Steffen Hölldobler et al. (Hrsg.): Ausgezeichnete Informatikdissertationen 2021,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2022 91

Werkzeuge und Methoden zum Lösen von Problemen mittels Baumweite[1]

Markus Hecher[2]

Abstract: In den letzten Jahrzehnten konnte ein beachtlicher Fortschritt im Bereich der Aussagenlogik verzeichnet werden, der sich durch überwältigend schnelle Computerprogramme (Solver) zur Lösung aussagenlogischer Formeln äußert. Einer der Gründe dieser Schnelligkeit befasst sich mit strukturellen Eigenschaften von Probleminstanzen, zum Beispiel der sogenannten Baumweite, welche versucht zu messen, wie groß der Abstand von Probleminstanzen zu einfachen Strukturen (Bäumen) ist. Diese Arbeit befasst sich mit Problemen der Künstlichen Intelligenz (KI) sowie Baumweite-basierenden Methoden und Werkzeugen zum Lösen dieser. Wir präsentieren einen neuen Typ von Problemreduktion, den wir als „zerlegungsangeleitet" bezeichnen. Dieser ist die Basis, um eine lange offen gebliebene Frage betreffend quantifizierter, aussagenlogischer Formeln (QBF) bei beschränkter Baumweite zu zeigen. Die Lösung der Frage ermöglicht ein neues Meta-Werkzeug zum Beweisen präziser unterer Laufzeitschranken einer Vielzahl von Problemen der KI. Trotz dieser Schranken implementieren wir einen Solver für Erweiterungen von SAT, der Baumweite effizient ausnutzt.

1 Einleitung

In den letzten Jahrzehnten konnte ein beachtlicher Fortschritt im Bereich der *Aussagenlogik* [Bi09; KL99] verzeichnet werden. Dieser äußerte sich dadurch, dass für das wichtigste Problem in diesem Bereich, genannt „SAT", welches sich mit der Erfüllbarkeit einer gegebenen aussagenlogische Formel befasst, überwältigend schnelle Computerprogramme („Solver") entwickelt wurden. Diese Solver liefern eine beeindruckende Leistung, weil sie oft selbst Probleminstanzen mit mehreren Millionen von Variablen spielend leicht lösen können. Das ist deswegen so bemerkenswert, weil SAT einer der bekanntesten Vertreter der NP-vollständigen Probleme ist [Co71]. Vom theoretischen Standpunkt aus bedeutet dies keine gute Nachricht, da solche Probleme im Allgemeinen nicht effizient gelöst werden können (angenommen P $\neq$ NP). Mit der Zeit hat sich sogar eine stärkere Annahme als P $\neq$ NP entwickelt, die wissenschaftlich weitestgehend akzeptiert und *Exponentialzeithypothese (EZH)* [IPZ01] genannt wird. Diese Hypothese besagt, dass man im schlimmsten Fall für das Lösen einer aussagenlogischen Formel exponentielle Laufzeit in der Anzahl der Variablen benötigt. Dieser vermeintliche Widerspruch zwischen Praxis und Theorie ist noch immer nicht vollständig geklärt, denn wahrscheinlich gibt es viele ineinandergreifende Gründe für die Schnelligkeit aktueller SAT-Solver. Einer davon befasst sich mit strukturellen Eigenschaften von Probleminstanzen, die indirekt und intern von diesen Solvern ausgenützt werden, was zumindest theoretisch demonstriert wurde [AFT11].

[1] Originaltitel in englischer Sprache: "Advanced Tools and Methods for Treewidth-Based Problem Solving".

[2] TU Wien, Logic and Computation, Favoritenstraße 9–11, 1040 Wien, Österreich, hecher@dbai.tuwien.ac.at.

Die Dissertation beschäftigt sich mit solchen strukturellen Eigenschaften, nämlich mit der sogenannten *Baumweite* [RS86]. Die Baumweite ist sehr gut erforscht und versucht zu messen, wie groß der Abstand von Probleminstanzen zu Bäumen ist (Baumnähe). Allerdings ist dieser Parameter sehr generisch und bei Weitem nicht auf Problemstellungen der Aussagenlogik beschränkt. Tatsächlich gibt es viele weitere Probleme, die parametrisiert mit Baumweite in polynomieller Zeit gelöst werden können. Ebenso gibt es viele Probleme in der Wissensrepräsentation und Künstlichen Intelligenz (KI), die bei beschränkter Baumweite in polynomieller Zeit gelöst werden können, obwohl man davon ausgeht, dass sie härter sind als SAT. Ein prominentes Beispiel solcher Probleme ist QSAT, welches sich mit der Gültigkeit einer gegebenen quantifizierten, aussagenlogischen Formel (QBF) befasst. Dies sind aussagenlogische Formeln, wobei gewisse Variablen existenziell bzw. universell quantifiziert werden können [Bi09; KL99]. Ein weiterer Vertreter solcher Probleme beschäftigt sich mit der Gültigkeit eines Antwortmengen (ASP) Programms [BET11], welches zusätzlich zur Erfüllbarkeit (SAT) fordert, dass Variablen nur dann auf wahr gesetzt werden, wenn eine Begründung vorliegt. Bemerkenswerterweise wird auch im Zusammenhang mit der Baumweite, ähnlich zu Methoden der klassischen Komplexitätstheorie, die Komplexität (Härte) solcher Probleme quantifiziert, was die exakte Laufzeitabhängigkeit beim Problemlösen in der Baumweite (Stufe der Exponentialität) beschreibt.

