Aussagenlogische SAT-Verfahren und ihre Anwendung bei der Lösung des HC-Problems in gerichteten Graphen

Holger H. Hoos

Diplomarbeit am Fachgebiet Intellektik
des Fachbereichs Informatik der TH Darmstadt

12. März 1996

Danksagung

Die vorliegende Arbeit wurde als Diplomarbeit am Fachgebiet Intellektik des Fachbereichs Informatik der Technischen Hochschule Darmstadt unter der Leitung von Prof.Dr. Wolfgang Bibel im Zeitraum von Oktober 1995 bis März 1996 angefertigt.

Mein besonderer Dank gilt Antje Strohmaier, unter deren vorbildlicher Betreuung diese Arbeit entstand. Ohne die zahllosen Gespräche, in denen Ideen und Probleme dargelegt und diskutiert wurden, wären die Arbeit in dieser Form kaum realisierbar gewesen. Weiterhin danke ich Prof. Wolfgang Bibel und Prof. Steffen Hölldobler, die mein Interesse für die Themen der Intellektik früh erkannten und durch ihre engagierte Unterstützung zum Gelingen dieser Arbeit einen wesentlichen Beitrag leisteten.

Mein aufrichtiger Dank gilt außerdem Prof. H.K.-G. Walter und Dr. Ulrike Brandt sowie Prof. Klaus Keimel, die mir als Lehrer und Dozenten im Verlauf meines Studiums wesentlich dabei halfen, solide theoretische Grundlagen und wichtige Methoden erfolgreich zu erarbeiten. In besonderer Weise möchte ich Prof. Peter Zahn danken, der mir nicht nur im Rahmen seiner Lehrveranstaltungen Wege zum tieferen Verständnis der mathematischen Logik wies, sondern auch durch seinen engagierten persönlichen Umgang in mancherlei Hinsicht mein Studium positiv beeinflußte.

Ich danke weiterhin Björn Eberlein, mit dessen Hilfe ich Zugang zu den Rechenkapazitäten der Gesellschaft für Schwerionenforschung in Weiterstadt erhielt, ohne welche der experimentelle Teil der Arbeit in dieser Form nicht möglich gewesen wäre. Des weiteren danke ich Prof. Friedrich Zimmermann und Dr. Thomas Miosga, die mir wesentlich dabei halfen, meinen Interessen auf dem Gebiet der Molekularbiologie im Rahmen meines Nebenfachstudiums nachzugehen; sie haben somit am Anwendungsaspekt dieser Arbeit (der schließlich leider weit weniger ausführlich als geplant behandelt werden konnte) einen wichtigen Anteil.

In besonderem Maße danke ich Sonia Ziesche, die durch mannigfaltige Unterstützung einen erheblichen Anteil am Gelingen dieser Arbeit hat und die in entscheidender Weise dazu beitrug, daß ich meine Interessen für molekularbiologische Grundlagen und Probleme im Rahmen einer ernsthaften Beschäftigung mit dem interdisziplinären Gebiet der Bioinformatik erfolgreich vertiefen konnte.

Darüber hinaus gilt mein aufrichtiger Dank den vielen hier Ungenannten, die auf verschiedenste Weise zum Gelingen meines Studiums und dieser Arbeit im besonderen beitrugen.

Kurzzusammenfassung

Ausgangspunkt der Überlegungen war die Fragestellung, inwieweit spezielle algorithmische Probleme aus dem Bereich der Molekularbiologie in geeignet kodierter Form durch leistungsfähige Modellfindungsverfahren für die Aussagenlogik gelöst werden können. Die Grundlage dieser Vorgehensweise bildet einerseits der GSAT-Algorithmus (Selman & al., 1992), ein leistungsfähiges, unvollständiges Verfahren zur Lösung des $\mathcal{NP}$-vollständigen Modellfindungsproblems in der Aussagenlogik (SAT); andererseits eine neuere Methode zur effizienten Transformation von Hamilton-Kreis-Problemen (HCP) in SAT-Probleme (Iwama & Miyzazaki, 1994).

Hierauf aufbauend, wird in der vorliegenden Arbeit gezeigt, wie das Erkennen von falschen Hybridisierungen (ein $\mathcal{NP}$-hartes Problem aus dem Bereich der physikalischen Genomkartierung) in eine Optimierungsvariante des SAT-Problems (MAX-SAT) transformiert und mithilfe einer erweiterten GSAT-Variante prinzipiell gelöst werden kann. Es stellte sich jedoch heraus, daß diese Vorgehensweise für realistische Problemgrößen kaum praktikabel ist, was im wesentlichen auf das verwendete Transformationsverfahren zurückzuführen ist.

Aufgrund dieses Resultats wurde im Hauptteil der Arbeit die SAT-Kodierung von HC-Problemen und das Verhalten von GSAT-ähnlichen Verfahren auf derart kodierten Problemen ausführlich untersucht. Dabei wurde zunächst die Struktur der SAT-kodierten Probleme charakterisiert; dabei konnte gezeigt werden, welche Struktureigenschaften der entsprechenden CNF-Formeln die Performanz von GSAT negativ beeinflussen. Unter Ausnutzung dieser Resultate wurde eine GSAT-Variante entwickelt und implementiert, welche durch implizite Kodierung der Klauseln für kodierte HC-Probleme einen erheblich reduzierten Platzaufwand aufweist.

Durch problemspezifische Anpassungen konnte die Performanz von GSAT-ähnlichen Verfahren auf HC-Problemen erheblich verbessert werden (eines der dabei verwendeten Konzepte ist leicht parallelisierbar und prinzipiell auf allgemeine SAT-Probleme übertragbar). Auf der Grundlage umfangreicher experimenteller Untersuchungen wurden die Eigenschaften dieser Verfahren in Abhängigkeit verschiedener Parameter charakterisiert. Versuche an unterschiedlich schwierigen Klassen von HC-Problemen zeigten ein adäquates Verhalten der untersuchten Verfahren in dem Sinne, daß einfachere Probleme erheblich leichter gelöst werden konnten. Weiterhin wurde gezeigt, daß die untersuchten (unvollständigen) Verfahren approximativ vollständig in dem Sinne sind, daß lösbare HC-Instanzen mit beliebig hoher Wahrscheinlichkeit gelöst werden können. Darüber hinaus konnte ein allgemeiner, vermutlich problemgrößenunabhängiger Zusammenhang zwischen der Erfolgsquote der Verfahren und einer oberen Schranke für die Laufzeit gezeigt werden.

Auf der Grundlage dieser Resultate ist die Laufzeit der untersuchten Verfahren für gegebene Problemgröße und gewünschte Erfolgsquote nach oben abschätzbar; daraus kann eine Abschätzung der maximalen Größe praktisch lösbarer Probleme gewonnen werden.

Inhaltsverzeichnis

Einleitung

Das Finden von Modellen für aussagenlogische Formeln in konjunktiver Normalform ist in vielen Bereichen der Intellektik von großer Bedeutung. Wie die Entscheidungsvariante ist auch diese konstruktive Version des Erfüllbarkeitsproblems (SAT) für die Aussagenlogik bekanntermaßen $\mathcal{NP}$-vollständig und bis heute nicht allgemein effizient lösbar.

Vor wenigen Jahren wurde mit dem GSAT-Algorithmus ein Verfahren zur Lösung dieses Problems entwickelt, welches in vielen Fällen wesentlich schneller ist als die bis dahin bekannten Algorithmen. Im Gegensatz zu etlichen älteren Verfahren (z.B. David-Putnam) ist GSAT allerdings unvollständig und somit kein Entscheidungsverfahren für SAT. Der Algorithmus kann jedoch so modifiziert werden, daß er zur approximativen Lösung von Optimierungsproblemen verwendet werden kann. Bis heute ist die Weiterentwicklung und Untersuchung GSAT-ähnlicher Verfahren ein Gebiet aktueller Forschung.

Mithilfe derartiger Verfahren ist es möglich geworden, Modelle zu aussagenlogischen Formeln mit mehreren Hundert Variablen zu finden. Daher eröffnet sich für $\mathcal{NP}$-Probleme aus anderen Bereichen der Informatik eine alternative und allgemeine Lösungsmethode: nämlich die Kodierung als aussagenlogisches SAT-Problem und anschließende Lösung mithilfe eines solchen schnellen Modellfindungsverfahrens. Wichtig ist es, hierbei sicherzustellen, daß die Komplexität des Problems durch die Kodierung nicht wesentlich erhöht wird. Aus neueren Untersuchungen sind für einige prominente $\mathcal{NP}$-vollständige Graphenprobleme geeignete SAT-Kodierungen bekannt.

Wie andere Arbeiten gezeigt haben, kann diese allgemeine Vorgehensweise für einige anwendungsrelevante Probleme (Graphfärbung, Steinerbäume etc.) bezüglich Laufzeit (und Qualität approximativer Lösungen) mit den besten derzeit bekannten Spezialalgorithmen konkurrieren. Unklar ist, ob dieser Ansatz prinzipiell auf eine große Klasse von Anwendungsproblemen übertragbar ist oder lediglich in bestimmten Einzelfällen eingesetzt werden kann.

Algorithmische Probleme in der Molekularbiologie

In vielen Bereichen der modernen Molekularbiologie treten algorithmische Probleme auf, deren möglichst effiziente Lösung eine wichtige Voraussetzung für die aktuelle Forschung in diesem Bereich darstellt. Oft handelt es sich dabei um Entscheidungs- oder Optimierungsprobleme, die $\mathcal{NP}$-hart und damit im allgemeinen (bislang) nicht effizient lösbar sind. In etlichen Fällen lassen sich die Aufgabenstellungen im Kern auf bekannte Standardprobleme (wie das Finden von Hamiltonschen Kreisen oder Steinerbäumen in Graphen) abbilden, zu deren Lösung in der Praxis bislang meist spezielle heuristische Optimierungs- und Suchverfahren eingesetzt werden. Angesichts der wachsenden Bedeutung der molekularbiologischen Forschung und ihrer Einsatzbereiche handelt es sich hier um ein besonders aktuelles und wichtiges Anwendungsgebiet der Informatik, innerhalb dessen Methoden der Intellektik möglicherweise einen wichtigen Beitrag bei der Lösung der anfallenden algorithmischen Probleme liefern können.

Zielsetzung

Der ursprüngliche Ausgangspunkt dieser Arbeit war die Fragestellung, inwieweit spezielle algorithmische Probleme aus dem Bereich der Molekularbiologie in geeignet kodierter Form durch leistungsfähige Modellfindungsverfahren für die Aussagenlogik gelöst werden können. Die praktische Anwendbarkeit dieser Vorgehensweise ist jedoch in entscheidendem Maße von den Eigenschaften der verwendeten SAT-Kodierung und der zur Lösung eingesetzten SAT-Verfahren abhängig. In diesem Zusammenhang sind folgende Punkte von besonderem Interesse:

- Welche Probleme aus dem Bereich der Molekularbiologie eignen sich besonders für die vorgeschlagene Vorgehensweise?
- Wie kann die Transformation eines speziellen Problems in die Aussagenlogik auf möglichst effiziente Weise vorgenommen werden?
- Besitzen die so erzeugten Formeln eine besondere Struktur, die möglicherweise für den zu verwendenden SAT-Algorithmus bedeutsam ist?
- Inwieweit kann die Leistung der bekannten, GSAT-ähnlichen Verfahren durch Parallelisierung oder Modifikation der entsprechenden Algorithmen verbessert werden?
- Kann die vorgeschlagene Vorgehensweise, angewandt auf spezielle molekularbiologische Probleme, bezüglich Lösungsqualität und Zeitaufwand mit den bekannten, speziellen Lösungsverfahren konkurrieren?

Als molekularbiologisches Anwendungsproblem wurde in dieser Arbeit ein $\mathcal{NP}$-hartes, kombinatorisches Problem aus dem Bereich der physikalischen Genomkartierung betrachtet, welches nach einem neueren Resultat von Karp auf ein spezielles Traveling-Salesman-Problem (TSP) abgebildet werden kann. Um eine solche TSP-Instanz wiederum auf ein verallgemeinertes SAT-Problem abzubilden, wird eine Erweiterung der von Iwama und Miyazaki entwickelte Methode zur effizienten Transformation des strukturell eng verwandten Hamilton-Kreis-Problems in Graphen (HCP) auf das Erfüllbarkeitsproblem der Aussagenlogik (SAT) eingesetzt.

Da bereits bei der praktischen Lösung der so transformierten Hamilton-Kreis-Probleme mithilfe von GSAT-ähnlichen Verfahren eine Reihe prinzipieller Probleme auftreten, konzentriert sich der Hauptteil der Arbeit auf die Untersuchung der folgenden Fragestellungen:

- Ist GSAT ein praktikables Verfahren zur Lösung von SAT-kodierten Hamilton-Kreis-Problemen (HCP) in gerichteten Graphen?
- Läßt sich die Performanz durch problemspezifische Anpassungen verbessern?
- Welche Varianten wirken sich günstig auf die Performanz des Verfahrens aus?
- Welcher Zusammenhang besteht zwischen verschiedenen Parametereinstellungen und Performanz?
- Können optimale Parametereinstellungen identifiziert werden?
- Wie sind die untersuchten Verfahren zu bewerten?
- Ist GSAT als adäquates Verfahren zur Lösung des HCP zu betrachten?

Einige dieser Fragestellungen sind sehr komplex und nur mit hohem experimentellem Aufwand zu untersuchen. Aufgrund des begrenzten Umfangs und zeitlichen Rahmens einer Diplomarbeit und der Tatsache, daß viele Aspekte der Thematik bislang praktisch noch nicht untersucht wurden, können nicht alle der aufgeworfenen Fragestellungen in diesem Zusammenhang erschöpfend behandelt werden. Insbesondere die praktische Anwendung des untersuchten Ansatzes auf das molekularbiologische Problem wird daher im Rahmen dieser Arbeit nur ansatzweise behandelt.

Aufbau der Arbeit

Die vorliegende Arbeit ist wie folgt gegliedert: Nach dieser Einleitung werden in Kapitel 1 die später benötigten Grundlagen aus dem Bereich der Aussagenlogik dargestellt; insbesondere wird hier das zentrale aussagenlogische Erfüllbarkeitsproblem (SAT) formal definiert. Kapitel 2 gibt einen Überblick über einige der wesentlichen Ansätze zur algorithmischen Lösung des SAT-Problems; hier wird der später ausführlich untersuchte GSAT-Algorithmus eingeführt.

Im folgenden Kapitel 3 wird ein algorithmischer Rahmen für SAT-Algorithmen (GenSAT) entwickelt, anhand dessen verschiedene SAT-Verfahren auf einheitliche und übersichtliche Weise formuliert werden können. Auf diese Weise wird eine Grundlage für die systematische Untersuchung einer ganzen Familie von SAT-Verfahren geschaffen. Hier werden auch einige für die praktische Anwendung der Verfahren sehr bedeutsame Implementierungsfragen behandelt. Das anschließende Kapitel 4 beschäftigt sich mit der Evaluierung von SAT-Verfahren. In diesem Bereich ist man bislang weitestgehend auf empirische Methoden angewiesen; die hier angestellten Überlegungen bilden die Grundlage für die im Rahmen dieser Arbeit durchgeführten experimentellen Untersuchungen. In Kapitel 5 werden

Möglichkeiten zur Parallelisierung der zuvor betrachteten GenSAT-Verfahren untersucht; angesichts der Anforderungen aus dem Anwendungsbereich einerseits und der wachsenden Bedeutung paralleler Rechnerarchitekturen andererseits handelt es sich hierbei um einen wichtigen Gesichtspunkt bei Entwurf, Bewertung und Anwendung von SAT-Verfahren.

Die Kapitel 6, 7 und 8 bilden den inhaltlichen Schwerpunkt der Arbeit und behandeln die Transformation von Hamilton-Kreis-Problemen in Graphen (HCP) auf aussagenlogische SAT-Probleme und deren Lösung mit den zuvor beschriebenen SAT-Verfahren. In Kapitel 6 werden Verfahren zur SAT-Kodierung von HCP-Instanzen vorgestellt und diskutiert; inbesondere werden Eigenschaften der so gewonnenen CNF-Formeln untersucht, welche für die Behandlung mit GSAT-ähnlichen Verfahren bedeutsam sind. Das folgende Kapitel 7 ist der Betrachtung von Verfahren zur Lösung der SAT-kodierten HCP-Instanzen gewidmet. Hierbei werden zunächst Probleme, die bei der Verwendung herkömmlicher GSAT-Varianten auftreten, diskutiert; anschließend wird gezeigt, wie das allgemeine GSAT-Verfahren in mehreren Schritten an die spezielle Struktur von SAT-kodierten HC-Problemen angepaßt werden kann, um diese Probleme zu beheben. Kapitel 8 beinhaltet eine ausführliche Darstellung der experimentellen Untersuchung der zuvor entwickelten Verfahren zur Lösung des kodierten HCP. Anhand der Versuchsergebnisse werden grundsätzliche Eigenschaften der Verfahren aufgezeigt und diskutiert. Ein Schwerpunkt liegt dabei auf der Untersuchung des Performanzverhaltens der betrachteten Algorithmen. Die hier gezeigten Resultate tragen zum einen zur Beantwortung der Frage nach der Adäquatheit der Verfahren bei und leisten damit einen Beitrag zur Evaluierung der Algorithmen; zum anderen erlauben sie eine Abschätzung der maximalen Größe der mithilfe dieses Ansatzes lösbaren Probleminstanzen.

In Kapitel 9 wird schließlich demonstriert, wie die zuvor untersuchte Vorgehensweise prinzipiell auf ein $\mathcal{NP}$-hartes algorithmisches Problem aus dem Bereich der physikalischen Genomkartierung übertragen werden kann. Die Arbeit schließt mit einem Resumé und Ausblick, in dem die wichtigsten Resultate sowie die sich hieraus ergebenden, weiterführenden Fragestellungen zusammengefaßt sind.

Kapitel 1

Das Erfüllbarkeitsproblem (SAT) in der Aussagenlogik

Dieses Kapitel beginnt mit einer kurzen Einführung in einige Grundlagen der Aussagenlogik, die für das Verständnis der weiteren Ausführungen benötigt werden. Anschließend wird das für diese Arbeit zentrale Erfüllbarkeitsproblem der Aussagenlogik definiert und der Zusammenhang mit dem eng verwandten Gültigkeitsproblem und dem Problem der Normalformtransformation dargestellt. Die in diesem Rahmen angeführten Resultate und ihre Beweise finden sich, ebenso wie die zugehörigen Definition, in den gängigen Lehrbüchern (beispielsweise [HW92]).

Das Kapitel schließt mit einer Einführung in das Konzept der Penalty-Logik (siehe [Pin92]); hierbei handelt es sich um eine nichtmonotone Erweiterung der Aussagenlogik, welche eine einfache Formalisierung von Optimierungsproblemen (siehe Kapitel 9) ermöglicht. Die Darstellung orientiert sich an der Arbeit von Gadi Pinkas; lediglich das Konzept der P-Erfüllbarkeit und die Definition des MaxSAT-Problems wurden ergänzt (vergl. hierzu auch [JKS95]).

1.1 Die Sprache der Aussagenlogik

Die Aussagenlogik basiert auf einer formalen Sprache, deren Elemente Aussagenvariablen, Wahrheitswerte und Junktoren sind; Aussagenvariablen und Wahrheitswerte werden durch Junktoren zu Formeln verknüpft, welche die aussagenlogischen Sätze repräsentieren. Die Syntax der aussagenlogischen Sprache kann folgendermaßen formal definiert werden:

Definition 1.1 (Syntax der Aussagenlogik)

$S = V \cup C \cup J \cup \{(,)\}$ *ist die* Symbolmenge der Aussagenlogik, *dabei ist* $V = \{x_i \mid i \in \mathbb{N}\}$ *die abzählbar unendliche Menge der* Aussagenvariablen, $C = \{\top, \bot\}$ *die Menge der*

Wahrheitswerte *und* $J = \{\neg, \wedge, \vee\}$ *die Menge der* Junktoren.

Die Menge der aussagenlogischen Formeln *ist folgendermaßen induktiv über der Symbolmenge* S *definiert:*

- *die Wahrheitswerte* $\top$ *und* $\bot$ *sind aussagenlogische Formeln;*
- *jede Aussagenvariable* $x_i \in V$ *ist eine aussagenlogische Formel;*
- *ist* F *eine aussagenlogische Formel, so auch* $\neg F$*;*
- *sind* F_1 *und* F_2 *aussagenlogische Formeln, so auch* $(F_1 \wedge F_2)$ *und* $(F_1 \vee F_2)$.

Auf diese Weise sind alle aussagenlogischen Formeln beschrieben.

Variablenbelegungen ordnen den Aussagenvariablen einer Formel Wahrheitswerte zu; gemäß den üblichen Definitionen der Junktoren auf diesen Wahrheitswerten läßt sich der Wert einer Formel unter einer gegebenen Belegung bestimmen. Auf diese Weise kann die Semantik der Aussagenlogik formal definiert werden:

Definition 1.2 (Semantik der Aussagenlogik)

Die Variablenmenge $Var(F)$ einer Formel F *ist die Menge aller in* F *auftretenden Aussagenvariablen.*

Eine Variablenbelegung der Formel F *ist eine Abbildung* $B : Var(F) \mapsto C$ *der Variablenmenge von* F *in die Menge der Wahrheitswerte. Die* Menge aller Variablenbelegungen der Formel F *wird mit* $Assign(F)$ *bezeichnet.*

Der Wert $Val_B(F)$ einer Formel F unter einer Belegung B *ist induktiv über die syntaktische Struktur von* F *definiert:*

- $F \equiv \top \Longrightarrow Val_B(F) = \top$
- $F \equiv \bot \Longrightarrow Val_B(F) = \bot$
- $F \equiv x_i \in V \Longrightarrow Val_B(F) = B(x_i)$
- $F \equiv \neg F_1 \Longrightarrow Val_B(F) = \neg\, Val_B(F_1)$
- $F \equiv F_1 \wedge F_2 \Longrightarrow Val_B(F) = Val_B(F_1) \wedge Val_B(F_2)$
- $F \equiv F_1 \vee F_2 \Longrightarrow Val_B(F) = Val_B(F_1) \vee Val_B(F_2)$

Die Wahrheitswerte $\top$ *und* $\bot$ *werden auch als* verum (wahr) *und* falsum (falsch) *bezeichnet; die Operatoren* $\neg$ *(*Negation*),* $\wedge$ *(*Konjunktion*) und* $\vee$ *(*Disjunktion*) sind auf diesen folgendermaßen definiert:*

$\neg$	
$\top$	$\bot$
$\bot$	$\top$

$\wedge$	$\top$	$\bot$
$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

$\vee$	$\top$	$\bot$
$\top$	$\top$	$\top$
$\bot$	$\top$	$\bot$

Da die Variablenmenge einer aussagenlogische Formel immer endlich viele Elemente enthält, ist auch die Anzahl der verschiedenen Variablenbelegungen zu einer gegebene Formel immer endlich; dabei besitzt eine Formel über n Variablen genau 2^n verschiedene Variablenbelegungen.

Durch Betrachtung der Werte einer Formel unter der Gesamtheit dieser Belegungen lassen sich die fundamentalen Begriffe Gültigkeit und Erfüllbarkeit definieren:

Definition 1.3 (Erfüllbarkeit & Gültigkeit)

Eine Variablenbelegung B heißt Modell einer Formel F, *wenn gilt* $Val_B(F) = \top$; *man sagt dann auch,* B erfüllt F.

Eine Formel F wird als (allgemein-)gültig *bezeichnet, wenn sie von allen denkbaren Variablenbelegungen erfüllt wird. Allgemeingültige Formeln heißen auch* Tautologien.

Besitzt eine Formel F mindestens ein Modell, so wird F erfüllbar *genannt.*

Oft werden Normalformen, d.h. syntaktisch eingeschränkte Klassen von aussagenlogischen Formeln, betrachtet. Die üblichen Normalformen aussagenlogischer Formeln werden dadurch definiert, daß die Verschachtelungstiefe und das Auftreten der Junktoren auf den einzelnen Ebenen der Verschachtelungsstruktur beschränkt wird.

Definition 1.4 (Normalformen)

Aussagenvariablen und ihre negierten Formen werden als Literale *bezeichnet.*

Formeln der Form $F = G_1 \wedge G_2 \wedge \ldots \wedge G_n$, wobei die G_i für beliebige aussagenlogische Formeln stehen, heißen Konjunktionen. *Analog werden Formeln der Form $F = G_1 \vee G_2 \vee \ldots \vee G_n$ als* Disjunktionen *bezeichnet.*

Eine Formel F ist in konjunktiver Normalform (CNF), *wenn sie als Konjunktion von Disjunktionen über Literalen $l_{i,j}$ vorliegt, also die Gestalt $F = c_1 \wedge c_2 \wedge \ldots \wedge c_n$ hat, wobei die c_i die Form $c_i = l_{i,1} \vee l_{i,2} \vee \ldots \vee l_{i,k_i}$ besitzen. Die Disjunktionen c_i werden als* Klauseln der Formel F *bezeichnet; dabei ist die* Länge $|c_i|$ einer Klausel c_i *als die Anzahl k_i ihrer Literale definiert.*

Auf analoge Weise ist eine Formel F in disjunktiver Normalform, *wenn sie als Disjunktion von Konjunktionen über Literalen vorliegt.*

Eine Formel F ist in konjunktiv-disjunktiver Normalform (CDNF), wenn sie in Form einer

Konjunktion von DNF-Formeln vorliegt.

Anmerkung: Die so definierten Normalformen enthalten Negationen nur auf der Ebene von Literalen; die maximale Schachtelungstiefe der Junktoren beträgt für CNF- bzw. DNF-Formeln maximal 2, für CDNF-Formeln maximal 3.

Beispiel: Die Formel $(x \wedge \neg y) \vee (\neg x \wedge y)$ ist in DNF; $(x \vee y) \wedge (\neg x \vee \neg y)$ ist eine CNF-Formel. Die Formel $x \wedge [(x \wedge \neg y) \vee (\neg x \wedge y)]$ befindet sich in CDNF; $\neg(x \wedge y)$ ist keine Normalformformel im oben definierten Sinne, da sie eine Negation eines zusammengesetzten Teilausdrucks enthält.

Für die Semantik einer aussagenlogischen Formel in CNF (oder DNF) spielt die Reihenfolge der Literale innerhalb der Klauseln ebenso wie die Reihenfolge der Klauseln innerhalb der Formel keine Rolle (Kommutativität der Junktoren). Weiterhin können mehrfache Auftreten von identischen Klauseln oder von Literalen innerhalb derselben Klausel vernachlässigt werden, ohne die Semantik der Formel zu beeinflussen (Idempotenz der Junktoren). Infolgedessen können die Klauseln einer CNF-Formel auch als Mengen von Literalen geschrieben werden und die Formel selbst als Menge über diesen Mengen. Die Schreibweise $c_i \in F$ bedeutet dann, daß c_i eine Klausel der Formel F ist; analog steht $l_{i,j} \in c_i$ für die Aussage „das Literal $l_{i,j}$ tritt in der Klausel c_i auf“.

Beispiel: Die Mengenschreibweise der CNF-Formel $F = (x \vee y) \wedge (\neg x \vee \neg y)$ ist gegeben durch $\{\{x, y\}, \{\neg x, \neg y\}\}$. Damit gilt $c_1 = (x \vee y) \in F$ und $x \in c_1$.

Die im folgenden definierte logische Äquivalenz von Formeln setzt syntaktisch verschiedene Formeln in Beziehung, welche bedeutungsgleich im Sinne der Semantik sind:

Definition 1.5 (Logische Äquivalenz von Formeln)

Zwei aussagenlogische Formeln F und G werden als äquivalent *bezeichnet, wenn sie unter jeder Variablenbelegung den gleichen Wert annehmen, wenn also für jede beliebige Belegung B gilt: $Val_B(F) = Val_B(G)$.*

Die wichtigste Eigenschaft einer Klasse von Normalformen besteht darin, daß zu jeder aussagenlogischen Formel mindestens eine äquivalente Formel in Normalform existiert (in der Regel besitzen aber mehrere Formeln die gleiche Normalform). Dies ist für die oben eingeführten Klassen CNF, DNF und CDNF gewährleistet, wie das folgende Resultat zeigt.

Satz 1.1 Normalformtransformation

Zu jeder aussagenlogischen Formel F kann eine äquivalente Formel F' in CNF (DNF,CDNF) konstruiert werden. Dieser als Normalformtransformation bezeichnete Vorgang kann mithilfe eines deterministischen Algorithmus in exponentieller Zeit (bezogen auf die Länge von F)

durchgeführt werden.

Der Beweis dieser Aussage kann auf einfache Weise unter der Verwendung der *De Morganschen Gesetze* geführt werden (vergl. [HW92], [Bib92]). Durch weitere Einschränkungen kann erreicht werden, daß die entsprechende Normalform zu einer gegebenen Formel immer eindeutig bestimmt ist (sog. kanonische Normalformen). Die konjunktive Normalform läßt sich darüberhinaus durch die Normierung der Klausellänge weiter verschärfen; die im folgenden definierten Klassen der k-CNF-Formeln sind besonders für komplexitätstheoretische Überlegungen nützlich (vergl. Abschnitt 1.2):

Definition 1.6 (k-CNF-Formeln)

Eine Formel F wird als k-CNF-Formel *bezeichnet, wenn sie in konjunktiver Normalform vorliegt, wobei alle Klauseln exakt die Länge k besitzen.*

Beispiel: Die Formel $(x \vee y) \wedge (\neg x \vee \neg y)$ ist in 2-CNF; $(x \vee \neg y \vee z)$ ist eine 3-CNF-Formel (mit einer Klausel).

Anmerkung: Üblicherweise wird bei der Definition der Länge einer Klausel c_i davon ausgegangen, daß c_i keine mehrfachen Auftreten von Literalen enthält (vergl. obige Anmerkungen zur Mengenschreibweise von Formeln).

1.2 Das Erfüllbarkeitsproblem (SAT)

Das Erfüllbarkeitsproblem der Aussagenlogik besteht darin, zu einer gegebenen aussagenlogischen Formel zu entscheiden, ob diese ein Modell besitzt.

Definition 1.7 (SAT-Problem)

Gegeben sei eine aussagenlogische Formel F. Das Erfüllbarkeitsproblem (Satisfiability Problem; SAT) *besteht in der Entscheidung, ob zu der Formel F eine Belegung B existiert, sodaß gilt: $Val_B(F) = \top$.*

Das so definierte SAT-Problem gehört zur Komplexitätsklasse $\mathcal{NP}$; $\mathcal{NP}$ ist die Klasse aller Probleme, die auf einer nichtdeterministischen Turingmaschine in polynomiell von der Problemgröße abhängiger Zeit gelöst werden können. Probleme dieser Klasse, die nur dann von einer *determininistischen* Maschine in polynomieller Zeit gelöst werden können, wenn dies gleichzeitig für *alle* Probleme aus $\mathcal{NP}$ zutrifft, werden auch als $\mathcal{NP}$*-hart* bezeichnet (solche

Probleme müssen nicht unbedingt selbst zur Klasse $\mathcal{NP}$ gehören, sie sind jedoch in einem gewissen Sinne mindestens genauso schwierig wie die schwierigsten $\mathcal{NP}$-Probleme). $\mathcal{NP}$-harte Probleme, die gleichzeitig zur Klasse $\mathcal{NP}$ gehören, werden *$\mathcal{NP}$-vollständig* genannt; derartige Probleme sind nach dem derzeitigen Erkenntnisstand allgemein nur mit exponentiell von der Problemgröße (in diesem Falle: Länge der Formel) abhängigem Zeitaufwand deterministisch lösbar und sind damit als besonders schwierige Probleme einzustufen.

Das Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist $\mathcal{NP}$-vollständig; wie alle übrigen $\mathcal{NP}$-vollständigen Probleme ist es sogar prototypisch für die Komplexitätsklasse $\mathcal{NP}$ in dem Sinne, daß sich alle anderen $\mathcal{NP}$-Probleme mit polynomiellem Zeitaufwand auf SAT reduzieren lassen. Historisch gesehen ist das Erfüllbarkeitsproblem das erste Problem, dessen $\mathcal{NP}$-Vollständigkeit nachgewiesen wurde.

Satz 1.2 (Cook, 1971)

Das Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist für Formeln in konjunktiver Normalform (CNF) $\mathcal{NP}$-vollständig.

Der Beweis dieses Satzes nutzt die Tatsache, daß zu jeder Formel eine äquivalente Formel in 3-CNF existiert und basiert im wesentlichen auf einer Kodierung der von einer nichtdeterministischen Turing-Maschine ausgeführten Berechnung in eine 3-CNF-Formel (vergl. [Rei90]). Das Erfüllbarkeitsproblem für k-CNF-Formeln wird auch als k-SAT-Problem bezeichnet. Aus dem angegeben Satz folgt direkt die $\mathcal{NP}$-Vollständigkeit des allgemeinenen Erfüllbarkeitsproblems der Aussagenlogik.

Bislang wurde das Erfüllbarkeitsproblem als Entscheidungsproblem betrachtet; für praktische Anwendungen ist es jedoch oft bedeutsam, im Falle der Erfüllbarkeit auch mindestens eine erfüllende Belegung zu kennen (vergl. Kapitel 6). Diese Variante wird auch als *Modellfindungsproblem* bezeichnet und ist (ebenso wie das SAT-Problem) $\mathcal{NP}$-hart (und damit vermutlich nicht allgemein deterministisch mit polynomiellem Zeitaufwand lösbar).

Der Beweis hierfür wird durch einen Algorithmus geliefert, welcher die Bestimmung einer erfüllenden Belegung auf die Lösung einer Anzahl von SAT-Problemen reduziert; dieses Transformationsverfahren weist eine linear von der Anzahl der in der Formel auftretenden Variablen abhängige Laufzeit auf (siehe [Rei90]). Aufgrund dieses Zusammenhangs wird das Modellfindungsproblem in dieser Arbeit fortan häufig unter die Bezeichnung „SAT-Problem" subsumiert.

1.3 Normalformtransformation und Gültigkeitsproblem

Die Komplexität des Erfüllbarkeits- bzw. Modellfindungsproblems ist eng an die Darstellung der Formel gebunden: Zwar ist das Problem für konjunktive Normalform (CNF) $\mathcal{NP}$-hart, es kann jedoch für DNF-Formeln in linearer Zeit entschieden werden, da die Klauseln der disjunktiven Normalform direkt den erfüllenden Belegungen der Formel entsprechen. Das SAT-Problem kann daher auf die Transformation der gegebenen Formel in DNF reduziert werden; bei derartigen Äquivalenzumformungen zwischen verschiedenen Normalformen handelt es sich um Normalformtransformationen im oben definierten Sinne.

Das einfachste derartige Transformationsverfahren beruht auf Äquivalenzumformungen nach den De Morganschen Regeln und den Distributivgesetzen (wegen deren Symmetrie das Verfahren sowohl von CNF nach DNF als auch umgekehrt funktioniert) und wird auch als *Distributionsverfahren* bezeichnet; es entspricht im wesentlichen einem „Ausmultiplizieren" der Klauseln. Dieses Verfahren verursacht im allgemeinen eine exponentielle Aufblähung der Formel (bezüglich der Anzahl ihrer Klauseln); zumindest erfordert es aber die Betrachtung einer expontiellen Anzahl von Literalkombinationen (vergl. [Bib92]).

Ein alternativer Ansatz wird als *definitorische Normalformtransformation* bezeichnet; dieser beruht im wesentlichen auf der Einführung neuer Variablen als Namen für Teilformeln der zu transformierenden Formel. Auch dieses Verfahren kann jedoch den Effekt der exponentiellen Aufblähung im allgemeinen nicht vermeiden (siehe [Bib92]).

In der Aussagenlogik ist (ebenso wie in der Prädikatenlogik) eine bemerkenswerte Symmetrie der Begriffe und Sätze vorzufinden, die letztendlich auf der Dualität der Junktoren „$\wedge$" und „$\vee$" beruht. In diesem Sinne läßt sich dual zum Erfüllbarkeitsproblem das *Gültigkeitsproblem für die Aussagenlogik (Validity Problem; VAL)* formulieren; dieses besteht in der Entscheidung, ob eine gegebene aussagenlogische Formel allgemeingültig ist. Zwischen Erfüllbarkeits- und Gültigkeitsproblem besteht folgender Zusammenhang:

Lemma 1.1

Eine aussagenlogische Formel F ist genau dann allgemeingültig, wenn ihre negierte Form $\neg F$ unerfüllbar ist.

Infolgedessen kann ein allgemeines Verfahren zur Lösung des SAT-Problems auch dazu benutzt werden, das Gültigkeitsproblem zu entscheiden. Da die Negation einer CNF-Formel mithilfe der De Morganschen Regeln leicht in linearer Zeit in eine äquivalente DNF-Formel umgeformt werden kann (und umgekehrt), läßt sich direkt ein Zusammenhang zwischen der Komplexität des Erfüllbarkeits- und des Gültigkeitsproblems auf verschiedenen Normalformen formulieren:

Satz 1.3 Zeitkomplexität von SAT & VAL

Das Gültigkeitsproblem (VAL) für DNF-Formeln und das Erfüllbarkeitsproblem (SAT) für CNF-Formeln sind $\mathcal{NP}$-hart.

Dagegen sind VAL für CNF-Formeln sowie SAT für DNF-Formeln in linearer Zeit entscheid-

bar.

