1D4-OS-02a-2

コミュニティ構造を利用した並列SATソルバの学習節共有手法の提案

Learnt Clause Sharing in Parallel SAT Solvers by Using Community Structure

薗部 知大 *1*2
Tomohiro Sonobe

*1国立情報学研究所 National Institute of Informatics
*2JST ERATO 河原林巨大グラフプロジェクト JST, ERATO, Kawarabayashi Large Graph Project, Japan

Learnt clause sharing in parallel SAT solvers is important. However, it is not desirable to share ones that are deleted without being used. This paper introduces a new method based on the community structure of SAT instances to share learnt clauses. Our method reduces the amount of shared clauses without deteriorating the solver performance.

1. はじめに

充足可能性問題 (Satisfiability problem, SAT 問題) は重要な NP 完全問題の一つとして知られている。この問題を解くプログラムは SAT ソルバと呼ばれ、現実世界の様々な問題 (ソフトウェア・ハードウェアの検証、計画生成等) を解くための基盤として使われているため、より大きな問題をより高速に解く意義は高い。

近年の多くの SAT ソルバはバックトラック探索型の Davis-Putnam-Logemann-Loveland (DPLL) アルゴリズムに、変数の値割り当てからの矛盾発生時に節を学習する Conflict-Driven Clause Learning (CDCL) を加えたものが主流である。並列 SAT ソルバでは、この DPLL+CDCL ベースの逐次 SAT ソルバをワーカーとして用い、ポートフォリオ戦略 [8] を採用しているものが多い (ManySAT、Plingeling、Penelope 等)。ポートフォリオ戦略では分割統治法のような探索空間の分割は行わず、各ワーカーは全空間を対象とする探索を行い、一番速く解を見つけたワーカーの解を全体の解として採用する。そのため、同一のソルバをワーカーとして使用しては並列効果が期待できないので、互いに異なる挙動を行うソルバをワーカーに採用することで、全体として異なる空間への探索を行うことが重要である。ポートフォリオ戦略では各ワーカーが独立に探索している状態であるため同期処理がほぼ必要なく、一つのワーカーの探索が中断してしまっても他のワーカーには影響がないため耐障害性にも優れているという利点を持つ。しかし、一般的には学習節を共有することでより性能が高まることが知られているため、多くのソルバではワーカー間で学習節の共有を行っている。

ポートフォリオ型の並列 SAT ソルバの学習節の共有に関して、全ての学習節を共有するのではなく、探索に役立つ長さの短いものや Literal Block Distance (LBD) [13] の低いものが共有の対象になる。しかし、SAT ソルバの探索では多くの節を学習し、すべての節を保存しておくとメモリの消費が顕著になり、その管理コストが探索の妨げになるため、定期的に役に立たない学習節を削除する機能が組み込まれている。そのため、他のワーカーから輸入した学習節が探索に影響を与えること無く削除されることが多々観測される。本質的に、それらの学習節を共有することは該当するワーカーの管理コストが増大するだけであるため、その共有を避けることで処理性能の向上が期待できる。

本研究では、対象の SAT 問題を、変数を頂点、同一節内に存在し合う変数間の関係を辺としたグラフ (Variable Incidence Graph, VIG) として考え、そのグラフから抽出されるコミュニティ構造に基づく節共有方法を提案する。コミュニティとはグラフにおける頂点集合の分割であり、コミュニティ内はエッジが密に連結し、コミュニティ間ではエッジが疎に連結しているほどよいコミュニティ分割とされている。コミュティ分割の良さを測る尺度の一つとしてモジュラリティ[5] があり、このモジュラリティを最大化するアルゴリズムが数多く提案されている [4, 11]。SAT 問題の VIG は高いモジュラリティを備えていることが示されていて [1]、コミュニティ構造を利用した SAT ソルバの解析 [10] や性能改善 [14, 2] がされている。一つのコミュニティを構成する変数集合は互いに密な関係にあり、SAT ソルバの探索は個々のコミュニティ内の変数に値を割り当てて探索を進めて行くと考えられる。本提案では学習節を構成する変数が所属するコミュニティを元に、各ワーカーの直近の探索状況から共有がのぞましい学習節を割り出し、対象の学習節が探索に影響を与えられると推測されるワーカーにのみ共有を行う。この手法を並列 SAT ソルバの Penelope に実装し、SAT Competition 2014 のアプリケーション部門の 300 問に対して実験を行ったところ、ソルバの性能を保ったまま、各ワーカーにおいて共有したが使われずに削除された学習節の割合の減少を確認できた。

