1E3-OS-4-9

節集合分割型分散SATに対する非同期バックトラッキングアルゴリズム

Asynchronous Backtracking Algorithms for Clause-set Partitioned Distributed SAT

下牧 昌太郎 Shotaro SHIMOMAKI
平山 勝敏 Katsutoshi HIRAYAMA

神戸大学大学院海事科学研究科
Graduate School of Maritime Sciences, Kobe University

We present M-ABTS and M-ABTDOS as new algorithms for solving clause-set partitioned distributed SAT, which consists of a set of CNF formulas that are distributed over agents but share some propositional variables. Major features of these algorithms are that agents solve their local CNF formulas by using any SAT solver and analyze conflicts by using the implication graph for learning nogoods efficiently. We provide an initial experimental result on the performance of these algorithms using some benchmark problem instances.

1. はじめに

マルチエージェントシステムにおける分散協調問題解決では，システム内のすべてのエージェントが互いに協力しながらある問題の解決を目指す．分散制約充足問題（DisCSP）[Yokoo 98] は分散協調問題解決の問題の一つであり，制約充足問題（CSP）における変数や制約が複数のエージェントに分散された問題である．DisCSP の目標は，分散されたすべての制約を満たす解が存在するか否か判定し，解が存在するならばそのうちの一つを求めることである．

一方，充足可能性判定問題（SAT）は CNF 式（CNF formula）の充足可能性を判定する問題であり，理論計算機科学，ソフトウエア科学，人工知能等の分野で数多くの研究が蓄積されている．SAT は，各命題変数を真または偽の値をとる変数，CNF 式の各節を制約と見なすことで CSP と見なすことができる．

本稿では，DisCSP の部分クラスである分散 SAT を扱う．特に，分散 SAT の自然な表現と考えられる節集合分割型分散 SAT，すなわち一部の命題変数を共有する CNF 式群が複数のエージェントに分散された SAT，を扱う．

この節集合分割型分散 SAT に対して，既存の DisCSP アルゴリズムを素朴に適用することも可能だが，より効率的な求解のためには SAT 固有の手法を導入することが望ましい．本稿では，DisCSP を解く代表的なアルゴリズムである非同期バックトラッキングアルゴリズム（ABT）[Yokoo 98] をベースとし，固有の手法として，1) 個々のエージェントが任意の SAT ソルバーを用いて自身の CNF 式を解く，2) 下位エージェントは，矛盾を発見した場合に単位伝播を用いた矛盾解析により学習節を生成する，ことを主な特徴とする新しいアルゴリズム M-ABTS（Multi-ABT for SAT）を提案する．さらに，エージェント間の優先順位を動的に変更するよう M-ABTS を拡張した M-ABTDOS（Multi-ABTDO for SAT）を提案する．

2. SAT

真または偽の値をとる命題変数 x とその否定 $\neg x$ を総称してリテラル（literal）とよぶ．このリテラルを論理和で結合したものを節（clause），節を論理積で結合したものを CNF 式とよぶ．CNF 式は，それを構成する節の集合と同一視でき，単に節集合ともよばれる．以下，本稿では CNF 式を節集合で表現する．

命題変数集合 X の各変数に対して真（1）または偽（0）を割り当てる関数 $X \to \{1, 0\}$ を真偽値割当て（truth assignment）とよぶ．与えられた CNF 式を真にするような真偽値割当てが存在する場合，CNF 式は充足可能（satisfiable）であるといい，そのような真偽値割当てを CNF 式のモデル（model）とよぶ．一方，そのような真偽値割当てが存在しない場合，CNF 式は充足不能（unsatisfiable）であるという．

3. 節集合分割型分散 SAT

従来の DisCSP では，エージェントが変数集合を分割して所有し，エージェント間で制約が共有される場合が多い．このように定式化された DisCSP を，本稿では変数集合分割型 DisCSP とよぶ．一方，変数集合分割型 DisCSP の双対表現に相当するのが制約集合分割型 DisCSP である．そこでは，エージェントが制約集合を分割して所有し，エージェント間では変数が共有される．

