Universität Potsdam

Naturwissenschaftliche Fakultät

Informatik

Diplom

Untersuchung des Berechnungsaufwands bei Antwortmengenprogrammierung

Betti Österholz

BioKom@gmx.de

1. Dezember 2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Antwortmengenprogrammierung es ein relativ neues, nichts desto trotz erfolgreiches Programmierparadigma. Die Antwortmengenprogrammierung ist aber leider NP-vollständig, das heißt, der Berechnungsaufwand steigt im allgemeinen exponentiell.

Was bedeutet dies aber konkret für die Berechnung von Antwortmengen?

Stößt man, wenn mit Antwortmengenprogrammierung gearbeitet wird, irgendwann an eine Schwelle, an der man wegen des exponentiellen Wachstums nicht mehr weiter kommt?

Gibt es vielleicht Unterklassen in der Antwortmengenprogrammierung die leichter zu lösen sind?

Wie kann Wissen darüber, welche Antwortmengenklassen wie schwer zu berechnen sind, dabei helfen, existierende Antwortmengensolver schneller zu machen?

Diese und ähnliche Fragen will ich mit dieser Arbeit untersuchen.

Mir geht es im vorliegendem Abhandlung um die Anwendung von Antwortmengenprogrammierung und weniger um theoretische Ergebnisse.

1.1 Lesart

Diese Arbeit ist in unterschiedliche Kapitel gegliedert. Es ist für das Verständniss nicht unbedingt erforderlich alle Kapitel zu lesen, Personen mit entsprechenden Vorwissen können Kapitel 2 oder/und 3 überspringen.

Kapitel 1 enthält die Einleitung.

In Kapitel 2 werden die Grundlagen von Antwortmengenprogrammierung, soweit sie für diese Arbeit relevant sind, behandelt.

In Kapitel 3 geht es um Phasenübergänge. Es enthält auch die Zusammenfassungen von Arbeiten über Phasenübergänge und Berechnungsaufwand aus dem Bereichen von SAT und Antwortmengenprogrammierung.

In Kapitel 4 sind die Modelle beschrieben, die zur Generierung der Testreihen verwendet wurden.

Kapitel 5 behandelt theoretischen Untersuchungen. In ihm werden Überlegungen über Berechnungsaufwand, Erfüllbarkeit und deren zusammenhängen bei Antwortmengenprogrammen gemacht. Es werden verschiedene Antwortsuchmodelle aufgestellt und betrachtet. Außerdem werden Vermutungen angestellt, über die Auswirkungen von Veränderungen bei bestimmten Parametern.

Kapitel 6 behandelt die Ergebnisse meiner praktischen Untersuchungen. In ihm werden die Ergebnisse der gemachten Testreihen vorgestellt und Vermutungen über die Gründe des gezeigten Verhaltens angestellt, beziehungsweise mit den in Kapitel 5 gemachten Aussagen in Verbindung gebracht.

Im letzten Kapitel 7 werden die Ergebnisse noch einmal in aller Kürze zusammengefasst und ein Ausblick auf mögliche zukünftige Arbeiten gegeben.

Die meiste Antwortmengenliteratur liegt in englischer Sprache vor, um nicht den Leser mit Übersetztungen und dadurch ungenaueren Begriffen zu verwirren, werden Begriffen aus der Antwortmengenprogrammierung in englisch benutzt. Wenn Begriffe in ihrer Bedeutung verwendet werden, für die es aber ein sehr gebräuchlichen englischen Begriff gibt, wird dieser beim ersten Gebrauch hinter dem deutschen Begriff in Klammern angegeben. Bei aus anderen Dokumenten übernommenen Diagrammen, wurde es bei der originalen Beschriftung belassen, um nicht durch Übersetzungsungenauigkeiten Fehler einzubauen.

Programmteile im Text sind im `Schreibmaschinenstiel` geschrieben.

Bei Zahlen wurde die amerikanische Schreibweise mit „.“ als Dezimaltrennzeichen verwendet, da die meisten Diagramme mit dem Tool gnuplot generiert wurden und die Schreibweise der Zahlen von ihm übernommen wurde.

1.2 Symbolverzeichnis

P	ein Antwortmengenprogramm
A	Anzahl der Atome
A_M	Menge der Atome
A_i;a	ein Atom
R	Anzahl der Regeln
r	eine Regel
r_i	i'te Regel des Programms
N	Anzahl der Variablen oder Atome bei SAT
L	Anzahl der Regeln bei SAT
k_i	Anzahl der Atome im Körper einer Regel r_i
Π	grounded Programm
I	Interpretation
X	Teilbelegung der Atome
AW_i	eine Antwortmenge
G	Abhängigkeitsgraph
G_p	positiver Abhängigkeitsgraph
G_{bh}	body head Graph

Kapitel 2

Grundlagen der Antwortmengenprogrammierung

2.1 Grundlagen Logik

2.1.1 Normal logic Programm

Antwortmengenprogrammierung ist ein Paradigma, das auf verschiedenen logischen Programmen arbeiten kann. Im folgenden werden zwei logische Programmarten vorgestellt.

A_M ist ein Satz von propositionalen Symbolen, seine Elemente A_i sind Atome.
Ein Regel r für ein normal logisches Programm, ist ein Tupel der Form:

$$A_0 \leftarrow A_1, ..., A_m, not\ A_{(m+1)}, ..., not\ A_k$$

Dabei ist $k \geq m \geq 0$ und alle A_i $(0 \leq i \leq k)$ sind Atome. A_i ist ein positives Literal und $not\ A_i$ ein negatives Literal.

Ein normal logic Programm ist ein endlicher Satz von normal Regeln.

Notation:

$$head(r) = A_0 \quad (2.1)$$

$$body(r) = \{A_1, \ldots, A_m, not\ A_{(m+1)}, \ldots, not\ A_k\} \quad (2.2)$$

$$body^+(r) = \{A_1, \ldots, A_m\} \quad (2.3)$$

$$body^-(r) = \{A_{(m+1)}, \ldots, A_k\} \quad (2.4)$$

$head(r)$ ist dabei der Kopf der Regel r und $body(r)$ ist der Körper. In $body^+(r)$ sind alle Atome die positiv im Körper vorkommen enthalten oder der positive Teil des Körpers. Dagegen sind in $body^-(r)$ alle Atome die negiert im Körper vorkommen, enthalten oder der negative Teil des Körpers. Ein normal logic Programm ist, wenn

$body^{-}(r_i) = \emptyset$ für alle Regeln r_i $1 \leq i \leq R$ des Programms ist, ein (definite) basic logic Programm.
Ein Fakt r ist eine Regel mit $body(r) = \emptyset$.

Ein constrain (Beschränkung) ist eine Regel, die niemals erfüllbar sein soll, um damit bestimmte Teilwahrheitswertbelegungen auszuschließen. Wenn F ein Atom ist, das in keiner anderen Regel vorkommt, ist $F \leftarrow not\ F, A_1, \ldots, A_m, not\ A_{(m+1)}, \ldots, not\ A_k$ ein constrain. Dieses wird häufig als $\leftarrow A_1, \ldots, A_m, not\ A_{(m+1)}, \ldots, not\ A_k$ geschrieben.

2.1.2 Interpretationen

Wahrheitswerte sind die Werte Wahr (true) und Falsch (false).

Eine Interpretation I ist eine Belegung aller Atome eines Antwortmengenprogramms mit Wahrheitswerten.

Eine partielle Interpretation ist eine Belegung eines Teils der Atome eines Antwortmengenprogramms mit Wahrheitswerten.

Bei einer 3 Werte Interpretation (3-valued Interpretation) kann jedes Atom 3 verschiedene Werte einnehmen Wahr, Falsch und Unbekannt (unknown). Geschrieben mit $\langle T, F \rangle$, wobei die Menge T die Atome enthält die Wahr sind, F die Falsch sind und alle Atome die nicht in einer der beiden enthalten sind, Unbekannt sind. Eine 3 Werte Interpretation ist total, wenn $T \cup F = A_M$ ist, ansonsten ist sie partial. Mit 3-wertigen Interpretationen arbeitet die 3-wertige Logik.

Ein Modell, ist eine Interpretation für ein Programm P, welches P erfüllt oder Wahr macht. M ist ein minimales Modell von P, wenn es für P kein Modell M_2 gibt mit $M_2 \subset M$.

2.2 SAT

Forschungen rund um das SAT Gebiet existiert schon länger als die Forschungen zum Feld Antwortmengenprogrammierung. Bei SAT Problemen, geht es darum, Modelle einer logischen Formel zu finden, beziehungsweise herauszufinden, ob eine Formel überhaupt erfüllbar (SATisfiable) ist. SAT Probleme gehören zu der Klasse der NP-harten Probleme.

NP-harte Probleme haben die Wissenschaft schon die letzten 35 Jahre beschäftigt [7]. Mit dem vorläufigen Ergebnis, dass es anscheinend keinen Algorithmus gibt, der garantieren kann NP-harte Probleme mit polynomiellen Berechnungsaufwand oder auch nur schneller als exponentiell zu lösen.

Viele NP-harte Problemklassen besitzen jedoch Unterklassen die polynomiell zu lösen sind, ein Beispiel dafür ist z.B. 2-SAT und Horn-SAT [7].

Bei Texten zu SAT ist zu beachten, dass dort, anders als bei der Antwortmengenprogrammierung, Atome als Variablen bezeichnet werden, da Atome in ihrer Wahrheitswertbelegung variable sind. Bei der Antwortmengenprogrammierung wird dies nicht gemacht, da sonst der Begriff Variable überladen wäre.

2.3 Antwortmengen

Eine Antwortmenge eines Programms P, ist ein minimales Modell von P. Antwortmengen zu finden oder zu sagen, dass es keine gibt, ist der Forschungsgegenstand der Antwortmengenprogrammierung oder kurz ASP (Answer Set Programming). ASP Probleme gehören zu der Klasse der NP-harten Probleme.

2.3.1 Ground Programme

Ein normales Antwortmengenprogramm P enthält im allgemeinen eine Zahl von Variablen. Beim so genannten grounding werden alle Variablen des Antwortmengenprogramms P ersetzt, durch die Atome beziehungsweise Instanzen, die sie annehmen können. Dadurch entsteht ein ground logisches Programm Π ohne Variablen.

2.3.2 Gelfound-Lifschitz Transformation

Das reduct oder die Gelfound-Lifschitz Transformation Π^X eines Antwortmengenprogramms P bezüglich einer Menge von Atomen (Interpretation) X ist das Programm, welches entsteht, wenn alle Regeln, in denen Atome aus X negiert vorkommen, gelöscht werden und vom daraus resultierenden Programm alle negierten Atome aus dem Körper gelöscht werden.

$$\Pi^X = \{head(r) \leftarrow body^+(r) | r \in \Pi \wedge body^-(r) \cup X = \emptyset\ \}$$

Das entstehende Antwortmengenprogramm Π^X enthält keine negierten Literale mehr. Daraus folgt, dass es immer ein Modell zum entstandenen Programm Π^X gibt, da es keine Widersprüche in Π^X mehr geben kann.

2.3.3 C_n Operator

Der C_n Operator ist ein Fixpunktoperator.

Π ist ein basic Programm und X eine Menge von Atomen die direkte Folge T_Π von X bezüglich Π ist:

$$T_\Pi(X) = \{head(r) | r \in \Pi \wedge body(r) \subseteq X\} \tag{2.5}$$
$$T_\Pi^0(X) = X \tag{2.6}$$
$$T_\Pi^i(X) = T_\Pi(T_\Pi^{i-1}(X)) \tag{2.7}$$

$T_\Pi(X)$ ist monoton, da $X \subseteq Y \Rightarrow T_\Pi(X) \subseteq T_\Pi(Y)$

$$\Longrightarrow T_\Pi^{i-1}(X) \subseteq T_\Pi^i(X) \tag{2.8}$$

Der C_n Operator ergibt sich wie folgt:

$$C_n(\Pi) = \bigcup_{i \geq 0} T^i_\Pi(\emptyset) \tag{2.9}$$

Ist $C_n(\Pi^X) = X$ wird X stable (stabiles) Modell von Π und P genannt.

Der C_n Operator ist so zu realisieren, dass er in polynomieller Zeit arbeitet. Da er die R Regeln des Antwortmengenprogramms maximal $min(R; A)$ mal durchgehen muss, denn bei jedem Durchgang muss mindestens ein neues Kopfatom geschlossen werden. Der Berechnungsaufwand pro Regel r_i steigt dabei linear mit der Anzahl ihrer Atome $(k_i + 1)$.
Demnach ist der maximale Berechnungsaufwand B_{max}:

$$B_{max} = (const * (\sum_{i=1}^{R}(k_i + 1)) * min(R; A)) \tag{2.10}$$

2.3.4 Well-founded Semantik

Die well-founded (wohl gegründete) Semantik oder WFS eines Programms ist charakterisiert durch eine 3-wertigen Interpretation, auch als well-founded Modell bezeichnet.

$$\begin{aligned} I_3(\Pi) &= \langle lfp(\Pi), gfp(\Pi)\rangle \\ L &= lowerbound \\ U &= upperbound \end{aligned}$$

$$\begin{aligned} L_0 &= \emptyset && (2.11) \\ U_0 &= A_M && (2.12) \\ L_i &= L_{i-1} \cup C_n(\Pi^{U_{i-1}}) && (2.13) \\ U_i &= U_{i-1} \cup C_n(\Pi^{L_{i-1}}) && (2.14) \\ lfp(\Pi) &= \bigcup_{i \geq 0} L_i && (2.15) \\ gfp(\Pi) &= A_M / \bigcap_{i \geq 0} U_i && (2.16) \end{aligned}$$

lfp (lowest fix point) und gfp (greatest fix point) sind jeweils die kleinsten und größten Fixpunkte. Alle Atome die in $lfp(\Pi)$ enthalten sind, sind auch in allen Antwortmengen zu Π enthalten und alle Atome die in allen Antwortmengen zu Π enthalten sind, sind auch in $gfp(\Pi)$ enthalten. Wird $L_0 = X$ und $U_0 = Y$ gesetzt, so werden alle Antwortmengen AW_i von P generiert, von denen X eine Teilmenge ist ($X \subseteq AW_i$) und die kein Element von Y enthalten ($Y \cap AW_i = \emptyset$).

Ist $lfp(\Pi) = gfp(\Pi)$, ist $lfp(\Pi)$ ein minimales Modell oder eine Antwortmenge von Π und P.

Wenn es ein $x \in lft(\Pi)$ gibt für das aber $x \notin gfp(\Pi)$ gilt, besitzt Π keine Antwortmenge. Beziehungsweise für $L_0 = X$ und $U_0 = Y$ gibt es für Π keine Antwortmengen.

Wenn $lfp(\Pi) \neq gfp(\Pi)$ es aber noch Antwortmengen gibt, können diese gesucht werden, indem für ein Atom a, das nicht in $lfp(\Pi) \cup A_M/gfp(\Pi)$ ist, beide Belegungen ausprobiert werden. Dafür kann es einmal zu den Atomen gepackt werden, die mit $lfp(\Pi)$ als zur Antwortmenge gehörend gefunden wurden, indem $L_0 = (lfp(\Pi) \cup \{a\})$ gesetzt wird. Andrerseits kann es zu den Atomen gepackt werden, die nicht in der Antwortmenge sind, indem $U_0 = ((A_M/gfp(\Pi)) \cup \{a\})$ gesetzt wird.

Das Ausführen der Operatoren kann als Schließen gesehen werden, da von den vorhandenen Informationen auf neue geschlossen wird. Wenn ein noch nicht verwendetes Atom aus $A_M/(lfp(\Pi) \cup A_M/gfp(\Pi))$ zu L_0 oder U_0 gepackt wird, wird an diesem Atom eine Entscheidung (choice) vollführt. Wenn das Atom einmal zur einen und auch separat zur anderen Menge gepackt wird, und die zurückgegebenen Antwortmengen vereinigt werden, werden für beide Belegungen die möglichen Antwortmengen gesucht. Auf diese Weise können mehrere oder auch alle Antwortmengen eines Antwortmengenprogramms gefunden werden.

Da L_i und U_i jeweils mit der Vereinigung ihrer Vorgänger mit einer Menge entstehen, sind sie monoton wachsend. Weiterhin ist die Menge der Atome die $C_n(\Pi^X)$ zurückgeben kann endlich, es können maximal alle Atome des Programms P zurückgegeben werden. Das heißt, dass ab einem bestimmten n $L_i = L_{i-1}$ für $i > n$ ist und für eine m $U_i = U_{i-1}$ für $i > m$ ist. Dann brauchen natürlich keine weiteren L_i und U_i mehr berechnet werden.

Das well-founded Modell kann in quadratischer Zeit berechnet werden([12]).

2.3.5 Abhängigkeitsgraph

Gegeben ist ein logisches Programm P, der Abhängigkeitsgraph G von P, ist der folgende gerichtete Graph: Die Knoten des Graphen sind die möglichen verschiedenen Atome des Programms. Alle Knoten p, q für die es eine Regel in P gibt, in der p der Kopf ist und q positiv im Körper der Regel vorkommt, sind durch eine positive Kante von p nach q verbunden. Wenn q negativ im Körper der Regel vorkommt, sind sie durch eine negative Kante von p nach q verbunden.

2.3.6 Zyklen

Ein negativer/ungerader (odd) Zyklus im Graphen enthält eine ungerade Anzahl von negativen/Einser Kanten. Ein gerader (even) Zyklus enthält eine gerade Anzahl von negativen/Einser Kanten. Normal logic Programme die nur gerade Zyklen besitzen, haben immer ein stabiles (stable) Modell, und diese können in polynomieller Zeit berechnet werden ([12]). Weiterhin wird in [12] gezeigt, dass bei Programmen mit einer begrenzten Anzahl von (nicht trivialen) geraden Zyklen alle Antwortmengen in polynomieller Zeit berechnet werden können. Wenn ein Programm keine solchen Zyklen enthält, besitzt es maximal ein stabiles Modell und damit maximal eine Antwortmenge. Ein Programm mit k (nicht trivialen) geraden Zyklen hat maximal 2^k stabile Modelle.

Im allgemeinen generieren gerade Zyklen Modelle und ungerade Zyklen zerstören Modelle.

2.3.7 Positiver Abhängigkeitsgraph

Gegeben ist ein logisches Programm P, der positive Abhängigkeitsgraph G_p von P, ist der folgende gerichtete Graph: Die Knoten des Graphen sind die möglichen verschiedenen Atome des Programms. Alle Knoten p, q für die es eine Regel in P gibt, in der p der Kopf ist und q positiv im Körper der Regel vorkommt, sind durch eine Kante von p nach q verbunden.

Der positiver Abhängigkeitsgraph von P entspricht also dem Abhängigkeitsgraph von P, aus dem alle negativen Kanten entfernt wurden.

2.3.8 Body head Graph

Der body head (Körper Kopf) Graph ist eine Erweiterung des Abhängigkeitsgraphen, in dem auch die Körper von Regeln berücksichtigt werden.

Gegeben ist ein logisches Programm P, der body head Graph von P, ist ein gerichteter Graph. Die Knoten des Graphen sind die möglichen verschiedenen Atome und die unterschiedlichen Körper des Programms. Alle Kopfknoten p und Körperknoten b für die es eine Regel in P gibt, in der p der Kopf und b der Körper der Regel ist, sind durch eine Kante von p nach b verbunden. Alle Kopfknoten p und Körperknoten b für die es eine Regel in P gibt und p Element $pos(b)$ ist, sind durch eine positive/Nuller Kante von b nach p verbunden. Wenn p Element $neg(b)$ ist, sind sie durch eine negative/Einser Kante von b nach p verbunden.

Dabei enthält die Atommenge $pos(b)$ alle positive und $neg(b)$ alle negativen Literale des Körpers b.

2.3.9 Loopformulars

Wie im Abschnitt 2.3.6 erläutert, sind Zyklen ein wichtiges Konstrukt bei der ASP. Sie bekommen noch eine extra Bedeutung mit dem Zeichen „$\leftarrow$“. Während in der üblichen Logik „$a \leftarrow b$“ die Bedeutung „a wenn b“ hat und damit in „$\neg a \vee b$“ übersetzt werden kann, bedeutet „$a \leftarrow b$“ in ASP „aus b folgt a“, damit kann a nur geschlossen werden, wenn b vorher Wahr ist. Nicht erlaubt ist dabei a zu schließen, wenn b nur Wahr ist, weil a Wahr ist, z.B. a kann nicht aus „$a \leftarrow a$“ geschlossen werden. Dies wird üblicherweise bei der Suche nach Antworten als Zirkelschluss abgelehnt.

Das Ausschließen solcher Zirkelschlüsse ist der einzige Unterschied bei der Antwortmengensuche zur Suche bei SAT. Durch Einfügen sogenannter Loopformulars (Schleifenformeln) können solche Zirkelschlüsse ausgeschlossen werden und so Antwortmengenprobleme direkt mit SAT Solvern gelöst werden, beziehungsweise Antwortmengenprobleme auf SAT-Probleme zurückgeführt werden. Näheres dazu in [14] und [21].

2.3.10 ASP ist nicht monoton

ASP ist nicht in dem Sinne monoton, dass durch Hinzufügen neuer Regeln zu einem Antwortmengenprogramm, sich die Menge seiner Antwortmenge nur durch zuzätzliche Antwortmengen vergrößern kann. Durch Hinzufügen neuer Regeln können sowohl mehr Antwortmengen generiert werden als auch weniger (z.B. durch constrains).

2.4 Antwortmengensuche

Es wird angenommen([12]), dass der Berechnungsaufwand für eine Antwortmengensuche im schlimmsten Fall (worst case) exponentiell mit der Anzahl der Variablen (Atome) wächst. Die Idee, die dahinter steckt ist, dass bei Einführung eines neuen Atoms alle anderen Atome im ungünstigsten Fall für die Möglichkeiten, dass das neue Atom Wahr ist oder nicht, überprüft werden müssen. Dann verdoppelt sich der Berechnungsaufwand mit jedem neuen Atom.

Dieser Fall kann sicherlich konstruiert werden. Aber die Wahrscheinlichkeit, dass eine konkrete Problemklasse oder Problem diesem Fall entspricht, ist äußerst gering. Meist sind Atome von anderen abhängig und es müssen dann nicht beide Fälle, dass es Wahr oder nicht ist, überprüft werden, oder das Antwortmengenprogramm ist schon ohne es unerfüllbar und damit auch mit. Durch ein neues Atom wird sich also normalerweise nicht der Berechnungsaufwand verdoppeln. Es reicht für ein exponentielles Wachstum allerdings aus, dass sich im Durchschnitt der Berechnungsaufwand für ein neues Atom um einen Faktor größer 1 vergrößert.

Unabhängig davon gibt es bei Antwortmengenprogrammen Unterklassen die mit polynomiellen Berechnungsaufwand gelöst werden können, z.B. basic logic Programme.

Die meisten deterministischen Algorithmen zur Suche von Antwortmengen gehen allgemein gesprochen so vor, dass immer zwei Schritte solange wiederholt werden, bis die geforderten Antwortmengen gefunden wurden. Der erste Schritt ist dabei auf die Wahrheitswerte aller Atome zu schließen, bei denen es unter den Voraussetzungen möglich ist. Im zweiten Schritt wird ein noch nicht belegtes Atom ausgewählt, eine Wahrheitswertbelegung an ihm ausprobiert und wenn dies nicht zum Ziel führt, auch die andere Wahrheitswertbelegung ausprobiert. Ausprobiert meint dabei, dass für die entstehende partielle Interpretation geforderten Antwortmengen gesucht werden. Bei diesem Ausprobieren wird das ursprüngliche Problem in zwei kleinere Probleme aufgespaltet. Diese Schritte werden solange wiederholt, bis die geforderten Antwortmengen gefunden wurden oder die Schritte nicht mehr ausführbar sind.

Da bei jedem Aufspalten des Problems, zwar ein Atom weniger zu betrachten ist, dafür aber dann zwei, nur um ein Atom kleinere, Probleme zu betrachten sind, kann diese Antwortmengensuche schnell exponentiellen Berechnungsaufwand erfordern.

2.4.1 Antwortmengensuche durch Aufspalten und Begrenzen mit WFS

Die aufspalten und begrenzen (branch and bound) Strategie ist eine der wohl am weitesten verbreitete.

Die Algorithmen arbeiten auf 3-wertigen Interpretationen. Bei der Initialisierung werden zwei leere Mengen als Mengen für die wahren und falschen Atome übergeben und das Programm gegroundet. Dann werden zwei Schritte solange wiederholt, bis genügend Antwortmengen entstanden sind oder keine mehr erzeugt werden kann. Die Interpretation $I = \langle T, F\rangle$ ist eine Antwortmenge, wenn $T \cup F = A_M$ alle Atome A_M aus P enthält. Die Interpretation I kann keine Antwortmenge mehr werden, wenn der $T \cap F$ nicht leer ist, also ein Atom sowohl Falsch als auch Wahr sein müsste.

Begonnen wird mit der Interpretation $I = \langle \emptyset, \emptyset \rangle$.

Schritt 1: Es wird die well-founded Semantik gebildet, wobei $L_0 = T$ und $U_0 = F$ gesetzt wird. Entsteht dabei eine Antwortmenge, wird diese zurückgegeben, ist keine mehr möglich, wird $\emptyset$ zurückgegeben, ansonsten wird zu Schritt 2 übergegangen.

Schritt 2: Dann wird ein Atom ausgesucht, das in $(A_M/(T \cup F))$ ist. Mit diesem Atom werden 2 neue Interpretationen aus der Interpretation $I = \langle T, F\rangle$ gebildet, wobei bei einer von den neuen Interpretationen das Atom zu T gepackt wird und bei der anderen zu F. Mit diesem beiden Interpretationen wird jeweils Schritt 1 wiederholt, bis genügend Antwortmengen entstanden sind. Dabei muss eventuell Schritt 1 auch nur mit Einer der beiden wiederholt werden.

Die erzeugten Antwortmengen werden zurückgegeben.

Schritt 2 wird dabei choice (Entscheidung) genannt.

Diese Art der Antwortmengensuche kann durch weitere Optimierungen ergänzt werden.

Im Grunde können in Schritt 1 auch andere Operatoren, als das reine Finden der well-founded Semantik, eingesetzt werden. Wichtig dabei ist, dass nur etwas geschlossen wird, das auch logisch geschlossen werden kann.

Damit ergibt sich für solche Systeme, für den Berechnungsaufwand bei der Suche nach allen Antwortmenge, die Formel:

$$P_t(P, I) = c(P, I) + WF_t(P, I) + H_t(P, I^{WF}) + P_t(P, I^+) + P_t(P, I^-) \qquad (2.17)$$

P	das Antwortmengenprogramm bezüglich dem die Antwortmengen gefunden werden sollen
I	eine 3 wertige Anfangsinterpretation, beim Starten des Allgorithmuss ist $I = \{\emptyset, \emptyset\}$
$WF_t(P, I)$	der Berechnungsaufwand der benötigt wird um die well-founded Semantik bezüglich P und I zu berechnen (oder für andere Operatoren in Schritt 1)
$H_t(P, I^{WF})$	der Berechnungsaufwand der für P und I^{WF} in die Heuristik fließt, hier wird auch keine Heuristik (wenn das choice Atom zum Beispiel zufällig gewählt wird) als Heuristik angesehen
I^{WF}	die well-founded Semantik zu der ursprünglichen Interpretation
I^+	die Interpretation in der das durch die Heuristik ausgesuchte Atom positiv vorkommt
I^-	die Interpretation in der das durch die Heuristik ausgesuchte Atom negativ vorkommt
$c(P, I)$	eine Funktion die den sonstigen Berechnungsaufwand für P und I angibt, zum Beispiel der Verwaltungsberechnungsaufwand

Für die Anzahl der choices um alle Antwortmenge zu finden ergibt sich die Formel:

$$P_c(P, I) = P_c(P, I^+) + P_c(P, I^-) + 1 \tag{2.18}$$

$P_t(P, I) = 0$ und $P_c(P, I) = 0$ wenn I eine Antwortmenge oder $T \cap F \neq \emptyset$ ist.

Meistens wird aber nur die erste Antwortmenge gesucht, dann ergeben sich die Formeln:
Für den Berechnungsaufwand:

$$\begin{aligned} P_t(P, I) = c(P, I) + WF_t(P, I) + H_t(P, I^{WF}) + \\ AW_{in}(P, I^+) * P_t(P, I^+) + (1 - AW_{in}(P, I^+)) * P_t(P, I^-) \end{aligned} \tag{2.19}$$

$AW_{in}(P, I^+)$	ist 1 wenn das Antwortmengenprogramm P bezüglich der Interpretation I^+ eine Antwortmenge enthält, sonst ist die Funktion 0

Und für die Anzahl der choices um alle Antwortmenge zu finden:

$$\begin{aligned} P_c(P, I) = (AW_{in}(P, I^+) * P_c(P, I^+)) + \\ ((1 - AW_{in}(P, I^+)) * P_c(P, I^-)) + 1 \end{aligned} \tag{2.20}$$

2.4.2 Systeme

2.4.2.1 Smodels

Smodels ist ein Antwortmengensolver von Patrik Simons, welcher auf einer Erweiterung von normal logic Programmen arbeitet. Er wird ständig weiterentwickelt, unterliegt den GNU public licens und ist damit jedermann frei zugänglich. Das Smodels System besteht aus zwei Komponenten lparse und smodels. Das

Programm lparse nimmt ein Antwortmengenprogramm, bildet die ground-Version, nimmt einige überflüssigen Teile heraus (z.B. doppelte Regeln) und wandelt es in eine für smodels leichter lesbare Form um.

Smodels arbeitet mit dem Aufspalten und Begrenzen mit WFS Ansatz aus Abschnitt 2.4.1 von Seite 11. Zum Einsatz kommen dabei auch einige Heuristiken und Operationen welche die Suche beschleunigen sollen, z.B. das Löschen von überflüssigen Regeln und Literalen.

Für mehr Details sei auf [19] verwiesen.

2.4.2.2 Nomore++

Bei dem im Institut Informatik Lehrstuhl Wissensverarbeitung an der Universität Potsdam entwickelten Antwortmengensolver NoMoRe handelt es sich um einen auf Graphen basierten Ansatz. Ein Solver der auf diesem Ansatz aufbaut, ist der beim gleichen Lehrstuhl entwickelte nomore++ Solver.

Grundlage dafür ist der body head Graph des Antwortmengenprogramms. Über Einfärbungen werden die einzelnen Knoten mit Werten belegt. Dabei gibt es zu den Werten der 3-wertigen Logik Wahr, Falsch und Unbekannt, noch weitere Werte wie wahrscheinlich Wahr. Wenn alle Kopfknoten mit einem Wahrheitswert Wahr oder Falsch belegt wurden, wurde eine Antwortmenge gefunden. Wenn dies wegen Widersprüchen nicht möglich ist, gibt es keine Antwortmenge. Das System nomore++ arbeitet mit einem erweiterten Aufspalten und Begrenzen mit WFS Ansatz aus 2.4.1. Zusätzlich zu den Belegungen der Atome, beziehungsweise der Köpfe, kann auch den Körpern der Regeln Werte zugeordnet werden. Desweiteren ist nicht nur das Schließen von einem Körper zu einem Kopf zugelassen, sondern auch andersherum. Dadurch gibt es nicht nur die Möglichkeit an Köpfen aufzuspalten, beziehungsweise einen choice zu machen, sondern auch für Körper. Der potentielle Suchraum wird dadurch natürlich größer, aber er kann auch stärker eingeschränkt werden.

