

Skalierbares SAT Solving und dessen Anwendung¹

Dominik P. Schreiber²

Abstract: Das Problem der aussagenlogischen Erfüllbarkeit (*Propositional Satisfiability*, SAT) besteht darin, eine Variablenbelegung für eine gegebene aussagenlogische Formel zu finden, sodass die Formel "wahr" ergibt, oder, wenn keine solche Belegung existiert, Unerfüllbarkeit zu attestieren. Das Lösen derartiger Probleme, genannt SAT Solving, hat aufgrund seiner theoretischen Bedeutung, seiner generischen Natur und seiner breiten Anwendbarkeit auf eine Vielzahl von Problemen viel Beachtung erlangt. In der vorgestellten Dissertation widmen wir uns einer Reihe von Herausforderungen zur Skalierbarkeit von SAT Solving in modernen Rechenumgebungen. Zunächst stellen wir einen dezentralen und flexiblen Ressourcenzuteilungs-Ansatz vor, um viele SAT-Aufgaben zugleich möglichst effizient zu lösen. Zweitens präsentieren wir ein vielfach ausgezeichnetes System für paralleles und verteiltes SAT Solving und, damit verbunden, den ersten praktikablen Ansatz, aus derartigen Systemen Beweise der Unerfüllbarkeit zu gewinnen. Zuletzt stellen wir in einer Anwendungsstudie einen radikal neuartigen Ansatz für SAT-basiertes hierarchisches Planen vor.

1 Einleitung

Seit mehr als zwei Jahrtausenden beschäftigen sich Philosophen und Mathematiker mit der Formalisierung und Analyse logischer Sachverhalte [Sm22]. Ein Meilenstein der Geschichte im Hinblick auf unser heutiges Informationszeitalter war George Booles *Mathematical Analysis of Logic* [Bo47], welche eine Formalisierung eingeführt hat, die wir heute als Boolesche Algebra kennen. Variablen in dieser Algebra können nur zwei mögliche Werte annehmen – *wahr* oder *falsch*. Durch Junktoren wie *nicht* (\neg), *und* (\wedge), und *oder* (\vee) können Variablen verknüpft und so komplexe logische Ausdrücke gebildet werden. Besonders verbreitet ist die Darstellung eines logischen Ausdrucks als Konjunktion (\wedge) von *Klauseln*. Eine Klausel ist dabei eine Disjunktion (\vee) von (evtl. negierten) Variablen.

Die Boolesche Algebra ist in den modernen Natur- und Ingenieurwissenschaften zu einem unverzichtbaren Werkzeug geworden. Insbesondere stellt sie einen Grundpfeiler für die Funktionsweise, den Entwurf und die Programmierung von Rechnern dar. Ebenso fundamental ist damit die *Aussagenlogische Erfüllbarkeit*, kurz SAT (von engl. *satisfiability*). Bei diesem Problem geht es darum, die Variablen eines logischen Ausdrucks so zu belegen, dass der Ausdruck *wahr* wird, oder zu erkennen, dass dies unmöglich ist. Das Problem SAT hat beträchtliche Aufmerksamkeit auf sich gezogen, und das nicht nur durch seine theoretische Signifikanz als das prototypische NP-vollständige³ Problem [Co00]. SAT-Instanzen zu lösen bedeutet auch über eine der allgemeinsten Formen der Wissensrepräsentation zu schließen. Somit können eine Fülle von Anwendungen von SAT Solving

¹ Englischer Titel der Dissertation: "Scalable SAT Solving and its Application" [Sc23]

² Institut für Theoretische Informatik, Karlsruher Institut für Technologie, dominik.schreiber@kit.edu

³ NP-Vollständigkeit charakterisiert eine Klasse schwieriger Probleme, die aufeinander reduziert werden können und für deren Lösung im Allgemeinen nur Algorithmen mit exponentieller Laufzeit bekannt sind.

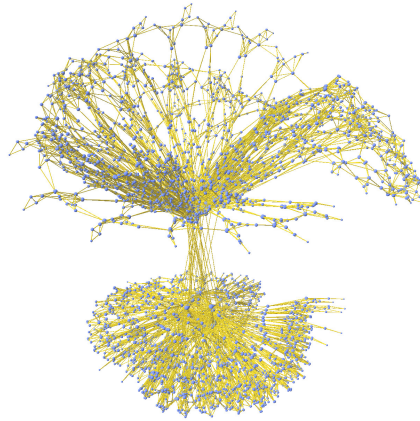


Abb. 1: Vereinfachte Visualisierung einer SAT-Instanz, welche die logische (In-)Äquivalenz zweier Multiplikations-Schaltkreise ausdrückt. Jeder Knoten entspricht einer Variablen; Knoten sind verbunden, wenn die Variablen gemeinsam in einer Klausel auftreten. Die Instanz besteht aus 13 018 Klauseln mit 4376 Variablen. In der *International SAT Competition 2023* ist es keinem sequentiellen SAT-Solver innerhalb von 5000 s gelungen, die *Unerfüllbarkeit* der Formel nachzuweisen. In diesem Fall bedeutet Unerfüllbarkeit, dass die verglichenen Schaltkreise in der Tat äquivalent sind.