Wegen der $\mathcal{NP}$-Vollständigkeit des SAT-Problems für CNF-Formeln ist auch das CNF-DNF-Transformationsproblem $\mathcal{NP}$-hart; andernfalls könnte das SAT-Problem für CNF-Formeln durch vorgeschaltete Transformation der Eingabeformel in DNF effizient (d.h. in polynomieller Zeit) gelöst werden. Die CNF-DNF-Transformation kann daher bislang nur mit exponentiellem Zeitaufwand deterministisch gelöst werden.

1.4 Nichtmonotone Logik und maximale Erfüllbarkeit

Um bestimmte Aspekte eines Anwendungsgebietes besser logisch modellieren zu können, werden oft Varianten und Erweiterungen der klassischen Aussagenlogik, wie sie in den vorhergehenden Abschnitten eingeführt wurde, betrachtet. Im Rahmen dieser Arbeit ist dabei besonders die sogenannte *Penalty-Aussagenlogik* (kurz Penalty-Logik) von Interesse. Diese unterscheidet sich von der klassischen Aussagenlogik dadurch, daß lediglich CNF-Formeln betrachtet werden, wobei jeder Klausel ein Gewicht zugeordnet wird. Die Syntax der Penalty-Logik kann damit wie folgt definiert werden:

Definition 1.8 (Syntax der Penalty-Logik)

Eine Klausel der Penalty-Logik *ist ein Paar $\gamma = (c, p)$ aus einer aussagenlogischen CNF-Klausel c und einer natürlichen Zahl $p > 0$, welche als* Gewicht der Klausel *γ bezeichnet wird.*

Eine penalty-logische Formel *$\Phi = \gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n$ ist eine Konjunktion von Klauseln γ_i der Penalty-Logik.*

Anmerkung: Die für CNF-Formeln der klassischen Aussagenlogik eingeführte Mengenschreibweise kann analog für penalty-logische Formeln verwendet werden.

Beispiel: $\Phi = (x \vee y,\ 1) \wedge (\neg x \vee \neg y,\ 2)$ ist eine penalty-logische Formel; die Mengenschreibweise für Φ lautet $\{(\{x, y\},\ 1), (\{\neg x, \neg y\},\ 2)\}$.

Der Penalty-Wert einer Formel unter einer gegebenen Variablenbelegung ist als die Summe der Gewichte aller (aussagenlogisch) unerfüllten Klauseln definiert:

Definition 1.9 (Semantik der Penalty-Logik)

Der Penalty-Wert *Pen_B* einer Klausel *$\gamma = (c, p)$* unter einer Variablenbelegung *B ist definiert* *als:*

$$Pen_B(\gamma) = \begin{cases} p & : \quad Val_B(c) = \bot \\ 0 & : \quad sonst \end{cases}$$

Der Penalty-Wert Pen_B einer Formel $\Phi = \gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n$ unter der Belegung B *ist definiert als*

$$Pen_B(\Phi) = \sum_{i=1}^{n} Pen_B(\gamma_i)$$

Beispiel: Für $\Phi = (x \vee y,\ 1) \wedge (\neg x \vee \neg y,\ 2)$ und die Belegung $B(x) = B(y) = \bot$ ergibt sich ein Penalty-Wert von $Pen_B(\Phi) = 1$, für $B'(x) = B'(y) = \top$ gilt $Pen_{B'}(\Phi) = 2$.

Die Penalty-Logik eignet sich besonders zur Modellierung von Optimierungsproblemen, welche neben starken Lösungsbedingungen (*"hard-constraints"*, welche immer erfüllt sein müssen) auch schwache Bedingungen (*"soft-constraints"*) umfassen; letztere dienen typischerweise zur Modellierung der Kosten einer Lösung (siehe Kapitel 9).

Der klassischen Erfüllbarkeit entspricht in der Penalty-Logik die Existenz einer Belegung, unter welcher der Penalty-Wert der gegebenen Formel eine bestimmte Schranke nicht überschreitet (*SAT-Problem für die Penalty-Logik*); diese Schranke sowie die Gewichte der Klauseln werden üblicherweise so gewählt, daß jede erfüllende Belegung mindestens alle starken Bedingungen im klassischen Sinne erfüllt.

Definition 1.10 (P-Erfüllbarkeit & SAT$_P$-Problem)

Eine Variablenbelegung B heißt Modell der penalty-logischen Formel Φ bezüglich der Schranke P (P-Modell von Φ), *wenn gilt:*

$$Pen_B(\Phi) \leq P$$

Eine penalty-logische Formel Φ ist erfüllbar bezüglich der Schranke P (oder P-erfüllbar), *wenn sie mindestens ein P-Modell besitzt.*

Das Erfüllbarkeitsproblem bezüglich der Schranke P (SAT$_P$) *besteht in der Entscheidung, ob eine gegebene Formel Φ der Penalty-Logik P-erfüllbar ist.*

Beispiel: Die Belegung B mit $B(x) = B(y) = \bot$ ist ein 1-Modell der Formel $\Phi = (x \vee y,\ 1) \wedge (\neg x \vee \neg y,\ 2)$. Damit ist Φ 1-erfüllbar. Weitherhin ist Φ 0-erfüllbar, ein 0-Modell ist gegeben durch $B''(x) = \top, B''(y) = \bot$.

Die Penalty-Logik kann als echte Verallgemeinerung der Aussagenlogik über der syntaktisch eingeschränkten Klasse der CNF-Formeln betrachtet werden, wie im folgenden gezeigt wird.

Definition 1.11 (Penalty-logische Erweiterung aussagenlogischer Formeln)

Die penalty-logische Erweiterung *einer aussagenlogischen CNF-Formel* $F = c_1 \wedge c_2 \wedge \ldots \wedge c_k$ *ist definiert als* $\Phi(F) := (c_1, 1) \wedge (c_2, 1) \wedge \ldots \wedge (c_k, 1)$.

Über den Begriff der penalty-logischen Erweiterung kann nun der Zusammenhang zwischen Erfüllbarkeit in der Penalty-Logik und in der klassischen Aussagenlogik hergestellt werden:

Satz 1.4

Eine aussagenlogische CNF-Formel F ist genau dann erfüllbar, wenn ihre penalty-logische Erweiterung $\Phi(F)$ 0-erfüllbar ist.

Anmerkung: Die 0-Erfüllbarkeit einer penalty-logischen Formel Φ entspricht damit der Erfüllbarkeit im Sinne der klassischen Aussagenlogik, da jedes 0-Modell von Φ den aussagenlogischen Teil aller Klauseln von Φ erfüllt.

Für die Modellierung vieler Anwendungssituationen ist es vorteilhaft, auch solches Wissen in logischer Form repräsentieren und verarbeiten zu können, welches Widersprüche enthält. Die entsprechenden Formeln besitzen dann zwar keine Modelle im Sinne der Aussagenlogik, wohl aber P-Modelle für verschiedene Penalty-Schranken $P > 0$. Besonders für die Modellierung von Optimierungsproblemen kann dieser Ansatz benutzt werden. Dabei kodiert man die Lösungsbedingungen als Klauseln, wobei die Penalty-Werte den Kosten für die Erfüllung der jeweiligen Bedingung entsprechen. Damit entsprechen die P-Modelle mit minimaler Penalty-Schranke P den optimalen Lösungen des Problems.

Zur formalen Beschreibung dieser Vorgehensweise wird zunächst der Begriff des Gewichts von Belegungen definiert:

Definition 1.12 (Gewicht von Belegungen)

Zu einer gegebenen Formel $\Phi = (c_1, p_1) \wedge (c_2, p_2) \wedge \ldots \wedge (c_k, p_k)$, *ist das* Gewicht einer Belegung B *definiert als*

$$W(B, \Phi) := \sum_{i=1}^{k} p_i - Pen_B(\Phi)$$

Anmerkung: Das Gewicht einer Belegung B entspricht genau der Summe der Gewichte der durch B erfüllten Klauseln.

Für den Anwendungsfall relevant ist nun das Problem, zu einer gegebenen Formel ein Modell maximalen Gewichts zu finden (MaxSAT-Problem[1]), welches als Optimierungsvariante des Modellfindungsproblems der klassischen Aussagenlogik betrachtet werden kann. Formal kann dieses Problem wie folgt definiert werden:

Definition 1.13 (MaxSAT-Problem)

Das Problem der maximalen Erfüllbarkeit (MaxSAT) *besteht darin, zu einer gegebenen penalty-logischen Formel Φ und einer Penalty-Schranke P ein P-Modell B zu finden, sodaß für jedes andere P-Modell B' gilt:*

$$W(B, \Phi) \geq W(B', \Phi)$$

Ein Modell mit dieser Eigenschaft wird als maximales P-Modell von Φ *bezeichnet.*

Beispiel: Die Formel $\Phi = (x \vee y, 1) \wedge (\neg x \vee \neg y, 2)$ besitzt genau zwei maximale 0-Modelle mit Gewicht 3, eines davon ist gegeben durch die Belegung $B''(x) = \top, B''(y) = \bot$. B'' ist außerdem maximales P-Modell von Φ für alle $P > 0$. Die Formel $\Phi' = \Phi \cup \{(z, 0.5), (\neg z, 0.5)\}$ besitzt kein 0-Modell, sie ist also nicht erfüllbar im aussagenlogischen Sinne. Allerdings weist Φ' 4 maximale 0.5-Modelle mit Gewicht 3.5 auf.

Anhand des Begriffs des maximalen Modells kann ein grundlegender Unterschied zwischen der klassischen Aussagenlogik und der Penalty-Logik verdeutlicht werden: In der klassischen Aussagenlogik kann eine CNF-Formel durch das Hinzufügen von Klauseln niemals neue Modelle gewinnen; bei Erweiterung der Formel nimmt die Anzahl der Modelle also monoton ab. Diese Eigenschaft gilt nicht für maximale Modelle in der Penalty-Logik: Hier ist es möglich, daß ein nicht-maximales P-Modell einer Formel Φ durch Hinzufügen einer Klausel γ mit geeignetem Penalty-Wert zu einem maximalen P-Modell der so erweiterten Formel $\Phi \cup \gamma$ wird. (Betrachte hierzu die Formeln $\Phi^* = \Phi \cup (z, 0.5)$ und Φ' aus dem obigen Beispiel: Die Belegung B' mit $B'(x) = \top$ und $B'(y) = B'(z) = \bot$ ist maximales Modell von $\Phi' = \Phi^* \cup (\neg z, 0.5)$, nicht aber von Φ^*.) Damit kann die Penalty-Logik als nichtmonotone Logik[2] gekennzeichnet werden.

[1] Dies kann alternativ als Problem der Minimierung des Penalty-Wertes einer Formel unter Betrachtung aller Variablenbelegungen formuliert werden.

[2] Das Konzept der Nichtmonotonie wird üblicherweise auf eine andere Weise definiert (vergl. [Pin92]). Man kann sich jedoch leicht überlegen, daß die hier gewählte Charakterisierung äquivalent zur gebräuchlichen Definition ist.

Kapitel 2

Verfahren zur Lösung des SAT-Problems

In diesem Kapitel werden einige der prominentesten Verfahren zur algorithmischen Lösung des aussagenlogischen SAT-Problems vorgestellt. Nach einem kurzen Überblick über die wichtigsten generellen Ansätze zur Lösung $\mathcal{NP}$-harter Probleme folgt die ausführlichere Darstellung dreier konkreter SAT-Verfahren: Davis-Putnam, GSAT und Simulated Annealing. Dabei ist das GSAT-Verfahren von besonderem Interesse, da es im weiteren Verlauf der Arbeit ausführlich untersucht und diskutiert wird. Das Kapitel endet mit einem kurzen Überblick über einige der zahlreichen weiteren SAT-Verfahren, welche in jüngerer Vergangenheit entwickelt wurden.

2.1 Algorithmische Ansätze zur Lösung von $\mathcal{NP}$-harten Problemen

$\mathcal{NP}$-harte Probleme, wie das im letzten Kapitel eingeführte Erfüllbarkeits-Problem für die Aussagenlogik, können bislang nicht allgemein durch deterministische Algorithmen in polynomieller Zeit gelöst werden. In der Komplexitätstheorie finden sich einige Hinweise darauf, daß dies aus prinzipiellen Gründen nicht möglich ist (es handelt sich hierbei um das bis heute nicht entschiedene $\mathcal{P} = \mathcal{NP}$-Problem). Da zahlreiche für diverse Anwendungsgebiete äußerst interessante Probleme $\mathcal{NP}$-hart bzw. $\mathcal{NP}$-vollständig sind (darunter bekannte Optimierungsprobleme wie das Problem des Handlungsreisenden), besteht aus der Praxis heraus ein enormer Bedarf an Algorithmen zur Lösung derartiger Probleme.

Sämtliche bekannten Verfahren zur Lösung $\mathcal{NP}$-harter Probleme weisen im allgemeinen Fall ein exponentielles Laufzeitverhalten bezüglich der Eingabegröße auf, was in der Praxis meist nicht akzeptabel ist (da bei realistischen Problemen die Rechenzeiten selbst auf den schnellsten Rechnern astronomische Größen erreichen). Folglich versucht man seitens der

Informatik, wenigstens unter gewissen Einschränkungen brauchbare Verfahren zu realisieren. Die wichtigsten Ansätze hierzu sollen im folgenden kurz charakterisiert werden.

2.1.1 Einschränkung der Problemklasse

Daß ein Problem insgesamt $\mathcal{NP}$-hart ist, bedeutet in der Regel nicht, daß dies auch für jede Teilmenge von Eingaben gilt. Folglich ist es oft sinnvoll, nach geeigneten Einschränkungen des Problems zu suchen, für die deterministisch-polynomielle Algorithmen konstruierbar sind.

Im Falle des SAT-Problems der Aussagenlogik kann man etwa für die Teilmenge der 2-CNF-Formeln oder die der Horn-Formeln (CNF-Formeln, in denen jede Klausel höchstens ein positives Literal enthält) deterministische Verfahren mit polynomieller Zeitkomplexität angeben (vergl. [CL88, DG84]). Allerdings handelt es sich hierbei um stark eingeschränkte Formelklassen, mit denen sich nur noch bestimmte Anwendungsprobleme modellieren lassen.

Generell ist es jedoch oft lohnend, nach deterministisch-polynomiell lösbaren Spezialfällen eines $\mathcal{NP}$-harten Problems zu suchen; insbesondere dort, wo der Mensch in der Lage ist, das entsprechende Problem (bei realistischen Problemgrößen) zu lösen. Gemäß der Turing-Churchschen These wäre in diesen Fällen davon auszugehen, daß die real auftretenden Probleminstanzen eine geringere Komplexität besitzen (da es trotz der massiv-parallelen Verarbeitungsweise kaum denkbar erscheint, daß das menschliche Gehirn Probleme mit exponentiellem Aufwand löst, vergl. [Bib92]).

Ein interessantes Beispiel für eine solche Situation ist die Interpretation von Kartenskizzen. Hierbei handelt es sich um ein im allgemeinen Falle $\mathcal{NP}$-vollständiges Problem, welches allerdings in der Praxis mit einem seit 1975 bekannten Verfahren („Waltz labeling") effektiv gelöst werden kann. Erst 1994 konnte gezeigt werden, welche Konstellationen innerhalb der Kartenskizzen für die Komplexität des Problems verantwortlich sind („T-Junctions"); diese tauchen in der Praxis jedoch praktisch nie auf, womit das erstaunlich gute Laufzeitverhalten des Waltz-Algorithmus erklärt werden kann (siehe [Sel94]).

2.1.2 Approximationsverfahren

Viele $\mathcal{NP}$-Probleme sind ihrem Wesen nach Optimierungsprobleme; Beispiele hierfür sind das Problem des Handlungsreisenden (Travelling Salesman Problem; TSP) sowie MaxSAT (vergl. Abschnitt 1.4). Approximative Algorithmen dienen in diesen Fällen dazu, suboptimale Lösungen in polynomieller Zeit zu finden. Dabei soll möglichst garantiert werden, daß die relative Abweichung der vom Approximationsalgorithmus gelieferten Lösung zur optimalen Lösung eine gegebene Schranke nicht überschreitet.

Für viele $\mathcal{NP}$-harten Optimierungsprobleme sind derartige Approximationsalgorithmen bekannt (beispielsweise für verschiedene Spezialfälle des TSP, siehe [Meh84]); allerdings unterscheiden sich verschiedene $\mathcal{NP}$-harte Probleme bezüglich der Existenz und Güte von

Approximationsverfahren erheblich. So ist bekannt, daß unter der Voraussetzung $\mathcal{P} \neq \mathcal{NP}$ für das allgemeine TSP kein approximativer Polynomialzeitalgorithmus existieren kann (siehe [Meh84]).

2.1.3 Weitere Ansätze

Zwei grundlegende Techniken werden häufig zur praktischen Lösung $\mathcal{NP}$-harter Probleme benutzt: Dynamische Programmierung sowie Branch-and-Bound-Verfahren. Beide Verfahren sind im wesentlichen effizientere Varianten der vollständigen Suche im Raum der möglichen Lösungen; es handelt sich hierbei um vollständige Verfahren, deren Laufzeit im ungünstigsten Falle jedoch exponentiell ist.

Bei der dynamischen Programmierung wird die Lösung des ursprünglichen Problems auf die Lösung einer Menge kleinerer Teilprobleme reduziert, welche nach zunehmender Größe geordnet gelöst werden. Die Effizienz einer solchen Vorgehensweise ist von Struktur und Größe des Suchraums abhängig; gute Resultate werden auf diese Weise unter anderem für das TSP oder das Rucksackproblem (KNAPSACK) erzielt (siehe [Meh84]).

Branch-and-Bound-Verfahren basieren darauf, mithilfe einer Bewertung von partiellen Lösungen die Suche im Raum der potentiellen Lösungen zu steuern; dabei werden „vielversprechende“ partielle Lösungen bevorzugt weiterverfolgt. Der Erfolg dieses Ansatzes ist durch die Qualität der Bewertungsfunktion bestimmt; obwohl die theoretische Bestimmung der Zeitkomplexität derartiger Verfahren oft sehr schwierig ist, sind in der Praxis mithilfe von Branch-and-Bound-Verfahren brauchbare Algorithmen für viele Probleme (u.a. spezielle Varianten des TSP) realisierbar (siehe [Meh84]).

Schließlich sind noch heuristische Verfahren zu erwähnen; damit sind algorithmische Verfahren gemeint, die in der Praxis gute Laufzeiteigenschaften zeigen, welche jedoch nicht durch theoretische Resultate erklärt werden können. In vielen Fällen handelt es sich auch hierbei um eine Steuerung der Suche im Raum der möglichen Lösungen; in einigen Fällen wird dabei in Kauf genommen, daß die Vollständigkeit des Verfahrens verlorengeht (d.h., es kann nicht mehr garantiert werden, daß der Algorithmus eine lösbare Probleminstanz auch in jedem Fall als solche erkennt). Heuristische Verfahren werden in der Regel auf experimentellem Wege untersucht und besitzen oft hohe Relevanz für praktische Anwendungen.

2.2 Davis-Putnam-Verfahren

Der Davis-Putnam-Algorithmus (DP; siehe [DP60]) ist eines der ältesten Verfahren zur Lösung des Erfüllbarkeitsproblems. In der ursprünglichen Formulierung basiert das Verfahren auf sukzessiver Elimination der Variablen durch Resolution über den Klauseln einer Formel. Dabei kann es im ungünstigsten Fall bei Ausführung eines einzelnen Resolutionsschrittes zu einer quadratischen Aufblähung der Klauselanzahl kommen, wodurch für Implementierungen dieses Verfahrens schnell Speicherplatzprobleme auftreten (vergl. [Gu95]).

Der DPL-Algorithmus (Loveland-DP; siehe [DLL62, Lov78]) vermeidet dieses Problem, indem die sukzessive Elimination der Variablen durch Belegung jeweils einer Variablen mit einem Wert erreicht wird. Auf diese Weise wird ein vollständiges Suchverfahren auf dem Raum aller Variablenbelegungen definiert; wird hierbei eine Tiefensuche realisiert (Backtracking-Verfahren), so kann der exponentielle Speicherbedarf des ursprünglichen DP-Verfahrens vermieden werden. In dieser Form kann der DP-Algorithmus auf den Ansatz der dynamischen Programmierung (s.o.) zurückgeführt werden, da in jedem Schritt zwei neue SAT-Instanzen entstehen.

Ein wichtiger Freiheitsgrad dieser Verfahren ist die Reihenfolge, in der die Variablen eliminiert werden. Hierbei können verschiedene Heuristiken zum Einsatz kommen, unter den gebräuchlichen finden sich (vergl. [Gu95]):

- die Einerklauselregel (Variablen, die in einer Klausel der Länge 1 auftreten, werden bevorzugt);
- die „pure literal“ Regel (Variablen, die ausschließlich in negierter bzw. nichtnegierter Form in der Formel auftauchen, werden bevorzugt);
- sowie die Regel der kürzesten Klausel (Variablen, die in einer Klausel kürzester Länge auftauchen, werden bevorzugt).

Die Reihenfolge der Variablenauswahl beeinflußt die Performanz von DP-Algorithmen entscheidend; so konnte durch den Einsatz neuer Heuristiken an diesem Punkt in jüngerer Vergangenheit die Leistungsfähigkeit von DP-Verfahren erheblich gesteigert werden. Als Beispiel hierfür sei der 1993 vorgestellte ASAT-Algorithmus [DABC93] genannt, bei dem die Variablen nach der Anzahl ihrer Auftreten in der Formel eliminiert werden.

2.3 GSAT und Varianten

Der GSAT-Algorithmus, ein 1992 von Selman, Levesque und Mitchell vorgestelltes SAT-Verfahren (siehe [SLM92]), ist ein heuristisches, lokales Suchverfahren im Raum aller Variablenbelegungen einer gegebene CNF-Formel. Ausgehend von einer zufällig gewählten Belegung wird in jedem Schritt des Verfahrens jeweils die Belegung einer Variablen geändert (*Variablenflip* oder kurz *Flip*). Hierbei betrachtet man die Anzahl der durch die aktuelle Belegung B erfüllten Klauseln der gegebenen Formel F; in jedem Schritt wird nun eine Variable so ausgewählt, daß die Änderung ihrer Belegung den maximalen Zuwachs in der Anzahl der erfüllten Klauseln ergibt.

Formal kann dies beschrieben werden mithilfe einer Zielfunktion über der Menge aller Variablenbelegungen, wobei der Wert dieser Funktion durch die Anzahl der durch eine gegebene Belegung *unerfüllten* Klauseln definiert wird.

Definition 2.1 (C_{unsat}, N_{unsat})

Sei F eine CNF-Formel in Mengenschreibweise. Die Menge C_{unsat} der unter einer gegebenen Belegung B unerfüllten Klauseln von F ist definiert als:

$$C_{unsat}(B, F) := \{c_i \in F \mid Val_B(c_i) = \perp\}$$

Für die Anzahl der unter B unerfüllten Klauseln wird auch die Schreibweise

$$N_{unsat}(B, F) := |C_{unsat}(B, F)|$$

verwendet.

Anmerkung: $N_{unsat}(B, F)$ kann auch durch Rückgriff auf die penalty-logischen Begriffe aus Abschnitt 1.4 definiert werden, es gilt die Beziehung $N_{unsat}(B, F) = Pen_B(\Phi(F))$.

Die Modelle der Formel sind nun genau die Variablenbelegungen, für welche die Zielfunktion N_{unsat} den Wert 0 annimmt; das SAT-Problem läßt sich also mithilfe dieser Formalisierung als Problem der globalen Minimierung einer Zielfunktion formulieren. Insbesondere kann nun der GSAT-Algorithmus als ein Verfahren des steilsten Abstieges ("steepest descent") bezüglich der so definierten Zielfunktion charakterisiert werden.[1]

Die Auswahl der zu ändernden Variablen wird anhand einer Bewertung vorgenommen, die der durch die Belegungsänderung dieser Variablen bewirkten Wertänderung der Zielfunktion entspricht. Diese Bewertung ("score") ist folgendermaßen definiert:

Definition 2.2 (Bewertung von Variablen)

Seien F eine CNF-Formel, x eine in F auftretende Aussagenvariable sowie B und B' Variablenbelegungen, die sich lediglich in der Belegung der Variablen x unterscheiden. Die Bewertung der Variablen x unter der Belegung B *ist dann definiert als*

$$score_F(B, x) := N_{unsat}(B, F) - N_{unsat}(B', F)$$

In jedem Schritt des Verfahrens wird nun die Belegung einer Variable mit maximaler Bewertung geändert; erhalten mehrere Variablen die gleiche maximale Bewertung, so wird eine von diesen zufällig (gleichverteilt) ausgewählt. Durch diese probabilistische Auswahl und die zufällig bestimmte Ausgangsbelegung ist der GSAT-Algorithmus ein nichtdeterministisches Verfahren.

[1] Für derartige Verfahren findet oft auch die Bezeichnung "local hill-climbing" Verwendung; diese stammt aus der dualen Betrachtungsweise, bei der eine Zielfunktion zu maximieren ist (im Falle von GSAT die Anzahl der durch eine Belegung *erfüllten* Klauseln).

```
procedure GSAT(F)
  for i := 1 to maxTries do
    B := zufällig gewählte Belegung der Variablen aus F;
    for j := 1 to maxSteps do
      if Val_B(F) = ⊤ then return(B);
      else
        x := zufällig gewählte Variable aus F mit maximalem score_F(B, x);
        ändere Wert von x in B;
        end if;
      end for;
    end for;
  return (''kein Modell gefunden'');
  end GSAT;
```

Eine wichtige Eigenschaft des GSAT-Algorithmus besteht in seiner Unvollständigkeit: Angesetzt auf eine erfüllbare Formel kann im allgemeinen nicht garantiert werden, daß GSAT ein Modell der Formel findet. In diesem Sinne ist GSAT vollständigen Verfahren wie DP grundsätzlich unterlegen. Aufgrund seiner guten Performanz bei der Lösung bestimmter Klassen von SAT-Problemen hat das Verfahren dennoch weite Beachtung gefunden.

Ein weiteres grundsätzliches Problem lokaler Minimierungsalgorithmen wie GSAT besteht in der Tatsache, daß die Suche in lokalen Minima „steckenbleiben" kann. Um dieser Problematik zu begegnen, werden im Falle von GSAT in einem Programmlauf mehrere Versuche durchgeführt (maxTries viele, äußere Schleife), wobei jeder Versuch mit der Auswahl einer zufälligen Variablenbelegung beginnt und (falls bis dahin kein Modell gefunden wurde) nach einer festgelegten Anzahl (maxSteps) von Schritten abgebrochen wird. Auf diese Weise wird auch die maximale Laufzeit des Algorithmus nach oben begrenzt und somit die Terminierung gewährleistet. Die günstige Wahl des Parameters maxSteps ist im allgemeinen von der Größe der Formel abhängig und hat einen erheblichen Einfluß auf die Effizienz des Verfahrens (siehe auch Abschnitt 8.8).

Seit seiner Vorstellung im Jahre 1992 ist der GSAT-Algorithmus ausgiebig auf seine Eigenschaften und Verbesserungsmöglichkeiten untersucht worden; in diesem Zusammenhang wurden zahlreiche Varianten des Basisverfahrens entwickelt und getestet. Eine wesentliche Verbesserung der Performanz für verschiedene Formelklassen wird vor allem durch zwei Erweiterungen erreicht: "Random Walk" und adaptive Klauselgewichte. In der "Random Walk"- Variante (siehe [SKC93]) wird mit einer Wahrscheinlichkeit p_{walk} statt der oben beschriebenen Variablenauswahl zufällig (gleichverteilt) eine Variable aus einer unerfüllten Klausel ausgewählt und neu belegt. Auf diese Weise wird die Erfüllung der entsprechenden Klausel durch die neue Belegung erzwungen. Die "Random Walk"-Strategie bewirkt so ein Abweichen von der Methode des steilsten Abstieges und erleichtert das "Entkommen" aus lokalen Minima der Zielfunktion.

In der zweiten Variante (siehe [SK93]) wird jeder Klausel der Formel ein Gewicht zugeordnet, welches zu Beginn des Algorithmus mit 1 initialisiert wird. Am Ende jedes Versuchs werden nun die Gewichte der zu diesem Zeitpunkt unerfüllten Klauseln um einen festen Betrag inkrementiert. Bei der Variablenbewertung wird nun jede unerfüllte Klausel entsprechend ihres Gewichtes berücksichtigt, sodaß beispielsweise eine Klausel mit Gewicht 3 nun 3-fach zählt:

Definition 2.3 (Variablenbewertung mit Klauselgewichten)

Seien F eine CNF-Formel in Mengenschreibweise und $W : F \mapsto \mathbb{N}$ eine Bewertungsfunktion, welche jeder Klausel c_i von F einen Gewichtswert $W(c_i)$ zuordnet. Das Gewicht einer Belegung *B wird dann definiert als*

$$W(B, F) := \sum \{W(c_i) \mid c_i \in C_{unsat}(B, F)\}$$

Seien x eine in F auftretende Aussagenvariable sowie B und B' Variablenbelegungen, die sich lediglich in der Belegung der Variablen x unterscheiden. Die gewichtete Bewertung der Variablen x unter der Belegung B *ist dann definiert als*

$$score_F(B, x) := W(B, F) - W(B', F)$$

Anmerkung: Die für den ursprünglichen GSAT-Algorithmus verwendete (ungewichtete) Bewertungsfunktion kann als Spezialfall der hier definierten gewichteten Bewertungsfunktion aufgefaßt werden, wobei allen Klauseln das Gewicht 1 zugeordnet wird. Weiterhin kann ein Zusammenhang mit den in Abschnitt 1.4 definierten penalty-logischen Begriffen hergestellt werden: Wird F zusammen mit der Gewichtsfunktion $w(c_i)$ auf den Klauseln als penalty-logische Formel $\Phi(F, w)$ interpretiert, so gilt $W(B, F) = W(B, \Phi(F, w))$. Der Zusammenhang zwischen adaptiven Klauselgewichten und Penalty-Logik wird nochmals in Abschnitt 3.3.3 aufgegriffen.

Durch diesen Mechanismus der adaptiven Klauselgewichte wird erreicht, daß die Erfüllung von Klauseln, die am Ende eines Versuchs häufig unerfüllt waren, zukünftig eine höhere Priorität erhält. Diese Vorgehensweise bringt vor allem für strukturierte Probleme erhebliche Vorteile und läßt sich gut mit der "Random Walk"-Strategie kombinieren.

2.4 Simulated Annealing

"Simulated Annealing" (SA; siehe [KGV82]) ist ein physikalisch motiviertes Verfahren zur probabilistischen Optimierung einer Zielfunktion (in diesem Zusammenhang auch als Energiefunktion E bezeichnet). Aufgrund des im letzten Abschnitts gezeigten Zusammenhangs zwischen globaler Optimierung und dem SAT-Problem kann SA prinzipiell auch zum Finden von Modellen aussagenlogischer Formeln benutzt werden (vergl. [Spe93, BAH+94a]). Dabei realisiert SA ebenso wie GSAT ein heuristisch gesteuertes, lokales Suchverfahren.

Das allgemeine Verfahren basiert auf einer Nachbarschaftsrelation auf der Menge der Variablenbelegungen; die so definierten Paare benachbarter Belegungen kennzeichnen die in den einzelnen Schritten des Verfahrens erlaubten Übergänge. Im Falle der Anwendung auf das SAT-Problem werden üblicherweise genau die Übergänge zwischen Belegungen mit Hamming-Distanz 1 zugelassen (diese Übergänge entsprechen den *Flips* in GSAT, d.h. der Wertänderung genau einer Variablen).

Der SASAT-Algorithmus beginnt, wie GSAT, mit einer zufällig gewählten Variablenbelegung und wählt in jedem Schritt – ausgehend von der aktuellen Belegung B – einen erlaubten Übergang $B \to B'$ aus. Dieser Übergang wird nun mit einer gewissen Wahrscheinlichkeit $P(\Delta E, T)$ vorgenommen; diese ist abhängig von der durch den Übergang verursachten Wertänderung ΔE in der Zielfunktion (diese Änderung entspricht hier dem Betrag nach der Bewertung $score_F(B, x)$, vergl. Abschnitt 2.3) sowie einem weiteren Parameter T, der üblicherweise als Temperatur bezeichnet wird. Es werden unter anderem folgende Wahrscheinlichkeitsverteilungen verwandt:

$$P(\Delta E, T) = 1/(1 + \exp(\Delta E/T)) \qquad \text{(Glauberverteilung, siehe [Gla63])}$$

$$P(\Delta E, T) = \begin{cases} 1 & : \ \Delta E \leq 0 \\ \exp(-\Delta E/T) & : \ \text{sonst} \end{cases} \qquad \text{(Metropolisverteilung, siehe [MRR}^{+}\text{53])}$$

Um zu garantieren, daß der Algorithmus gegen ein lokales Minimum konvergiert, muß die Verteilung der Übergangswahrscheinlichkeiten $P(B \to B', T)$ folgender Bedingung genügen:

$$\frac{P(B \to B', T)}{P(B' \to B, T)} = \exp(-\frac{E(B') - E(B)}{T})$$

Der Temperaturparameter T wird im allgemeinen im Verlauf des Verfahrens variiert. Üblicherweise wird zu Beginn eine relativ hohe Temperatur gewählt, welche dann im Verlauf des Verfahrens schrittweise gesenkt wird, bis eine vorgegebene minimale Temperatur (oft 0) erreicht ist. Das Verhalten des Algorithmus ist stark von der Temperatur abhängig: je höher die Temperatur ist, desto größer ist die Wahrscheinlichkeit, daß Übergänge akzeptiert werden, welche eine Wertzunahme der Zielfunktion (also eine „Verschlechterung“) bewirken. Somit kann der Algorithmus bei hohen Temperaturen leichter aus lokalen Minima entkommen.

Der Temperaturverlauf wird in Form einer diskreten (meist monoton fallenden) Funktion über der Anzahl s der Schritte des Verfahrens angegeben (“annealing schedule”); in diesem Zusammenhang wurden die verschiedensten Funktionen eingesetzt, einige der gebräuchlichsten sind im folgenden zusammengestellt:

(1)	$T(s+1) = T(s) - \Delta T$	(arithmetische Abkühlung)
(2)	$T(s+1) = T(s) - \Delta T/s$	
(3)	$T(s+1) = T(1)/\exp(s \cdot \Delta T)$	(geometrische Abkühlung)

Wie im Fall von GSAT kann das SA-Verfahren derart realisiert werden, daß mehrere Versuche durchgeführt werden, wobei jeder Versuch mit einer zufällig gewählten Variablenbelegung beginnt und spätestens nach einer vorgegebenen Anzahl maxSteps von Schritten abgebrochen wird. Der SASAT-Algorithmus läßt sich damit insgesamt wie folgt formulieren:

```
procedure SASAT(F)
  for i := 1 to maxTries do
    B := zufällig gewählte Belegung der Variablen aus F;
    for j := 1 to maxSteps do
      if Val_B(F) = ⊤ then return(B);
      else
        repeat
          accept := false;
          x := zufällig gewählte Variable aus F;
          ΔE(x) := −score_F(B, x);
          accept := true mit Wahrscheinlichkeit P(ΔE(x), T(j));
        until accept = true;
        ändere Wert von x in B;
        end if;
      end for;
    end for;
  return (``kein Modell gefunden'');
  end SASAT;
```

Eine bemerkenswerte Eigenschaft von SA-Verfahren besteht darin, daß unter bestimmten Voraussetzung (u.a. Temperaturabsenkung in infinitesimal kleinen Schritten gegen 0) die Konvergenz gegen ein globales Minimum garantiert werden kann (siehe [GG84]). Somit kann mithilfe des SASAT-Algorithmus ein theoretisch vollständiges SAT-Verfahren konstruiert werden. Für die praktische Anwendung ist dies jedoch nur von geringer Bedeutung, da die für die Vollständigkeit erforderliche infinitesimale Abkühlung aufgrund der diskreten Natur des Verfahrens nur endlich approximiert werden kann, wodurch die Vollständigkeitseigenschaft verloren geht.

Ein generelles Problem beim Einsatz von SA-Algorithmen ist die geeignete Festlegung der Parameter. Insbesondere die Wahl des Temperaturverlaufs hat erhebliche Auswirkungen auf die Performanz des Algorithmus. Hier sind zwei Faktoren gegeneinander abzuwägen: Je langsamer die Temperaturabsenkung erfolgt, desto höher ist die Wahrscheinlichkeit, eine Lösung zu finden; allerdings bedingt eine langsame Abkühlung auch längere Laufzeiten. Für die Performanz des SASAT-Algorithmus hat sich der Einsatz von Temperaturverlauf (2) unter Verwendung der Metropolisverteilung für die Übergangswahrscheinlichkeiten in experimentellen Untersuchungen als besonders günstig erwiesen (siehe [BAH+94b]).

Das SASAT-Verfahren kann relativ leicht mithilfe künstlicher neuronaler Netze realisiert werden (Boltzmann-Maschine, vergl. [Pin92]). Die Parallelität dieses Berechnungsmodells kann allerdings nur begrenzt genutzt werden, da die theoretischen Konvergenzresultate, die sicherstellen, daß zumindest lokale Minima der Zielfunktion immer erreicht werden, nur für den Fall eines sequentiellen Ablaufes bekannt sind. Dennoch konnten für eine spezielle parallele Variante des Verfahrens im Rahmen experimenteller Untersuchungen vielversprechende Resultate bezüglich Konvergenz- und Laufzeitverhalten erreicht werden (siehe [BL93]).