Kapitel 3

Phasenübergang

Der Phasenübergang beschreibt eine Region bezüglich eines Parameters, in der ein anderer Parameter eine zu seinem sonstigen Verhalten relativ starken Sprung in seinen Werten durchmacht. Ein Beispiel ist der Phasenübergang des Wassers von fest zu flüssig, während die Temperatur allmählich steigt, nimmt beim Schmelzpunkt die Festigkeit des Wassers plötzlich stark ab. Phasenübergänge wurden auch in anderen Bereichen außerhalb der Physik gefunden, z.B. der Mathematik, Chemie oder Informatik.

Oft haben diese Phasenübergänge auch einen Einfluss auf andere Parameter. Während beim Wasser innerhalb der Phasen die Energiemenge, die nötig ist, um es um ein Grad zu erwärmen, ungefähr konstant ist, nimmt diese Energiemenge bei den Phasenübergängen plötzlich rapide zu.

Ähnliches ist auch häufig für Phasenübergänge bei Berechnungsproblemen für dem Berechnungsaufwand zu beobachten. Im Bereich von Phasenübergängen (z.B. von lösbaren zu nicht lösbaren Problemen) ist der Berechnungsaufwand oft höher.

Wie in [7] bemerkt, können viele NP-harte Probleme durch einen Parameter charakterisiert sein, der angibt wie stark das Problem eingeschränkt wurde. Wenn die Problemklassen bezüglich dieses Parameters betrachtet werden, gibt es einen Phasenübergang von meist erfüllbar zu meist nicht erfüllbar. Mit dem Anwachsen der Problemgröße werden die Kurven, z.B. für die Erfüllbarkeit oder den Berechnungsaufwand, ausgeprägter. In vielen Fällen kann ein (Skalierungs-) Parameter gefunden werden, der durch eine Skalierungsfunktion aus anderen Parametern gebildet wird, so dass die Kurven für die Erfüllbarkeit sich bezüglich ihm in einem Bereich schneiden und dort einen Phasenübergang haben. Wenn sich Kurven in einem Punkt überschneiden, wird dies Überschneidungspunkt (crossover point) genannt. Bezüglich eines solchen Skalierungsparameters sind auch Verläufe von Kurven für andere Parameter (z.B. Zeit) ähnlich, das heißt z.B., dass die Kurven bezüglich ihm ungefähr beim gleichen Wert ein Maximum aufweisen.

3.1 Phasenübergang bei Graphen

Bei der Suche nach Hamiltonschen Zyklen, in Graphen mit per Zufall generierten Kanten, ist der Skalierungsparameter durch die Formel $\frac{E}{N \log N}$ bestimmt [7], wobei E die Anzahl der Kanten im Graph ist und N die Anzahl der Knoten (nodes). Bezüglich dieses Parameters gibt es einen Überschneidungspunkt bei der Erfüllbarkeitskurve, wie im Diagram 3.1 aus [7] Seite 3 für verschiedene Anzahl von Knoten N zu sehen. Deutlich erkennbar ist, dass mit Zunahme der Anzahl der Kanten, bei rund 0.7, es eine starke Zunahme der Graphen gibt, die Hamiltonschen Zyklen besitzen. Von einem Bereich (Phase 1) in der so gut wie kein Graph einen Hamiltonschen Zyklus hat, zu einem Bereich (Phase 2) wo fast alle Graphen Hamiltonschen Zyklen besitzen. In der Nähe dieses Überschneidungspunktes wurde auch das Maximum des Berechnungsaufwandes gefunden.

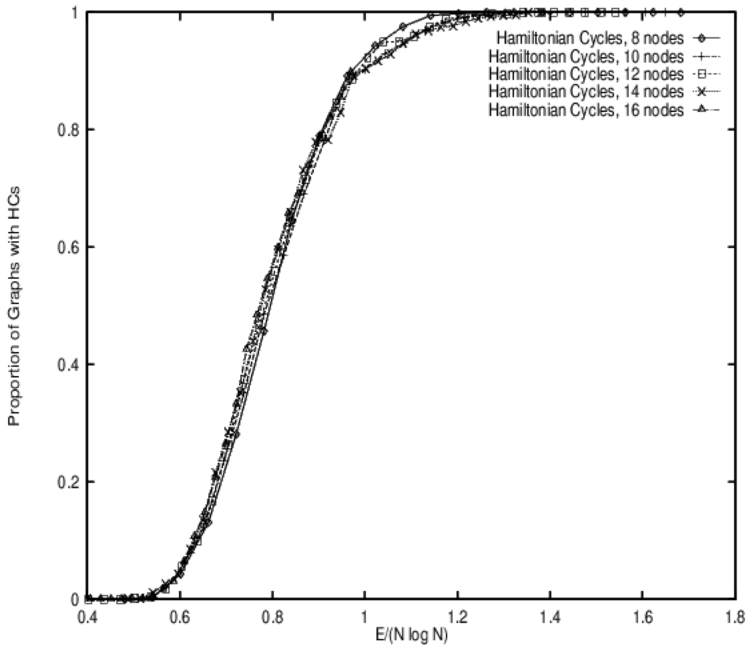

Abbildung 3.1: Hamiltonsche Zyklen in Zufallsgraphen

3.2 Untersuchungen des Berechnungsaufwands und Phasenübergang bei SAT

Der Berechnungsaufwand und die Verteilung von logischen Programmen mit und ohne Modellen bei SAT Solvern wurden schon mehrfach untersucht, zum Beispiel in [16], [17], [3] und [6]. In [6] sind einige Wahrscheinlichkleitsuntersuchungen hinsichtlich des Berechnungsaufwands bei SAT Problemklassen zu finden. Mit dem Ergebnis, dass große Bereiche des Parameterraums Probleme darstellen, die im Durchschnitt mit einer Wahrscheinlichkeit gegen 1 mit polynomiell wachsenden Aufwand zu lösen sind.

In [16], [17] und [3] wurden statistische Untersuchung anhand verschiedener Modelle gemacht.

Beim fixed clause-length (feste Klausellänge) oder k-SAT Modell, besitzt jede Klausel eine feste Anzahl k von Literalen. Die Literale werden aus der Menge der Variablen zufällig ausgewählt und mit einer Wahrscheinlichkeit von 0.5 negiert. Beim constant-probability (konstante Wahrscheinlichkeit) Modell, hat jede Variable eine constante Wahrscheinlichkeit in einer Klausel aufgenommen zu werden, wobei die Wahrscheinlichkeit, dass sie negiert aufgenommen wird, wieder 0.5 ist. Die Auswahl geschieht natürlich immer zufällig. Eine Erweiterung zum constant-probability Modell ist das εk-SAT Modell, bei dem trivial zu lösende Klauseln verboten sind (z.B. die Klausel (a) oder $(a \vee not\ a)$). Eine andere Erweiterung ist das $[k, l]$-SAT Modell, bei dem die Klausellänge der Klauseln gleichmäßig zwischen der Länge k und l verteilt ist.

Die restlichen Parameter beider Modelle sind Anzahl der Klauseln und Anzahl der Variablen für die Probleme bzw. Formeln. Für eine Testreihe wurde eine Anzahl von Testpunkten mit jeweils mehreren Problemen (in der Größenordnung von 1000) mit festen Parameterwerten generiert, wobei sich die verschiedenen Testpunkte in einem Parameterwert (meist der Klauselzahl) unterschieden.

In Abbildung 3.2 übernommen aus [3] ist ein typisches Ergebnis zu sehen. Verwendet wurde dabei das fixed clause-length Modell mit 3 Literalen 3-SAT und 50 Variablen. Auf der x-Achse ist dabei der Parameter $m/n = Klauseln/Variablen$ aufgetragen, da sich dieser als charakteristischer Skalierungsparameter gezeigt hat. Im oberen Diagramm sind auf der y-Achse der Median der calls ((Sub-)Aufrufe) des DPLL-Solvers aufgeführt, was einer Art Berechnungsaufwand entspricht. Die y-Achse des unteren Diagrammes zeigt den Anteil der lösbaren Probleme des Testpunktes an. Im unterem Diagramm ist deutlich ein Phasenübergang zu erkennen, von einem Bereich mit überwiegend lösbaren, zu einem Bereich mit überwiegend nicht lösbaren Problemen. Dieser Phasenübergang spielt sich hauptsächlich zwischen dem m/n Werten 4 und 5 ab.

Im oberen Diagramm ist ein leicht-schwer-leicht Muster (easy-hard-easy pattern) zu erkennen. Wobei das Maximum ungefähr dort zu liegen scheint, wo die Kurve im unterem Diagramm den 0.5 Wert erreicht.

In Abbildung 3.3, übernommen aus [16] Seite 4, sind die Diagramme mit den Kurven für $n = 25$ Variablen und verschiedene Werte für die Anzahl der Literale pro Klausel k zu sehen. Dabei zeigt diesmal das untere Diagramm den Median des Berechnungsaufwandes in Form von logarithmisch aufgetragenen Median der DP (Davis Putnam Prozedur) steps, die Prozedur ist in [16] auf Seite 2 zu finden. Auch hier tauchen wieder die gleichen Muster in der gleichen Kombination auf. Weiterhin ist zu erkennen, dass mit zunehmenden k auch das Maximum des Berechnungsaufwandsmedians und der Kurvendurchgang durch den 0.5 Wert im

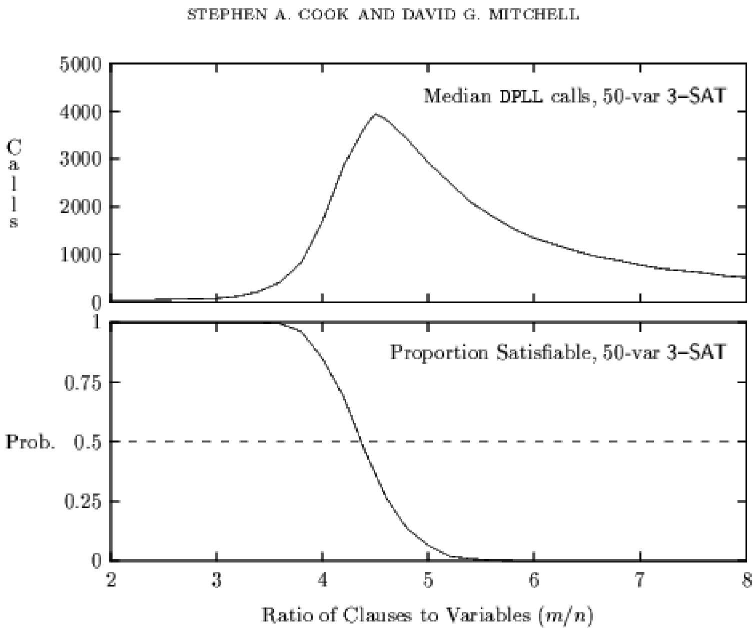

Abbildung 3.2: Performance des Davis Putman Allgorithmus und Erfüllbarkeit für Random 3-SAT mit 50 Variablen

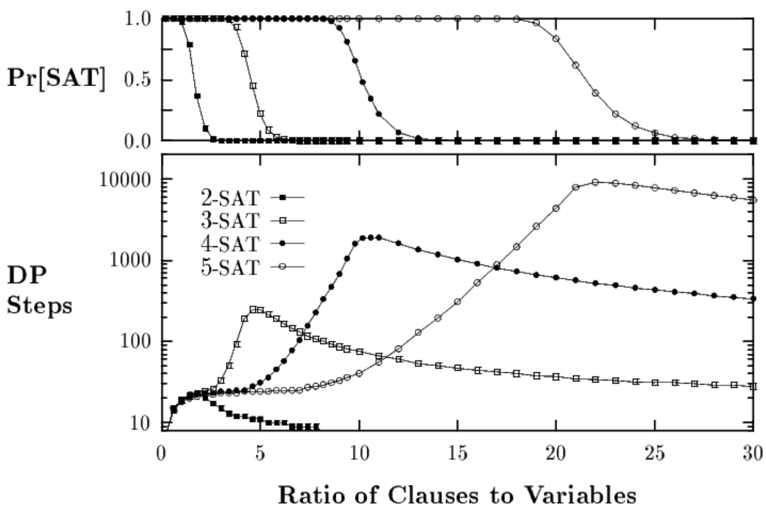

Abbildung 3.3: Median steps des Davis Putman Allgorithmus für Random *k*-SAT mit ausgewählten Werten von *k*

oberen Diagramm nach rechts wandert. Dies entspricht einer Zunahme des m/n Parameters. In dem unteren Diagramm ist weiterhin eine Zunahme des Maximums zu sehen.

2-SAT Probleme sind eine Ausnahme. Wie in [16] besprochen, sind 2-SAT Probleme in $O(n + m)$ Zeit zu lösen, Untersuchungen dort, lassen auch einen linearen Anstieg des Berechnungsaufwands vermuten.

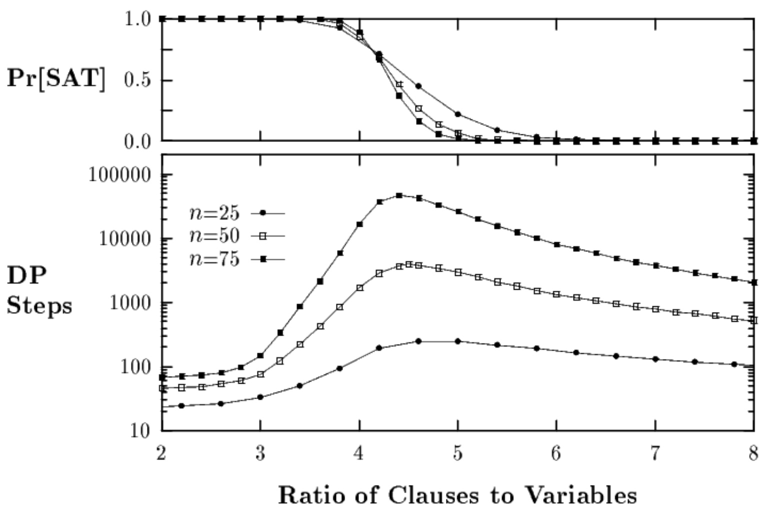

Abbildung 3.4: Steps des Davis Putman Algorithmus für Random 3-SAT mit ausgewählten Werten von n

In Abbildung 3.4, übernommen aus [16] Seite 6, sind die Diagramme mit den Kurven für verschiedene Anzahl von Variablen ($n = 25; 50; 75$) und 3 Literale pro Klausel (3-SAT) zu sehen. Die Art der Diagramme entspricht dabei denen in Abbildung 3.3. Zu sehen ist hierbei, dass sowohl das Maximum beim Berechnungsaufwand, als auch der Phasenübergang bei der Erfüllbarkeit anscheinend invariant bezüglich des m/n Parameters ist. In beiden Diagrammen werden die Kurven mit zunehmenden n steiler. Desweiteren wächst der Berechnungsaufwand mit zunehmenden n wahrscheinlich exponentiell.

Das constant-probability (konstant Wahrscheinlichkeit) Modell lässt [17] vermuten, dass keine schweren Probleme damit generiert werden können und es damit nicht für den Test von SAT-Solvern geeignet ist.

In [16] wird das εk-SAT Modell untersucht, das ein ähnliches Verhalten zeigt wie das k-SAT Modell, davon abgesehen, dass die Probleme anscheinend wesentlich einfacher sind. Auch andere Modelle zeigen eine zum k-SAT Modell ähnliches Verhalten. Beim $[k, l]$-SAT Modell wird davon ausgegangen, dass es im Durchschnitt genauso schwer ist wie 2-SAT.

Das leicht-schwer-leicht Muster wird in [17] damit begründet, dass in Formeln, mit wenigen Klauseln pro Variable, die Variablen wenig beschränkt werden und so die meisten Belegungen gültig sind. Bei vielen Klauseln pro Variable, sind die Variablen zu sehr beschränkt und es kann schnell ermittelt werden, dass keine gültigen Belegungen möglich sind. Dazwischen, im schwer Bereich, müssen viele Variablen belegt werden, um herauszufinden, ob dies zu einer gültigen Belegung führt. Da bei der Erfüllbarkeitswahrscheinlichkeit von 0.5 am schlechtesten vorausgesagt werden kann, ob die Formel erfüllbar ist oder nicht, liegt dort der schwierigste Bereich. In [16] wird dazu noch die Ergänzung gemacht, dass das Maximum sich durchaus etwas verschieben kann, wenn z.B. der Algorithmus dazu ausgelegt wurde erfüllbare Formeln schneller zu lösen.

Bei der Frage, ob zufällig generierte Probleme als Test für SAT-Solver benutzbar sind, gehen die Meinungen auseinander. Hauptargument dafür, ob dies möglich ist, ist anscheinend, ob schwere Probleme generiert werden können. Während in Mitchell's Dokumenten davon ausgegangen wird, dass der schwere Bereich dafür durchaus geeignet ist, wurden in anderen Dokumenten keine passenden Parameter dafür gefunden.

3.3 Untersuchungen des Berechnungsaufwandes und Phasenübergang bei ASP

In [21] Kapitel 4 wurden von Yuting Zhao Phasenübergänge bei der Antwortmengensuche untersucht. Das Dokument [13] ist die leicht veränderte Ausgliederung des Kapitels zu einem eigenen Dokument. Darin werden 3 ASP Systeme benutzt, Smodels 2.27, DLV (Mai 16, 2003 Version) und ASSAT 2.0, der mit dem SAT-Solver Chaff2 arbeitet.

Bei seinem fixed bodylength Modell k-LP haben die Regeln die Länge k, das heißt $(k-1)$ Körperatome. Das Modell ist an das fixed bodylength k-SAT Modell bei SAT angelehnt. Dieses Modell entspricht dem $(k-1)$-LP Modell, das in diesem Dokument verwendet wurde und in Abschnitt 4.1.1 auf Seite 23 beschrieben wird. Wobei bei Yuting Zhao allerdings kein Atom doppelt in einem Körper einer Regel auftaucht und auch keine Regel doppelt auftaucht.

Yuting Zhao verwendet, angelehnt an SAT, L als Nummer der Regeln und N als Nummer der Atome.

Diagramm 3.5 ist aus [21] Seite 82. Es handelt sich dabei um das 2-LP (in [21] 3-LP) Modell mit 150 Atomen. Auf der x-Achse ist der L/N ($= R/A$) Faktor aufgetragen, $L/N = 5$ heißt z.B. die Antwortmengenprogrammklasse mit $N = 150$ Atomen und $L = 150*5 = 750$ Regeln. Auf der linken Seite ist die Wahrscheinlichkeit pro (=probability) aufgetragen, mit der in der Klasse die Programme erfüllbar waren, beziehungsweise mindestens eine Antwortmenge hatten. Dazu gehört die mit pro bezeichnete Kurve. Auf der rechten Seite ist für die restlichen Kurven die Zeitskala in Sekunden der Durchschnittszeit aufgetragen. Die Testläufe wurden mit 3 verschiedenen ASP Solvern durchgeführt, DLV, Smodels und ASSAT. Die Kurven, deren Legendenbeschriftung mit D beginnt, gehören zu DLV, S steht für Smodels und A für ASSAT. Die Endung total bedeutet, dass es sich um alle

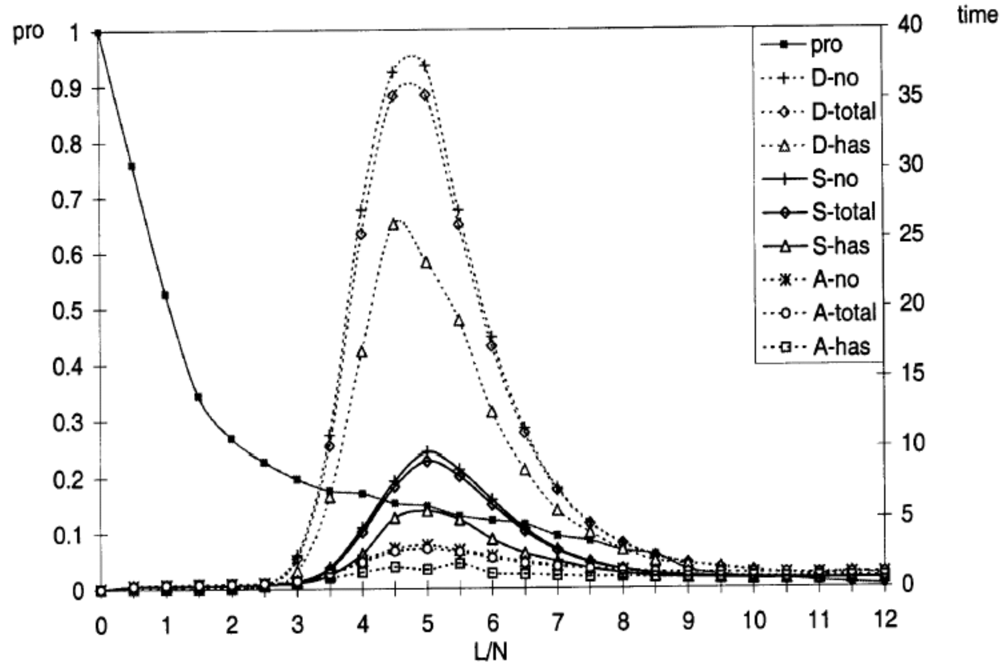

Abbildung 3.5: 2-LP bei 150 Atomen

generierten Beispiele handelt. Die no Kurven beinhalten nur Daten von Antwortmengenprogrammen, die keine Antwortmengen hatten. Die has Kurve bezieht sich dabei auf die Zeit um eine Antwortmenge zu finden.

Wie zu sehen, unterscheiden sich die drei Solver deutlich in der Zeit, um eine Antwortmenge zu finden oder zu sagen, dass es keine gibt. DLV schneidet wohl am schlechtesten ab, da es dafür ausgelegt ist, mit der mächtigsten Sprache der drei Solver zu arbeiten. ASSAT schneidet am besten ab, da es von der Erfahrung, die bisher mit SAT Solvern gemacht wurden, profitiert. Allerdings ist es mit dieser Version von ASSAT, im Gegensatz zu den anderen Solvern, noch unmöglich weitere Antwortmengen zu finden. Wenn aber von den konkreten Berechnungszeiten abgesehen wird, ähneln sich die Kurven der drei Solver doch sehr.

Interessant ist, dass alle drei Solver ein leicht-schwer-leicht Muster aufweisen und ungefähr an der gleichen Stelle bei $L/N = 5$ ein Maximum besitzen. Die Schwierigkeit der Problemlösung liegt anscheinend in den Problemen selbst und ist weniger von den Solvern abhängig.

In [21] wird weiterhin bemerkt, dass das Maximum in einer Region liegt, in der die meisten Probleme nicht lösbar sind (im Bereich in dem annähernd der 0.1-0.2-ste Anteil der Probleme lösbar ist) und nicht wie bei SAT, in der Übergangsregion von lösbaren zu nicht lösbaren Problemen. Der Grund wird in der Nichtmonotonität von ASP gesehen.

Im Vergleich zu dem Diagramm 3.4, für die entsprechende Klasse von SAT Problemen, fällt auf, dass das Maximum sich bei ähnlichen L/N Werten befindet. Während sich bei SAT die Kurve für den Anteil der erfüllbare Probleme noch eine Zeitlang in der Nähe von 1 befindet, sinkt die entsprechende Kurve bei ASP gleich ab. In diesem Sinne gibt es bei ASP anscheinend keinen Phasenübergang bei der Erfüllbarkeit, da es nur eine Phase der größtenteils unerfüllbaren Probleme gibt.

Aufgrund der Ähnlichkeit zu den SAT Problemen kann aber angenommen werden, dass sich auch hinter dem Maximum bei den ASP Zeitkurven ein versteckter Phasenübergang eines Parameters verbirgt, der nicht erfasst oder abgebildet wurde.

Weiterhin ist der Umstand, das Probleme nicht lösbar sind, anscheinend in diesem Modell, schwerer herauszufinden, als das Probleme lösbar sind.

In [21] wird festgestellt, dass mit steigender Anzahl von Atomen die Kurven wie bei SAT ausgeprägter werden. Die Erfüllbarkeitskurven fallen stärker ab und die Berechnungsaufwandskurven haben einen größeren Bogen im schwer Bereich. Auch dies ist unabhängig vom verwendeten Solver.

Beim Mischen von Regeln mit den Körperlängen 2 und 3 in einem Verhältnis 1 zu 1, wiederholen sich die Ergebnisse, wenn von den konkreten L/N Werten abgesehen wird. Die Kurven sind in x-Achsen Richtung (L/N Achse) nur gedehnt (beziehungsweise gestaucht).

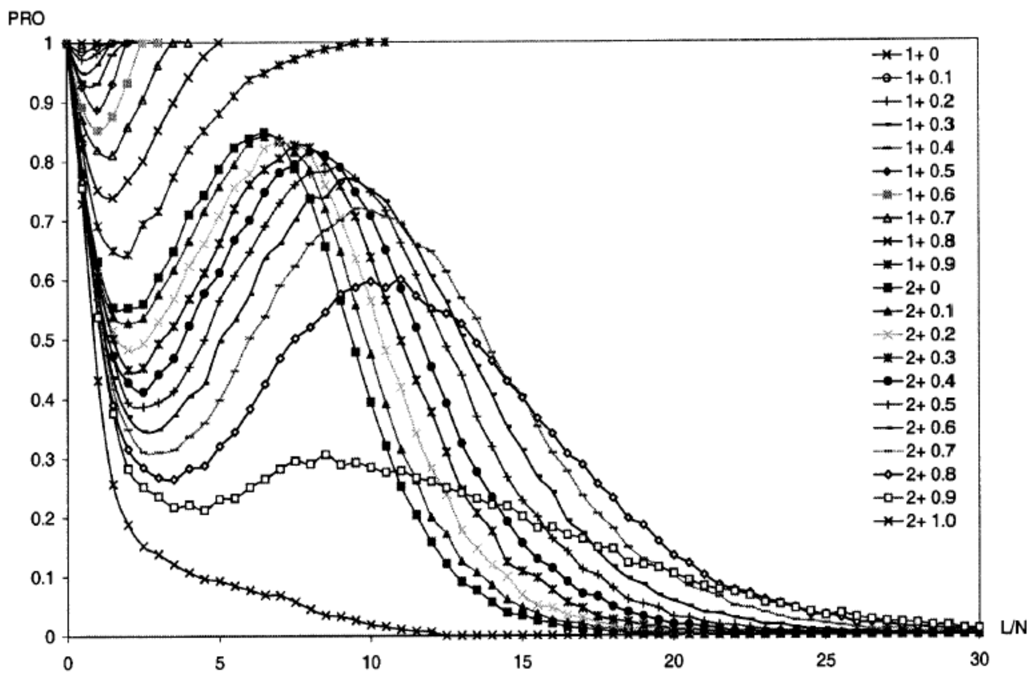

Abbildung 3.6: Erfüllbarkeit von 0 + p-LP und 1 + p-LP mit 100 Atomen

Ein weiteres Modell ist das $k+p$-LP Modell, wobei der Anteil p der Regeln aus Regeln mit der Körperlänge $(k-1)$ und der Rest aus Regeln mit der Körperlänge k besteht. Grafik 3.6 (aus [21] Seite 94) gibt die Erfüllbarkeitswahrscheinlichkeiten bei 100 Atomen mit unterschiedlichen k und p Werten an.

Alle Kurven, außer der für 2-LP (entspricht $2+1$-LP aus [21]) zeigen ein nicht-monotones Verhalten. Der Grund für den dargestellten Kurvenverlauf liegt wohl in der Nichtmonotonität von ASP.

Das Verhalten bei $1+p$-LP bei hohen L/N Werten beruht anscheinend darauf, dass Fakten nicht widerlegt werden können. Wenn ein Programm also genug Fakten beherbergt, um alle Widersprüche außer Kraft zu setzen, ist es erfüllbar. Die Wahrscheinlichkeit, dass ein Atom ein Fakt ist, nimmt in dem $1+p$-LP Modell mit steigendem L/N Faktor zu.

Das fallen-steigen-fallen (drop-rise-drop) Muster beim den $2+p$-LP Modellen wird mit steigender Anzahl von Atomen ausgeprägter ([21] Seite 90).

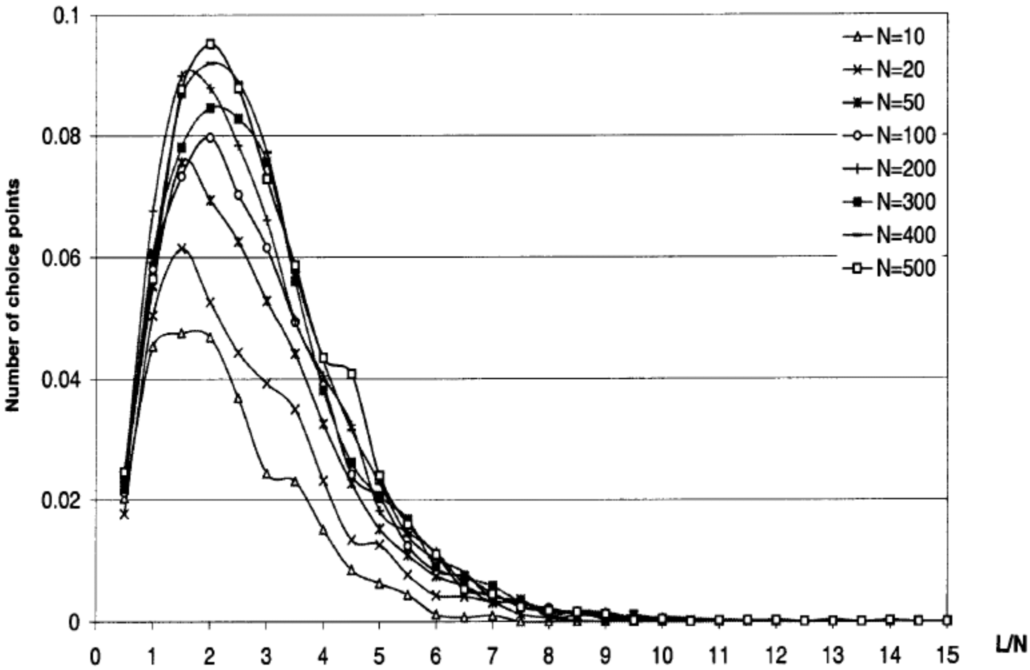

Abbildung 3.7: Durchschnittliche choices für 1-LP mit ausgewählter Anzahl N von Atomen

Grafik 3.7 aus [21] Seite 92 zeigt die durchschnittliche Anzahl von choices bei dem 1-LP Modell (das $2+0$-LP oder 2-LP Modell aus [21]) bei unterschiedlichen Werten für die Anzahl der Atome N. Auch hier zeigt sich das leicht-schwer-leicht Muster, dabei liegt das einem Maximum um $L/N = 2$.

Kapitel 4

Modelle für die Antwortmengen-programmgenerierung

Zu beachten ist, dass bei den nachfolgenden Modellen die Parameter zur Generierung der Programme dienen. Die entstehenden Programme können durchaus andere Parameter haben, z.B. können nicht alle Atome verwendet sein oder es können Regeln doppelt generiert werden. Modelle bei denen die Parameter zur Generierung immer die gleichen sind, wie die Parameter des entstehenden Programms, sind aufwändiger zu realisieren und ob sich dieser zusätzliche Aufwand für diese Untersuchung lohnt, ist stark zu bezweifeln.

Die hier aufgeführten Modelle sollten sich in ihrer Reihenfolge immer mehr den Anwendungsbeispielen annähern. Die meisten Anwendungsbeispiele enthalten Regeln mit unterschiedlicher Körperlänge, inklusive Fakten. Jedes Atom des Programms ist meist auch im Kopf einer Regel enthalten, sonst wäre das Atom von vornherein nur mit Falsch zu belegen.

4.1 Randomisierte Antwortmengenprogramme

Die randomisierten Antwortmengenprogramme sind an Modelle aus dem Bereich SAT angelehnt und enthalten keine Variablen, sind also schon in der ground Version.