profitieren. Innerhalb der letzten Jahrzehnte haben zunehmend effiziente Lösungsansätze ermöglicht, SAT als effizientes Werkzeug für verschiedenste Anwendungen zu nutzen, darunter automatisches Planen [KS92] und Scheduling [Gr12], Entwurf elektronischer Bausteine [MSS00], formale Verifikation [CI01], Kryptoanalyse [EME22], Beweisen von Theoremen [HKM16] und erklärbare KI [Da20]. Solche Anwendungen haben wiederum die Forschung an effizientem SAT Solving vorangetrieben [Fi23]. Nichtsdestotrotz ergeben sich in der Praxis stets Probleminstanzen, die mit dem Stand der Technik nicht realistisch lösbar sind. Abb. 1 zeigt ein besonders praxisrelevantes Beispiel.

Ziel unserer Forschung ist es, den Horizont der praktikabel und verlässlich lösbaren Probleme zu erweitern. Unser Hauptaugenmerk liegt dabei auf der effizienten Nutzung moderner, massiv paralleler Rechenumgebungen wie Hochleistungsrechenzentren (HPC, *high performance computing*) sowie Cloud Computing. Im Speziellen widmen wir uns drei zentralen Forschungsfragen: Wie können wir moderne verteilte Rechenumgebungen effizient für SAT Solving nutzen? Wie können wir SAT Solving in solchen Umgebungen vollständig vertrauenswürdig machen, um deren Einsatz für kritische Anwendungen zu ermöglichen? Und: Wie können komplexe Anwendungen effizienteren Gebrauch von SAT-Solvern machen, damit zuvor unlösbare Probleme bewältigt werden können?

In der vorliegenden Kurzfassung der entsprechenden Dissertation [Sc23] beschreiben wir zunächst unsere Beiträge zu dezentraler Ressourcenzuteilung für das Lösen von SAT-Instanzen und ähnlichen Aufgaben (Abschnitt 2) und thematisieren dann die effiziente Parallelisierung von SAT Solving selbst (Abschnitt 3). Anschließend beschreiben wir, wie aus skalierbarem SAT Solving Beweise der Unerfüllbarkeit gewonnen werden können

(Abschnitt 4) und umreißen eine Anwendungsfallstudie zu SAT-basierter hierarchischer Planung (Abschnitt 5). Schließlich ziehen wir ein kurzes Fazit (Abschnitt 6).

2 Dezentrale Verformbare Ressourcenzuteilung

Unseren ersten Beiträge sind durch eine simple Beobachtung motiviert: Die gleichzeitige Verarbeitung *vieler* unabhängiger Aufgaben kann deutlich effizienter sein als dieselben Rechenressourcen für die Verarbeitung einer *einzig*en Aufgabe aufzuwenden – insbesondere dann, wenn die genutzte Parallelisierung *innerhalb* einer Aufgabe nicht linear skaliert. Daher untersuchen wir die effiziente Verteilung von SAT-Instanzen und ähnlichen Aufgaben mit unbekannter Ausführungszeit auf große verteilte Umgebungen.

Starre Ressourcenzuteilungen, die für jede Aufgabe einmalig fixiert werden und dann unverändert bleiben, können nur sehr eingeschränkt auf unvorhersehbare Ereignisse wie frühzeitig abgeschlossene Aufgaben oder viele zugleich eintreffende Aufgaben reagieren. Für SAT-Instanzen, bei welchen bereits die Vorhersage *sequentieller* Laufzeiten eine Herausforderung darstellt [Hu14], können starre Ressourcenzuteilungen für parallel zu verrichtende Aufgaben zu stark suboptimaler Systemauslastung und/oder hohen Wartezeiten führen. Im Gegensatz dazu nennen wir eine parallele Aufgabe *verformbar* (engl. *malleable*), wenn sie eine *während ihrer Ausführung* schwankende Zahl von Rechenressourcen unterstützt [Fe97]. Verformbarkeit wird seit längerem als ein mächtiges Paradigma für faire und flexible Ressourcenzuteilung wahrgenommen [Gu14], wurde aber bisher hauptsächlich für iterative Berechnungen mit gleichmäßiger Parallelisierung und (nahezu) linearem Skalierungsverhalten untersucht. SAT und andere NP-schwierige Berechnungen sind dagegen unvorhersehbar und nur begrenzt skalierbar, weshalb wir an einer besonders umsichtigen Verwaltung der vorhandenen Ressourcen interessiert sind. Ein flexibles *on-demand*-Verarbeitungssystem mit verformbarer Ressourcenzuteilung könnte eintreffenden Aufgaben praktisch verzögerungsfrei erste Rechenressourcen bereitstellen, triviale Aufgaben bereits in Sekundenbruchteilen abschließen und schwierigeren Aufgaben ähnlich schnell einen fairen Anteil aller vorhandener Ressourcen anbieten.

