorithm for Complex Local Problems, in *Proceedings of the Third International Conference on Multi-Agent Systems*, pp. 372–379 (1998).

[横尾 98c] 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：石塚 満〕

1999 年 8 月 9 日 受理

著者紹介

平山 勝敏（正会員）は，前掲（Vol.15, No.2, p.354）参照．

横尾 真（正会員）は，前掲（Vol.15, No.2, p.354）参照．