Bei den nachfolgenden Modellen wurden auch nicht, wie in [21], doppelte Vorkommen von Regeln oder Atomen im Körper vermieden. Ihr Effekt sollte für den Parameterbereich von Interesse vernachlässigbar sein. Doppelungen heraus zu filtern, würde aber die Generierung von Antwortmengenprogrammen aufwändiger machen.

4.1.1 Feste Körperlänge (fixed bodylength)

In diesem k-LP Modell werden die generierten Antwortmengenprogramme durch drei Parameter bestimmt, der Anzahl der Atome, Regeln und Literale pro

Regelkörper. Jedes Antwortmengenprogramm hat dann die vorgegebene Anzahl von Regeln und k Literale pro Körper. Der Kopf einer Regel wird aus einer Menge von Atomen, mit der geforderten Anzahl von Atomen, zufällig ausgewählt. Dabei hat jedes Atom die gleiche Wahrscheinlichkeit gewählt zu werden. Für die einzelnen Literale des Körpers einer Regel wird aus der gleichen Menge von Atomen zufällig eines ausgewählt, auch dabei haben alle Atome die gleiche Wahrscheinlichkeit gewählt zu werden. Dieses wird dann mit der Wahrscheinlichkeit von 0.5 negiert.

Ich bin dabei von der Notation in [21] und k-SAT abgewichen, da ich von der Definition von normal logic Programmen aus Abschnitt 2.1.1 auf Seite 4 ausgehe. Das k-LP Modell in diesem Dokument entspricht also eher dem $(k + 1)$-LP Modell in [21] und dem $(k + 1)$-SAT Modell.

4.1.2 Gemischte Körperlänge (mixed bodylength)

Dieses k-pLP Modell ist an das constant-probability (konstante Wahrscheinlichkeit) Modell von SAT angelehnt. In ihm werden die generierten Antwortmengenprogramme durch drei Parameter bestimmt, der Anzahl der Atome, der Regeln und der durchschnittlichen Anzahl k der Literale pro Regelkörper. Der Kopf einer Regel wird aus einer Menge von Atomen, mit der geforderten Anzahl von Atomen, zufällig ausgewählt. Dabei hat jedes Atom die gleiche Wahrscheinlichkeit gewählt zu werden. Für die Literale des Körpers einer Regel, wird aus der gleichen Menge von Atomen, jedes mit der gleichen Wahrscheinlichkeit ausgewählt und dieses dann mit der Wahrscheinlichkeit von 0.5 negiert. Die Wahrscheinlichkeit, für ein Atom entweder unnegiert oder negiert gewählt zu werden, beträgt:

$$\frac{\text{durchschnittliche Anzahl der Literale pro Regelkörper}}{(\text{Anzahl der Atome}) * 2}$$

Der Vorteil dieses Modells besteht darin, dass mit ihm alle möglichen Antwortmengenprogramme generiert werden können. Der Nachteil ist, dass der Anteil der interessanten anwendungsnahen Antwortmengenprogramme darunter wohl eher gering ist.

Dieses Modell kann auch so betrachtet werden, dass die Antwortmengenprogramme aus der Mischung von k-LP Antwortmengenprogrammen entstehen.

4.1.3 Gemischte Körperlänge mit zusammenhängenden Abhängigkeitsgraphen

Antwortmengenprogramme für Anwendungsbeispiele haben oft die Eigenschaft, dass die Atome alle zusammenhängen, das heißt, dass die Abhängikkeitsgraphen des Programms aus einer Zusammenhangskomponente bestehen.

Dies wurde mit folgendem Modell nachgebildet. Dafür sind die Atome mit $(1 \ldots A)$ und die Regeln mit $(1 \ldots R)$ durchnummeriert. Bei der Generierung der Programme werden die ersten $A - 1$ Regeln des Antwortmengenprogramms P so generiert, dass im Körper der i'ten Regel das Atom $i + 1$ vorkommt (negiert oder nicht negiert ist dabei gleich wahrscheinlich) und im Kopf der Regel ein Atom kleiner als $i + 1$. Dadurch sind alle Atome miteinander verbunden und kleinere Atome tauchen etwas häufiger auf. Dann wurden die restlichen Regeln mit zufällig gewählten Atomen als Kopf erzeugt und in die Körper aller Regeln solange zufällig gewählte und zufällig negierte Atome eingefügt, bis die gewünschte durchschnittliche Körperlänge erreicht war.

In einem zweiten Modell wurde dafür gesorgt, dass jedes Atom mindestens in einem Kopf einer Regel vorkommt. Die Generierung funktioniert wie im ersten Modell, nur dass im Kopf der i'ten Regel für $i < A$ das Atom $i + 1$ vorkommt und im Körper der Regel mindestens ein Atom kleiner als $i + 1$, negiert oder unnegiert.

4.2 Anwendungsbeispiele

Soweit wurden nur zufällig erzeugte Antwortmengenprogramme auf den benötigten Berechnungsaufwand hin untersucht. Die eigentliche Frage, die sich aber stellt, ist, wie der Berechnungsaufwand bei Antwortmengenprogrammen für Probleme aus der Anwendung aussehen. Leider sind für statistische Untersuchungen eine große Anzahl, von in einer Weise zufällig generierten, Instanzen nötig, das heißt, unter Festlegung einiger Metaparameter (zum Beispiel Atomzahl, Regelzahl) werden die anderen Parameter (zum Beispiel die einzelnen Regeln) der Instanz zufällig generiert. Leider gibt es aber keine (bekannten) zufälligen Parameter der Antwortmengenprogramme für Anwendungsbeispiele. Ein Anwendungsbeispiel tritt bei der Anwendung auf und ist nicht zufällig erzeugt.

Im Nachfolgenden werde ich deshalb untersuchen, wie sich der Berechnungsaufwand bei anwendungsnahen Klassen, wie der Suche nach hamiltonschen Zyklen bei Graphen, verhält.

Ob aus Untersuchungen des Berechnungsaufwandes von zufälligen Antwortmengenprogrammen auf den Berechnungsaufwand von Antwortmengenprogrammen für Anwendungsprobleme geschlossen werden kann, ist allerdings fraglich. Es kann behauptet werden, dass bei zufälligen Antwortmengenprogrammen weniger Strukturen enthalten sind, als bei Anwendungsbeispielen. Diese Strukturen können, auf der einen Seite, den Berechnungsaufwand erhöhen, da es mehr Abhängigkeiten zwischen den unterschiedlichen Atomen gibt, die berücksichtigt werden müssen. Auf der anderen Seite, könnten diese Strukturen auch dem Antwortmengensolver zusätzliche Informationen liefern, die den Berechnungsaufwand verringern, siehe zum Beispiel splitting sets oder stratifikation ([1]).

Sicherlich können Antwortmengenprogramme so konstruiert werden, dass sie nicht in das Bild von den zufälligen Antwortmengenbeispielen passen, allerdings sind dies dann auch keine Anwendungsbeispiele.

Die Frage ist aber, ob Antwortmengenprogramme von Anwendungsproblemen durch bestimmte Parameterausprägungen charakterisiert sind, wie sie so, bei zufällig generierten Programmen, nicht oder sehr selten auftreten. Ist dies nicht der Fall, kann von Eigenschaften bei zufällig generierten Programmen durchaus auf Eigenschaften bei Programmen von Anwendungsproblemen geschlossen werden. Dies aber zu beweisen, ist im allgemeinen nicht möglich, da alle möglichen Anwendungsbeispiele einer Klasse in allen möglichen Programmumsetzungen betrachtet werden müssten.

Die Behauptung, reale Anwendungsprobleme gehören zur Klasse der schwere Probleme, ist nicht haltbar. Bei der Anwendung geht es meist darum, Etwas zu automatisieren und nicht (wie oft in der Forschung) darum, schwere Probleme zu lösen. Wenn jemand beispielsweise einen Roboter in seiner Fabrik einsetzt, um Gegenstände zwischen einzelnen Arbeitsplätzen zu transportieren, ist es durchaus wahrscheinlich, dass er die Gegebenheiten so verändert, dass die Aufgabe für den Roboter möglichst leicht wird. Die Probleme werden aber oft allein durch ihre Größe aufwändig (z.B. durch die Steuerung von 100 abhängigen Maschinen und Menschen auf einem Raster von 10000 mal 10000).

Ein anderes Problem ist, was zu den Anwendungsproblemen gezählt wird. Kommen Anwendungsprobleme nur in der Praxis vor oder auch in der Forschung? Sind Probleme, deren Anwendung das Testen von Antwortmengensolvern ist, auch Anwendungsprobleme?

Das Generierungsmodell, das hier als Vertreter der Anwendungsbeispiele gewählt wurde, ist das Finden von Hamiltonsche Zyklen auf randomisierte Graphen, da es noch relativ einfach ist, durchaus auch in Anwendungsproblemen zu finden ist und schon früher untersucht wurde.

4.2.1 Hamiltonsche Zyklen auf randomisierte Graphen

Die Antwortmengenprogramme zur Berechnung von hamiltonschen Zyklen auf einem gerichteten Graphen bestehen aus zwei Teilen. Einen Teil zur Berechnung der Zyklen und einen Teil der den Graphen darstellt (Datenteil).

Der Teil zur Berechnung der Zyklen wurde aus [21] Seite 113 übernommen. Es handelt sich dabei um ein modifiziertes Antwortmengenprogramm der Universität Kentucky ASP Benchmark Webseite `http://www.cs.engr.uky.edu/ai/benchmark-suite/hamcyc.sm` . Dabei wurden ein paar Atome umbenannt, um mit der Niemelä's Kodierung konsistent zu sein. Das Antwortmengenprogramm lautet:

```
{hc(X,Y)}:- arc(X,Y).
:- 2{hc(X,Y):arc(X,Y)}, vertex(Y).
:- 2{hc(X,Y):arc(X,Y)}, vertex(X).
:- vertex(X), not r(X).
r(Y):- hc(X,Y),arc(X,Y), initialvtx(X).
r(Y):- hc(X,Y),arc(X,Y),r(X), not initialvtx(X).
initialvtx(0).
```

Listing 4.1: Hämiltonsche Zyklen auf Randomisierte Graphen

Der Teil, welcher den Graphen darstellt, besteht aus Fakten der Form `vertex(X).` für die Knoten und `arc(X,Y).` für die Kanten, wobei `X` und `Y` Namen für Knoten des Graphen sind, die aus dem Bereich der natürlichen Zahlen kommen. Der Knoten 0 muss immer vorhanden sein. Die Kante `arc(X,Y).` geht von Knoten `X` zu Knoten `Y`.

Die Parameter zur Generierung von Graphen sind die Anzahl der Knoten N und die Anzahl der Kanten E. Bei der Generierung werden zuerst N Knoten erzeugt und mit den Zahlen $0, ..., (N - 1)$ beschriftet. Dann werden E Paare von diesen Zahlen (den Knoten) gebildet, wobei sowohl die erste Zahl, als auch die zweite Zahl des Paares jeweils gleich verteilt, zufällig aus dem Bereich $0, \ldots, (N-1)$ gewählt wird. Doppelte Kanten werden ausgeschlossen.

Kapitel 5

Theoretische Überlegungen

Komplexe mathematische Theorien sind nur jeweils auf bestimmte Antwortmengenklassen anzuwenden, zum Beispiel auf zufällig erzeugte Programme mit fester Körperlänge. Es ist aber eine allgemeine Charakterisierung sinnvoll.

Viele der Überlegungen dürften unter Fachleuten bekannt sein, sind aber trotzdem anscheinend nicht in der Literatur zu finden.

5.1 Wahrscheinlichkeitsuntersuchung

Es wurde versucht, für die Antwortmengenprogrammierung Klassen von Antwortmengenprogrammen zu Wahrscheinlichkeitsantwortmengenprogramme zusammenzufassen. Dabei wird für den Kopf und Körper einer Regel angegeben, mit welcher Wahrscheinlichkeit die einzelnen Atome und Literale in ihr vorkommen. Beim Kopf sollte jeweils genau ein Atom gewählt werden und beim Körper sollten die Atome unabhängig voneinander gewählt werden. Diese Wahrscheinlichkeitsantwortmengenprogramme sind Grundlage für die Generierung der Antwortmengenprogramme bei gemischter Körperlänge aus Abschnitt 4.1.2 von Seite 24.

Ich habe versucht, mithilfe von einfacher Wahrscheinlichkeitsrechnung, allgemeine Aussagen über diese Wahrscheinlichkeitsantwortmengenprogramme zu treffen, z.B. mit welcher Wahrscheinlichkeit eine zufällig aus dem Wahrscheinlichkeitsantwortmengenprogramm generiertes Antwortmengenprogramm eine Antwortmenge hat. Leider ist dieser Versuch, wegen der Abhängigkeiten, zum Scheitern verurteilt.

Beispiel:

r_1: a(0.5).
r_2: b $\leftarrow$ a.
r_3: c $\leftarrow$ not c,a,b.

Die Wahrscheinlichkeiten sind hinter den Atomen in Klammern angegeben, wenn sie nicht 1 sind. In diesem Beispiel soll die erste Regel r_1 mit der Wahrscheinlichkeit 0.5 der Fakt a sein. Mit der Wahrscheinlichkeit 0.5 ist a also in allen

Antwortmengen des Programms. Damit ist auch b durch r_2 mit der Wahrscheinlichkeit 0.5 in allen Antwortmengen des Programms. Wenn nun a und b unabhängig voneinander betrachtet werden, wird die Regel r_3, wenn a in der Antwortmenge ist, nur mit der Wahrscheinlichkeit 0.5 verletzt. Wenn a in der Antwortmenge ist, muss aber auch durch Regel r_2 b in ihr sein, damit wird Regel r_3 dann immer verletzt. Um diese bei den Wahrscheinlichkeitsrechnungen zu berücksichtigen, müsste die Abhängigkeit von b und a explizit in den Rechnungen aufgenommen werden. Dann ist die Wahrscheinlichkeitsrechnung aber nicht mehr einfach.

5.2 Überlegungen zum Suchbaum

Im Nachfolgendem sind theoretische Überlegungen aufgeführt, die die Berechnungskomplexität der aufspalten und begrenzen (branch and bound) Algorithmen, im Hinblick auf deren Suchbaum, beleuchtet. Dabei wird von einfachen Überlegungen ausgegangen, die dann konkretisiert werden.

Ausgegangen wird von einem Antwortmengenprogramm P mit A Atomen. Damit ist die Anzahl der möglichen Interpretationen $|I| = 2^A$.

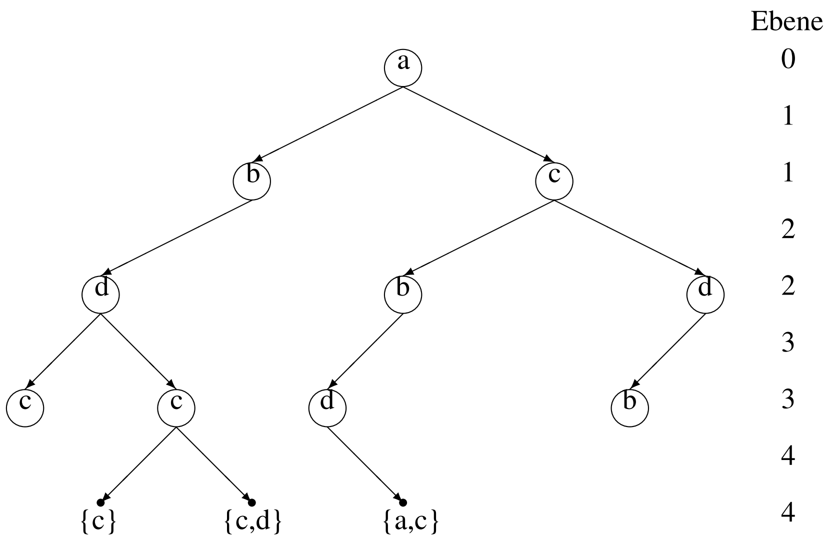

Abbildung 5.1: Suchbaumbeispiel

Der Suchbaum (siehe Suchbaumbeispiel 5.1), ist ein Baum, dessen Knoten, ohne die Blätterknoten der Ebene A, für einzelne Atome stehen, über die, bei der Suche an dieser Stelle, Aussagen gemacht werden. Der Wurzelknoten symbolisiert das erste Atom, bei dem bei der Suche eine Aussage getroffen wird. Die Aussagen oder Annahmen werden durch die abgehenden Kanten symbolisiert. Eine abgehende Kante eines Knotens, ist eine Kante die den Knoten enthält und nicht in Richtung des Wurzelknotens geht. Der Einfachheit halber kann die Vorstellung verwendet werden, dass die Kante, die nach links von einem Knoten abgeht, bedeutet, dass das zugehörige Atom mit Falsch (false) belegt wird, und Kante, die nach rechts abgeht, bedeutet, das Atom wird mit Wahr (true) belegt. Über jedes Atom, über das eine Aussage getroffen wurde, kann keine weitere Aussage getroffen werden. Ein Atom das mit Falsch belegt wurde, kann nicht mehr mit Wahr belegt werden. So taucht jedes Atom von der Wurzel bis zu einem Blatt nur höchstens einmal auf.

Die Knotenebene n im Suchbaum enthält alle Knoten, die n Kanten vom Wurzelknoten entfernt sind. In der Kantenebene n im Suchbaum sind die Kanten, die über $n-1$ Kanten mit dem Wurzelknoten verbunden sind, oder die Kanten, die zur Knotenebene n führen. Die Belegung eines Knotens besteht aus den Aussagen zu den Atomen, die der Pfad vom Wurzelknoten zum Knoten symbolisiert. Die Blätter der Ebene A repräsentieren die Antwortmengen des Programms P.

Von jedem Knoten, der ein Atom repräsentiert, können maximal zwei Kanten ausgehen, da über ein Atom die zwei Aussagen Wahr und Falsch getroffen werden können. Wenn das Antwortmengensuchprogramm schließen kann, dass eine Aussage nicht getroffen werden kann, gibt es auch nicht die dazugehörige Kante im Baum und damit auch nicht den von der Kante induzierten Unterbaum.

Es ist leicht erkennbar, dass es besser ist, einen Pfad in der Ebene n auszuschließen, als zwei Pfade in der Ebene $n+1$. Daraus folgt, einen Pfad in der Ebene n auszuschließen ist besser, als 2^i Pfade in der Ebene $n+i$.

Nachfolgend ist $K(n)$ die Anzahl der Kanten in Ebene n und $K = \sum_{n=1}^{A} K(n)$ die Anzahl der Kanten im gesamten Graph. Die Wurzel des Suchbaums wird als oben bezeichnet und oben dargestellt.

Im Folgenden geht es um das Wachstum des Berechnungsaufwands und nicht um konkrete Zahlen die den realen Berechnungsaufwand wiederspiegeln. Deshalb kann als Größe für den Berechnungsaufwand die Anzahl der Kanten im Suchbaum herangezogen werden und damit auch die Anzahl der Knoten, deren Zahl genau um Eins größer ist, als die der Kanten. Dies ist realistisch, da jede Annahme Zeit und damit Berechnungsaufwand benötigt. Wenn eine untere Abschätzung für einen Algorithmus getätigt werden soll, kann die Anzahl der Kanten des Suchbaums mit der minimalen Zeit oder dem minimalen Berechnungsaufwand, die das Suchprogramm für eine Kante benötigt, multipliziert werden. Hier wird deshalb die Anzahl der Kanten und die Höhe des Berechnungsaufwand synonym gebraucht.

Für obere Abschätzungen des Berechnungsaufwands ist dieser Ansatz allerdings nur bedingt zu gebrauchen. Der Gesamtberechnungsaufwand t_{ins} ergibt sich aus dem Mittelwert der Berechnungsaufwände für die Kanten $\bar{t}_{Kanten}$ (beziehungsweise Knoten $\bar{t}_{Knoten}$) multipliziert mit der Anzahl der Kanten ($t_{ins} = K * \bar{t}_{Kanten}$ bzw. $t_{ins} = (K+1) * \bar{t}_{Knoten}$). Wenn der Mittelwert der Berechnungsaufwände für die Kante also maximal so stark wächst wie die Anzahl der Kanten, ändert er nichts an der Wachstumsart des Gesamtberechnungsaufwands.

Alle Überlegungen dieses Abschnitts sind auch auf den Suchbaum von nomore++ anzuwenden. Zu den Knoten für die Atome kommen nur noch die Knoten für die Körper, die ja auch mit Wahrheitswerten belegt werden können. Der Rest ist identisch.

5.2.1 Suche von allen Antwortmengen

Hier soll zuerst die Suche von allen Antwortmengen betrachtet werden, da dies einen leichteren Fall darstellt. Nachfolgend sind einige Modelle für die Suche nach Antwortmengen aufgestellt, die in der Reihenfolge immer komplexer und leistungsfähiger werden.

5.2.1.1 Ausprobieren aller Belegungen

Die einfachste Suche, ist die, bei der alle möglichen Belegungen von Atomen für Interpretationen ausprobiert werden. Dabei werden die zurückgegeben, die Antwortmengen darstellen.

Es ist $K(n)$ die Anzahl der Kanten in Ebene n und K die Anzahl der Kanten im gesamten Graph, dann ist:

$$K(1) = 2 \tag{5.1}$$

$$K(n) = K(n-1) * 2 = 2^n \tag{5.2}$$

$$K = \sum_{n=1}^{A} 2^n = 2^{A+1} - 2 \tag{5.3}$$

In Kantenebene A stellen die Kanten das Ausprobieren der Belegungen dar, damit sind dort alle Kanten vorhanden.

Das Wachstum der Kanten und damit des Berechnungsaufwands ist in diesem Modell eindeutig exponentiell.

Bei 64 Atomen sind das $K = 2^{65} - 2$ Kanten, fast doppelt soviel wie 2^{64}. Eine Zahl, die im Hinblick auf 64 bit Pointer und Rechnerarchitektur auftaucht, von der es heißt, dass sie größer ist, als die Zahl der Atome auf der Erde. Der Versuch auf diese Weise alle Antwortmengen eines Programms mit nur 64 Atomen zu suchen, wäre also vergleichbar damit, alle Atome der Erde zu untersuchen. Ein Unterfangen, das sicherlich auch noch in naher Zukunft viel zu aufwändig ist.

Modernen Antwortmengensolvern ist es möglich alle Antwortmengen von Programmen mit einigen hundert Atomen zu finden. Für Programme mit 64 Atomen benötigen sie normalerweise nur Millisekunden, das zeigt, wie leistungsfähig sie sind. Wenn bei einem Antwortmengenprogramm mit 64 Atomen allerdings alle möglich Interpretationen Antwortmengen sind, ist die Erzeugung jeder dieser 2^{64} Antwortmengen zu aufwändig.

5.2.1.2 Ausschließen einiger Annahmen

Die Einsparungen beim Berechnungsaufwand basieren auf der Ausschließung von Annahmen über Atome.

$K_d(n)$ ist die Anzahl der Kanten oder Annahmen über Atome, die in Ebene n ausgeschlossen (deleted) werden können. Dabei werden nur Kanten gezählt, die in

dieser Ebene noch vorkommen. Also keine Kanten, die in der Ebene nicht existieren, weil auf einer niedrigeren Ebene eine ihrer Vorgängerkanten ausgeschlossen wurde.
Damit ist:

$$
\begin{aligned}
0 &\leq K_d(n) \leq 2K^-(n-1) \leq 2^n \quad n = 1 \ldots A && (5.4)\\
K^-(1) &= 2 - K_d(1) && (5.5)\\
K^-(n) &= K^-(n-1) * 2 - K_d(n) && (5.6)\\
K &= \sum_{n=1}^{A} K^-(n) && (5.7)\\
&= K^-(1) + K^-(2) + \ldots K^-(A)\\
&= (2 - K_d(1)) + ((2 - K_d(1)) * 2 - K_d(2)) + (((2 - K_d(1)) * 2 -\\
&\quad K_d(2)) * 2 - K_d(3)) + \ldots + (\ldots (((2 - K_d(1)) * 2 - K_d(2)) * 2\\
&\quad -K_d(3)) * 2 - K_d(4)) \ldots) * 2 - K_d(A))\\
&= (2 - K_d(1)) + (2 - K_d(1)) * 2 - K_d(2) + (2 - K_d(1)) * 4 -\\
&\quad K_d(2) * 2 - K_d(3) + \ldots + (2 - K_d(1)) * 2^{A-1} -\\
&\quad K_d(2) * 2^{A-2} - K_d(3) * 2^{A-3} - K_d(4) * 2^{A-4} \ldots - K_d(A)\\
&= (2 - K_d(1)) * (1 + 2 + 4 + \ldots + 2^{A-1}) -\\
&\quad K_d(2) * (1 + 2 + 4 + \ldots + 2^{A-2}) -\\
&\quad K_d(3) * (1 + 2 + 4 + \ldots + 2^{A-3}) - \ldots - K_d(A)\\
&= (2 - K_d(1)) * (2^A - 1) - K_d(2) * (2^{A-1} - 1) -\\
&\quad K_d(3) * (2^{A-2} - 1) - \ldots - K_d(A) && (5.8)\\
&= (2^{A+1} - 2) - K_d(1) * (2^A - 1) - K_d(2) * (2^{A-1} - 1) -\\
&\quad K_d(3) * (2^{A-2} - 1) - \ldots - K_d(A) && (5.9)
\end{aligned}
$$

Wie aus Formel 5.9 ersichtlich ist, bringt das Ausschließen von Kanten um so mehr, je früher es geschieht.

Dies soll an folgender vereinfachten Betrachtung verdeutlicht werden. Um sie überschaubar zu halten, wird nur das Ausschließen einer Annahme über ein beliebiges Atom a betrachtet. Dieses Atom wird nur im Suchbaum bewegt und es wird auch davon ausgegangen, dass pro Ebene nur jeweils ein Atom den Knoten in dieser Ebene zugeordnet ist.

Entweder wird eine Annahme über ein Atom a in Ebene n_1 ausgeschlossen oder es können $2^{n_2-n_1}$ Annahmen in der Ebene n_2 ausgeschlossen werden. Die Annahmen über andere Atome gelten sowohl für eine Annahme in Ebene n_1, als auch für $2^{n_2-n_1}$ Annahmen in Ebene n_2. Denn mit jeder Ebene, die a nach unten rutscht, verdoppeln sich die gleiche Annahmen über a mit den gleichen Grundbedingungen. Weil in dieser Betrachtung davon ausgegangen wird, dass Annahmen über andere Atome nicht ausgeschlossen werden, können Atome zwischen Ebene n_1 und n_2 keinen Einfluss auf a haben. Wenn sie zum Ausschließen von mehr Kanten von Atom a in Ebene n_2 führen, würde a in Ebene n_1 zum Ausschließen von ebenso vieler Kanten bei diesen Atomen führen.

$dK_d(n_1, n_2)$ ist die Anzahl von zusätzlichen Kanten im Suchbaum, wenn das Atom a von Ebene n_1 zu Ebene n_2 verschoben wird. Dann ist:

$$\begin{aligned} dK_d(n_1, n_2) &= (2^{A-(n_1-1)} - 1) - 2^{n_2-n_1} * (2^{A-(n_2-1)} - 1) \\ &= (2^{A-(n_1-1)} - 1) - 2^{n_2-n_1} * 2^{A-(n_2-1)} + 2^{n_2-n_1} \\ &= (2^{A-(n_1-1)} - 1) - 2^{A-n_2+1+n_2-n_1} + 2^{n_2-n_1} \\ &= 2^{A-(n_1-1)} - 1 - 2^{A-(n_1-1)} + 2^{n_2-n_1} \\ &= 2^{n_2-n_1} - 1 \end{aligned} \tag{5.10}$$

Das heißt, die Anzahl der zusätzlichen Kanten wächst exponentiell mit der Anzahl der Ebenen, die das Atom nach unten verschoben wird. Damit wächst die Einsparung von Kanten exponentiell mit der Zahl der Ebenen, mit der eine Annahme früher verworfen wird.

Im Allgemeinen allerdings werden viele verschiedene Annahmen über viele verschiedene Atome verworfen. Wenn eine Annahme verworfen wird, geschieht dies aufgrund einer Menge (auch $\emptyset$) von Annahmen, die früher getroffen wurden und mit der die Annahme bezüglich des Programms unvereinbar ist. Wenn solche Mengen von Annahmen (diese stellen 3-wertige Interpretationen dar) die zusammen unvereinbar sind, betrachtet werden, ist es vorteilhaft, die Annahmen bei der Suche in einer Weise zu treffen, dass möglichst schnell, möglichst viele dieser Mengen in der Menge der getroffenen Annahmen enthalten sind. Damit Kanten möglichst früh und reichlich gelöscht werden können.

5.2.1.3 Choice und schließen von Atomen

In diesem Modell gibt es nur zwei Operatoren, die bei der Suche ausgeführt werden: choice und schließen von Atomen. Die Operationen werden an den Stellen der Knoten des Suchbaums ausgeführt. Beim Schließen von Atomen, wird aus den bisher getroffenen Annahmen eine neue Annahme über ein Atom geschlossen, über das es noch keine Annahmen gibt. Im Suchbaum gibt es an dem entsprechenden Knoten nur eine abgehenden Kante. Bei einem choice werden beide Annahmen über ein Atom weiterverfolgt. In diesem Fall hat der entsprechende Knoten zwei ausgehende Kanten.

Dieses Modell kann keine Antwortmengensuche bewerkstelligen, da ein Ast niemals abgebrochen werden kann, es also immer eine Antwortmenge geben müsste. Es ist aber als Vorstufe zum Modell in 5.2.1.4 auf Seite 35 nützlich.

Wenn alle Annahmen einer Ebene geschlossen werden, ist $K_d(n) = K^-(n-1)$.

Für den choice Operator wird angenommen, dass er jeweils auf c_n'ten Teil der Knoten der Ebene n angewendet wird. Dadurch wird nur noch beim Anteil $(1 - c_n)$ der Knoten der Ebene n geschlossen.

Damit ist:

$$
\begin{aligned}
0 \leq \quad c_n \quad & \leq 1 \quad n = 1 \ldots A \quad (c_n * K^-(n-1)) \in \mathbb{N} \\
K_d(1) \quad & = \quad 1 + c_1 \quad c_1 \in \{0, 1\} \\
K_d(n) \quad & = \quad K^-(n-1) * (1 - c_n) \\
K^-(n) \quad & = \quad K^-(n-1) * 2 - K^-(n-1) * (1 - c_n) \\
K^-(n) \quad & = \quad K^-(n-1) * (1 + c_n) \\
K \quad & = \quad \sum_{n=1}^{A} K^-(n) \\
K \quad & = \quad \sum_{n=1}^{A} \prod_{i=1}^{n} (1 + c_i) \qquad (5.11)
\end{aligned}
$$

Die ersten choices haben demnach den stärksten Einfluss auf den Berechnungsaufwand.

Ist die Anzahl der Ebenen, in denen es choices gibt, gleich oder kleiner einer Konstanten C, wächst die Zahl der Kanten maximal linear mit der Anzahl der Atome des Programms P.
Da:

$$
\begin{aligned}
\prod_{i=1}^{n} (1 + c_i) \quad & \leq \quad \prod_{i=1}^{C} 2 \leq 2^C \\
\Rightarrow K \quad & \leq \quad \sum_{n=1}^{A} 2^C \leq A * 2^C \qquad (5.12)
\end{aligned}
$$

Jeder choice erzeugt genau einen zusätzlichen Ast. Ohne einen choice ist nur ein Ast vorhanden. Es sind also insgesammt die Anzahl der choices plus 1 Äste vorhanden. Jeder Ast hat maximal die Tiefe A. Daraus folgt, dass die Zahl der Kanten auch maximal linear mit der Anzahl der Atome des Programms P wächst, wenn die Anzahl der choices insgesamt gleich oder kleiner einer Konstanten C_2 ist
Da:

$$K \leq A * (C_2 + 1) \qquad (5.13)$$

Weiterhin ist das Minimum der Kanten, da zu jedem choice mindestens zwei Kanten gehören, begrenzt durch:

$$min(K) = 2C_2 \qquad (5.14)$$

Aus 5.13 und 5.14 folgt:

$$2C_2 \leq K \leq A * (C_2 + 1) \qquad (5.15)$$

Die Anzahl der Kanten wächst also linear mit der Anzahl der choices und Atomen.