2.5 Weitere Verfahren

Neben den dargestellten Verfahren wurde eine Vielzahl anderer Algorithmen zur Lösung des SAT-Problems vorgeschlagen und untersucht. In diesem Abschnitt wird ein kurzer Überblick über einige der neueren Ansätze gegeben; für eine ausführlichere und systematische Darstellung sei der interessierte Leser auf [Gu95] verwiesen.

Neben den bislang erwähnten GSAT-Varianten wurde eine große Anzahl von Modifikationen dieses Verfahrens untersucht (vergl. Kapitel 3); ein besonders interessanter Ansatz ist hierbei der Versuch, die spezifischen Stärken vollständiger, systematischer Suchverfahren (wie DPL) mit denen von unvollständigen lokalen Suchverfahren (wie GSAT) zu verbinden. Ein solches Verfahren findet sich in [GM94]; dieses basiert auf einem *"dynamic backtracking"*-Ansatz, einem systematischen Suchverfahren, welches auf der Speicherung von Informationen über den bereits durchsuchten Teil des Suchraums basiert. Experimentelle Untersuchungen auf einer speziellen Klasse schwieriger strukturierter SAT-Probleme deuten darauf hin, daß der vorgeschlagene Algorithmus für erfüllbare Formeln eine vergleichbare Performanz wie für GSAT aufweist und auf unerfüllbaren Formeln eine effiziente Implementierung des DP-Verfahrens deutlich übertrifft.

Gegenstand etlicher Untersuchungen zu lokalen Suchverfahren sind Mechanismen zur Behandlung oder Vermeidung lokaler Optima in der Zielfunktion. Neben den bereits erwähnten GSAT-Varianten sei in diesem Zusammenhang die in [Mor93] beschriebene *Breakout-Methode* erwähnt. Diese weist große Ähnlichkeit zu GSAT mit adaptiven Klauselgewichten auf, verwendet aber keinen maxSteps-Parameter und zeigt daher ein leicht verändertes Laufzeitverhalten.

Auf einem völlig anderen algorithmischen Ansatz basiert die in [HD94] vorgestellte *MASK-Methode*. Es handelt sich dabei um ein populations-basiertes, evolutionäres Suchverfahren zur Lösung des SAT-Problems, welches auf der Evaluierung und dynamischen Modifikation (*Evolution*) einer Menge von partiellen Variablenbelegungen (*Population*) beruht. Der MASK-Algorithmus ist ein unvollständiges SAT-Verfahren, welches leicht parallelisierbar ist und zeigt auf einer Reihe von Klassen strukturierter SAT-Probleme ein verhältnismäßig gutes Performanzverhalten.

Während die MASK-Methode auf einem biologisch inspirierten algorithmischen Ansatz beruht (und starke Analogie zur Familie der sogenannten *genetischen Algorithmen* besitzt), ist in jüngster Vergangenheit auch die duale Vorgehensweise, nämlich die Realisierung eines konventionellen SAT-Algorithmus mit molekularbiologischen Methoden, verstärkt untersucht worden. Auslöser dieser Ende 1994 einsetzenden Entwicklung auf dem Gebiet des biomolekularen Rechnens war eine Arbeit von L.M.Adleman (siehe [Adl94]), in der die Lösung einer Instanz des $\mathcal{NP}$-vollständigen Hamilton-Pfad-Problems in gerichteten Graphen mithilfe molekularbiologischer Methoden auf massiv parallele Weise beschrieben wurde. In der Folge wurde die Übertragbarkeit dieses Ansatzes auf die Lösung anderer $\mathcal{NP}$-vollständiger Probleme, unter anderem auf das SAT-Problem, untersucht (siehe [Lip94, Lip95]). Der dort vorgeschlagene unvollständige *BioSAT-Algorithmus* basiert auf einer hochgradig parallelen Suche im Raum der Belegungen, welcher explizit in Form von DNA-Molekülen (vergl. Abschnitt 9.1) repräsentiert ist. Dieses Verfahren kann unter Verwendung molekularbiologischer Standardverfahren realisiert werden und dabei Parallelitätsgrade von bis zu 10^{14} erreichen.

Da das biomolekulare Rechnen ein sehr junges und bislang wenig untersuchtes Grenzgebiet zwischen Informatik und Molekularbiologie darstellt, ist davon auszugehen, daß viele der derzeitigen Einschränkungen und Probleme in der praktischen Anwendung durch die Weiterentwicklung der verwendeten Methoden und Verfahren überwunden werden können (vergl. [Adl95]). Gerade bei der Lösung $\mathcal{NP}$-harter Probleme könnte daher dieser Ansatz in der Zukunft eine wichtige Bedeutung erlangen.

Kapitel 3

Algorithmen der GenSAT-Familie

Vor allem vor dem Hintergrund vergleichender Untersuchungen algorithmischer Verfahren erweist es sich oft als nützlich, einen allgemeinen Rahmen zu definieren, von welchem sich die zu untersuchenden spezifischen Verfahren als Instanzen ableiten lassen. Anhand des gut untersuchten GSAT-Verfahrens (siehe Abschnitt 2.3) läßt sich diese Vorgehensweise gut illustrieren. Aufgrund seiner Leistungsfähigkeit wurden viele Varianten des ursprünglichen GSAT-Algorithmus untersucht, um einerseits die für die Performanz wichtigen Konzepte des Verfahrens zu isolieren und andererseits Verbesserungsmöglichkeiten des ursprünglichen Algorithmus zu untersuchen. Diese Varianten lassen sich mithilfe des GenSAT-Rahmens auf einheitliche und übersichtliche Weise beschreiben, wodurch die Charakterisierung ihrer Eigenschaften und die Abgrenzung gegenüber ähnlichen Verfahren erleichtert wird. [1]

Aber auch für den Vergleich scheinbar grundsätzlich verschiedener SAT-Verfahren kann der GenSAT-Rahmen verwandt werden, wie in diesem Kapitel am Beispiel von GSAT und einem auf dem Simulated-Annealing-Konzept basierenden Algorithmus demonstriert wird. [2]

Am Ende des Kapitels wird gezeigt, wie bei der Implementierung von GenSAT-Verfahren unter Ausnutzung der Problemstruktur die Laufzeit in bestimmten Fällen drastisch reduziert werden kann.

3.1 Der GenSAT-Rahmen

Der GenSAT-Rahmenalgorithmus erhält als Eingabe eine aussagenlogische Formel F in CNF. Weitere Parameter sind maxTries und maxSteps, welche die maximale Anzahl von Versuchen bzw. Schritten pro Versuch[3] festlegen. Der Rahmenalgorithmus kann folgender-

[1] Diese Vorgehensweise geht auf [GW93] zurück; in diesem Papier wurden auch die meisten der hier beschriebenen Varianten untersucht.

[2] Zum Vergleich von GSAT und Simulated-Annealing gibt es verschiedene Untersuchungen; die hier eingesetzte Methode wurde in [BAH$^+$94a] beschrieben.

[3] Als *Versuch* wird in diesem Zusammenhang ein Durchlauf der äußeren `for`-Schleife bezeichnet.

weise notiert werden:

```
procedure GenSAT(F)
  for i := 1 to maxTries do
    B := init(F);
    for j := 1 to maxSteps do
      if Val_B(F) = ⊤ then return(B);
      else
        B' := select(F, B, j);
        B := B';
        end if;
      end for;
    end for;
  return (''kein Modell gefunden'');
  end GenSAT;
```

Der angegebene Rahmenalgorithmus eignet sich zur einheitlichen Beschreibung von SAT-Verfahren, die auf dem Prinzip einer lokalen Suche im Raum der möglichen Variablenbelegungen einer Formel basieren. Zur vollständigen Beschreibung einer konkreten Instanz müssen die Prozeduren init und select spezifiziert werden. Dabei erzeugt init zu Beginn jedes Versuchs eine Ausgangsbelegung der Variablen. Solange keine erfüllende Belegung erreicht ist, liefert die Prozedur select in jedem Schritt zur aktuellen Variablenbelegung B eine Folgebelegung B'. Der Übergang von der aktuellen zur Folgebelegung wird durch Änderung von Variablenwerten realisiert. Die Wertänderung einer Variablen wird dabei als *Flip* bezeichnet; werden in einem Schritt mehrere Variablenwerte geändert, spricht man von einem *Multiflip*. Ist nach einer Anzahl maxSteps solcher Schritter kein Modell gefunden worden, wird ein neuer Versuch gestartet. Nach maximal maxTries Versuchen terminiert das Verfahren.

Das ursprüngliche GSAT-Verfahren kann als Instanz dieses Rahmenalgorithmus formuliert werden. Dabei wählt init eine zufällige Variablenbelegung, indem jeder Variablen mit gleicher Wahrscheinlichkeit der Wert $\top$ oder $\bot$ zugeordnet wird. Die Prozedur select liefert prinzipiell nur Belegungen, die mithilfe eines einzigen Flips erreichbar sind (d.h. die Hamming-Distanz zwischen zwei aufeinanderfolgenden Belegungen beträgt immer genau 1). Dabei wird zunächst die Menge derjenigen Variablen berechnet, deren Wertänderung eine maximale Zunahme der Anzahl der erfüllten Klauseln bewirkt. Auf diese Weise wird bezüglich der Bewertungsfunktion ein Verfahren des steilsten Aufstiegs realisiert. Anschließend wird zufällig (gleichverteilt) ein Element dieser Menge ausgewählt. Schließlich gibt select die Belegung zurück, die durch einen Flip der so gewählten Variable entsteht. Formal kann diese Instanziierung des GenSAT-Rahmens folgendermaßen angegeben werden:

```
procedure init(F)
  for all x ∈ Var(F) do
    with probability 0.5 do
```

```
      B(x) := ⊤;
    else
      B(x) := ⊥;
      end with;
    end for;
  return (B);
  end init;
```

```
procedure select_g(F, B, j)
  maxScore := max{score_F(B, x) | x ∈ Var(F)};
  possFlips := {x ∈ Var(F) | score_F(B, x) = maxScore};
  x := rndElement(possFlips);
  B' := B;
  B'(x) := ¬B'(x);
  return (B');
  end select_g;
```

3.2 init-Varianten

Eine naheliegende Variationsmöglichkeit des Basisverfahrens besteht darin, bereits durch die init-Funktion einen möglichst günstigen Ausgangspunkt für die lokale Suche berechnen zu lassen. Solche Varianten erscheinen deshalb attraktiv, weil der durch sie verursachte zusätzliche Aufwand nur einmal in jedem Versuch anfällt; dieser könnte sich somit leicht amortisieren.

Für den ursprünglichen GSAT-Algorithmus wurde beispielsweise vorgeschlagen, auf einer zufällig ausgewählten Belegung zu Beginn eines jeden Versuchs zunächst ein Simulated Annealing durchzuführen. Eine alternative Möglichkeit basiert auf der Idee, die jeweils besten Belegungen aus vorhergegangenen Versuchen für die Berechung des Ausgangspunkts für den nächsten Versuch zu verwenden. Dabei wird für jede Variable das boolesche Mittel ihrer Werte unter den besten bislang erreichten Belegungen gebildet; diese Methode wird auch als "averaging in" bezeichnet (siehe [SK93]). Experimentelle Untersuchungen an Random-3-SAT-Problemen[4] zeigten jedoch, daß diese Varianten keine deutlichen Vorteile für die Performanz des Verfahrens bringen (siehe [GW93]). Insgesamt scheint die Auswahl der Ausgangsbelegung generell nur wenig Einfluß auf das Verhalten von GenSAT-Verfahren auszuüben.

[4]Dabei handelt es sich um zufällig ausgewählte Elemente der Menge aller 3-CNF-Formeln (siehe auch Abschnitt 4.1.1).

3.3 select-Varianten

Für die select-Funktion wurde ebenfalls eine Vielzahl veschiedener Varianten untersucht.

3.3.1 Indifferente Varianten

Ein interessanter Ansatz besteht darin, die lokale Suche indifferenter zu gestalten. Konkret kann dies mithilfe einer select-Funktion realisiert werden, welche aus der Menge aller direkt benachbarten Belegungen statt eines Elements mit maximaler Bewertung eine Variablenbelegung mit einer Bewertung ≥ 0 auswählt. Existieren keine solchen Belegungen, wird aus der Menge der direkt benachbarten Variablenbelegungen ein Element zufällig ausgewählt. Gegenüber dem ursprünglichen GSAT-Verfahren wird in dieser Variante lediglich gewährleistet, daß keine verschlechternden Schritte ausgeführt werden, solange noch günstigere Alternativen bestehen. Dieses als ISAT (für "indifferent SAT"; siehe [GW93]) bezeichnete Verfahren wird durch folgende select-Funktion charakterisiert:

```
procedure select_i(F, B, j)
  possFlips := {x ∈ Var(F) | score_F(B, x) ≥ 0};
  if possFlips = ∅ then
    possFlips := Var(F);
  x := rndElement(possFlips);
  B' := B;
  B'(x) := ¬B'(x);
  return (B');
  end select_i;
```

Eine etwas weniger indifferente Variante dieses Verfahrens erhält man, indem die Bedingung „$\geq$ 0“ durch „$>$ 0“ ersetzt wird (vergl.[HHSW94]). In diesem Fall wählt select ein Element aus der Menge aller direkt benachbarten Variablenbelegungen, welche die Anzahl der erfüllten Klauseln echt erhöhen. Damit werden, wann immer dies möglich ist, verbessernde Schritte ausgeführt. Diese Variante wird als I_2SAT bezeichnet und durch die folgende select-Funktion realisiert:

```
procedure select_i2(F, B, j)
  possFlips := {x ∈ Var(F) | scoreF(B, x) > 0};
  if possFlips = ∅ then
    possFlips := Var(F);
  x := rndElement(possFlips);
  B' := B;
  B'(x) := ¬B'(x);
  return (B');
  end select_i2;
```

Während die select-Prozedur des ursprünglichen GSAT-Algorithmus eine Nachbarbelegung mit global optimaler Bewertung liefert, benötigen die vorgestellten indifferenten Varianten lediglich lokale Informationen über die Nachbarbelegungen und sind damit für eine Parallelisierung des Verfahrens besser geeignet. Wie aus der Untersuchung von Gent & Walsh (siehe [GW93]) hervorgeht, liegt allerdings die Performanz des ISAT-Verfahrens signifikant unterhalb der von GSAT; erst durch die Kombination mit anderen select-Strategien kann dieser Nachteil behoben werden.

Der I_2SAT-Algorithmus zeigt jedoch auf Random-3-CNF-Formeln ein ähnliches Performanzverhalten wie GSAT (siehe [HHSW94]). Folglich scheint es wichtig für die Performanz von GenSAT-Verfahren zu sein, daß durch einen Schritt eine Verbesserung der Bewertung erreicht wird, wann immer dies möglich ist (I_2SAT). Nicht entscheidend ist es jedoch, die jeweils maximal mögliche Verbesserung zu realisieren (GSAT).

3.3.2 Zufallsschritte

Die bislang vorgestellten select-Funktionen realisieren die Auswahl einer Belegung abhängig von ihrer Bewertung. Ein weitere Variationsmöglichkeit besteht darin, zusätzlich eine von der Bewertung unabhängige, zufällige Auswahl aus der Menge der direkten Nachbarbelegungen zuzulassen. Durch diese verschiedenartigen Selektionsmechanismen sind prinzipiell zwei Arten von Schritten möglich: konventionelle (Hillclimb-)Schritte und *Zufallsschritte*. Die Entscheidung, welche Art von Selektionsmechanismus in einem konkreten Schritt zum Einsatz kommt, wird über eine Zufallsvariable realisiert, wobei die Wahrscheinlichkeit für einen Zufallsschritt über einen zusätzlichen Parameter p_{rnd} eingestellt werden kann.

Erfolgt im Falle eines Zufallsschrittes die Auswahl der Folgebelegung zufällig aus der Menge *aller* direkten Nachbarbelegungen, so spricht man von der "Random Noise"-Variante (vergl. [SKC94]). Alternativ kann man nur solche Nachbarbelegungen in Betracht ziehen, die durch Änderung einer Variablen, die wenigstens in *einer* aktuell unerfüllten Klausel auftritt, erreichbar sind; diese Variante geht auf Selman & Kautz zurück und wird als "Random Walk" bezeichnet (vergl. [SKC94]). Im Gegensatz zu "Random Noise" wird hierbei sichergestellt, daß mindestens eine vorher unerfüllte Klausel durch den gewählten "Random Walk"-Schritt erfüllt wird (vergl. Abschnitt 2.3).

Die Selektions-Funktionen für "Random Noise" (select_{rn}) und "Random Walk" (select_{rw}) lassen sich folgendermaßen formulieren:

```
procedure select_rn(F, B, j)
  with probability p_rnd do
    possFlips := {x ∈ Var(F)};
    x := rndElement(possFlips);
    B' := B;
    B'(x) := ¬B'(x);
    return (B');
```

```
  else
    return( selectg(F, B, j) );
    end with;
  end selectrn;

procedure selectrw(F, B, j)
  with probability prnd do
    Cunsat := {c ∈ F | ValB(c) = ⊥};
    possFlips := {x ∈ Var(F) | x ∈ c, c ∈ Cunsat};
    x := rndElement(possFlips);
    B' := B;
    B'(x) := ¬B'(x);
    return (B');
  else
    return( selectg(F, B, j) );
    end with;
  end selectrw;
```

Während die "Random-Noise"-Variante keine Verbesserung gegenüber dem ursprünglichen GSAT-Algorithmus darstellt, kann die Performanz des Verfahrens durch "Random Walk" signifikant verbessert werden. Für schwierige Random-3-SAT-Probleme wurde ermittelt, daß sich eine optimale Performanz für p_{rnd} zwischen 0.5 und 0.6 ergibt (vergl.[SKC93]); bei dieser Einstellung werden Hillclimb- und Walk-Schritte mit gleicher Wahrscheinlichkeit ausgeführt.

3.3.3 Varianten mit Gedächtnis

Die bislang vorgestellten Varianten der select-Funktion waren gedächtnislos in dem Sinne, daß bei der Auswahl der Folgebelegung keine Informationen über vergangene Zustände des Verfahrens (d.h. Belegungen und deren Bewertungen) verwendet wurden.

Eine einfach zu realisierende select-Variante mit Gedächtnis beruht auf der Idee, bei der Auswahl zwischen Variablen-Flips mit maximaler Bewertung das *Alter* der jeweiligen Variablen zu berücksichtigen, d.h. die Anzahl der Schritte des Verfahrens, in denen die entsprechende Variable nicht mehr verändert wurde. Ausgewählt wird dann die Folgebelegung, die durch die Umbelegung der ältesten Variable erreicht wird; zu Beginn jedes Versuchs wird das Alter aller Variablen zurückgesetzt. Diese als "history" (Historie) bezeichnete Variante (siehe [GW93]) begünstigt eine *faire Selektion* der Folgebelegung in dem Sinne, daß beispielsweise eine gerade geänderte Variable im folgenden Schritt nicht erneut geändert werden kann, es sei denn, es gibt keine Alternative (d.h. keine andere Folgebelegung mit gleicher Bewertung). Diese Variante der select-Funktion bewirkt eine signifikante Verbesserung gegenüber dem ursprünglichen GSAT-Algorithmus; in Kombination mit "Random Walk" läßt sich dieser Effekt allerdings nicht beobachten (siehe [GW93],[BAH+94a]).

Eine weitere Technik, die oft als *Tabu-Suche* bezeichnet wird, beruht darauf, nach jedem Variablenübergang die entsprechende Variable für eine bestimmte Zeit zu sperren; damit wird sichergestellt, daß der Wert dieser Variablen erst nach einer gewissen Anzahl von Schritten wieder geändert werden darf. Auf diese Weise kann ein „Steckenbleiben“ des Verfahrens in Zyklen, welche auf Übergängen einer kleinen Gruppe von Variablen beruhen, verhindert werden. Andererseits kann diese Strategie auch das Auffinden von Lösungen erschweren; dies ist insbesondere zu befürchten, wenn die Tabu-Zeit, also die Anzahl der Schritte, für die eine Variable jeweils gesperrt wird, zu groß gewählt ist.

Auch die in Abschnitt 2.3 dargestellte Technik der adaptiven Klauselgewichte läßt sich unter die Rubrik der select-Varianten mit Gedächtnis fassen. Wie bereits erwähnt, ist diese Variante gut mit “Random Walk” kombinierbar; verschiedene Untersuchungen zeigen, daß die Technik der adaptiven Klauselgewichte vor allem bei besonders strukturierten Problemen Verbesserungen der Performanz bewirkt (als Beispiele werden vor allem Planungsprobleme — wie „Türme von Hanoi“ — genannt; siehe u.a. [SKC93]). Bemerkenswert ist, daß zwischen dieser Variante und der in Abschnitt 1.4 eingeführten Penalty-Logik ein enger Zusammenhang besteht: Aus der Sicht der Penalty-Logik gesehen, handelt es sich hierbei um ein Verfahren, welches durch Modifikation der Klauselgewichte im Rahmen eines Lernprozesses versucht, die gegebene Formel derart umzuformen, daß deren aussagenlogische Modelle durch das GSAT-Basisverfahren leichter gefunden werden können. Die Erhöhung der Gewichte von unerfüllten Klauseln entspricht dabei einer Elimination von P-Modellen für Schranken $P > 0$. Durch die Modifikation der Bewertungsfunktion, welche die Berücksichtigung von Klauselgewichten erlaubt (vergl. Abschnitt 1.4), erhält man ferner aus den betrachteten SAT-Algorithmen entsprechende Verfahren zur Lösung des MaxSAT-Problems (siehe auch Abschnitt 9.5).

3.4 Vollständige GenSAT-Algorithmen

Die bisher betrachteten Algorithmen der GenSAT-Familie sind sämtlichst unvollständige SAT-Verfahren, können also nicht dazu verwandt werden, die Unerfüllbarkeit einer gegebenen Formel zu beweisen. Die Ursache für diese Unvollständigkeit liegt im wesentlichen in der Tatsache, daß der lokale Suchprozeß sowohl bei der Initialisierung, als auch bei der Auswahl der einzelnen Schritte durch nichtdeterministische Entscheidungen gesteuert wird; dabei findet insbesondere keine systematische Durchmusterung des Suchraums statt.

Es stellt sich nun die Frage, inwieweit diese nichtdeterministischen Elemente entscheidenden Einfluß auf die Performanz der betrachteten lokalen Suchverfahren haben. Einen wichtigen Beitrag zur Beantwortung dieser Frage liefert eine Untersuchung von Gent & Walsh (siehe [GW93]). Dort werden GenSAT-Varianten mit deterministischen init- und select-Funktionen betrachtet und bezüglich ihrer Performanz auf Random-3-CNF-Formeln mit einigen der bereits angesprochenen, nichtdeterministischen GenSAT-Varianten verglichen. Dabei wurde festgestellt, daß der im ursprünglichen GSAT-Algorithmus vorhandene Nichtdeterminismus ohne signifikante Auswirkungen auf die Performanz des Verfahrens eliminiert werden kann.

Entscheidend ist es jedoch sicherzustellen, daß jegliche Auswahl eines Elementes aus einer Menge (ob deterministisch oder nicht) in dem Sinne einem „Fairness"-Kriterium genügt, daß — über eine große Anzahl einzelner Auswahlprozesse hinweg betrachtet — jedes Element der Menge mit gleicher Häufigkeit ausgwählt wird.

Konkret bedeutet dies für die Auswahl der Ausgangsbelegung eines Versuchs durch die init-Funktion, daß im Mittel jede einzelne Variable mit gleicher Häufigkeit mit den Werten $\top$ und $\perp$ belegt wird. Ebenso sollte die select-Funktion bei der Auswahl aus mehreren, identisch bewerteten Übergängen keinen Variablenübergang gegenüber einem anderen bevorzugen. Diese Forderung ist sehr plausibel, da jede Unausgewogenheit in der Behandlung von Belegungen einzelner Variablen dazu führt, daß für bestimmte Formeln Übergänge, die für das Erreichen einer Lösung essentiell sind, behindert werden. Unklar ist bislang, ob auch die durch die "Random-Walk"-Technik realisierbaren Verbesserungen der Performanz mithilfe fairer, deterministischer Entscheidungen erreichbar sind; es wird jedoch vermutet, daß dies der Fall ist (siehe [BAH$^+$94a]).

Eine unmittelbare Konsequenz dieser Resultate ist die Möglichkeit, GenSAT-Varianten zu formulieren, welche sich bezüglich ihrer Performanz mit den besten der beschriebenen nichtdeterministischen Varianten messen können, dabei jedoch vollständige Verfahren zur Lösung des SAT-Problems realisieren. Bei näherer Betrachtung zeigt sich jedoch, daß die so realisierte Vollständigkeit für die Praxis kaum relevant ist, da die Entscheidung, daß eine gegebene Formel unerfüllbar ist, prinzipiell eine exponentielle Laufzeit des entsprechenden GenSAT-Algorithmus erfordert. Im Gegensatz hierzu können beispielsweise Varianten des vollständigen Davis-Putnam-Verfahrens (vergl. Abschnitt 2.2) die Unerfüllbarkeit bestimmter SAT-Instanzen mit wesentlich geringerem Zeitaufwand feststellen.

3.5 SASAT als probabilistische GenSAT-Instanz

Bei den bislang dargestellten GenSAT-Instanzen handelte es sich jeweils um Varianten des GSAT-Algorithmus. Der GenSAT-Rahmen kann allerdings als Hilfsmittel zum Vergleich grundsätzlich verschiedener SAT-Verfahren verwendet werden. Der Rahmenalgorithmus gibt dabei die grundlegenden Komponenten eines lokalen Suchverfahrens vor, während die heuristische Steuerung der Suche durch die jeweilige Instantiierung der Funktionen init und select beschrieben wird.

Diese Vorgehensweise läßt sich gut am Beispiel des in Abschnitt 2.4 eingeführten SASAT, einem Simulated-Annealing-Verfahren, demonstrieren. Hierbei handelt es sich um ein Verfahren, welches sich in der Auswahl der Übergänge erheblich von den bislang betrachteten GenSAT-Varianten unterscheidet. Während dort im wesentlichen eine gleichverteilte Auswahl aus einer Reihe von Alternativen erfolgte, beruht die nichtdeterministische Auswahl der Übergänge im SASAT-Verfahren auf einer komplexeren Wahrscheinlichkeitsverteilung, bei der die Wahrscheinlichkeit für die Auswahl eines Übergangs abhängig von dessen Bewertung ist. Analog zu den indifferenten GenSAT-Varianten und im Unterschied zu GSAT handelt es sich dabei um eine lokale Entscheidung, d.h. für die Auswahl eines Übergangs wird lediglich

Information über diesen Übergang selbst (etwa in Form der Bewertung) benötigt.

Der SASAT-Algorithmus kann als GenSAT-Instanz formuliert werden, indem die gleiche init-Funktion wie für GSAT (vergl. Abschnitt 3.1) sowie folgende select-Funktion verwendet wird:

```
procedure select_SA(F, B, j)
  accept := false;
  repeat
    x := rndElement(Var(F));
    ΔE(x) := −score_F(B, x);
    with probability P(ΔE(x), T(j)) do
      accept := ⊤;
  until accept = ⊤;
  B' := B;
  B'(x) := ¬B'(x);
  return (B');
  end select_SA;
```

Eine interessante Eigenschaft dieser GenSAT-Instanz ist ihre theoretische Vollständigkeit (vergl. Abschnitt 2.4). Die hierzu erforderliche Absenkung des Temperaturparameters in infinitesimal kleinen Schritten ist jedoch auf digitalen Rechnern nicht realisierbar, weshalb diese Eigenschaft für die Praxis ebensowenig nutzbar ist, wie die Vollständigkeit der im letzten Abschnitt erwähnten deterministischen Varianten.

Wie bereits erwähnt, kann eine einheitliche Formulierung auf der Basis des GenSAT-Rahmens für einen Vergleich verschiedener SAT-Verfahren sehr hilfreich sein. Konkret sei hier der Fall einer vergleichenden Untersuchung zum Laufzeitverhalten von GSAT und SASAT genannt (vergl. [BAH+94b]). Die Grundlage solcher experimenteller Untersuchungen bilden in der Regel die gemessenen realen Laufzeiten (Angabe in CPU-Sekunden) und/oder die Anzahl der benötigten Elementaroperationen (in diesem Falle Variablen-Flips) für die Lösung bestimmter Testprobleme. Aus Gründen, die in Kapitel 4 ausführlich erörtert werden, ist die letztgenannte Meßmethode unbedingt der ersteren vorzuziehen. Der damit erforderliche Vergleich des Zeitaufwandes für die Auswahl der Folgebelegung läßt sich sehr viel leichter an der GenSAT-Formulierung der Algorithmen durchführen, wo dieses Problem durch eine Analyse der Zeitkomplexität für die jeweiligen select-Funktionen gelöst wird.

Für den tatsächlichen Vergleich des Laufzeitverhaltens von GSAT und SASAT muß allerdings eine wesentlich präzisere Formalisierung beider Algorithmen zugrundegelegt werden, da bestimmte Aspekte der Implementierung wesentlichen Einfluß auf das Laufzeitverhalten der Verfahren haben (dieser Punkt wird im folgenden Abschnitt näher betrachtet).

3.6 Effiziente Implementierung von GenSAT-Instanzen

Obwohl der GenSAT-Rahmen und seine Instanzen zunächst sehr einfach erscheinen, ist die Implementierung keineswegs trivial. Daher soll in diesem Abschnitt eine effiziente Implementation von GSAT (als GenSAT-Instanz) diskutiert werden, die von großer Bedeutung für die Performanz des Verfahrens ist und sich auf viele Varianten übertragen läßt.

Betrachten wir zunächst die naheliegende Implementierung der oben definierten select-Funktion für GSAT. Die für den Zeitaufwand entscheidenden Schritte sind die Bestimmung des Wertes maxScore und der Menge possFlips. Diese Schritte können kombiniert werden, indem maxScore zunächst auf einen Wert gesetzt wird, der sicherlich kleiner ist, als alle auftretenden Variablenbewertungen; bei einer CNF-Formel F über n Variablen mit m Klauseln kann hier der Wert $-(m+1)$ benutzt werden. Nun werden die Bewertungen für alle Variablen sukzessive berechnet; wird dabei eine Variable x mit Bewertung $score_F(B,x) \geq$ maxScore gefunden, so wird possFlips $:= \{x\}$ und maxScore $:= score_F(B,x)$ gesetzt; ist $score_F(B,x) =$ maxScore, so wird x zur Menge possFlips hinzugefügt. Auf diese Weise sind die Variablen possFlips und maxScore am Ende der Iteration korrekt gesetzt.

Der Zeitaufwand dieser select-Implementierung ist linear abhängig von der Variablenanzahl n und vom Zeitaufwand $Time(score_F)$ für die Berechnung einer Variablenbewertung $score_F(B,x)$ (wobei davon ausgegangen wird, daß dies die aufwendigste Operation in der Iteration über alle Variablen ist).

Eine Implementierung von $score_F(B,x)$, die sich an der Definition orientiert (siehe Abschnitt 2.3), bestimmt zunächst die Anzahl unter der aktuellen Belegung B unerfüllten Klauseln und dann die Anzahl der unter der um den Wert von x geänderten Belegung B' unerfüllten Klauseln. Ist die Länge der Klauseln in F von der Größenordnung $CL(n)$, so ergibt sich für die Berechnung des Wertes $N_{unsat}(B,F)$ ein Zeitaufwand von $\Theta(m \cdot CL(n))$; für k-CNF-Formeln entspricht dies $\Theta(m)$.

Für die select-Implementierung ergibt sich damit insgesamt ein Zeitaufwand von $\Theta(n \cdot m \cdot CL(n))$; dies entspricht im häufig untersuchten Fall der Random-k-CNF-Formeln mit konstantem Variablen/Klausel-Verhältnis einer Zeitkomplexität von $\Theta(n^2)$.

Diese Implementierung kann jedoch entscheidend verbessert werden; hierbei wird die Tatsache ausgenutzt, daß sich in der Regel nicht alle Variablenbewertungen in jedem Schritt des Algorithmus ändern. Somit bietet es sich an, diese Werte einmal zu Beginn eines Versuchs zu berechnen und zu speichern, um dann nach jedem Schritt nur die jeweils geänderten Bewertungen neu berechnen zu müssen. Diese als *"score-by-need"* bezeichnete Technik (die auch in der GSAT-Implementierung von Selman & Kautz realisiert wurde) kann den Zeitaufwand der select-Funktion drastisch reduzieren, wie im folgenden genauer gezeigt wird. Hierzu werden zunächst einige wichtige Begriffe formal definiert:

Definition 3.1 (Variablen- & Klausel-Abhängigkeit)

Zwei Variablen x_1, x_2 *einer CNF-Formel* F *heißen* abhängig voneinander, *wenn sie gemeinsam in einer Klausel vorkommen, d.h. eine Klausel* $c \in F$ *existiert, für welche gilt* $\{x_1, x_2\} \subseteq Var(c)$.

Ist x *eine Variable von* F*, so ist die* Menge der von x abhängigen Variablen *definiert als*

$$V_{dep}(x, F) := \{y \in Var(F) \mid y \text{ abhängig von } x\}$$

Eine Klausel $c \in F$ *heißt* abhängig von der Variablen x, *wenn* x *in ihr auftritt, d.h. falls* $x \in Var(c)$ *gilt. Die* Menge der von x abhängigen Klauseln *ist dann definiert als*

$$C_{dep}(x, F) := \{c \in F \mid c \text{ abhängig von } x\}$$

Um den Zeitaufwand für die Aktualisierung der Variablenbewertungen zu beurteilen, betrachten wir nun folgende Situation: Die Bewertung $score_F$ einer jeden Variablen unter der aktuellen Belegung ist in einem Feld gespeichert und mit konstantem Zeitaufwand zugreifbar. Die Anzahl u der unter der aktuellen Bewertung unerfüllten Klauseln ist ebenfalls bekannt. Nun wird von der select-Funktion eine Variable x selektiert und deren Belegung geändert. Durch diesen Übergang können lediglich von x abhängige Klauseln ihren Wert Val_B ändern; um die Anzahl der erfüllten Klauseln zu aktualisieren, sind also lediglich die Werte der Klauseln in $C_{dep}(x, F)$ neu zu berechnen.

Aufgrund der Definition der Variablenbewertung (vergl. Abschnitt 2.3) kehrt sich für die Bewertung von x lediglich das Vorzeichen um. Für jede beliebige andere Variablen $y \neq x$ gilt nun: Die Bewertung von y bleibt durch den Übergang unverändert, wenn y unabhängig von x ist, wenn also gilt:

$$y \notin V_{dep}(x, F)$$

Daraus folgt direkt, daß nach einem Übergang der Variablen x lediglich die Bewertungen der von x abhängigen Variablen $y \in V_{dep}(x, F)$ aktualisiert werden müssen. Ist $VD(n)$ eine obere Schranke für die Mächtigkeit der Mengen $V_{dep}(x, F)$ bei gegebener Formel F, so können folglich die durch den Übergang einer Variablen verursachten Bewertungsänderungen für sämtliche Variablen von F mit Zeitaufwand $O(VD(n) \cdot Time(score_F))$ berechnet werden. Nun müssen jedoch für die Berechnung der Bewertungsänderungen lediglich diejenigen Klauseln betrachtet werden, in denen die geänderte Variable x tatsächlich auftritt. Ist bei gegebener Formel F eine obere Schranke für die Mächtigkeit der Mengen $C_{dep}(x, F)$ durch $CD(n)$ gegeben, so ergibt sich folglich insgesamt für die Funktion select ein Zeitaufwand von $O(CD(n) \cdot VD(n) \cdot CL(n))$.

Anmerkung: Das hier angegebene Kriterium ist lediglich hinreichend; tatsächlich kann durch eine verfeinerte Analyse ein notwendiges Kriterium gefunden werden, mit dessen Hilfe der Aufwand für die Aktualisierung der Variablenbewertungen nochmals reduziert werden kann. Die zugrundeliegende

Idee basiert auf einer genaueren Betrachtung der Situationen, in denen eine erfüllte Klausel durch einen Variablenübergang unerfüllt werden kann (*sensitive Abhängigkeit* zwischen Klauseln und Variablen). Die auf diese Weise erreichbare effiziente Aktualisierung der Variablenbewertungen ist in der GSAT-Implementierung von Selman & Kautz realisiert.

Ist für eine gegebene Klasse von CNF-Formeln die Schranke $VD(n)$ (oder der Erwartungswert für die Mächtigkeit der Menge $V_{dep}(x, F)$) kleiner als die Anzahl n der Variablen in F, kann durch die "score-by-need"-Technik der Zeitaufwand für die select-Funktion echt reduziert werden. Ein extremes Beispiel hierfür bildet die häufig untersuchte Klasse der Random-k-CNF-Formeln mit konstantem Variablen/Klausel-Verhältnis r : Für eine solche Formel kann die erwartete Mächtigkeit von $V_{dep}(x, F)$ durch die Konstante $r \cdot k \cdot (k-1)$ nach oben abgeschätzt werden.[5] Mithilfe der "score-by-need"-Technik kann also in diesem Falle der erwartete Zeitaufwand für select, der ausschlaggebend für den Zeitaufwand des gesamten Verfahrens ist, von $\Theta(n^2)$ auf $\Theta(1)$ reduziert werden! (Eine analoge Vorgehensweise wurde in [BL93] für ein Simulated-Annealing-Verfahren realisiert; auf diese Weise konnte dort für eine bestimmte Problemklasse eine Beschleunigung um einen Faktor 100 erzielt werden.)