どちらの定式化が望ましいかは基本的に問題に依存する．分散 SAT の場合，各エージェントの問題に関する知識は基本的に節で記述されるため，制約集合分割型（すなわち節集合分割型）による定式化がより適切であると考えられる．

例 1 エージェントがそれぞれ次のような知識をもつとする．

- クエリエージェント：$Wind$（風が吹く）．$\neg BasinSell$（桶が売れない）．
- エージェント 1：$\neg Wind \vee Dust$（風が吹けば埃が舞う）．
- エージェント 2：$\neg Rat \vee BasinGnawed$（鼠が増えれば桶が傷む）．$\neg BasinGnawed \vee BasinSell$（桶が傷めば桶が売れる）．
- エージェント 3：$\neg Dust \vee Blind$（埃が舞えば目を病む人が増える）．$\neg Blind \vee Shamisen$（目を病む人が増えれば三味線が売れる）．
- エージェント 4：$\neg Shamisen \vee Cat$（三味線が売れれば猫が減る）．$\neg Cat \vee Rat$（猫が減れば鼠が増える）．

連絡先: 平山勝敏，神戸大学大学院海事科学研究科，〒 658-0022 神戸市東灘区深江南町 5-1-1，hirayama@maritime.kobe-u.ac.jp

表 1: 分散 SAT アルゴリズムの特徴比較

アルゴリズム	定式化	通信方式	完全性	局所問題
Multi-DB [Hirayama 05]	変数集合分割型	局所同期型	なし	複雑
PBR [Amir 05]	節集合分割型	同期型	あり	複雑
DPLL ABT [Ruiz 11]	変数集合分割型	非同期型	あり	単純
提案アルゴリズム	節集合分割型	非同期型	あり	複雑

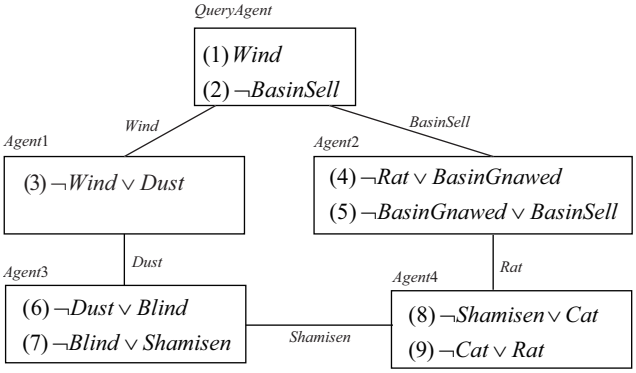

図 1: 節集合分割型分散 SAT

この状況は図 1 のような節集合分割型分散 SAT として定式化できる．各エージェントは節集合で表現された知識をもち，同じ命題変数を共有するエージェントはリンクで結ばれている．リンクに付けられたラベルは共有される命題変数である．

4. アルゴリズム

筆者らの知る限り，分散 SAT を解くアルゴリズムとして以下のアルゴリズムが提案されている．

Multi-DB [Hirayama 05]．変数集合分割型の分散 SAT を対象とし，局所同期型通信を行なう局所探索法に基づく非厳密解法である．複雑な局所問題にも対応する．

Partition-Based Reasoning [Amir 05]．節集合分割型の分散 SAT を対象とし，トークンベースの同期型通信を行なう木探索法に基づく厳密解法である．複雑な局所問題にも対応する．

DPLL ABT [Ruiz 11]．変数集合分割型の分散 SAT を対象とし，非同期型通信を行なう木探索法に基づく厳密解法である．基本的に 1 エージェント 1 変数の問題に対応する．

それに対し，本研究で提案するアルゴリズムは，節集合分割型分散 SAT を対象とし，非同期通信を行なう木探索法に基づく厳密解法であり，複雑な局所問題にも対応する．以上，従来アルゴリズムおよび提案アルゴリズムの特徴を表 1 にまとめる．