Die Formeln 5.12 und 5.15 gelten auch für die nachfolgenden Modelle.

5.2.1.4 Realistische Suche

Zu den Operationen aus dem Modell in 5.2.1.3 auf Seite 33 kommt in diesem Modell noch die Operation „Widerspruch finden". Bei dieser Operation können beide Annahmen über ein Atom verworfen werden. Aus dem entsprechenden Knoten gehen daher keine Kanten ab. Diese Operation wird vor der choice Operation ausgeführt.

Mit den drei Operationen dieses Modells, enthält es alle grundlegenden Elemente von Antwortmengensolvern wie smodels und nomore++. Die Unterschiede der Systeme liegen in Dingen, wie der verwendeten Logik, die Art wie die Operationen arbeiten, der Herangehensweise und den verwendeten Heuristiken.

Im Nachfolgendem sei c_n, wie gehabt, der Anteil der choice und w_n der Anteil der Widersprüche in Ebene n.

$$\begin{aligned}
0 \leq \quad w_n & \leq 1 \quad n = 1 \ldots A \quad (w_n * K^-(n-1)) \in \mathbb{N} \\
0 \leq \quad c_n & \leq 1 \quad n = 1 \ldots A \quad (c_n * (1 - w_n) * K^-(n-1)) \in \mathbb{N} \\
K^-(n) & = (K^-(n-1) * (1 - w_n)) * 2 * (1 + c_n) \\
K^-(n) & = K^-(n-1) * 2 * (1 - w_n + c_n - w_n c_n) \\
K^-(n) & = \sum_{n=1}^{A} \prod_{i=1}^{n} 2 * (1 - w_n + c_n - w_n c_n) \\
K^-(n) & = \sum_{n=1}^{A} 2^n \prod_{i=1}^{n} (1 - w_n + c_n - w_n c_n) \qquad (5.16)
\end{aligned}$$

Nach 5.2.1.2 ist es folglich vorteilhaft, um die Kantenzahl möglichst gering zu halten, möglichst früh, möglichst viele Widersprüche zu finden und wo dies nicht möglich ist, mögliche Annahmen zu schließen. So selten und so spät wie möglich sollte der choice Operator eingesetzt werden.

Diesem Vorgehen stehen die nötigen Informationen gegenüber. Der choice Operator kann, bis auf die letzte Ebene, immer eingesetzt werden, ohne dass Antwortmengen verloren gehen. Zum Ausschließen einer Annahme werden meist Informationen bzw. frühere Annahmen benötigt. Damit werden zum Schließen von nur einer Annahme im allgemeinen weniger Informationen benötigt, als zum Ausschließen beider Annahmen, also zum Schließen eines Widerspruchs.

Dem Suchalgorithmus können mehr Informationen zur Verfügung gestellt werden, dies kann beispielsweise durch Heuristiken oder look ahead (Vorausschauen) geschehen. Die Bereitstellung dieser Informationen kostet allerdings zusätzliche Zeit. Die Zeit, die der Suchalgorithmus insgesamt benötigt, sollte aber möglichst klein sein.

5.2.2 Suche nach einer Antwortmenge

Meistens werden allerdings nicht alle Antwortmengen eines Programms gesucht, sondern nur Eine. Bei der Suche nach allen Antwortmengen geht es darum, möglichst viel von Suchbaum auszuschließen oder wegzuschneiden. Im Gegensatz dazu, geht es bei der Suche nach einer Antwortmenge darum, möglichst wenig vom Baum bei der Suche zu benutzen.

Wenn ein Programm sehr viele Antwortmengen hat, ist das bei der Suche nach allen Antwortmengen schlecht, da alle Antwortmengen gefunden werden sollen und der Suchraum damit wahrscheinlich größer wird. Bei der Suche nach einer Antwortmenge ist dies aber vorteilhaft, da sehr viele Wege zum Ziel führen und damit die Wahrscheinlichkeit, auf Abwegen zu geraten, geringer wird.

Die beiden Zielsetzungen unterscheiden sich also signifikant. Bei der Suche nach einer Antwortmenge ist es entscheidend, möglichst wenig Umweg zu machen. Dieser Umweg ergibt sich aus der Zahl der falschen Entscheidungen, multipliziert mit den durchschnittlichen Kanten des Unterbaums, der an dieser falschen Entscheidung hängt. Bei der Suche können maximal A mal wirklich falsche Entscheidungen getroffen werden, in dem Sinne, dass von einem Pfad abgewichen wird, der zu einer Antwortmenge führt. Diese falschen Entscheidungen sind von den wrong choices (falsche Entscheidungen) die smodels ausgibt, zu unterscheiden. Da smodels auch choices zählt, die im Suchbaum einer schon vorher gemachten falschen Entscheidung liegen und an denen man sich daher eigentlich nicht mehr falsch entscheiden kann.

Der Untersuchbaum einer falschen Entscheidung, ist der Untersuchbaum der durchsucht werden muss, bis sicher ist, dass er zu keiner Antwortmenge führt. Er ist also eine Art falscher Suchbaum, bei dem gezeigt werden muss, dass die Menge aller seiner Antwortmengen leer ist. In diesem Sinne sind hier doch noch die Überlegungen nützlich, die bei der Suche nach allen Antwortmengen gemacht wurden. Ein Unterschied ist allerdings, dass bei diesen Untersuchbäumen in der Regel schon Vorwissen vorhanden ist.

Es ist allerdings auch möglich, dass bei einem Suchalgorithmus nicht immer die ganzen falschen Untersuchbäume durchsucht werden. Es ist auch eine Suche möglich, bei der mehrere Pfade parallel durchsucht werden und die dann die zuerst gefundene Antwortmenge zurückgibt. Oder auch eine Suche, die aus den aktuell, noch nicht beendeten Pfaden, sich immer, mithilfe einer Heuristik, den Aussichtsreichsten für die nächste Operation aussucht. Beide Suchen haben den Vorteil, dass falsche Untersuchbäume zum Teil nur teilweise durchsucht werden, bevor eine Antwortmenge gefunden wird. Der Nachteil der Suchen ist, dass dabei wahrscheinlich auch mehrere weitere Untersuchbäume teilweise durchsucht werden, die weitere Antwortmengen enthalten. Da nur bei Untersuchbäumen, die vollständig durchsucht wurden, sicher ist, welche von ihnen Antwortmengen enthalten und welche nicht.

Das führt auch dazu, dass bei Programmen die keine Antwortmengen haben, unabhängig von der Art der Suche, der ganze Suchbaum durchsucht werden muss. Was zu dem Modell in Abschnitt 5.2.1.4 führt.

Wenn falsche Untersuchbäume vollständig durchsucht werden, ist K_{wc_n} die Anzahl der Kanten des falschen Untersuchbaumes, der, bei der Suche nach einer Antwortmenge, durch die falsche Entscheidung in Ebene n auftritt. Da nur eine Antwortmenge gesucht wird, kann in jeder Ebene maximal eine falsche Entscheidung auftreten. Wenn keine falsche Entscheidung in Ebene n auftritt ist $K_{wc_n} = 0$. Gibt es keine Antwortmenge ist natürlich jede Entscheidung eine falsche Entscheidung.

Danach ist die Anzahl der Kanten K^1 im Suchbaum bei der Suche nur einer Antwortmenge:

$$K^-(n) \quad = \quad \sum_{n=1}^{A}(1 + K_{wc_n}) \tag{5.17}$$

$$0 \leq \quad K_{wc_n} \quad \leq 2^{A-n+1} - 1 \quad n = 1 \ldots A \tag{5.18}$$

Mit wachsenden n nimmt K_{wc_n} wahrscheinlich mehr als exponentiell ab. Dafür gibt es zwei Argumente. Das Erste ist die auferlegte Begrenzung aus Formel 5.18. Der falsche Untersuchbaum kann nur noch die Kanten eines Baumes der Tiefe von $a - n$ haben, plus die eine Kante für die falsche Entscheidung. Das zweite Argument ist, dass in Ebene n schon $n - 1$ die Annahmen der Ebenen unter ihr als Vorwissen vorhanden sind. Wie schon in Abschnitt 5.2.1.4 beschrieben, begünstigt Vorwissen das Beschneiden des Untersuchbaums. Damit schrumpfen die falschen Untersuchbäume mit steigenden n noch stärker als durch 5.18 vorgegeben.

Weiterhin nimmt mit der Zunahme von Vorwissen, beziehungsweise n, auch die Wahrscheinlichkeit falsche Entscheidungen zu machen ab, da Vorwissen genutzt werden kann, sie zu vermeiden.

5.2.2.1 Weitere Antwortmengen suchen

Nachdem eine Antwortmenge gefunden wurde, können auch weitere Antwortmengen gesucht werden. Dazu können die Informationen, die beim Finden der schon gefundenen Antwortmengen gewonnen wurden, weiter verwendet werden. Das heißt unter anderem, dass schon durchsuchte Teilbäume nicht noch einmal durchsucht werden müssen. Es muss auch nicht mehr bei der Wurzel des Suchbaums begonnen werden, sondern es können, vom Blatt der letzten gefundenen Antwortmenge beginnend, die nicht durchsuchten Teilbäume zu den auf dem Pfad zur Wurzel liegenden choices benutzt werden. Dadurch ist die Anzahl der besuchten Kanten des Suchbaums, die zum Finden einer weiteren Antwortmenge besucht wurden, wahrscheinlich geringer, als die zum Finden der ersten Antwortmenge.

Nachdem alle Antwortmengen gefunden wurden, werden bei der Suche nach einer weiteren Antwortmenge alle noch verbleibenden, nicht durchsuchten Teilbäume, durchsucht und ausgegeben, dass keine weitere Antwortmenge existiert. Wenn angenommen wird, dass nur schlecht voraus gesehen werden kann, ob eine Entscheidung falsch oder richtig ist, dürften die noch übrig gebliebenen, nicht durchsuchten Teilbäume, in ihrer Kantenzahl, in der Größenordnung der Kantenzahl des Suchbaums für die erste Antwortmenge liegen.

5.3 Kurvenverhalten

Das Nachfolgende wurde schon größtenteils in Abschnitt 3 angesprochen, es soll aber hier noch weiter vertieft werden. Es geht dabei um einen Erklärungsversuch, der aber nicht unbedingt zutreffen muss.

Dabei sollte im Hinterkopf behalten werden, dass wenn die Suchbaumtiefe linear abnimmt, die Anzahl der Kanten im Suchbaum wahrscheinlich exponentiell abnimmt. Das Modell im Abschnitt 5.2.1.4 auf Seite 35 dient als Grundlage der Aussagen.

5.3.1 Kurvenverhalten SAT

Bei kleinen m/n (R/A) Faktor gibt es mehr Modelle, da wenige Klauseln (Regeln) pro Variable (Atom) vorhanden sind und damit auch weniger unvereinbare Atomkombinationen. Die SAT Formel ist dann wenig durch Klauseln beschränkt. Da es viele Modelle gibt, führen natürlich auch viele Pfade im Suchbaum zum Ziel (einem Modell), daher ist die Anzahl der falschen Entscheidungen, die getroffen werden können, geringer. Die durchschnittliche Tiefe pro Untersuchbaum dürfte allerdings größer sein, da es für getroffene Annahmen weniger Regeln gibt, die diese zu einem Widerspruch führen können.

Bei großen m/n Faktor gibt es wenige Modelle, da viele Klauseln (Regeln) pro Variable (Atom) vorhanden sind und damit auch viele unvereinbare Atomkombinationen. Die Atome der SAT Formel werden durch die vielen Klauseln stark begrenzt. Viele oder alle Wege im Suchbaum führen dann zu keinem Ziel, wenn aber falsche Entscheidungen gemacht wurden, kann dies viel schneller erkannt werden, da es viel mehr Klauseln gibt, die bestimmte Teilbelegung von Variablen ausschließen. Es werden wahrscheinlich oft falsche Entscheidungen getroffen, aber dann gibt es nur einen kleinen falschen Untersuchbaum.

Zwischen den beiden Bereichen, einem bestimmten „mittel“ m/n Faktor, beim Phasenübergang der Erfüllbarkeit, gibt es mittelmäßig viele falsche Entscheidungen und die falschen Untersuchbäume haben im Durchschnitt mittlere Tiefe. Dadurch wird der zu durchsuchende Untersuchbaum und damit auch der Berechnungsaufwand viel größer.

So entsteht das beobachtete leich-schwer-leicht Muster.

5.3.2 Kurvenverhalten ASP

Die Unterschied zwischen den hier untersuchten ASP Programmen und SAT Formeln der in 3.2 Seite 15 beschriebenen SAT Untersuchungen, ist einerseits die andere Bedeutung des „←“ Symbols und andererseits, dass es ein unnegiertes Kopfatom gibt.

Die andere Bedeutung des „←“ Symbols bewirkt, wie in Abschnitt 2.3.9 auf Seite 9 beschrieben, das einige Modelle keine Antwortmengen sind. Eine Folge davon ist, dass nur Atome die auch im Kopf einer Regel vorkommen, auch in einer Antwortmenge vorkommen können.

Das unnegierte Kopfatom sorgt dafür, dass in jeder Regel mindestens ein unnegiertes Atom vorkommt. Dadurch können keine Atome negiert als Fakten vorkommen und ein Atom, das als Fakt vorkommt, kann nicht widerlegt werden. Also wenn ein Atom als Fakt vorkommt, kommt es auch in allen Antwortmengen vor und es gibt keine Möglichkeit, dass es wegen eines Zirkelschlusses ungültig sein muss.

Diese Unterschiede werden durch die Schließoperationen berücksichtigt, die polynomiell realisierbar ist. Wenn angenommen wird, dass dadurch die Anzahl der choices und damit des Berechnungsaufwands sich nur im geringeren Maße verändert, dürfte der Berechnungsaufwand bei ASP sich ähnlich wie bei äquivallenten Problemen bei SAT verhalten, der Anteil der erfüllbaren Probleme aber anders. Wenn also der Berechnungsaufwand bei 3-SAT ein bestimmtes Verhalten zeigt, dürfte diese Verhalten auch bei 2-LP auftauchen. Das Verhalten des Anteils der erfüllbaren Probleme kann aber anders sein.

Neben diesen gibt es wohl auch noch den nachfolgenden Zusammenhang.

Durch längere Regeln muss mehr Vorwissen vorhanden sein, um über das Kopfatom zu schließen. Dadurch steigt die Anzahl der choices, da öfter Atome im Körper unbelegt sind.

Kapitel 6

Auswertung der Testläufe

Leider kann von Ergebnissen, die Beispiele bei einem Antwortmengensolver hervorbringen, nicht auf die Ergebnisse, die sie bei anderen Antwortmengensolver hervorbringen, geschlossen werden. Wenn ein Problem bei einem Antwortmengensolver doppelt so viele choice points hat, wie ein anderes Problem, muss diese Relation nicht bei einem anderen Antwortmengensolvern erhalten bleiben. Der Grund dafür können schon kleine Implementationsunterschiede sein. Wenn beispielsweise ein Antwortmengensolver die Atome als choice points nimmt die am häufigsten vorkommen und ein anderer Antwortmengensolver nicht, kann es schon Unterschiede geben.

All die kleinen Implementationsunterschiede zu berücksichtigen, ist allerdings unmöglich. Deshalb sind die Ergebnisse der Antwortmengensolver für einzelne Beispiele nicht unbedingt zu allgemeinen Aussagen über die Antwortmengensolver heranziehbar. Dies ist ein Gegenstand der nachfolgenden Untersuchungen.

Bei den einzelnen Testläufen wurde für einen Testpunkt dabei, soweit nicht anders angegeben, 1000 Antwortmengenprogramme generiert und ausgewertet.

Die Testpunkte sind meist großflächig erhoben worden, so dass möglichst alle relevanten Bereiche vorhanden sind und von mir auch schnell eine neue Untertestreihe ausgewertet werden konnte, z.B. Testreihen bei konstanten R/A Faktor. Durch das aufgetretene exponentielle Wachstum, sind allerdings einige Bereiche nur schwer zugänglich, so dass teilweise Testreihen abgebrochen wurden.

Auf der beigelegten DVD sind viel mehr Daten vorhanden, als hier erwähnt werden. Allerdings werde ich versuchen, alle wesentlichen Aspekte abzuhandeln.

Die Testreihen werden im Nachfolgendem mit Namen gekennzeichnet, die dem Verzeichnisnamen entsprechen, in dem ihre Daten abgelegt sind. Dieser Name wird als Eigenname verwendet.

In den Verzeichnissen zu den Testreihen sind zwei Unterverzeichnisse und eine Anzahl von Dateien zu finden. Im Unterverzeichnis „ergebnisse“ befinden sich für jeden generierten Testpunkt eine Datei mit den ermittelten Werten für die Antwortmengenprogramme. Das Unterverzeichnis oder zip Archiv „tabellen“ enthält eine Reihe von Unterverzeichnissen, jeweils für die Diagramme oder Auswertungen verschiedener Untertestreihen. Die Datei „info.txt“ im Testreihenstammverzeichnis enthält einige Informationen, z.B. über die Namensgebung von Verzeichnissen und Dateien. Die restlichen Dateien in diesem Verzeichnis, sind Implementierungsdateien für die Testreihe.

Die Testreihen wurden zum Einen auf meinen privaten Rechner, AMD Atlon 1800+ 512 MB Arbeitsspeicher unter Suse Linux 9.0, generiert. Bei diesem Rechner hatte ich die alleinige Zugangskontrolle, so dass die Zeitwerte in den Testläufen nicht durch Benutzer verfälscht wurden. Andere Testreihen wurden unter Benutzung, von Rechnern der Universität Potsdam erstellt. Diese Rechner konnten parallel von anderen Benutzern verwendet werden. Auch wurde eine Testreihe teilweise auf verschiedenen Rechnern zusammengestellt. So dass man, bei der Auswertung der Zeit, besondere Vorsicht walten lassen muss. Andere Parameter, wie choice points oder der Suchraum, sind davon aber nicht betroffen. Auch auf den Universitätsrechnern wurden die Testreihen unter neueren Linux Version erstellt, allerdings mit verschiednen.

Die Werte der Parameter wurden bei zwei Testreihen auf meinen privaten Rechner mit BMTool von Wolfgang Faber erfasst und sind in dessen Ausgabeform gespeichert. Bei den restlichen Testläufen wurden die Werte durch eigene Parser erfasst und in der BMTool Ausgabeform ausgegeben.

Alle Antwortmengenprogramme wurden zu Beginn von lparse übersetzt, dabei wurden teilweise auch die Programme vereinfacht, so dass die Parameterwerte bei der Generierung nicht die gleichen sind, wie die von den Antwortmengensolver ausgegebenen Werte.

Die Generierungsparameter sind implizit im Dateinahmen für die Testpunkte gespeichert. Für mehr Informationen sei hier auf die Infodatei in den entsprechenden Verzeichnissen verwiesen.

Parameter die bei den smodels Testreihen erfasst wurden, sind von der Ausgabe von smodels übernommen.
Zu diesen Parametern gehören:

- die Zeit (time)
- die Anzahl der choice points (Entscheidungspunkte)
- die Anzahl der wrong choices (falschen Entscheidungen), wobei von smodels hier jeder choice point gezählt wird, bei dem beide möglichen Belegungen ausprobiert werden
- die Anzahl der Antwortmengen
- die Anzahl der Wahrheitswertzuweisungen (truth assignments)
- die Anzahl der Atome des von smodels gelesenen Programms
- die Anzahl der Regeln des von smodels gelesenen Programms

Von der nomore++ Ausgabe wurden die folgenden Parameter gespeichert:

- die Zeit (time)
- die Anzahl der choice points (Entscheidungspunkte)
- die Anzahl der Antwortmengen
- die Anzahl der Atome des von nomore++ gelesenen Programms
- die Anzahl der Regeln des von nomore++ gelesenen Programms

Bei der Anzahl der Antwortmengen wurden, für die Testreihen auf meinem privat Rechner, von BMTool nur erfasst, ob es Antwortmengen gibt, aber nicht wie viele.

Leider wurden die Werte der Atom- und Regelanzahl nicht von Anfang an erhoben, so dass sie teilweise bei den Testreihen cases und casesN fehlen.

Die nachfolgenden Diagramme stellen nur eine kleine Auswahl aller generierten Diagramme dar. Sie sollen charakteristische Sachverhalte verdeutlichen, auf der beigelegten DVD sind weitere Diagramme zu den Testreihen zu finden. Für weitere Erläuterungen der Daten auf der DVD sei auf die „liesmich.txt“ und „info.txt“ Dateien verwiesen.

Da die Diagramme auf der DVD größtenteils automatisch generiert wurden, sind sie teilweise kritisch zu betrachten. Gerade die 3-dimensionalen Diagramme sehen teilweise seltsam aus, da gnuplot Schwierigkeiten hatte die Kurvenoberfläche aus den gegebenen Daten zu berechnen, da ihm wohl einige Zwischenwerte fehlten.

Alle getroffenen Aussagen betreffen natürlich, wenn nicht anders gekennzeichnet, immer nur die untersuchten Bereiche. Da Aussagen außerhalb dieser, aus nachfolgend erwähnten Gründen, nicht ohne weiteres möglich sind.

6.1 Feste Körperlänge

6.1.1 Testreihe cases mit smodels

Bei der Testreihe cases wurde, für die Generierung der Antwortmengenprogramme, das Modell für feste Körperlänge (fixed bodylength) aus 4.1.1 auf Seite 23 verwendet. Für die so generierten Antwortmengenprogramme wurden alle Antwortmengen mit smodels Version 2.27 berechnet.

Abbildung 6.1 zeigt die Anzahl der Antwortmengenprogramme des 2-LP Modells die erfüllbar sind, zwischen 10 und 1500 Regeln und 10 und 150 Atomen. Die Höhenlinien, in der Regel-Atom-Ebene des Diagramms, zeigen ein annähernd lineares Verhalten, wobei die Verlängerung des gradlinigen Teils in die Nähe des Koordinatenursprungs kommt. Der R/A Faktor sollte demnach auf diesen Höhenlinien ungefähr konstant sein. Dieser Sachverhalt unterstreicht die Aussagekraft des R/A Faktors.

Im Nachfolgendem ist mit dem Wachstum des Berechnungsaufwands das Wachstum der Berechnungsaufwandskurve oder Zeitkurve im Hinblick auf die Regel-

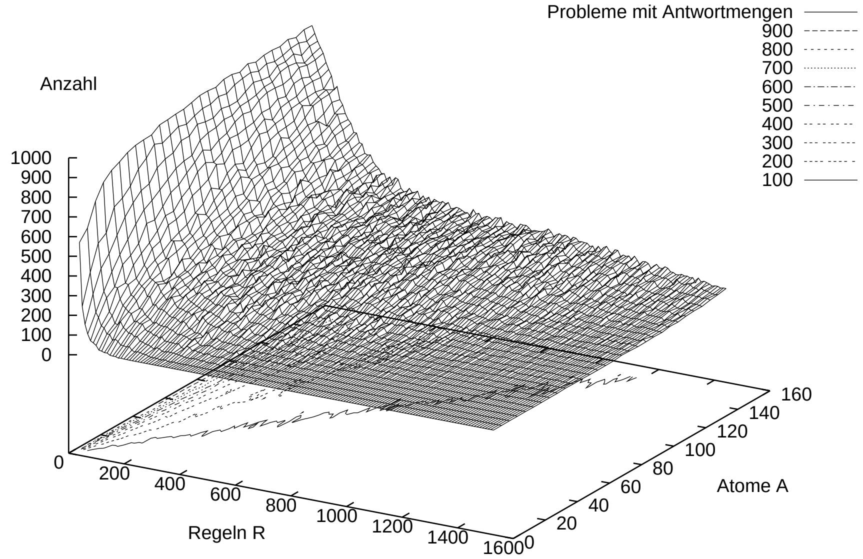

Abbildung 6.1: 2-LP Anzahl der erfüllbaren Antwortmengenprogramme

oder Atomanzahl bei einem konstanten Wert der Skalierungsformel, z.B. R/A, gemeint.

Auch ist zu erkennen, dass es anscheinend keine Phase gibt, in der die meisten Antwortmengen erfüllbar sind. Das die Anzahl der erfüllbaren Probleme mit der Zahl der Regeln von Anfang an sinkt, kann dadurch erklärt werden, dass mit der Zunahme der Regeln, auch die Zahl der ungeraden Zyklen zunimmt. Die Anzahl der geraden Zyklen nimmt natürlich auch zu, nur steigt die Erfüllbarkeit nicht dadurch, dass mehr Antwortmengen möglich sind. Mit Zunahme der Regelanzahl können nur mehr Widersprüche hinzukommen.

Der anscheinend lineare Verlauf der Höhenlinien lässt darauf schließen, dass sich an diesem Bild auch bei mehr Regeln oder Atomen nichts ändert.

Da bei dieser Testreihe alle Antwortmengen eines Programms berechnet wurden, ist die Anzahl der wrong choices gleich der Anzahl der choice points. Aussagen, die für die choice points gemacht werden, gelten also auch für wrong choices.

Abbildung 6.2 zeigt die durchschnittliche Anzahl der choices points die auf der Suche nach den Antwortmengen durchlaufen wurden. Deutlich ist hier der schwer Bereich zu erkennen. Das Maximum liegt dabei anscheinend auch auf einer Geraden, bei rund $R/A = 4$, wobei es mit steigender Regel- oder Atomzahl stark zunimmt.

Abbildung 6.3 zeigt den Median der Anzahl der choices points. Das Aussehen

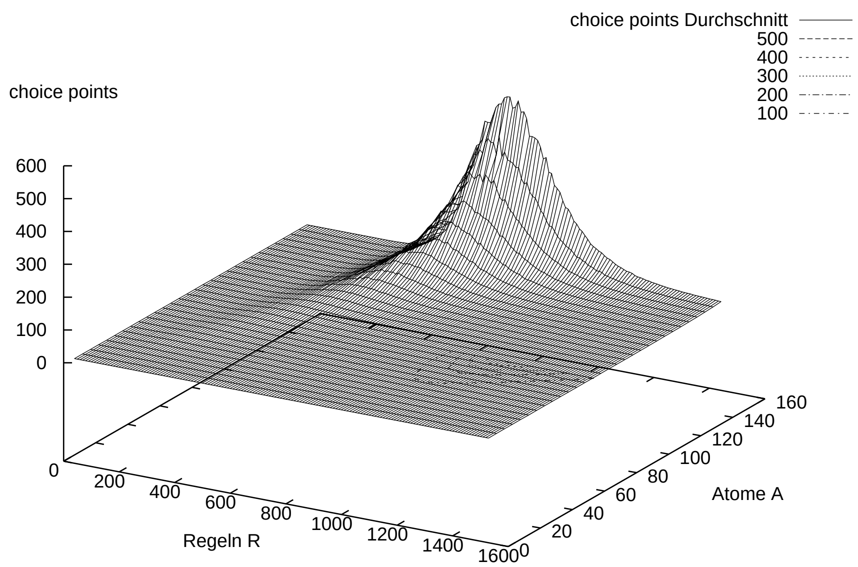

Abbildung 6.2: 2-LP durchschnittliche Anzahl der choice points

von Abbildung 6.3 ähnelt stark dem von Abbildung 6.2, nur dass die konkreten Werte anscheinend kleiner sind. Das lässt darauf schließen, dass sich Durchschnitt und Median ähnlich verhalten.

Abbildung 6.4 zeigt das Maximum der Anzahl der choice points für die Testpunkte. Auch sie ähnelt den beiden vorangegangenen Abbildungen 6.2 und 6.3. Allerdings ist sie wesentlich stärker zerklüftet, was daher rührt, dass hier nur ein Antwortmengenprogramm für die Punkte herangezogen wurde. Weiterhin scheint die Höhe der Werte deutlich über denen der Durchschnitts- und Medianswerte zu liegen. Für die Anzahl der maximalen choice points gibt es eine obere Grenze bei smodels, die bei 2^A liegt, im Durchschnitt oder Median braucht man wesentlich weniger, das Maximum kann sich ihr annähern.

Abbildung 6.5 zeigt das Minimum der choice points in den jeweiligen Testpunkten. Die Werte die erreicht werden, sind im Verhältnis zu den anderen drei Abbildungen 6.2, 6.3 und 6.4 sehr gering. Es kann davon ausgegangen werden, dass, wenn die Anzahl der generierten Testbeispiele pro Testpunkt erhöht wird, sich die Anzahl der minimalen choice points irgendwann überall bei 0 befindet. Das heißt, die leichten Probleme, die mit wenigen choice points zu lösen sind, nehmen vielleicht mit steigendem Durchschnitt der choice points ab, aber wenn genug Probleme generiert werden, werden immer welche zu finden sein.