Bei dieser, bereits in der ursprünglichen GSAT-Implementierung von Selman und Kautz eingesetzten Technik handelt es sich keineswegs um einen bloßen „Implementierungstrick“ sondern vielmehr um eine Schlüsseltechnik, die für die Performanz von GSAT-ähnlichen Verfahren von entscheidender Bedeutung ist. Es muß deshalb besonders betont werden, daß der durch diese Technik bewirkte Effizienzgewinn stark abhängig von der Struktur der betrachteten Formeln ist. Dabei stellen die oben betrachteten Random-k-CNF-Formeln mit konstantem Variablen/Klausel-Verhältnis einen besonders günstigen Fall dar; andererseits gibt es Formelklassen (wie die in Kapitel 6 betrachteten SAT-kodierten HC-Probleme), für die auf diese Weise kein Effizienzgewinn erreicht werden kann.

Das der "score-by-need"-Technik zugrundeliegende allgemeine Prinzip läßt sich auf die Realisierung weiterer Aspekte von GenSAT-Verfahren übertragen. Insbesondere läßt sich die "Random-Walk"-Strategie besonders effizient realisieren, indem für jede Variable memoriert wird, ob sie unter der aktuellen Belegung in einer unerfüllten Klausel auftritt (diese Technik findet sich ebenfalls in der GSAT-Implementation von Selman & Kautz). Diese Information wird dann, ebenso wie die Variablenbewertungen, im Rahmen der init-Funktion berechnet und anschließend lediglich in den nötigen Fällen aktualisiert.

Prinzipiell ist zu betonen, daß der GenSAT-Rahmen nicht als Vorlage für die Implementierung der entsprechenden Algorithmen betrachtet werden sollte. Wie anhand der "score-by-need"-Technik demonstriert wurde, erfordern effiziente Implementierungen von GenSAT-Verfahren oft besondere Methoden, welche zusätzliche Informationen über die Struktur des Problems (d.h. der gegebenen Formel) ausnutzen.

[5] Dabei wird die erwartete Anzahl von Klauseln, in denen eine gegebene Variable x auftritt, durch $r \cdot k$ abgeschätzt; ferner teilt sich jedes Auftreten von x die betrachtete Klausel mit genau $k-1$ weiteren Variablen; diese Abschätzung ist allerdings ungenau, da sie nicht berücksichtigt, daß sich zwei Variablen unter Umständen mehrere Klauseln teilen.

Kapitel 4

Vergleich und Bewertung von SAT-Algorithmen

Angesichts der verschiedenen in den vorhergehenden Kapiteln vorgestellten Verfahren zur Lösung des SAT-Problems stellt sich die Frage, auf welche Weise die vorgeschlagenen Algorithmen zu beurteilen sind. Von besonderem Interesse sind dabei vergleichende Bewertungen verschiedener Verfahren. Aufgrund der Tatsache, daß theoretische Resultate zur Performanz von SAT-Verfahren in der Regel nicht erzielt werden können, ist man im wesentlichen auf die experimentelle Untersuchung der Verfahren angewiesen. Dabei werden Implementierungen der zu untersuchenden Verfahren in der Regel auf einen Satz von Testformeln angesetzt. In den einzelnen Versuchsläufen werden bestimmte Leistungsdaten empirisch ermittelt; diese bilden schließlich die Grundlage für die Bewertung des Verfahrens.

Die in diesem Kapitel dargestellten Überlegungen bilden die Grundlage für die methodische Vorgehensweise der im Rahmen dieser Arbeit durchgeführten experimentellen Untersuchungen an SAT-kodierten Hamilton-Kreis-Problemen (siehe Kapitel 6ff.). Somit können die dort erzielten Resultate auch als Beitrag zur Evaluierung von GenSAT-Verfahren an SAT-kodierten Anwendungsproblemen aufgefaßt werden.

4.1 Formelklassen für experimentelle Untersuchungen

Die experimentelle Untersuchung von SAT-Algorithmen geschieht im allgemeinen auf einem Satz von Testformeln, der Teilmenge einer bestimmten Formelklasse ist. Die Auswahl der Formelklasse und die Erzeugung der konkreten Probleminstanzen für den Testsatz sind dabei keinesfalls triviale Aufgaben und haben einen erheblichen Einfluß auf die Aussagekraft der experimentellen Resultate. Im Gegensatz zu vielen anderen Problembereichen der Informatik gibt es für die Evaluierung von SAT-Verfahren bislang keine standardisierten Testsätze ("benchmark-problems"), wodurch die Testsätze im allgemeinen jeweils vom Experimentator erzeugt werden. Dadurch sind Vergleiche zwischen verschiedenen Untersuchungen schwierig durchzuführen und oft wenig aussagekräftig.

4.1.1 Random k-CNF-Formeln

Die Teilklasse k-CNF der aussagenlogischen Formeln in konjunktiver Normalform wurde bereits in Abschnitt 1.1 defininiert. Solche Formeln können auf sehr einfache Weise zufällig erzeugt werden, indem für jede Klausel zufällig eine Menge von k Literalen gewählt wird. Die Menge der insgesamt verfügbaren Literale wird dabei durch die gegebene Anzahl n der Variablen bestimmt. Bei diesem Verfahren ist darauf zu achten, daß die erzeugten Klauseln keine komplementären oder mehrfachen Auftreten von Literalen enthalten (sonst genügt die Formel nicht der Definition der Klasse k-CNF). Ferner muß bei der Konstruktion der Klauseln die Auswahlwahrscheinlichkeit für alle Literale gleich sein, da ansonsten die Schwierigkeit der erzeugten Formeln unter Umständen erheblich verändert wird. Die auf diese Weise erzeugten Formeln werden als *Random-k-CNF-Formeln* bezeichnet.

Dieses Erzeugungsverfahren ist einfach zu implementieren und erlaubt die effiziente Erzeugung großer Formelsätze; die so gewonnenen Formelmengen weisen jedoch einige gravierende Probleme auf. Zum einen ist über die Erfüllbarkeit der erzeugten Formeln nichts bekannt. Für die Untersuchung unvollständiger SAT-Verfahren ist jedoch die Erfüllbarkeit der Testformeln eine notwendige Voraussetzung für eine sinnvolle Interpretation der Ergebnisse (da die Unerfüllbarkeit in diesem Fall vom Algorithmus nicht festgestellt werden kann). Aus einer Menge zufällig erzeugter k-CNF-Formeln müssen also für derartige Untersuchungen mithilfe eines (vollständigen) Entscheidungsverfahrens für SAT die erfüllbaren Formeln herausgefiltert werden. Diese Vorgehensweise führt in der Praxis zu großen Schwierigkeiten, da ein solcher Selektionsvorgang im allgemeinen wesentlich zeitaufwendiger ist, als die eigentlichen Experimente an den zu untersuchenden SAT-Verfahren.

Eine mögliche Lösung dieses Problems besteht darin, bereits bei der Erzeugung der Formeln deren Erfüllbarkeit zu gewährleisten. Dies kann im einfachsten Falle dadurch geschehen, daß zu Beginn des Erzeugungsprozesses eine zufällige Variablenbelegung festgelegt wird und sodann nur solche Klauseln zugelassen werden, die unter dieser Belegung erfüllt sind. Diese einfache Modifikation des oben beschriebenen Verfahrens ermöglicht die schnelle Erzeugung großer Mengen erfüllbarer k-CNF-Formeln; die so erzeugten Formeln werden auch als *"forced" (Random-k-CNF-) Formeln* bezeichnet. Unglücklicherweise sind die so erzeugten Formelsätze für SAT-Verfahren wesentlich leichter lösbar, als die nach dem oben beschriebenen Verfahren gefilterten allgemeinen Random-k-CNF-Formeln, sodaß sie zur Performanzuntersuchung von SAT-Verfahren kaum geeignet erscheinen (vergl. [BAH+94b]). Neuere Untersuchungen deuten jedoch darauf hin, daß für 3-CNF-Formeln mithilfe einer einfachen Modifikation[1] des beschriebenen Verfahrens zur Erzeugung von "forced"-Formeln erfüllbare, aber sehr schwierige Formeln konstruiert werden können (siehe [KD95]).

Zur Untersuchung unvollständiger SAT-Verfahren auf der Formelklasse Random-k-CNF

[1] Dabei werden beim Erzeugungsprozess lediglich Klauseln zugelassen, die durch die gewählte Belegung in genau einem oder drei Literalen erfüllt werden.

scheint es unumgänglich zu sein, die oben beschriebene Filtermethode anzuwenden. Hier wäre es äußerst sinnvoll, einmalig eine Menge von normierten Testsätzen erfüllbarer k-CNF-Formeln zu erzeugen, um diese dann global für verschiedene Untersuchungen zu nutzen.

Ein schwerwiegender Einwand gegen Unterschungen auf der Grundlage dieser Formelklasse besteht in der Tatsache, daß sich die so erzeugten Formeln in ihrer Schwierigkeit erheblich unterscheiden und daß diese Unterschiede der Probleminstanzen ferner kaum objektiv (d.h. unabhängig von dem verwendeten SAT-Verfahren) meßbar sind. Allerdings ist bekannt, daß Formeln mit sehr wenigen Modellen für SAT-Verfahren, die auf lokaler Suche basieren (z.B. GenSAT-Verfahren), besonders schwierig sind. Eine besonders einfache und in der Vergangenheit oft verwendete Methode, um Random-k-CNF-Formeln mit wenigen Modellen zu erzeugen, basiert auf der Tatsache, daß die Anzahl der Modelle von Random-k-CNF-Formeln mit konstanter Variablenzahl im Mittel bei steigender Klauselanzahl abnimmt. Betrachtet man für die Menge aller k-CNF-Formeln mit einem festen Variablen/Klausel-Verhältnis die Wahrscheinlichkeit P_{sat}, daß eine aus dieser Menge beliebig gewählte Formel erfüllbar ist, so läßt sich bei wachsendem Klausel/Variablen-Verhältnis ein deutlicher Übergang von größtenteils erfüllbaren ($P_{sat} \approx 1$) zu größtenteils unerfüllbaren Formeln ($P_{sat} \approx 0$) feststellen. Der Bereich des Klausel/Variablen-Verhältnisses, indem dieser Übergang stattfindet, wird auch als *Transitions-* oder *Crossover-Bereich* bezeichnet; für Random-3-CNF-Formeln liegt dieser bei einem Klausel/Variablenverhälnis von ≈ 4.26 (vergl. [CA93]).

Durch empirische Untersuchungen ist bekannt, daß die Formeln aus diesem Bereich für vollständige Verfahren (wie DP, vergl. Abschnitt 2.2) besonders schwierig sind. Dennoch wird ihre Eignung als Bewertungsgrundlage für SAT-Verfahren (insbesondere, sofern sie wie GSAT auf lokaler Suche basieren) in letzter Zeit angezweifelt; ein Kritikpunkt basiert dabei auf der Beobachtung, daß die erfüllbaren Formeln aus dem Crossover-Bereich gewöhnlich noch immer eine relativ große Anzahl von Modellen besitzen (vergl. [CI95]).

Infolge der Kritik an der Verwendung von Random-k-CNF-Formeln als Bewertungsgrundlage für SAT-Algorithmen wurden in letzter Zeit spezielle Erzeugungsverfahren für Klassen von CNF-Formeln mit bestimmten, kontrollierbaren Eigenschaften entwickelt (vergl. [AIM93]); insbesondere können auf diese Weise Formeln über beliebig vielen Variablen und beliebigem Klausel/Variablen-Verhältnis konstruiert werden, die genau ein Modell besitzen. Neuere Untersuchungen deuten darauf hin, daß die so erzeugten Formeln als Grundlage zur Bewertung von SAT-Verfahren geeignet sind (siehe [CI95]); allerdings ist über die Eigenschaften dieser Formelklassen recht wenig bekannt.

4.1.2 SAT-kodierte Anwendungsprobleme

Ein alternativer Ansatz zur Bewertung von SAT-Verfahren ist die Evaluierung anhand von SAT-kodierten Problemen aus anderen Bereichen der Informatik oder Mathematik. Im Hinblick auf die Untersuchung der praktischen Einsetzbarkeit von SAT-Verfahren zur Lösung von Anwendungsproblemen scheint diese Vorgehensweise geeigneter zu sein, als die Betrachtung zufällig erzeugter Formeln einer bestimmten syntaktischen Struktur (wie etwa

Random-k-CNF).

Probleme bestehen hierbei einerseits in der Auswahl geeigneter Anwendungsprobleme und andererseits in deren Kodierung als SAT-Probleme. Bei der Problemauswahl lassen sich prinzipiell zwei Vorgehensweisen unterscheiden: die Betrachtung einzelner realer, dem Anwendungsbereich entnommener Probleminstanzen (etwa das 8-Damen-Problem) sowie die Untersuchung ganzer Klassen von Anwendungsproblemen (etwa anhand der Klasse der Hamilton-Kreis-Probleme in Graphen). Ein gravierender Nachteil der ersteren Methode besteht darin, daß die erzielten Resultate in der Regel nur sehr schwer auf andere Probleminstanzen übertragen werden können, wodurch die Relevanz der entsprechenden Untersuchung leicht in Frage gestellt werden kann.

Betrachtet man hingegen ganze Problemklassen, treten bei der Auswahl von Testmengen im wesentlichen die gleichen Probleme auf, wie im Falle der Random-k-CNF-Formeln. Dennoch weist dieser Ansatz durch seinen Anwendungsbezug gewisse Vorzüge auf; unter anderem erlaubt er neben einer vergleichenden Bewertung verschiedener SAT-Verfahren auch einen Vergleich zwischen spezialisierten Algorithmen für das entsprechende Anwendungsproblem zum einen und SAT-Verfahren auf den kodierten Problemeninstanzen zum anderen. Aus solchen Untersuchungen können Erkenntnisse über die Eigenschaften der eingesetzten SAT-Kodierung sowie des betrachteten SAT-Verfahrens gewonnen werden.

Jedenfalls ist bei einer Untersuchung auf der Grundlage von SAT-kodierten Anwendungsproblemen das Kodierungsverfahren von entscheidender Bedeutung für die Performanz des betrachteten SAT-Verfahrens. Für viele Probleme aus verschiedenen Bereichen der Informatik und der Mathematik sind SAT-Kodierungen bekannt; in vielen Fällen dienen sie als Grundlage komplexitätstheoretischer Überlegungen (wie etwa beim Nachweis der $\mathcal{NP}$-Vollständigkeit von Problemen durch Reduktion auf das SAT-Problem) und erzeugen Formeln, die bereits aufgrund ihrer Variablenanzahl für SAT-Verfahren als kaum handhabbar erscheinen. Für einige algorithmische Probleme sind effektive SAT-Kodierungen in dem Sinne bekannt, daß die kodierten Probleme durch SAT-Verfahren mit vertretbarem Aufwand gelöst werden können. Beispiele hierfür sind das k-Färbungsproblem für Graphen, das n-Damen-Problem (siehe [SLM92]), diverse Planungsprobleme (siehe [KS92]) sowie spezielle Scheduling-Probleme (siehe [CB94]).

Weiterhin wurden 1994 von Iwama und Miyazaki SAT-Kodierungen für verschiedene Graphprobleme (Hamiltonsche Kreise, Cliquen und Graphfärbungen) angegegeben, welche effizient bezüglich der Anzahl der benötigten Aussagenvariablen sind, d.h. für Probleme der Größe n höchstens $n \log n$ Variablen benötigen (siehe [IM94]). Im Rahmen der vorliegenden Arbeit wurde eine solche Kodierung des Hamilton-Kreis-Problems in gerichteten Graphen und die Performanz von GSAT auf verschiedenen Teilklassen der so kodierten Probleminstanzen ausführlich untersucht (siehe Kapitel 6–8).

4.2 Leistungsmessung

Die Performanz eines SAT-Verfahrens auf einem gegebenen Satz von Testformeln kann nach verschiedenen Kriterien untersucht werden. Entscheidend für sämtliche Verfahren ist in diesem Zusammenhang das *Laufzeitverhalten*. Für unvollständige Verfahren wie GSAT ist darüber hinaus die *Erfolgsrate* eine wichtige Kenngröße; diese gibt an, mit welcher Wahrscheinlichkeit das Verfahren zu einer erfüllbaren Formel tatsächlich ein Modell findet. Im Fall des MaxSAT-Problems (vergl. Abschnitt 1.4) ist ferner die *Güte der Lösungen* von Interesse, d.h. die Kosten der gefundenen Modelle.

Selbstverständlich ist das Laufzeitverhalten eines SAT-Algorithmus entscheidend von der Laufzeitumgebung (Maschine und Betriebssystem) der jeweiligen Implementierung abhängig. Eine Praxis, die sich augenscheinlich in vielen wissenschaftlichen Publikationen dieses Gebietes etabliert hat, ist die Angabe von Laufzeiten in CPU-Sekunden (also tatsächlicher Rechenzeit). Mag eine solche Angabe für die praktische Anwendbarkeit des Verfahrens auf der jeweils verwendeten Maschine von Interesse sein, so ist sie doch für die vergleichende Betrachtung verschiedener Studien nur äußerst begrenzt verwertbar, da es äußerst schwierig ist, die verschiedenen Ausführungsumgebungen bezüglich ihrer Auswirkungen auf das Laufzeitverhalten des Verfahrens quantitativ in Beziehung zu setzen. Noch kritischer erscheint es in diesem Lichte, sich bei der Bewertung eigener Versuchsreihen auf derartige Laufzeitangaben aus anderen Untersuchungen zu beziehen.

Ein sinnvollerer Ansatz zur Evaluierung des Laufzeitverhaltens besteht darin, den für das Finden eines Modells betriebenen Aufwand bezogen auf einen Satz elementarer Operationen zu betrachten, welcher spezifisch für das jeweilige Verfahren festzulegen ist. Der so definierte *Lösungsaufwand* wird durch Zählung der Ausführung von Elementaroperationen gemessen. Der Satz von Elementaroperationen sollte allgemein derart beschaffen sein, daß jede der gezählten Operationen den gleichen realen Zeitaufwand aufweist. Um dies zu erreichen, ist es denkbar, die Elementaroperationen entsprechend zu gewichten. Gleichzeitig sollten diese einen möglichst direkten Bezug zur allgemeinen Formulierung des Algorithmus aufweisen, um eine weitestgehende Implementierungsunabhängigkeit zu gewährleisten.

Da es in der Praxis oft schwierig ist, derartige Elementaroperationen aus den zu untersuchenden SAT-Verfahren zu gewinnen, kann die erste Forderung dahingehend abgeschwächt werden, daß lediglich *im statistischen Mittel* der gleiche reale Zeitaufwand verlangt wird. Im Falle von GSAT ist es üblich, als elementare Operation einen Durchlauf der innersten Schleife, also die Wertänderung einer Variablen (Flip), zu betrachten; der Lösungsaufwand kann damit implementierungsunabhängig in Flips/Modell angegeben werden.

Um einen Bezug zur realen Laufzeit herzustellen, empfiehlt es sich, den realen Zeitaufwand für die Ausführung der elementaren Operationen anzugeben. Da diese Größe im wesentlichen die implementierungsabhängigen Aspekte des Laufzeitverhaltens beinhaltet, eignet sie sich zur Bewertung von Implementierungen und Laufzeitumgebungen. Für GSAT kann nach der oben angedeuteteten Betrachtungsweise dieser Wert in der Einheit Flips/CPU-Sekunde angegeben werden.

Ein wichtiger Aspekt bei der Bewertung von SAT-Algorithmen ist aufgrund dieser Betrachtungsweise die für die einzelnen Elementaroperationen erreichbare Performanz. Dabei kann unter Umständen, wie im Falle der GenSAT-Algorithmen gezeigt wurde (vergl. Abschnitt 3.6), durch spezielle Implementierungstechniken das Laufzeitverhalten entscheidend verbessert werden. Ebenfalls von großer praktischer Bedeutung sind Möglichkeiten zur Parallelisierung der Elementaroperationen (für GenSAT-Verfahren werden diese genauer in Kapitel 5 diskutiert). Wenn auch eine effiziente Parallelisierung von SAT-Verfahren im allgemeinen aufgrund der $\mathcal{NP}$-Vollständigkeit des SAT-Problems vermutlich nicht zu erreichen ist, so können doch auf diese Weise für spezielle (anwendungsrelevante) Probleme bedeutende Beschleunigungen erzielt werden. Ein eindrucksvolles Beispiel hierfür liefert das in Abschnitt 2.5 erwähnte massiv parallele BioSAT-Verfahren.

4.3 Bewertung von SAT-Algorithmen

Bei der experimentellen Untersuchungen zur Leistungsfähigkeit von SAT-Verfahren ergeben sich verschiedenartige Schwierigkeiten. Kriterien zur Leistungsmessung und die dabei zu berücksichtigenden Probleme wurden bereits im letzten Abschnitt diskutiert. Weitere Probleme ergeben sich aus der Tatsache, daß die betrachteten Algorithmen in der Regel nichtdeterministisch sind (vergl. Abschnitt 2.4), sodaß bei gleichen Testproblemen mit unterschiedlichen Meßergebnissen zu rechnen ist. Tatsächlich zeigt sich, daß in der Regel die empirisch ermittelte Varianz der Meßgröße über verschiedenen Versuchen auf einer Probleminstanz derart groß ist, daß Verteilungsaussagen kaum möglich erscheinen. Um zuverlässige Aussagen über den Erwartungswert zu treffen, wäre folglich eine hohe Anzahl von Einzelversuchen erforderlich. Da jedoch für größere Instanzen schwerer Probleme die Laufzeiten sehr schnell wachsen, ist es in der Regel nicht möglich, Versuchsreihen in dem entsprechenden Umfang durchzuführen.

Eine weitere Schwierigkeit ergibt sich aus der Tatsache, daß oft die Einstellung bestimmter Parameter (wie maxSteps im Falle von GenSAT-Verfahren) einen starken Einfluß auf die Performanz des betrachteten Algorithmus hat. Somit sind oft umfangreiche Experimente nötig, um günstige Parameterwerte zu finden.

Keinesfalls unproblematisch ist auch die Auswahl geeigneter Testprobleme (vergl. Abschnitt 4.1). Hier sind prinzipiell zwei Fälle zu unterscheiden: zufällig erzeugte Formeln aus einer größeren Klasse von SAT-Problemen zum einen und aus realen Instanzen von Anwendungsproblemen gewonnene Formeln zum anderen. Im ersten Fall ist es von großer Wichtigkeit, daß die verwendete Testmenge einen möglichst repräsentativen Ausschnitt aus der gesamten Problemklasse darstellt. Besonders bei großen und uneinheitlichen Problemklassen (z.B. Random-3-CNF) verursacht diese Forderung Schwierigkeiten, da die zum Erzielen statistisch verläßlicher Aussagen benötigten Testmengen aufgrund ihrer Größe im allgemeinen experimentell nicht mehr handhabbar sind.

Im zweiten Fall besteht das Problem eher im Finden geeigneter Anwendungsprobleme. In der Regel ist bereits die Kodierung in ein aussagenlogisches SAT-Problem problematisch, da diese einen erheblichen Einfluß auf die Performanz des zu untersuchenden SAT-Verfahrens hat (bei ungeschickter Kodierung können ursprünglich relativ leichte Probleme in der SAT-Darstellung praktisch unlösbar werden, vergl. [IM94]). Weiterhin sind die an speziellen Anwendungsproblemen gemessenen Ergebnisse in der Regel nur sehr bedingt auf andere Problemklassen übertragbar, sodaß bei dieser Vorgehensweise selten relevante Aussagen über die *generellen* Eigenschaften des betrachteten Algorithmus erzielt werden können.

Zusammenfassend läßt sich also feststellen, daß statistisch gesicherte Aussagen über die Performanz von SAT-Verfahren nur in wenigen Fällen und mit großem experimentellem Aufwand zu erzielen sind. Dennoch wird die Leistung dieser Verfahren, mangels Alternativen, aufgrund experimenteller Untersuchungen bewertet; dabei ist es üblich, eine Mittelwertbildung sowohl über die einzelnen Versuche an einer Formel als auch über den Ergebnissen für die verschiedenen Formeln eines Testsatzes vorzunehmen, sofern dieser in etwa dem im letzten Absatz genannten Kriterium genügt. Die auf diese Weise erzielten Leistungswerte (etwa der mittlere Lösungsaufwand) vermitteln oftmals einen realistischen Eindruck von den Eigenschaften der betrachteten Algorithmen. Neben der quantitativen Leistungsmessung auf der Ebene elementarer Operationen sollte jedoch prinzipiell die im Rahmen des zu untersuchenden Verfahrens zu erzielende Beschleunigung der realen Laufzeit durch besondere Implemtierungstechniken bzw. Parallelisierung des Algorithmus bei der Bewertung berücksichtigt werden (vergl. Abschnitt 3.6).

Ein letzter und wesentlicher Aspekt bei der Beurteilung von SAT-Verfahren, insbesondere wenn diese auf der Grundlage kodierter Anwendungsprobleme erfolgt, besteht in der Betrachtung der *Adäquatheit* (vergl. [Bib92]). Dabei wird ein Verfahren genau dann als adäquat bezeichnet, wenn es für „einfache“ Probleme eine wesentlich bessere Performanz zeigt als für „schwierige“ Probleme. Die Unterscheidung zwischen einfachen und schwierigen Problemen kann dabei durch komplexitätstheoretische Resultate oder auf der Grundlage experimenteller Performanzuntersuchungen für eine große Klasse bekannter Lösungsverfahren getroffen werden. Im ersteren Fall folgt etwa konkret, daß ein adäquates SAT-Verfahren auf der Klasse der Horn-Formeln (SAT-Problem mit polynomiellem Zeitaufwand) wesentlich effizienter arbeiten muß, als auf der Klasse der Random-3-CNF-Formeln ($\mathcal{NP}$-vollständiges SAT-Problem). Im letzteren Fall hingegen folgt etwa, daß eine Klasse SAT-kodierter Anwendungsprobleme, für welche ein effizienter Spezialalgorithmus bekannt ist, auch von einem adäquaten SAT-Verfahren effizient gelöst werden sollte.

Der sehr starke Begriff der Adäquatheit läßt sich abschwächen zur *problemspezifischen Adäquatheit*, indem lediglich eine bestimmte Problemklasse betrachtet wird, welche sowohl einfache als auch schwierige Teilklassen (im oben definierten Sinne) enthält. Die Untersuchung der problemspezifischen Adäquatheit bietet sich besonders für SAT-kodierte Anwendungsprobleme an; ein geeignetes Beispiel bildet die Klasse der SAT-kodierten Hamilton-Kreis-Probleme in gerichteten Graphen, die in Kapitel 6 ausführlich behandelt wird.

Kapitel 5

Parallelisierung von GenSAT-Algorithmen

Da im Bereich der Hoch- und Höchstleistungsrechner schon seit einigen Jahren ein deutlich erkennbarer Trend hin zu Parallelrechnerarchitekturen mit immer größeren Prozessoranzahlen festzustellen ist, liegt die Frage der Parallelisierbarkeit von SAT-Verfahren nahe. Wegen der $\mathcal{NP}$-Vollständigkeit des Erfüllbarkeitsproblems ist nicht zu erwarten, daß mithilfe paralleler Algorithmen deutliche Effizienzsteigerungen im allgemeinen Falle erzielbar sind.

Dennoch ist es aus verschiedenen Gründen interessant, sich mit der Parallisierung von SAT-Verfahren zu beschäftigen. Zum einen existieren, wie bereits erwähnt, Teilklassen des Problems mit polynomieller Zeitkomplexität, sodaß zu erwarten ist, daß eine effiziente Parallelisierung zumindest für eingeschränkte Varianten des SAT-Problems möglich ist. Damit verbindet sich die Hoffnung, einige anwendungsrelevante Aufgabenstellungen auf solche Teilklassen von SAT-Problemen reduzieren zu können. Zum anderen sind bei massiver Parallelisierung (also unter der Annahme extrem großer Prozessoranzahlen) selbst für den $\mathcal{NP}$-vollständigen Fall praktisch relevante Beschleunigungen für gewisse Problemgrößen zu erzielen (bei einer realen Laufzeit von einigen Jahren ist eine Beschleunigung um einen Faktor 1000 bereits ein qualitativer Unterschied bezüglich der praktischen Lösbarkeit eines Problems). Gerade vor dem Hintergrund neuerer Ergebnisse auf dem Gebiet des biomolekularen Rechnens, wo Parallelitätsgrade größer 10^{14} erzielbar sind, erhält dieses Argument neues Gewicht (vergl. Abschnitt 2.5).

Für die in Kapitel 3 eingeführten GenSAT-Algorithmen sollen in den folgenden Abschnitten die prinzipiellen Möglichkeiten der Parallelisierung angedeutet werden. Von diesen Überlegungen unberührt bleibt die wichtige Frage, inwieweit inhärent parallele SAT-Verfahren gefunden werden können, welche sich in ihren Eigenschaften von diesen, durch Parallelisierung sequentieller Algorithmen gewonnenen Verfahren, prinzipiell unterscheiden.

5.1 Direkte Parallelisierung

Die naheliegendsten Möglichkeiten zur Parallelisierung von GenSAT-Algorithmen ergeben sich durch direkte Parallelisierung der `for`-Schleifen. Verschiedene Schleifendurchläufe können genau dann problemlos parallel ausgeführt werden, wenn sie datenunabhängig sind, d.h. keiner der Durchläufe Datenveränderungen bewirkt, die von einem anderen Schleifendurchlauf vorausgesetzt werden. Außerdem müssen Zugriffe auf gemeinsam genutzte Daten (vor allem Schreiboperationen) für die meisten Parallelrechnermodelle synchronisiert werden.

5.1.1 Parallele Versuche

Betrachtet man die äußere `for`-Schleife des GenSAT-Algorithmus unter diesen Gesichtspunkten, ergibt sich folgende Situation: Soweit die einzelnen Versuche unabhängig voneinander sind, ist die parallele Ausführung aller Schleifendurchläufe möglich; dies ist für viele GenSAT-Varianten, insbesondere GSAT, der Fall, nicht jedoch für GSAT mit adaptiven Klauselgewichten. Dieser Parallelisierungsansatz läuft im wesentlichen darauf hinaus, den gesamten Algorithmus mehrfach und parallel auf dieselbe Eingabeformel anzusetzen, daher ist die Synchronisation von Datenzugriffen unproblematisch: Durch mehrfache Datenhaltung ist prinzipiell ein vollständig paralleler Zugriff auf alle wichtigen Daten (insbesondere Eingabeformel und aktuelle Belegung) möglich. Durch eine solche parallele Ausführung mehrerer Versuche ist ein maximaler Parallelitätsgrad[1] von maxTries erreichbar; der Synchronisationsaufwand ist vernachlässigbar gering, da lediglich in dem Fall, daß ein Modell gefunden wurde, alle parallelen Versuche terminiert werden müssen.

5.1.2 Parallele Schritte

Eine grundlegend andere Situation liegt für die innere `for`-Schleife vor: Hier ist jeder Schleifendurchlauf von allen vorherigen abhängig, da jeder Schritt nur ausgehend von der aktuellen Belegung bewertet werden kann. Dennoch ist eine Parallelisierung nicht völlig ausgeschlossen. Geht man davon aus, daß die Datenabhängigkeit lediglich die aktuelle Belegung betrifft,[2] ergibt sich zunächst folgender Ansatz für die parallele Ausführung zweier Schleifendurchläufe $i, i+1$: Da ungewiß ist, welche aktuelle Belegung nach Schritt i vorliegt, werden für Schritt $i+1$ alle durch Schritt i erreichbaren Belegungen berücksichtigt. Bei gegebener Eingabeformel über n Variablen wären damit für die bislang betrachteten GenSAT-Instanzen n Ausgangsbelegungen für Schritt $i+1$ zu betrachten, von denen nach Ausführung von Schritt i nur die tatsächlich erreichte verwendet wird. Um zwei Schritte parallel auszuführen, muß also im wesentlichen der $n+1$-fache Aufwand eines einzelnen

[1] Der *Parallelitätsgrad* gibt an, wieviele Prozessoren im Rahmen eines parallelen Algorithmus simultan arbeiten.

[2] Wird die "score-by-need"-Technik eingesetzt, ist diese Voraussetzung nicht erfüllt, da in diesem Falle eine weitere Datenabhängigkeit über die gespeicherten Bewertungen besteht. Dennoch gelten die folgenden Überlegungen prinzipiell auch für diesen Fall.

Schleifendurchlaufs in Kauf genommen werden. Will man auf diese Weise k Schritte parallel ausführen, entsteht insgesamt ein Aufwand von $O(\exp(k))$ Schritten; somit handelt es sich hier um einen äußerst ineffizienten Parallelisierungsansatz. Durch leichte Modifikationen kann jedoch die Effizienz erheblich verbessert werden; dies bedingt allerdings die Betrachtung konzeptuell veränderter GenSAT-Verfahren und wird daher in den folgenden Abschnitten ausführlicher behandelt.

5.1.3 Parallele select-Implementierungen

Weitere Möglichkeiten der direkten Parallelisierung ergeben sich durch parallele Implementierungen der select-Funktion. Diese Parallelisierungsansätze können zwar lediglich spezifisch für konkrete GenSAT-Instanzen entwickelt werden; dennoch ergeben sich für verschiedene Instanzen gewisse Analogien. Einige grundlegende Ansätze werden im folgenden am Beispiel von GSAT betrachtet.

Ausgehend von der naheliegenden Implementation der Funktion select (vergl. Abschnitte 3.1, 3.6) können die Bewertungen für alle n Variablen der Eingabeformel simultan mit maximalem Parallelitätsgrad n berechnen werden — und damit auch die Menge possFlips der möglichen Belegungsübergänge. Auf diese Weise kann die select-Funktion mit einem parallelen Zeitaufwand von $\Theta(1) \cdot Time(score_F)$ realisiert werden. Für indifferente select-Varianten wie I_2SAT (vergl. Abschnitt 3.3), deren Selektionskriterium lediglich lokale Information über die Nachbarbelegungen benötigen (und somit nicht — wie im Falle von GSAT — die Kenntnis etwa eines globalen Maximums der Bewertung erfordern), ist eine parallele Berechnung mit $p < n$ Prozessoren der Menge possFlips noch effizienter zu realisieren: Dabei werden solange die Bewertungen von jeweils p Variablen simultan berechnet, bis eine Variable gefunden wurde, die das Selektionskriterium erfüllt. Um eine faire Auswahl zu gewährleisten, wird die Reihenfolge, in der die n Aussagenvariablen dabei (in Blöcken von jeweils p) betrachtet werden, in jedem Schritt mithilfe einer zufällige Permutation bestimmt.[3]

Für die in vielen Fällen wesentlich effizientere "score-by-need"-Implementierung der select-Funktion (vergl. Abschnitt 3.6) kann ein ähnlicher Parallelisierungsansatz gewählt werden. Da hier nicht die Berechnung der Menge possFlips sondern die Aktualisierung von Variablenbewertungen nach einem Flip den zeitbestimmenden Schritt darstellt, bietet es sich an, die Menge $V_{dep}(x, F)$ der von der ausgewählten Variable x abhängigen Variablen der Eingabeformel F simultan neu zu bewerten. Ist $VD(n)$ eine obere Schranke für die Mächtigkeit der Mengen $V_{dep}(x, F)$, so kann auf diese Weise bei einem maximalen Parallelitätsgrad von $VD(n)$ die select-Funktion für GSAT mit einem parallelen Zeitaufwand von $O(\log(n) + Time(score_F))$ berechnet werden. Ähnlich wie zuvor kann der parallele Zeitaufwand im Fall von lokalen Selektionskriterien (s.o.) auf $\Theta(Time(score_F))$ reduziert werden.[4]

[3] Der (sequentielle) Zeitaufwand für die Bestimmung einer solchen Permutation liegt bei $O(n)$.

[4] Die Bestimmung der maximalen Bewertung erfordert einen parallelen Zeitaufwand von $\Theta(\log n)$ auf Maschinenmodellen, welche einen gemeinsamen Speicher ohne simultane Schreibzugriffe aufweisen (CREW/EREW-PRAM, siehe [KR90]).

Abschließend sei an dieser Stelle noch die Möglichkeit der parallelen Berechnung bzw. Aktualisierung von einzelnen Bewertungen erwähnt. Unter Ausnutzung von Klauselabhängigkeiten läßt sich für GSAT die Funktion $score_F$ bei einem maximalen Parallelitätsgrad von $O(CD(n) \cdot CL)$ mit einem parallelen Zeitaufwand von $O(\log(CD(n)))$ berechnen; dabei ist $CD(n)$ eine obere Schranke für die Mächtigkeit der Mengen $C_{dep}(x, F)$ der von einer beliebigen Variablen x abhängigen Klauseln und CL die maximale Länge der Klauseln der Eingabeformel F.[5]

Der Synchronisationsaufwand ist für alle hier erwähnten Parallelisierungen der select-Funktion vernachlässigbar; insgesamt läßt sich die select-Funktion für GSAT unter Verwendung der "score-by-need"-Technik bei einem maximalen Parallelitätsgrad von $O(VD(n) \cdot CD(n) \cdot CL)$ mit einem Zeitaufwand von $\Theta(\log n \cdot \log CD(n))$ realisieren, dabei ist n die Anzahl der in der Eingabeformel vorkommenden Aussagenvariablen ($CD(n)$,$VD(n)$ und CL wie oben).