本論文の構成は以下の通りである。第 2. 節では SAT 問題と SAT ソルバについて概説し、第 3. 節では SAT 問題とグラフのコミュニティ構造に関して説明する。第 4. 節では並列 SAT ソルバにおけるコミュニティ構造を利用した学習節の共有手法を提案し、第 5. 節では実験結果を報告する。最後に第 6. 節で本論文のまとめを述べる。

2. SAT 問題と SAT ソルバ

一般的に SAT 問題は**乗法標準形** (Conjunctive Normal Form, **CNF**) で与えられ、**節** (clause) と呼ばれる論理式が論理積で連結されている。節は**リテラル** (Boolean 変数の正か負の出現) が論理和で連結された論理式である。与えられた CNF を真にするためにはすべての節を真にする必要があり、一つの節を真にするためには対象の節内のリテラルのうち最低一つを真にしなければならない。

SAT ソルバは SAT 問題を解くためのプログラムであり、

連絡先: (e-mail:tominlab@gmail.com)

変数に真偽 ($True$ or $False$) の値を代入していく二分木の探索を行う。SAT ソルバは、まず値を割り当てる変数の選択 (**decision**) を行い、それに付随する強制的な値割り当ての伝播 (**propagation**) を行う。propagation では、ある節が一つのリテラルを除く他の全てのリテラルが偽になる (リテラル a なら $a = False$、リテラル $\neg a$ なら $a = True$ の値割り当て) と、この節 (**単位節**と呼ぶ) を真にするためには残るリテラルを真にするしか無いため、このリテラルに強制的に真となる値を割り当てる。decision と propagation を繰り返す途中に、ある節の全てのリテラルが偽になると (この節を empty clause と呼ぶ)、対象の CNF は偽となってしまうため値割り当てに**矛盾** (conflict) が生じる。矛盾が生じると一つ前の変数割り当てをやり直す**バックトラック**を行う。このアルゴリズムは Davis-Putnam-Logemann-Loveland (**DPLL**) アルゴリズムと呼ばれ、近年の多くの SAT ソルバの基盤アルゴリズムとして使用されている。

DPLL アルゴリズムに加えて、矛盾発生時にそれまでの値割り当てを解析して新たな節を学習する Conflict-Driven Clause Learning (**CDCL**) [12] が多くのソルバで採用されている。学習節が加わることで対象の問題の大きさは増大するが探索空間の枝刈りが促進される。一方で、全ての学習節を保存しておくとメモリの消費が著しくなるため、管理コストを下げるために一定期間毎に一定数の学習節を削除することがある。節の長さが短いほど枝刈りが進むため、従来は節の長さを基準に削除をしていたが、近年では Literal Block Distance (**LBD**) を学習節の評価尺度として用いているソルバが多い。

ある変数に対する decision とそれに付随する propagation によって値が割り当てられる変数の集合を**リテラルブロック**と呼び、LBD は節内の異なるリテラルブロックの総数である。例えば矛盾発生時に節 ($a \vee b \vee c \vee d$) が学習され、a と b、c と d がそれぞれ同一のリテラルブロックに属している時、この節の LBD は 2 となる。同一リテラルブロックに属する変数集合はお互いに値割り当て関係が propagation によって依存していると考えられるため、例えば LBD が 2 の節は異なるブロックに属する二つの変数に値が割り当てられると節内の全ての変数に値が割り当てられる可能性が高い。そのため低い LBD を備える節は、長さが短い節と実質同等の振る舞いをすることが期待できる。近年の多くのソルバは LBD を実装していて、LBD の高い学習節を優先的に削除している。