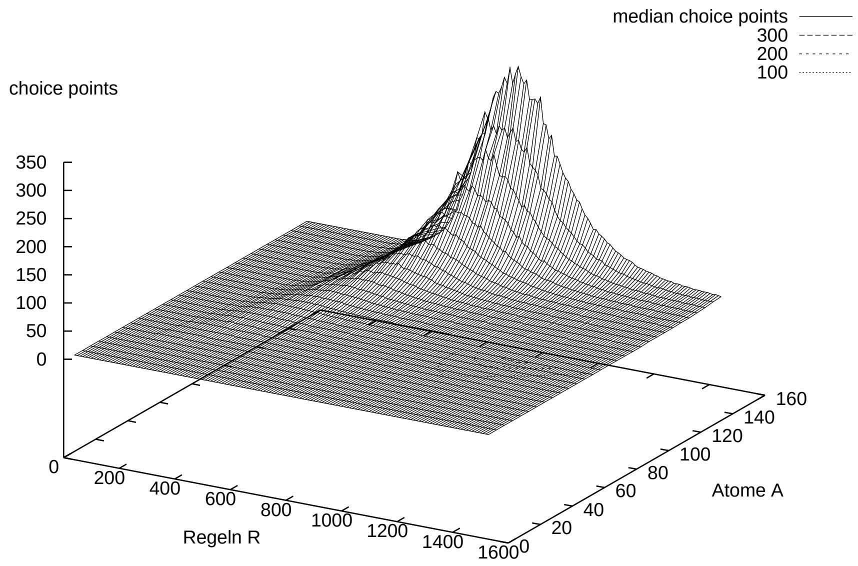

Abbildung 6.3: 2-LP Median der Anzahl der choice points

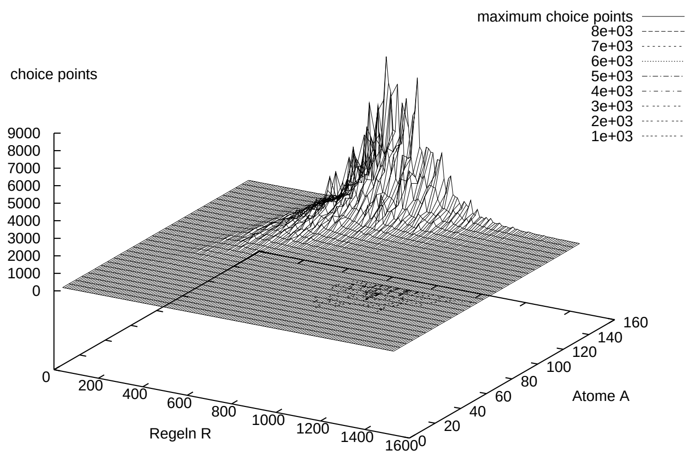

Abbildung 6.4: 2-LP maximale Anzahl der choice points

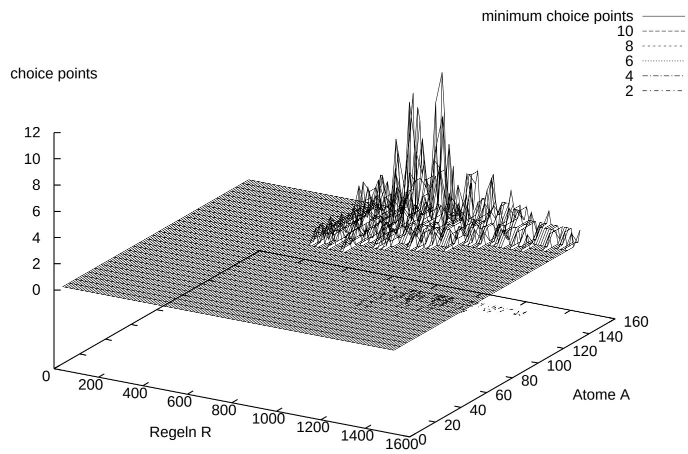

Abbildung 6.5: 2-LP minimale Anzahl der choice points

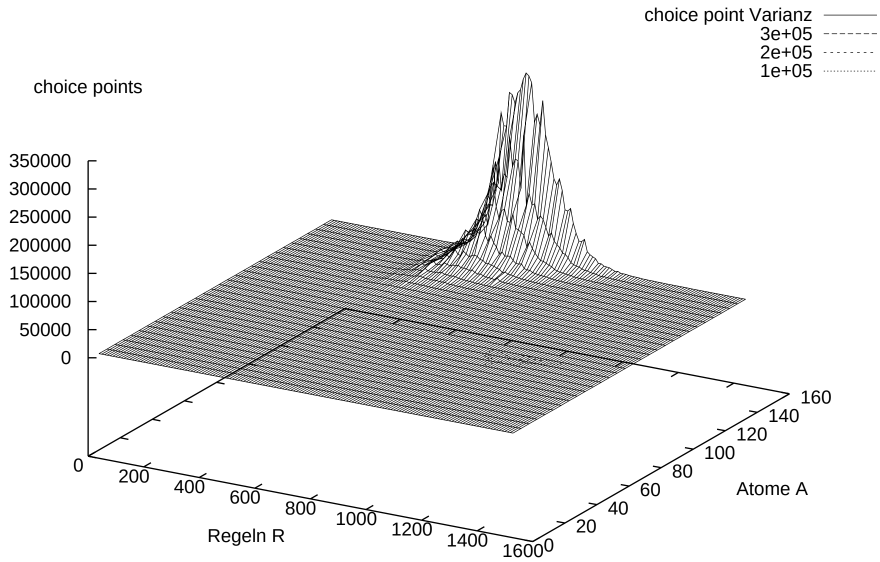

Abbildung 6.6: 2-LP Varianz in der Anzahl der choice points

Abbildung 6.6 zeigt die Varianz der choice points. Auch hier ist das gleiche Muster wie bei den drei Abbildungen 6.2, 6.3 und 6.4 zu finden.

Mit diesen Abbildungen für die choice points, kann auf die innere Verteilung der choice point Anzahl eines Testpunktes im schweren Bereich geschlossen werden. Es kann davon ausgegangen werden, dass diese Verteilung bei 0 beginnt. Sie ist asymmetrisch, wobei mehr Probleme im Bereichen mit wenigen choice points liegen und die wenigen Probleme mit mehr choice points den Durchschnitt nach oben drücken. Im Bereich von sehr vielen choice points liegen demnach nur wenige Probleme.

Wenn das Muster der Kurven 6.2 und 6.3 als Muster für die Schwere von Problemklassen angenommen wird, kann davon ausgegangen werden, dass mit der Schwere der Problemklasse das Maximum der Verteilung in ihr stark zunimmt.

Abbildung 6.7 zeigt die durchschnittliche Zeit, die zum Berechnen aller Antwortmengen benötigt wurde. Auch hier fällt die große Ähnlichkeit zu der Abbildung 6.2 auf. In der Tat ist dem auch so, dass bei allen Testreihen, das Verhalten der Zeiten, aber auch anderer Berechnungsaufwandsparameter, wie der Anzahl der wrong choices und Wahrheitswertzuweisungen, dem der choice points stark ähnelt.

Die Ähnlichkeit der Zeit und der choice points ist auch aus Abbildung 6.8 ersichtlich. Für dieses Diagramm wurden für die einzelnen Probleme die ermittelte Anzahl der choice points durch die ermittelte Zeit dividiert. Dargestellt ist jeweils der Durchschnitt für die einzelnen Testpunkte. Zu sehen ist, dass im schwer Be-

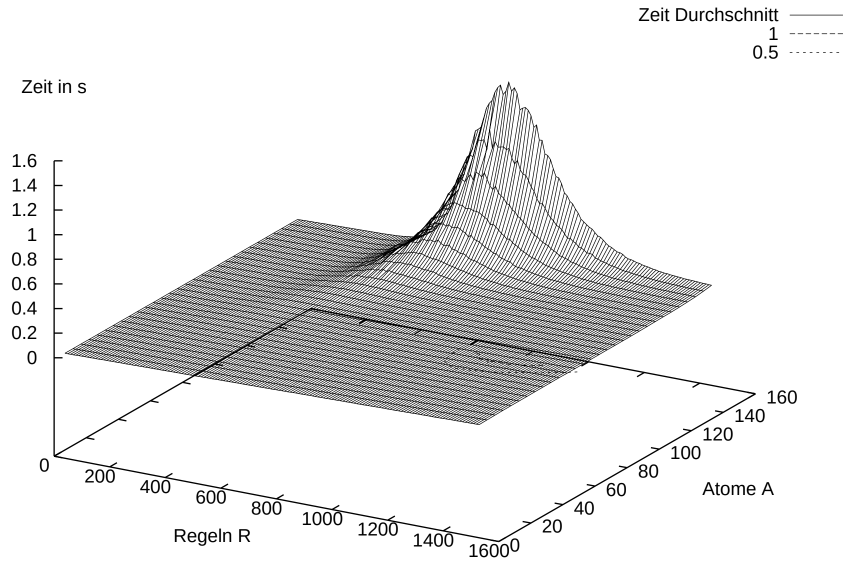

Abbildung 6.7: 2-LP durchschnittliche Zeit

reich dieser Quotient zwar stark zunimmt, aber der Verlauf des Maximums, bei Klassen größeren Probleme, stark abflacht. Da die Anzahl der choice points auf einer R/A Linie mindestens exponentiell wächst (siehe 6.10), der Unterschied zwischen choice point und Zeit aber nicht, dürfte sich die Zeit genauso verhalten wie die choice points.

Interessant ist, dass anscheinend mit der Schwere der Probleme auch die durchschnittliche Zeit pro choice point abnimmt.

Abbildung 6.9 zeigt die minimal benötigte Zeit. Deutlich zu sehen ist, die nahezu lineare Zunahme mit der Anzahl der Regeln. Der Unterschied zu den minimalen durchschnittlichen choice points aus Abbildung 6.5 ist also ein Faktor, der linear mit der Anzahl der Regeln wächst. Zu beachten ist, dass das lineare Wachstum der Zeit mit der Anzahl der Regeln, schon allein durch die Zeit entsteht, die zum Einlesen der Regeln benötigt wird.

Die Ähnlichkeit des Verhaltens der choice points und der Zeit ist also sehr groß, deshalb können durchaus die choice points zur weiteren Betrachtung des Berechnungsaufwands bei dieser smodels Version herangezogen werden. Allerdings ist das Verhalten, wie Abbildung 6.8 zeigt, nicht identisch, so dass zur Untersuchung des Berechnungsaufwands eines ASP Solvers allein die choice points wohl nicht ausreichen.

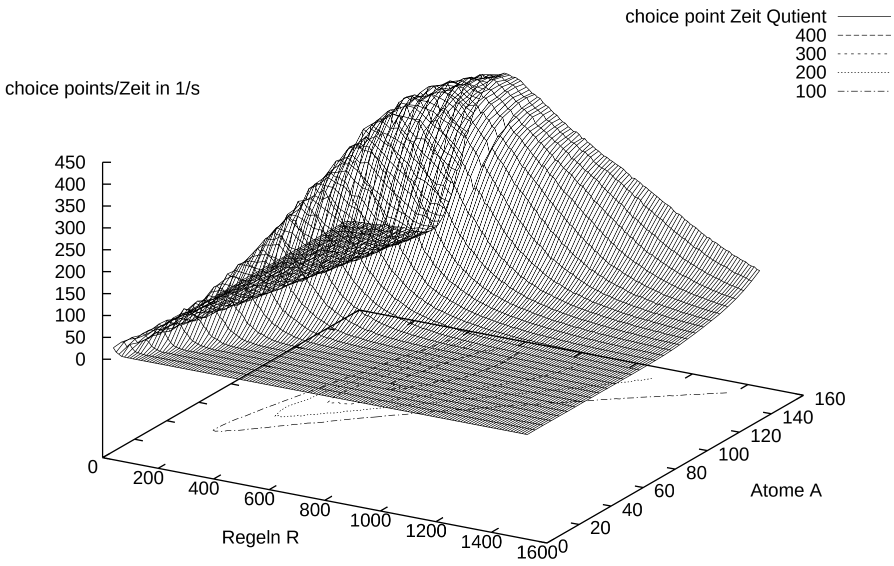

Abbildung 6.8: 2-LP durchschnittlicher choice point Zeit Quotient

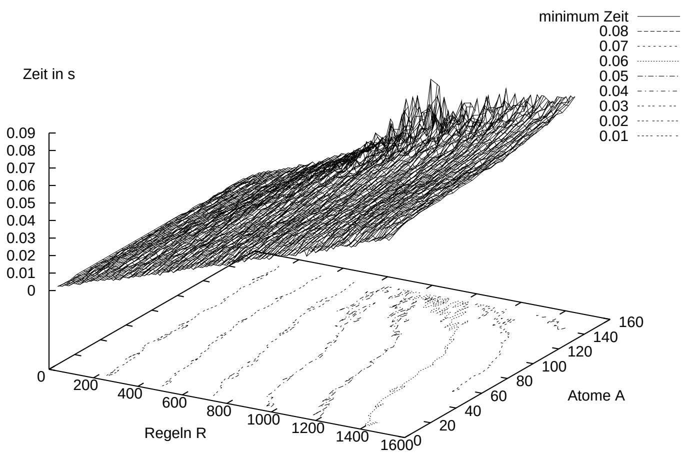

Abbildung 6.9: 2-LP minimale Zeit

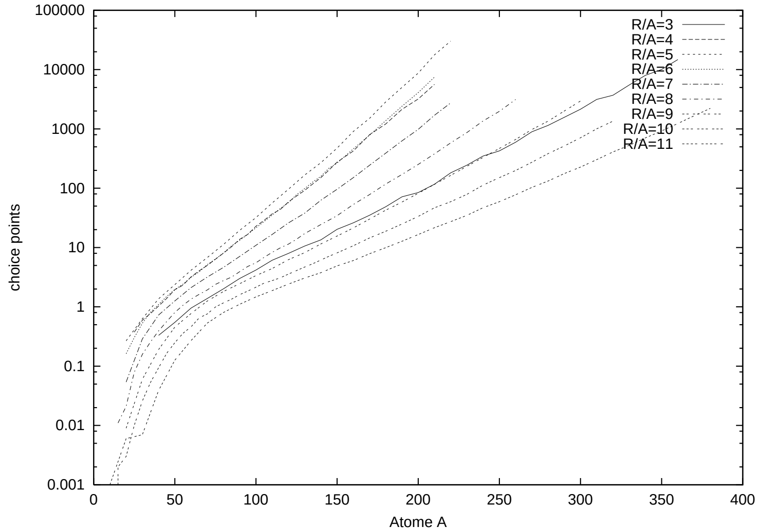

Abbildung 6.10: Durchschnittliche choice points für 2-LP für verschiedene R/A Faktoren

Das mindestens exponentielle Wachstum der Anzahl der choice points ist gut in Abbildung 6.10 zu sehen. In ihr wurden für die R/A Faktoren 3 bis 11 die Kurven dargestellt. Nach einem sehr starken Wachstum für Problemklassen mit durchschnittlich weniger als einen choice point, gehen die Kurven in eine eher stabile Phase über. Das Anstiegsverhalten aller Kurven ist sehr ähnlich. Wenn die Kurven in unterschiedlichen Diagrammen dargestellt werden, sind sie ohne eine Beschriftung der x-Achse nicht zu unterscheiden. Alle Kurven enden außerdem ungefähr in der gleichen Größenordnung von choice points.

Der Bereich unterhalb von einem choice point, kann als Anlaufphase gesehen werden, in der z.B. Heuristiken und look ahead noch nicht so viel Einsparungen bringen.

Außer bei 1-LP und 0-LP findet sich dieses Muster auch bei anderen k-LP Faktoren. Sehr kleine und große R/A Faktoren sind allerdings schwierig zu untersuchen, da sehr große Programme generiert werden müssen.

Demnach ist das Wachstum überall für k größer 1 mindestens exponentiell, nur der konkrete Anstieg unterscheidet sich. Das führt dazu, dass für eine Verdopplung der durchschnittlichen benötigten choice points (oder Zeit) die Programme nur um einen konstanten Faktor (an Regeln und Atomen, so dass R/A konstant ist) vergrößert werden müssen. Damit können auch in den leicht Bereichen die Probleme schnell schwer bis unlösbar werden. Exponentielles Wachstum ist sozusagen ein zweischneidiges Schwert, da es nicht nur bedeutet, dass der Aufwand exponentiell mit der linearen Zunahme der Programmgröße zunimmt, sondern auch exponentiell mit der linearen Abnahme der Programmgröße abnimmt. Es kann schnell der Eindruck entstehen, dass in bestimmten Bereichen (bei bestimmten R/A Werten) das Wachstum weniger als exponentiell wächst, nur weil die untersuchten Programmgrößen noch zu klein waren und der exponentielle Anteil nicht auffiel.

Für die Kurven aus Abbildung 6.10 und weiteren R/A Faktor Kurven wurden exponentielle Funktionen, die sich ihnen annähern, mit dem Auge approximiert. Bereiche unterhalb von einem choice point wurden dabei ignoriert. Die Funktionen haben die Form: 2^{R*a+b}, wobei a und b vom R/A Faktor abhängige Konstanten sind und R die Anzahl der Regeln ist. Demnach ist $1/a$ die Anzahl der Regeln um die die Programme vergrößert werden müssen, bei konstanten R/A Faktor, um die durchschnittliche Anzahl choice points zu verdoppeln.

Im Nachfolgenden seien einige aufgeführt, wobei $f_k(R/A, R)$ die Funktion für k-LP mit dem Faktor R/A ist.

$$\begin{aligned}
f_2(2,R) &= 2^{R*0.0065-4.0} \\
f_2(3,R) &= 2^{R*0.016-3.0} \\
f_2(4,R) &= 2^{R*0.0184-3.0} \\
f_2(5,R) &= 2^{R*0.016-3.0} \\
f_2(6,R) &= 2^{R*0.0125-3.0} \\
f_2(7,R) &= 2^{R*0.0094-3.0} \\
f_2(8,R) &= 2^{R*0.007-3.0} \\
f_2(9,R) &= 2^{R*0.0053-3.0} \\
f_2(10,R) &= 2^{R*0.0041-3.0} \\
f_2(11,R) &= 2^{R*0.0033-3.0} \\
f_2(12,R) &= 2^{R*0.0027-3.0}
\end{aligned}$$

Wegen $f_2(F,R) = 2^{R*a+b}$ und $R = F * A$ ist $f_2(F,A) = f_2(F,R/F) = 2^{A*F*a+b}$ für die gleichen a und b, wobei F für die entsprechenden R/A Faktoren steht. Wenn also ein Programm bei konstanten R/A Faktor um $R/A * a$ Atome vergrößert wird, verdoppelt sich im Durchschnitt der Berechnungsaufwand.

Abbildung 6.11 zeigt exemplarisch die Kurve für den R/A Faktor 5 mit der dazugehörigen Approximation $f_2(5,R)$. Bei anderen R/A Faktoren sind die Verhältnisse von Originalkurve zu Approximation ähnlich. Deutlich zu sehen ist, dass die Originalkurve anscheinend noch stärker als exponentiell wächst und einen nach oben offenen Bogen beschreibt. Damit dürfte smodels sogar noch die worst case (schlechtester Fall) Abschätzung für Antwortmengenprobleme (NP-vollständige Probleme) überschreiten. Diesem Sachverhalt sollte wohl bei zukünftigen Entwicklungen ein besonderes Augenmerk geschenkt werden. Es ist natürlich nicht ausgeschlossen, dass das Wachstum für noch größere Anzahlen von Regeln in ein rein exponentielles Wachstum übergeht. Der nach oben offenen Bogen könnte z.B. mit dem Nachlassen der Wirksamkeit der Heuristiken und des look ahead erklärbar sein. In diesem Fall wäre es vielleicht sinnvoll, diese so zu gestalten, dass sie

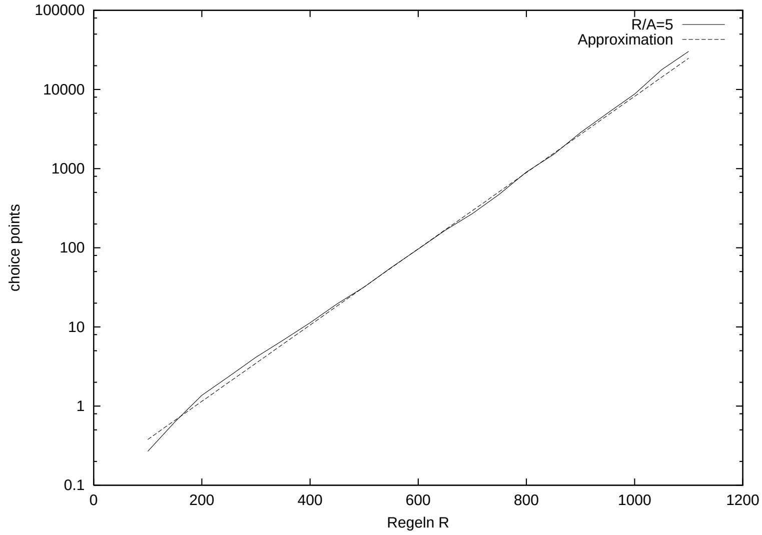

Abbildung 6.11: Durchschnittliche choice points für 2-LP bei den R/A Faktoren 5 mit Approximation

sich abhängig von der Programmgröße mehr Zeit nehmen, beziehungsweise mehr Aufwand betreiben.

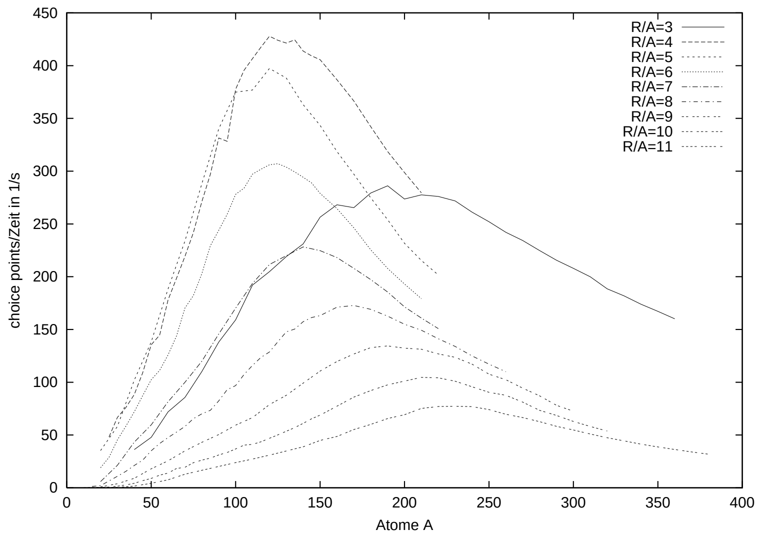

Abbildung 6.12: 2-LP choice points Zeit Quotient für verschiedene R/A Faktoren

Interessant ist auch Abbildung 6.12, sie zeigt die choice point Zeit Quotienten für die R/A Faktoren 3 bis 11. Bei allen Kurven deutlich zu sehen ist ein wenig-viel-wenig Muster. Während in der Anfangsphase durchschnittlich am wenigsten choice points pro Zeiteinheit besucht werden, steigt mit der Atomzahl die Zahl der choice points pro Zeiteinheit auf ein Maximum und sinkt danach wieder ab. In der 3-dimensionalen Abbildung 6.8 für den choice point Zeit Quotienten war dieses Verhalten noch nicht zu sehen, da das Maximum aller Kurven in Abbildung 6.12 anscheinend bei rund 150 Atomen liegt, dem Ende der Skala in Abbildung 6.8. Anzunehmen ist, dass sich die Kurven mit der Zeit weiter abflachen werden.

Über den Grund dieses wenig-viel-wenig Verhalten kann ich nur spekulieren. Dieses wenig-viel-wenig Verhalten könnte auf die choice point Vermeidungsstrategien (z.B. Heuristiken und lookahead) zurückzuführen sein. Während am Anfang, durch sie die Anzahl der choice points stark reduziert wird, versagen diese mit der Zeit. Sie können allerdings nur solange schlechter werden, bis sie so gut wie nichts mehr bringen. Dann kommt ein zweiter Effekt zu tragen. Bei größeren Programmen gibt es auch mehr Möglichkeiten weiter zu schließen, da mehr Regeln berücksichtigt werden müssen, so dass bei größer werdenden Programmen mehr Zeit zum Schließen zwischen den choice points verbraucht wird.

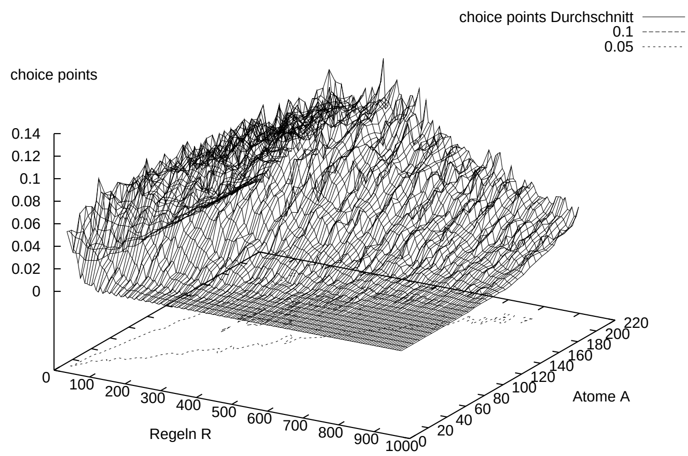

Abbildung 6.13: Durchschnittliche choice points für 1-LP

Abbildung 6.13 zeigt die durchschnittlichen choice points bei 1-LP. Deutlich zu erkennen ist ein leicht-schwer-leicht Muster, wobei das Maximum ungefähr bei dem $R/A = 2$ Faktor liegt. Allerdings ist kein exponentielles Wachstum zu sehen. Die annähernd linearen Höhenlinien auf dem Boden des Diagramms deuten auf ein lineares Wachstum des Berechnungsaufwandes hin. Die Werte liegen aber alle unter durchschnittlich einen choice point, so dass Aussagen, wie für Abbildung 6.10, schwierig sind. Beim Faktor $R/A = 1$ wurde eine Testreihe bis 8000 Regeln oder Atomen durchgeführt, die Anzahl der durchschnittlichen choice points blieb dabei ungefähr konstant und die durchschnittliche Zeit nahm linear zu. Bei anderen R/A Faktoren war das Verhalten ähnlich, auch wenn sie nicht so weit untersucht wurden. So dass aus dem untersuchten Bereich auf ein zumindest maximal polynomielles Wachstum geschlossen werden kann. Dies ist auch das zu erwartende Ergebnis, da 1-LP polynomiell lösbar sein sollte. Mit Gewissheit zu sagen das 1-LP polynomiell mit der verwendeten smodels Version lösbar ist, ist aus den oben aufgeführten Gründen allerdings unmöglich.

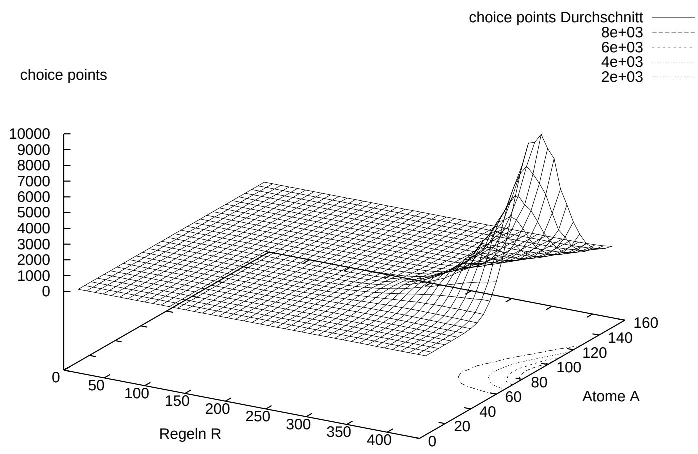

Abbildung 6.14: Durchschnittliche choice points für 3-LP

Abbildung 6.14 zeigt die durchschnittlichen choice points bei 3-LP. Wieder ist ein diesmal stark ausgeprägtes leicht-schwer-leicht Muster zu erkennen, dessen Maximum bei ungefähr $R/A = 5$ liegt. Das Diagramm zeigt nur bis zu 450 Regeln, da danach der Berechnungsaufwand für weitere Berechnungen zu groß wird. Der Anstieg dieser Kurve ist also um einiges stärker als der Anstieg bei 2-LP.

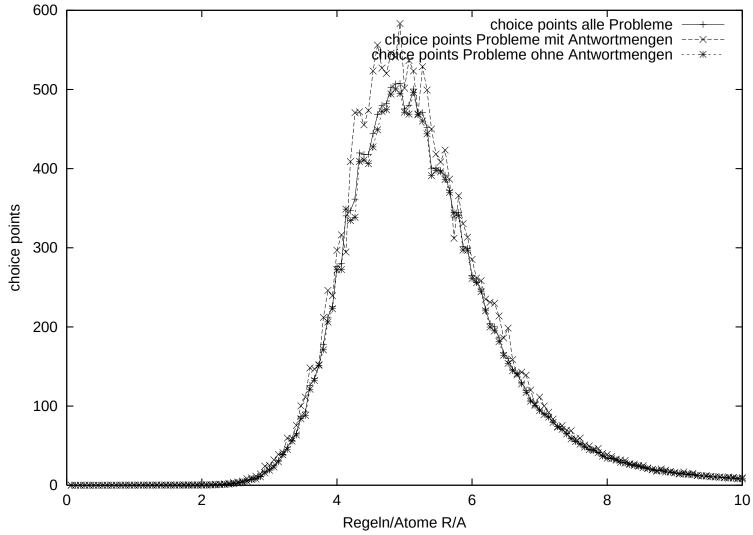

Abbildung 6.15: 2-LP choice points für erfüllbare und nicht erfüllbare Probleme bei 150 Atomen

Wie in Abbildung 6.15 zu sehen, unterscheidet sich die durchschnittliche Anzahl der choice points, welche für erfüllbare, nicht erfüllbare und alle Probleme benötigt werden, nicht sehr. Für erfüllbare Probleme wird im Durchschnitt dabei etwas mehr Zeit benötigt als für nicht erfüllbare. Das kann nach Abschnitt 5.2.1.4 damit erklärt werden, dass, wenn alle Antwortmengen gesucht werden, erfüllbare Probleme die sind, bei denen auch die letzte Ebene des Suchbaums angelaufen wird. Bei nicht erfüllbaren Problemen geschieht dies nicht. Somit ist der Suchbaum bei erfüllbaren Problemen etwas tiefer und es gibt im Durchschnitt etwas mehr choice points. Zu beachten ist hier noch, dass im schwer Bereich nur noch wenige erfüllbare Probleme vorhanden sind, wodurch die Kurve für erfüllbare Probleme auf der Basis von wenigen Problemen berechnet wurde und damit ungenauer ist.

Wie in Abbildung 6.16 zu sehen, liegt der Quotient von choice point für nicht erfüllbare Probleme, zu choice point für erfüllbare Probleme, knapp unter eins. In der Anfangsphase allerdings, bis $R/A = 2$, liegt die durchschnittliche Anzahl der choice points für nicht erfüllbare Probleme anscheinend bei 0 und nähert sich dann den Werten für erfüllbare Probleme an.

Abbildung 6.17 beinhaltet die Kurven für die choice points bei 50 Atomen für 1-LP, 2-LP und 3-LP. Zu sehen ist, dass sich das Maximum nach rechts, in Bereiche von höheren R/A Faktoren, verschiebt. Weiterhin gibt es wohl bei linearen Anstieg von k, einen exponentiellen Anstieg der Werte für choice points, in den unterschiedlichen Bereichen der Kurve, z.B. von Maximum zu Maximum.

Die Abbildung 6.18 zeigt die durchschnittliche Anzahl der Antwortmengen bei 150 Atomen und wurde aus der Testreihe casesAll entnommen. Die Testreihe

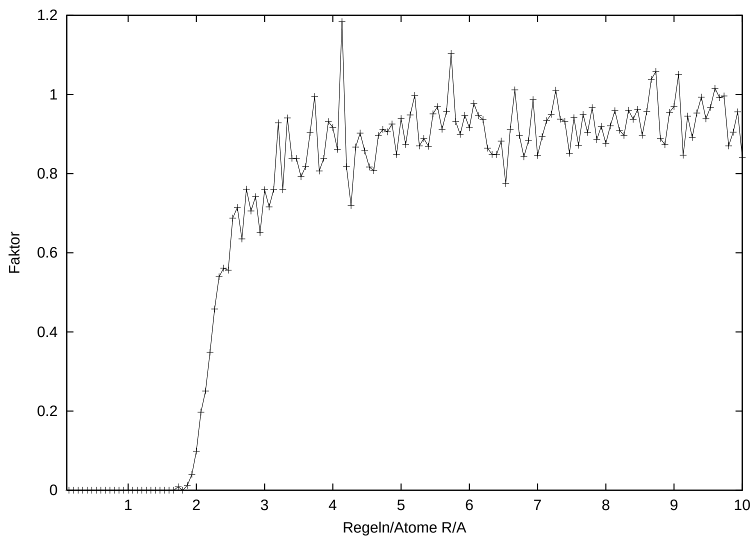

Abbildung 6.16: 2-LP Quotient der choice points für erfüllbare und nicht erfüllbare Probleme bei 150 Atomen

casesAll entspricht der Testreihe cases, mit den Unterschieden, dass bei ihr die Anzahl der Antwortmengen mit erhoben wurde, sie auf den Universitätsrechnern lief und sie mit weniger Testpunkten erhoben wurde. Deutlich zu sehen ist ein sehr starker Abfall der Antwortmengenzahl mit steigendem Faktor R/A.