4.1 ABT

非同期バックトラッキングアルゴリズム（ABT）[Yokoo 98] は，非同期かつ並行に動作する最初の完全な分散制約充足アルゴリズムであり，基本的に 1 エージェント 1 変数のみからなる変数集合分割型 DisCSP を対象とする．ABT では，あらかじめエージェント間に任意の固定的な優先順位が導入されており，エージェントは基本的に 2 種類のメッセージ（ok?メッセージと nogood メッセージ）を利用する．詳細は次の通りである．

- まず，優先順位の高い上位エージェントが変数の値を決め，ok?メッセージでそれを下位エージェントに送る．
- ok?メッセージを受け取った下位エージェントは，上位エージェントとの制約を満たすように変数の値を変更する．その際，もし変数の値を変更する必要がなければ何もしないが，値を変更した場合は同様に ok?メッセージを使って自分よりも下位のエージェントにその値を知らせる．一方，上位エージェントとの制約を満たす値がない場合には，その原因となっている上位エージェントの変数の値の組合せ（nogood）を求め，それを最も優先順位の低い上位エージェントに nogood メッセージで送る．
- nogood メッセージを受け取ったエージェントは，現在の状態をチェックして nogood で指摘された状態と同じなら変数の値を変更するが，そうでなければ無視する．

ABT は，解が存在するならば，いずれその一つに収束し，解が存在しなければ，いずれ空の nogood が生成される．つまり，ABT は有限時間で必ず答えを返すことができる．

4.2 動的優先順位変更を行なう ABT

前述の通り ABT では，エージェント間に任意の固定的な優先順位が導入されている．この優先順位はアルゴリズムの性能に大きく影響することが経験的に知られているが，問題例に依存しない最適な優先順位を事前に求めることは困難である．そこで，アルゴリズムの実行中にエージェント間の優先順位を動的に変更し，ABT を効率化することが提案されている．

4.2.1 AWC

非同期弱コミットメント探索アルゴリズム（AWC）[Yokoo 98] は，動的な優先順位変更を行なう ABT ベースの分散制約充足アルゴリズムである．ABT からの主な変更点は，下位エージェントがある上位エージェントに nogood メッセージを送る際，近傍（制約を共有しているエージェントの集合）のどのエージェントよりも優先順位が上になるように自分の順位を上げる点である（図 2(a)）．AWC は，エージェントが受信した nogood をすべて記録して制約として扱うことにより，アルゴリズムの完全性を保証できる．しかし，nogood をすべて記録するには指数サイズのメモリを必要とする．従って，実用的には nogood を適宜捨てることでメモリの消費を抑える工夫がなされる [平山 00]．この場合，アルゴリズムとしての完全性は一般に失われる．

4.2.2 ABTDO

ABTDO [Zivan 05] は，nogood によるメモリ消費量を多項式サイズに抑えながら動的な優先順位変更を実現した ABT ベースの分散制約充足アルゴリズムである．その基本アイデアは，下位エージェントがある上位エージェントに nogood メッセージを送る際，nogood の送り先であるエージェントのすぐ下に位置するように自分の優先順位を上げるというもので

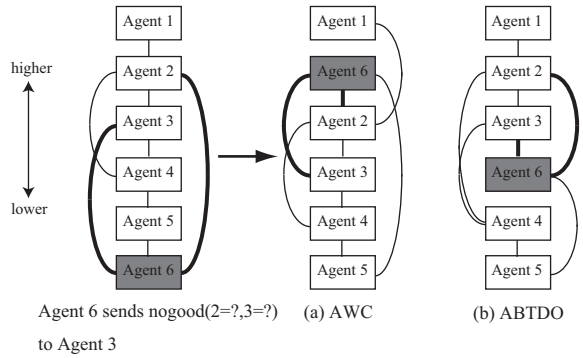

図 2: 動的な優先順位変更

ある*1（図 2(b)）．ABTDO では，ok?メッセージと nogood メッセージ以外にエージェント間の新しい優先順位を提案する order メッセージが利用される．

