1M2-OS-02b-3

GPGPU による MaxSAT オラクルを用いた SAT ソルバの試作

Implementation of a SAT Solver guided by MaxSAT Oracle using GPGPU

山口 順也 *1　ソフィ トゥレ *2　井上 克巳 *1*2
Junya Yamaguchi　Sophie Tourret　Katsumi Inoue

*1東京工業大学 Tokyo Institute of Technology
*2国立情報学研究所 National Institute of Informatics

A Boolean satisfiability testing (SAT) problem is a problem of determining whether there is a variable assignment that satisfies a given propositional formula. In recent years, General-Purpose computing on GPUs (GPGPU) is attracting more attention as an accelerator for high speed computation. However, as far as we know, there are no examples where GPGPU is used for SAT algorithm to improve performance greatly. In this paper, we propose a Weighted Partial MaxSAT (WPMS) solver using GPU computation, named "MaxSAT oracle", and show that MaxSAT oracle can solves faster than the state-of-the-art WPMS solver for specific problems. Furthermore, we propose a SAT solver guided by MaxSAT oracle, named "Batsat", and evaluate its performance.

1. はじめに

命題論理における充足可能性判定 (Boolean/propositional satisfiability testing; SAT) 問題とは，与えられた命題論理式を充足させるような変数割り当てが存在するかを判定する問題である．SAT 問題は，初めて NP 完全であることが証明された NP 問題として知られており，計算機科学にとって中心的な問題であると言える．また SAT 問題には AI プランニング，スケジューリングや回路検証など，幅広い応用問題が知られており，少しでも効率良く解く事の出来る SAT ソルバの開発は極めて重要な課題である．

近年， GPU を使って数値計算をはじめとした汎用的な計算を行う GPGPU（General-Purpose computing on GPUs）技術が注目されている．GPGPU を用いた既存の SAT アルゴリズムの並列化に関する研究はいくつか知られている [Gulati 10][Meyer 10][藤井 11][Yoo 14] が，その成果は限定的である．そこで本研究では，GPGPU を用いた並列処理の結果に基づいて解の探索を行う完全 SAT ソルバを提案することを目的とし，提案した SAT ソルバの性能を評価する．

本稿の構成は以下の通りである．まず，第 2 章では SAT 問題の定義について述べ，SAT に対するアルゴリズムを説明する．第 3 章では CUDA を用いた重み付き部分 MaxSAT（Weighted Partial MaxSAT; WPMS）ソルバである MaxSAT オラクルを提案し，その評価を行う．第 4 章では MaxSAT オラクルを用いた完全 SAT ソルバである Batsat を提案する．そして，第 5 章で Batsat 性能評価実験を行い，第 6 章で結論を述べる．

2. 背景

2.1 SAT 問題

充足可能性判定 (SAT) とは，与えられた（命題）論理式が，充足可能か充足不能かを判定する問題であり， SAT の各インスタンスを SAT 問題と呼ぶ [井上 10]．SAT 問題に入力される論理式は CNF 式で与えられることが多く，ここで CNF 式とは節を連言（$\wedge$）で結合したものであり，節はリテラルを選言（$\vee$）で結合したものである．リテラルは変数 v または変数の否定 $\neg v$ で，変数は 0（偽）または 1（真）の値を取る．また，変数の有限集合 V に対して，写像 $I : V \to \{0, 1\}$ を割り当てと呼び，I によってすべての変数が $\{0, 1\}$ に割り当てられる時，I を完全割り当てと呼び，そうでない時は部分割り当てと呼ぶ．割り当て I によって 論理式 ψ が 1 に評価される時 I は ψ を充足すると言い，論理式を充足するような割り当てをモデルと言う．ある論理式にモデルが存在する時は充足可能（`SAT`）であり，そうでない時は充足不能（`UNSAT`）である．