5.2 Vergrößerung der Nachbarschaft

Während bislang nur Parallelisierungsansätze betrachtet wurden, welche lediglich den Zeitaufwand, nicht aber das funktionale Verhalten der betrachteten GenSAT-Instanzen veränderten, werden im folgenden neue, parallele GenSAT-Algorithmen eingeführt.

Eine interessante Variante der parallelen Ausführung von Schritten (s.o.) besteht darin, von der aktuellen Belegung ausgehend verschiedene Schritte simultan weiterzuverfolgen. Dies kann beispielsweise derart realisiert werden, daß in jedem Schritt b Alternativen weiterverfolgt werden, sodaß insgesamt nach k Schritten b^k Folgebelegungen erreicht sind. Diese bilden nun die Menge possFlips, aus der — möglicherweise nach Anwendung eines weiteren Selektionskriteriums (z.B. maximale Bewertung) — die tatsächliche Folgebelegung durch zufällige Auswahl eines Elementes bestimmt wird. Für den Suchvorgang bedeutet dies, daß ausgehend von der aktuellen Position im Suchraum jeweils mit Grad b verzweigt wird, wobei nach jeweils k Schritten von den so erreichten Positionen alle bis auf eine verworfen werden; von dort aus wird dann die lokale Suche fortgesetzt. Diese Vorgehensweise läßt sich auch als simultane Vorausschau (*look-ahead*) um k Schritte interpretieren. Die Wahl von b und k ist zunächst relativ beliebig; sollen jedoch jeweils alle Alternativen in jedem der k Schritte simultan berechnet werden, muß b^k entsprechend dem realisierbaren Parallelitätsgrad gewählt werden.

Das eben skizzierte Verfahren kann alternativ (und unabhängig von der so realisierten Parallelität) durch eine entsprechend modifizierte Nachbarschaftsrelation auf der Menge der Belegungen der Eingabeformel F charakterisiert werden. Derartig erweiterte Nachbarschaftsrelationen sind dadurch gekennzeichnet, daß sie (im Gegensatz zu denjenigen Nachbarschaftsrelationen, welche im Zusammenhang mit den in Kapitel 3 dargestellten GenSAT-Instanzen betrachtet wurden) auch Übergänge zwischen Belegungen zulassen, die

[5] Um dies zu erreichen, werden für alle von x abhängigen Klauseln die Wertänderung simultan in allen Literalen berechnet (Zeitaufwand $O(1)$); die Summe der Wertänderungen kann dann mit Zeitaufwand $O(\log(CD(n)))$ gebildet werden.

eine Hamming-Distanz größer als 1 aufweisen, sich also in mehr als einem Variablenwert unterscheiden (vergl. [BL93]).

Durch eine derartige Erweiterung der Nachbarschaftsrelation wird die Menge possFlips der möglichen Folgebelegungen entsprechend vergrößert, wodurch der Aufwand für die Berechnung dieser Menge steigt. Allerdings kann, wie in Abschnitt 5.1.3 gezeigt, diese Berechnung sehr gut parallelisiert werden, insbesondere falls ein lokales Selektionskriterium verwendet wird. Eine analoge Situation ergibt sich auch bei Einsatz der "score-by-need"-Technik; dabei ist zu beachten, daß mit der Hamming-Distanz zwischen der aktuellen Belegung und der gewählten Folgebelegung in der Regel auch die Anzahl der abhängigen Variablen bzw. Klauseln zunimmt, und somit der Aufwand für die Aktualisierung der Bewertungen wächst. Gleichzeitig steigt der zur Speicherung der Bewertungen benötigte Platzbedarf mit der Anzahl der möglichen Belegungsübergänge, woraus sich eine klare Restriktion für die verwendete Nachbarschaftsrelation ergibt.

Im Zusammenhang mit der im Rahmen dieser Arbeit durchgeführten experimentellen Untersuchungen an SAT-kodierten Hamilton-Kreis-Problemen (siehe Kapitel 6ff.) wurde eine GenSAT-Instanz betrachtet, welche auf einer problemspezifisch erweiterten Nachbarschaftsrelation beruht. Die hier erzielten Ergebnisse deuten darauf hin, daß mithilfe dieser Vorgehensweise bereits im sequentiellen Fall deutliche Performanzsteigerungen gegenüber konventionellen GenSAT-Algorithmen (wie etwa GSAT) erreichbar sind. Wichtige Konzepte wie "score-by-need" oder die Random-Walk-Strategie können dabei problemlos übernommen werden.

5.3 Simultane Änderung von Variablenbelegungen

Ein weiterer Ansatz zum Entwurf paralleler GenSAT-Instanzen orientiert sich an einer von Boissin und Lutton vorgeschlagenen und untersuchten Methode zur Parallelisierung eines Simulated-Annealing-Algorithmus (siehe [BL93]). Dabei wird in jedem Schritt für mehrere Variablen simultan und unabhängig voneinander entschieden, ob deren Wert geändert wird oder nicht. Im allgemeinen ist zunächst in keiner Weise garantiert, daß der aus der Kombination dieser Einzelübergänge resultierende Multiflip in irgendeiner Weise optimal oder auch nur günstig ist. Um dieser Problematik zu begegnen, wird im parallelen SA-Algorithmus von Boissin und Lutton jeder so konstruierte Multiflip nochmals dem Metropolis-Test unterworfen (d.h. die tatsächliche Ausführung des gesamten Schritts wird anhand einer Metropolis-verteilten Zufallsvariable entschieden). Auf diese Weise wird gewährleistet, daß bezüglich der Bewertungsfunktion günstige Schritte bevorzugt ausgeführt werden, wobei die Ausführungswahrscheinlichkeit mit steigender Güte des Schrittes streng monoton wächst (vergl. Abschnitt 2.4). Obwohl dieses parallele SA-Verfahren in experimentellen Untersuchungen gute Konvergenz- und Laufzeiteigenschaften zeigte, konnte das theoretische Vollständigkeitsresultat für SA für diese Variante bislang nicht bewiesen werden.

Auf der Basis der Formulierung des SASAT-Verfahrens als GenSAT-Instanz (siehe Abschnitt 3.5) kann auch das parallele Verfahren von Boissin und Lutton auf einfache Weise in den

GenSAT-Rahmen übertragen werden. Das resultierende probabilistische GenSAT-Verfahren kann dann über eine select-Funktion mit erweiterter Nachbarschaft realisiert werden; bei einem Parallelitätsgrad von n (Anzahl der Variablen der gegeben CNF-Formel) sind dann in jedem Schritt Übergänge zu beliebigen Folgebelegungen möglich. Ein gravierender Nachteil dieses Ansatzes besteht in der Tatsache, daß die "score-by-need" nicht mehr effektiv einsetzbar ist, da die Menge der möglichen Übergänge eine exponentielle Größe bezüglich der Anzahl der Variablen aufweist und somit der für die Speicherung der Bewertungen benötigte Platzaufwand nicht akzeptabel ist.

Dieses Problem kann jedoch behoben werden, indem man bei simultanen Variablenübergängen nur Mengen paarweise unabhängiger Variablen zuläßt (*unabhängige Multiflips*). In diesem Fall ergibt sich jede durch einen Multiflip verursachte Bewertungsänderung als Summe der durch die entsprechenden Einzelflips verursachten Bewertungsänderungen. Aus diesem Grund kann auf eine zusätzliche Bewertung des gesamten Übergangs (wie sie im Fall des oben erwähnten, parallelen SA-Verfahrens durch den Metropolis-Test realisiert wird) verzichtet werden. Die Variablenabhängigkeiten lassen sich für jede gegebene Eingabeformel F mit vertretbarem Aufwand vor der Ausführung des eigentlichen SAT-Algorithmus bestimmen, sodaß diese Art von Multiflips in jedem Falle ohne wesentliche Effizienzverluste ausgeführt werden können. Der auf diese Weise erreichbare maximale Parallelitätsgrad ist gleich der Mächtigkeit einer maximalen Menge paarweise unabhängiger Variablen von F und damit stark von der Struktur der Formel F abhängig.

Der Ansatz der unabhängigen Multiflips kann leicht mit einer erweiterten Nachbarschaft kombiniert werden, indem alternativ (evtl. mit einer bestimmten Wahrscheinlichkeit) Schritte der einen bzw. der anderen Art durchgeführt werden (analoge Vorgehensweise zur Random-Walk-Erweiterung von GSAT, siehe Abschnitt 3.3.2).

5.4 Unabhängige und kooperierende Agentenmodelle

Bislang wurden lediglich Parallelisierungsmöglichkeiten innerhalb des GenSAT-Rahmens diskutiert. Ein hiervon grundlegend verschiedener Ansatz zur parallelen Lösung von SAT-Problemen besteht in der simultanen Ausführung mehrerer, möglicherweise verschiedener sequentieller SAT-Verfahren auf derselben Probleminstanz. Diese Vorgehensweise wird durch den Begriff des *Multi-Agenten-Modells* gefaßt: Hierbei bearbeitet eine Menge von eigenständigen Prozessen, die *Agenten*, simultan dieselbe Probleminstanz; hat ein Agent eine Lösung gefunden, werden alle übrigen Agenten terminiert.

Realisieren dabei alle Agenten den gleichen Algorithmus, handelt es sich um ein *homogenes Multi-Agenten-Modell*, anderenfalls spricht man von einem *heterogenen Multi-Agenten-Modell*. Weiterhin unterscheidet man *unabhängige Modelle*, in denen die einzelnen Agenten keinerlei Information austauschen können, von *kooperativen Modellen*, welche einen Informationsaustausch zwischen Agenten zulassen.

Die in Abschnitt 5.1.1 aufgezeigte Möglichkeit der GenSAT-Parallelisierung durch simultane

Ausführung mehrerer Versuche kann in diesem Sinne als Realisierung eines unabhängigen, homogenen Multi-Agenten-Modells angesehen werden; dabei entspricht jeder der simultan ablaufenden GenSAT-Versuche einem Agenten, d.h. einem eigenständigen GenSAT-Prozess (der nur einen Versuch ausführt).

GenSAT-Parallelisierungen in Form von unabhängigen Multi-Agenten-Modellen sind in vielen Varianten denkbar: als homogene Modelle mit Agenten, welche jeweils eine bestimmte GenSAT-Variante realisieren, oder als heterogene Modelle, wobei verschiedene Agenten die gleiche GenSAT-Instanz mit unterschiedlichen Parametereinstellungen (z.B. Random-Walk mit verschiedenen Walk-Wahrscheinlichkeiten) oder sogar unterschiedliche GenSAT-Varianten (z.B. GSAT, ISAT und GSAT mit Random Walk) realisieren. Der so erreichte Parallelitätsgrad entspricht (bei sequentiellen Agenten) der Anzahl der Agenten.

Anders als die zuvor behandelten Parallelisierungsmöglichkeiten sind jedoch Multi-Agenten-Modelle nicht allein durch die Bestrebung nach Verringerung des Zeitaufwandes motiviert. Wie bereits in Abschnitt 4.1.1 erwähnt, ist für etliche Klassen von SAT-Problemen eine Abhängigkeit der Schwierigkeit des Problems von einem oder mehreren Parametern dergestalt gegeben, daß die schwierigsten Probleme in einem bestimmten Parameterbereich, dem Transitionsbereich, auftreten. Dort läßt sich unter anderem für eine gebenene Probleminstanz eine besonders große Varianz des Lösungsaufwandes beim Einsatz verschiedener Verfahren, verschiedener Parametereinstellungen und sogar bei verschiedenen Versuchen mit einem einzigen probabilistischen Verfahren feststellen (vergl. [HW93]). Aufgrund dieser Beobachtung ist zu erwarten, daß für ein gegebenes Verfahren bereits durch den Übergang zu einem homogenen, unabhängigen Multi-Agenten-Modell eine Verbesserung der Performanz erzielt werden kann, selbst wenn dieses Modell lediglich auf einer sequentiellen Maschine (durch Multiplexing) realisiert wird (vergl. [HW94]). Noch lohnender erscheint der Einsatz heterogener Multi-Agenten-Modelle mit möglichst verschiedenen Agenten; auf diese Weise kann bei gegebener Probleminstanz die Varianz des Lösungsaufwandes bei Verwendung verschiedener Lösungsverfahren ausgenutzt werden.

Wie durch experimentelle Untersuchungen am Beispiel des $\mathcal{NP}$-vollständigen Graph-Färbungs-Problems gezeigt wurde, kann durch den Übergang zu kooperativen Multi-Agenten-Modellen eine weitere Verbesserung der Performanz erreicht werden (siehe [HW93]). Dort wurde ein heterogenes, kooperatives Multi-Agenten-Modell unter Einsatz verschiedener heuristischer Lösungsverfahren realisiert. Die Kooperation der Agenten erfolgt mithilfe von *Hinweisen* ("hints"), die von einzelnen Agenten erzeugt, in einen gemeinsamen Speicherbereich ("blackboard") geschrieben, und von beliebigen anderen Agenten gelesen werden können. Im beschriebenen Fall wurden als Hinweise partielle Lösungen der Probleminstanz verwendet. Aufgrund der engen strukturellen Verwandschaft des Graph-Färbungs-Problems zu bestimmten Klassen von SAT-Problemen ist davon auszugehen, daß zumindest in bestimmten Fällen durch den Einsatz heterogener, kooperativer Multi-Agenten-Modelle auch für das SAT-Problem Performanzsteigerungen gegenüber den bekannten Verfahren (vergl. Kapitel 2) erzielt werden können. Hierbei ergeben sich jedoch neue Problemstellungen — wie etwa das Finden guter Strategien zur Erzeugung und Verwertung von Hinweisen — deren Lösung für den Erfolg derartiger Verfahren von entscheidender Bedeutung ist.

Kapitel 6

Die SAT-Transformation von HCP

Das *Hamilton-Kreis-Problem* ("Hamilton Circuit Problem", kurz HCP) besteht in der Aufgabe, in einem gegebenen Graphen einen geschlossenen Weg zu finden, der jeden Knoten genau einmal durchläuft. Instanzen des HCP können als aussagenlogische SAT-Probleme kodiert werden, sodaß die Erfüllbarkeit der erzeugten CNF-Formel äquivalent ist zur Existenz eines Hamiltonschen Kreises im gegebenen Graphen. Darüberhinaus entsprechen die Modelle der Formel genau den Hamiltonschen Kreisen im Graphen, sodaß es prinzipiell möglich ist, mithilfe eines konstruktiven SAT-Verfahrens für die Aussagenlogik Hamiltonsche Kreise in Graphen zu bestimmen. Über derartige Kodierungen kann die $\mathcal{NP}$-Vollständigkeit des HCP gezeigt werden.

Damit eine solche Vorgehensweise praktikabel wird, muß die Kodierung von HCP nach SAT so vorgenommen werden, daß die Problemgröße dabei nur unwesentlich wächst; außerdem muß das Kodierungsverfahren durch einen zeiteffizienten Algorithmus realisierbar sein. Solche effizienten Kodierungsverfahren sind mittlerweile bekannt (siehe [IM94]), und die auf diese Weise erzeugten SAT-Probleme sind von einer prinzipiell durch die genannten Verfahren handhabbaren Größenordnung. Somit liegt es nahe, die Idee der Lösung von HCP-Instanzen mithilfe aussagenlogischer SAT-Verfahren nunmehr einer genaueren Untersuchung zu unterziehen.

Obwohl das HCP im allgemeinen $\mathcal{NP}$-hart ist, variiert die Schwierigkeit einzelner Instanzen enorm. Für zufällig gewählte Instanzen ist diese in hohem Maße vom Ausgrad der Knoten des zugrundeliegenden Graphen (also der Anzahl der Kanten, die jeden Knoten verlassen) abhängig. Dies ermöglicht die gezielte Erzeugung von unterschiedlich schwierigen Probleminstanzen und damit die Untersuchung der Adäquatheit von SAT-Algorithmen (im Zusammenspiel mit den verwendeten Kodierungsverfahren); diese sollten nämlich idealerweise „leichte" Instanzen wesentlich effizienter behandeln können, als die wirklich schwierigen Fälle.

Ausgehend von der ursprünglichen Motivation dieser Arbeit ist die SAT-Kodierung von HCP-Instanzen im Zusammenhang mit der MaxSAT-Transformation eines speziellen molekularbioblogischen Anwendungsproblems von Interesse (siehe Kapitel 9).

6.1 Das Hamilton-Kreis-Problem in gerichteten Graphen

Formal kann das Hamilton-Kreis-Problem (HCP) folgendermaßen definiert werden (vergl. [Jun90]):

Definition 6.1 (HCP in gerichteten Graphen)

Gegeben sei ein gerichter Graph $G = (V, E)$ mit Knotenmenge V und Kantenrelation E. Ein Hamiltonscher Kreis *ist ein einfacher Kreis in G, der jeden Knoten des Graphen genau einmal durchläuft; ein solcher Kreis entspricht einer Knotenfolge $(v_1, v_2, \ldots, v_n)$ für die gilt:*

- $\bigcup\{v_i \mid 1 \leq i \leq n\} = V$
- $v_i \neq v_j$ *für alle* $i \neq j$
- $(v_i, v_{i+1}) \in E$ *für alle* $1 \leq i \leq n-1$
- $(v_n, v_1) \in E$

Das Hamilton Kreis Problem (Hamilton Circuit Problem; HCP) *besteht in der Entscheidung, ob G einen Hamiltonschen Kreis enthält. Graphen, die einen solchen Kreis enthalten, werden als* Hamiltonsche Graphen *bezeichnet.*

V = {1,2,3,4}

E	1	2	3	4
1	$-$	$\oplus$	$+$	$-$
2	$+$	$-$	$\oplus$	$-$
3	$+$	$-$	$-$	$\oplus$
4	$\oplus$	$-$	$+$	$-$

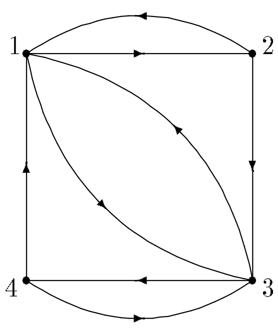

Abbildung 6.1: Beispielgraph G mit Knotenmenge V und Kantenrelation E; die Kanten, welche den Hamiltonschen Kreis bilden, sind mit $\oplus$ markiert.

Beispiel 6.1 Hamiltonsche Kreise in Graphen

Der Graph G in Abbildung 6.1 enthält genau einen Hamiltonschen Kreis; dieser entspricht der Knotenfolge $(1, 2, 3, 4)$.

Die $\mathcal{NP}$-Vollständigkeit dieses Problems wird formal durch die Reduktion des Erfüllbarkeitsproblems in der Aussagenlogik (SAT) auf das HCP bewiesen (siehe [Rei90]). Grundlage des Beweises ist eine Abbildung zwischen CNF-Formeln und Graphen, die so beschaffen ist, daß die Erfüllbarkeit der Formel der Existenz eines Hamiltonschen Kreises im entsprechenden Graphen äquivalent ist.

Das HCP läßt sich in verschiedener Weise verallgemeinern. Wird die Bedingung, daß der zu findende Weg geschlossen ist, fallengelassen, erhält man das *Hamilton-Pfad-Problem (HPP)* in Graphen. Dieses hat Anwendungen in verschiedenen Bereichen (u.a. Testen von komplexen Schaltungen und Robotik) und kann auf ähnliche Weise wie das HCP effizient als SAT-Problem kodiert werden. Eine andere Verallgemeinerung ist das bekannte *Problem des Handlungsreisenden* ("Traveling-Salesman Problem", kurz TSP) in bewerteten Graphen. Dabei sind die Kanten des Graphen mithilfe einer Kostenfunktion bewertet; das Problem besteht darin, einen Hamiltonschen Kreis zu finden, dessen Kosten eine gegebene Schranke nicht überschreiten (vergl. [Jun90]). Oft wird die konstruktive Version des TSP auch so formuliert, daß eine möglichst optimale, d.h. kostenminimale Lösung zu finden ist. Dieses Optimierungsproblem spielt in vielen Anwendungsbereichen, auch außerhalb der Informatik, eine wichtige Rolle (z.B. Steuerung von Produktionsprozessen, Tourenplanung; vergl. [Rei94]).

6.2 Die SAT-Kodierung von HCP-Instanzen

Um HCP-Instanzen mithilfe von SAT-Verfahren zu lösen, wird ein *effizientes Transformationsverfahren*[1] von HCP auf SAT benötigt; darunter soll ein Verfahren verstanden werden, welches für eine beliebige HCP-Instanz mit n Knoten in polynomieller Zeit eine CNF-Formel mit $O(n \log n)$ Variablen liefert, die genau dann erfüllbar ist, wenn die HCP-Instanz lösbar ist. Da ein konstruktives Verfahren gewünscht ist, wird außerdem gefordert, daß die Lösungen einer HCP-Instanz mit polynomiellem Zeitaufwand aus den Modellen der zugehörigen CNF-Formel berechnet werden können. Der Begriff der effizienten Transformation kann auf andere Problemklassen verallgemeinert werden.

Eine weitere Motivation zur Untersuchung solcher Transformationsverfahren liegt in der Annahme, daß die (problemgrößenabhängige) Anzahl der zur SAT-Kodierung eines Problems benötigten Variablen als Maß für die Komplexität des Problems angesehen werden kann. Dies führt zum Begriff der *SAT-Variablen-Komplexität* (siehe [IM94]); dabei können

[1] Synonym soll hierfür auch der Begriff der effizienten Kodierung benutzt werden.

für ein konkretes Problem obere Schranken für dieses Komplexitätsmaß durch Angabe entsprechender Transformationsverfahren bewiesen werden.

6.2.1 Effiziente Kodierung nach Iwama & Miyazaki

Für die Reduktion verschiedener Graph-Probleme auf SAT wurden 1994 von Iwama und Miyazaki effiziente Verfahren angegeben (siehe [IM94]). Das Basisverfahren für HCP beruht dabei auf folgender Idee:

Es werden Permutationen der Knoten des gegebenen Graphen G betrachtet. Eine solche Permutation repräsentiert genau dann einen Hamiltonschen Kreis, wenn

- aufeinanderfolgende Knoten immer durch eine Kante im Graphen verbunden sind und
- der letzte Knoten der Permutation mit dem ersten durch eine Kante verbunden ist.

Zur Kodierung in SAT werden jedem Knoten v_i die Aussagenvariablen $v_{i,k}, v_{i,k-1}, \ldots, v_{i,1}$ zugeordnet, deren Belegung die Position des Knotens in der Permutation als Binärzahl kodiert (die geeignete Wahl von k wird später erläutert)[2]. Für den letzten Knoten können diese Variablen eingespart werden, da er ohne Beschränkung der Allgemeinheit immer an letzter Stelle der Permutation gesetzt werden kann.

Durch eine beliebige Variablenbelegung B wird damit jedem Knoten des Graphen G eine Position zugeordnet; dies kann wie folgt formal definiert werden:

Definition 6.2 (Knotenindizes und Knotenbelegungen)

Gegeben sei eine Instanz des HCP durch den gerichteten Graphen G mit n Knoten. Die Menge N der Knotenindizes *von G sei definiert als:*

$$N = \{\nu \mid 1 \leq \nu \leq n-1\}$$

Jedem Knoten des Graphen G sei eineindeutig ein Knotenindex aus N zugeordnet. Seien $v_{i,j}$ die Aussagenvariablen aus der Iwama-Miyazaki-Kodierung von G, wobei jedem beliebigen Knoten mit Knotenindex i jeweils k Variablen $v_{i,1}, \ldots, v_{i,k}$ zugeordnet sind.

Zu jeder Variablenbelegung B wird nun eine Abbildung $vv_B : N \mapsto N$ definiert, welche jedem Knoten seine durch B kodierten Knotenpositionen zuordnet:

$$vv_B(i) := \sum_{j=1}^{k} iv_B(v_{i,j}) \cdot 2^{j-1} \qquad iv_B(v_{i,j}) := \begin{cases} 1 & : \; B(v_{i,j}) = \top \\ 0 & : \; sonst \end{cases}$$

Für beliebige B werden die Abbildungen vv_B als Knotenbelegung *von G bezeichnet.*

---

[2] Die Knotenpositionen werden mit 0 beginnend numeriert, um einen direkten Bezug der Binärkodierung zur Belegung der Aussagenvariablen herzustellen.

Beispiel 6.2

Für den Beispielgraphen G werden den Knoten $1, 2, 3$ jeweils 2 Aussagenvariablen zugeordnet; diese sind in Abbildung 6.2 als $v_{1,1}, v_{1,2}, \ldots$ neben den Knoten notiert. Die angegebene Variablenbelegung kodiert gerade den durch die Knotenpermutation $(1, 2, 3, 4)$ charakterisierten Hamiltonschen Kreis von G. Dem Knoten 4, dem keinen Aussagenvariablen zugeordnet wurden, wird grundsätzlich die Position 3 in der durch die Variablenbelegung kodierten Permutation zugeordnet.

i	1	2	3
$vv_B(i)$	0	1	2

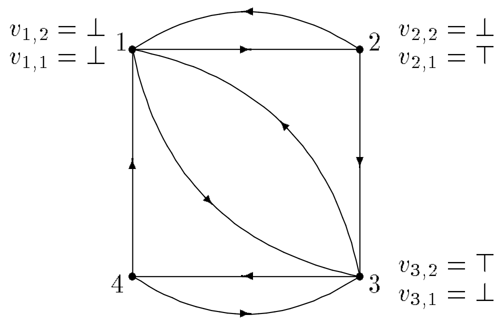

Abbildung 6.2: Beispielgraph G mit Knotenvariablen (rechts); die angegebene Variablenbelegung induziert gerade die Knotenbelegung, die dem Hamiltonschen Kreis in G entspricht (links).

Der Einfachheit halber seien im folgenden beliebige Knotenbelegungen vv_B des Beispielsgraphen G als $(vv_B(1), vv_B(2), vv_B(3))$ notiert. Damit enspricht die Knotenbelegung $(0, 2, 1)$ der Permutation $(1, 3, 2, 4)$, während die Knotenbelegung $(0, 3, 1)$ dem Knoten 2 eine ungültige Position zuordnet und damit keiner gültigen Permutation entspricht.

Eine Knotenbelegung kodiert genau dann eine Permutation der Knoten im Graphen, wenn

- jedem Knoten eine gültige Position zugeordnet ist (ungültige Positionen sind solche, die größer sind als die Anzahl der Knoten im Graphen) und
- niemals zwei verschiedenen Knoten die gleiche Position zugeordnet ist.

Der HCP-Instanz wird nun eine CNF-Formel über den Knotenvariablen zugeordnet, welche genau dann erfüllt ist, wenn

- die Variablenbelegung eine Permutation kodiert,
- Knoten mit aufeinanderfolgenden Positionen in dieser Permutation immer durch eine Kante im Graphen verbunden sind,

- der letzte Knoten der Permutation mit dem ersten durch eine Kante verbunden ist.

Zur Darstellung des Verfahrens zur effizienten SAT-Kodierung von HCP-Instanzen wird folgende Schreibweise benutzt:

Definiere $C(i,l)$ als CNF-Klausel über den Variablen $v_{i,1},\ldots,v_{i,k}$ so, daß der Wert von $C(i,l)$ genau dann $\perp$ ist, wenn für die Belegung von Knoten i gilt: $vv_B(i) = l$. Es sei daran erinnert, daß Klauseln als Mengen von Literalen aufgefaßt werden können, sodaß zwei Klauseln durch Vereinigung ihrer Literalmengen kombiniert werden können.

Beispiel 6.3

Für die Kodierung des Beispielgraphen G aus Abbildung 6.1 (s.o.) ergibt sich damit beispielsweise: $C(1,2) = \neg v_{1,2} \vee v_{1,1}$; es gilt: $C(1,2) = \perp$ gdw. $vv_B = 2$.

Damit kann das Kodierungsverfahren nach Iwama und Miyazaki folgendermaßen formalisiert werden:

Gegeben sei das HCP über einem gerichteten Graphen $G = (V,E)$ mit Knotenmenge $V = \{v_0,\ldots,v_{n-1}\}$ und Kantenrelation $E \subseteq V \times V$. Jedem Knoten v_i mit $0 \le i \le n-2$ werden $k = \lceil \log_2(n-1) \rceil$ Aussagenvariablen $v_{i,1},\ldots,v_{i,k}$ zugeordnet.

Schritt 1: Erzeuge zu jedem Knoten v_i mit $0 \le i \le n-2$ die Klauseln $C(i,l)$ für $n-1 \le l \le 2^k - 1$. Diese Klauselmenge ist genau dann erfüllt, wenn allen Knoten gültige Positionen zugeordnet sind.

Schritt 2: Erzeuge zu jedem Paar von Knoten v_i, v_j mit $0 \le i,j \le n-2$ die Klauseln $C(i,l) \cup C(j,l)$ für $0 \le l \le n-2$. Diese Klauselmenge ist genau dann erfüllt, wenn keinem Paar verschiedener Knoten dieselbe Position zugeordnet wird.

Schritt 3: Erzeuge zu jedem Paar von Knoten v_i, v_j mit $0 \le i,j \le n-2$ und $(v_i,v_j) \notin E$ die Klauseln $C(i,l-1) \cup C(j,l)$ für $1 \le l \le n-2$. Diese Klauselmenge ist genau dann erfüllt, wenn alle Knoten mit aufeinanderfolgenden Positionen durch Kanten im Graphen verbunden sind.

Schritt 4.1: Erzeuge zu jedem Knoten v_i mit $0 \le i \le n-2$ und $(v_i, v_{n-1}) \notin E$ die Klausel $C(i,n-2)$.

Schritt 4.2: Erzeuge zu jedem Knoten v_i mit $0 \le i \le n-2$ und $(v_{n-1}, v_i) \notin E$ die Klausel $C(i,0)$. Die Schritte 4.1 und 4.2 sind Spezialisierungen von Schritt 3 für den Knoten v_{n-1}, dem implizit immer die Position $n-1$ zugeordnet ist.

Dieses Verfahren erzeugt zu einem Graphen mit n Knoten eine CNF-Formel mit $(n-1)\lceil \log_2(n-1) \rceil$ Variablen und $\Theta(n^3)$ Klauseln; es besitzt ferner eine polynomielle Zeitkomplexität (wie in Abschnitt 6.2.3 gezeigt wird) und realisiert damit eine effiziente Transformation von HCP auf SAT. Es ist sowohl für ungerichtete als auch für gerichtete Graphen verwendbar.

Beispiel 6.4 Iwama-Miyazaki-Kodierung

Im folgenden ist die Iwama-Miyazaki-Kodierung für den Graphen G in Abbildung 6.1 (s.o.) angegeben; dabei sind links jeweils die Bedingungen (in prädikatenlogischer Form) aufgeführt, rechts finden sich die zugehörigen Klauseln.

(1)	$\forall i \in \{1,2,3\}: \; v_i \leq 3$	$\neg v_{i1} \vee \neg v_{i2}$
(2)	$\forall (i,j) \in \{(1,2),(1,3),(2,3)\}: \; v_i \neq v_j$	$v_{i2} \vee v_{i1} \vee v_{j2} \vee v_{j1}$ $\neg v_{i2} \vee v_{i1} \vee \neg v_{j2} \vee v_{j1}$ $v_{i2} \vee \neg v_{i1} \vee v_{j2} \vee \neg v_{j1}$
(3)	$\forall (i,j) \notin E: \; v_i \neq v_j + 1$	$v_{32} \vee v_{31} \vee v_{22} \vee \neg v_{21}$ $v_{32} \vee \neg v_{31} \vee \neg v_{22} \vee v_{21}$
(4.1)	$\forall (i,n) \notin E: \; v_i \neq n-2$	$\neg v_{12} \vee v_{11}$ $\neg v_{22} \vee v_{21}$
(4.2)	$\forall (n,j) \notin E: \; v_j \neq 0$	$v_{22} \vee v_{21}$

Zur Kodierung des HCP über dem gegebenen Graphen G mit 4 Knoten und 8 Kanten werden somit insgesamt 6 Aussagenvariablen und 17 Klauseln benötigt.

6.2.2 Weitere Optimierungen

Eine einfache Überlegung zeigt, daß eine Möglichkeit zur wesentlichen Verringerung der zur Kodierung benötigten Aussagenvariablen im allgemeinen Fall nicht zu erwarten ist: Zu einem Graphen mit n Knoten sind $n!$ Permutationen der Knoten möglich. Da jede von diesen einen Hamiltonschen Kreis repräsentieren kann, müssen alle diese Permutationen unterschiedlichen Variablenbelegungen in der SAT-Kodierung entsprechen. Wegen $\ln n! \approx (n+1/2)\ln n - n + 1/2\ln(2\pi)$ (siehe [BS79]) bedeutet dies aber, daß die SAT-Kodierung $\Theta(n \log n)$ Aussagenvariablen benötigt. Somit ist die Anzahl der durch das Kodierungsverfahren von Iwama und Miyazaki benötigten Aussagenvariablen der Größenordnung nach nicht mehr zu unterbieten.

Dennoch sind kleinere Einsparungen, besonders für spezielle Teilklassen von HCP, zu erreichen. So geben Iwama und Miyazaki für den Fall des HCP in ungerichteten Graphen ein verbessertes Verfahren an, durch welches im ungünstigsten Falle eine, in der Regel sogar mehrere Variablen eingespart werden können. Dieses Verfahren basiert auf der Tatsache, daß jeder ungerichtete Graph mit n Knoten und Knotengrad $\geq n/2$ mindestens einen Hamiltonschen Kreis besitzt (vergl. [IM94]). Diese Eigenschaft kann in polynomieller Zeit geprüft werden, sodaß für das folgende Verfahren davon ausgegangen werden kann, daß mindestens ein Knoten einen Grad $d \leq n/2 - 1$ aufweist. Für diesen Knoten kann nun statt seiner Position in der Permutation kodiert werden, durch welche Kante er mit seinem Vorgänger in der Permutation verbunden ist. Auf diese Weise kann bei der Kodierung der Knotenpermutation mindestens eine Variable eingespart werden. Dieses Prinzip ist auch für mehrere Knoten mit Grad $\leq n/2-1$ anwendbar, solange diese nicht direkt durch Kanten im Graphen verbunden sind.[3]

Insgesamt kann auf diese Weise eine HCP-Instanz mit n Knoten auf ein SAT-Problem über $(n-l-1)\lceil\log_2(n-1)\rceil + \sum d_i$ Variablen realisiert werden; dabei ist $\{u_1,\dots,u_l\}$ eine Menge von l Knoten im gegebenen Graphen mit der Eigenschaft, daß zwei dieser Knoten niemals durch eine Kante direkt miteinander verbunden sind und d_i bezeichnet den Grad des Knotens u_i aus dieser Menge.

Ein Nachteil des Kodierungsverfahrens nach Iwama und Miyazaki ist die große Anzahl von Klauseln in der entstehenden CNF-Formel. Obwohl viele Untersuchungen an GSAT und ähnlichen SAT-Verfahren zu dem Ergebnis kommen, daß eine im Verhältnis zur Variablenanzahl große Zahl von Klauseln sich eher günstig auf das Verhalten des Algorithmus auswirkt, ist a priori nicht klar, ob auch im speziellen Fall der SAT-kodierten HC-Probleme eine große Klauselanzahl unproblematisch ist. Allerdings ist kein alternatives Kodierungsverfahren bekannt, welches mit weniger als $\Theta(n^3)$ Klauseln auskommt.

Eine naheliegende Kodierung für das HCP besteht darin, jeder Kante des Graphen eine Variable zuzuordnen, deren Belegung kodiert, ob die Kante Bestandteil des gesuchten Hamiltonschen Kreises ist. Eine solche Kodierung benötigt im günstigsten Falle $\Theta(n)$ Variablen (bei konstantem Knotengrad), im ungünstigsten Falle jedoch $\Theta(n^2)$ Variablen. Zumindest für Graphen mit konstantem (und kleinem) Knotengrad wäre diese Kodierung vom Standpunkt der Variablenanzahl der Iwama-Miyazaki-Transformation nicht automatisch unterlegen. Die Eigenschaft eines Hamiltonschen Kreises, daß jeder Knoten genau einmal besucht wird, kann auf dieser Idee basierend leicht kodiert werden. Das Problem besteht in der Kodierung der Bedingung, daß der gesuchte Weg im Graphen zusammenhängend und geschlossen ist. Die naheliegende Kodierung dieser Eigenschaft erfordert die Einführung von $\Theta(n^2)$ zusätzlichen Variablen zur Kodierung der Erreichbarkeitsrelation zwischen den Knoten; die benötigte Transitivität dieser Relation ist dann durch $\Theta(n^3)$ Klauseln auszudrücken. Dieser Kodierungsansatz führt damit zu einer SAT-Darstellung mit $\Theta(n^2)$ Variablen und $\Theta(n^3)$ Klauseln, ist also dem Iwama-Miyazaki-Verfahren deutlich unterlegen und kaum praktikabel.

Beispiel 6.5

Für den Graphen G aus Abbildung 6.2 (s.o.) ist die hier angesprochene, „naive“ SAT-Kodierung des HCP im folgenden angegeben; dabei sind die Kantenvariablen mit E_i benannt, während die Variablen zur Kodierung der Erreichbarkeitsrelation mit R_{ij} bezeichnet sind. In der Darstellung der aus der Kodierung gewonnenen CNF-Formel sind links jeweils die Bedingungen (in prädikatenlogischer Form) aufgeführt, rechts findet sich jeweils ein Beispiel für die zugehörigen Klauseln.

[3] Eine genauere Darstellung findet sich in [IM94].

