Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 2567

Paralleler SAT-Solver mit Konfliktklauselbehandlung

Jil Tietjen[1]

Abstract: In diesem Paper wird ein paralleler SAT-Solver beschrieben, der durch Multithreading parallelisiert wird. Dabei ist das Besondere, dass verschiedene Solverparameter untersucht werden und ein Konfliktklauselhandling implementiert wird, welches auf einer Klassifikation der Konfliktklauseln basiert. Als Benchmark wird der beliebig skalierbare Colored Graph von Knuth[Kn15] verwendet, womit möglichst unterschiedliche Eingabegrößen untersucht werden können. Der parallele SAT-Solver kann in einigen Fällen den sequentiellen SAT-Solver schlagen.

Keywords: parallel search, extended clause learning, CDCL SAT solver, Naive Bayes classifier

1 Einleitung

Das aussagenlogische Erfüllbarkeitsproblem (Satisfiability Problem) kommt in der Informatik in der Theorie und in der Praxis oft vor. SAT-Solver dienen dazu kombinatorischen Probleme effizient zu lösen, Verifikation vorzunehmen (nach einem geeigneten Mapping) oder praktische Probleme, wie zum Beispiel das Lösen von Sudokus oder die Stundenplanung. Es soll entschieden werden, ob eine aussagenlogische Formel erfüllbar ist oder nicht. Hierfür wird eine erfüllende Belegung für die Variablen (mit true oder false) gesucht.
Da SAT-Solver in den letzten Jahren an Relevanz zugenommen haben, wird bei der SAT-Competition[2] regelmäßig versucht, bestehende Solver zu übertreffen. Dabei ist eine mögliche Herangehensweise, SAT-Solver zu parallelisieren.
Ein gängiges Vorgehen ist der kompetitive Ansatz, bei dem mehrere Instanzen eines sequentiellen SAT-Solvers mit verschiedenen Parametern ausgeführt werden. Dabei arbeitet jede Instanz auf der gesamten Formel [SH11].
In diesem Paper wird ein paralleler SAT-Solver, beschrieben, der auf einem bestehenden sequentiellen SAT-Solver basiert und durch ein Konfliktklauselhandling mit Hilfe von Algorithmen aus dem Bereich des Machine-Learnings erweitert wird, die in diesem Zusammenhang noch nicht eingesetzt wurden. Es soll überprüft werden, ob der parallele SAT-Solver in umfangreichen Benchmarks den sequentiellen SAT-Solver schlagen kann.

[1] Universität Bremen, AG Rechnerarchitektur, Bibliothekstraße 1, 28359 Bremen, jiltietj@informatik.uni-bremen.de
[2] http://baldur.iti.kit.edu/sat-competition-2016/index.php?cat=results

doi:10.18420/in2017_259

2 Grundlagen

2.1 Satisfiability Problem

Das Erfüllbarkeitsproblem (kurz: SAT) ist ein Problem der Aussagenlogik, welches vermutlich nicht in polynomieller Zeit gelöst werden kann, da es NP-vollständig ist. Als Eingabe gibt es einen aussagenlogischen Ausdruck **A** in konjunktiver Normalform, für den herausgefunden werden soll, ob er erfüllbar ist oder nicht. Es wird eine erfüllende Belegung der Variablen mit true oder false gesucht. Es muss eine Belegung geben, so dass jede Klausel erfüllt ist [Co71].
Der SAT-Solver kann feststellen, ob eine Formel erfüllbar ist oder nicht. Zur Lösung des Erfüllbarkeitsproblems wird häufig der Davis-Putnam-Logemann-Loveland-Algorithmus (kurz: DPLL) verwendet. Das heißt, dass eine erfüllende Variablenbelegung schrittweise konstruiert wird. Sobald ein Konflikt auftritt werden mittels Backtracking Teile der Belegung rückgängig gemacht [AB09].
Viele SAT-Solver verwenden das Klauselllernen (Conflict-Driven Clause Learning) [PB03]. Kann eine Formel aufgrund eines Konfliktes in der Belegung der Variablen nicht mehr erfüllt werden, wird eine neue Klausel hinzugefügt, die diese Belegung ausschließt. Das Hinzufügen der Klauseln verändert die Erfüllbarkeit der Formel nicht.