連絡先: 山口 順也，東京工業大学 工学部 情報工学科，yamaguchi.j.ab@m.titech.ac.jp

2.2 SAT アルゴリズム

SAT アルゴリズムには `SAT` と `UNSAT` の両方を証明出来る完全アルゴリズムと，`SAT` しか証明できない確率的アルゴリズムが存在する．1962 年に Davis ら が提案した完全 SAT アルゴリズムである DPLL[Davis 62] は現在の殆どの完全 SAT アルゴリズムの土台となっている．DPLL は大きく次の 3 つのパートで構成される．

Decision 変数選択ヒューリスティックに則って行われる恣意的な変数の割り当てを決定による割り当てと呼ぶ．決定はモデルの仮定であり，決定と後述する単位伝搬を繰り返して全ての節を充足させると `SAT` と判定する．

Unit Propagation まだ充足していない節中に未割り当てなリテラルが 1 つだけある状態の節を単位節と呼び，単位節中のこの様なリテラルは真に割り当てられることが決まる．この割り当てを含意による割り当てと呼び，単位節がなくなるまで含意による割り当てを連鎖的に導き出す処理を単位伝搬と呼ぶ．単位伝搬ではある変数に対して同時に真と偽を割り当てようとすることがあり，この様な状態が生じる事を矛盾と呼ぶ．

Backtracking 単位伝搬で矛盾が生じた時に，最後の試していない真偽値がある決定の真偽値を変更する処理がバックトラックである．バックトラックによって決定がない所まで遡ってしまった場合 `UNSAT` と判定する．

さらに 1996,69 年に Silva ら [Silva 96][Marques-Silva 99], 1997 年に Bayardo ら [Bayardo 97] は DPLL をベースにし，矛盾が起きた際に新たな制約節を学習するアルゴリズムを提唱した．この節学習アルゴリズム は CDCL（Conflict Driven Clause Learning）と呼ばれ，CDCL アルゴリズムを利用した

SAT ソルバのことを CDCL ソルバと呼ぶ．CDCL では矛盾を引き起こした含意による割り当てを連鎖的に辿っていくことで，その矛盾が生じた原因となった割り当てを発見する．このような原因となった割り当てを再び起こさない様に新たな制約節を追加することで DPLL に比べて大幅な探索空間の削減を実現した．

2.3 CUDA

近年，GPU の持つ高い演算能力をグラフィクス処理以外の汎用計算に利用する技術である GPGPU（General-Purpose computing on GPUs）が注目を集めている．2006 年に NVIDIA によって公開された GPGPU のための統合開発環境 CUDA（Compute Unified Device Architecture）は，ライブラリ，コンパイラ，デバッガなどから構成されていて，C/C++ 言語をベースに拡張されたプログラミング言語を用いて記述する．

3. MaxSAT オラクル

3.1 重み付き部分 MaxSAT 問題

SAT 問題の拡張として，重み付き部分 MaxSAT（WPMS）問題が知られている．WPMS 問題とは，与えられた重み付き CNF 式に対して，次の 2 つの条件を満たす完全割り当て I を決定する問題である．

1) すべてのハード節が I によって充足される
2) 充足されないソフト節の重みの総和が I の時に最小

ここで，重み付き CNF 式 (weighted CNF formula; WCNF 式) とは重み付き節の集合であり，重み付き節とは，節 C と非負整数 w のペア (C, w) であり，w を 節 C の重みと呼ぶ．また，$+\infty$ の重みを持つ重み付き節はハード節，それ以外の重み付き節はソフト節と呼ばれる．全てのハード節を充足させる割り当てが存在しない時，WCNF 式は充足不能となる．

3.2 MaxSAT オラクル

本研究では，CUDA によって実装された WPMS ソルバである MaxSAT オラクルを提案する．MaxSAT オラクルのアルゴリズムは非常にシンプルで，力まかせ探索を並列化することで探索の高速化を図っている．WCNF 式が入力されると，ある割り当てに対して一つのスレッドを対応させるように，可能な割り当ての数だけスレッドを作成する．各スレッドでは，そのスレッドに対応する割り当てによって充足しない節の重みの総和を計算する．そして，全てのスレッドの計算が終了するまで同期した後，重みの総和が最小となる割り当てを決定する．