Um diese Vision in die Tat umzusetzen, stellen wir einen Ansatz vor, der ohne jegliche Annahmen an die Ausführungszeit eintreffender Aufgaben eine schnelle, gerechte und

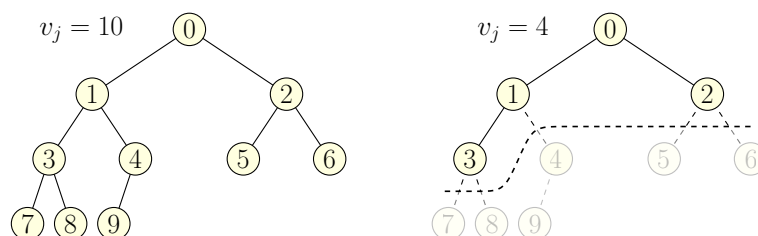


Abb. 2: Beispiel für Prozessbäume in unserem Ressourcenzuteilungssystem. Links: Eine Aufgabe j besitzt $v_j = 10$ verteilte Prozesse, nummeriert von 0 bis 9. Rechts: Eine aktualisierte Zuteilung $v_j := 4$ löst eine Verkleinerung des Prozessbaumes auf vier Prozesse aus.

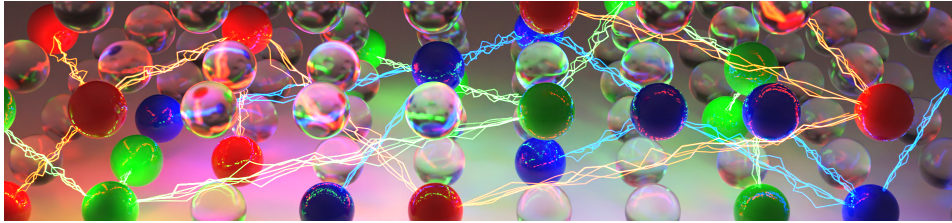


Abb. 3: Bildhafte Darstellung eines möglichen Systemzustands im vorgestellten Ressourcenzuteilungssystem. Jede Kugel entspricht einem Prozess, wobei jede Farbe einer konkreten Aufgabe entspricht (untätige Prozesse sind spiegelnd dargestellt). Die Verbindungen zwischen Kugeln entsprechen den Kanten im Prozessbaum der entsprechenden Aufgabe (siehe Abb. 2).

skalierbare Ressourcenzuteilung garantiert. Dies erreichen wir durch ein zweiteiliges dezentrales Verfahren. Erstens berechnen die m Prozesse im System kooperativ für jede aktive Aufgabe j eine gerechte Anzahl $v_j \in \mathbb{N}^+$ von Prozessen, sodass die Systemauslastung maximiert wird. Diese Verteilung berücksichtigt für jede Aufgabe einen *maximalen Ressourcenbedarf* $d_j \in \mathbb{N}^+$ und bestimmt ansonsten v_j proportional zur *Priorität* $p_j > 0$ jeder Aufgabe. Zweitens werden jeder Aufgabe j genau v_j konkrete Prozesse zugewiesen, sodass die Ressourcenzuteilung über die Zeit hinweg möglichst stabil bleibt. Jede Aufgabe im System wird durch einen *vollständigen Binärbaum* von v_j Prozessen repräsentiert, welcher sich auf der Blattebene je nach Bedarf erweitert oder verkleinert (siehe Abb. 2). Neue Prozesse für einen wachsenden Prozessbaum werden gefunden, indem die momentanen Blattknoten spezielle *Anfrage-Nachrichten* versenden. Diese werden auf geeignete Weise durch das System geleitet, bis sie von einem untätigen Prozess akzeptiert werden. Sowohl für die Berechnung der Zuteilungen als auch die Zuordnung der Anfrage-Nachrichten haben wir Algorithmen gefunden, die mit m Prozessen in $\mathcal{O}(\log m)$ Tiefe⁴ ausführbar sind.