Beitrag. Wir präsentieren Methoden und Werkzeuge, um präzise Laufzeitresultate (*obere Laufzeitschranken*) für prominente Fragmente der Antwortmengenprogrammierung (ASP), welche ein kanonisches Paradigma zum Lösen von Problemen der Wissensrepräsentation darstellt, zu erhalten. Unsere Resultate basieren auf dem Konzept der dynamischen Programmierung, die angeleitet durch eine sogenannte Baumzerlegung und ähnlich dem Prinzip „Teile-und-herrsche“ funktioniert. Solch eine *Baumzerlegung* ist eine konkrete, strukturelle Zerlegung einer Probleminstanz, die sich stark an der Baumweite orientiert.

Des Weiteren präsentieren wir einen neuen Typ von Problemreduktion, den wir als „*decomposition-guided (DG)*“, also „zerlegungsangeleitet“, bezeichnen. Dieser Reduktionstyp erlaubt es, Baumweiteerhöhungen und -verringerungen während einer Problemreduktion von einem bestimmten Problem zu einem anderen Problem präzise zu untersuchen und zu kontrollieren. Zusätzlich ist dieser neue Reduktionstyp die Basis, um eine lange offen gebliebene Frage betreffend quantifizierter, aussagenlogischer Formeln zu zeigen. Tatsächlich sind wir damit in der Lage, präzise *untere (Laufzeit-)schranken*, unter der Annahme der Exponentialzeithypothese, für das Problem QSAT bei beschränkter Baumweite zu zeigen. Mit anderen Worten können wir mit diesem Konzept der DG-Reduktionen zeigen, dass das Problem ℓ-QSAT (QSAT beschränkt auf Quantorenrang ℓ) parametrisiert mit Baumweite k im Allgemeinen nicht besser als in einer Laufzeit, die ℓ-fach exponentiell in der Baumweite und polynomiell in der Instanzgröße ist[3]. Dieses Resultat präzisiert die Beobachtung, dass das Lösen von QSAT für Baumweite hierarchische Laufzeiten verursacht [AO14; PV06], und hebt auf nicht-inkrementelle Weise ein bekanntes Ergebnis für

[3] „ℓ-fache Exponentialität“ meint eine Laufzeit eines Turms der Zahlen 2 der Höhe ℓ mit k an der Spitze.

Quantorenrang 2 [LM17] auf beliebige Quantorenränge. Damit beantworten wir via DG-Reduktionen auf konstruktive Weise eine Frage betreffend deckender unterer Schranken, die seit 2004 offen geblieben ist [Ch04]. Die Antwort darauf hat weitere Konsequenzen.

Das Resultat über die untere Schranke des Problems QSAT erlaubt es, ein *neues Meta-Werkzeug* zum Beweisen unterer Schranken vieler Probleme der Wissensrepräsentation und künstlichen Intelligenz, zu etablieren. In weiterer Konsequenz können wir damit auch zeigen, dass die oberen Schranken sowie die DG-Reduktionen dieser Arbeit unter der Hypothese EZH „eng“ sind, d.h., sie können wahrscheinlich nicht mehr signifikant verbessert werden. Die Ergebnisse betreffend der unteren Schranken für QSAT und das dazugehörige Werkzeug konstituieren eine präzise Hierarchie von über Baumweite parametrisierte Laufzeitklassen. Diese Laufzeitklassen können verwendet werden, um die Härte von KI-relevanten Probleme für das Ausnützen von Baumweite zu quantifizieren und entsprechend ihrer Baumweite-Laufzeitabhängigkeit zu kategorisieren.