3.3 性能評価

前節で提案した MaxSAT オラクルの性能を，実験によって評価した．実験環境は以下の通りである．

表 1: 実験環境

OS	Red Hat 5.3.1-6
CPU	Intel Core i7-4770 CPU @ 3.40GHz
RAM	16.00 GB
GPU	GeForce GTX 1080, 8GB VRAM

MaxSAT オラクルは，各スレッドで 8 バイトの結果保存用領域を確保するため，n 変数の WCNF 式に対しては，$2^n \times 8$ バイトの領域が必要となる．本実験環境では 最大で 28 個の変数をもつ WCNF 式について WPMS 問題を解くことが出来たので，性能評価で用いる問題には 変数の数が 28 個以下のランダム WPMax k-SAT 問題を対象にした．ここでランダム WPMax k-SAT 問題とは，全ての節のサイズが k で，節中のリテラルはランダムに決定された WCNF 式による WPMS 問題を指す．具体的には，ソフト節の重みを 1〜100 の値をランダムに決定し，k，変数の数，節の数，ハード節の数に関して以下のパターンのものを作成した．

k	3, 5, 7
変数の数	16, 20, 24, 28
節の数	500, 1000, 2000, 4000, 8000
ハード節の数	変数の数の 0.5, 1, 1.5, 2 倍

上記のパターンの各組み合わせに対して 5 問ずつ，計 $(3 \times 4 \times 5 \times 4) \times 5 = 1200$ 問のランダム WPMax k-SAT 問題を対象に実験を行った．

性能の比較対象として，MaxSAT ソルバの性能を競う競技会である MaxSAT Evaluations 2015[*1] で全 9 部門中 3 部門で 1 位に輝き， Weighted Partial MaxSAT 部門でも 3 位に入賞している `ahmaxsat-ls-1.68`[*2] を比較用の WPMS ソルバとした．

3.4 結果

対象のランダム WPMax k-SAT 問題に対する実行時間 (秒) を，問題の変数の数ごとに分類して平均した結果を図 1 に示す．MaxSAT オラクルは，どの変数の数でも最新の WPMS ソルバである `ahmaxsat` より平均的に短い実行時間で問題を解くことが出来た．16 変数の問題に対してソルバの差は殆どないが，これは MaxSAT オラクルに GPGPU のための初期化処理のオーバーヘッドがある為だと考えられる．また 28 変数の問題でもソルバの差が縮まっている．GeForce GTX 1080 の CUDA コア数は 2560 個なので，同時に実行できる最大グリッド数は 2560 個でありその時のスレッド数は $2560 \times 1024 \simeq 2^{21}$ である．よって 2^{22} 個以上のスレッドを使用するような問題，すなわち 22 個以上の変数を持つ問題に対しては待機スレッドが発生し，求解の効率が落ちていると推測される．

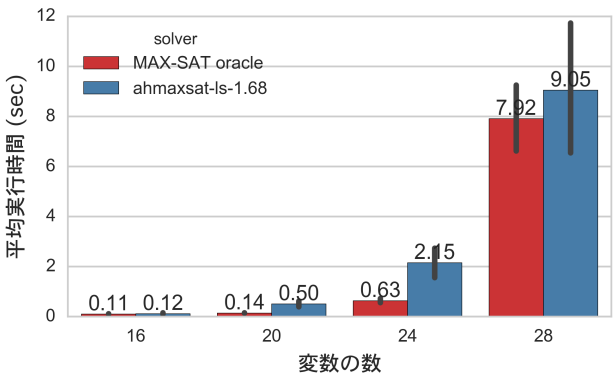

図 1: 平均求解時間の比較

以上の考察を踏まえ，MaxSAT オラクルの利点を最大に活かす事が出来る 20 変数の問題に対する平均実行時間を，節の数ごとに分類した結果を図 2 に示す．`ahmaxsat` の実行時間は