(1)	$\forall i \neq j \neq k \neq i \in V: \neg R_{ik} \vee \neg R_{kj} \vee R_{ij}$	$\neg R_{12} \vee \neg R_{23} \vee R_{13}$ $\ldots$
(2)	$\forall (i,j) \in E: \neg E_{ij} \vee R_{ij}$	$\neg E_{12} \vee R_{12}$ $\ldots$
(3)	$\forall i \neq j \in V: R_{ij}$	R_{12} $\ldots$
(4.1)	$\forall i \in V: \bigvee\{E_{ik} \mid (i,k) \in E\}$	$E_{12} \vee E_{13}$ $\ldots$
	$\forall (i,j),(i,k) \in E: j \neq k \Rightarrow \neg E_{ij} \vee \neg E_{ik}$	$\neg E_{12} \vee \neg E_{13}$ $\ldots$
(4.2)	$\forall j \in V: \bigvee\{E_{kj} \mid (k,j) \in E\}$	$E_{21} \vee E_{41}$ $\ldots$
	$\forall (i,j),(k,j) \in E: i \neq k \Rightarrow \neg E_{ij} \vee \neg E_{kj}$	$\neg E_{21} \vee \neg E_{41}$ $\ldots$

Dabei kodieren die Bedingungen (1) die Transitivität und (2) die Verankerung der Erreichbarkeitsrelation, (3) den Zusammenhang des Weges und (4.1) sowie (4.2) die Beschränkung des Knotenein- und -ausgrades. Diese Kodierung des HCP über dem Beispielgraphen G benötigt insgesamt 20 Aussagenvariablen und 72 Klauseln.

Eine allgemeine Schwierigkeit bei der SAT-Kodierung von Problemen besteht in der Kodierung globaler Eigenschaften der gesuchten Lösungen. Im Falle von HCP betrifft dies die Eigenschaft, daß der durch die Variablenbelegung repräsentierte Weg im Graphen zusammenhängend und geschlossen ist. Die Iwama-Miyazaki-Transformation umgeht dieses Problem durch eine Kodierung, welche auf der ausschließlichen Betrachtung zusammenhängender und geschlossener Wege beruht.

6.2.3 Die Struktur der Formeln

Da die Struktur der untersuchten Menge von CNF-Formeln erheblichen Einfluß auf die Performanz der verwendeten SAT-Verfahren haben kann, sollen nun die mithilfe der Iwama-Miyazaki-Transformation aus HCP-Instanzen erhaltenen Formeln genauer charakterisiert werden.

Wie oben beschrieben, wird durch das Transformationsverfahren eine HCP-Instanz über n Knoten in eine CNF-Formel über $(n-1)\lceil\log_2(n-1)\rceil$ Aussagenvariablen kodiert. Dabei werden in den Schritten 1, 4.1 und 4.2 mindestens $\Theta(n)$ und höchstens $\Theta(n^2)$ Klauseln der Länge $k = \lceil\log_2(n-1)\rceil$ und in den Schritten 2 und 3 jeweils $\Theta(n^3)$ Klauseln der Länge $2k$ erzeugt; genauer gilt für die Gesamtzahl c der erzeugten Klauseln: $(n-1)^3/2 < c < (n-1)^3 \cdot 3/2$.

Damit beträgt das Klausel/Variablen-Verhältnis der erzeugten Formeln $\Theta(n^2/\log n)$. Aus dieser Abschätzung folgt unmittelbar, daß die erzeugte CNF-Formel polynomielle Länge bezüglich der Problemgröße n besitzt; da ferner jede Klausel in polynomieller Zeit berechnet werden kann (wie aus dem Transformations-Algorithmus leicht ersichtlich ist), folgt insgesamt, daß die Iwama-Miyazaki-Transformation polynomielle Zeitkomplexität (bezüglich n) aufweist.

Zwischen den Variablen besteht eine ausgeprägte Abhängigkeitsstruktur; so besitzen je zwei Variablen, die demselben Knoten zugeordnet sind, wesentlich mehr gemeinsame Klauseln, als dies für Variablen verschiedener Knoten der Fall ist. Dies resultiert aus der Tatsache, daß alle Klauseln Bedingungen über Knotenpositionen kodieren und somit jeweils alle Variablen des entsprechenden Knotens enthalten. Weiterhin ist jede Variable von allen anderen abhängig, d.h. zwei beliebige Variablen besitzen immer eine gemeinsame Klausel. Diese Eigenschaft unterscheidet die HC-Formeln wesentlich von anderen Formelklassen wie beispielsweise Random-3-CNF; sie hat insbesondere zur Folge, daß die für den GSAT-Algorithmus verwendete "score-by-need"-Methode zur effizienten Bewertung von Belegungen (vergl. Abschnitt 3.6) nicht greift.

6.3 Zur Schwierigkeit von HCP-Instanzen

Wie bereits angedeutet, unterscheiden sich einzelne Teilklassen von HCP in ihrer Komplexität grundlegend. Dabei hat der Knotengrad des zugrundeliegenden Graphen einen entscheidenden Einfluß auf die Komplexität des HCP. Da in den experimentellen Untersuchungen im Rahmen dieser Arbeit das HCP in gerichteten Graphen untersucht wird, ist hier besonders der Ausgrad der Knoten von Bedeutung.

Für Graphen mit Ausgrad 1 ist das HCP mithilfe eines naheliegenden Verfahrens in linearer Zeit lösbar: Man starte bei einem beliebigen Knoten und folge jeweils der einzigen Kante, die den zuletzt besuchten Knoten verläßt. Dabei werden bereits besuchte Knoten memoriert; trifft man im Verlauf des Verfahrens erneut auf einen solchen Knoten, so ist das HCP für diesen Graphen genau dann lösbar, wenn die Menge der besuchten Knoten gleich der Menge aller Knoten ist. Da die Menge der besuchten Knoten in jedem Schritt des Verfahrens genau um ein Element erweitert wird, besitzt dieser Algorithmus lineare Zeitkomplexität.

In der nach Iwama und Miyazaki kodierten SAT-Darstellung eines solchen trivialen Problems ist die Information über den Knotengrad allerdings nicht mehr explizit enthalten; die entsprechende CNF-Formel weist keine auffälligen Unterschiede zur kodierten Form eines HCP über einem Graphen mit Ausgrad 2 auf. Mithin ist nicht a priori davon auszugehen, daß die Einfachheit eines HCP mit Ausgrad 1 auch in der SAT-kodierten Form erhalten bleibt. Vielmehr wäre durchaus denkbar, daß durch eine ungünstige Kodierung oder ein ungünstiges SAT-Verfahren der Komplexitätsvorteil für diese Teilklasse von HCP verloren geht. Damit wäre das gesamte Verfahren (also die Kombination aus Transformations- und SAT-Algorithmus) als inadäquat für HCP zu betrachten.

Während das HCP für Graphen mit Ausgrad 1 lineare Zeitkomplexität aufweist, ist es für Ausgrad 2 bereits $\mathcal{NP}$-hart. Solche Instanzen können mithilfe einer einfachen Erweiterung des oben für Ausgrad 1 beschriebenen Lösungsverfahrens zu einem Backtracking-Verfahren gelöst werden, dabei sind jedoch im ungünstigsten Falle exponentielle Laufzeiten in Kauf zu nehmen. Die aus der SAT-Transformation dieser Probleme erhaltenen CNF-Formeln unterscheiden sich kaum von den aus Graphen mit Ausgrad 1 erhaltenen. Beide Formelklassen haben in der Regel nur ein Modell, da die zugrundeliegenden Graphen in den meisten Fällen nur einen Hamiltonschen Kreis besitzen[4]; daher ist zu vermuten, daß diese Formelklassen für GenSAT-Verfahren besonders schwierig lösbar sind (vergl. [CI95]).

[4] Diese Aussage ist für Ausgrad 1 offensichtlich wahr; für Ausgrad 2 basiert sie auf Beobachtungen aus dem experimentellen Teil der Arbeit.

Kapitel 7

Verfahren zur Lösung von HCP

In diesem Kapitel werden verschiedene, im Rahmen dieser Arbeit untersuchte Verfahren zur Lösung des HCP dargestellt. Zunächst wird dabei der Einsatz von GSAT auf SAT-kodierten HCP-Instanzen diskutiert; die dabei aufgezeigten Schwierigkeiten sind der Ausgangspunkt für einige problemspezifische Modifikationen des Verfahrens. Zunächst wird gezeigt, wie der für praktische Anwendungen problematische, kubische Platzbedarf von GSAT auf den betrachteten Formeln mithilfe einer *impliziten Klauselrepräsentation* erheblich reduziert werden kann. Durch eine Erweiterung der möglichen Belegungsübergänge in den einzelnen Schritten des Verfahrens kann der GSAT-Algorithmus weiterhin an die spezielle Problemstruktur der SAT-kodierten HC-Probleme angepaßt werden. Auf diese Weise gelang man zu einem neuen, problemspezifischen Algorithmus GHC zur Lösung des HCP. Eine weitere, durch bekannte Heuristiken zur Lösung des Traveling-Salesman-Problems motivierte Modifikation dieses Verfahrens führt zum GSHC-Algorithmus, einem sehr problemspezifischen HC-Verfahren. Die hier entwickelten Verfahren sind sämtlichst als Instanzen des GenSAT-Rahmens formalisierbar und demonstrieren die sukkzessive Anpassung eines allgemeinen SAT-Verfahrens (GSAT) an die spezielle Struktur einer Problemklasse (HCP).

7.1 GSAT-Varianten auf SAT-kodierten Instanzen

Der ursprünglichen Idee folgend, das HCP in SAT-kodierter Form mithilfe eines allgemeinen SAT-Verfahrens zu lösen, liegt es nahe, die SAT-Transformation nach dem Verfahren von Iwama & Miyazaki durchzuführen und zur Lösung des transformierten Problems den bereits ausführlich behandelten GSAT-Algorithmus zu verwenden. Aufgrund der Ergebnisse verschiedener Untersuchungen zum GSAT-Verfahren und seiner Varianten sowie der besonderen Struktur der betrachteten CNF-Formeln bietet es sich an, verschiedene Modifikationen des ursprünglichen Verfahrens einzusetzen; besonders vielversprechend erscheinen hierbei die „Random-Walk"-Variante sowie die Verwendung adaptiver Klauselgewichte (vergl. Abschnitte 3.3.2, 3.3.3).

Bei dieser Vorgehensweise ist zu bedenken, daß die Formeln aus der SAT-Kodierung nach Iwama & Miyazaki für ein HCP über n Knoten aus $\Theta(n^3)$ Klauseln bestehen.Verschiedene Untersuchungen kommen zwar zu dem Schluß, daß ein hohes Klausel/Variablen-Verhältnis sich eher günstig auf die Performanz des GSAT-Algorithmus auswirkt (siehe [KD95, CI95]), dennoch treten in der Praxis Probleme auf: Zum einen kann es bei größeren HCP-Instanzen schnell zu Platzproblemen bei der Speicherung der Formel kommen, zum anderen wächst mit steigender Formelgröße auch in der Regel der Aufwand für jeden einzelnen Schritt des Verfahrens. Dies wird unter anderem dadurch verursacht, daß mit steigendem Klausel/Variablen-Verhältnis der Aufwand zur Aktualisierung der Bewertungen anwächst.

Insbesondere greift im Falle der SAT-kodierten HC-Instanzen die in Abschnitt 3.6 vorgestellte "score-by-need"-Technik zur effizienten Aktualisierung der Bewertungen (die für Random-3-CNF-Formeln mit festem Klausel/Variablen-Verhältnis nur konstanten Zeitaufwand benötigt) nicht. Aufgrund der Variablenabhängigkeiten in den entsprechenden Formeln (vergl. Abschnitt 6.2.3) müssen nach jedem Schritt die Bewertungen für alle Variablenübergänge aktualisiert werde, sodaß insgesamt pro Schritt ein kubischer Zeitaufwand (bezüglich der Variablenanzahl) ensteht.

7.1.1 Verringerung des Platzbedarfs

Das bereits angedeutete Problem des kubischen Platzbedarfs der SAT-kodierten HCP-Instanzen kann auf verschiedene Weise behoben werden. Eine interessante Möglichkeit besteht in der *impliziten Kodierung* der Klauseln (diese wurde — basierend auf einer in der GSAT-Implementierung von Selman & Kautz realisierten Vorgehensweise für SAT-kodierte Graphfärbungsprobleme — im Rahmen dieser Arbeit entwickelt). Dabei wird anstelle der *expliziten Erzeugung* einer CNF-Formel im Zuge der SAT-Transformation als Vorverarbeitung der ursprünglichen Probleminstanz (hier HCP über einem gegeben Graphen) jede Klausel genau dann generiert, wenn sie innerhalb des SAT-Verfahrens tatsächlich benötigt wird (Kodierung nach Bedarf, "encoding-by-need"). Dies entspricht einer Modifikation der Bewertungsfunktion für Variablenbelegungen, die für jede Belegung die Ableitung ihrer Bewertung direkt aus der unkodierten Problemstellung ermöglicht.

Im konkreten Fall des HCP kann dies am einfachsten dadurch realisiert werden, daß zur Bewertung einer Variablenbelegung B eine modifizierte Form der Iwama-Miyazaki-Transformation des Graphen durchgeführt wird, wobei statt der CNF-Formel F direkt deren Wert $Val_B(F)$ unter der Belegung B berechnet wird. Dieser Wert kann sukzessiv berechnet werden, sodaß jede erzeugte Klausel unmittelbar ausgewertet und anschließend sofort „vergessen" wird. Auf analoge Weise kann auch die Menge der in aktuell unerfüllten Klauseln auftretenden Variablen, die zur Realisierung der Random-Walk-Variante benötigt wird, berechnet werden.

Diese Methode der *impliziten Klauselkodierung* benötigt keinen Platz zur Speicherung von Klauseln; außerdem entfällt die SAT-Kodierung der ursprünglichen Probleminstanz als Vorverarbeitungsschritt (genaugenommen wird diese in die Berechung der Bewertungsfunktion verlagert). Allerdings muß nun die unkodierte Probleminstanz gespeichert werden, was im vorliegenden Fall des HCP einen Platzaufwand von $O(n^2)$ verursacht.[1]

Ein offensichtlicher Nachteil der impliziten Klauselkodierung liegt in der Tatsache, daß diese eine problemspezifische Modifikation des allgemeinen GSAT-Verfahrens darstellt. Zwar kann für jedes SAT-kodierte Problem auf kanonische Weise eine Bewertungsfunktion aus dem Kodierungsverfahren konstruiert werden; allerdings ist die Zeitkomplexität des so gewonnenen Vefahrens oft der des expliziten GSAT-Verfahrens auf dem SAT-kodierten Problem deutlich unterlegen. Im vorliegenden Fall (HCP mit Iwama-Miyazaki-Kodierung) kann die implizite Klauselkodierung ohne Performanzeinbußen eingesetzt werden; dies ist möglich, da hier durch die Verwendung der "score-by-need"-Technik keine wesentliche Effizienzsteigerung bei der Bewertung von Variablenbelegungen zu erreichen ist (vergl. Abschnitt 3.6). Im Gegenteil ist es bei Verwendung der impliziten Klauselkodierung sogar möglich, die Performanz des SAT-Verfahrens zu steigern, indem bei der Berechnung von Übergangsbewertungen Aspekte der Problemstruktur ausgenutzt werden, die nach der SAT-Kodierung nicht mehr explizit erkennbar sind.[2]

Für GSAT-Varianten, welche mit klauselbezogenen Informationen arbeiten, kann durch implizite Klauselkodierung der Platzbedarf in der Regel höchstens um einen konstanten Faktor reduziert werden. Sollen beispielsweise adaptive Klauselgewichte verwendet werden, muß auch bei impliziter Klauselkodierung für jede Klausel das aktuelle Gewicht gespeichert werden.

Eine weitere Möglichkeit zur Reduzierung des Platzbedarfs SAT-kodierter HCP-Instanzen besteht in der Verwendung alternativer Darstellungsformen für die SAT-Formel. Beispielsweise kann durch den Übergang zu CDNF-Formeln die zur Kodierung des HCP über einem Graphen mit n Knoten benötigte Klauselzahl auf $\Theta(n^2 \log n)$ reduziert werden (auf eine ausführlichere Darstellung muß in diesem Rahmen aus Platzgründen verzichtet werden).

7.1.2 RBR-Erweiterung der CNF-Formeln

Etliche Untersuchungen anderer Autoren kommen zu dem Schluß, daß sich die Performanz von SAT-Algorithmen mit wachsendem Klausel/Variablen-Verhältnis der Probleminstanzen verbessert. Für vollständige Verfahren ist diese Annahme plausibel (vergl. Abschnitt 4.1.1); für unvollständige lokale Suchverfahren wie GSAT stützt sie sich im wesentlichen auf experimentelle Resultate (siehe [CI95, KD95]). Ausgehend von dieser Beobachtung ist es naheliegend zu versuchen, eine gegebene SAT-Instanzen durch das Hinzufügen von Klauseln leichter lösbar zu machen. Dabei muß sichergestellt werden, daß die erweiterte Formel äquivalent zur Ausgangsformel ist; bei der Erweiterung dürfen also insbesondere keine Modelle

[1] Für konstanten Knotengrad reduziert sich der Platzbedarf bei Adjazenzlistenrepräsentation des Graphen auf $O(n)$.

[2] Auf Details kann in diesem Zusammenhang aus Zeit- und Platzgründen nicht weiter eingegangen werden; die Auswirkungen dieser Verbesserungen sind aus den in Abschnitt 8.3 dargestellten Ergebnissen von Laufzeitmessungen ersichtlich (siehe Tabelle auf Seite 85; vergl. Werte für GSAT und GHC).

der ursprünglichen Formel „verlorengehen".

Für stark strukturierte Probleme, die sich als besonders schwierig für GenSAT-Verfahren erwiesen haben (insbesondere Probleme mit hierarchischer Struktur, wie das Graph-Kreuzworträtsel-Problem, vergl. [Kon94]), konnte gezeigt werden, daß durch Hinzufügen von Resolventen aus Klauseln der ursprünglichen Formel eine deutliche Verbesserung der Performanz von GSAT erreicht wird (siehe [KD95]). [3] Das dabei verwendete Verfahren der *eingeschränkten Resolventen-Erweiterung* wird im folgenden näher beschrieben. Dazu wird zunächst der Begriff der Resolvente formal definiert:

Definition 7.1 (Resolvente)

Sind c_1, c_2 Klauseln einer CNF-Formel in Mengendarstellung und gilt für eine Aussagenvariable x: $x \in c_1$ und $\neg x \in c_2$, so ist die Resolvente der Klauseln c_1, c_2 *definiert als*

$$R(c_1, c_2) := c_1 \backslash \{x\} \cup c_2 \backslash \{\neg x\}$$

Beispiel: Als Resolvente der Klauseln $c_1 = \{x, \neg y, \neg z\}$ und $c_2 = \{w, x, y\}$ ergibt sich die Klausel $R(c_1, c_2) = \{w, x, \neg z\}$.

Bei der eingeschränkten Resolventen-Erweiterung einer CNF-Formel werden zu dieser alle Resolventen von Klauseln der ursprünglichen Formel hinzugenommen, deren Länge eine gegebene Schranke b nicht überschreitet.

Definition 7.2 (RBR$_k$-Erweiterung)

Für eine CNF-Formel F mit m Klauseln (in Mengenschreibweise) ist die eingeschränkte Resolventen-Erweiterung mit Schranke k (RBR$_k$-Erweiterung) *definiert als*

$$RBR_k(F) := F \cup \{r \mid r = R(c_i, c_j) \text{ mit } 1 \leq i < j \leq m \text{ und } |r| \leq k\}\}$$

Beispiel: Die RBR$_3$-Erweiterung der Formel $F = \{\{x, y, z\}, \{w, \neg x, y\}, \{x, \neg y\}\}$ ist gegeben durch $RBR_k(F) = F \cup \{\{w, y, z\}, \{x, z\}\}$; für die RBR$_2$-Erweiterung von gilt hingegen: $RBR_2(F) = F \cup \{\{x, z\}\}$.

Die RBR$_k$-Erweiterung ist eine Äquivalenzumformung für beliebige Schranken k, d.h. für beliebige Variablenbelegungen B gilt: B ist genau dann Modell von F, wenn B auch Modell

[3] Für unstrukturierte Formelklassen wie Random-3-SAT scheint dies nicht zu gelten (vergl.[CI95]).

von $RBR_k(F)$ ist. Sie kann für eine Formel mit k Klauseln mithilfe eines sequentiellen, deterministischen Verfahrens in $O(m^2)$ Schritten auf naheliegende Weise berechnet werden.[4]

Aufgrund der ausgeprägten Struktur der SAT-kodierten HC-Probleme wurde das Verfahren der RBR-Erweiterung auch bei den experimentellen Untersuchungen im Rahmen dieser Arbeit eingesetzt. Wie in [KD95] wurde die RBR-Erweiterung der Formel dabei als Vorverarbeitungsschritt zu einem GenSAT-Verfahren realisiert. Im Gegensatz zu den in [KD95] betrachteten Formeln beinhalten die kodierten HCP-Instanzen Klauseln unterschiedlicher Länge. Daher wurden für HCP-Instanzen mit k Variablen pro Knoten, welche aus Klauseln der Länge k bzw. $2k$ bestehen (vergl. Abschnitt 6.2.3), in der vorliegenden Untersuchung sowohl die RBR_k- als auch die RBR_{2k}-Erweiterung betrachtet.

Durch die Abhängigkeit von der Anzahl der Klauseln der gegebenen Formel kann der quadratische Zeitaufwand für die RBR-Erweiterung im Falle der SAT-kodierten HCP-Instanzen keinesfalls vernachlässigt werden. Weiterhin ist zu bedenken, daß durch jede Erhöhung des Klausel/Variablen-Verhältnisses der Zeitaufwand für die von GenSAT-Implementierungen in jedem Schritt durchgeführte Aktualisierung der Bewertung von Variablenbelegungen ansteigt (vergl.[CI95]). Somit verursacht die RBR-Erweiterung der ursprünglichen CNF-Formel zunächst eine Geschwindigkeitseinbuße für das nachgeschaltete GenSAT-Verfahren.

7.2 Das GHC-Verfahren

Aufgrund der besonderen Abhängigkeitsbeziehungen innerhalb der Gruppe von Variablen, die jeweils einem Knoten im Graphen zugeordnet sind (vergl. Abschnitt 6.2.3), bietet es sich an, eine problemspezifische Modifikation des GSAT-Algorithmus mit impliziter Klauselkodierung vorzunehmen. Zunächst sei daran erinnert, daß die Belegung der einem bestimmten Knoten zugeordneten Variablen dessen Position innerhalb der aktuellen Permutation der Knotenmenge entspricht (vergl. Abschnitt 6.2.1). Unter dieser Betrachtungsweise entsprechen die in einem GSAT-Schritt erlaubten Variablenübergänge genau den Übergängen zwischen Knotenpositionen, deren Binärdarstellung sich gerade in einer Ziffer unterscheiden.

Beispiel 7.1 Belegungsübergänge unter GSAT

Man betrachte das Beispiel aus Abbildung 6.1; sind die dem Knoten 2 zugeordneten Variablen mit $v_{2,2} = \top$, $v_{2,1} = \bot$ belegt, so sind für diese genau folgende Belegungsübergänge unter GSAT möglich:

$$\begin{array}{llr} & 2 \simeq (1,0) \rightarrow (0,0) \simeq 0 & [\,\mathsf{flip}(v_{2,2})\,] \\ \text{und} & 2 \simeq (1,0) \rightarrow (1,1) \simeq 3 & [\,\mathsf{flip}(v_{2,1})\,] \end{array}$$

Damit ist unter GSAT beispielsweise folgender direkte Übergang zwischen Knotenbelegun-

[4] Dazu wird im wesentlichen für je zwei verschiedene Klauseln überprüft, ob sie eine Resolvente zulässiger Länge besitzen (siehe Definition).

gen möglich (zur Schreibweise vergl. Beispiel in Abschnitt 6.2.1):

$$(0,2,1) \to (0,3,1) \qquad [\,\mathsf{flip}(v_{2,2})\,]$$

Dabei kodiert die Knotenbelegung (0,3,1) keine gültige Knotenpermutation, da der Knoten 2 mit einer ungültigen Knotenposition belegt ist.

Es bietet sich nun an, die genannte Einschränkung fallenzulassen und nunmehr Übergänge zwischen beliebigen Positionen eines Knotens als zulässig zu betrachten. Dies wird durch eine Erweiterung der Nachbarschaftsrelation auf den Variablenbelegungen der betrachteten Formel erreicht, indem nun direkte Übergänge zwischen Belegungen zugelassen werden, die sich im Wert beliebig vieler, demselben Knoten zugeordneter Variablen unterscheiden. Diese Übergänge werden somit durch Multiflips realisiert (vergl. Abschnitt 5.2).

Durch Verwendung einer impliziten Klauselkodierung ergibt sich, wie bereits im Falle des allgemeinen GSAT-Verfahrens erläutert wurde (vergl. Abschnitt 7.1.1), ein deutlich reduzierter Platzbedarf des zu realisierenden Verfahrens. Weiterhin bietet es sich an, die Auswahl der Ausgangsbelegung und die Belegungsübergänge so zu beschränken, daß *immer* gültige Knotenpositionen kodiert werden (vergl. Abschnitt 6.2.1). Diese Einschränkung ist sehr einfach zu realisieren und ermöglicht für eine HCP-Instanz mit n Knoten die Einsparung von bis zu $\Theta(n^2)$ Klauseln, da Schritt 1 der Iwama-Miyazaki-Kodierung entfallen kann. Sowohl für explizite als auch für implizite Klauselkodierung ergibt sich daher im allgemeinen Fall eine Verringerung des Aufwands für die Neuberechnung bzw. die Aktualisierung der Bewertung von Belegungsübergängen; im günstigsten Fall kann so der Zeitbedarf für diese Schritte halbiert werden.[5]

Weiterhin wird durch die Beschränkung auf gültige Knotenpositionen im allgemeinen eine Verkleinerung des Suchraums bewirkt. Diese ist allerdings als nahezu unbedeutend einzuschätzen, da sich im günstigsten Falle (s.o.) höchstens eine Halbierung der Suchraumgröße ergibt. A priori ist unklar, ob durch die fragliche Beschränkung tatsächlich eine Verbesserung der Performanz erreicht werden kann; denkbar wäre auch, daß durch diese Verringerung der Freiheitsgrade die lokale Suche erschwert wird (da beispielsweise bestimmte Übergänge, welche ungültige Knotenpositionen als Zwischenschritte enthalten, nicht mehr möglich sind).

Auf der Grundlage dieser Überlegungen führen wir eine neue, problemspezifische GenSAT-Variante GHC (Greedy Hamilton-Circuit) ein, welche sich von GSAT in folgenden Punkten unterscheidet:

- Verwendung der impliziten Klauselkodierung;
- Problemspezifische Erweiterung der Nachbarschaftsrelation (Multiflips);
- Einschränkung auf Belegungen, die ausschließlich gültige Knotenpositionen kodieren.

[5] Dieser Fall liegt vor wenn gilt: $n = 2^j + 2$ für beliebiges $j \in \mathbb{N}$; dies bedeutet gerade, daß fast die Hälfte aller kodierbaren Knotenpositionen ungültig sind.

Beispiel 7.2 Belegungsübergänge unter GHC

Man betrachte das Beispiel aus Abbildung 6.1; folgender Belegungsübergang für die dem Knoten 2 zugeordneten Variablen $v_{2,2}, v_{2,1}$ ist unter GHC möglich, nicht jedoch unter GSAT:

$$2 \simeq (1,0) \rightarrow (0,1) \simeq 1 \qquad [\, vv_B(2) := 1\,]$$

Damit ist unter GHC folgender direkte Übergang zwischen Knotenbelegungen möglich, der unter GSAT nur über einen Zwischenschritt realisiert werden kann:

$$(0,2,1) \rightarrow (0,1,1) \qquad [\, vv_B(2) := 1\,]$$

Der folgende Übergang hingegen ist zwar unter GSAT möglich, nicht jedoch unter GHC, da er zu einer ungültigen Position für Knoten 2 führt:

$$(0,2,1) \rightarrow (0,3,1) \qquad [\, \mathsf{flip}(v_{2,2})\,]$$

Um das GHC-Verfahren formal als GenSAT-Algorithmus beschreiben zu können, sei zunächst an die Definition der Knotenindexmenge N und der Knotenbelegung vv_B aus Abschnitt 6.2.1 erinnert. Darauf aufbauend kann nun die erweiterte Nachbarschaftsrelation zwischen Variablenbelegungen folgendermaßen definiert werden:

$$\mathit{Neighbour}(B,B') :\Leftrightarrow \exists! i \in N : [vv_B(i) \neq vv_{B'}(i) \wedge \forall k \in N \backslash \{j\} : vv_B(k) = vv_{B'}(k)]$$

Zur Formulierung des GHC-Verfahrens muß nun zunächst die für GSAT verwendete Bewertungsfunktion für Variablenübergänge (vergl. Abschnitt 2.3) auf beliebige Belegungsübergänge (Multiflips) verallgemeinert werden. Dies wird durch folgende Definition geleistet:

Definition 7.3 (Bewertung von Belegungsübergängen)

Seien F eine CNF-Formel mit m Klauseln in Mengenschreibweise und B, B' Variablenbelegungen (die sich in der Belegung beliebig vieler Variablen unterscheiden). Die Bewertung des Belegungsübergangs *$B \rightarrow B'$ ist dann definiert als*

$$score_F(B,B') := N_{unsat}(B,F) - N_{unsat}(B',F)$$

Wie in Abschnitt 7.1 bezeichnet dabei $N_{unsat}(B,F)$ die Anzahl der unter B unerfüllten Klauseln von F.

Obwohl die Definition für beliebige CNF-Formeln F formuliert wurde, werden im folgenden speziell die aus der Iwama-Miyazaki-Kodierung einer HCP-Instanz G erhaltenen Formel

F_G betrachtet. Die nächste Definition sagt aus, daß nur solche Variablenbelegungen als gültig betrachtet werden sollen, die jedem Knoten im Graphen G einen gültigen Knotenwert zuordnen:

Definition 7.4 (Gültige Belegungen)

Sei F_G die aus der Iwama-Miyazaki-Kodierung der HCP-Instanz G erhaltene Formel. Dann ist die Menge der gültigen Variablenbelegungen *zu dieser Formel definiert durch:*

$$\mathit{ValidAssign}(F_G) := \{B \in \mathit{Assign}(F_G) \mid \forall i \in N : vv_B(i) < n-1\}$$

Nunmehr sind alle Voraussetzungen erfüllt, um den GHC-Algorithmus (ohne Random Walk) als GenSAT-Instanz zu formulieren; dies geschieht wie üblich durch Angabe der entsprechenden init- und select-Funktion:

```
procedure init(F_G)
  B := rndElement(ValidAssign(F_G));
  return (B);
  end init;
```

```
procedure select(F_G, B, j)
  N_B := {B' | Neighbour(B, B')};
  maxScore := max{score_{F_G}(B, B') | B' ∈ N_B};
  B' := rndElement({B' ∈ N_B | score_{F_G}(B, B') = maxScore});
  return (B');
  end select;
```

Insgesamt wird durch die Erweiterung der Nachbarschaftsrelation zunächst der Zeitaufwand der select-Funktion erhöht. Für eine Probleminstanz mit n Knoten und k Variablen pro Knoten (im Beispiel gilt $n = 4$, $k = 2$) vergrößert sich die Anzahl der möglichen Folgebelegungen zu einer beliebig gegegeben Belegung im ungünstigsten Fall von $(n-1) \cdot k \approx n \log_2 n$ für GSAT auf $(n-1) \cdot 2^k \approx (n-1)^2$ für GHC. Die in Abschnitt 8.3 dargestellten Leistungsmessungen zeigen jedoch, daß für die untersuchten Problemgrößen (bis $n = 50$ Knoten) bei vergleichbarem Implementierungsaufwand GHC ca. 10–40 mal mehr Schritte pro Sekunde ausführt als GSAT (siehe Tabelle auf Seite 85). Dies erklärt sich in erster Linie durch die in Abschnitt 7.1.1 erwähnten Optimierungen der impliziten Klauselkodierung (Ausnutzung von speziellen Problemstrukturen). Weiterhin kann die Berechnung oder Aktualisierung der Übergangsbewertungen leicht und effizient parallelisiert werden (vergl. Abschnitt 5.2), sodaß diese Erweiterung der Nachbarschaftsrelation nicht mit einer Erhöhung des parallelen Zeitaufwands verbunden ist.

Auch für dieses Verfahren kann die "Random-Walk"-Variante realisiert werden. Für einen "Random-Walk"-Schritt wird dabei ein Knoten zufällig ausgewählt, dem eine Variable zugeordnet ist, die in einer aktuell unerfüllten Klausel auftritt. Für diesen Knoten wird nun (gleichverteilt) eine beliebige, gültige Knotenposition (ungleich der alten) ausgewählt; dies entspricht einem Multiflip der diesem Knoten zugeordneten Variablen. Diese Vorgehensweise entspricht der "Random-Walk"-Strategie bei GSAT, übertragen auf die entsprechend erweiterte Nachbarschaftrelation. Wie bei GSAT wird im so erweiterten GHC-Verfahren jeweils mit einer festen Wahrscheinlichkeit walkProb ein "Random-Walk"-Schritt anstelle eines Standardschrittes des Verfahrens ausgeführt. Die Implementierung dieser Variante ist, ebenso wie im Falle von GSAT mit impliziter Klauselkodierung, nahezu ohne zusätzlichen Aufwand möglich, indem für jeden Knoten die Anzahl der unerfüllten Klauseln, in denen ihm zugeordnete Variablen auftreten, mitgeführt wird. Auf die Formalisierung des GHC-Verfahrens mit Random-Walk soll an dieser Stelle verzichtet werden; die Vorgehensweise ist hierbei im wesentlichen dieselbe wie für GSAT mit Random-Walk.

Abschließend sei angemerkt, daß die der Erweiterung der Nachbarschaftsrelation zugrundelegende Idee einer problemspezifischen Partitionierung der Variablenmenge prinzipiell auf den Fall allgemeiner SAT-Instanzen übertragbar ist. Konkret könnte dabei die Partitionierung auf der Grundlage einer Analyse der Abhängigkeitsbeziehungen zwischen den Variablen der betrachteten Formel erfolgen (vergl. Abschnitt 6.2.3).

7.3 Das GSHC-Verfahren

Der im vorherigen Abschnitt dargestellte GHC-Algorithmus wurde als problemspezifische Variante des allgemeinen GSAT-Verfahrens eingeführt. Um den Einfluß derartiger spezifischer Anpassungen eines allgemeinen Verfahrens auf dessen Performanz genauer untersuchen zu können, wird in diesem Abschnitt das GSHC-Verfahren (Greedy Swap Hamilton Circuit) eingeführt. Dieses entstand in Anlehnung an bekannte heuristische Verfahren für das Problem des Handlungsreisenden (TSP), welches als Verallgemeinerung des HCP aufgefaßt werden kann (vergl. [Rei94]).

Der GSHC-Algorithmus läßt sich als Variante des GHC-Verfahrens charakterisieren, in der die Belegungen der Knotenvariablen zu *jedem* Zeitpunkt gültigen Permutationen der Knoten (und damit geschlossenen Wege) im Graphen entsprechen. Dies wird erreicht, indem zu Beginn eines Versuchs die Knotenwerte zunächst auf der Grundlage einer zufällig ausgewählten Permutation der Knoten initialisiert werden. In jedem Schritt des Verfahrens werden nun jeweils zwei Knotenwerte vertauscht, was einer Vertauschung der Knotenposition in der aktuellen Permutation entspricht; eine solche Vertauschung zweier Knotenwerte wird als *Swap* bezeichnet. Die Knotenvariablen, deren Werte in einem Schritt zu vertauschen sind, werden (analog zur Vorgehensweise bei GHC) anhand einer Bewertung ausgewählt, die sich aus der Änderung in der Anzahl der erfüllten Klauseln der SAT-kodierten Probleminstanz ergibt. Hierbei sind die Bedingungen aus den Schritten 1 und 2 der Iwama-Miyazaki-Kodierung immer erfüllt (erstere durch die Beschränkung des Wertebereichs der Knotenvariablen wie bei

GHC, zweitere aufgrund der eben beschriebenen Vorgehensweise) und können deshalb bei der Bewertung der Variablenvertauschungen vernachlässigt werden. Somit werden lediglich die Bedingungen aus den Schritten 3 und 4 der Iwama-Miyazaki-Kodierung berücksichtig; diese stellen sicher, daß der durch die Knotenwerte kodierte Rundweg keine im Graphen nicht vorhandenen Kanten beinhaltet.

Beispiel 7.3 Belegungsübergänge unter GSHC

Man betrachte das Beispiel aus Abbildung 6.1. Folgender direkte Belegungsübergang für die Knotenwerte ist unter GSHC möglich, nicht jedoch unter GHC oder GSAT:

$$(0,2,1) \rightarrow (0,1,2) \qquad [\,\mathsf{swap}(vv_B(2), vv_B(3))\,]$$

Dieser Übergang kann unter GHC oder GSAT nur über Zwischenschritte realisiert werden.