Schlussendlich und trotz der genannten Resultate betreffend unterer Schranken sind wir in der Lage, eine effiziente Implementierung von Algorithmen basierend auf dynamischer Programmierung, die entlang einer Baumzerlegung angeleitet wird, zur Verfügung zu stellen. Dabei funktioniert unser Ansatz dahingehend, passende *Abstraktionen* von Instanzen zu finden, die dann sukzessive und auf rekursive Art und Weise verfeinert und verbessert werden. Inspiriert durch die enorme Effizienz und Effektivität der SAT-Solver, ist unsere Implementierung ein *hybrider Ansatz*, weil sie den starken Gebrauch von SAT-Solvern zum Lösen diverser Subprobleme, die während der dynamischen Programmierung auftreten, pflegt. Dabei stellt sich heraus, dass sich der resultierende Solver unserer Implementierung in Bezug auf die Effizienz beim Lösen von zwei kanonischen, SAT-verwandten Zählproblemen kompetitiv zu bestehenden Solvern verhält. Tatsächlich können wir Instanzen, die die oberen Schranken von Baumweite 260 übersteigen, lösen, womit die Wichtigkeit der Berücksichtigung von Baumweite in modernen Solver-Designs unterstrichen wird.

Überblick und Struktur. Als Kurzüberblick über Schlüsselresultate dieser Arbeit und einiger bestehender Resultate, stellen wir Tabelle 1 zur Verfügung, welche einen Auszug ausgewählter unterer und oberer Schranken, die in der Doktorarbeit bewiesen werden, kurz zusammenfasst. Bei den dargestellten Problemen handelt es sich um Varianten von SAT, QSAT sowie wichtiger Fragmente von ASP, die unter anderem essenziell für das Modellieren in der Wissensrepräsentation sind. Diese Tabelle wird in der Arbeit präzisiert und deutlich ergänzt, um auf die Konsequenzen für weitere Probleme der KI aufmerksam zu machen. Zu diesem Zweck verweisen wir allerdings auf auf Kapitel 6 [He21, Tabelle 6.1].

In der nächsten Sektion wird kurz auf notwendige Grundlagen eingegangen, danach wird in Sektion 3 das Konzept der „decomposition-guided“ Reduktionen erläutert, was einer vereinfachten Darstellung von Kapitel 4 der Doktorarbeit entspricht. In den folgenden Sektionen 4 und 5 stellen wir neue untere Schranken für QSAT und ASP unter Verwendung von DG-Reduktionen vor, was Kapitel 5.1 und 5.2 behandelt. Sektion 6 skizziert entspre-

Problem	Laufzeitabhängigkeit in der Baumweite k			
	Exponentialität	Laufzeit*	Obere Schranke	Untere Schranke
SAT	einfach exponentiell	$2^{\Theta(k)}$	△[SS10]	▽[IPZ01]
TIGHT ASP	einfach exponentiell	$2^{\Theta(k)}$	▲ Thm. 3.8	▼ Prop. 3.9
NORMAL ASP	übereinfach exp.	$2^{\Theta(k \cdot \log(k))}$	▲ Thm. 3.16	▼ **Thm. 3**
ι-TIGHT ASP	übereinfach exp.	$2^{\Theta(k \cdot \log(\iota))}$	▲ Thm. 4.27	▼ Corr. 4.28
ASP	doppelt exponentiell	$2^{2^{\Theta(k)}}$	△[JPW09]	▼ **Thm. 2**
ℓ-QSAT, Quantorenrang ℓ	ℓ-fach exponentiell	$\mathsf{tower}(\ell, \Theta(k))$	△[Ch04]	▼ **Thm. 1**

Tab. 1: Auszug einiger Schlüssellaufzeitresultate der Arbeit, bestehend aus oberen Schranken sowie dazu passenden Härteergebnissen (untere Schranken) für Probleme parametrisiert mit Baumweite. Ideen fett gedruckter Aussagen werden in dieser Fassung skizziert. Die Spalte „Laufzeit*“ berücksichtigt keine Faktoren, die *polynomiell* ($\mathrm{poly}(n)$) in der Instanzgröße n sind. *Die Funktion* $\mathsf{tower}(\ell, k)$ ist ein Turm von Exponenten von 2 der Höhe ℓ mit k an der Spitze. Bekannte obere Schranken sind durch „△“ markiert, während neue Ergebnisse durch „▲“ markiert sind. Untere Schranken nehmen die EZH an; neue Ergebnisse sind durch „▼“, bekannte Resultate durch „▽“ markiert.

chend Kapitel 7 einen Überblick über Ideen zum praktischen Lösen von Problemen mit Baumweite, sodass trotz theoretischer Schranken beachtliche Ergebnisse möglich sind.

2 Grundlagen

Eine *Formel* F ist eine Konjunktion von *Klauseln*, die Disjunktionen von Variablen oder deren Negation sind, vgl. [Bi09]. Der *Quantorenrang* einer *quantifizierten Bool'schen Formel (QBF)* $Q = \exists V_1 . \forall V_2 . \cdots \exists V_\ell . F$ ist die Anzahl ℓ alternierender Quantoren [KL99].