*1 http://www.maxsat.udl.cat/
*2 http://www.lsis.org/abramea/ressources.html

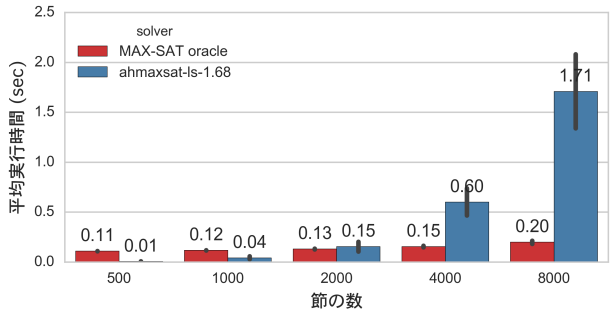

図 2: 20 変数の問題に対する平均求解時間の比較

節数に対して指数的に増加している一方で，MaxSAT オラクルは定数倍の増加に抑えられている．8000 節の問題では平均して約 8 倍速く求解出来るなど，変数が少なく節の数が多い WPMS 問題で MaxSAT オラクルは非常に有効であると言える．

4. 提案 SAT ソルバ

4.1 概要

本研究では `Minisat 2.2`[*3] をベースに，MaxSAT オラクルを組み込んだ完全 SAT ソルバである Batsat を提案する．`Minisat` は MIT ライセンスでソースコードが公開されている CDCL ソルバで，C++ 言語によって非常に簡潔に記述された拡張がし易い CDCL ソルバとして知られている．通常の CDCL ソルバはある未割り当ての変数に対して 0 か 1 を割り当てる事で探索を進めていくが，Batsat はバッチを用いて複数の変数をまとまて割り当てることを繰り返す事で解の探索を行う．ここでバッチは空集合でない変数の有限集合であり，探索中の未割り当てな変数によって構成される．バッチに含まれる変数の数をバッチのサイズと呼び，バッチの変数への割り当てを単なる部分割り当てと区別してバッチ割り当てと呼ぶ．

4.2 バッチ割り当て順番問題

ある探索途中の時点において，現在の部分割り当てによって既に充足している節と，偽に割り当てられているリテラルをすべて取り除く．未割り当て変数の中からバッチ B を選択し，バッチに選ばれなかった未割り当て変数の集合を U とすると，現在の部分割り当てによってまだ充足されていない節である未解決節には次の 3 パターンが存在する．

1. B のリテラルを 1 つ以上含み，U のリテラルを含まない節．これは B のバッチ割り当てによって，必ず充足させる必要がある節である．このような未解決節をバッチハード節と呼ぶ．
2. B のリテラルを含まず，U のリテラルを 1 つ以上含む節．このような未解決節は，B のバッチ割り当てでは絶対に充足させることができない節である．
3. B のリテラル と U のリテラルをそれぞれ 1 つ以上含む節．このような未解決節をバッチソフト節と呼ぶ．

以上の考察より，WPMS 問題の拡張であるバッチ割り当て順位問題を考えることが出来る．バッチ割り当て順位問題は，ある CNF 式 ψ と現在の部分割り当て I，及び ψ, I に対するあるバッチ B を入力とし，次の条件を満たすバッチ割り当て $I_B : B \to \{0,1\}$ を順位の高い順番にすべて出力する問題である．

1) すべてのバッチハード節が I_B によって充足される．
2) I_B によって充足されないバッチソフト節の重みの総和が小さいほど順位が高い．

ここでバッチソフト節の重みとは，バッチソフト節の重要度を表す非負整数である．充足されないバッチソフト節の重みの総和を小さくするバッチ割り当ては，重要なバッチソフト節をなるべく多く充足するようなバッチ割り当てである．このようなバッチ割り当てを優先的に試す事で，貪欲的に SAT 解へ近づくことがバッチ割り当て順位問題を作成する意図である．本研究では，全てのバッチソフト節を均等に扱うために全てのバッチソフト節の重みに 1 を設定したが，バッチソフト節のサイズによって重みの付け方を変える等の様々なヒューリスティックを考慮する余地がある．