2.2 Naive Bayes

Naive Bayes ist ein Klassifikator, der Objekte zu einer Klasse zuordnet, zu der das Objekt mit einer gewissen Wahrscheinlichkeit gehört [Zh04]. Die Klassifikatoren vereinfachen das Problem durch die Voraussetzung der Unabhängigkeitsannahme. Es wird angenommen, dass die Merkmale unabhängig voneinander sind, gegeben der Klasse. Die Annahme ist stark vereinfachend, aber verfälscht das Ergebnis in der Praxis kaum. Bevor das Naive Bayes benutzt wird, muss der Klassifikator trainiert werden. Dazu werden Trainingsdaten eingesetzt, mit denen gelernt wird, welche Abhängigkeiten die Merkmale von der Klasse haben.

3 Paralleler SAT-Solver mit Konfliktklauselhandling

Die erste Herangehensweise ist, dass ein bestehender sequentieller SAT-Solver parallel auf verschiedenen Threads versucht die Formel zu lösen. Dabei werden unterschiedliche Suchparameter für die einzelnen SAT-Solver verwendet. Sobald ein SAT-Solver eine Lösung findet, wird der Vorgang beendet. Somit wird der kompetitive Ansatz in diesem Paper verfolgt [SH11].
Beim Verwenden der unterschiedlichen Suchparameter kann festgestellt werden, dass diese zu sehr unterschiedlichen Laufzeiten führen und teilweise den sequentiellen SAT-Solver

schlagen. Beispielsweise kann eine andere Restart-Strategie auf bestimmten Formeln dazu führen, dass nur noch ein Fünftel der Zeit benötigt wird. Das liegt daran, dass der sequentielle SAT-Solver eine Default-Suchstrategie verwendet, die nicht für jedes Problem optimal ist. Somit kann eine der Suchstrategien beim parallelen SAT-Solver besser auf das Problem abgestimmt sein und so schneller zu einer Lösung kommen. Aus diesem Grund werden aus dem Repertoire der bestehenden SAT4J-Suchstrategien möglichst unterschiedliche Suchparameter gewählt, um möglichst viele Probleme abzudecken. Diese Suchparameter unterscheiden sich durch Einstellungen, wie unterschiedliche Restart-Strategien, Datenstrukturen und Vereinfachungsstrategien. Es gibt jeweils so viele Threads, wie es auch Prozessoren in dem jeweiligen Rechner gibt. Sind zum Beispiel acht unterschiedliche Threads möglich, können acht unterschiedliche Suchstrategien verwendet werden. Dies erhöht die Wahrscheinlichkeit, dass eine bessere Laufzeit erzielt wird als die Default-Variante des sequentiellen SAT-Solvers.

Dieser kompetitve Ansatz wird zusätzlich durch ein Konfliktklauselhandling erweitert. Hat einer der Solver eine Konfliktklausel gefunden, könnte diese auch für die anderen Solver nützlich sein. Daher wird die Konfliktklausel an die anderen Solver übergeben. Hierbei dürfen allerdings nicht zu viele Klauseln geteilt werden, da sonst der Aufwand für das Übergeben und Verwalten der Klauseln zu hoch wird. Daher ist es nötig, die nützlichen Konfliktklauseln herauszufiltern.

Das Besondere beim Konfliktklauselhandling in dieser Arbeit ist, dass ein Naive-Bayes-Klassifikator verwendet wird, mit dem die Konfliktklauseln klassifiziert werden. Durch die Klassifikation können die Konfliktklauseln in Klassen geeigneter und ungeeigneter Konfliktklauseln eingeteilt werden. Es werden nur die geeigneten Konfliktklauseln an die verschiedenen Solver übergeben, so dass diese beim Lösen des Problems auf alle neu gefundenen Konfliktklauseln, die als geeignet klassifiziert wurden, zugreifen können. Ziel ist es, dass dadurch die Performance des parallelen SAT-Solvers gesteigert wird.