Ein *Antwortmengen (ASP) Programm* [BET11] ist eine Menge an Regeln der Form $H \leftarrow B^+, B^-$ über Variablen-Mengen H, B^+ und B^- mit der intuitiven Bedeutung, dass, wenn alle Variablen in B^+, aber keine Variablen in B^-, hergeleitet werden können, eine Variable in H hergeleitet werden muss. Ein Programm ist *normal*, wenn $|H| = 1$, und „*tight*“, wenn es keine zyklischen Abhängigkeiten über alle Regeln zwischen Variablen in B^+ und H gibt.

Für Formel, QBF oder Program $\mathcal{U}$, gibt $\mathsf{var}(\mathcal{U})$ die *Menge ihrer Variablen* an. Der *Primalgraph* $\mathcal{G}_\mathcal{U}$ von Formel, QBF oder Programm $\mathcal{U}$ hat als Knoten $\mathsf{var}(\mathcal{U})$ mit einer Kante zwischen je zwei Variablen, die in einer gemeinsamen Klausel oder Regel vorkommen.

Eine *Baumzerlegung* $\mathcal{T} = (T, \chi)$ eines Graphen $\mathcal{G}$ besteht aus Baum T und Funktion χ, die jedem Knoten t in T eine Menge an Knoten von $\mathcal{G}$ zuordnet [RS86]. Es muss gelten (i) *Knoten abgedeckt:* für jeden Knoten v von $\mathcal{G}$ gibt es einen Knoten t in T, sodass $v \in \chi(t)$; (ii) *Kanten abgedeckt:* für jede Kante e von $\mathcal{G}$ gibt es einen Knoten t in T, sodass $e \subseteq \chi(t)$; (iii) *Verbundenheit:* wenn für je drei Knoten t_1, t_2, t_3 von T, t_2 auf dem eindeutigen Pfad von t_1 nach t_3 liegt, dann $\chi(t_1) \cap \chi(t_3) \subseteq \chi(t_2)$. Die *Weite* von $\mathcal{T}$ ist der größte Wert $|\chi(t)|$ aller Knoten t in T. Die *Baumweite* von $\mathcal{G}$ ist die kleinste Weite aller Baumzerlegungen von $\mathcal{G}$.

3 „Decomposition-guided“ (DG) Reduktionen

Für eine Vielzahl von Problemen in der KI gibt es Lösungsmethoden, die auf der Ausnutzung von Baumweite basieren, was anhand spezialisierter Implementierungen, z.B. [CW19; FHZ19; KP18], aber auch genereller frameworks [BB19; Bl16; La12] festgestellt werden kann. Gerade bei dem kanonischen Zählproblem für SAT konnte kürzlich bei der „Model Counting Competition 2021“ ein Solver gewinnen [KJ21], der Baumweite verwendet.

Motiviert durch diese Ausnutzung von Baumweite zur Problemlösung, stellt sich die Frage, wie man Instanzen zwischen unterschiedlichen Formalismen konvertieren (umwandeln) kann, sodass strukturelle Eigenschaften in der Form von Baumweite weitestgehend erhalten bleiben. Solche Reduktionen haben theoretische Relevanz aber auch praktische Anwendung. Ein Beispiel wäre die Verwendung von SAT-Solvern, indem man eine zu lösende Instanz eines Problems der KI derart in eine aussagenlogische Formel kodiert, sodass die Baumweite der Instanz linear erhalten bleibt oder zumindest nur sparsam erhöht wird. Diese Frage wird mit Hilfe von speziellen Reduktionen, die *„decomposition-guided“ (DG)*, also zerlegungsangeleitet, sind. Die Idee dieser Reduktionen ist inspiriert durch dynamische Programmierung, die Probleme mittels Traversion der Baumzerlegung von den Blättern bis zur Wurzel des Baumes in Teilen löst. Dabei werden Probleme in Teile reduziert, um präzise Baumweite-Garantien der resultierenden Instanz zu erhalten.

Das Konzept von DG-Reduktionen ist in Abbildung 1 vereinfacht dargestellt, wobei eine Instanz $\mathcal{I}$ eines Quellproblems $\mathtt{P}$ als auch eine Baumzerlegung $\mathcal{T}$ vom Graphen $\mathcal{G}_\mathcal{I}$ angenommen wird. Diese Art der Reduktion hat den Vorteil, dass sie per Konstruktion bereits neben einer Instanz des Zielproblems automatisch auch eine Baumzerlegung liefert. Des Weiteren lässt sich damit sofort eine bestimmte Weiteabhängigkeit bzw. im Idealfall sogar Baumweiteabhängigkeit des Zielproblems vom Quellproblem zeigen.