4.3 M-ABTS と M-ABTDOS

節集合分割型分散 SAT に対して，既存の DisCSP アルゴリズムを素朴に適用することも可能だが，より効率的な求解のためには SAT 固有の手法を導入することが望ましい．本稿では，ABT をベースとし，固有の手法として，1) 個々のエージェントが任意の SAT ソルバーを用いて自身の CNF 式を解く，2) 下位エージェントは，矛盾を発見した場合に単位伝播を用いた矛盾解析により学習節を生成する，ことを主な特徴とする新しいアルゴリズム M-ABTS（Multi-ABT for SAT）を提案する．さらに，エージェント間の優先順位を動的に変更するよう M-ABTS を拡張した M-ABTDOS（Multi-ABTDO for SAT）を提案する．

4.3.1 SAT ソルバーの利用

エージェントの局所問題が複数変数からなる DisCSP を解く場合，代表的な厳密解法に Multi-ABT [Hirayama 04] と Multi-AWC [横尾 00] があるが，いずれも各エージェントは局所問題を解く際に独自の解法を利用する．一方，節集合分割型分散 SAT では，各エージェントの局所問題は一般に複数の命題変数からなる節集合となるため，特に節集合のサイズが大きい場合には，独自の解法を設計するよりも既存の SAT ソルバーを利用する方が得策だと思われる．提案アルゴリズムでは，エージェントの局所問題を解く際に代表的な高速 SAT ソルバーである MiniSAT 2.20 [鍋島 10] を利用する．

例 2 例 1 で，エージェント 2 は $Rat = 0$, $BasinGnawed = 0$, $BasinSell = 0$ として節 (4)(5) を満たし，ok?メッセージで $Rat = 0$ をエージェント 4 に送る．同様に，エージェント 3 は $Dust = 1$, $Blind = 1$, $Shamisen = 1$ として節 (6)(7) を満たし，ok?メッセージで $Shamisen = 1$ をエージェント 4 に送る．それらを受け取ったエージェント 4 は節集合 $\{\neg Rat, Shamisen, \neg Shamisen \vee Cat, \neg Cat \vee Rat\}$ を MiniSAT 2.20 を用いて解く．その結果，$Rat = 0$ および $Shamisen = 1$ のもとでは自身の節集合は充足不能（UNSAT）と分かる．

*1 [Zivan 05] では，ABTDO は優先順位を任意の順序に変更できるアルゴリズムであり，本文中のように優先順位を変更するバージョンは ABTDOng と呼ばれている．

4.3.2 矛盾解析による節学習

1 エージェント 1 変数の DisCSP を対象とする ABT では，上位エージェントとの制約を満たす値がない場合，下位エージェントは nogood を生成する．nogood の典型的な生成方法は，変数の各値についてそれを禁止する上位エージェントの変数の値の組合せを一つずつ選択し，それらの和集合を nogood とするというものである [平山 00]．1 エージェント複数変数の DisCSP に対応する Multi-ABT と Multi-AWC でも，nogood 生成についてはエージェント内部の変数単位で同様の処理を行なっている．しかし提案アルゴリズムでは，エージェントは局所問題を既存 SAT ソルバーで解くため，このような変数単位の処理を別途行なうことは非効率的である．

一方，最近の高速 SAT ソルバーでは，通常，単位伝播による含意関係を表す含意グラフを使って矛盾の原因を解析し nogood に相当する学習節を生成する [鍋島 10]．提案アルゴリズムではこれと同様の方法を用いる．手続きの概要は次の通りである．