Der Klassifikator wird durch Trainingsdaten[3] mit bekannter Klassifikation anhand von fünf Merkmalen trainiert.

Das erste Merkmal ist, ob die Klausel eine Horn-Klausel ist, oder nicht. Eine Horn-Klausel ist eine Klausel mit höchstens einem positiven Literal. Ein Merkmal von Horn-Formeln (Formeln, die nur aus Horn-Klauseln bestehen) ist, dass sie eine bessere Berechnungszeit haben als allgemeine SAT-Instanzen. Es ist allerdings nicht klar, ob die Horn-Klauseln in SAT-Instanzen einen essentiellen Vorteil bringen oder nicht. Zwei weitere Merkmale sind, ob es positive bzw. negative Literale in der Klausel gibt. Da das Vorkommen einer Variable oder eines Literals entscheidend für die Belegung einer Variablen sein kann, ist dies ebenfalls ein Merkmal. Das vierte Merkmal ist die Größe der einzelnen Klauseln. Generell sind kleinere Konfliktklauseln zu bevorzugen, da sie häufiger für Unit Propagation verwendet werden können [MK11]. Das letzte Merkmal ist die Distanz der Indizies der Variablen innerhalb einer Klausel. In der Praxis lässt sich oft beobachten, dass Variablen, die eng zusammenhängen, auch nahe Indizes haben [Bj09]. Für die Klausel $(x_3 \vee \neg x_9 \vee x_{15})$ wäre die maximale Distanz beispielsweise $15 - 3 = 12$. Das Ergebnis zeigt dann, wie nah

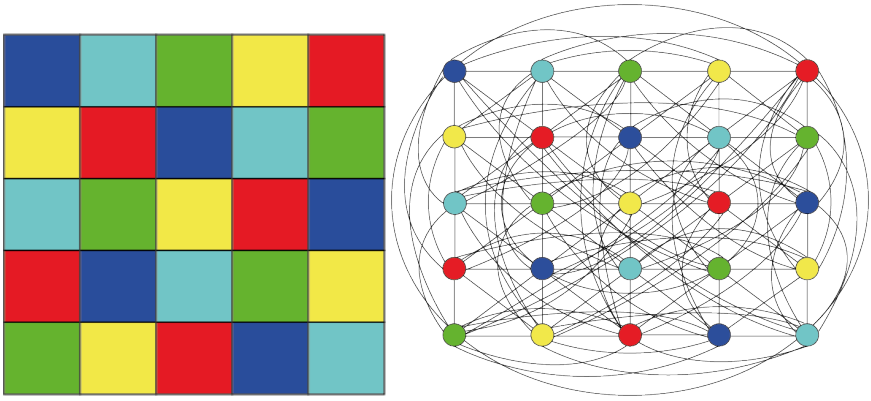
[3] SAT-Instanzen für das Generieren der Trainingsdaten stammen von http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