4.3 アルゴリズム

Batsat の探索アルゴリズムの疑似コードを Algorithm 1 に示す．バッチレベル $blevel$ はそれまでに何個のバッチが選ばれたかを表し，バッチ割り当て A は現在のバッチ割り当てを

Algorithm 1 Batsat の探索アルゴリズム

```
1:  現在のバッチレベル blevel ← 0
2:  現在のバッチ割り当て A ← ∅
3:  バッチ割り当て記録用データベース DB ← ∅
4:  while true do
5:      while true do
6:          単位伝搬を行う
7:          if 矛盾が発生または A = ∅ then break
8:          else
9:              あるリテラル l ∈ A を決定し, A ← A/l
10:         end if
11:     end while
12:     if 矛盾が発生 then
13:         if blevel = 0 then return UNSAT
14:         end if
15:         矛盾を解析して節を学習
16:         blevel ← 学習節によるバックジャンプ
17:         A ← DB[blevel] の次の順位のバッチ割り当て
18:     else
19:         if 未割り当ての変数がない then return SAT
20:         end if
21:         C ← バッチを選び, MaxSAT オラクルで解いたバッチ割り当て順位問題の解
22:         if C = ∅ then
23:             blevel ← バックトラック
24:             if blevel = 0 then return UNSAT
25:             end if
26:             A ← DB[blevel] の次の順位のバッチ割り当て
27:         else
28:             A ← C の最初の順位のバッチ割り当て
29:             DB[blevel] ← C
30:             blevel ← blevel + 1
31:         end if
32:     end if
33: end while
```

*3 http://minisat.se/

表す．Batsat は解探索の分岐で変数の代わりにバッチを用いて分岐を行っており，MaxSAT オラクルによって解いたバッチ割り当て順位問題の，順位が高いバッチ割り当てほど探索に有効だと仮定して順番に試していく．また，変数の決定と単位伝搬を交互に繰り返しながら，一つの変数ずつバッチ割り当てを行うことで，CDCL の節学習アルゴリズムを Batsat でも使用出来る様にしている．

5. 評価

本節では，提案した SAT ソルバの性能を実験によって評価する．実験に用いる問題は SAT ソルバの世界的な競技会である SAT Competition[*4] で出題された問題を公開している SATLIB[*5] を利用し，SATLIB から Application 部門の `SAT` 問題 5 問，`UNSAT` 問題 2 問を対象に，タイムアウトを 300 秒に設定して実験を行った．実験環境は表 1 と同じ構成である．また，比較 SAT ソルバとして，`Minisat 2.2` を用いた．

5.1 結果

`Minisat` 及び Batsat の各問題に対する実行時間を表 2 に示す．`Minisat` は非常に高速に問題を解いているのに対し，

表 2: Minisat 2.2 と Batsat の実行時間（秒）の比較

問題	変数	節数	S	`Minisat`	Batsat
`bw_large.a`	59	4675	`S`	0.0018	0.04274
`bw_large.c`	3016	50457	`S`	0.0907	15.2962
`bw_large.d`	6325	131973	`S`	0.0862	> 300
`bmc-ibm-1`	9685	55180	`S`	0.0238	> 300
`bmc-ibm-4`	28161	134213	`S`	0.0714	219.926
`bf2670-001`	1393	3306	`U`	0.0032	0.06897
`bf1355-638`	2177	5951	`U`	0.0023	0.38030

Batsat は 300 秒以上かかった問題もあり，求解速度は十分に速いとは言えない結果となった．

5.2 考察

Batsat の求解が遅い原因として，バッチ割り当て順位問題の作成方法に問題があると考えられる．選択したバッチによるハード節，ソフト節の判別など，バッチ割り当て順位問題の作成には非常にコストがかかるので，コストを上回るような利益をもたらすような問題を作る必要がある．ソフト節の重みをどのように設定するかは，Batsat の探索の効率を左右する大きな要因と考えられる．