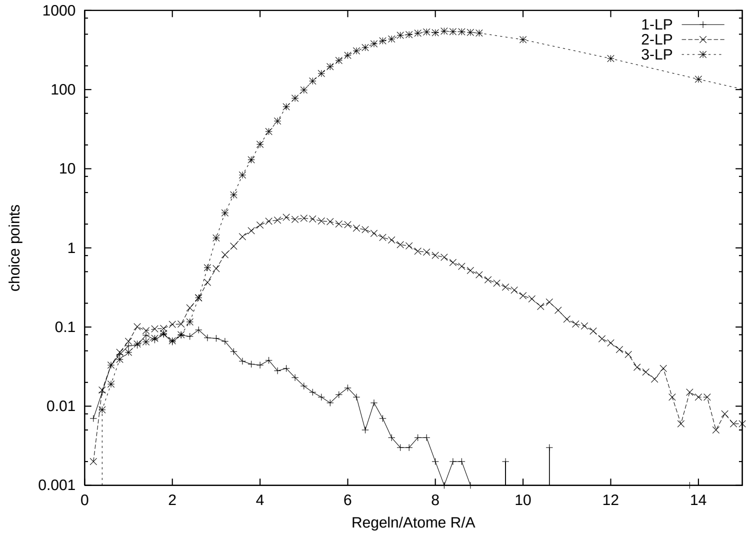

Abbildung 6.17: 1-LP, 2-LP und 3-LP choice points bei 50 Atomen

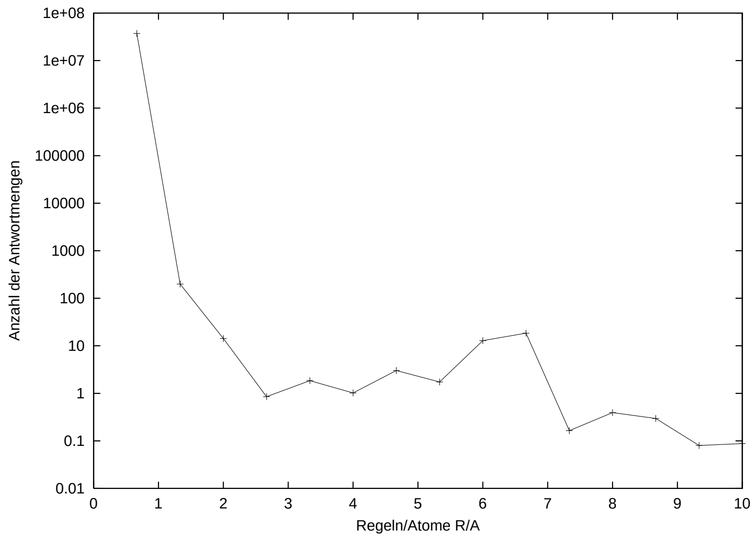

Abbildung 6.18: Anzahl der Antwortmengen für 2-LP bei 150 Atomen

6.1.2 Testreihe cases1 mit smodels

Die Testreihe cases1 entspricht der Testreihe cases, mit weniger Testpunkten und es wurde nur nach einer Antwortmenge gesucht. Auch cases1 wurde auf meinen Heimrechner erhoben.

Die Erfüllbarkeitsanteile entsprechen natürlich denen aus cases.

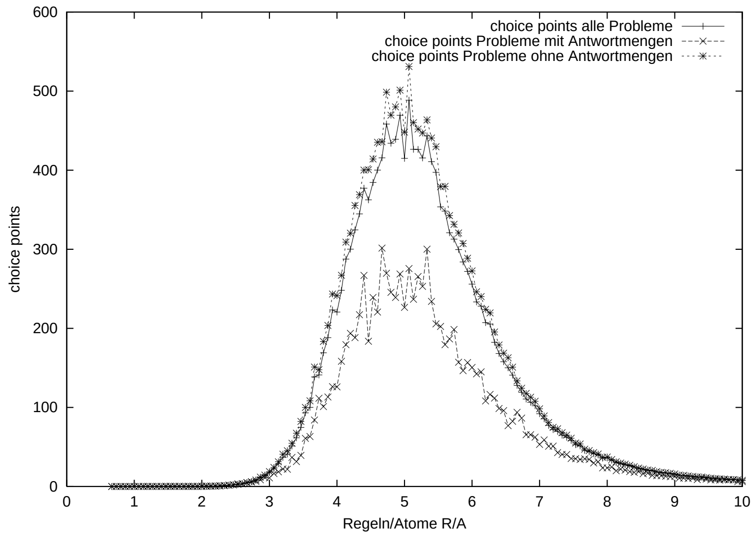

Abbildung 6.19: Anzahl der choice points der erfüllbaren und unerfüllbaren Probleme bei 2-LP für 150 Atome

Die Abbildung 6.19 zeigt die durchschnittliche Anzahl der choice points von erfüllbar, unerfüllbar und allen Problemen für 2-LP bei 150 Atomen. Die Ähnlichkeit zur Abbildung 3.5 auf Seite 20 aus [21] Seite 82 ist auffallend. Ich kann also seine Ergebnisse für k-LP trotz leicht unterschiedlicher Modelle bestätigen. Anscheinend sind die Effekte von doppelten Atomen im Körper und doppelten Regeln vernachlässigbar, zumindest im untersuchten Bereich. Die Gründe für den Kurvenverlauf wurden schon in Abschnitt 5.2.2 auf Seite 35 erläutert und werden hiermit bestätigt.

Die Kurve von unerfüllbaren Problemen ähnelt dabei, wie zu erwarten, der von 6.15, bei der alle Antwortmenge berechnet wurden.

Wie in Abbildung 6.20 zu sehen, liegt der Quotient von choice point für nicht erfüllbare Probleme, zu choice point für erfüllbare Probleme rund bei 1.5. In der Anfangsphase allerdings, bis $R/A = 2$, liegt die durchschnittliche Anzahl der choice points für nicht erfüllbare Probleme anscheinend bei 0 und nähert sich dann dem Wert 1.5 an, wonach er bei einem Maximum im schweren Bereich bei rund $R/A = 4$ wieder leicht abnimmt.

Die Abbildung 6.21 zeigt die durchschnittliche Anzahl der choice points von erfüllbar, unerfüllbar und allen Problemen für 2-LP bei dem Faktor $R/A = 5$. Die

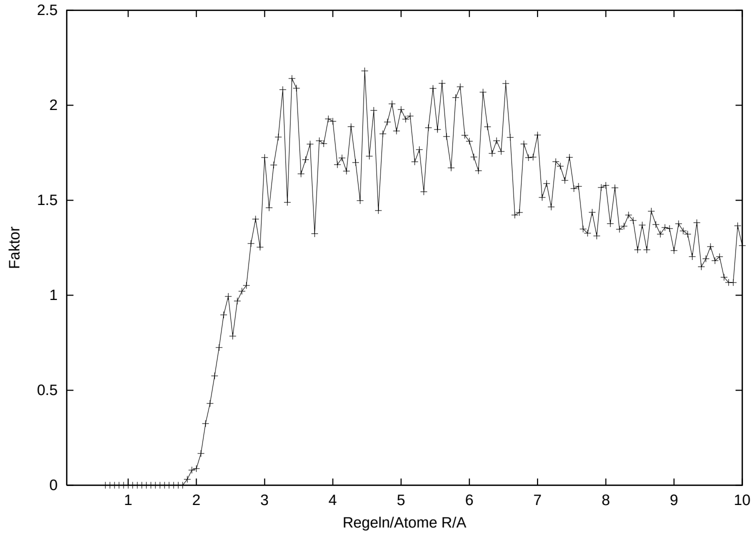

Abbildung 6.20: Quotient von choice points der erfüllbaren und unerfüllbaren Probleme bei 2-LP und 150 Atomen

drei Kurven zeigen, abgesehen von den konkreten Anstiegswerten, ein ähnliches Verhalten.

Durch die Abbildung 6.22 wird dies bestätigt. Hier wurde für die einzelnen Testpunkte der Logarithmus der Anzahl der choice points bei unerfüllbaren Problemen durch den Logarithmus bei erfüllbaren Problemen geteilt. Diese Kurve ist sozusagen der Quotient der Exponentenfunktionen der choice point Wachstumsfunktionen für unerfüllbare und erfüllbare Probleme.

Nach anfänglichen stärkeren Schwankungen bei unter 60 Atomen, pendelt sich die Kurve ziemlich stabil bei einem Wert von rund 1.1 ein. Das heißt, das Wachstum der Kurven unterscheidet sich nur durch einen konstanten Faktor im Exponenten (a in 2^{R*a+b}).

Das Wachstumsverhalten ist also weitestgehend unabhängig davon, ob nur eine oder alle Antwortmengen gesucht werden, und ob die Probleme erfüllbar sind oder nicht.

Ein weiterer Unterschied zu der Testreihe cases ist, dass nun die choice points und wrong choices Werte nicht mehr identisch sind.

In Abbildung 6.23 sind die durchschnittlichen Unterschiede zwischen choice points und wrong choices für 2-LP bei 150 Atomen aufgeführt. Es gibt zwar ein wenig-viel-wenig Muster, aber im Vergleich zu den tatsächlich gemachten choice points, sind die Werte gering und die Kurve weniger ausgeprägt. Die Anzahl der wrong choices liegt nur etwas unter der Anzahl der choice points. Wobei in dem leicht Bereichen der Unterschied noch größer ist, dort werden also proportional weniger wrong choices gemacht. Bedingt durch die kleineren Werte in den untersuchten leicht Bereichen, kann dies aber nicht verallgemeinert werden.

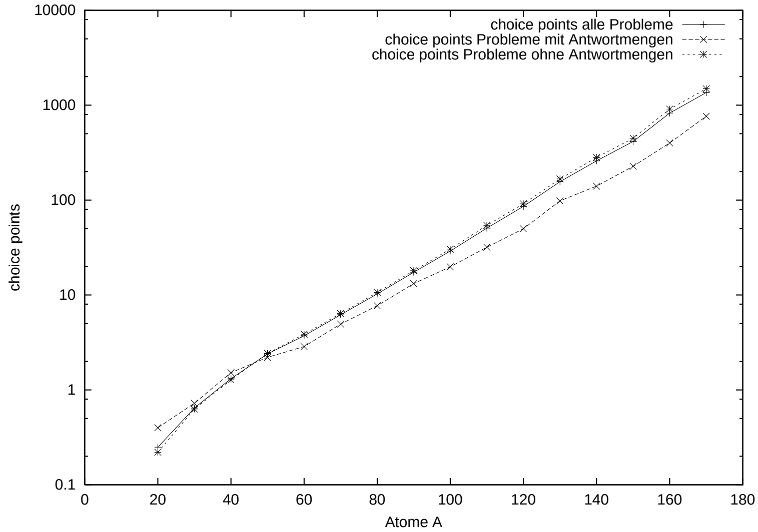

Abbildung 6.21: Anzahl der choice points der erfüllbaren und unerfüllbaren Probleme bei 2-LP für $R/A = 5$

In Abbildung 6.24 ist die gleiche Kurve für einen konstanten Faktor $R/A = 5$ logarithmisch aufgetragen. Demnach wächst der Unterschied choice points und wrong choices ungefähr exponentiell. Aber auch hier ist der Unterschied nur gering, so dass sich die Kurven für choice points und wrong choices sehr ähnlich verhalten.

Von den oben aufgeführten Unterschieden abgesehen, sind die Werte und Kurven für die Testreihe cases1 sehr ähnlich denen zu cases.

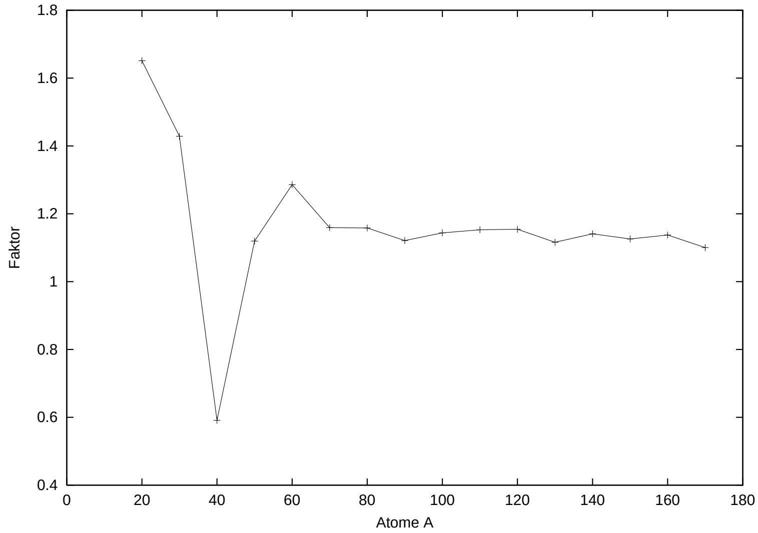

Abbildung 6.22: Quotient von log choice points der unerfüllbaren und erfüllbaren Probleme bei 2-LP und $R/A = 5$

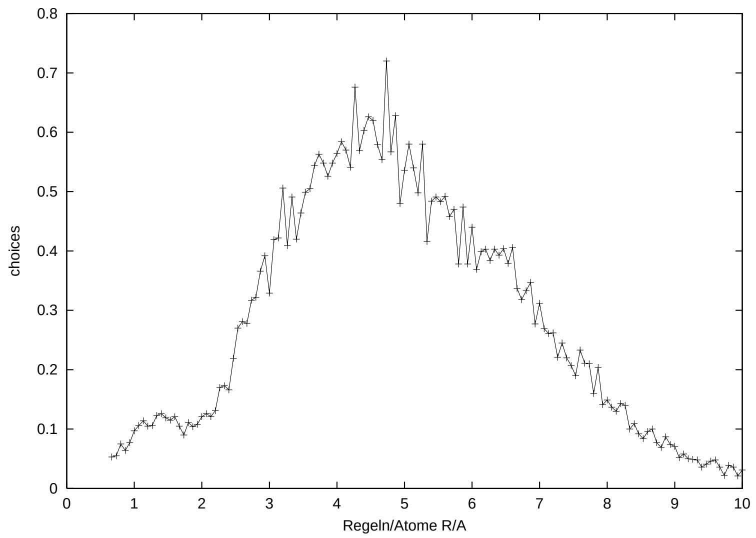

Abbildung 6.23: Differenz choice points und wrong coices bei 2-LP und 150 Atomen

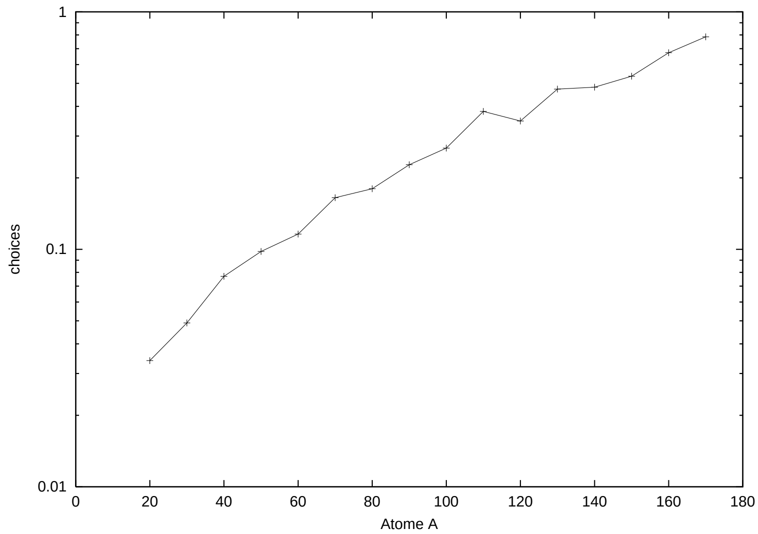

Abbildung 6.24: Differenz choice points und wrong coices bei 2-LP und $R/A = 5$

6.1.3 Untersuchung eines Testreihenpunktes bei cases

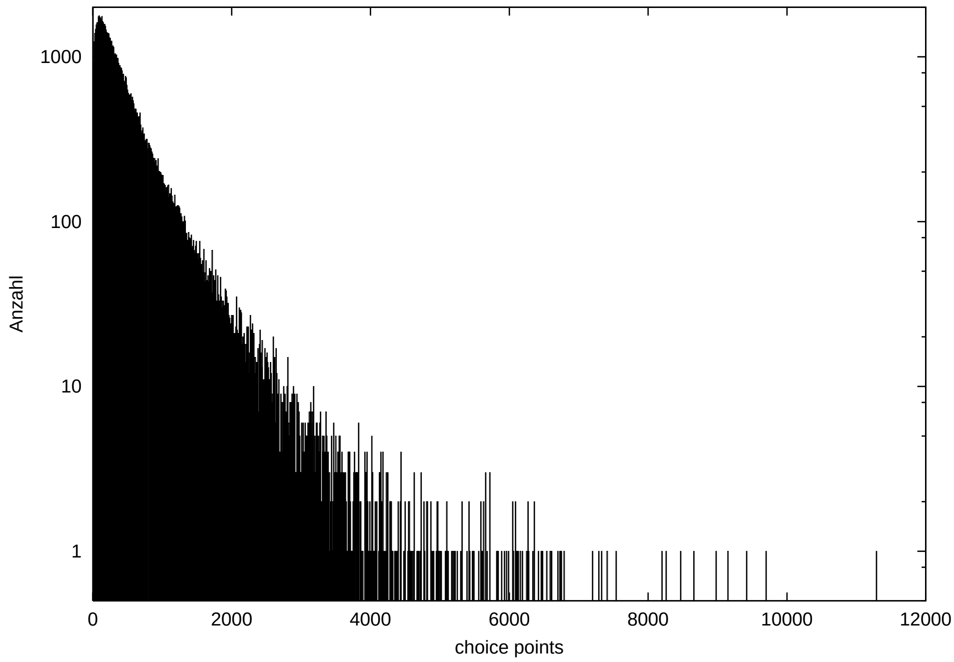

Abbildung 6.25: Klassen für choice points bei 2-LP für 150 Atome und 750 Regeln

Um die Vermutungen über die innere Verteilung eines Testpunktes zu bestätigen, wurden für verschiedene Testpunkte mehrere Probleme generiert und die Ergebnisswerte in mehrere Klassen aufgeteilt. In Abbildung 6.25 ist eine carakteristische Kurve bei 2-LP für 150 Atome und 750 Regeln für 88828 generierte Probleme zu sehen. Gesucht wurde dabei nur eine Antwortmenge, was der Testreihe cases1 entspricht. Angegeben ist die Anzahl der Probleme die choice points in den entsprechenden Klassen hatten. Eine Klasse umfasst dabei choice points in einem Abstand von 10, z.B. die Anzahl der Probleme mit 100 bis 110 choice points. Die y-Achse der Anzahlen ist logarithmisch eingeteilt.

Diese Kurve bestätigt die Vermutungen über die Verteilung innerhalb der Testpunkte. Nach einem sehr starken anfänglichen Anstieg der Anzahl, hat die Kurve ein Maximum bei relativ wenigen choice points. Danach sinkt die Kurve weniger stark und gleitet in hohen choice point Werten aus.

Bei anderen Testpunkten und Berechnungsaufwandsparametern (Zeit, wrong choices usw.) sieht die Kurve ähnlich aus, ist aber eventuell mehr gedehnt, beziehungsweise gestaucht, und eventuell mehr oder weniger ausgeprägt. Dabei ist die Dehnung der Kurve im Bereich der größeren Werte stärker. Mit den Werten für Durchschnitt, Median, Maximum und Minimum können die Veränderungen abgeschätzt werden.

6.2 Gemischte Körperlänge

6.2.1 Testreihe caseMix mit smodels

Die Testreihe caseMix wurde mithilfe von smodels Version 2.28 und den Programmgenerierungsmodell für gemischte Körperlänge aus Abschnitt 4.1.2 von Seite 24 generiert. Gesucht wurde jeweils nur immer eine Antwortmenge. Die Testreihe lief auf den Universitätsrechnern.

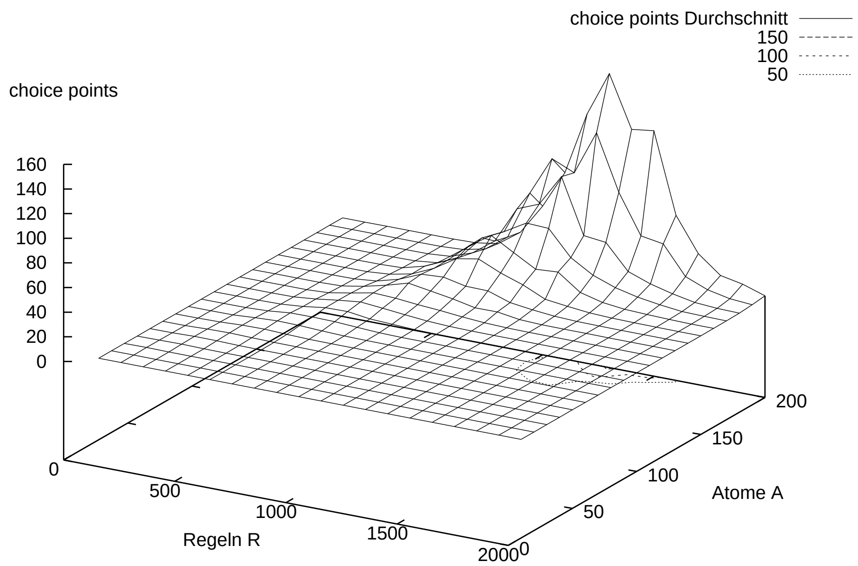

Abbildung 6.26: Durchschnittliche choice points bei durchschnittlich 3.8 Literalen

Abbildung 6.26 zeigt die durchschnittlichen choice points bei durchschnittlich 3.8 Literalen pro Regelkörper. Auch hier wiederholt sich das Muster aus Abbildung 6.2 (Seite 44) und 6.14 (Seite 54). Die Werte sind geringer als bei 3-LP. Probleme mit gemischter Körperlänge sind anscheinend leichter zu berechnen. In der Tat wiederholen sich die Muster für choice points, wrong choices, Zeit und Wahrheitswertzuweisungen, von cases1 mit niedrigeren Werten. So dass, die Ergebnisse für diese Parameter von cases, mit den Einschränkungen von cases1, auch für caseMix gelten.

Das Berechnungsaufwandswachstum ist bei caseMix geringer als bei cases1. Der Grund dafür ist mit dem Modell aus Abschnitt 5.2.1.4 auf Seite 35 gut zu erklären. Die Programme von caseMix sind eine Mischung aus Regeln mit unterschiedlicher Körperlänge. Darin sind auch 0-LP Regeln, also Fakten, und 1-LP Regeln enthalten. Fakten benötigen keine Informationen um Informationen zu liefern, sie legen also schon am Anfang der Suche Atome als Wahr fest, ohne das choices nötig wären. Regeln mit Körperlänge 1 (1-LP) können mit wenig Aufwand weitere Informationen liefern. Dieses gibt dem Solver bei gemischter Körperlänge des Generierungsmodells aus 4.1.2 einen Vorsprung. Im Gegensatz zu dem k-LP Modell für feste Körperlänge, mit k größer 1, können schon am Anfang mit wenig Aufwand Informationen gewonnen werden, welche die weitere Suche vereinfachen.

Das dennoch exponentielle Wachstum kommt wohl von dem Anteil der Regeln mit Körperlänge größer 1, bei denen die zusätzlichen Informationen nur zum Teil oder gar nicht weiter helfen.

Im Nachfolgenden werden nicht weiter die Ähnlichkeiten beleuchtet, sondern neue Muster betrachtet. Für die Verifizierung der Ähnlichkeiten, sei auf die beiliegende DVD verwiesen.

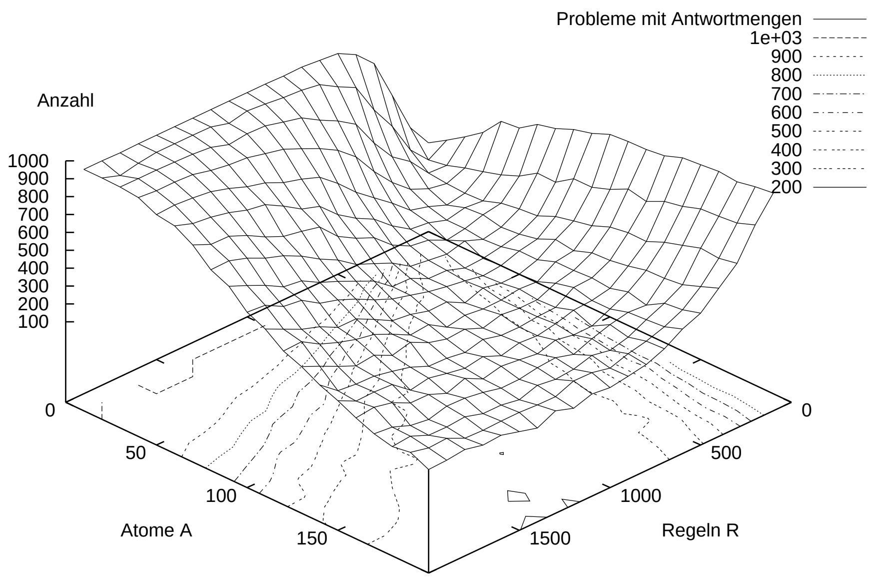

Abbildung 6.27: Anteil der erfüllbare Probleme mit durchschnittlich 3.8 Literalen

Abbildung 6.27 zeigt die erfüllbaren Probleme bei durchschnittlich 3.8 Literalen. Hier tritt ein deutlicher Unterschied zu den Kurven (6.1) bei konstanter Körperlänge zu Tage. Es gibt nicht nur viele erfüllbare Probleme bei wenigen Regeln pro Atom, sondern auch bei vielen Regeln pro Atom. Auch dies liegt wohl am Anteil der Regeln die Fakten darstellen. Fakten sind nicht widerlegbar. Wenn also ein Programm die richtigen Fakten beinhaltet, um die Widersprüche zu beheben, ist es erfüllbar. Mit Zunahme der Regeln nimmt auch die Zahl der Fakten zu. Damit werden mehr Programme erfüllbar, die es ohne sie nicht wären. Bei wenigen Regeln pro Atom bewirkt die Zunahme der Regeln allerdings, wie bei cases, eine Abnahme der erfüllbaren Programme, da der Anteil der Fakten noch nicht ausreicht um mehr Programme erfüllbar zu machen.

Die Zunahme der Fakten bei vielen Regeln pro Atom ist auch ein Grund für den in 6.26 zu sehenden niedrigen Berechnungsaufwand in diesem Bereich. Alles was aus Fakten zu schließen ist, benötigt nicht viel Berechnungsaufwand.

Auch hier sind, wie in cases, die Höhenlinien wieder annähernd Geraden, die in Richtung Ursprung verlaufen. Was für die Ähnlichkeit zu cases und die Aussagekraft der Skalierungsformel R/A auch bei caseMix spricht.

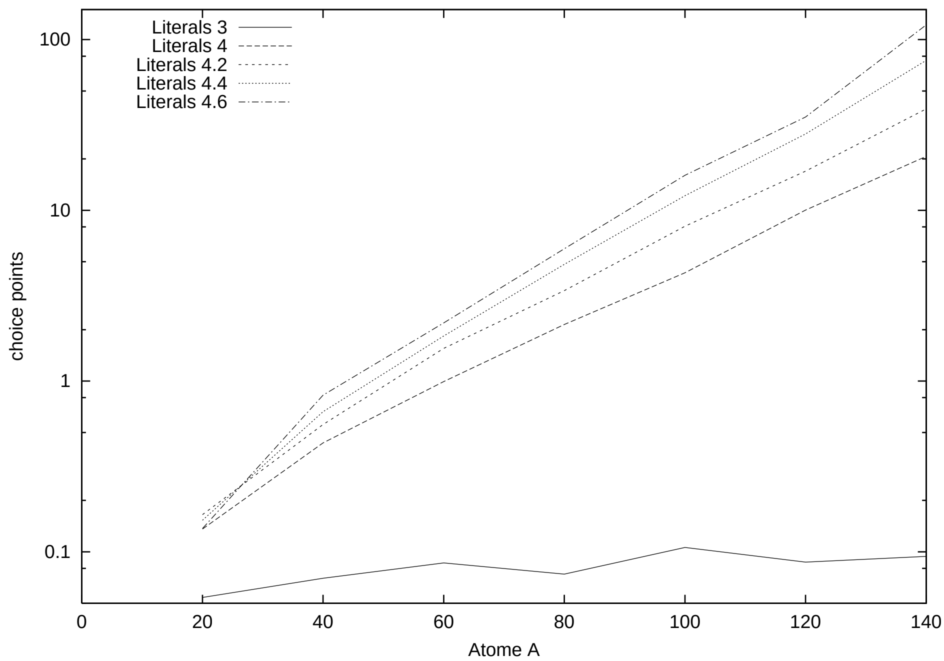

Abbildung 6.28: Durchschnittliche choice points bei unterschiedlichen Literalen und $R/A = 5$

Abbildung 6.28 zeigt für die durchschnittliche Literalzahlen von 3, 4, 4.2, 4.4 und 4.6 die durchschnittlichen choice points, bei dem $R/A = 5$ Faktor. Bei allen Literalzahlen gibt es anscheinend ein exponentielles Wachstum, auch wenn es bei 3 relativ schwach ausfällt.

Mit steigender Zahl der durchschnittlichen Literale, wird der Anstieg der Wachstumskurven größer.

Leider ist ein 3-dimensionales Diagramm, das als zwei seiner Achsen Literale und choice points besitzt, wegen des starken Anstiegs der choice points mit der Literalzahl, wenig hilfreich. Deshalb nähere ich mich dem Problem von einer anderen

Seite.

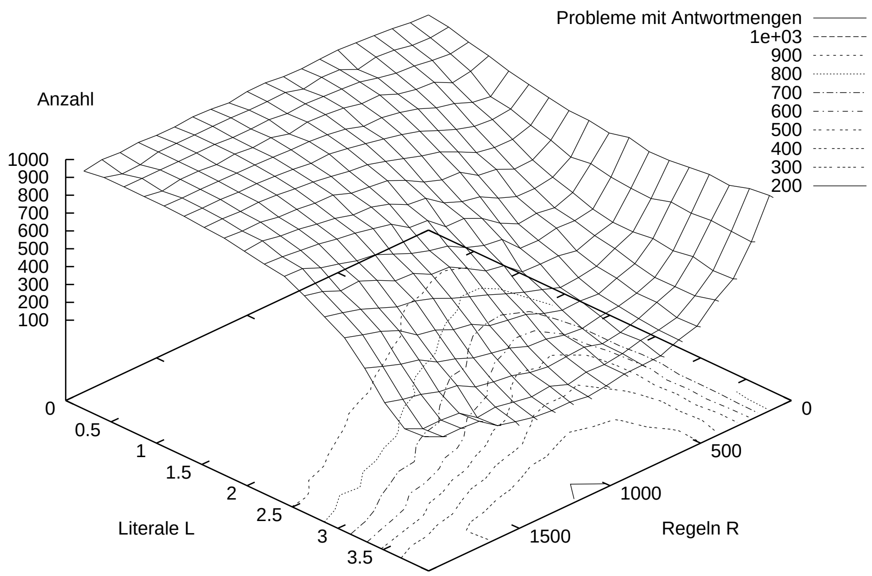

Abbildung 6.29: Anteil der erfüllbare Probleme bei unterschiedlichen Literalen und $A = 150$

Die in Abbildung 6.29 dargestellten Anzahlwerte von erfüllbaren Problemen, zu Regeln- und Literalzahlen bei 150 Atomen, sind gut einsichtig. Das Diagramm ähnelt in gewisser Weise dem aus 6.27. Für das viel-wenig-viel Muster, wenn die Literalzahl constant ist, ist wohl die gleiche Erklärung wie bei 6.27 anzuwenden. Mit sinkender Literalzahl nimmt der Anteil der erfüllbaren Probleme zu, da der Anteil der kurzen Regeln und damit auch der Fakten zunimmt. Um so höher die durchschnittliche Literalzahl wird, um so mehr lange Regeln gibt es und um so schwerer ist ein Programm zu erfüllen.

Mithilfe des Zusammenhanges der nächsten Kurve, kann diese Kurve zum Einschätzen des Verhaltens des Berechnungsaufwands bei unterschiedlichen Literalzahlen genutzt werden.