そのほかに、直近の学習節内の変数を優先的かつ動的に変数を選択する Variable State Independent Decaying Sum (**VSIDS**) [9] が decision のヒューリスティックとして標準的に使われていて、探索を最初からやり直す**リスタート** [6] 等が組み込まれている。近年の代表的なソルバとして、MiniSAT、Lingeling、Glucose、GlueMiniSat 等が挙げられる。

並列 SAT ソルバは DPLL+CDCL に基づく逐次の SAT ソルバをワーカーとして使用し、**ポートフォリオ戦略** [8] で探索を行うものが主流である。ポートフォリオ戦略では分割統治法のような探索空間の分割を行わず、各ワーカーが全探索空間を対象とした探索を行い、一番速く解を発見したワーカーの解を全体の解として採用する。ポートフォリオの利点は煩雑な探索空間の分割が必要なく、並列化が容易である等が挙げられる。並列化の効果を上げるためには、各ワーカーが同一の探索を行っても意味が無いため、可能な限り各ワーカーの探索を異なるものにする必要がある。これは **diversification** [7] と呼ばれ、用いるソルバを変えたり、ソルバの探索パラメータを変更することで実現する。一方で、ワーカー間の協調も重要であり、各ワーカーの学習節を共有することが望ましい。全ての学習節を共有すると通信のオーバーヘッドが大きくなるため、長さの短いものや LBD の低いものが対象になる。近年の代表的なソルバとして、ManySAT、Plingeling、Penelope 等が挙げられる。

3. SAT 問題のコミュニティ構造

SAT ソルバが数万以上の変数を含む現実世界から派生した問題を実用的な時間で処理が可能な理由の一つに、SAT 問題に内在する問題構造の存在が挙げられる。回路検証や計画生成等の現実世界の問題から変換して生成された SAT 問題には、一つの変数の値が決まると、長さの短い節を通して、その変数に関連の深い変数の値も propagation によって値が連鎖的に決まることが多々ある。このような特徴が問題構造の一種として考えられているものの、これまでにこの構造に関する明確な定義は存在しない。

この構造をとらえるために、SAT 問題をグラフとして扱い、そのグラフの構造性を**モジュラリティ**という観点から解析した研究が近年紹介された [1]。この研究では SAT 問題を、変数を頂点、同一節内に登場する変数間の関係を辺とする Variable Incidence Graph (**VIG**) に変換している。与えられた SAT 問題の変数の集合を $X = \{x_1, ..., x_n\}$、節の集合を Γ とすると、VIG は辺に重みがある無向グラフ $G(X, w)$ で定義され、w は辺の重みの関数で、

$$w(x, y) = \sum_{c \in \Gamma, x, y \in c} \frac{2}{|c|(|c| - 1)} \quad (1)$$

が成り立つ。ここで $|c|$ は節 c の長さで、長さが 1 の節は propagation により簡約化済みであると仮定している。なお、x と y に辺が存在しない場合は $w(x, y) = 0$ とする。これはつまり、長さ k の節に対して、内部の全変数ペアに辺をはり、その重みは辺の総数 $\frac{k(k-1)}{2}$ を逆数とした完全グラフを構築して、これを全ての節に対して組み合わせたグラフであるといえる。このグラフに対して、コミュニティ検出アルゴリズムを適用することで、辺が密につながった頂点集合 (コミュニティ) を発見することができる。検出したコミュニティは、コミュニティ内部で辺が密に、コミュニティ間では辺が疎になっていることが望ましく、モジュラリティはその度合いを測る指標である。頂点 x の次数を $deg(x) = \sum_{y \in X} w(x, y)$ とし、VIG に対する頂点集合の分割 $P = \{P_1, ..., P_n | X = \sum_{i \leq n} P_i, i \neq j$ の時 $P_i \cap P_j = \emptyset\}$ が与えられたとき、この分割のモジュラリティは

$$Q(G, P) = \sum_{P_i \in P} \frac{\sum_{x,y \in P_i} w(x, y)}{\sum_{x,y \in X} w(x, y)} - \left(\frac{\sum_{x \in P_i} deg(x)}{\sum_{x \in X} deg(x)} \right)^2 \quad (2)$$