sich die Literale in einer Klausel stehen.
Es wurden weitere Merkmale ausprobiert, die sich aber als nicht performant erwiesen haben, da Merkmale bei denen die komplette Formel durchlaufen werden muss zu viel Zeit beanspruchen. Daher beruhen alle Merkmale nur auf einer einzelnen Klausel.
Zuerst muss das Naive Bayes Klassifikator trainiert werden. Dazu wird die Aktivität der Klausel betrachtet, das heißt, es wird gezählt, wie oft die Klausel ein Grund für einen Konflikt war. Hat die Klausel eine hohe Aktivitat wir die Klausel als geeignete Konfliktklausel eingestuft.
Die Datensätze werden anhand ihrer Merkmale bewertet und das Naive Bayes liefert zu jeder Konfliktklausel einen Wahrscheinlichkeitswert. Es werden unterschiedliche Wahrscheinlichkeiten (50%, 60%, 70%, 80% und 90%) verwendet. Das bedeutet, dass nur Konfliktklauseln an die anderen Solver übergeben werden, die den entsprechenden Wahrscheinlichkeitswert überschreiten und dadurch als geeignete Konfliktklausel klassifiziert wurden. Hier kann durch Ausprobieren herausgefunden werden, welches Verhältnis am Besten geeignet ist. Ist der Wahrscheinlichkeitswert zu hoch, ist der Aufwand zu groß, die Konfliktklauseln zu bewerten im Verhältnis zu der Anzahl der Konfliktklauseln, die gefunden werden. Ist der Wert zu niedrig, sind es zu viele Konfliktklauseln, die übergeben werden und der Arbeitsaufwand wird zu hoch.
Die anderen Solver werden in ihrem Vorgang nicht unterbrochen, wenn ein Solver eine neue Konfliktklausel bildet. Die anderen holen sich die Konfliktklauseln, wenn sie keine Lösung gefunden haben und Backtracking vorgenommen werden muss. Hierbei wird an die entsprechende Stelle zurückgesprungen, bei der die Entscheidung für die Belegung getroffen wurde. Unter Berücksichtigung der neuen Konfliktklauseln wird eine neue Belegung vorgenommen. Durch diese Art des Vorgehens werden Nebenläufigkeitsprobleme vermieden.

4 Evaluation

Die Evaluation wird anhand des Colored Graph von Knuth [Kn15] durchgeführt. Das Besondere an diesem Benchmark ist, dass es beliebig skalierbar ist. Dadurch können Benchmarks in unterschiedlichen Größen generiert werden. Die Idee ist es, ein Schachfeld so einzufärben, dass es keine zwei Felder gleicher Farbe gibt, auf denen sich zwei Damen gegenseitig schlagen können. Man hat also einen ungerichteten Graphen mit n^2 Knoten. Zwei Knoten sind genau dann durch eine Kante verbunden, wenn sie sich in der gleichen Reihe, Spalte oder Diagonale befinden. Es wird nun eine n-Färbung dieses Graphen gesucht, wodurch ein klassisches Färben des Graphen auf einem speziellen Graphen stattfindet. Aus diesen Bedingungen werden die Klauseln für das Benchmark gebildet.
In Abbildung 1(a) sieht man ein Beispiel des Feldes für einen Colored Graph mit fünf verschiedenen Farben. Da es in jeder Diagonalen, Horizontalen und Vertikalen nur eine Farbe gibt, ist dieses Benchmark erfüllbar. Entsprechend zu dem Feld ergibt sich folgender Graph in Abbildung 1(b).
Der parallele SAT-Solver ist in Java geschrieben und basiert auf dem SAT4J-Solver. Als sequentieller Vergleich wird der unmodifizierte SAT4J-Solver mit Default-Suchstrategie

verwendet. Das Benchmark wurde auf einem Rechner mit einem Intel Xeon CPU $E3-1240\,V2$ mit 3, 40 GHz Taktfrequenz, 8 Kernen und einem 32 GB Arbeitsspeicher ausgeführt. In Tabelle 1 werden die Ergebnisse des Benchmarks dargestellt. Dabei wird gemessen,



(a) Schachfeld von Knuth für fünf Farben (b) Colored Graph von Knuth für fünf Farben

Abb. 1: Colored Graph von Knuth

wie lange der parallele SAT4J-Solver mit Konfliktklauselhandling benötigt eine Lösung zu finden im Vergleich zum sequentiellen SAT4J-Solver (p =paralleler SAT-Solver). SAT bedeutet, dass die Formel erfüllbar ist und UNSAT dementsprechend das Gegenteil. Durch

n	Ergebnis	SAT4J	p 50%	p 60%	p 70% Bayes	p 80% Bayes	p 90% Bayes
2	UNSAT	1	30	39	6	3	19
3	UNSAT	2	81	29	12	10	29
4	UNSAT	2	54	63	12	11	33
5	SAT	6	29	54	10	11	34
6	UNSAT	5953	11775	12191	5925	5919	8661
7	SAT	40	1047	873	63	31	411