Das in Abbildung 6.30 zu sehende Verhalten ist charakteristisch für caseMix. Es zeigt die durchschnittlichen choice points und erfüllbaren Probleme bei durchschnittlich 3 Literalen und 150 Atomen. Das Maximum der choice points liegt in der Nähe des Minimums der erfüllbaren Probleme. Da dieses Verhalten in allen bei casesMix untersuchten Bereichen auftritt, kann davon ausgegangen werden, dass das Maximum des Berechnungsaufwands, bei den Voraussetzungen wie bei caseMix, immer in der Nähe des Minimums der Anzahl der erfüllbaren Probleme

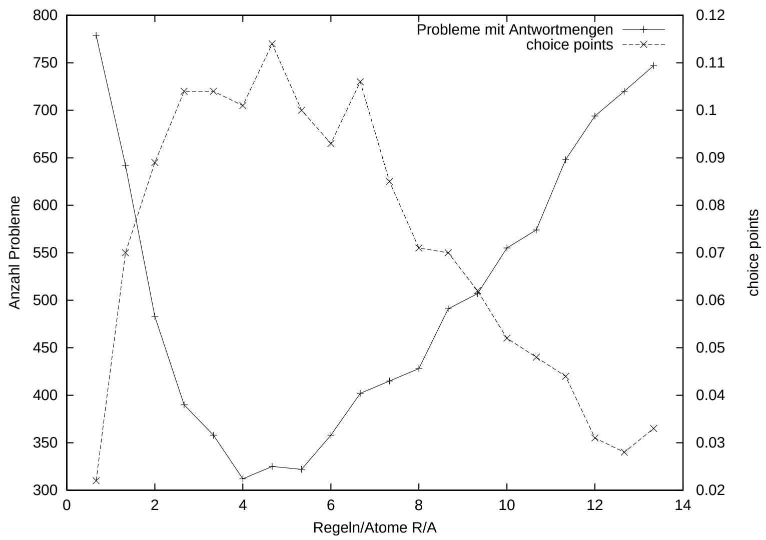

Abbildung 6.30: Anteil der erfüllbare Probleme und durchschnittliche choice points bei durchschnittlich 3 Literalen und $A = 150$

liegt.

Zusammen mit Kurve 6.29 kann daher gut geschätzt werden, wie sich das Maximum der Anzahl des Berechnungsaufwands bei unterschiedlichen durchschnittlichen Literalzahlen verhält. Wünschenswert ist den Skalierungsfaktor R/A auch auf Literale L bei caseMix zu erweitern. Leider ist die Formel des Skalierungsfaktors dann nicht mehr so einfach und kann mit den mir zur Verfügung stehenden Mitteln nur schwer ermittelt werden. Ihre Form müsste wohl $R/(A * f(L))$ entsprechen, wobei $f(L)$ etwas stärker wächst als $L * \log(L)$.

Abbildung 6.31 zeigt das Wachstum der durchschnittlichen choice points bei 150 Atomen und unterschiedlichen Anzahlwerten von Regeln. Die x-Achse steht hierbei für die durchschnittlichen Literalzahl. Deutlich zu sehen ist ein starkes Wachstum oberhalb von durchschnittlich einem choice point. Ob das Wachstum exponentiell ist, ist schwer zu sagen, da die Kurven wohl alle mit der Zeit in den leicht Bereich abdriften. Um dies festzustellen wäre eine Skalierungsfunktion hilfreich. Allerdings kann aufgrund des starken Wachstums ein exponentielles Wachstum mit Zunahme der Literale angenommen werden.

Das Wachstum des Berechnungsaufwands ist mit Zunahme der durchschnittlichen Literale anscheinend wesentlich stärker, als bei der Zunahme der Regeln und Atome bei konstanten R/A. Wenn also schwere Probleme bei gemischter Körperlänge generiert werden sollen, ist eine Steigerung der durchschnittlichen Literale wohl der beste Weg dahin.

Da Programme mit gemischter Körperlänge immer alle möglichen Körperlängen beinhalten können, ist anzunehmen, dass die gefundenen Muster überall bei gemischter Körperlänge, nur in unterschiedlicher Ausprägung, zu finden sind. Dies

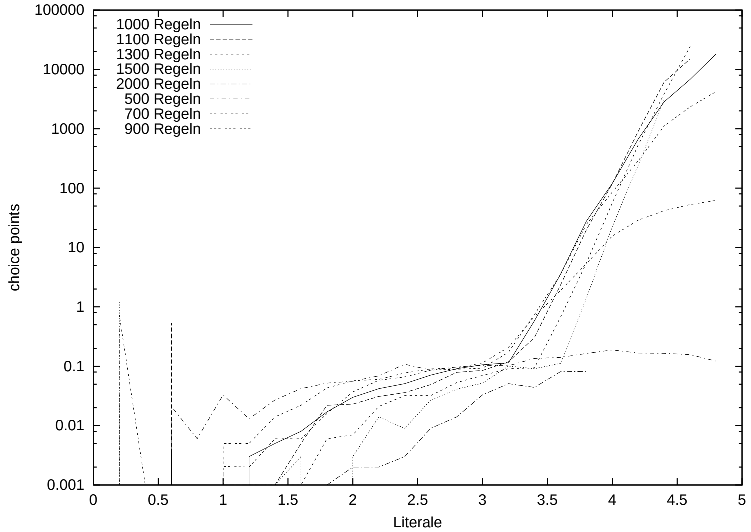

Abbildung 6.31: Durchschnittliche choice points bei unterschiedlichen vielen Regeln und $A = 150$

heißt auch, dass das Wachstum wohl überall exponentiell ist. Bei geringer Anzahl von Atomen, Regeln oder durchschnittlicher Anzahl von Literalen ist es nur so gering, dass es nicht auffällt.

Programme mit gemischter Körperlänge zeigen auch eine Art Mischung der Eigenschaften von Programmen mit fester Körperlänge mit den Körperlängen, die sie enthalten. Es ist z.B. anzunehmen das Programme die nur Körperlänge 0 und 1 enthalten, polynomiell lösbar sind.

6.2.2 Testreihe casesCon und casesConR mit smodels

Die Testreihe casesCon wurde mithilfe des ersten Modells (jedes Atom kommt einmal im Körper vor) und casesConR mithilfe des zweiten Modells (jedes Atom kommt einmal im Kopf vor) des Abschnitts 4.1.3 auf Seite 24 erhoben. Der verwendete Solver ist smodels Version 2.28. Gesucht wurde jeweils nur immer eine Antwortmenge. Gerechnet wurde auf den Universitätsrechnern.

Diese Testreihen sind von der Generierung der Programme der Testreihe casesMix sehr ähnlich. Der Hauptunterschied ist, dass alle Atome über Regeln miteinander verbunden sind. Alle Programme, die mithilfe der in casesCon und casesConR benutzten Generierungsmodelle generiert werden können, können auch mit dem Generierungsmodell, das in caseMix benutzt wird, generiert werden.

Leider ist mir bei der Erstellung der Generatoren ein kleiner Programmierfehler unterlaufen, so dass ein Atom mehr als Kopf möglich waren. Dies sollte sich aber, bei der großen Anzahl von Atomen mit denen Programme generiert wurden, nicht wesentlich auswirken.

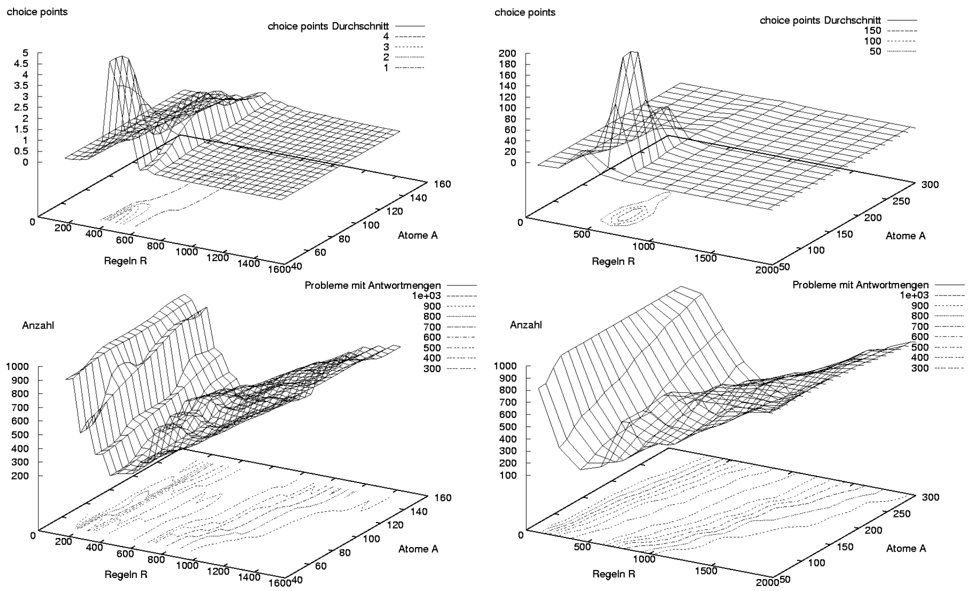

Abbildung 6.32: Durchschnittliche choice points und Erfüllbarkeit bei 10 Literalen für casesCon und casesConR

Auf Abbildung 6.32 zeigen die oberen Diagramme die durchschnittliche Anzahl von choice points und die unteren die Anzahl der erfüllbaren Probleme. Die Diagramme auf der linken Seite gehören zu casesCon und die auf der rechten Seite zu casesConR. Alle Diagramme beziehen sich auf Probleme mit durchschnittlich 10 Literalen pro Körper. Das Erste, was auffällt ist, dass sowohl casesCon als auch casesConR wesentlich einfacher sind als caseMix. Während bei caseMix schon ab durchschnittlich 5 Literalen pro Körper bei 150 Atomen eine Testreihe für verschiedene Regelanzahlen sehr schwer auszuführen ist, ist es bei casesCon und casesConR selbst bei durchschnittlich 10 Literalen kein großes Problem, trotz des starken Wachstums mit den durchschnittlichen Literealen. An sich hatten die gemachten Testreihen casesCon und casesConR wohl zu kleine Werte als Obergrenze für die Atom-, Regelzahl und die durchschnittlichen Literale. Gute Grenzen sind allerdings nur schwer auszuloten, da durch das exponentielle Wachstum die Grenze zwischen zu leichten und zu schweren Problemen schmal ist.

Weiterhin fällt auf, dass die Höhenlinien, bei den unteren Erfüllbarkeitsdiagrammen, zwar noch ungefähr gradenlinig sind, aber die Verlängerung der gradlinigen Abschnitte nicht mehr ungefähr in der Nähe des Koordinatenursprungs verläuft. Die R/A Skalierungsformel hat bei casesCon und casesConR seine Aussagekraft verloren. Die Erfüllbarkeitskurven zeigen eine grabenähnliche Struktur, die sich, im Gegensatz zum Verhalten der Erfüllbarkeit bei caseMix (siehe Abbildung 6.27 Seite 65), nur allmählich verbreitert.

Die oberen Diagramme für die durchschnittlichen choice points fallen anscheinend völlig aus dem Bild, wie es sich bisher ergeben hat. Statt eines Höhenzuges der mit steigender Atom- und Regelzahl exponentiell wächst, sind hier zwei Höhenzüge zu sehen, die mit steigender Anzahl von Regeln und Atomen kleiner werden, wobei bei casesConR noch höhere Werte als bei casesCon erreicht werden, casesConR also anscheinend noch schwieriger ist als casesCon. Der schwere Bereich liegt ungefähr dem Bereich, in dem die Erfüllbarkeitskurve ihr Minimum hat.

Der Hauptunterschied zwischen caseMix und casesCon sowie casesConR ist, dass alle Atome in den Programmen zusammenhängen. Dadurch ist es wahrscheinlicher, dass aus vorhandenen Informationen auf den Wert eines weiteres Atoms geschlossen werden kann. Der Anteil der erfüllbaren Probleme müsste sich also erhöhen. Weiterhin bewirkt ein höherer Anteil von Fakten stärker als bei caseMix, dass die Programme erfüllbar sind, weil eher Regeln vorhanden sind um sie zu nutzen, beziehungsweise in denen die dazugehörigen Atome vorkommen. Dadurch dürfte die Erfüllbarkeitskurve, im Gegensatz zu caseMix, einen flacheren Graben zeigen und das rechte Ufer des Grabens, das Ufer mit mehr Regel, dürfte in den Bereich von weniger Regeln verschoben sein. Wie in den Diagrammen ersichtlich.

Die choice point Diagramme könnten so erklärt werden, dass der rechte Teil, bei mehr Regeln, der choice point Kurve bei caseMix sozusagen abgeschnitten wird. So dass die coice point Kurven bei casesCon und casesConR nur eine Art linken Teil der casesMix Kurve zeigen. Dieser Schnitt wird dadurch bewirkt, dass sich die Informationen, durch den Zusammenhang der Atome, stärker auswirken und sich die Zahl der choice points, gerade bei mehr Atomen und Regeln, stärker verringert.

Das Verhalten der Berechnungsaufwandskurven erschwert natürlich das Finden von schweren Bereichen.

Die in Abbildung 6.33 gezeigten Diagramme gehören zur Testreihe casesCon. Sie sind vertikal genauso wie in Abbildung 6.32 geordnet, allerdings befinden sich

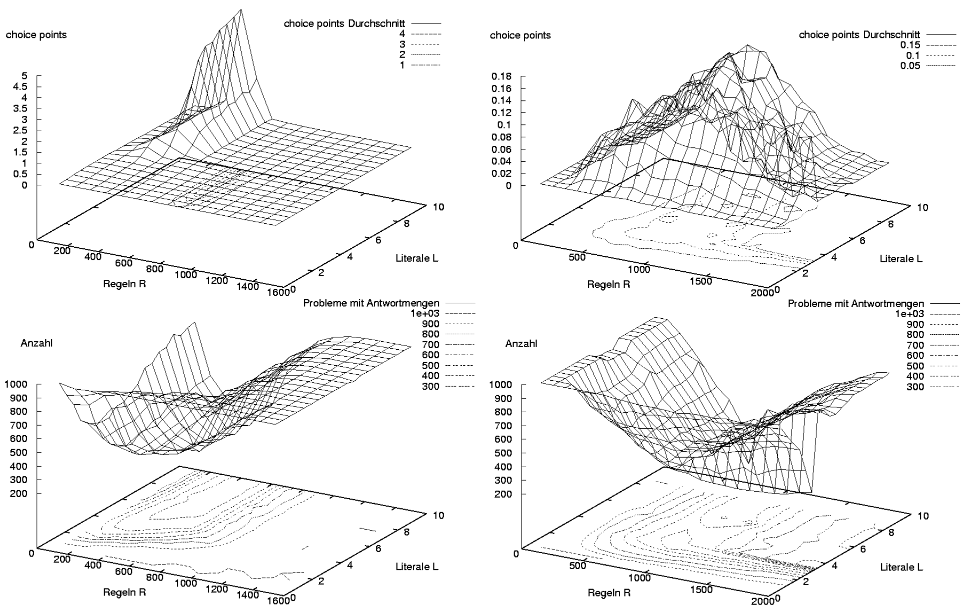

Abbildung 6.33: Durchschnittliche choice points und Erfüllbarkeit bei casesCon für 50 und 300 Atome

jetzt links die Diagramme für 50 Atome und rechts die für 300 Atome.

In den beiden oberen choice point Diagrammen ist wieder der mit der Literalzahl wachsende Höhenzug zu sehen. Wobei er links bei 50 Atomen noch wesentlich höher und ausgeprägter ist, als rechts bei 300 Atomen. Dies deutet auf ein starkes Wachstum der durchschnittlichen choice point Zahl, bezüglich einer konstant gehaltenen Skalierungsformel, hin. Für ein besseres Verständnis der Zusammenhänge, müssen wohl die durchschnittlichen choice points zusammen mit der Atomzahl, Regelzahl und durchschnittlichen Literale gesehen werden, was allerdings schwierig ist. Die Höhenlinien in einer solchen Darstellung sind wahrscheinlich kegelartige Gebilde, die in Richtung der Literaleachse offen sind.

Die unteren Erfüllbarkeitsdiagramme zeigen wieder grabenartige Strukturen, die, von Anfangswerte bei wenigen Literalen abgesehen, fast parallel zur Literalachse verlaufen. Es ist einsichtig, dass bei sehr wenigen Literalen mehr Programme erfüllbar sind, da bei ihnen auch ein sehr hoher Faktenanteil existiert. Die Tiefe der Gräben ist ungefähr gleich. Der Graben für 300 Atome ist weniger ausgeprägt und nach rechts, in den Bereich von mehr Regeln, verschoben. Allerdings ist die Verschiebung nicht mit der Skalierungsformel R/A aufhebbar.

Desweiteren ist beim Diagramm für 300 Atome, beim Graben ein Seitenarm zwischen 2 bis 4 Literalen zu sehen, der sich in Richtung der Regelachse allmählich abflacht. Ein entsprechender Seitenhöhenzug ist darüber im zugehörigen choice point Diagramm für 300 Atome zu sehen. Die Erfüllbarkeit ist also anscheinend mit dem Berechnungsaufwand gekoppelt. Bei beiden Atomzahlen sind die Probleme anscheinend um so schwieriger, um so unwahrscheinlicher sie erfüllbar sind.

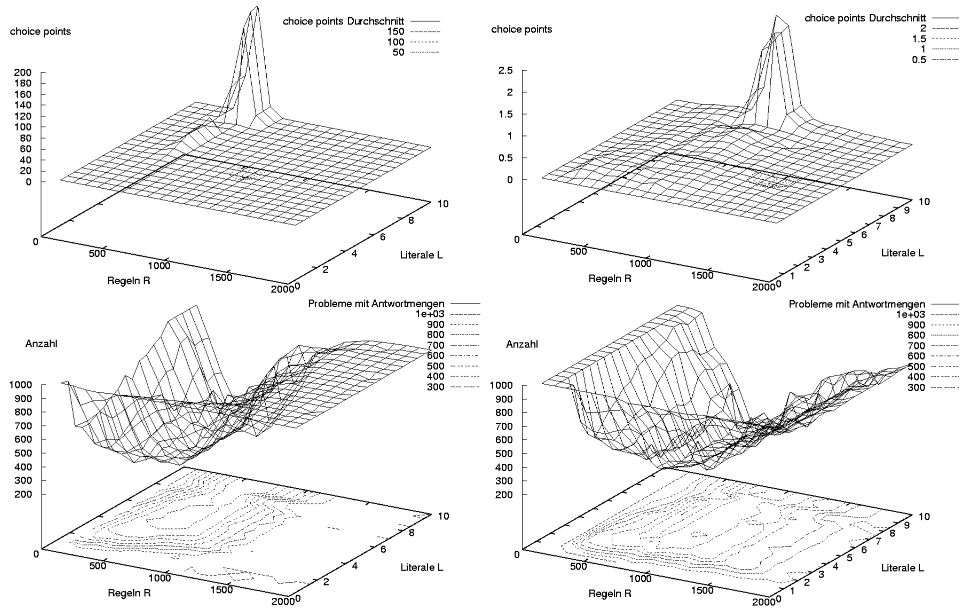

Abbildung 6.34: Durchschnittliche choice points und Erfüllbarkeit bei casesConR für 100 und 300 Atomen

Die Aufteilung in Abbildung 6.34 entspricht vertikal der aus Abbildung 6.33, nur befinden sich jetzt links die Diagramme für 100 Atome und rechts die für 300 Atome. In beiden Abbildungen wurde die linke Seite jeweils so gewählt, dass es beim Berechnungsaufwand einen deutlichen Ausschlag gibt. Bei casesConR sind die Maximalausschläge des Berechnungsaufwands höher als bei casesCon. Dies könnte daran liegen, dass bei casesCon, im Gegensatz zu casesConR, Atome auch in keinem Kopf vorkommen können, wodurch sie von vornherein mit Falsch belegt werden können. Dadurch werden schon am Anfang zusätzliche Informationen bereitgestellt und die Suche wird einfacher.

Ansonsten ähnelt das Verhalten von casesConR sehr dem von casesCon.

Wahrscheinlich gibt es auch bei den Programmen zu den Generierungsmodellen zu casesCon und casesConR ein exponentielles Wachstum, allerdings sind die schweren Bereiche für eine feste durchschnittliche Literalzahl lokal begrenzt.

Durch das gezeigte Berechnungsaufwandsverhalten sind wahrscheinlich die hier verwendeten Generierungsmodelle, aus Abschnitt 4.1.3 auf Seite 24 für zusammenhängende Atome, eher schlechter für das Abschätzen des Berechnungsaufwands bei Anwendungsproblemen geeignet.

6.2.3 Testreihe graph1 mit smodels

Bei der Testreihe graph1, wurden die Programme gemäß Abschnitt 4.2.1 auf Seite 26 generiert. Hier wurde also nach Hamiltonschen Zyklen in Graphen gesucht. Gerechnet wurde auf Universitätsrechnern.

Bei den nachfolgenden 3-dimensionalen Diagrammen, die eine Kurvenfläche enthalten, gibt es am oberen linken Rand, in dem Bereich mit wenigen Knoten aber mehr Kanten, auch eine Fläche, obwohl dort keine Daten vorhanden sind. Bei V Knoten ist die maximale Anzahl der Kanten gleich $E = V^2$, also alle möglichen Paare von Knoten. Diese Unstimmigkeit wird von mir in Kauf genommen, da Kurvenflächen anschaulicher sind und gnuplot sie dann so generiert.

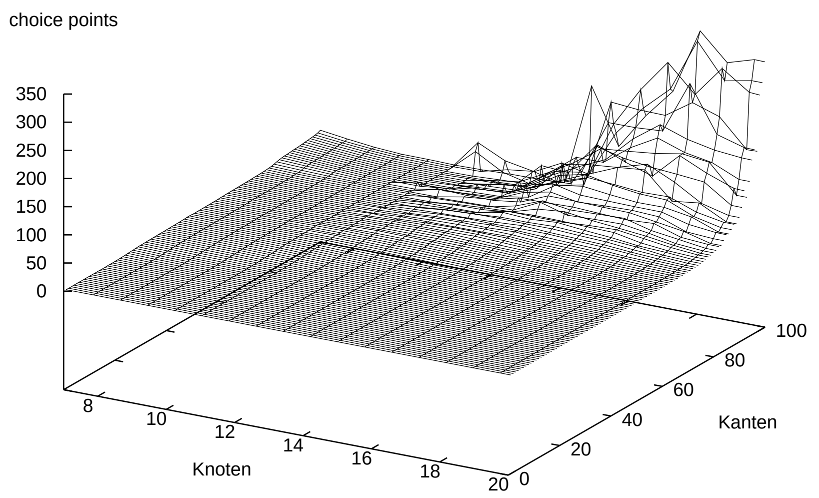

Abbildung 6.35: Durchschnittliche choice points bei graph1

Der Höhenzug aus Abbildung 6.35 für die durchschnittlichen choice points, weist Ähnlichkeiten mit den bisher behandelten Höhenzügen für die choice points von cases und caseMix auf, nur ist diesmal die x-Achse die Anzahl der Knoten und die y-Achse die Anzahl der Kanten. Auch zeigt der Höhenzug mehr Unebenheiten auf, als die vorangegangenen.

Die Abbildung 6.36 zeigt für die unterschiedlichen Testpunkte, die Anzahl der Probleme die erfüllbar waren, beziehungsweise die Anzahl der Graphen die mindestens einen Hamiltonschen Zyklus besaßen. Wie zu erwarten, steigt die Wahrscheinlichkeit, dass Graphen einen Hamiltonschen Zyklus haben, ab einen bestimmten Kanten/Knoten Verhältnis plötzlich stark an, es gibt also einen Phasen-

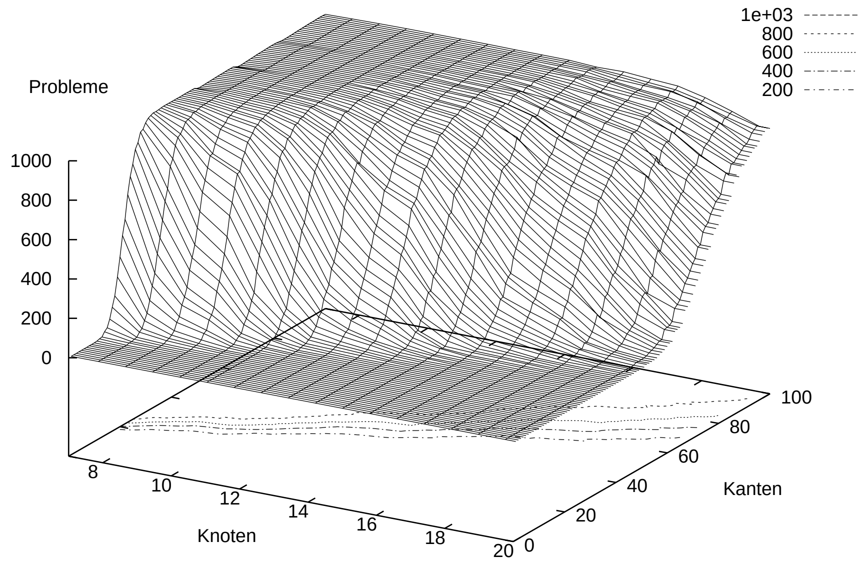

Abbildung 6.36: Erfüllbare Probleme bei graph1

übergang. Allerdings sind die Höhenlinien nahezu gradlinig, was auf eine Skalierungsformel ungleich $\frac{E}{N \log N}$ schließen lässt, siehe Abschnitt 3.1 Seite 15. Die erhobenen Daten lassen vermuten, dass die Skalierungsformel irgendwo zwischen $\frac{E}{N \log N}$ und $\frac{E}{N}$ liegt. Für die Formeln bewegt sich die 50 Prozent Marke der Erfüllbarkeit in unterschiedliche Richtungen, wenn die Knotenzahl erhöht wird. Ein Grund dafür könnte die andere Art von Graphen sein, die hier verwendet wird.

Abbildung 6.37 zeigt verschiedene Parameter der Programme. All diese Parameter zeigten innerhalb der Testpunkte nur wenig Varianzen.

Links oben ist die durchschnittliche Anzahl der Atome der Programme aufgeführt. Zu sehen ist, dass sie anscheinend linear mit der Anzahl der Kanten und Knoten wächst. Ein ähnliches Bild ergibt sich für die Anzahl der Regeln im Diagramm links unten, nur ist deren Wachstum etwas höher. Daraus resultiert die Kurve rechts unten für der R/A Faktoren. Ihr Verhalten ist nicht mehr linear.

Das Verhalten der durchschnittlichen Anzahl der Literale, ist auch nicht linear. Anscheinend wächst sie mit der Anzahl der Kanten, aber sinkt schwach mit der Anzahl der Knoten. Die linearen Höhenlinien, die anscheinend in Richtung Koordinatenursprung verlaufen, lassen auf eine Skallierungsformel $E/N = Kanten/Knoten$ schließen.

Aufgrund der ständigen Veränderung der durchschnittlichen Literale, lässt sich die Testreihe graph1 nur schwer mit den zuvor beschriebenen Testreihen, z.B. ca-

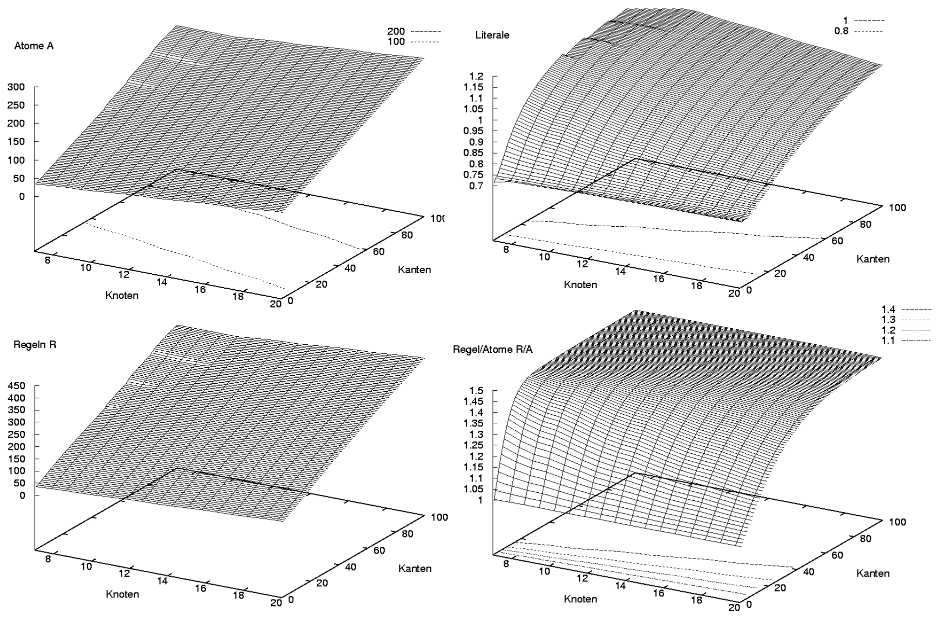

Abbildung 6.37: Durchschnittliche Atome, Literale, Regeln und Regeln durch Atome R/A bei graph1

seMix, vergleichen.

Wenn die Werte für durchschnittliche Literale und R/A von graph1 auf caseMix bezogen werden, befinden sie sich im leichten Bereich, denn beide Parameter nehmen nur geringe Werte an.

In Abbildung 6.38 sind einige choice point Kurven und die Kurve der erfüllbaren Probleme für Graphen mit 20 Knoten gegenüber gestellt.

Das Diagramm oben links zeigt die durchschnittlichen choice points und die Anzahl der erfüllbaren Probleme. Die Achse für die Erfüllbarkeit ist links, die Achse für die choice points rechts, diese ist auch logarithmisch unterteilt. Während der Phasenübergang bei der Erfüllbarkeit ungefähr zwischen 50 und 100 Kanten liegt, liegt das Maximum der choice points ungefähr bei 180 Kanten. Das Wachstum der choice point Kurve beginnt anscheinend mit dem Phasenübergang der Erfüllbarkeitskurve. Nachdem die choice point Kurve das Maximum durchlaufen hat, sinkt sie relativ schnell von einem Wert von rund 10^6, auf einen Wert knapp über 10 durchschnittliche choice points.

Das Diagramm links oben zeigt den Median der choice points. Zu beachten ist, dass dies das Einzige der 4 Diagramme aus 6.38 ist, bei dem die choice point Skala nicht logarithmisch unterteilt ist. Nach einem relativ starken Wachstum der Kurve, in der Nähe des Phasenübergang der Erfüllbarkeit, bei rund 60 bis 150 Kanten, geht die Kurve in einen langsameren Anstieg über, der annähernd linear aussieht. Dabei

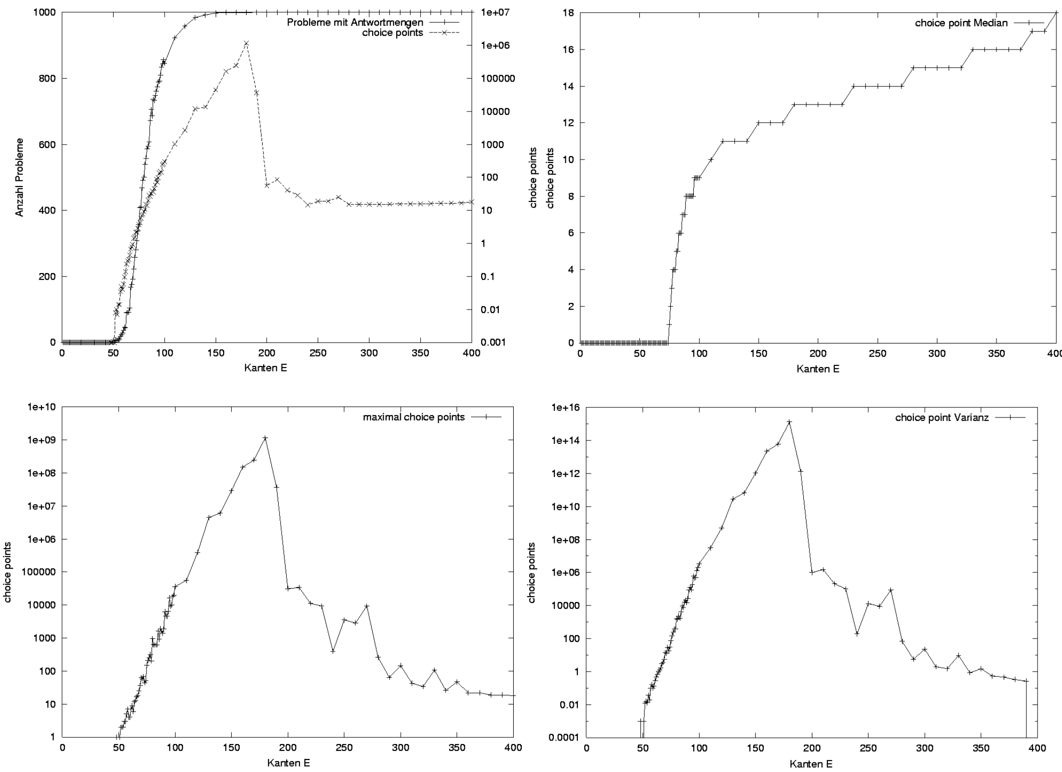

Abbildung 6.38: Durchschnittliche, median, maximal choice points und coice point Varianz bei graph1

erreicht sie am Ende gerade mal 18 choice points. Ab ungefähr 250 Kanten dürften die Werte für die durchschnittlichen und die median choice points ungefähr gleich sein. Davor fehlt der median choice point Kurve aber völlig eine schwer Phase, beziehungsweise ein lokales Maximum.