と定義される。モジュラリティは $[0, 1]$ の値を取り、ランダムな分割を行うと 0 に近い値となり、0.3 以上の値は高い構造性を示している [5]。このモジュラリティを目的関数としてその最大化を目指すコミュニティ検出アルゴリズムが数多く提案されている (louvain 法 [4] や label propagation 法 [11] 等)。変数 (VIG の頂点) が密につながっている集合という意味で、コミュニティは SAT 問題の構造を表現したものであり、実際の SAT 問題のモジュラリティが 0.9 を超えているものも報告されている [1]。他にもコミュニティ構造を利用した SAT ソルバの解析 [10] や性能改善 [14, 2] がされている。

4. コミュニティ構造を利用した節共有

並列 SAT ソルバでは、長さが短い (または LBD が小さい) 学習節が全ワーカー間で共有されることでより高速な探索が実現できる。しかし、他のワーカーから得た学習節が必ずしも探索の役にたつ訳ではなく、中には全く影響を与えること無く削除されてしまうことがある。結果的にそのような節の共有は通信コストとメモリ消費を増大させるだけであるため、避けることが望ましい。本論文において、探索に影響を与えない節というのは、その節が探索中に単位節にならなかった (propagation の対象にならなかった) ものを指す。節内のリテラルが真になってしまうことを除けば、探索に影響を与えない節というのは、内部のリテラルがその時点の探索の対象になっていなかったことが考えられる。

SAT 問題を VIG として見ると、それに対するコミュニティは互いに関連の深い変数の集合と考えられる。つまり、あるコミュニティ内の変数に値が割り当てられると、そのコミュニティ内の変数全体を対象とする探索 (値割り当て) が進行していくと予想される。ポートフォリオ型の並列 SAT ソルバにおける各ワーカーは、diversification によって異なる探索を行うため、異なるコミュニティに対する探索が並列で進行していると仮定できる。そのため、異なるコミュニティを探索中のワーカーから送られてくる学習節は現在の探索の対象ではない変数を含んでいると考えられ、将来的に削除される可能性が高い。

この問題を解消するために、本研究では学習節内に含まれるコミュニティを元に、それらのコミュニティを探索中のワーカーのみに共有を行う手法を提案する。その疑似コードを Algorithm 1 に示す。最初の関数 CONSTRUCTV2C は、コミュニティ検出を行い、変数の番号からコミュニティの番号へのマッピング (Variable to Community, $v2c$) を構築する。この関数は探索開始前に一度呼ばれる。構築する VIG に関して、全ての節を使用するとグラフの大きさが巨大になるため、使用する節の長さは 5 以下のものとする。この関数はマスターワーカー (ID が 0 のワーカー) からのみ呼びされる。二番目の関数 UPDATEDESIRECOMMS は各ワーカーから呼ばれ、各コミュニティに属する変数の VSIDS スコアの平均値を計算し、その上位のコミュニティを共有が望ましいコミュニティとして登録する。登録するコミュニティの数は、検出されたコミュニティの個数とワーカーの総数に依存する。この登録されたコミュニティを元に、学習節の共有を行う。例えば学習節 $(x \vee y \vee z)$ があり、x がコミュニティ 0、y が 2、z が 6 にそれぞれ属するのであれば、0、2、6 を登録しているワーカーにのみこの学習節は共有される。ただし、LBD が 3 以下のものは無条件で全ワーカーで共有させるものとする。VSIDS のスコアは decision をする際の優先度を示しているため、高いものほど現在の探索の対象になっている変数であると考えられる。この関数は定期的に実行され、登録するコミュニティを更新していく。本論文の実験では各ワーカーが 3 回リスタートを行う度に更新をする。