1. エージェントの節集合が UNSAT だと判明した場合，上位エージェントに由来する単位節（例 2 の $\neg Rat$ と $Shamisen$ に相当）およびそれ以外の単位節を起点として繰り返し単位伝播を適用する．その際，簡単化された節にはその簡単化の原因となった上位エージェントに由来する単位節を「理由」としてすべて記録しておく．
2. 単位伝播が終了すると，その終了状態に応じて次のように学習節を生成する．
   (a) 空節（矛盾）が得られた場合，その「理由」（上位エージェントに由来する単位節群）に対して各要素の否定の論理和を学習節とする．なお，空節が複数得られた場合，それぞれについて同じ要領で学習節を生成する．
   (b) 空節を含まない節集合が得られた場合，残されたすべて節の「理由」の和集合に対して各要素の否定の論理和を学習節とする．

例 3 例 2 で，エージェント 4 は $Rat = 0$ および $Shamisen = 1$ のもとで UNSAT である．このときエージェント 4 は，$\neg Rat$, $Shamisen$ を起点にもとの節集合上で単位伝播を行い，順に

$$\{\neg Rat,\ Shamisen,\ \neg Shamisen \vee Cat,\ \neg Cat \vee Rat\}$$

$$\Rightarrow \{Cat\,(Shamisen),\ \neg Cat\,(\neg Rat)\}$$

$$\Rightarrow \{\square\,(Shamisen, \neg Rat)\}$$

のように節集合を簡単化する．なお，簡単化された各節の「理由」を丸括弧内に示している．ここでは空節が 1 つ得られたので，エージェント 4 は，その「理由」から $Rat \vee \neg Shamisen$ という学習節を生成してエージェント 3 に送る．

4.3.3 動的な優先順位変更

ABT の場合と同様に，M-ABTS においてもエージェント間の優先順位を動的に変更することができる．本研究では，ABTDO に対応するアルゴリズムとして M-ABTDOS（Multi-ABTDO for SAT）を提案する．

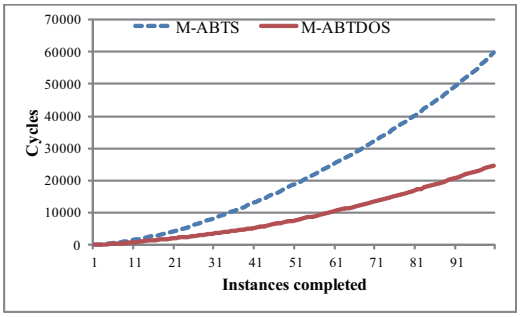

図 3: ランダムな問題例（uf20-91）におけるサイクル数の cactus プロット

5. 評価

M-ABTS と M-ABTDOS の性能を評価するための実験を行なう．本実験では，まずランダムな問題例として，SATLIB ベンチマーク問題 uf20-91（20 変数 91 節）の 01 番から 0100 番までの問題例 100 問を用いる．なお，節集合分割型分散 SAT とするために，各問題例について 91 個のエージェントを用意し，各節を 1 エージェントに割当て，同じ変数を共有するエージェントどうしをリンクで結ぶ．さらに，ランダムではなく何らかの構造をもった問題例として，University of Artois の Gauvain Bourgne 博士が作成した分散 SAT の問題例 14 問を用いる．

提案アルゴリズムは同期型離散事象シミュレータ上に Java 言語で実装されている．実装環境は，CPU: Intel Core i7-870@2.93GHz, Memory: 8GB, OS: Ubuntu 11.04, Compiler: Java 1.6.0_22 である．アルゴリズムの評価のための指標として，本シミュレータではアルゴリズムのサイクル数（cycle）と模擬実行時間（simulated runtime）を測定した．サイクル数とは，シミュレータ上でのメッセージ受信，計算，メッセージ送信を 1 サイクルとするもので，DisCSP アルゴリズムの評価の際に伝統的に利用されている．一方，模擬実行時間とは，各メッセージについてそれが発信されるまでに消費された累積経過時間を記録しておき，全メッセージ中でのその最大値である．

図 3 はランダムな問題例での各アルゴリズムの cactus プロットで，x 軸は解かれた問題例の個数，y 軸は，x 軸に対応する個数の問題例を解くのに要したサイクル数である．この cactus プロットは x 軸に近いほどよい．