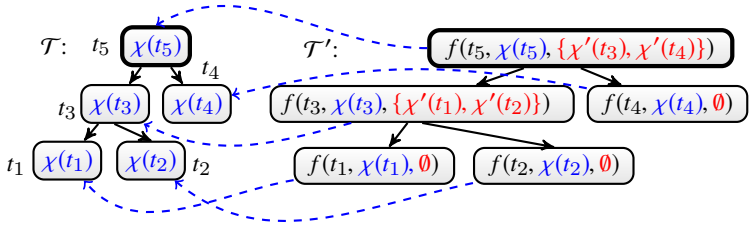

Abb. 1: Vereinfachte Darstellung einer DG-Reduktion von einem Quellproblem $\mathtt{P}$ nach einem Zielproblem $\mathtt{P}'$. Dabei nehmen wir an, dass eine Instanz $\mathcal{I}$ des Problems $\mathtt{P}$ sowie eine Baumzerlegung $\mathcal{T} = (T, \chi)$ von $\mathcal{G}_\mathcal{I}$ gegeben ist. Die Reduktion hängt sowohl von $\mathcal{I}$ als auch von der Zerlegung $\mathcal{T}$ ab und ist daher „zerlegungsangeleitet“. Sie wird für jeden Knoten t von T konstruiert und liefert auch eine Baumzerlegung $\mathcal{T}' = (T, \chi')$ von $\mathcal{G}_{\mathcal{I}'}$ der konstruierten Instanz $\mathcal{I}'$ des Problems $\mathtt{P}'$. Des Weiteren hängt $\chi'(t)$ funktional von $\chi(t)$ sowie $\chi'(t')$ aller Kindknoten t' von t ab (siehe f).

Ansatz für 2-QSAT [LM17]	**Neuer Ansatz für ℓ-QSAT**
(SAT, k)	(SAT, k)
$\downarrow$	$\downarrow$**DG-Reduktion**
(2-QSAT, $\log(n)$)	(2-QSAT, $\log(k)$)
(SAT, k)	(ℓ-QSAT, k)
$\downarrow$**?**	$\downarrow$**DG-Reduktion**
(3-QSAT, $\log(\log(n))$	((ℓ+1)-QSAT, $\log(k)$)

Tab. 2: Bestehende Beweisansätze unterer Schranken sind ähnlich zur linken Spalte, wo die EZH durch Reduktion von SAT angewendet wird. Der Parameter (Baumweite) k der Formel wird nicht direkt verwendet; der Parameter der Zielinstanz ist eine Funktion in der Variablenanzahl n der SAT Formel. Unser Ansatz ist in der rechten Spalte zu sehen, der direkt die Baumweite k der Formel via DG-Reduktion exponentiell verkleinert. Dies lässt sich daher auf Quantorenrang ℓ generalisieren.

4 Neues Meta-Werkzeug via unterer Schranken für QBFs

Für das Zeigen oberer Schranken lässt sich oft eine DG-Reduktion nach QSAT, die die Baumweite linear erhält, verwenden, um die selbe obere Schranke wie jene für QBFs, zu erhalten [Ch04]. Alternativ gibt es auch Meta-Theoreme, mit dessen Hilfe man zumindest die Existenz eines effizienten, parametrisierten Algorithmus nachweisen kann [Co90], ohne evtl. exakte obere Schranken zu erhalten. Die Literatur kennt auch bereits Einzelergebnisse für untere Schranken; darunter befinden sich auch welche für Probleme höherer Stufen der Polynomiellen Hierarchie [LM17; MM16]. Bemerkenswerterweise gibt es allerdings noch *kein Meta-Werkzeug*, das es ermöglicht, auf einfache Art und Weise untere Schranken zu zeigen. Wir adressieren diesen Mangel und stellen mit Hilfe von DG-Reduktionen und folgender präziser unterer Schranken für QSAT solch ein Werkzeug zur Verfügung.

Theorem 1 (Schranke von ℓ-QSAT) *Gegeben sei eine QBF der Form Q mit Quantorenrang ℓ, sodass die Baumweite von $\mathcal{G}_Q$ gleich k ist. Dann kann unter der Annahme der EZH die Validität von Q nicht in der Zeit* $\mathsf{tower}(\ell, o(k)) \cdot 2^{o(|\mathsf{var}(Q)|)}$ *entschieden werden.*

Der Beweisansatz von Theorem 1 ist innovativ, weil er eine DG Selbstreduktion von ℓ-QSAT auf $(\ell + 1)$-QSAT durchführt. Entgegen bestehender Beweise unterer Schranken, wird der zusätzliche Quantorenrang verwendet, um konstruktiv und präzise Baumweite exponentiell zu verkleinern. Dies ist in Tabelle 2 kurz skizziert und führt zu unteren Schranken für eine Vielzahl an Problemen der KI via DG-Reduktionen von ℓ-QSAT [He21, Tabelle 6.1].

5 Sind normale ASP Programme „härter“ als SAT?

Unter Anwendung einer DG-Reduktion von 2-QSAT und Theorem 1 kann man die folgende untere Schranke für ASP zeigen, die die bestehende obere Schranke komplettiert [JPW09].