5. 実験

本提案手法を代表的な並列 SAT ソルバの一つである Penelope [3] に実装し比較実験を行った。Penelope は ManySAT をベースにしたポートフォリオ型のソルバであり、LBD 等の近年の技術を揃えている。ベンチマークは SAT Competition 2014 *1 のアプリケーション部門の 300 問を用い、一問あたりの実行処理時間の上限は 5000 秒とした。実験に使用した環境は、CPU が Intel Xeon X5650 2.67GHz(6 コア) を二つ、RAM が 96GB、OS が Ubuntu14.04、コンパイラが GCC 4.8.4 のマシンである。ワーカー (スレッド) の数は 8 とした。コミュニティ検出アルゴリズムとして、高速かつ高モジュラリティを実現する louvain 法 [4] を用いた。

*1 http://satcompetition.org/2014/files/sc14-app.tar

Algorithm 1 提案手法の疑似コード

```
1:  procedure ConstructV2C
2:      coms = communityDetection()
3:      v2c = an array with |X| elements
4:      for i = 0 to |coms| − 1 do
5:          for all v ∈ coms[i] do
6:              v2c[v] = i
7:  procedure UpdateDesireComms
8:      c = the number of detected communities
9:      t = the number of running workers
10:     d = max(c/t, 1)
11:     ave_score = average VSIDS score for each communities
12:     select the top-d highest scored communities
```

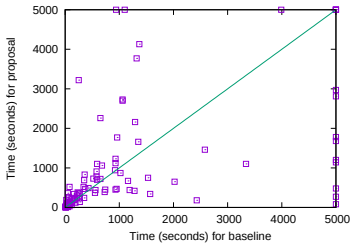

図 1: 充足可能な問題に対する処理時間

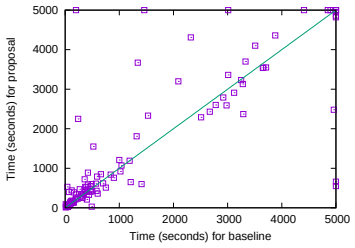

図 2: 充足不可能な問題に対する処理時間

元の Penelope の処理時間 (baseline) を x 軸、提案手法を組み込んだ Penelope の処理時間を y 軸にプロットした散布図を、充足可能な問題 (SAT) に対する結果を図 1、充足不可能な問題 (UNSAT) に対する結果を図 2 に示す。5000 秒以内に解けた問題数は、baseline は SAT が 112 問で UNSAT が 116 問、提案手法は SAT が 118 問で UNSAT が 115 問であった。充足可能な問題に対して一定の効果が見られたが、処理時間が増大した問題も一定数存在している。そして充足不可能な問題に対しては若干の性能下降が確認できる。なお、louvain 法によるコミュニティ検出にかかる処理時間はたかだか 10 秒程度であり、その時間も処理時間に含まれている。

一方で、本研究の目的であった、一度も使用されずに削除される共有学習節の削減には効果が見られた。各ワーカーにおいて他のワーカーから共有された学習節のなかで、一度も使用さ

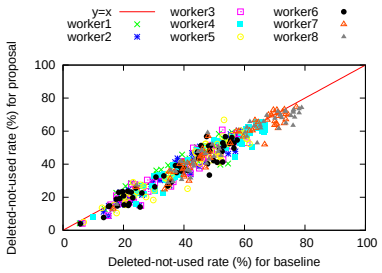

図 3: 充足可能な問題に対する未使用に終わり削除された節の割合

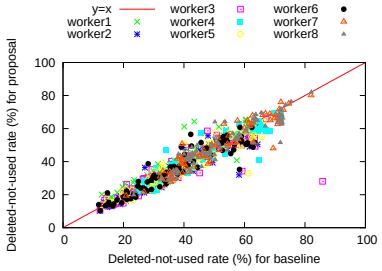

図 4: 充足不可能な問題に対する未使用に終わり削除された節の割合

れずに削除された節の割合を算出し、元の Penelope (baseline) の割合を x 軸に、提案手法の割合を y 軸にプロットした散布図を、充足可能な問題に対する結果を図 3、充足不可能な問題に対する結果を図 4 に示す。なお、制限時間以内に終了しなかった問題と、処理時間が 100 秒未満の解答が容易な問題はこれらの図から除外している。充足可能・不可能どちらの問題に対しても全体的に割合は減少しており、全ワーカーの累計 (散布図中のサンプル点の総数) で、充足可能な問題では 301/448(67%)、充足不可能な問題では 442/656(67%) で割合の低下が確認できた。