Die Formalisierung des GSHC-Verfahrens erfolgt im wesentlichen analog zu der für GHC demonstrierten Vorgehensweise. Zur Beschreibung des GSHC-Algorithmus als GenSAT-Instanz können die gleichen init- und select-Funktionen wie für GHC verwendet werden, lediglich die Menge der gültigen Belegungen und die Nachbarschaftsrelation müssen auf folgende Weise umdefiniert werden:

$$\begin{aligned} \mathit{ValidAssign}(F_G) := \{B \in \mathit{Assign}(F_G) \mid\ & \forall i \in N : vv_B(i) < n-1 \\ & \wedge \forall (i,j) \in N^2 : i \neq j \rightarrow vv_B(i) \neq vv_B(j)\} \end{aligned}$$

$$\begin{aligned} \mathit{Neighbour}(B,B') :\Leftrightarrow \exists!(i,j) \in N^2 : [\ & i<j\ \wedge vv_B(i) = vv_{B'}(j) \wedge vv_B(j) = vv_{B'}(i) \\ & \wedge \forall k \in N\backslash\{i,j\} : vv_B(k) = vv_{B'}(k)\ ] \end{aligned}$$

Durch diese Definitionen wird sichergestellt, daß genau diejenigen Belegungen, welche gültige Knotenpermutationen kodieren, als gültig angesehen werden und weiterhin genau diejenigen Belegungen als direkte Nachbarn gelten, welche durch Vertauschen zweier Knotenwerte ineinander überführbar sind.

Durch die Einschränkung der Knotenbelegungen ergibt sich für GSHC verglichen mit GHC eine deutliche Verkleinerung des Suchraums: Für einen Graphen mit n Knoten besteht der Suchraum unter GHC aus $(n-1)^{n-1}$ Knotenbelegungen, unter GSHC spielt sich die lokale Suche hingegen in einem Suchraum von $(n-1)! \approx ((n-1)/e)^{n-1} \cdot \sqrt{2\pi(n-1)}$ Knotenpermutationen ab (Abschätzung nach der Stirlingschen Formel, siehe [BS79]).[6] Die Größe der direkten Nachbarschaft einer Belegung beträgt $n-1$, dies entspricht im wesentlichen der Situation unter GHC. Wie dort, kann auch bei GSHC durch Optimierung der impliziten Klauselkodierung unter Ausnutzung spezieller Problemstrukturen die Leistung des Verfahrens (Schritte/Sekunde) enorm gesteigert werden. Die in Abschnitt 8.3 dargestellten Leistungsmessungen zeigen, daß sich mit wachsender Problemgröße ein deutlicher

[6] Genauer gilt: $\lim_{n\rightarrow\infty}(n-1)!/(n-1)^{n-1} \approx \lim_{n\rightarrow\infty} \sqrt{2\pi(n-1)}/e^{n-1} = 0$

Leistungsvorteil von GSHC gegenüber GHC ausprägt: Während für $n = 10$ Knoten noch keine wesentlichen Leistungsunterschiede gemessen wurden, ist für $n = 50$ bereits ein Faktor 2 zugunsten von GSHC festzustellen. Ähnlich wie bei GHC kann auch im Falle von GSHC die Leistung durch eine Parallelisierung der Berechnung oder Aktualisierung von Übergangsbewertungen auf einfache Weise weiter gesteigert werden.

Analog zum Übergang von GSAT zu GHC ist für das GSHC-Verfahren a priori unklar, ob aus der weiterreichenden Beschränkung der Freiheitsgrade eine Verbesserug der Performanz des lokalen Suchverfahrens folgt. Durch die deutliche Verkleinerung des Suchraums ist jedoch zu erwarten, daß Lösungen von GSHC leichter gefunden werden als von GHC oder GSAT.

Der GSHC-Algorithmus kann durch die “Random-Walk”-Strategie erweitert werden; jeder “Random-Walk”-Schritt ist dabei eine bewertungsunabhängige Vertauschung, wobei mindestens eine der beteiligten Variablen in einer aktuell unerfüllten Klausel auftritt. Analog zur Vorgehensweise bei GSAT und GHC werden diese Schritte mit einer festen Wahrscheinlichkeit walkProb anstelle der Standardschritte des Verfahrens ausgeführt.

Auch GSHC läßt sich als Instanz des GenSAT-Rahmens und damit als allgemeines SAT-Verfahren formulieren. Im Gegensatz zu GHC handelt es sich hierbei jedoch um ein sehr direkt auf das HCP zugeschnittenes Verfahren; es ist daher sehr fraglich, ob die zentrale algorithmischen Idee (Vertauschung von Knoten- bzw. Variablenwerten) auch im Falle des allgemeinen SAT-Problems vorteilhaft eingesetzt werden kann.

Kapitel 8

Experimentelle Untersuchung an HCP-Instanzen

In diesem Kapitel wird die experimentelle Untersuchung der zuvor beschriebenen Verfahren an Instanzen des Hamilton-Kreis-Problems (HCP) dargestellt. Dabei werden zunächst Motivation und Rahmenbedingungen der Versuchsserien erläutert. Es folgt eine Beschreibung der verwendeten Testinstanzen und der Verfahren zu ihrer Erzeugung; anschließend wird die Durchführung der einzelnen Versuchsreihen beschrieben.

Im Mittelpunkt des Kapitels steht die ausführliche Darstellung und Diskussion der Versuchsergebnisse. Dabei werden zunächst verschiedene in Kapitel 7 vorgestellte Erweiterungen GSAT-ähnlicher Verfahren (RBR-Erweiterung, adaptive Klauselgewichte und Random Walk) betrachtet. Anschließend wird die Abhängigkeit zwischen der Performanz der untersuchten Verfahren (GSAT, GHC und GSHC) und den Einstellungen der Parameter walkProb (Walk-Wahrscheinlichkeit) und maxSteps (obere Schranke für die Anzahl der Schritte pro Versuch) ausführlich analysiert. Schließlich wird dargestellt, wie anhand der hier erzielten Ergebnisse Prognosen über obere Laufzeitschranken für die verwendeten Verfahren abgeleitet werden können und inwiefern auf diese Weise die Größe praktisch lösbarer Probleminstanzen begrenzt ist. Das Kapitel schließt mit einer Zusammenfassung der wichtigsten Resultate.

Die meisten Versuchsergebnisse sind in Form von Diagrammen dargestellt, diese finden sich in 9.6 am Ende der Arbeit.

8.1 Grundlagen und Rahmenbedingungen

Aufgrund des immensen Umfangs der Versuchsreihen, die im Rahmen der hier beschriebenen Untersuchung durchgeführt wurden, ergab sich eine Reihe von Problemen. Zum einen wurden aufgrund verschiedener Faktoren (s.u.) extrem große Rechenkapazitäten benötigt;

zum anderen mußten aufgrund von Anzahl und Struktur der verschiedenen Versuchsreihen die Erzeugung von Probleminstanzen, die Durchführung von Versuchsreihen und die Auswertung der Ergebnisse weitgehend automatisiert werden. Zu diesem Zwecke wurde neben den im letzten Kapitel beschriebenen Verfahren eine Reihe von Programmen realisiert, die zusammengenommen eine leistungsfähige Experimentierumgebung bilden.[1]

Um diese Umgebung auf verschiedenen Rechnerplattformen nutzen zu können, wurde für die Implementierung sämtlicher Programme die Programmiersprache C unter Beachtung des ANSI-Standards gewählt. Aufgrund des prototypischen Charakters dieser Untersuchung und des Zeitrahmens wurden dabei zwar die algorithmischen Ansätze, in der Regel jedoch nicht die C-Implementierungen unter Laufzeit- oder Speicherplatzgesichtspunkten optimiert.

Um angesichts des engen Zeitrahmens eine maximale Anzahl von Versuchsreihen durchführen zu können, wurden die Experimente auf insgesamt bis zu 7 Rechnern parallel ausgeführt. Dabei wurden Vorbereitung und Auswertung der Versuchsreihen sowie die kleinsten Probleme auf einem PC (antonio) unter MS-DOS ausgeführt. Für mittlere Problemgrößen wurden Workstations von Hewlett-Packard (itihp3, itihp7) und Sun (sun10, sun16) unter Unix eingesetzt; die aufwendigsten Experimente wurden auf DEC-Alpha-Workstations (axp605, axp603) unter VMS durchgeführt.

Name	*Rechnertyp*	*Modell*	*Arbeitsspeicher*
axp605	Alpha Server 2100	5/250 (2 CPU)	512 MB
axp603	Alpha Server 2100	4/200 (3 CPU)	448 MB
sun10	Sparc Station	10	128 MB
sun16	Sparc Classic		32 MB
itihp7	HP 9000	712/60	96 MB
itihp3	HP 9000	400t	12 MB
antonio	Highscreen PC	486/100	4 MB

Die meisten der eingesetzten Programme wurden so konzipiert, daß alternativ eine Bearbeitung einzelner Probleminstanzen bei interaktiver Eingabe der Parameter (vor allem zu Testzwecken), oder eine vollständig dateigesteuerte Bearbeitung einer größeren Menge von Testinstanzen (zur Durchführung von Versuchsreihen) möglich ist.

Der Standard-Zufallszahlengenerator der Programmiersprache C ist in verschiedenen Entwicklungsumgebungen unterschiedlich realisiert, wobei die Qualität der so erzeugten Pseudo-Zufallszahlen stark variiert. Um ein möglichst identisches Verhalten auf allen Rechnerplattformen zu gewährleisten, wurde für Realisierung der im Rahmen dieser Arbeit eingesetzten probabilistischen Verfahren in allen Fällen ein aus der GSAT-Implementierung von Selman und Kautz entnommener Zufallszahlengenerator verwendet.

[1] In diesem Zusammenhang wurde auch das GSAT-Verfahren neu implementiert; dies wurde nötig, da die verfügbare GSAT-Implementierung von Selman & Kautz sich als relativ unübersichtlich und schwer portabel erwiesen hat. Durch die Reimplementierung konnte ferner eine weitestgehende Analogie zu den übrigen Verfahren bezüglich Programmstruktur und Performanzmessung realisiert werden.

8.2 Erzeugung der Testinstanzen

Im Rahmen der experimentellen Untersuchung wurden die drei im letzten Kapitel beschriebenen Verfahren (GSAT,GHC,GSHC) in der durch "Random-Walk" erweiterten Variante eingesetzt. Da es sich hierbei um unvollständige Verfahren handelt, wurden als Testinstanzen ausschließlich Graphen verwendet, die mindestens einen Hamilonschen Kreis enthalten. Es wurden zwei Klassen von probabilistisch erzeugten HCP-Instanzen mit jeweils unterschiedlichen Knotenzahlen (entspricht der Problemgröße) untersucht:

- $\mathrm{HC}_1(n)$: gerichtete Graphen mit n Knoten und einem Knotenausgrad von 1.
- $\mathrm{HC}_2(n)$: gerichtete Graphen mit n Knoten und einem Knotenausgrad von 2.

Diese Klassen von Instanzen haben sehr unterschiedliche Eigenschaften (vergl. Abschnitt 6.3): Die Klasse HC_1 enthält für jede Problemgröße n nur einen Typ lösbarer Probleme, für welche der zugrundeliegende Graph lediglich aus dem Hamiltonschen Kreis besteht. Diese Probleme haben nur eine Lösung und sind mit einem geeigneten Verfahren in linearer Zeit lösbar.

Die Klasse HC_2 der Probleme mit Knotenausgrad 2 ist bereits $\mathcal{NP}$-hart, kann also im allgemeinen unter der Annahme $\mathcal{P} \neq \mathcal{NP}$ von keinem deterministischen Verfahren mit polynomiellem Zeitaufwand gelöst werden. Im Gegensatz zu der Situation bei HC_1 sind jedoch die Instanzen dieser Klasse sehr unterschiedlich beschaffen.

Die für die experimentelle Untersuchung verwendeten Probleminstanzen wurden nach einem (auch von der Arbeitsgruppe um K.Iwama eingesetzten) Verfahren erzeugt, bei dem zunächst ein Hamiltonscher Kreis erzeugt wird und anschließend solange zufällig gewählte Kanten zu diesem Graphen hinzugefügt werden, bis für alle Knoten der gewünschte Ausgrad erreicht ist. Die so erzeugten gerichteten Graphen werden in einem speziellen, auf einer geordneten Inzidenzlistendarstellung basierenden Format als Klartext-Datei gespeichert. Die Implementierungen des GHC- resp. GSHC-Verfahrens arbeiten (wegen impliziter Klauselrepräsentation) direkt auf diesen Dateien.

Für die Experimente an HCP-Instanzen der Klasse HC_1 wurde für jede Problemgröße nur *eine* Probleminstanz verwendet, da das eingesetzte Erzeugungsverfahren für Knotengrad 1 nur identische Instanzen liefert. Bei der experimentellen Untersuchung der Problemklassen $\mathrm{HC}_2(n)$ wurden dagegen in der Regel eine Testmenge von 100 Instanzen eingesetzt; für große Knotenzahlen n wurden wegen Laufzeitbeschränkungen in einzelnen Fällen eine Teilmenge von 10 Probleminstanzen verwendet. Für die Bearbeitung mit GSAT wurden die Probleminstanzen von einem im Rahmen dieser Arbeit entwickelten Konvertierungswerkzeug (HC2SAT), welches die SAT-Kodierung nach Iwama & Miyazaki realisiert, in CNF-Formeln übersetzt. Diese wurden als Textdateien im NP-Format[2] gespeichert.

[2] Hierbei handelt es sich um ein weit verbreitetes Klartext-Format zur Darstellung aussagenlogischer CNF-Formeln (welches unter anderem auch durch die GSAT-Implementierung von Selman & Kautz unterstützt wird).

Generell wurde für jede Problemgröße nur ein Satz von Testinstanzen für die Experimente mit den unterschiedlichen HC-Verfahren verwendet. Obwohl nicht garantiert werden kann, daß die probabilistisch erzeugten Testmengen repräsentativ für die Problemklasse HC_2 sind, ist auf diese Weise eine Vergleichbarkeit der Resultate für die verschiedenen Verfahren gegeben.

8.3 Abschätzung der realen Laufzeit

Die reale Laufzeit (CPU-Zeit) eines Experiments ist einerseits von der Implementierung des eingesetzten Verfahrens und andererseits von einer Reihe weiterer Faktoren abhängig; dies sind im einzelnen:

- die Anzahl I der Probleminstanzen in der Testmenge;
- die Anzahl maxTries der Versuche;
- die maximale Anzahl maxSteps der Schritte pro Versuch;
- für GHC und GSHC (implizite Klauselrepräsentation): die Problemgröße n (Anzahl der Knoten des Graphen);
- für GSAT (explizite Klauselrepräsentation): die Anzahl der Variablen und Klauseln in der entsprechenden CNF-Formel.

Eine obere Schranke der für ein Experiment benötigten CPU-Zeit T kann abgeschätzt werden durch:

$$T \leq I \cdot \mathsf{maxTries} \cdot \mathsf{maxSteps} \cdot t(n)$$

Dabei ist $t(n)$ der von der Problemgröße und dem verwendeten Verfahren abhängige mittlere CPU-Zeitaufwand pro Schritt des Verfahrens. Die reale Laufzeit wird in der Regel unterhalb der so errechneten Schranke T liegen, da ein Versuch nur dann maxSteps viele Schritte umfaßt, wenn er erfolglos bleibt, also kein Modell gefunden wurde. Dennoch ist diese Abschätzung hilfreich beim Entwurf von Versuchsreihen, da mit ihrer Hilfe sichergestellt werden kann, daß die Laufzeiten geplanter Experimente sich innerhalb eines praktikablen Bereichs bewegen. Der dazu benötigte Wert $t(n)$ kann mithilfe eines „Eichexperiments“ für die jeweilige Problemgröße n ermittelt werden.

Im vorliegenden Fall wurde dazu für die Testmengen der Problemklasse $\mathrm{HC}_1(n)$ (jeweils eine Probleminstanz) 10 Versuche à 10 Schritte für $n = 10$ und jeweils ein Versuch mit 100 Schritten für $n \in \{20, 50\}$ durchgeführt; hierbei wurde für alle Verfahren eine Walk-Wahrscheinlichkeit von 0.5 spezifiziert. Diese Eichexperimente wurden für alle verwendeten Verfahren auf den zur Durchführung sämtlicher Versuchsserien eingesetzten Rechnern ausgeführt;[3] aus den hierfür benötigten CPU-Zeiten (Messung mithilfe von in die jeweiligen

[3] Für GSAT wurden die Iwama-Miyzaki-kodierten Probleminstanzen verwendet; der Zeitaufwand für die Kodierung wurde in dieser Messung *nicht* berücksichtigt.

Programme integrierten Betriebssystemaufrufen) wurden folgende Leistungszahlen $t(n)^{-1}$ (Schritte/CPU-Sekunde) für die implementierten Verfahren ermittelt:

$t(n)^{-1}$		Alpha 5/250	HP 712/60	PC 486/100
GSAT	$HC_1(10)$	500	79.37	–
	$HC_1(20)$	76.92	10.49	–
	$HC_1(50)$	4.29	–	–
GHC	$HC_1(10)$	6667	519.79	–
	$HC_1(20)$	1111	81.30	14.33
	$HC_1(50)$	161.29	9.99	–
GSHC	$HC_1(10)$	6675	613.16	–
	$HC_1(20)$	2272	153.85	26
	$HC_1(50)$	341.30	28.9	–

Wegen der analogen Formelstruktur gelten diese Werte auch für die Problemklasse HC_2. Damit ergibt sich aufgrund der obigen Laufzeitabschätzung beispielsweise für ein GSAT-Experiment auf einer Testmenge von 100 Instanzen der Klasse $HC_2(50)$ mit maxTries=10 und maxSteps=10^5 unter Verwendung der leistungsfähigsten Maschine (DEC-Alpha) eine maximale Laufzeit von ca. $23 \cdot 10^6$ CPU-Sekunden (entspricht 266 CPU-Tagen!).

Aus den empirisch bestimmten Leistungszahlen wird zunächst deutlich, daß die Leistung von GSAT deutlich unterhalb der von GHC liegt, welche ihrerseits für große n gegenüber der Leistung von GSHC abfällt. Dies erklärt sich durch die Verwendung impliziter Klauselkodierung für GHC und GSHC und den auf dieser Grundlage realisierten Optimierungen der select-Funktion (vergl. Abschnitte 7.2 und 7.2). Wie aufgrund der Laufzeitbetrachtungen in den Abschnitten 3.6, 7.1 zu erwarten, steigt der Zeitaufwand pro Schritt für alle Verfahren mit wachsender Problemgröße. Weiterhin fällt auf, daß der damit verbundene Leistungsabfall bei den verschiedenen Verfahren unterschiedlich stark erfolgt. Dies deutet darauf hin, daß die zuvor erwähnten Optimierungen für GHC und GSHC den Leistungsverlauf *qualitativ* beeinflussen, d.h. mehr als eine Beschleunigung um einen konstanten Faktor bewirken.

8.4 Durchführung der Experimente

Um die verfügbaren Rechenzeitressourcen möglichst optimal zu nutzen, wurden alle größeren Experimente zu Serien zusammengefaßt und mithilfe von Steuerdateien automatisch durchgeführt. Diese Steuerdateien sowie die Parameterdateien für die einzelnen Experimente wurden mithilfe eines hierfür entwickelten Hilfsprogramms weitgehend automatisch erzeugt.

Die Ergebnisse jedes Experiments wurden automatisch in jeweils zwei Klartext-Dateien zusammengefaßt: einer Berichts-Datei, welche die Parametereinstellungen und die ermittelten Performanzwerte für jeden Versuch sowie für das gesamte Experiment enthält; und einer Resultats-Datei, in welcher zu jeder Probleminstanz alle gefundenen Lösungen protokolliert sind. Im Rahmen der vorliegenden Untersuchung wurden lediglich bestimmte Aspekte der Performanz bei der Auswertung betrachtet; dies sind im einzelnen:

- die *mittlere Erfolgsquote* (success-rate) sr, definiert als Quotient der Anzahl der erfolgreichen Versuche und der Anzahl maxTries aller Versuche;
- der *mittlere Lösungsaufwand* (steps per solution) ss, definiert als Mittelwert der zum Erreichen einer Lösung benötigten Anzahl von Schritten über allen erfolgreichen Versuchen;
- für Eichexperimente: die Anzahl der Schritte pro CPU-Sekunde.

Erfolgsquote und Lösungsaufwand sind wichtige Maße für die Performanz von unvollständigen Verfahren: zusammen erlauben sie eine Abschätzungen darüber, wie oft das Verfahren ausgeführt werden muß, um eine Lösung des gegebenen Problems mit einer bestimmten Wahrscheinlichkeit zu erreichen, und welcher Zeitaufwand für das Finden der Lösung zu erwarten ist. Um den Gesamtzeitaufwand abzuschätzen, ist zu beachten, daß jeder erfolglose Versuch einen Zeitaufwand von maxSteps Schritten verursacht.

Wegen der großen Anzahl von Einzelexperimenten wurde zur automatisierten Extraktion und Tabellierung dieser Werte und der zugehörigen Parametereinstellungen ein Hilfsprogramm entwickelt und eingesetzt.

8.5 GSAT mit adaptiven Klauselgewichten

Viele Autoren berichten, daß sich für strukturierte Probleme die Verwendung von adaptiven Klauselgewichten günstig auf die Performanz von GSAT-ähnlichen Verfahren auswirkt (vergl. Abschnitt 3.3.3). Im Fall der SAT-kodierten HC-Probleme, welche eine ausgeprägte Struktur besitzen (vergl. Abschnitt 6.2.3), wäre also eine Verbesserung der Performanz von GSAT durch die Verwendung adaptiver Klauselgewichte zu erwarten.

Um diese Vermutung zu überprüfen, wurde auf einer identischen Testmenge von 100 $HC_2(10)$-Problemen GSAT mit Random Walk *mit* bzw. *ohne* adaptive Klauselgewichte (GSAT+clw bzw. GSAT-clw) ausgeführt. Die Gewichte wurden zu Beginn jedes Experiments mit 1 initialisiert und am Ende jedes erfolglosen Versuchs ggf. um 1 erhöht (vergl. Abschnitt 2.3); für jede Probleminstanz wurden 100 Versuche durchgeführt. Die Walk-Wahrscheinlichkeit wurde prinzipiell mit walkProb=0.5 auf den als optimal ermittelten Wert gesetzt (vergl. Abschnitt 8.7). Die so ermittelten Performanzwerte sind in der folgenden Tabelle wiedergegeben:

	ms	sr	ss
GSAT-clw	1000	32.31%	484.31
	2000	52.41%	876.11
GSAT+clw	1000	36.76%	479.33
	2000	57.29%	872.09

Dabei bezeichnet ms den Wert des Parameters maxSteps, sr die Erfolgsrate und ss den mittleren Lösungsaufwand (in Schritten pro erfolgreichem Versuch). Während der Lösungsaufwand kaum verändert ist, läßt sich eine leichte Zunahme der Erfolgsquote feststellen (die allerdings aufgrund der relativ hohen Abweichungen zwischen den Ergebnissen einzelner Versuche nicht als signifikant zu betrachten ist). Eine wesentliche Verbesserung der Performanz durch die Verwendung adaptiver Klauselgewichte kann aufgrund dieser Ergebnisse für die betrachtete Problemklasse nicht angenommen werden. Diese Annahme wird durch analoge Ergebnisse für die Testmenge der Klasse $HC_2(8)$ bestätigt.

Resultat 8.1

Die Methode der adaptiven Klauselgewichte („weights“) hat keinen signifikanten Einfluß auf die Performanz von GSAT auf Iwama-Miyazaki-kodierten Probleminstanzen der Klasse HC_2.

Dieses Resultat hat eine wichtige Auswirkung auf die Beurteilung der Methode der impliziten Klauselkodierung. Als Nachteil dieses Verfahrens wurde angeführt, daß die erwünschte erhebliche Reduzierung des Platzbedarfs von GSAT für die gegebene Problemklasse nicht erreicht werden kann, wenn adaptive Klauselgewichte verwendet werden sollen (siehe Abschnitt 7.1.1). Durch das hier erzielte Ergebnis verliert dieser Einwand erheblich an Bedeutung, wodurch das Verfahren der impliziten Klauselkodierung an Attraktivität gewinnt.

Aufgrund der chronologischen Abfolge und Organisation der Versuchsserien wurden im Rahmen dieser Arbeit dennoch fast alle GSAT-Experimente unter der Verwendung adaptiver Klauselgewichte (also *ohne* implizite Klauselkodierung) durchgeführt.

8.6 GSAT auf RBR-erweiterten Formeln

Die Fragestellung, ob sich die Performanz von GSAT durch vorgeschaltete eingeschränkte Resolventen-Erweiterung der SAT-kodierten HC-Instanzen (vergl. Abschnitt 7.1.2) verbessern läßt, wurde an der Standardtestmenge von 100 zufälligen $HC_2(10)$-Problemen untersucht. Dabei wurde zu jeder Formel zunächst die RBR_4 bzw. RBR_8-Erweiterung berechnet; anschließend wurde auf die entsprechend modifizierten Testmengen GSAT mit Random Walk angewandt.

Um eine möglichst optimale Performanz zu erzielen, wurde die Walk-Wahrscheinlichkeit auf den als optimal ermittelten Wert walkProb=0.5 gesetzt (vergl. Abschnitt 8.7). Trotz des Resultats aus Abschnitt 8.5 wurden prinzipiell adaptive Klauselgewichte verwendet, da a priori nicht auszuschließen war, daß diese durch die RBR-Erweiterung der Formeln eine veränderte Wirkung zeigen könnten. Für jede Probleminstanz wurden 10 Versuche durchgeführt (angesichts des ohnehin hohen Zeitaufwands wurde dieser Wert gewählt, um zunächst einen groben Eindruck von der Performanz auf den RBR-erweiterten Formeln zu erhalten).

Die so erzielten Performanzwerte für die RBR-erweiterten Formeln sind im folgenden zusammen mit den entsprechenden Vergleichswerten für die ursprünglichen Testmengen (ohne RBR-Erweiterung, gleiche Paramtereinstellungen für GSAT) aufgeführt:[4]

	ms	sr	ss
ohne RBR	1000	36.76%	479.33
	5000	84.04%	1742.35
mit RBR_4	1000	30.70%	491.04
	5000	81.00%	1787.14
mit RBR_8	1000	29.30%	456.55
	5000	81.60%	1670.26

Aufgrund der Resultate anderer Autoren (vergl. Abschnitt 7.1.2) wäre für die hier betrachteten, stark strukturierten Formeln (stark asymmetrische Variablenabhängigkeiten, vergl. Abschnitt 6.2.3), eine Performanzsteigerung durch die RBR-Erweiterung zu erwarten gewesen. Die hier erzielten Ergebnisse bestätigen diese Erwartung jedoch nicht; während der mittlere Lösungsaufwand keine signifikante Veränderung zeigt, scheint die Erfolgsquote auf den RBR-erweiterten Formeln sogar eher leicht niedriger zu sein, als auf den ursprünglichen Probleminstanzen. Somit gilt:

Resultat 8.2

Die eingeschränkte Resolventen-Erweiterung (RBR) hat für die Iwama-Miyazaki-kodierten Probleminstanzen der Klasse HC_2 keinen signifikanten Einfluß auf die Performanz von GSAT.

Für die untersuchte Parametrisierung scheint somit die RBR-Erweiterung der gegebenen Formeln keinen signifikanten Vorteil für die anschließende Anwendung von GSAT mit Random Walk zu bringen.[5] Unter Berücksichtigung des keinesfalls unerheblichen, durch die RBR-Erweiterung selbst verursachten Aufwand (vergl. Abschnitt 7.1.2) erscheint diese Art der Präkonditionierung für die hier untersuchte Problemklasse als nicht lohnend und wurde folglich im Rahmen dieser Untersuchung nicht weiter angewandt.

[4] Wiederum bezeichnet ms den Wert des Parameters maxSteps, sr die Erfolgsrate und ss den mittleren Lösungsaufwand (in Schritten pro erfolgreichem Versuch).

[5] Dieses Resultat schließt allerdings Performanzverbesserungen durch RBR-Erweiterung für andere Walk-Wahrscheinlichkeiten (insbesondere walkProb=0) nicht aus.

8.7 Abhängigkeit der Performanz von der Walk-Wahrscheinlichkeit

Aus experimentellen Untersuchungen anderer Autoren ist bekannt, daß die Performanz von GSAT durch die Random-Walk-Methode für Formelklassen mit sehr unterschiedlichen strukturellen Eigenschaften signifikant verbessert werden kann (vergl. Abschnitt 3.3.2).

Bereits erste experimentelle Sondierungen mit den Verfahren GSAT, GHC und GSHC auf den Testmengen für die Problemklassen $HC_1(10)$ und $HC_2(10)$ ergaben, daß sich durch die Verwendung der Random-Walk-Strategie eine signifikante Verbesserung der Performanz erzielen läßt. Aufgrund dieses, den Erwartungen voll entsprechenden Resultats wurde anschließend der Einfluß der Walk-Wahrscheinlichkeit auf die Performanz der Verfahren genauer untersucht. Die entsprechenden Varianten der Verfahren ohne Random Walk sind dabei durch den Wert walkProb=0 gekennzeichnet; andererseits charakterisiert walkProb=1 jeweils eine Variante, bei der ausschließlich Random-Walk-Schritte ausgeführt werden.

Für die unterschiedlichen Verfahren wurden Experimente auf den Testmengen für die Problemklassen $HC_1(10)$ (eine Instanz) und $HC_2(10)$ (100 Instanzen) durchgeführt; dabei wurden für jede Probleminstanz jeweils maxTries=100 Versuche durchgeführt. Aufgrund eines zuvor im Rahmen von einzelnen Sondierungsexperimenten beobachteten Zusammenhangs zwischen dem Wert des Parameters maxSteps und der Erfolgsquote wurde maxSteps so gewählt, daß Erfolgsquoten zwischen 20% und 100% für eine mittlere Walk-Wahrscheinlichkeit von 0.5 erreicht werden konnten. Um den Einfluß des Parameters walkProb auf die Performanz der Verfahren zu charakterisieren, wurden nun für diese festgewählten maxSteps-Werte bei Walk-Wahrscheinlichkeiten zwischen 0 und 1 jeweils Erfolgsquote und mittlerer Lösungsaufwand für die einzelnen Verfahren experimentell ermittelt. Die Ergebnisse dieser Versuchsreihen sind in den Abbildungen 1-6a,1-6b,2-6a und 2-6b (siehe Anhang) graphisch dargestellt.

Die entsprechenden Kurvenverläufe für die Erfolgsquote weisen jeweils ein deutliches Maximum auf (siehe Abb.1-6a,2-6a). Für GHC und GSAT liegt dieses bei einer Walk-Wahrscheinlichkeit von 0.4 für die Problemklasse HC_1 und bei 0.5 für HC_2. Für GSHC ergibt sich jedoch ein anderes Bild: Während hier für HC_2 das Maximum bei 0.4 liegt, wird es für HC_1 bereits bei 0.2 angenommen. Diese Beobachtung deutet darauf hin, daß die für GSHC charakteristische und sehr problemspezifische Methode des Vertauschens von Knotenwerten durch die Random-Walk-Strategie auf eine grundsätzlich andere Weise beeinflußt wird, als dies für die GSAT-typischen Änderungen einzelner Knotenwerte in jedem Schritt der Fall ist.[6]

Weiterhin lassen die zwischen den Problemklassen HC_1 und HC_2 beobachteten Unterschiede in der Lage des Maximums darauf schließen, daß Random-Walk-Schritte im Falle der erheblich schwierigeren HC_2-Probleme eine wichtigere Rolle für das Finden von Lösungen spielen. Dieses Resultat kann vermutlich dadurch erklärt werden, daß die Bewertungsfunk-

[6] Der für GSAT auf HC_2-Problemen festgestellte optimale Wert von walkProb=0.5 wurde im Rahmen anderer Untersuchungen oft verwendet und wird dort als Empfehlung Bart Selmans angegeben.

tion im Falle von HC_2-Problemen eine größere Anzahl lokaler Minima aufweist (in jedem Falle existieren mehr Knotenpermutationen, welche *fast* ausschließlich Kanten des Graphen benutzen).[7]

Die Minima der empirisch ermittelten Kurven liegen jeweils an den Randpunkten; dies bedeutet, daß sowohl die reine Random-Walk-Strategie (walkProb=1) als auch die jeweiligen Verfahren ohne Random Walk (walkProb=0) allen Mischformen bezüglich der Erfolgsquote unterlegen sind. Besonders für walkProb-Werte zwischen 0.6 und 0.8 fällt die Erfolgsquote aller Verfahren steil ab; die reine Random-Walk-Strategie findet schließlich praktisch keine Lösungen mehr.

Eine interessante Beobachtung ergibt der Vergleich zwischen dem Verlauf der Erfolgsquote über der Walk-Wahrscheinlichkeit mit dem des mittleren Lösungsaufwandes (Abb.1-6b,2-6b): Obwohl jenseits des optimalen Wertes mit wachsender Walk-Wahrscheinlichkeit immer weniger häufig Lösungen gefunden werden, nimmt der Aufwand pro erfolgreichem Versuch zunächst weiter zu. Da die untersuchten Probleminstanzen in der Regel nur eine Lösung besitzen, kann dieses Phänomen nur durch die Annahme erklärt werden, daß in diesem Bereich die häufigen Random-Walk-Schritte die gerichtete Suche behindern und so einerseits die Erfolgsquote sinkt, andererseits aber der Aufwand pro erfolgreichem Versuch steigt.

Interessanterweise ist der mittlere Lösungsaufwand für die Verfahren ohne Random-Walk am geringsten. Dies erscheint plausibel, wenn man bedenkt, daß in diesem Fall eine sehr direkte Suche nach einem lokalen Optimum der Bewertungsfunktion realisiert wird. Entspricht dieses lokale Optimum einer Lösung, ist der Lösungsaufwand[8] entsprechend gering; anderenfalls ist es aufgrund dieser Eigenschaft der Suche ohne Random-Walk-Schritte schwieriger, das lokale Optimum wieder zu verlassen.

Resultat 8.3

*Die Random-Walk-Strategie hat einen qualitativen Einfluß auf die Performanz aller betrachteten Verfahren; dabei hängt die Erfolgsquote des Verfahrens jeweils von der Wahrscheinlichkeit für Walk-Schritte (*walkProb*) ab.*

Bei festem Wert für den Parameter maxSteps *liegt der optimale* walkProb*-Wert für GSAT und GHC bei 0.5, für GSHC auf der Problemklasse* HC_1 *bei 0.2, auf der Problemklasse* HC_2 *bei 0.4.*

Aufgrund dieser Ergebnisse wurden für alle weiteren Experimente die hier ermittelten, optimalen Walk-Wahrscheinlichkeiten benutzt.

[7] Zur Bedeutung lokaler Minima in der Bewertungsfunktion siehe auch Abschnitt 2.3.

[8] Bei der Bestimmung des mittleren Lösungsaufwandes werden erfolglose Versuche nicht berücksichtigt, vergl. Abschnitt 8.4.

8.8 Abhängigkeit der Performanz vom Parameter maxSteps

Der Parameter maxSteps stellt für alle Verfahren eine obere Schranke für die Anzahl der Schritte pro Versuch dar und bestimmt daher in jedem Fall den Aufwand für jeden fehlgeschlagenen Versuch. Inwieweit allerdings die Erfolgsquote der betrachteten Verfahren vom Wert dieses Parameters beeinflußt wird, ist a priori keineswegs klar. Zwar ist davon auszugehen, daß mit steigendem maxSteps-Wert weder Erfolgsquote noch mittlerer Lösungsaufwand abnehmen, doch inwieweit sich für feste Walk-Wahrscheinlichkeit und Problemgröße ein asymptotisches Verhalten ergibt und welche Performanzwerte überhaupt erreicht werden können, ist zunächst ungewiß. Wegen der angedeuteten Abhängigkeit der Laufzeit vom gewählten maxSteps-Wert scheint es jedoch von großer Bedeutung zu sein, diesen möglichst günstig zu wählen.

8.8.1 Die sr/ms-Charakteristik

Um den Zusammenhang zwischen der maximalen Anzahl der Schritte und den Performanzwerten zu klären, wurden mehrere Versuchsreihen mit jeweils variablem maxSteps-Parameter durchgeführt. Dabei wurden alle Verfahren (GSAT, GHC und GSHC) auf den Testmengen für die Problemklassen $\mathrm{HC}_1(10)$ und $\mathrm{HC}_2(10)$ untersucht; für jede Instanz wurden 100 Versuche durchgeführt. Die Walk-Wahrscheinlichkeit wurde auf die zuvor ermittelten optimalen Werte (siehe Abschnitt 8.7) gesetzt, für GSAT wurden außerdem adaptive Klauselgewichte verwendet. Motiviert von der Tatsache, daß es sich zumindest bei HC_2 um eine Klasse $\mathcal{NP}$-harter Probleme handelt, wurde für den maxSteps-Parameter jeweils eine ungefähr exponentiell ansteigende Folge von Werten gewählt. In der graphischen Darstellung der experimentell ermittelten Performanzwerte (Abbildungen 1-2a,1-2b,2-2a und 2-2b) wurde deshalb für die maxSteps-Achse eine logarithmische Teilung gewählt.