In massiv parallelen Experimenten auf bis zu 6144 Rechenkernen eines Hochleistungsrechners (≤ 128 Knoten von SuperMUC-NG, Leibniz Rechenzentrum) konnten wir bei Hunderten von gleichzeitigen Aufgaben sehr kurze Warte- und Reaktionszeiten beobachten – durchschnittlich 10 ms für die Bereitstellung eines initialen Prozesses für eine eintreffende Aufgabe, 1 ms für die Berechnung fairer Zuteilungen für alle aktiven Aufgaben und 6 ms für die Bereitstellung der gewünschten zusätzlichen Ressourcen für eine Aufgabe mit gewachsenem Ressourcenanspruch. Dank diesen schnellen Reaktionen konnten wir zu fast allen Zeitpunkten eine nahezu optimale Systemauslastung ($> 99\%$) feststellen. Wir haben verschiedene Fallstudien für mögliche Anwendungen unseres Systems durchgeführt, darunter k -means-Clustering (Bachelorarbeit M. Dörr) und hierarchische Planung (Masterarbeit N. Wilhelm). Die zentrale betrachtete Anwendung im Rahmen unserer Arbeit ist indes SAT Solving, wie wir in den folgenden Abschnitten weiter ausführen.

⁴ Die *Tiefe* (*span*, *depth*) gibt für eine parallele Berechnung die Länge des kritischen Pfades durch das Netzwerk aller Datenabhängigkeiten an [Sa19].

3 Skalierbares Verteiltes SAT Solving

Im Folgenden beschäftigen wir uns mit der effizienten Parallelisierung von SAT Solving selbst – alleinstehend sowie innerhalb der in Abschnitt 2 beschriebenen Umgebung.

Die bisher besten anwendungs-unabhängigen parallelen SAT-Solver⁵ sind *Portfolios mit Klauselaustausch*, bei welchen eine Reihe unterschiedlich parametrisierter sequentieller Solver das gegebene Problem gleichzeitig bearbeiten. Jeder Solver produziert pro Sekunde Tausende von sogenannten *Konfliktklauseln* – logische Konsequenzen des Problems, die für den Lösungsfortschritt hilfreich sind. Gelegentlich tauschen die Solver eine Auswahl dieser Klauseln untereinander aus. Einige solcher Systeme zeigen ordentliche Performance auf wenigen Dutzend Kernen [BS18, FB22]. In verteilten Rechenumgebungen mit Hunderten von Kernen werden bisherige Ansätze jedoch zunehmend ineffizient. Konkret erreichte der vorherige Stand der Technik in verteiltem SAT Solving, HORDESAT [BSS15], auf industriellen Benchmarks eine Median-Beschleunigung⁶ von 13 auf 2048 Kernen, was einer (Ressourcen-)Effizienz von lediglich $13/2048 = 0.6\%$ entspricht.

Wir verwenden HORDESAT als Ausgangspunkt und überdenken konsequent dessen algorithmische Bestandteile. Insbesondere entwerfen wir einen kompakten Ansatz für periodischen globalen Klauselaustausch, der für Tausende von Solvern geeignet ist. Die zentrale Motivation unseres Ansatzes ist, mittels eines skalierbaren Protokolls eine begrenzte Menge der *global besten paarweise verschiedenen* Konfliktklauseln zu identifizieren und dann an alle Solver weiterzureichen (siehe Abb. 4). Unser Ansatz erkennt zuverlässig Duplikate innerhalb einer Austausch-Operation und kann zusätzlich mithilfe einer verteilten Filter-Operation zeitlich versetzte Duplikate eliminieren. Zudem ist unser verteilter Solver *performbar* und kann daher im Rahmen von flexibler Ressourcenzuteilung (Abschnitt 2) verwendet werden. Auf pragmatischer Ebene haben wir verschiedene Maßnahmen zur Verringerung des benötigten Arbeitsspeichers ergriffen und die momentan besten sequentiellen SAT-Solver in unser System integriert.

⁵ Andere Parallelisierungen, die auf einer expliziten Partitionierung des Suchraums beruhen, sind auf bestimmten kombinatorischen Problemen sehr erfolgreich aber erreichen auf vielfältigen Anwendungsproblemen ausgeführt nicht die Performance Portfolio-basierter Ansätze [BS18].

⁶ Die Beschleunigung (*speedup*) eines parallelen Ansatzes P auf einer bestimmten Eingabe ist definiert als die Laufzeit des besten bekannten sequentiellen Algorithmus geteilt durch die Laufzeit von P [Sa19].

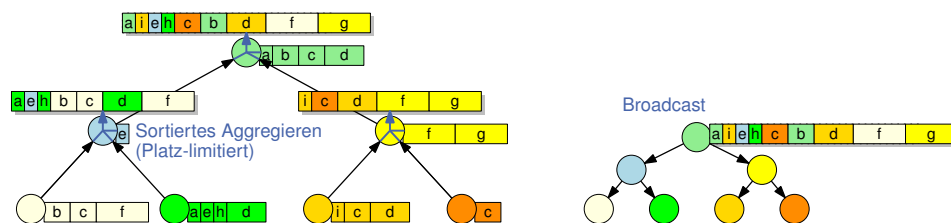


Abb. 4: Klauselaustausch-Operation von MALLOBSAT. Links: Sieben Prozesse aggregieren ihre jeweils besten lokal produzierten Klauseln (dargestellt durch einzelne Buchstaben) entlang eines Binärbaums, wobei kurze Klauseln stets priorisiert und wiederholt auftretende Klauseln eliminiert werden. Rechts: Der so entstandene Austausch-Puffer wird an alle Prozesse weitergeleitet.