また，Batsat は求解の際に変数をバッチに分割して探索を行う．ある節中の 2 つの変数が異なるバッチに存在していれば変数間にはその節による制約関係があるにも関わらず，バッチ割り当て問題ではこのような制約は考慮されない．したがって，このようなバッチ間の制約関係の数が少なくなる様なバッチの選び方ができれば，無駄な探索を減らすことに繋がると考えられる．

6. まとめ

本稿では，GPGPU を用いた重み付き部分 MaxSAT ソルバである MaxSAT オラクルを提案し，MaxSAT オラクルを SAT ソルバの解探索の分岐に用いる完全 SAT ソルバを提案した．提案した SAT ソルバに関しては我々が期待する性能を示すことは出来なかったが，節数が非常に多い WPMS 問題に対して従来の WPMS ソルバより約 8 倍の速度で求解出来るなど，MaxSAT オラクルに関してはその有用性を示すことが出来た．

今後の課題としては，単位伝搬や CDCL の節学習といった既存のアルゴリズムを　MaxSAT オラクルに適した形に修正することが挙げられる．また，バッチ割り当て順位問題を解くことで貪欲的に解に近づくのではなく，なるべく多くのバッチソフト節を満たさないように決定を行い，矛盾を生じさせることで節学習を促すような利用の方法も考えられる．さらに，本研究では MaxSAT オラクルを完全 SAT ソルバに組み込んだが，確率的 SAT ソルバや MaxSAT ソルバへの適応も検証していきたい．

*4 http://www.satcompetition.org/

*5 http://www.cs.ubc.ca/ hoos/SATLIB/benchm.html

参考文献

[Bayardo 97] Bayardo, R. J., Jr. 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 and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, pp. 203–208, AAAI Press (1997)

[Davis 62] Davis, M., Logemann, G., and Loveland, D.: A Machine Program for Theorem-proving, *Commun. ACM*, Vol. 5, No. 7, pp. 394–397 (1962)

[Gulati 10] Gulati, K. and Khatri, S. P.: Boolean Satisfiability on a Graphics Processor, in *Proceedings of the 20th Symposium on Great Lakes Symposium on VLSI*, GLSVLSI '10, pp. 123–126, New York, NY, USA (2010), ACM

[Marques-Silva 99] Marques-Silva, J. P. and Sakallah, K. A.: GRASP: a search algorithm for propositional satisfiability, *IEEE Transactions on Computers*, Vol. 48, No. 5, pp. 506–521 (1999)

[Meyer 10] Meyer, Q., Schnfeld, F., Stamminger, M., and Wanka, R.: 3-SAT on CUDA: Towards a massively parallel SAT solver, in *Proc. Int. Conf. High Performance Computing Simulation*, pp. 306–313 (2010)

[Silva 96] Silva, J. P. M. and Sakallah, K. A.: GRASP-A new search algorithm for satisfiability, in *Proc. Int. Conf. Computer Aided Design*, pp. 220–227 (1996)

[Yoo 14] Yoo, T., Kim, S., Yeom, Y., and Kang, J.: A Study of the Parallelization of Hybrid SAT Solver using CUDA, *Advanced Scienceand Technology Letters*, Vol. 48, pp. 19–24 (2014)

[井上 10] 井上 克巳, 田村 直之：SAT ソルバーの基礎 (＜特集＞最近の SAT 技術の発展), 人工知能学会誌, Vol. 25, No. 1, pp. 57–67 (2010)

[藤井 11] 藤井 宏憲, 藤本 典幸：SAT アルゴリズムにおける BCP 処理の GPU を用いた並列化, 研究報告計算機アーキテクチャ（ARC）, Vol. 2011, No. 27, pp. 1–8 (2011)