Theorem 2 (Schranke allgemeiner Programme) *Sei ein beliebiges Programm Π gegeben, sodass die Baumweite von $\mathcal{G}_\Pi$ gleich k ist. Dann kann unter Annahme der EZH die Konsistenz von Π nicht in der Zeit* $2^{2^{o(k)}} \cdot \mathrm{poly}(|\mathsf{var}(\Pi)|)$ *entschieden werden.*

Interessanterweise ergeben sich beim Beweis von Theorem 2 keine Hürden. Das Ergebnis deckt sich auch mit den Erwartungen aus der klassischen Komplexitätstheorie, da sich das Problem auf der zweiten Stufe der Polynomiellen Hierarchie befindet [BET11].

Im Vergleich dazu sieht es bei dem Fragment der *normalen Programme* anders aus: Sowohl SAT als auch die Konsistenz normaler Programme sind NP-vollständige Probleme. Allerdings ist bekannt, dass man im Allgemeinen solche Programme nicht in eine logische Formel kodieren kann, sodass die Antwortmengen exakt über die Modelle der Formel repräsentiert werden, ohne einen subquadratischen Mehraufwand in der Anzahl der Variablen in Kauf zu nehmen [Ja06; LR06]. Nichtsdestotrotz bleibt die *Frage offen*: Ist es im Vergleich zu SAT „härter", die Konsistenz von normalen Programmen für Baumweite zu entscheiden?

Zu einem gewissen Grad lässt sich diese Frage tatsächlich affimieren, allerdings ist der zugehörige Beweis aufwändiger als Theorem 2 und die Konsequenzen weitreichender.

Theorem 3 (Schranke normaler Programme) *Sei ein beliebiges normales Programm* Π *gegeben, sodass die Baumweite von* $\mathcal{G}_\Pi$ *gleich* k *ist. Dann kann unter Annahme der EZH die Konsistenz von* Π *nicht in der Zeit* $2^{o(k \cdot \log(k))} \cdot \mathrm{poly}(|\mathsf{var}(\Pi)|)$ *entschieden werden.*

Bemerkenswerterweise betrifft hier der Mehraufwand nicht bloß die Anzahl der Variablen, sondern unter EZH ist dieses Problem für Baumweite tatsächlich aufwändiger zu lösen, als SAT. Eine informelle Begründung liegt darin, dass man auch mit kleiner Baumweite weitreichende Erreichbarkeitsprobleme oder auch transitive Abschlüsse formulieren kann, sodass die beteiligten Variablen über die gesamte Baumzerlegung verteilt sind.

Diese Beobachtungen führen zu einer Programmfamilie, die als ι*-tight* bezeichnet wird, wobei ι den Grad zwischen „tight" ($\iota = 1$) und „normal" ($\iota = k$ bei Baumweite k) angibt und daher direkter den zugrundeliegenden Lösungsaufwand widerspiegelt, vgl. Tabelle 1.

6 Ausnutzung von Baumweite trotz theoretischer Schranken

Trotz der starken unteren Schranken dieser Arbeit, ist es möglich, Solver zu entwickeln, die sogar Instanzen höherer Baumweite lösen können. Dafür analysieren wir SAT-verwandte Zählprobleme [SS10], die für quantitatives Schließen immer mehr an Bedeutung gewinnen. Die Schranken von Tabelle 1 gelten auch für diese Zählprobleme, vgl. [He21, Tabelle 6.1].

Unser Ansatz zum effizienten Ausnutzen von Baumweite liegt in der Vereinigung verschiedener Konzepte. (1) **Abstraktionen:** Wir berechnen bestimmte Abstraktionen des Primalgraphs, sodass der Lösungsprozess durch eine Baumzerlegung dieser Abstraktionen und dynamischer Programmierung angeleitet wird. (2) **Hybrides Lösen**: Handhabbare Subprobleme, die beim Lösen anhand der Abstraktion auftreten, werden an effiziente SAT-Solver übergeben, was unseren Ansatz in einen Hybriden zwischen dynamischer Programmierung und bestehenden SAT-Solvern verwandelt. (3) **Verfeinerung**: Nicht handhabbare Subprobleme sorgen für eine Verfeinerung der Abstraktionen, sodass wieder (rekursiv) dynamische Programmierung ausgeführt wird, bis gewisse Abbruchkriterien erreicht sind.

Die empirischen Ergebnisse liefern folgende Hauptbeobachtungen. Zuerst ist es bemerkenswert, dass wir Instanzen mit Baumzerlegungen lösen können, deren Weiten sogar 260 übersteigen. Der Grund hierfür liegt darin, dass der hybride Ansatz versucht, stark unstrukturierte Teile des Primalgraphs bevorzugt an bestehende SAT-Solver zu übergeben. Weiters kann man beobachten, dass für ein aufwändigeres Zählproblem, das eine doppelt exponentielle Laufzeitabhängigkeit in der Baumweite inne hat, im Vergleich nur Baumzerlegungen bis zu einer Weite von 99 gelöst werden konnten. Zwar ist dies beachtlich, zeigt jedoch auch die praktische Relevanz der Untersuchung unterschiedlicher Baumweite-Abhängigkeiten.