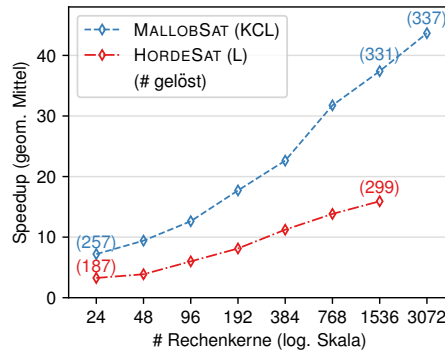


Abb. 5: Skalierverhalten von MALLOBSAT (unser Ansatz) gegenüber HORDESAT (vorheriger Ansatz) auf 400 Probleminstanzen der International SAT Competition 2021. Dargestellt ist das geometrische Mittel der Beschleunigungen auf allen von *beiden Ansätzen* gelösten Instanzen. Die eingeklammerten Zahlen geben zusätzlich die jeweilige Anzahl der gelösten Instanzen an.

In Experimenten auf bis zu 3072 Kernen konnte unser Ansatz namens MALLOBSAT die bisher besten Beschleunigungen von verteiltem SAT Solving mehr als verdoppeln (siehe Abb. 5). Wir konnten nachweisen, dass der Klauselaustausch zwischen Solvern den Hauptgrund für diese Skalierbarkeit darstellt. In der renommierten *International SAT Competition* [Fr21] hat MALLOBSAT die Cloud-Kategorie (1600 Threads auf 100 Maschinen) vier Jahre in Folge dominiert und sich auch auf moderat paralleler Hardware (64 Threads) als äußerst konkurrenzfähig erwiesen. Infolgedessen wurde unser System 2022 von Amazon-Wissenschaftler B. Cook als “*der mit Abstand mächtigste SAT-Solver des Planeten*” [Co22] (übers.) betitelt. Darüber hinaus heben wir die Vorteile einer Kombination aus paralleler Aufgabenverarbeitung und verformbarem SAT Solving hervor: In Experimenten auf bis zu 6400 Kernen lässt sich beobachten, dass beim gleichzeitigen Lösen vieler Aufgaben die schwierigen Aufgaben über die Zeit hinweg zunehmend viele Rechenressourcen erhalten, welche durch das Lösen einfacherer Aufgaben nach und nach verfügbar werden. Das führt zu signifikanten Performanceverbesserungen und, damit einhergehend, wesentlich höherer Effizienz als mit starrer Ressourcenzuteilung.

4 Verteilte Erzeugung von Unerfüllbarkeits-Beweisen

Heutige sequentielle Solver sind vertrauenswürdig in dem Sinne, dass sie bei Bedarf einen unabhängig verifizierbaren *Beweis* für die Unerfüllbarkeit einer Probleminstanz ausgeben können [Cr17]. Effizienten parallelen SAT-Solvern fehlt diese fundamentale Technik bisher, was deren Einsatz für kritische Anwendungen stark einschränkt. Ein wichtiges Beispiel ist Software- und Hardware-Verifikation, denn hier bedeutet eine unerfüllbare Formel üblicherweise, dass die betrachtete Korrektheitseigenschaft innerhalb der gewählten Schranken *stets gewahrt bleibt* [Cl01]. Die Unerfüllbarkeit einer Formel ist also oft eine kritische Eigenschaft, und ein expliziter Nachweis dafür kann unverzichtbar sein.

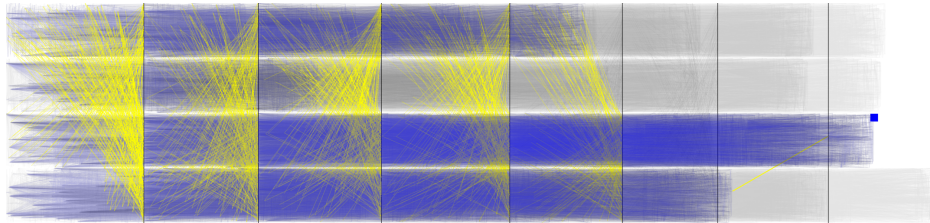


Abb. 6: Visualisierung einer Beweisstruktur mit vier Solvoren (v.o.n.u.), die während des Lösens Netzwerke von Klausel-Abhängigkeiten herleiten (v.l.n.r.) und in partielle Beweise schreiben. Vertikale Linien markieren Klauselaustausch-Operationen. Die verfolgten transitiven Abhängigkeiten des finalen Schlussschritts (blaues Quadrat rechts) sind eingefärbt – horizontal/blau für lokale, diagonal/gelb für externe Abhängigkeiten. Die betrachtete Formel hat 180 Variablen und wurde in unter 2 s gelöst, wobei 32 730 Herleitungen mit insgesamt 581 163 Abhängigkeiten produziert wurden.