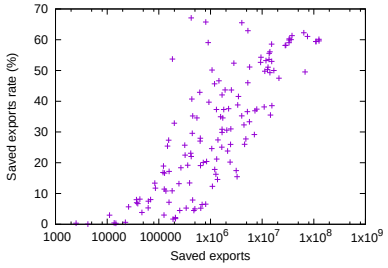

図 5: 共有が行われなかった学習節の総数とその割合

最後に、提案手法によって共有が省かれた学習節の総数と、それによる通信の削減割合 (共有が省かれた学習節の総数を a、共有された学習節の総数を b とすると $\frac{a}{a+b}$) をそれぞれ図 5 の x 軸と y 軸に示す。共有が省かれる個数が多いほど削減割合も増えており、50%を超えている問題も存在する。つまり、学習節の共有をある程度省いてもソルバの性能 (処理時間) を保つことが本提案手法によって実現できている。この事実は、ワーカー間通信のオーバーヘッドが大きい環境 (例えば分散並列環境) では有効的である。また、今回は 8 ワーカーでの実験だが、より多くのワーカーが存在する場合では共有される学習節が増大するため、そのような環境でも提案手法は有効であると考えられる。

6. まとめ

本論文では、SAT 問題のコミュニティ構造を利用したポートフォリオ型並列 SAT ソルバの学習節共有方法の提案を行った。各ワーカーにおいて、他のワーカーから共有された学習節が一度も探索に影響を与えること無く削除されてしまうことが多々あり、提案手法がそのような無駄な学習節の削減に効果的であることを実験で確かめた。しかし、そのような学習節の共有を省くことで、本来共有が望ましかったものの共有まで省いてしまう可能性があるため、性能面でのトレードオフも確認できた。一方で、学習節の共有量をかなり落としても性能は維持されているため、ワーカー間の通信コストが高い分散並列環境や、よりワーカーの多い環境での有効性が期待できる。また、今回の実験でコミュニティ検出は探索の開始前にしか行わなかったが、コミュニティ構造は学習節の追加によって変化することが知られている [1] ため、探索の途中に再度コミュニティ検出を行うことで性能の向上も期待できる。

参考文献

[1] C. Ansótegui, J. Giráldez-Cru, and J. Levy. The community structure of SAT formulas. In *SAT*, pages 410–423, 2012.

[2] C. Ansótegui, J. Giráldez-Cru, J. Levy, and L. Simon. Using community structure to detect relevant learnt clauses. In *SAT*, pages 238–254, 2015.

[3] G. Audemard, B. Hoessen, S. Jabbour, J.-M. Lagniez, and C. Piette. Revisiting clause exchange in parallel sat solving. In *SAT*, pages 200–213, 2012.

[4] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, 2008(10):10008, 2008.

[5] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[6] C. P. Gomes, B. Selman, H. Kautz, et al. Boosting combinatorial search through randomization. *AAAI*, 98:431–437, 1998.

[7] L. Guo, Y. Hamadi, S. Jabbour, and L. Sais. Diversification and intensification in parallel SAT solving. In *CP*, pages 252–265, 2010.

[8] Y. Hamadi, S. Jabbour, and L. Sais. ManySAT: a parallel SAT solver. *JSAT*, 6(4):245–262, 2009.

[9] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *DAC*, pages 530–535, 2001.

[10] Z. Newsham, V. Ganesh, S. Fischmeister, G. Audemard, and L. Simon. Impact of community structure on sat solver performance. In *SAT*, pages 252–268, 2014.

[11] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

[12] J. P. M. Silva and K. A. Sakallah. Grasp—a new search algorithm for satisfiability. In *ICCAD*, pages 220–227, 1996.

[13] L. Simon and G. Audemard. Predicting learnt clauses quality in modern sat solver. In *IJCAI*, 2009.

[14] T. Sonobe, S. Kondoh, and M. Inaba. Community branching for parallel portfolio sat solvers. In *SAT*, pages 188–196, 2014.