Wo das Maximum der Kurve für durchschnittliche choice points herkommt, wird bei der Betrachtung der Kurve links unten für die maximalen choice points ersichtlich. Sie zeigt annähernd den Kurvenverlauf der durchschnittlichen choice points. Das Maximum der Kurve für die maximalen choice points liegt ungefähr bei 10^9, also rund um den Faktor 1000 höher, als das Maximum der durchschnittlichen choice points. Da nur 1000 Probleme für einen Testpunkt generiert wurden, ergeben sich die durchschnittlichen choice points anscheinend im schwer Bereich fast ausschließlich aus einigen wenigen überschweren Problemen.

Diese Tatsache wird auch von der, im Diagramm rechts unten dargestellten, choice point Varianz bestätigt, die innerhalb des schwer Bereichs sehr hoch ist.

Ob der median der choice points, bei Problemen aus der Testreihe graph1, auch ein anderes Verhalten zeigen kann, dürfte schwer zu ermitteln sein, da bei mehr

Kanten der schwer Bereich so gut wie nicht mehr zugänglich ist.

Das gezeigte Verhalten der choice points lässt vermuten, dass die choice point Verteilung, innerhalb eines Testpunktes im schwer Bereich, sehr asymmetrisch ist. Es gibt wahrscheinlich viele Probleme mit relativ wenigen choice points, so dass dort ein starkes Maximum oder eine starke Ansammlung auftritt. Dann gibt es noch einige wenige Probleme mit sehr vielen choice points, die den Durchschnitt der choice points dominieren.

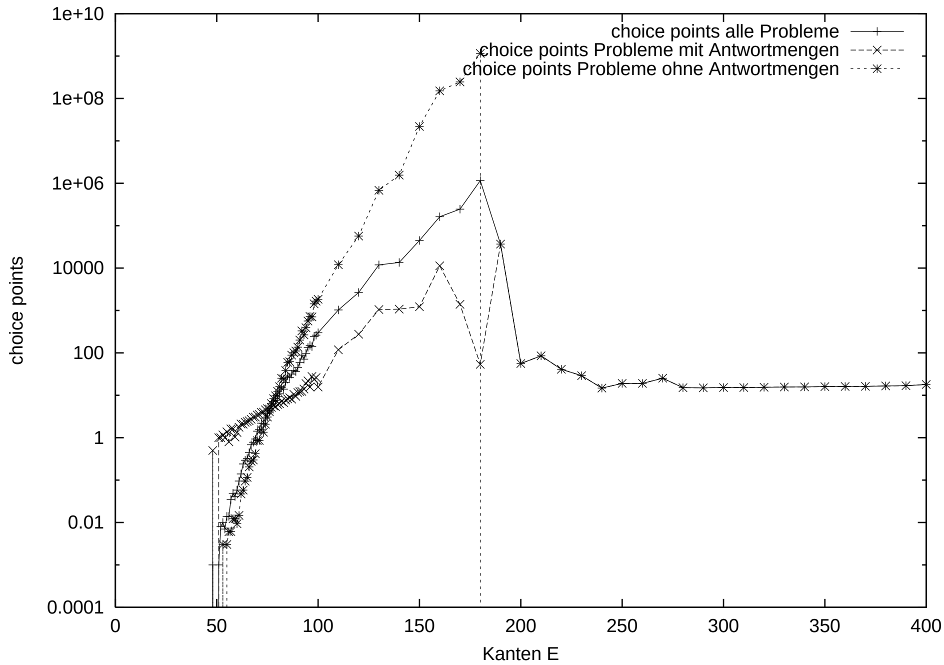

Abbildung 6.39: Durchschnittliche choice points bei erfüllbaren und unerfüllbaren Graphen

Woher die überschweren Probleme kommen, wird aus Abbildung 6.39 ersichtlich. In ihr sind die durchschnittlichen choice points für die erfüllbaren, unerfüllbaren und alle Probleme eingetragen. Die Kurve für die erfüllbaren Probleme fängt erst bei rund 50 Kanten an und die der unerfüllbaren hört bei rund 180 Kanten auf, da außerhalb dieses Bereiches keine erfüllbaren oder unerfüllbaren Probleme vorhanden waren. Deutlich zu sehen ist, dass im schwer Bereich die Kurve der erfüllbaren Probleme deutlich unter der von den unerfüllbaren liegt, die durchschnittlichen choice points werden also von den wenigen unerfüllbaren Problemen dominiert. Mit steigender Kantenzahl wird es demnach immer schwerer zu beweisen, dass ein Graph keinen Hamiltonschen Zyklus besitzt. Im schwer Bereich hat auch die Kurve der erfüllbaren Probleme einen Ausschlag nach oben, nur ist dieser deutlich schwächer, als der, der unerfüllbaren Probleme. Die Kurve der unerfüllbaren Probleme hat anscheinend einen exponentiellen Anstieg, der nur abbricht weil keine unerfüllbaren Probleme mehr vorhanden waren. Interessant wäre noch herauszufinden, wie sich diese Kurve fortsetzt, indem für die Testpunkte mehr Probleme generiert werden. Auch das Maximum der durchschnittlichen choice points aller Probleme könnte dadurch weiter nach rechts, in den Bereich von mehr Kanten, verschoben werden.

Insgesammt zeigt das Verhalten der Testreihe graph1, doch deutliche Unterschiede zu dem Verhalten von den anderen hier untersuchten Testreihen für zufällig generierte Programme. Es gibt aber auch viele Parallelen.

6.3 Vergleich von smodels und nomore++

Beim Vergleich von unterschiedlichen Solvern ist folgendes zu beachten.

Die Höhe der Kurvenwerte ist aussagekräftig über den tatsächlichen choices oder Zeitaufwand der Solver, sie kann aber keine Aussagen über die Güte der zugrunde liegenden Ideen treffen. Da die Ideen beim Solver eventuell in einer schnelleren Implementation umgesetzt werden können. Der nomore++ Solver ist ein relativ neuen Ansatz, bei der verwendeten Version handelt es sich wohl um eine der Ersten überhaupt lauffähigen. Wohingegen smodels schon wesentlich älter ist und wohl so einige Optimierungen hinter sich hat. Es ist sehr wahrscheinlich, dass nomore++ allein durch ein paar Implementationsänderungen verbessert werden kann, ohne dabei die grundlegende Arbeitsweise zu ändern.

Deshalb können wohl allein mit den choices oder den Zeiten, die ein Solver für ein Beispiel benötigt, keine wirklichen Aussagen darüber getroffen werden, wie gut der verwendete Ansatz ist.

Der Anstieg einer geigneten Kurve (z.B. bei konstanten R/A Faktor) ist eher ein Indiz für die Güte der zugrunde liegenden Ideen, da er eher das Wachstum des Berechnungsaufwands darstellt. Mit ihm lassen sich auch Abschätzungen für Beispiele mit anderen Parametern machen.

6.3.1 Vergleich von cases und casesN

Die Testreihe casesN entspricht der Testreihe cases mit dem Unterschied, dass als ASP-Solver nomore++ Version 0.4.10 verwendet wurde. Gerechnet wurde auf meinen Heimrechner. Wegen der großen Ähnlichkeit des Verhaltens wird casesN hier im Vergleich zu cases betrachtet.

Da sowohl smodels als auch nomore++ deterministisch arbeiten, sind natürlich die Kurven für die Anzahl der erfüllbaren und damit auch unerfüllbaren Probleme, von statistischen Ungenauigkeiten abgesehen, gleich.

Im Allgemeinen ähnelt sich das Verhalten der Kurven für cases und casesN sehr, weshalb hier vor allem die Unterschiede betrachtet werden.

Abbildung 6.40 zeigt die durchschnittlichen choice points der erfüllbaren, unerfüllbaren und aller Probleme für die Testreihe cases mit smodels und casesN mit

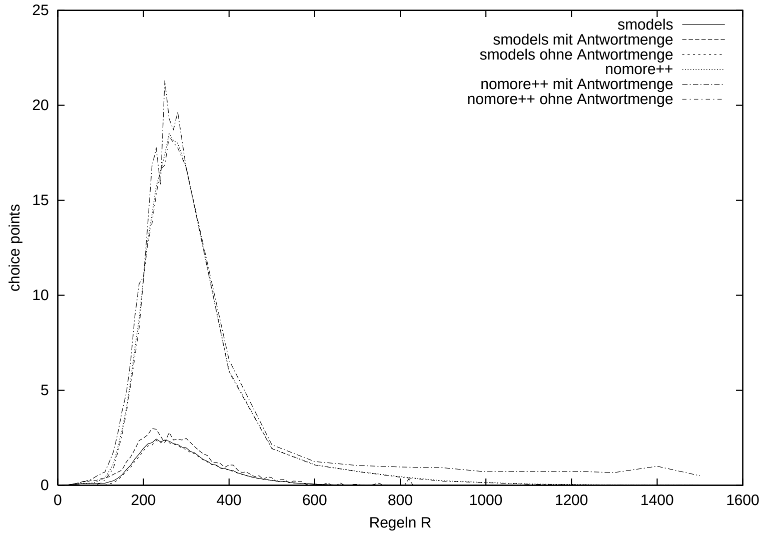

Abbildung 6.40: Durchschnittliche choice points bei erfüllbaren, unerfüllbaren und allen Problemen für cases und casesN bei 50 Atomen

nomore++. Der Verlauf der Kurven ist anscheinend unabhängig vom verwendeten Solver, nur die Höhe der konkreten Werte unterscheidet sich. So ist die Anzahl der durchschnittlichen choice points für nomore++ durchweg höher als für smodels.

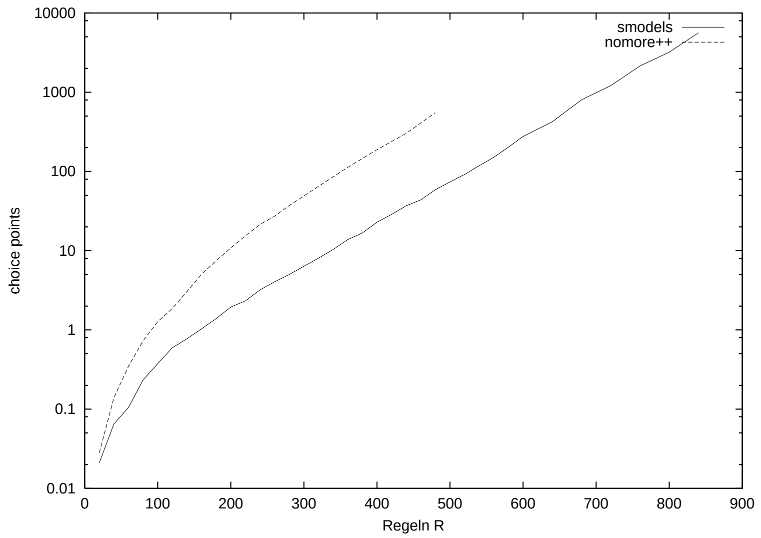

Abbildung 6.41: Durchschnittliche choice points für cases und casesN bei $R/A = 4$

Abbildung 6.41 zeigt das Wachstum bei Faktor $R/A = 4$. Ersichtlich ist, dass es auch bei nomore++ anscheinend eine anfängliche Phase mit sehr starken Wachstum gibt und das Wachstum danach in einen annähernd stabile exponentielle Phase übergeht. Auch hier sind also die Kurven für smodels und nomore++ vom Verlauf her sehr ähnlich. Die Werte der nomore++ Kurve sind aber überall wesentlich höher, als die Werte der smodels Kurve.

Leider sind die Werte von nomore++ schon sehr früh so hoch, dass eine weitere Berechnung des Kurvenverlaufs sehr aufwändig wurde und abgebrochen werden musste. Deshalb sind nur Vermutungen über die stabile Phase möglich.

Es sieht aber so aus, als ob die Kurve von nomore++ bei höheren Werten, in der stabilen Phase, annähernd parallel zu der von smodels verläuft. Zu beachten ist, dass in einer exponentiellen Darstellung ein konstanter Abstand bedeutet, dass der Quotient der Werte ein konstanter Faktor ist. Dieser Faktor scheint hier rund 10 zu sein, dass heißt nomore++ braucht durchschnittlich für Probleme ungefähr 10 mal mehr choice points als smodels.

Weitergehende Untersuchungen wären auf jeden Fall interessant. Beispielsweise für die Frage, ob der Berechnugsaufwand bei nomore++ stärker als exponentiell wächst, wie anscheinend bei smodels.

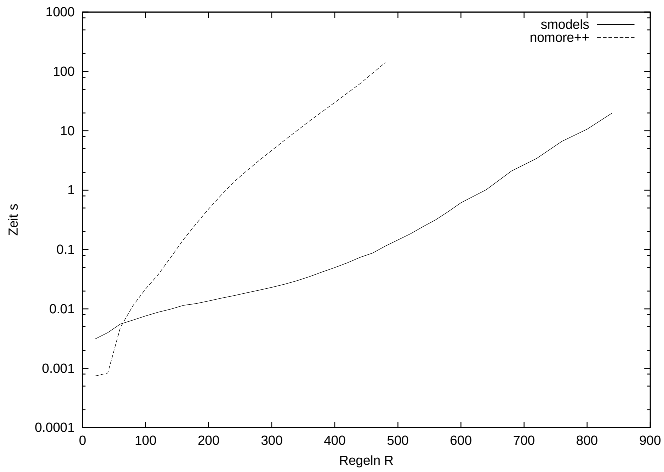

Abbildung 6.42: Durchschnittliche Zeit für cases und casesN bei $R/A = 4$

Abbildung 6.42 zeigt die durchschnittliche Zeit bei cases und casesN. Hier schneidet nomore++ noch schlechter als bei den choice points ab. Nur bei sehr kleinen Werten liegt nomore++ kurzzeitig vorn. Die Endwerte der nomore++ Kurve liegen bei rund 140 Sekunden, was bei 1000 Problemen pro Testpunkt rund 39 Stunden Laufzeit für den Testpunkt bedeutet.

Allerdings beschreibt die Kurve von smodels einen nach oben offenen und die von nomore++ einen nach unter offenen Bogen. Beide Kurven enden ungefähr parallel, also mit ähnlichem exponentiellen Wachstum (bei der Approximation 2^{x*a+b} wäre der Faktor a für beide ähnlich). Es ist also durchaus möglich, dass sich nomore++ beim Wachstum gegenüber smodels noch verbessert. Wenn das Wachstum bei nomore++ irgendwann geringer als bei smodels wird, werden die Werte bei nomore++ auch ihrgendwann kleiner, als die von smodels sein. Da es sich bei den Problemen um noch relativ kleine Probleme, im Sekunden- bis Minutenbereich Berechnungszeit handelt, ist es durchaus möglich, dass nomore++ bei Problemen im Stundenbereich gegenüber smodels nicht mehr gar so schlecht abschneidet.

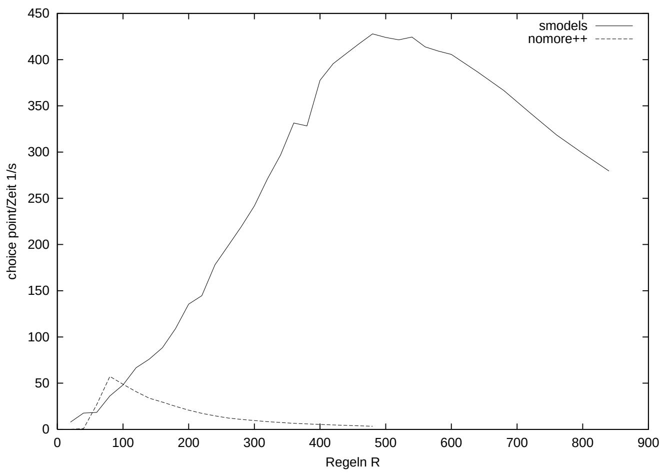

Abbildung 6.43: Durchschnittliche choice points Zeit Quotient für cases und casesN bei $R/A = 4$

Das unterschiedliche Verhältnis von Zeit zu choice points wird auch aus Abbildung 6.43 ersichtlich. Sie zeigt den choice point Zeit Quotient für cases und casesN. Die nomore++ Kurve scheint eine kleinere Version der smodels Kurve zu sein. Dies erklärt, dass sich die durchschnittliche Zeit bei nomore++ noch schlechter verhält, als die durchschnittlichen choice points, da bei nomore++ auf einen choice point durchschnittlich viel mehr Zeit kommt.

Die kleinere Kurve bei nomore++, beziehungsweise das frühe Maximum, könnte bedeuten, dass bei nomore++ die Heuristiken viel eher versagen als bei smodels.

Es ist also sinnvoll, die Zeit und die choice points für einen Solver zueinander in Beziehung zu bringen. Denn die choice points allein helfen bei der Frage, wie lange ein Problem zur Berechnung braucht, nicht weiter. Außerdem können so weitere Zusammenhänge bemerkt und für die Verbesserung der Solver genutzt werden.

Die vorgehenden Zusammenhänge zeigen, dass es durchaus Unterschiede zwischen den Solvern beim Wachstum des Berechnungsaufwands gibt, die aus dem leicht-schwer-leicht Muster nicht ersichtlich werden.

Kapitel 7

Abschluss

7.1 Zusammenfassung

Bei allen untersuchten Testreihen gab es immer wiederkehrende Muster, in fast allen Bereichen. Allerdings treten diese Muster erst bei größeren Werten, beziehungsweise Problemen, auf, bei kleineren Werten geht es oft in statistischen Rauschen unter.

Eines diese Muster war, dass es Skalierungsformeln gibt, z.B. R/A, bezüglich derer das Verhalten der Kurven ähnlich ist. Diese Skalierungsformeln können für die Bestimmung des Berechnungswachstums genutzt werden. Wenn der Wert der Skalierungsformel (der Faktor) konstant gehalten wird, zeigte sich meist ein monotones Wachstum. Dieses Wachstum ist wahrscheinlich bei allen untersuchten Bereichen, außer 0-LP und 1-LP, exponentiell. Das exponentielle Wachstum erschwert aber eine genaue Untersuchung, da es dadurch sehr schnell zu aufwändig wird weitere Testpunkte zu generieren und es nur schmale aussagekräftige Bereiche gibt, die nicht immer leicht zu finden sind.

Bei der Untersuchung von Solvern, bei verschiedenen konstanten Faktoren für die Skalierungsformeln R/A und fester oder gemischter Körperlänge, zeigte sich, bei konstanter Literalzahl, ein ähnliches Wachstumsverhalten. Bei der Untersuchung von Solvern bietet sich daher diese Skalierungsformel bei Faktoren an, bei denen das Wachstum schneller steigt, da dann auch schwere Programme einfach zu generieren sind.

Anzahl choice points, wrong choices, Wahrheitswertbelegung und Zeit zeigen alle ähnliches Verhalten, auch wenn sich das konkrete Wachstum unterscheidet. Dieser Unterschied verändert sich im allgemeinen geringer als die konkreten Werte. Danach ist es nicht mehr so wichtig, welcher Parameter als Indikator für das Berechnungsaufwandswachstum genommen wird. Das Verhalten des Unterschieds kann allerdings dennoch für die Untersuchung des Solvers nützlich sein, z.B. beim Unterschied von choice point Wachstum zum Zeitwachstum.

Wenn es um den Vergleich des konkreten Berechnungsaufwands geht, ist es unerlässlich auch die Zeit zu untersuchen. Dies kann auch geschehen, indem sie zu anderen Parametern, z.B. choice points, in Relation gesetzt wird. Wie der gemachte Vergleich zwischen nomore++ und smodels zeigt, können sich beispielsweise die durchschnittlichen choice points um einen deutlichen anderen Faktor unterscheiden als die durchschnittlichen Zeiten. Das choice point zu Zeit Verhältnis hängt sogar nicht nur von den Solvern ab, sondern variiert auch bei Programmen mit unterschiedlichen Parametern für einen Solver.

Kleinere Unterschiede bei den Modellen zur Generierung der Programme beeinflussen nicht sehr das Ergebnis. Bei Untersuchungen ist es demnach durchaus zulässig, Modelle zur Generierung zu wählen, mit denen sich leichter Programme generieren lassen können.

Die großen Ähnlichkeiten bei verschiedenen Generierungsmodellen, Berechnungsaufwandsparametern und Solvern, eröffnet eine Möglichkeit unterschiedliche Solver auf allgemeiner Basis zu vergleichen. Dafür könnte beispielsweise beim 2-LP Generierungsmodell die choice point Kurven und Zeitkurven der Solver für den konstanten Faktor $R/A = 4$ verglichen werden.

Einfache theoretische Modelle, wie die in Abschnitt 5 vorgestellten, können zur Erklärung des Verhaltens hilfreich sein.

Die Informationen, die zur Lösung eines Problems zur Verfügung stehen, spielen eine zentrale Rolle. Einerseits können mehr Informationen den Suchvorgang vereinfachen, andererseits wird um mehr Informationen zu berücksichtigen und zu finden mehr Zeit benötigt. Daher ist es wahrscheinlich nicht möglich einen Antwortmengensolver zu bauen, der in allen Problemklassen der Beste ist, da für verschiedene Problemklassen verschiedene Arten von Informationen unterschiedlich nützlich sind. Es ist daher vorteilhaft, sich bei der Wahl des Solvers an der Problemklasse zu orientieren und bei speziellen Problemen auch spezielle Suchallgorithmen zu verwenden.

Es kann wahrscheinlich niemals mit Sicherheit gesagt werden, dass sich Anwendungsprobleme anders verhalten als die zufälligen generierten Probleme. Es können nur mithilfe von Einzellanwendungsfällen Vermutungen angestellt werden. Beide Annahmen, dass sie sich ähnlich und das sie sich nicht ähnlich verhalten, sind demnach gleich gut. Allerdings sind Solver nicht bei allen Anwendungsproblemen testbar. Auch ist die Untersuchung von Anwendungstestreihen wohl schwieriger, als bei den hier benutzten Modellen zufällig generierter Programme. Dadurch bieten sich zufällig generierte Programme eher für allgemeinere Untersuchungen an.

Wenn zufällig generierte Programme sich ähnlich wie Anwendungsprogramme verhalten, dürfte es auch ein paar allgemeine Parameter bei der Entwicklung von Programmen geben, mit denen ein geringeres Wachstum des Berechnungsaufwands begünstigt werden kann. Die hier gemachten Untersuchungen lassen vermuten, dass es besser sein sollte Programme so zu schreiben, dass die Körper möglichst kurz sind, möglichst wenig Regeln auf ein Atom kommen oder sehr viele, die Atom- und Regelzahl im allgemeinen möglichst klein ist und es viele Abhängigkeiten zwischen den Atomen gibt.

Ein Problem bei der Erzeugung von Testreihen ist, dass sie mit wenigen generierten Daten oder bei leichten Probleme (leicht Bereich), leicht zu erheben sind, aber dafür auch ungenaue Ergebnisse (Rauschen stärker, weniger Daten) erzeugt werden. Anders herum allerdings, wenn mehr generierte Daten erzeugt wurden oder die Probleme schwerer sind, ist es auch schwerer (zeitaufwändiger) die Daten zu erheben, dafür sind die Ergebnisse aber auch besser.

Dieser Sachverhalt erklärt auch zum Teil die falschen Schlussfolgerungen, die in früheren Untersuchungen über den Berechnungsaufwand bei SAT und ASP gemacht wurden. Aus den untersuchten Bereichen konnte nicht ersehen werden, dass auch im leichten Bereich oder bei gemischter Körperlänge schwere Probleme generiert werden können und das Wachstum auch dort annähernt exponentiell ist, daher wurde fälschlicherweise angenommen, dass dies nicht der Fall ist.

Mit den hier aufgezeigten Methoden sollten aber solche Fehler besser vermieden werden können.

7.2 Ausblick

Die Leistung von Antwortmengen- und auch SAT Solvern könnte vielleicht dadurch gesteigert werden, dass der Berechnungsaufwand der eingesetzten Heuristiken so angepasst wird, dass sie wahrscheinlich rund eine Größenordnung weniger Zeit benötigen als der restliche Algorithmus. Um den Berechnungsaufwand Faktor für die Heuristiken zu ermitteln, bietet sich die Formel 2^{x*a+b} an. Wobei a kleiner als das a in der Approximation sein sollte.

Interessant wäre auch ein Vergleich mehrerer Solver bezüglich eines konstanten R/A Faktors und konstanter Literalanzahl, bei fester oder gemischter Körperlänge. Eine weitere Fortführung der hier gemachten Untersuchungen könnte auch weitere Einblicke verschaffen. Dies dürfte, wegen des exponentiellen Wachstums, aber sehr zeitaufwändig sein.

Von nomore++ sind mittlerweile auch neuere, ausgereiftere Versionen erschienen. Eine Untersuchung des Wachstumsverhaltens der neusten Version würde vielleicht mehr zugunsten von nomore++ ausfallen.

Für die Untersuchung des Wachstumsverhaltens wäre es sehr hilfreich, für die unterschiedlichen Modelle, geeignete Skalierungsformeln zu haben, die z.B. auch die Literale berücksichtigen. Hilfreich dafür wäre eine genauere Bestimmung der Maxima der Kurven für verschiedene Parameterwerte.

Die Formel für den Berechnungsaufwand für ein Generierungsmodelle zu bestimmen, gestalltet sich noch schwieriger als das Finden von Skalierungsformeln. Mit einer solchen Formel, oder einer guten Approximation für sie, könnten allerdings Solver noch besser untersucht, eingeordnet und verbessert werden. Sie würde wahrscheinlich auch das Verständnis verbessern, wodurch Probleme schwer werden.

Mit Untersuchungen von Testreihen für anderen Anwendungsprobleme und der Einordnung dieser in das bisherige Bild, könnten eventuell größere Zusammenhänge aufgedeckt werden.

Die statistischen Untersuchungen weiterer Strukturparameter für Antwortmengenprogrammen, wie beispielsweise die Anzahl der negativen Zyklen und/oder die durchschnittliche Länge dieser, und deren Zusammenhänge zu Berechnungsaufwands- und Erfüllbarkeitsparametern, würde das Verständniss der Zusammenhänge verbessern und außerdem einige bisher aufgestellten Behauptungen überprüfen.

Es wurde auch eine Testreihe unter smodels ohne lookahead mit gemischter Körperlänge mit den Namen casesMixNLA begonnen. Leider wurden innerhalb der Zeit keine aussagekräftigen Bereiche erreicht. Eine Fortführung dieser Testreihe wäre, in Hinblick auf die Untersuchung des Einflusses des lookaheads, sinnvoll.

7.3 Eidesstattliche Erklärung

Ich versichere hiermit, dass die vorliegende Arbeit und die verwendeten Ideen, soweit nicht anders gekennzeichnet, von mir stammen.

Wörtliche oder sinngemäße Übernahmen aus anderen Werken wurden als solche gekennzeichnet.

Literaturverzeichnis

[1] B, C: *Knowledge Representation, Reasoning and Declarative Problem Solving*. CAMBRIDGE University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 2003. ISBN 0 521 81802 8.

[2] B-N, E. und P.L. K: *Kinetic Theory of Random Graphs*. In: *AIP Conference Proceedings 776*, 2005.

[3] C, S A. und D G M: *Finding Hard Instances of the Satisfiability Problem: A Survey*. In: D, G und P (Herausgeber): *Satisfiability Problem: Theory and Applications*, Band 35, Seiten 1–17. American Mathematical Society, 1997.

[4] E, S T. und V L: *Definitions in answer set programming*. Proc. Logic Programming and Nonmonotonic Reasoning 7, Seiten 114–126, 2004.

[5] F, J. und M. P: *Probailistic analysis of the Davis Putnam procedure for solving the satisfiability problem*. Discrete Applied Math., 5:77–87, 1983.

[6] F, J V. und R. P. S: *Average Case Results for Satisfiability Algorithms Under the Random-Clause-Width Model*. Annals of Mathematics and Artificial Intelligence, 20(1-4):357–391, 1997.

[7] F, J. und C. M: *Phase transitions in the properties of random graphs*, 1995.

[8] G, H-O: *Gibbs Measures and Phase Transition*. de Gruyter, Berlin; New York, 1988. ISBN 0-89925-462-4.

[9] I, A: *Wahrscheinlichkeitstheorie und Statistik, Grundlagen-Resultate-Anwendungen*. B.G. Teubner GmbH, Stuttgart/Leipzig/Wiesbaden, 2001. ISBN 3-519-02395-4.

[10] K, H A. und B S: *Planning as Satisfiability*. In: *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, Seiten 359–363, 1992.

[11] L, J: *A Model-Theoretic Counterpart of Loop Formulas*, 2004.

[12] L , F und X Z : *On Odd and Even Cycles in Normal Logic Programs.* AAAI 2004, Seiten 80–85, 2004.

[13] L , F und Y Z : *Answer Set Programming Phase Transition: A Study on Randomly Generated Programs.* ICLP 2003, Seiten 239–253, 2003.

[14] L , F und Y Z : *ASSAT: computing answer sets of a logic program by SAT solvers*. Artificial Intelligence, 157:115–137, 2004.

[15] M G , G T : *Stochastik für Studienanfänger der Informatik*. Carl Hanser Verlag München Wien, 1996. ISBN 3-446-18636-0.

[16] M , D G. und H J. L : *Some Pitfalls for Experimenters with Random SAT*. Artificial Intelligence, 81(1-2):111–125, 1996.

[17] M , D G., B S und H J. L : *Hard and Easy Distributions for SAT Problems*. In: R , P und P S (Herausgeber): *Proceedings of the Tenth National Conference on Artificial Intelligence*, Seiten 459–465, Menlo Park, California, 1992. AAAI Press.

[18] S , H : *Wahrscheinlichkeitsrechnung*. Bibliographisches Institut & F.A. Brockhaus AG, Mannheim, 1992. ISBN 3-411-15841-7.

[19] S , P : *A Model-Theoretic Counterpart of Loop Formulas*, 2004.

[20] G , A , K R und J S. S : *The Well-Founded Semantics for General Logic Programs*. Journal of the ACM, 38(3):620–650, 1991.

[21] Z , Y : *Answer set programming : SAT based solver and phase transition*. Yuting Zhao, Hong Kong, 2003.