Wir stellen den ersten praktikablen und skalierbaren Ansatz zur Erzeugung solcher Zertifikate für paralleles SAT Solving vor. Eine Schlüsseltechnologie für unseren Ansatz ist das Klausel-basierte Beweisformat *LRAT* [Cr17], bei welchem jeder einzelne Beweisschritt einer hergeleiteten Konfliktklausel entspricht und die Abhängigkeiten zwischen den Beweisschritten mithilfe von global eindeutigen IDs explizit ausgedrückt werden. Unser Lösungsansatz orchestriert wie in Abschnitt 3 viele sequentielle Solver, welche nun individuell ihre hergeleiteten Klauseln im LRAT-Format in partielle Beweisdateien ausgeben. Sobald ein Solver Unerfüllbarkeit attestiert, spult unser verteilter Algorithmus gewissermaßen den Lösungsvorgang in der Zeit zurück, indem jeder Solver seinen eigenen Beweis in umgekehrter Reihenfolge traversiert und dabei alle transitiven Abhängigkeiten des finalen Schlussschritts zurückverfolgt (Abb. 6). Insbesondere wird jede Klauselaustausch-Operation umgekehrt: Die IDs aller als notwendig erkannten *extern* erzeugten Klauseln werden zu ihrem jeweiligen Herkunftsprozess umverteilt, welcher deren Abhängigkeiten dann weiter verfolgt. Alle so verfolgten Beweisschritte werden über ein verteiltes Protokoll in einen zentralen Beweis aggregiert, welcher dann unabhängig validiert werden kann.

In Experimenten in einer Cloudumgebung hat unser Ansatz unter vertretbarem Zusatzaufwand bis zu 80 GB große und dennoch praktikabel überprüfbare Beweise generiert. Selbst bei einer Berechnung mit 1600 Solver-Threads funktioniert die Beweisstruktur im Durchschnitt ähnlich schnell wie das verteilte Lösen selbst, während das Überprüfen der Beweise als sequentielle Operation mehr Zeit benötigt (im Durchschnitt die vierfache Laufzeit des eigentlichen Lösens). Insgesamt stellt die beobachtete Performance eine substantielle Verbesserung gegenüber jeglichen vorherigen Ansätzen für parallele Beweisgenerierung dar. Der vorgestellte LRAT-basierte Ansatz entspricht somit einem wichtigen Schritt, um skalierbares SAT Solving für kritische Anwendungen zu erschließen.

5 Fallstudie: SAT-basierte Hierarchische Planung

Zu guter Letzt berichten wir von einer umfangreichen Fallstudie zu angewandtem SAT Solving. Konkret stellen wir einen SAT-basierten Ansatz für die Planung von hierarchi-

schen Aufgabennetzwerken mit totaler Ordnung (*Totally Ordered Hierarchical Task Networks*, TOHTN) vor – ein beliebter Zweig der automatisierten Planung, der reichhaltige Möglichkeiten für die Modellierung komplexer Planungsaufgaben bereit hält [BAH19]. Eine Problem Instanz enthält einen initialen *Weltzustand* sowie eine initiale Sequenz von zu verrichtenden *Aufgaben*. Eine Aufgabe ist *primitiv* oder *hierarchisch*, wobei eine primitive Aufgabe einer atomar ausführbaren Operation mit Vor- und Nachbedingungen gegenüber dem Weltzustand entspricht. Eine hierarchische Aufgabe hingegen wird durch Anwendung von bedingten *Reduktionsregeln* rekursiv in eine Sequenz von Teilaufgaben zerlegt.

Üblicherweise liegen Planungsprobleme in einer *parametrisierten* Form vor. Wenn ein Logistik-Problem beispielsweise Mengen von Trucks T und Orten L beinhaltet, kann aus einer parametrisierten Aufgabe $drive(\tau, \alpha, \omega)$ eine konkrete Aktion $drive(t, x, y)$ *instanziiert* werden, um einen bestimmten Truck $t \in T$ von Ort $x \in L$ nach Ort $y \in L$ zu bewegen. Anfänglich liegen nur der initiale Weltzustand und die initialen Aufgaben instanziiert, also ohne freie Parameter vor. Viele Ansätze für TOHTN-Planung, darunter alle SAT-basierten Verfahren, beruhen auf einem Vorverarbeitungsschritt, welcher sämtliche potentiell relevanten Objekte instanziiert, um dann mit einer vereinfachten Repräsentation des Problems zu arbeiten. Im obigen Beispiel entspräche das der Instanziiierung der Aufgabe $drive(\tau, \alpha, \omega)$ für alle möglichen Kombinationen von Trucks τ und Paaren von Orten α, ω . Auf einigen Problemen führt dies zu einer kombinatorischen Explosion der Eingabedaten noch bevor mit der eigentlichen Planung begonnen werden kann [WTH19].