7 Ausblick

Diese Arbeit führt zu weiteren Forschungsfragen, die sich vor allem am steigenden Interesse von Baumweite widerspiegeln[4]. DG-Reduktionen dienen als Werkzeug für untere und obere Schranken. Allerdings sind deren Stärken, Schwächen sowie Beschränkungen oder mögliche Erweiterungen weitestgehend unerforscht. Im Bereich erklärbarer KI können sie zu beweisbaren Solver-Durchläufen für SAT-Erweiterungen, wie z.B. Zählproblemen, beitragen. Aktuell liefert Theorem 2 ein Werkzeug für untere Schranken von Baumweite, was wir aktuell für *allgemeinere Parameter* generalisieren. Jüngere Arbeiten generalisieren und erweitern dieses Ergebnis für „Constraint Programming“, was andere Ausdrücke unterer Schranken ermöglicht [FHK20], die sich aber bestimmt weiter verallgemeinern und in der Datenbanktheorie anwenden lassen. Existierende Erweiterungen der EZH erlauben eventuell noch genauere untere Schranken zu zeigen. In Anbetracht theoretischer SAT Modelle und gezeigter Zusammenhänge dieser Modelle mit Baumweite, vgl. [AFT11], erwarten wir weitere Konsequenzen der unteren Schranke von Theorem 3 in Bezug auf das interne Ausnutzen struktureller Eigenschaften von ASP-Solvern. Aktuelle Fortschritte von Zählproblemen vereinen Baumweite mit SAT-basierten Techniken, wobei Baumweite als Heuristik in den Solver kodiert wird [KJ21]. Wir sehen dabei Synergiepotenzial mit Sektion 6.

Literatur

[AFT11] Atserias, A.; Fichte, J. K.; Thurley, M.: Clause-Learning Algorithms with Many Restarts and Bounded-Width Resolution. Journal of Artificial Intelligence Research 40/, S. 353–373, 2011.

[AO14] Atserias, A.; Oliva, S.: Bounded-width QBF is PSPACE-complete. Journal of Computer and System Sciences 80/7, S. 1415–1429, 2014.

[BB19] Bannach, M.; Berndt, S.: Practical Access to Dynamic Programming on Tree Decompositions. Algorithms 12/8, S. 172, 2019.

[BET11] Brewka, G.; Eiter, T.; Truszczyński, M.: Answer set programming at a glance. Communications of the ACM 54/12, S. 92–103, 2011.

[4] „Treewidth“ liefert mehr als 21.100 Ergebnisse auf Google Scholar (Februar 2022).

[BHW21] Besin, V.; Hecher, M.; Woltran, S.: Utilizing Treewidth for Quantitative Reasoning on Epistemic Logic Programs. Theory and Practice of Logic Programming 19/5-6, S. 891–907, 2021.

[Bi09] Biere, A.; Heule, M.; van Maaren, H.; Walsh, T., Hrsg.: Handbook of Satisfiability. IOS Press, 2009.

[Bl16] Bliem, B.; Charwat, G.; Hecher, M.; Woltran, S.: D-FLAT^2: Subset Minimization in Dynamic Programming on Tree Decompositions Made Easy. Fundamenta Informaticae 147/1, S. 27–61, 2016.

[Ch04] Chen, H.: Quantified Constraint Satisfaction and Bounded Treewidth. In: ECAI'04. IOS Press, S. 161–165, 2004, ISBN: 1-58603-452-9.

[Co71] Cook, S. A.: The Complexity of Theorem-Proving Procedures. In: STOC'71. ACM, S. 151–158, 1971.

[Co90] Courcelle, B.: Graph Rewriting: An Algebraic and Logic Approach. In: Handbook of Theoretical Computer Science, Vol. B. Elsevier, S. 193–242, 1990.

[CW19] Charwat, G.; Woltran, S.: Expansion-based QBF Solving on Tree Decompositions. Fundamenta Informaticae 167/1-2, S. 59–92, 2019.

[FH21] Fandinno, J.; Hecher, M.: Treewidth-Aware Complexity in ASP: Not all Positive Cycles are Equally Hard. In: AAAI'21. S. 6312–6320, 2021.

[FHK20] Fichte, J. K.; Hecher, M.; Kieler, M. F. I.: Treewidth-Aware Quantifier Elimination and Expansion for QCSP. In: CP'20. Bd. 12333. LNCS, Springer, S. 248–266, 2020.

[FHP20] Fichte, J. K.; Hecher, M.; Pfandler, A.: Lower Bounds for QBFs of Bounded Treewidth. In: LICS'20. ACM, S. 410–424, 2020.