表 2 は構造のある問題例での各アルゴリズムの模擬実行時間である．なお，1 回のランについてシミュレータの最大実行時間を 3600 秒とし，これを越えた場合は処理を打ち切った．打ち切った場合，表中に T.O. と表記している．

今回の実験結果では，ランダムな問題例では M-ABTDOS の方が明らか優れているが，構造のある問題例では必ずしも M-ABTDOS が優れているとは言えない．ランダムな問題例ではエージェントは 1 節しかもたないが，構造のある問題例ではエージェントは比較的多くの節をもつため，M-ABTDOS のようにエージェント単位で優先順位を変更すると不必要に優先順位を変えてしまい変更の効果があまり出ないと考えられる．また，構造のある問題例では 4 つの例で両者ともタイムアウトした．今後，効率化のための方策を考える必要がある．

表 2: 構造をもった問題例における模擬実行時間 [sec]

Instance	#var	#cls	#ag	M-ABTS	M-ABTDOS
flat30-1-8	90	300	8	0.518	**0.429**
flat30-1-32	90	300	30	**0.906**	0.979
ii8a1-4	66	186	4	**0.211**	0.350
ii8a1-32	66	186	32	0.663	**0.597**
medium-8	116	953	8	**0.822**	0.844
medium-64	64	254	64	**6.571**	8.399
par8-1-c-4	64	254	4	**0.738**	0.887
par8-1-c-16	64	254	16	4.284	**2.350**
ssa7552-32	1363	3032	32	**1.278**	1.532
ssa7552-256	1363	3032	256	4.860	**3.618**
logi-a-64	828	6718	64	T.O.	T.O.
logi-a-512	828	6718	481	T.O.	T.O.
logi-b-64	843	7301	62	T.O.	T.O.
logi-b-512	843	7301	485	T.O.	T.O.

6. まとめ

本稿では，節集合分割型分散 SAT に対して，DisCSP を解く代表的なアルゴリズムである ABT をベースとした M-ABTS およびその拡張版である M-ABTDOS を提案した．今後の課題はアルゴリズムの効率化である．

参考文献

[Amir 05] Amir, E. and McIlraith, S.: Partition-Based Logical Reasoning for First-Order and Propositional Theories, *Artif. Intell.*, Vol. 162, No. 1–2, pp. 49–88 (2005)

[平山 00] 平山 勝敏, 横尾 真：分散制約充足における nogood 学習の効果, 人工知能学会誌, Vol. 15, No. 2, pp. 355–361 (2000)

[Hirayama 04] Hirayama, K. and Yokoo, M.: An Easy-Hard-Easy Cost Profile in Distributed Constraint Satisfaction, *IPSJ Journal*, Vol. 45, No. 9, pp. 2217–2225 (2004)

[Hirayama 05] Hirayama, K. and Yokoo, M.: The Distributed Breakout Algorithms, *Artif. Intell.*, Vol. 161, No. 1–2, pp. 89–115 (2005)

[鍋島 10] 鍋島 英知, 宋 剛秀：高速 SAT ソルバーの原理, 人工知能学会誌, Vol. 25, No. 1, pp. 68–76 (2010)

[Ruiz 11] Ruiz, E.: Distributed SAT, *Artificial Intelligence Review*, Vol. 35, No. 3, pp. 265–285 (2011)

[Yokoo 98] Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K.: The Distributed Constraint Satisfaction Problem: Formalization and Algorithms, *IEEE Trans. Knowledge and Data Engineering*, Vol. 10, No. 5, pp. 673–685 (1998)

[横尾 00] 横尾 真, 平山 勝敏：複雑な局所問題に対応する分散制約充足アルゴリズム, 人工知能学会誌, Vol. 15, No. 2, pp. 348–354 (2000)

[Zivan 05] Zivan, R. and Meisels, A.: Dynamic Ordering for Asynchronous Backtracking on DisCSPs, *CP-2005*, pp. 32–46 (2005)