Wir präsentieren den ersten SAT-basierten TOHTN-Planungsansatz, welcher die parametrisierte Problembeschreibung beibehält und damit die beschriebene Vorverarbeitung vermeidet. Entgegen vorherigen Ansätzen instanziiieren wir im obigen Beispiel nicht etwa $|T| \times |L|^2$ Aktionen, sondern kodieren die *parametrisierte* Aufgabe gemeinsam mit ihren Parametern τ, α, ω und deren jeweiligen Wertebereichen *direkt* in Aussagenlogik. Somit ist es letztlich der SAT-Solver, der über die konkrete Instanziiierung einer Aufgabe entscheidet. Unser integrierter Ansatz namens LILOTANE (*Lifted Logic for Task Networks*) alterniert zwischen einer vorsichtigen Instanziiierung der Planungshierarchie, deren Kodierung in Aussagenlogik und dem eigentlichen SAT Solving. Die erzeugten Formeln sind im Vergleich zu vorherigen SAT-basierten Ansätzen oft um ein bis zwei Größenordnungen kleiner. Dementsprechend erzielte LILOTANE in der *International Planning Competition 2020* den zweiten Platz (hinter einem nicht vollständigen Greedy-Algorithmus) und konnte dort die meisten Probleme lösen und die kürzesten Pläne finden.

Abschließend stellen wir eine Möglichkeit vor, hierarchische Planung auf massiv paralleler Hardware durchzuführen, indem wir unsere Ressourcenzuteilung (Abschnitt 2) mit verformbarem SAT Solving (Abschnitt 3) einsetzen. In einem Experiment auf 2348 Kernen bearbeitet unser System 587 Planungsinstanzen zugleich, was im Durchschnitt vier Kernen pro Instanz entspricht. Die erzielbare Beschleunigung gegenüber einer sequentiellen Ausführung ist stark abhängig von der betrachteten Planungsdomäne und speziell von der Schwierigkeit der konstruierten SAT-Instanzen. Die Resultate reichen daher von geringfügiger Verlangsamung (ca. 5%) bis hin zu signifikanter Beschleunigung ($\leq 24,8\times$). Insofern dient unsere Anwendungsstudie zusätzlich als erfolgreicher Stresstest für unsere Beiträge zu skalierbarem SAT Solving im Kontext einer anspruchsvollen Anwendung.

6 Fazit

Im Gesamten betrachtet hat unsere Arbeit zu erheblichen Fortschritten bei skalierbarem SAT Solving und dessen effizienter Anwendung geführt. Damit konnten wir erfolgreich den Horizont der logischen Probleme erweitern, die in modernen Rechenumgebungen praktikabel lösbar sind. Insbesondere haben unsere Beiträge zu verteiltem SAT Solving sowie zu hierarchischer Planung internationale Aufmerksamkeit in der Wissenschaft sowie Industrie erlangt und werden von unabhängigen Dritten verwendet und weiterentwickelt.

Im Anschluss an die vorgestellte Forschung konzentrieren wir uns momentan auf eine Weiterentwicklung und Anwendung der beschriebenen Techniken zu verteiltem SAT Solving und insbesondere auf eine Verringerung der verbleibenden redundanten Arbeit, die zwischen den sequentiellen Solvern verrichtet wird.