[FHZ19] Fichte, J. K.; Hecher, M.; Zisser, M.: An Improved GPU-Based SAT Model Counter. In: CP'19. Bd. 11802. LNCS, Springer, S. 491–509, 2019.

[Fi22] Fichte, J. K.; Hecher, M.; Thier, P.; Woltran, S.: Exploiting Database Management Systems and Treewidth for Counting. Theory and Practice of Logic Programming 22/1, S. 128–157, 2022.

[He21] Hecher, M.: Advanced Tools and Methods for Treewidth-Based Problem Solving, Doktorarbeit, Universität Potsdam, 2021, S. 184.

[He22] Hecher, M.: Treewidth-aware reductions of normal ASP to SAT - Is normal ASP harder than SAT after all? Artificial Intelligence 304/, S. 103651, 2022.

[IPZ01] Impagliazzo, R.; Paturi, R.; Zane, F.: Which Problems Have Strongly Exponential Complexity? Journal of Computer and System Sciences 63/4, S. 512–530, 2001.

[Ja06] Janhunen, T.: Some (in)translatability results for normal logic programs and propositional theories. Journal of Applied Non-Classical Logics 16/1-2, S. 35–86, 2006.

[JPW09] Jakl, M.; Pichler, R.; Woltran, S.: Answer-Set Programming with Bounded Treewidth. In: IJCAI'09. Bd. 2, S. 816–822, 2009.

[KJ21] Korhonen, T.; Järvisalo, M.: Integrating Tree Decompositions into Decision Heuristics of Propositional Model Counters (Short Paper). In: CP'21. Bd. 210. LIPIcs, Dagstuhl Publishing, 8:1–8:11, 2021.

[KL99] Kleine Büning, H.; Lettman, T.: Propositional Logic: Deduction and Algorithms. Cambridge University Press, 1999.

[KP18] Kiljan, K.; Pilipczuk, M.: Experimental Evaluation of Parameterized Algorithms for Feedback Vertex Set. In: SEA. Bd. 103. LIPIcs, Dagstuhl Publishing, 12:1–12:12, 2018.

[La12] Langer, A.; Reidl, F.; Rossmanith, P.; Sikdar, S.: Evaluation of an MSO-Solver. In: ALENEX'12. SIAM / Omnipress, S. 55–63, 2012.

[LM17] Lampis, M.; Mitsou, V.: Treewidth with a Quantifier Alternation Revisited. In: IPEC'17. Bd. 89. LIPIcs, Dagstuhl Publishing, 26:1–26:12, 2017.

[LR06] Lifschitz, V.; Razborov, A. A.: Why are there so many loop formulas? ACM Transactions on Computational Logic 7/2, S. 261–268, 2006.

[MM16] Marx, D.; Mitsou, V.: Double-Exponential and Triple-Exponential Bounds for Choosability Problems Parameterized by Treewidth. In: ICALP'16. Bd. 55. LIPIcs, Dagstuhl Publishing, 28:1–28:15, 2016.

[PV06] Pan, G.; Vardi, M. Y.: Fixed-Parameter Hierarchies inside PSPACE. In: LICS'06. IEEE Computer Society, S. 27–36, 2006.

[RS86] Robertson, N.; Seymour, P. D.: Graph Minors. II. Algorithmic Aspects of Tree-Width. Journal of Algorithms 7/3, S. 309–322, 1986.

[SS10] Samer, M.; Szeider, S.: Algorithms for propositional model counting. Journal of Discrete Algorithms 8/1, S. 50–64, 2010.

Markus Hecher wurde am 11. Mai 1990 geboren und hat seine Masterarbeit, die mit dem Würdigungspreis der Stadt Wien ausgezeichnet wurde, an der TU Wien abgeschlossen. Für einen Hauptteil seiner Doktorarbeit wurde er mit dem „Marco Cadoli Best Student Paper Award“ ausgezeichnet [He22], andere Teile sind auch publiziert, z.B. [FH21; FHP20]. Eine beteiligte Arbeit wurde mit „Among Best Papers“ zu einer Journal-Version eingeladen [Fi22]. Eine Folgearbeit wurde mit „Best Paper“ als auch ”Best Student Paper“ ausgezeichnet [BHW21]. 2021 wurde Markus an die UC Berkeley eingeladen, um ein Semester lang an einem spezialisierten SAT-Programm teilzunehmen. Für die binationale Doktorarbeit wurde Markus mit dem „Award of Excellence 2021“ des BMBWF Österreich ausgezeichnet. Er war bereits zwei Mal Mitgewinner bei wissenschaftlichen Wettbewerben. Co-Organisierte Events und detailliertere Daten sind verfügbar unter `https://dbai.tuwien.ac.at/staff/hecher/`.