Tab. 1: Ergebnisse des Benchmarks in ms

das Benchmark konnte herausgefunden werden, dass die Konfliktklauseln mit 80% das beste Ergebnis erzielen und dadurch den sequentielle SAT-Solver in zwei von sechs Fällen schlägt. Ist das Verhältnis zwischen der Anzahl der erstellen Konfliktklauseln und des Aufwandes zur Klassifikation der Konfliktklauseln geeignet gewählt, benötigt der parallele SAT-Solver eine geringere Laufzeit.
Zusätzlich wird der parallele SAT-Solver anhand von verschiedenen SAT-Instanzen[4] gegen den sequentiellen SAT-Solver getestet. Hierbei werden nur die Konfliktklauseln mit 80% verwendet, da damit die besten Ergebnisse beim Colored Graph erzielt werden. Die Ergebnisse sind in Tabelle 2 dargestellt und man sieht, dass der parallele SAT-Solver in drei

[4] SAT-Instanzen stammen von http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

Datei	Variablen/Klauseln	Ergebnis	SAT4J	parallel 80% Bayes
ssa7552-038	1501/3575	SAT	32	302
ssa0432-003	435/1027	UNSAT	19	23
aim-50-1_6-yes1-4	50/80	SAT	13	8
ssa2670-141	986/2315	UNSAT	43	50
ssa7552-159	1363/3032	SAT	17	9
2bitadd_11	649/1562	SAT	2738	56
aim-50-1_6-yes1-2	50/80	SAT	2	26
aim-100-1_6-no-2	100/160	UNSAT	1	17

Tab. 2: Ergebnisse des Benchmarks in ms

von acht Fällen den sequentiellen SAT-Sovler schlagen kann, also ist das Verhältnis ähnlich zu dem Benchmark des Colored Graph.

5 Fazit

In diesem Paper wurde ein paralleler SAT-Solver basierend auf einem bereits bestehenden SAT-Solver untersucht. Der parallele SAT-Solver verwendete ein besonderes Konfliktklauselhandling, indem die Konfliktklauseln anhand eines Naive Bayes bewertet und klassifiziert wurden.
Die Konfliktklauseln wurden anhand von unterschiedlichen Merkmalen klassifiziert und den anderen Solvern zur Verfügung gestellt. Dabei kam es drauf an, ab welcher Wahrscheinlichkeit eine Konfliktklausel als geeignete Konfliktklausel klassifiziert wurde.
Im Benchmark wurden die unterschiedlichen Werte für die Wahrscheinlichkeit gegenüber gestellt. Durch die Auswertung der Benchmarks lässt sich feststellen, dass die Wahrscheinlichkeit von 80% das beste Ergebnis erzielte und den sequentiellen SAT-Solver in zwei von sechs Fällen beim Colored Graph von Knuth schlug. Wie man im Benchmark sehen konnte, war es möglich mit diesem parallelen Ansatz einen sequentiellen SAT-Solver zu schlagen. Insbesondere bei erfüllbaren Formeln hat sich der parallele Ansatz als besser herausgestellt.

Literatur

[AB09] A. Biere, H.van Maaren, T. Walsh: Handbook of Satisfiabiliy. IOS Press, 2009.

[Bj09] Successful SAT Encoding Techniques, http://jsat.ewi.tudelft.nl/addendum/Bjork_encoding.pdf, Stand: 03.04.2017.

[Co71] Cook, S. A.: , The complexity of theorem-proving procedures, 1971.

[Kn15] Knuth, D. E.: , The Art of Computer Programming: Volume 4 Combinatorial Algorithms, 2015.

[MK11] M. Kaufmann, S. Kottler: , Beyond Unit Propagation in SAT Solving, 2011.

[PB03] P. Beame, H. Kautz, A. Sabharwal: , Understanding the Power of Clause Learning, 2003.

[SH11] A Short Overview on Modern Parallel SAT-Solvers, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6140776, Stand: 18.06.2017.

[Zh04] Zhang, H.: , The Optimality of Naive Bayes, 2004.