Literaturverzeichnis

- [BAH19] Bercher, Pascal; Alford, Ron; Höller, Daniel: A Survey on Hierarchical Planning – One Abstract Idea, Many Concrete Realizations. In: Proc. IJCAI. S. 6267–6275, 2019.
- [Bo47] Boole, George: The mathematical analysis of logic. Philosophical Library, 1847.
- [BS18] Balyo, Tomáš; Sinz, Carsten: Parallel Satisfiability. In (Hamadi, Youssef; Sais, Lakhdar, Hrsg.): Handbook of Parallel Constraint Reasoning. Springer, 2018.
- [BSS15] Balyo, Tomáš; Sanders, Peter; Sinz, Carsten: Hordesat: A massively parallel portfolio SAT solver. In: Proc. SAT. S. 156–172, 2015.
- [Cl01] Clarke, Edmund; Biere, Armin; Raimi, Richard; Zhu, Yunshan: Bounded model checking using satisfiability solving. In: Formal methods in system design. S. 7–34, 2001.
- [Co00] Cook, Stephen: The P versus NP problem. Clay Mathematics Institute, 2000.
- [Co22] Cook, Byron: Automated reasoning’s scientific frontiers. Amazon Science, 2022.
- [Cr17] Cruz-Filipe, Luis; Heule, Marijn J. H.; Hunt, Warren A.; Kaufmann, Matt; Schneider-Kamp, Peter: Efficient Certified RAT Verification. In: Proc. CADE. S. 220–236, 2017.
- [Da20] Darwiche, Adnan: Three modern roles for logic in AI. In: Proc. ACM SIGMOD-SIGACT-SIGAI Symp. Principles of Database Systems. S. 229–243, 2020.
- [EME22] Erlacher, Johannes; Mendel, Florian; Eichlseder, Maria: Bounds for the security of AS-CON against differential and linear cryptanalysis. IACR Transactions on Symmetric Cryptology, S. 64–87, 2022.
- [FB22] Fleury, Mathias; Biere, Armin: Scalable Proof Producing Multi-Threaded SAT Solving with Gimsatul through Sharing instead of Copying Clauses. In: Proc. Pragmatics of SAT. 2022.
- [Fe97] Feitelson, Dror G.: Job scheduling in multiprogrammed parallel systems. IBM Research Report, 1997.
- [Fi23] Fichte, Johannes K.; Le Berre, Daniel; Hecher, Markus; Szeider, Stefan: The Silent (R)evolution of SAT. Comm. ACM, 66(6):64–72, 2023.

- [Fr21] Froleys, Nils; Heule, Marijn J. H.; Iser, Markus; Jarvisalo, Matti; Suda, Martin: SAT competition 2020. *Artificial Intelligence*, 301:103572, 2021.
- [Gr12] Großmann, Peter; Hölldobler, Steffen; Manthey, Norbert; Nachtigall, Karl; Opitz, Jens; Steinke, Peter: Solving periodic event scheduling problems with SAT. In: *Proc. IEA/AIE*. S. 166–175, 2012.
- [Gu14] Gupta, Abhishek; Acun, Bilge; Sarood, Osman; Kalé, Laxmikant V.: Towards realizing the potential of malleable jobs. In: *Proc. HiPC*. S. 1–10, 2014.
- [HKM16] Heule, Marijn J. H.; Kullmann, Oliver; Marek, Victor: Solving and verifying the boolean pythagorean triples problem via cube-and-conquer. In: *Proc. SAT*. S. 228–245, 2016.
- [Hu14] Hutter, Frank; Xu, Lin; Hoos, Holger H.; Leyton-Brown, Kevin: Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.
- [KS92] Kautz, Henry A.; Selman, Bart: Planning as Satisfiability. In: *Proc. ECAI*. Jgg. 92. Citeseer, S. 359–363, 1992.
- [MSS00] Marques-Silva, João P.; Sakallah, Karem A.: Boolean satisfiability in electronic design automation. In: *Proc. DAC*. S. 675–680, 2000.
- [Sa19] Sanders, Peter; Mehlhorn, Kurt; Dietzfelbinger, Martin; Dementiev, Roman: *Sequential and Parallel Algorithms and Data Structures*. Springer, 2019.
- [Sc23] Schreiber, Dominik P.: *Scalable SAT Solving and its Application*. Dissertation, Karlsruher Institut für Technologie, 2023.
- [Sm22] Smith, Robin: *Aristotle’s Logic*. The Stanford Encyclopedia of Philosophy, 2022.
- [WTH19] Wichlacz, Julia; Torralba, Alvaro; Hoffmann, Jörg: Construction-planning models in minecraft. In: *Proc. ICAPS Workshop on Hierarchical Planning*. 2019.



Dominik P. Schreiber (geb. 1994) ist wissenschaftlicher Mitarbeiter am Karlsruher Institut für Technologie (KIT). Vor seiner Promotion mit Auszeichnung am KIT-Institut für Theoretische Informatik bei Prof. Peter Sanders hat D. Schreiber Softwaretechnik an der Universität Stuttgart sowie Informatik im Doppelmaster zwischen dem Institut National Polytechnique in Grenoble (Frankreich) und dem KIT studiert. Sein wissenschaftlicher Schwerpunkt sind Lösungsverfahren für NP-schwere Probleme, hpts. aussagenlogische Erfüllbarkeit und automatisierte Planung, mit speziellem Fokus auf Parallelverarbeitung. Seine Publikationen wurden für zwei Best Paper Awards nominiert und die von ihm entwickelte Software wurde in internationalen Wettbewerben vielfach ausgezeichnet. D. Schreiber war bereits Mitglied im Programmkomitee der *International Conference on Automated Planning & Scheduling* (ICAPS) und Co-Organisator der *International Planning Competition* (IPC). Er ist Alumnus der Studienstiftung des Deutschen Volkes.