Die Ergebnisse dieser Untersuchung sind einigermaßen überraschend. Für alle Verfahren findet sich ein ungefähr sigmoider Verlauf von Erfolgsquote und mittlerem Lösungsaufwand über dem Logarithmus von maxSteps. Dabei fällt auf, daß die entsprechenden Kurven für die Problemklasse HC_2 (Abb.2-2a,b) wesentlich glatter sind als für HC_1 (Abb.1-2a,b). Dies kann durch die unterschiedliche Anzahl der Probleminstanzen in den jeweiligen Testmengen erklärt werden: Während $\mathrm{HC}_1(10)$ nur ein Element enthält, besteht $\mathrm{HC}_2(10)$ aus 100 zufällig konstruierten Instanzen; vermutlich aufgrund kleiner Strukturunterschiede zwischen den verschiedenen Probleminstanzen scheinen sich die Unregelmäßigkeiten in den Verläufen der experimentell ermittelten Performanzwerte durch die Mittelwertbildung auszugleichen. Diese Vermutung wird durch die Beobachtung bestätigt, daß auch durch eine drastische Erhöhung der Anzahl der Versuche für die Testmenge der Problemklasse HC_1 eine Glättung des Kurvenverlaufs nicht erreicht werden kann (die in Abbildung 1-2a dargestellten Kurven sind für GSAT und GHC mit 1000 Versuchen pro Instanz berechnet worden, während für GSHC 100 Versuche pro Instanz durchgeführt wurden).

Für die Problemklasse $\mathrm{HC}_2(10)$ erscheinen die Kurvenverläufe für die verschiedenen Verfahren (siehe Abbildung 2-2a) im wesentlichen identisch zu sein, sieht man von einer Verschiebung entlang der maxSteps-Achse ab. Darüberhinaus konnte folgende einfache Approximationsfunktion für den beobachteten Kurvenverlauf gefunden werden:

$$sr(ms) = 100 \cdot (1 - 2^{-ms/ms_{50}})$$

Dabei bezeichnet sr die Erfolgsquote in Prozent, ms den maxSteps-Wert und ms_{50} den sogenannten maxSteps-*Halbwert* des betrachteten Verfahrens, d.h. denjenigen maxSteps-Wert, für den eine Erfolgsquote von 50% erreicht wird. Der so beschriebene Zusammenhang zwischen Erfolgsquote und maximaler Anzahl von Schritten wird im folgenden auch als *sr/ms-Charakteristik* bezeichnet.

Um die Güte der Approximation festzustellen, wurde die sr/ms-Charakteristik mithilfe des Programms GNUFIT von Carsten Grammes an die in Abbildung 2-2a dargestellten empirisch ermittelten Werte angepaßt; die Anpassung benutzt den Marquardt-Levenberg-Algorithmus, welcher durch schrittweise Anpassung der Kurvenparameter (hier ms_{50}) die χ^2-Abweichungenzwischen Kurvenwerten und Meßwerten minimiert. Die auf diese Weise angepaßten sr/ms-Charakteristiken sind in Abbildung 2-3 zusammen mit den gemessenen Erfolgsquoten dargestellt. Die Güte dieser Approximation ist aus der empirischen Standardabweichung s zwischen Kurvenwerten und Meßwerten ersichtlich:

Verfahren	ms_{50}	s
GSAT+w	1624	1.68
GHC+w	377	1.98
GSHC+w	142	1.39

Aufgrund dieses Ergebnisses erscheint die Wahl der Approximationsfunktion gerechtfertigt. Als direkte Konsequenz ergibt sich, daß auf den untersuchten $HC_2(10)$-Problemen für beliebige, fest gewählte Erfolgsquoten zwischen den von den verschiedenen Verfahren benötigten maxSteps-Werten jeweils ein identisches Verhältnis von GSAT:GHC:GSHC = 1624:377:142 besteht.

Resultat 8.4

Bei optimaler Einstellung des walkProb*-Parameters ergibt sich für alle Verfahren auf der Problemklasse $HC_2(10)$ der gleiche Zusammenhang zwischen dem Wert ms des Parameters* maxSteps *und der Erfolgsquote sr. Dieser Zusammenhang (sr/ms-Charakteristik) kann durch eine einfache Funktion gut approximiert werden ($sr(ms) = 1 - 2^{ms/ms_{50}}$).*

8.8.2 Eigenschaften der sr/ms-Charakteristik

Weiterhin zeigt sich eine unerwartete Eigenschaft der untersuchten Verfahren: Obwohl diese unvollständig sind, nähert sich ihre Erfolgsquote für die untersuchten Problemklassen mit wachsendem Wert von maxSteps asymptotisch der 100%-Marke. Aufgrund des funktionalen Zusammenhangs ist es damit nicht nur möglich, für beliebig gegebenen maxSteps-Wert die zu erwartende Erfolgsquote zu prognostizieren, sondern es kann mithilfe der Umkehrfunktion von $sr(ms)$ zu jeder gewünschten Erfolgsquote derjenige maxSteps-Wert angegeben werden, der benötigt wird, um diese zu erreichen.

Probabilistische Verfahren, welche die Eigenschaft besitzen, daß für lösbare Probleminstanzen mit beliebig großer Wahrscheinlichkeit eine Lösung gefunden werden kann, sollen als *approximativ vollständig* bezeichnet werden. Demnach gilt:

Resultat 8.5

GSAT, GHC und GSHC sind approximativ vollständige Verfahren für die Problemklasse $HC_2(10)$.

Bei logarithmischer Darstellung von maxSteps zeigt die sr/ms-Charakteristik einen nahezu linearen Verlauf maximaler Steigung zwischen den Erfolgsquoten 30% und 80% (siehe Abb.2-2a). In diesem logarithmischen Bereich der Charakteristik (hier besteht ein logarithmischer Zusammenhang zwischen Erfolgsquote und maxSteps) wird durch eine Erhöhung von maxSteps um einen konstanten Faktor die Erfolgsquote absolut betrachtet am stärksten gesteigert.

Eine weitere, grundlegende Eigenschaft der sr/ms-Charakteristik beruht auf folgendem Zusammenhang: Werden mit maxSteps $= m$ im mittel $sr(m)/100 = q$ Lösungen gefunden, so liegt dieser Wert für maxSteps $= k \cdot m$ bei $sr(k \cdot m)/100 = 1 - (1 - q)^k$ (wie unmittelbar aus der Definition von $sr(ms)$ folgt). Die gleiche mittlere Anzahl von Lösungen ergibt sich jedoch für den Fall, daß k unabhängige Versuche mit jeweils maxSteps $= m$ durchgeführt werden.[9] Dies bedeutet, daß der „Erfolg pro Schritt" weitgehend unabhängig davon ist, auf wieviele Versuche eine gegebene Anzahl von Schritten verteilt wird.

Resultat 8.6

Mittels der sr/ms-Charakteristik kann für GSAT, GHC und GSHC auf der Problemklasse $HC_2(10)$ für jede gewünschte Erfolgswahrscheinlichkeit die Gesamtanzahl der benötigten Schritte angegeben werden. Sofern der verwendete maxSteps*-Wert im logarithmischen Bereich der Charakteristik liegt, spielt die Anzahl der Versuche, auf welche diese Schritte verteilt werden, für die Erfolgswahrscheinlichkeit eines Experiments nahezu keine Rolle.*

Hieraus kann geschlossen werden, daß die in den ersten m Schritten der lokalen Suche

[9] Aufgrund der Abweichungen der Meßwerte von der Charakteristik gilt dieses Resultat nur näherungsweise. Anders als im logarithmischen Bereich der Charakteristik, wo die Anpassung besonders gut ist, gilt der beschriebene Zusammenhang für kleine m sicherlich nicht.

erreichte Position im Suchraum für die Erfolgsquote der folgenden Suchphase von untergeordneter Bedeutung ist. Dieses Resultat steht in Übereinstimmung mit den Ergebnissen anderer experimenteller Untersuchungen, wonach die Performanz von GSAT-ähnlichen Verfahren durch eine Optimierung des Ausgangspunkts der lokalen Suche (`init`-Funktion) bezüglich der Bewertungsfunktion kaum beeinflußt wird (vergl. [GW93]). Weiterhin kann der gezeigte Zusammenhang für die Parallelisierung des Verfahrens ausgenutzt werden, indem anstelle eines Versuchs mit maxSteps $= k \cdot m$ eine Reihe von k Versuchen mit jeweils maxSteps $= m$ parallel ausgeführt werden. Diese einfache Möglichkeit der Parallelisierung ist für alle untersuchten Verfahren anwendbar und besitzt den Vorteil einer guten Skalierbarkeit.

Würden diese Resultate nur für HC_2-Probleme der Größe 10 gelten, wären sie insgesamt wenig interessant. Aufgrund weiterer Untersuchungen an HC_2-Instanzen anderer Größen ergab sich jedoch die Vermutung, daß der für $HC_2(10)$ beobachtete funktionale Zusammenhang zwischen Erfolgsquote und maximaler Anzahl von Schritten unabhängig von der Problemgröße sein könnte. Um diese Hypothese zu prüfen, wurden zwei weitere Versuchsreihen ausgeführt: Einerseits wurde GSAT auf eine Testmenge von 100 Probleminstanzen der Klasse $HC_2(8)$ angesetzt; dabei wurden die gleichen Parametereinstellungen wie für die $HC_2(10)$-Probleme verwendet. Andererseits wurden die Performanzwerte für GHC (ebenfalls mit der zuvor beschriebenen Parametrisierung) auf einer Testmenge aus 10 Instanzen der Problemklasse $HC_2(20)$ ermittelt. Ausgehend von der Hypothese der Problemgrößenunabhängigkeit der sr/ms-Charakteristik wurden entsprechende Kurvenanpassungen vorgenommen; die Ergebnisse dieser Untersuchungen sind als Diagramme in den Abbildungen 2-2c,2-2d,2-2e und 2-2f dargestellt.

Diese Resultate stützen die zuvor geäußerte Vermutung,[10] sind jedoch zunächst zu schwach, um als ausreichender Beleg gelten zu können. Da eine analoge Untersuchung der Hypothese an größeren Problemen aus Rechenzeitgründen nicht möglich war, kam eine indirekte Methode zur Anwendung, welche im folgenden beschrieben wird. Auf diese Weise erhält man:

Resultat 8.7

Unabhängig von der Problemgröße kann für alle untersuchten Verfahren auf der Problemklasse HC_2 der Zusammenhang zwischen maxSteps*-Wert und Erfolgsrate mithilfe der sr/ms-Charakteristik beschrieben werden. Damit lassen sich alle zuvor für die Teilklasse $HC_2(10)$ erzielten, auf der sr/ms-Charakteristik beruhenden Resultate (inbesondere approximative Vollständigkeit der Verfahren), auf die gesamte Problemklasse HC_2 verallgemeinern.*

[10] Dabei ist für die $HC_2(20)$-Probleme zu berücksichtigen, daß aufgrund von Rechenzeitenbeschränkungen lediglich eine Testmenge aus 10 Instanzen verwendet werden konnte, und somit größere Abweichungen in den Ergebnissen erwartete werden mußten.

8.8.3 Problemgrößenabhängigkeit des maxSteps-Halbwerts

Eine weitere, im Grunde sogar interessantere Fragestellung ist die nach dem Zusammenhang zwischen der Problemgröße (Anzahl der Knoten im Graphen) und dem maxSteps-Wert, der benötigt wird, um eine bestimmte, feste Erfolgsquote zu erreichen. Geht man von der Gültigkeit der zuvor untersuchten Hypothese aus, reduziert sich die Aufgabenstellung darauf, für die verschiedenen Verfahren die Abhängigkeit des Halbwertes ms_{50} von der Problemgröße zu bestimmen. Dies kann auf elegante Weise mit praktikablem Aufwand bewerkstelligt werden: Kennt man nämlich einen einzigen Punkt am Beginn des logarithmischen Bereichs der entsprechenden sr/ms-Charakteristik, kann daraus der maxSteps-Halbwert (und jeder andere Punkt der Kurve) mithilfe der Umkehrfunktion von $sr(ms)$ bestimmt werden. Ein solcher Punkt ist jedoch mit einem sehr viel geringeren Aufwand zu bestimmen als der Halbwert selbst; gelingt es also, einen entsprechenden, minimalen maxSteps-Wert zu erraten, genügt ein einziges Experiment zur Bestimmung des gesuchten maxSteps-Halbwertes (auf welcher Grundlage ein solches Raten möglich ist, wird später in diesem Abschnitt erläutert).

Die so gewonnenen Prognosen über den maxSteps-Halbwert wurden für verschiedene Problemgrößen überprüft. Die Ergebnisse dieser Experimente entsprachen durchweg und mit verhältnismäßig hoher Genauigkeit den Erwartungen; dies kann als starker Beleg für die Gültigkeit der zugrundeliegenden Hypothese über die Problemgrößenunabhängigkeit der sr/ms-Charakteristik interpretiert werden.

Die auf diese Weise experimentell ermittelte Abhängigkeit des maxSteps-Halbwerts von der Problemgröße n ist für die verschiedenen Verfahren in Abbildung 2-4 graphisch dargestellt. Im Kurvenverlauf für GSAT zeigt sich bei $n = 9$ ein deutlicher Knick; dieser ist auf eine besondere Eigenschaft der verwendeten SAT-Kodierung der HC-Probleme zurückzuführen. Danach werden für Problemgrößen $n \in \{2^k + 1 : k \in \mathbb{N}\}$ genau k Variable pro Knoten benötig, wobei alle möglichen Knotenwerte gültige Positionen kodieren (somit entstehen *keine* Klauseln in Schritt 1 der Iwama-Miyazaki-Kodierung). Für Problemgrößen $n \in \{2^k + 2 : k \in \mathbb{N}\}$ werden hingegen $k + 1$ Variablen pro Knoten benötigt; allerdings ist dann fast die Hälfte aller möglichen Knotenwerte ungültig, wodurch Schritt 1 der Iwama-Miyazaki-Kodierung eine maximale Anzahl von $\Theta(n)$ Klauseln liefert. Auf diese Weise ergibt sich für alle k zwischen den Problemgrößen $2^k + 1$ und $2^k + 2$ jeweils eine sprunghafte Vergrößerung des Suchraumes, welche sich im dargestellten Kurvenverlauf widerspiegelt. Für GHC und GSHC ist dieses Phänomen nicht zu beobachten, da diese Verfahren ohnehin nur gültige Knotenwerte zulassen.

Der in Abbildung 2-4 graphisch dargestellte Zusammenhang zwischen Problemgröße und Lage des maxSteps-Halbwerts kann für GHC recht gut durch die Exponentialfunktion $ms_{50}(n) = 380 \cdot 1.64^{n-10}$ approximiert werden. Mithilfe dieser Approximation ist es nun möglich, für einen festen maxSteps-Wert die Abhängigkeit der Erfolgsrate von der Problemgröße n zu prognostizieren. Dabei wird für gegebene Problemgröße n zunächst der zugehörige maxSteps-Halbwert $ms_{50}(n)$ bestimmt; durch diesen Wert wird die sr/ms-Charakteristik für n festgelegt, mit deren Hilfe schließlich die Erfolgsrate für den gewählten maxSteps-Wert prognostiziert werden kann. Man beachte, daß diese Prognose auf drei Hypothesen beruht:

1. Die Approximation für die sr/ms-Charakteristik ist korrekt.
2. Die sr/ms-Charakteristik ist problemgrößenunabhängig.
3. Die Approximation für die Lage des maxSteps-Halbwerts ist korrekt.

Um zu prüfen, inwieweit die so gewonne Prognose über die Abhängigkeit der Erfolgsrate von der Problemgröße n zutrifft, wurden für eine Reihe von Problemgrößen die Erfolgsraten von GHC für feste maxSteps-Werte von 1000 bzw. 5000 Schritten experimentell ermittelt. (Wie üblich wurden pro Instanz der Testmengen $\mathrm{HC}_2(n)$ 100 Versuche mit dem optimalen walkProb-Wert von 0.5 durchgeführt.) Die Ergebnisse dieser Stichproben sind zusammen mit dem prognostizierten Zusammenhang in Abbildung 2-5 graphisch dargestellt.

Die relativ geringen Abweichungen zwischen prognostizierten und gemessenen Erfolgsraten bestätigen die der Prognose zugrundeliegenden Hypothesen (s.o.), insbesondere den postulierten exponentiellen Zusammenhang zwischen dem maxSteps-Halbwert (welcher eine obere Schranke für den Zeitaufwand des untersuchten GHC-Algorithmus charakterisiert) und der Problemgröße. Aufgrund der $\mathcal{NP}$-Vollständigkeit der betrachteten Problemklasse HC_2 entspricht dieses Resultat den Erwartungen.

Resultat 8.8

Für GHC auf der Problemklasse HC_2 ist der maxSteps-Halbwert exponentiell abhängig von der Problemgröße. Dieser Zusammenhang erlaubt für beliebige Problemgrößen und maxSteps-Werte eine Prognose der Erfolgsrate unter GHC.

8.8.4 Die sr/ms-Charakteristik für HC_1

Der für die einzelnen Verfahren auf der Problemklasse HC_1 ermittelte Zusammenhang zwischen maxSteps-Wert und Erfolgsrate kann mithilfe der gleichen Funktion approximiert werden, die auch für die Problemklasse HC_2 eingesetzt wurde (siehe Abschnitt 8.8.1); diese Approximation ist graphisch in Abbildung 1-3 dargestellt. Die Abweichung zwischen Approximationsfunktion und Meßwerten ist für $\mathrm{HC}_1(10)$ deutlich größer als für $\mathrm{HC}_2(10)$, was auf die Unterschiede in der Größe der verwendeten Testmengen zurückzuführen ist (100 Instanzen $\mathrm{HC}_2(10)$, eine Instanz für $\mathrm{HC}_1(10)$; vergl. Abschnitt 8.2). Die empirische Standardabweichung zwischen approximierten und gemessenen Werten ist jedoch noch immer so gering, daß die Approximation insgesamt als zulässig erscheint. Verglichen mit den entsprechenden Ergebnissen für $\mathrm{HC}_2(10)$ (siehe Abb.2-3) erscheinen jedoch die sr/ms-Charakteristiken der einzelnen Verfahren für $\mathrm{HC}_1(10)$ entlang der maxSteps-Achse verschoben, sodaß für beliebige Erfolgsraten der entsprechende maxSteps-Wert für $\mathrm{HC}_1(10)$ um einen Faktor 2.3 bis 3.6 unterhalb des Wertes für $\mathrm{HC}_2(10)$ liegt.

Resultat 8.9

Auch für die Problemklasse HC_1 kann der Zusammenhang zwischen maxSteps *und Erfolgsrate durch die sr/ms-Charakteristik beschrieben werden. Die* maxSteps*-Halbwerte liegen bei Problemgröße 10 für alle untersuchten Verfahren deutlich unterhalb der entsprechenden Werten für HC_2.*

Die experimentellen Untersuchungen an der Problemklasse HC_1 deuten außerdem darauf hin, daß die approximative Vollständigkeit der untersuchten Verfahren nur mithilfe der Random-Walk-Strategie (also für walkProb > 0) erreicht werden kann. Bestimmt man nämlich die sr/ms-Charakteristik für die Verfahren ohne Random Walk, so scheint mit wachsendem maxSteps-Wert die Erfolgsrate für GSAT und GHC gegen ca. 40% und für GSHC gegen ca. 80% zu konvergieren (siehe Abbildung 1-1a). Diese Beobachtung unterstreicht die Bedeutung der Random-Walk-Strategie für die Performanz aller untersuchten Verfahren.

Resultat 8.10

Auf der Problemklasse HC_1 kann die approximative Vollständigkeit der untersuchten Verfahren nur unter Verwendung der Random-Walk-Strategie erreicht werden.

Aufgrund der Ähnlichkeiten in der Formelstruktur und der Abhängigkeit der Erfolgsrate von der Walk-Wahrscheinlichkeit für die Problemklassen HC_1 und HC_2 (vergl. Abschnitt 8.7) ist anzunehmen, daß die approximative Vollständigkeit der untersuchten Verfahren auch für HC_2 nur unter Einsatz der Random-Walk-Strategie erzielt werden kann.

8.9 Problemspezifische Adäquatheit der Verfahren

Aufgrund der Komplexitätsunterschiede der Problemklassen HC_1 und HC_2 (vergl. Abschnitt 6.3) kann die problemspezifische Adäquatheit der betrachteten Verfahren auf der Grundlage eines Performanzvergleichs auf diesen Problemklassen untersucht werden. Betrachtet man hierzu zunächst den Verlauf der Erfolgsrate über dem Parameter maxSteps für die Testmenge $\mathrm{HC}_1(10)$ bzw. $\mathrm{HC}_2(10)$ (siehe Abb.1-2a,1-3 vs. 2-2a,2-3), so stellt man fest, daß bei gleichem Kurvenverlauf (vergl. Abschnitt 8.8.4) die maxSteps-Halbwerte auf $\mathrm{HC}_1(10)$ für alle Verfahren um einen Faktor zwischen 2.3 und 3.6 unterhalb des entsprechenden Werts auf $\mathrm{HC}_2(10)$ liegen. Vergleicht man die entsprechenden Werte für den mittleren Lösungsaufwand, ergibt sich ein ähnlicher Faktor (siehe Abb.1-2b vs. Abb.2-2b). Daraus folgt insgesamt, daß trotz der Ähnlichkeiten der Formelstruktur (vergl. Abschnitt 6.2.3) alle untersuchten Verfahren auf der Testmenge $\mathrm{HC}_1(10)$ deutlich effizienter ablaufen als auf der Testmenge $\mathrm{HC}_2(10)$, wobei der *qualitative* Zusammenhang zwischen Aufwand und Lösungsgüte allerdings kaum

Unterschiede aufweist.

Wesentlich interessanter ist es jedoch in diesem Zusammenhang, die Abhängigkeit der Performanz von der Problemgröße für die beiden Problemklassen zu vergleichen. Die entsprechenden experimentellen Resultate für GHC (siehe Abb.1-3 vs. Abb.2-3) deuten darauf hin, daß für beide Problemklassen eine exponentielle (oder bestenfalls leicht subexponentielle) Abhängigkeit des maxSteps-Halbwerts von der Problemgröße vorliegt. Während das asymptotische Wachstum für HC_2 ungefähr bei $\Theta(1.64^n)$ liegt, scheint es für HC_1 etwa durch $\Theta(1.24^n)$ gegeben zu sein.

Wäre GHC ein problemspezifisch adäquates Verfahren für Iwama-Miyzaki-kodierte HC-Probleme, müßte es die Problemklasse HC_1 mit linearem oder zumindest polynomialem Zeitaufwand lösen, da für diese Problemklasse ein lineares Lösungsverfahren bekannt ist (vergl. Abschnitt 6.3). Folglich gilt:

Resultat 8.11

GHC (und damit auch GSAT) kann nicht als problemspezifisch adäquates HC-Verfahren betrachtet werden und stellt damit auch kein adäquates Verfahren für das allgemeine SAT-Problem dar.

Dennoch ist das GHC-Verfahren nicht als völlig inadäquat bezüglich der Klasse der Iwama-Miyzaki-kodierten HC-Probleme zu betrachten, da es auf den wesentlich leichteren HC_1-Probleme eine signifikant bessere Performanz als auf den $\mathcal{NP}$-harten HC_2-Problemen zeigt. Da bislang völlig unklar ist, ob überhaupt ein generell adäquates Verfahren für die betrachtete Problemklasse existiert, kann die Signifikanz dieses Resultats nur durch einen Vergleich mit anderen HC- bzw. SAT-Verfahren festgestellt werden.

8.10 Laufzeitprognosen und maximal praktikable Problemgrößen

In Abschnitt 8.3 wurde bereits eine obere Schranke der für ein HC-Experiment benötigte CPU-Zeit angegeben. Aufgrund der Ergebnisse aus Abschnitt 8.8 kann diese Abschätzung nun erheblich verbessert werden. Insbesondere ist es nunmehr möglich, die mittlere CPU-Zeit zu prognostizieren, die benötigt wird, um auf einer gegebenen Menge von I Instanzen der Problemklasse $HC_2(n)$ eine beliebig gewählte Erfolgsrate sr zu erreichen. Wird für jede Instanz nur ein Versuch durchgeführt (maxTries=1), kann eine obere Schranke für den CPU-Zeitaufwand T folgendermaßen abgeschätzt werden:

$$T \leq I \cdot ms(sr) \cdot t(n)$$

Dabei bezeichnet $t(n)$ wiederum den von der Problemgröße und dem verwendeten Verfahren abhängigen mittleren CPU-Zeitaufwand pro Schritt des Verfahrens (vergl. Abschnitt 8.2). Weiterhin steht $ms(sr)$ für den mithilfe der ms/sr-Charakteristik ermittelten Wert für den Parameter **maxSteps**, wobei der **maxSteps**-Halbwert $ms_{50}(n)$ in Abhängigkeit von der Problemgröße n ermittelt wird.

Diese Abschätzung ist für Erfolgsraten aus dem logarithmischen Bereich der entsprechenden sr/ms-Charakteristik besonders genau, wohingegen sich insbesondere für sehr hohe Erfolgsraten eine größere Ungenauigkeit ergibt. Dies ist jedoch keine ernsthafte Einschränkung, da sich zum Erreichen hoher Erfolgsraten aufgrund der oben erörterten Eigenschaften der sr/ms-Charakteristik ohnehin die Verteilung der benötigten Gesamtanzahl von Schritten auf mehrere Versuche (mit entsprechend niedrigeren **maxSteps**-Werten) empfiehlt. Konkret bedeuted dies, daß beispielsweise zum Erreichen einer Erfolgswahrscheinlichkeit von 91% die benötigte Gesamtanzahl von $ms(91\%)$ Schritten besser auf zwei (möglicherweise simultan ablaufende) Versuche mit einer Erfolgsrate von jeweils 70% verteilt werden sollten, als einen einzelnen Versuch mit einer Erfolgsrate von 91% durchzuführen; da im ersteren Falle die Einzelversuche im logarithmischen Bereich der sr/ms-Charakteristik liegen, kann ihre Laufzeit genauer prognostiziert werden.

Die oben angegebene Abschätzung der CPU-Zeit ist allerdings notwendigerweise ungenau, da sie nur den Fall erfolgloser Versuche erfaßt. Versuche, in denen eine Lösung gefunden wird, benötigen in der Regel weniger als **maxSteps** Schritte; eine gute Abschätzung des tatsächlich benötigten Zeitaufwands müßte also zusätzlich den mittleren Lösungsaufwand berücksichtigen (dieser gibt die mittlere Anzahl der pro erfolgreichem Versuch benötigtem Schritte an). Damit ergibt sich für die mittlere CPU-Zeit für einen Versuch insgesamt folgende Abschätzung:

$$T \leq I \cdot (sr \cdot ss(sr) + (100 - sr) \cdot ms(sr))/100 \cdot t(n)$$

Die Bezeichnungen sind wie oben gewählt (sr wird in Prozent angegeben), außerdem steht $ss(sr)$ für den problemgrößenabhängigen mittleren Lösungsaufwand für erfolgreiche Versuche. Diese Größe muß für die Anwendung dieser Abschätzung hinreichend genau approximiert werden, was zur Zeit leider nicht allgemein möglich ist. Die experimentellen Resultate deuten jedoch darauf hin, daß für die Abhängigkeit des mittleren Lösungsaufwands pro erfolgreichem Versuch von der maximalen Anzahl der Schritte pro Versuch (siehe Abb.1-2b,2-2b,2-2d,2-2f) eine sehr ähnliche Approximation wie im Falle der sr/ms-Charakteristik verwendet werden könnte, wodurch auch für diesen Wert eine Prognose für beliebige **maxSteps**-Werte möglich wäre.

Aufgrund der in den Abbildungen 2-2a bis 2-2f dargestellten Meßwerte für Erfolgsrate und mittleren Lösungsaufwand ergibt sich unabhängig von der Problemgröße folgender Zusammenhang für die jeweiligen Halbwerte:

$$ss(50\%) \approx ms(50\%)/2.2$$

Dabei kann $ss(sr)$ aus der gemessenen (ggf. interpolierten) Größe $ss(ms)$ mithilfe der bekannten Approximation für $sr(ms)$ bestimmt werden.

Geht man davon aus, daß alle Versuche unter Verwendung des Halbwerts für maxSteps durchgeführt werden, so läßt sich für GHC aus der oben gezeigten Laufzeitschranke die maximale Größe praktisch lösbarer Probleme abschätzen. Die hierzu benötigten CPU-Zeiten pro Schritt $t(n)$ werden (wie in Abschnitt 8.3 erläutert) mithilfe von Eichexperimenten ermittelt. Zu jeder gegebenen CPU-Zeitschranke T_{max} kann nun die maximale Größe der mithilfe von GHC auf einer bestimmten Maschine lösbaren HC_2-Probleme folgendermaßen bestimmt werden:

- Approximiere den Zusammenhang zwischen Problemgröße und CPU-Zeit pro Schritt durch die Funktion $t(n)$ (Stützstellen aus Eichexperimenten);
- bestimme $ms(50\%)$ nach der Approximation aus Abschnitt 8.8.3;
- errechne $T(n)$ aus den zuvor bestimmten Werten unter der Annahme $ss(50\%) = ms(50\%)/2.2$;
- bestimme die maximale Größe der in der gegeben Zeit lösbaren Probleme durch $n_{max} = \max\{n \mid T(n) \leq T_{max}\}$.

Wird der GHC-Algorithmus (bei optimaler Walk-Wahrscheinlichkeit) auf einem Rechner vom Typ Alpha 5/250 eingesetzt, so ergeben sich auf diese Weise folgende Laufzeitprognosen:

- Bei einer Testmenge von einem HC_2-Problem und einer maximalen Laufzeit von 100 CPU-Stunden liegt die maximale Problemgröße für das Erreichen einer Erfolgsquote von 50% bei $n = 31$.
- Um auf einer Testmenge von 100 Probleminstanzen der Größe $n = 20$ mit jeweils 100 Versuchen pro Instanz eine Erfolgsrate von 50% zu erreichen, wird eine CPU-Zeit von ca. 96 Stunden benötigt.[11]
- Um ein einziges Problem der Größe $n = 50$ (mit 50% Erfolgsquote) zu lösen, müßte eine CPU-Zeit von ca. 21 Jahren aufgewendet werden!

Resultat 8.12

Auf der Grundlage der zuvor gewonnen Approximationen kann mithilfe von Eichexperimenten der CPU-Zeitaufwand für GHC-Experimente relativ gut prognostiziert werden. Demnach liegt für die benutzte Implementierung auf einem extrem schnellen Rechner vom Typ DEC-Alpha 250/5 die maximale Größe der in vertretbarer Zeit (wenige CPU-Stunden) lösbaren HC_2-Probleme zwischen $n = 25$ und $n = 30$.

[11] Diese Prognose wurde experimentell bestätigt.

Wegen des exponentiellen Laufzeitverhaltens des GHC-Algorithmus ist also die maximale Größe der auf diese Weise lösbaren HC_2-Probleme scharf begrenzt. Obwohl durch Optimierungen in der Implementierung, durch Parallelisierung oder Verwendung eines anderen Verfahrens zur SAT-Transformation diese Grenze hinausgeschoben werden kann, ist aufgrund der $\mathcal{NP}$-Vollständigkeit des HCP auf der Problemklasse HC_2 zu vermuten, daß dies nur in begrenztem Maße und unter erheblichem Aufwand möglich ist.

8.11 Zusammenfassung der wichtigsten Resultate

In diesem Kapitel wurden die Performanzeigenschaften der Verfahren GSAT, GHC und GSHC auf Iwama-Miyazaki-kodierten HC-Problemen ausführlich untersucht. Dabei wurden zwei Teilklassen von HCP-Instanzen betrachtet: HC_1 mit Knotenausgrad 1 (lineare Zeitkomplexität für HCP) und HC_2 mit Knotenausgrad 2 ($\mathcal{NP}$-hartes HCP). Dabei konnten folgende Resultate erzielt werden:

- Beim Performanz-Vergleich der Verfahren auf identischen Mengen von Testproblemen liegt GSHC vor GHC und GHC deutlich vor GSAT (siehe Abb.1-1a,1-2a,2-2a).
- Weder die Verwendung von adaptiven Klauselgewichten („clause weights") noch die eingeschränkte Resolventen-Erweiterung (RBR) der SAT-kodierten HCP-Instanzen hat einen signifikanten Einfluß auf die Performanz von GSAT.
- Die Random-Walk-Strategie hat einen qualitativen Einfluß auf die Performanz aller betrachteten Verfahren; dabei hängt die Erfolgsquote des Verfahrens jeweils von der Wahrscheinlichkeit für Walk-Schritte (walkProb) ab (vergl. Abb.1-1a,1-2a).

  Bei festem Wert für den Parameter maxSteps liegt der optimale walkProb-Wert für GSAT und GHC bei 0.5, für GSHC auf der Problemklasse HC_1 bei 0.2, auf der Problemklasse HC_2 bei 0.4 (siehe Abb.1-6a,2-6a).

Eine ausführliche Untersuchung der Abhängigkeit der Performanz (Erfolgsrate und mittlerer Lösungsaufwand) der Verfahren von der maximalen Anzahl der Schritte pro Versuch (maxSteps) ergab folgende Resultate:

- Bei optimaler Einstellung des walkProb-Parameters ergibt sich für alle Verfahren auf HC_2-Problemen der gleiche Zusammenhang zwischen dem Wert von maxSteps und der Erfolgsquote (sr/ms-Charakteristik). Dieser Zusammenhang ist unabhängig von der Problemgröße (siehe Abb.2-2a,2-2c,2-2e).
- Danach sind GSAT, GHC und GSHC approximativ vollständig in dem Sinne, daß lösbare HCP-Instanzen mit beliebig hoher Wahrscheinlichkeit gelöst werden können. Für HC_1-Probleme wurde gezeigt, daß diese Eigenschaft nur mithilfe der Random-Walk-Strategie erreicht werden kann (vergl. Abb.1-1a,1-2a).

- Die sr/ms-Charakteristik kann durch eine einfache Funktion approximiert werden ($sr(ms) = 1 - 2^{ms/c}$). Auf diese Weise ist einerseits eine Prognose der Erfolgsraten für beliebige maxSteps-Werte möglich; andererseits kann für jede vorgegebene Erfolgsrate der benötigte maxSteps-Wert ermittelt werden (siehe Abb.2-3).

- Die sr/ms-Charakteristik ist so beschaffen, daß in einem weiten Bereich eine Verdoppelung des maxSteps-Werts die gleiche Wirkung zeigt, wie zweimalige Ausführung des Verfahrens. Daraus kann geschlossen werden, daß in diesem Bereich der Ausgangspunkt der lokalen Suche keinen wesentlichen Einfluß auf die Erfolgsrate hat (in dieser Situation bringen folglich Optimierungen bei der Initialisierung der lokalen Suche keinen Vorteil).

- Bei optimaler walkProb-Einstellung ergibt sich aus den experimentellen Daten eine exponentielle Abhängigkeit des für eine beliebig, aber fest gewählte Erfolgsquote benötigten maxSteps-Wertes von der Problemgröße (siehe Abb.2-4). Die hieraus gewonnenen Prognosen über die Lage der sr/ms-Charakteristiken für beliebige Problemgrößen konnten an Stichproben bestätigt werden.

- Aus diesem Zusammenhang kann unter Verwendung der sr/ms-Charakteristik der Verlauf der Erfolgsquote für beliebige Problemgrößen bei festem maxSteps-Wert approximiert werden (siehe Abb.2-5).

- Weiterhin sind mithilfe dieser Approximationen die Laufzeit der untersuchten Verfahren für gegebene Problemgröße und Erfolgsquote sowie die maximale Größe praktisch lösbarer Probleme abschätzbar.

Im Zusammenhang mit der Betrachtung der Adäquatheit der untersuchten Verfahren ergaben sich folgende Resultate:

- Die qualitative Abstufung der Schwierigkeit der Problemklassen entspricht für alle Algorithmen der Erwartung: HC_1-Probleme (prinzipiell linear lösbar) sind wesentlich leichter als HC_2-Probleme ($\mathcal{NP}$-hart).

- Zumindest für GHC konnten auf den Problemklassen HC_1 und HC_2 deutliche Hinweise auf eine exponentielle Abhängigkeit des Lösungsaufwands von der Problemgröße festgestellt werden. Wegen der linearen Zeitkomplexität der HC_1-Probleme kann daher GHC nicht als adäquates Verfahren zur Lösung von HC-Problemen betrachtet werden.

- Wegen der engen Verwandtschaft mit GHC und aufgrund von Performanzvergleichen ist davon auszugehen, daß GSAT zusammen mit der Iwama-Miyazaki-Kodierung kein adäquates HC-Verfahren darstellt und damit auch nicht als adäquates Verfahren für das allgemeine SAT-Problem betrachtet werden kann.

Aufgrund der $\mathcal{NP}$-Vollständigkeit der Problemklasse HC_2 und der Struktur der Iwama-Miyazaki-kodierten Probleminstanzen sind diese Probleme offenbar für lokale Suchverfahren wie GSAT besonders schwierig. Im Gegensatz zu Random-3-SAT-Problemen handelt es sich aber hierbei um eine Problemklasse mit relativ starkem Anwendungsbezug. Aufgrund dieser Eigenschaften bieten sich die hier betrachteten HC_2-Probleme als Benchmark-Probleme für die Bewertung von SAT-Verfahren an; gleichzeitig sind sie in Verbindung mit der strukturell eng verwandten Klasse der SAT-kodierten HC_1-Probleme prinzipiell für problemspezifische Adäquatheitsuntersuchungen geeignet.

Kapitel 9

Übertragung auf ein molekularbiologisches Problem

In diesem Kapitel wird dargestellt, auf welche Weise die vorher untersuchten Verfahren (effiziente SAT-Kodierung von HC-Problemen und deren Lösung mithilfe von SAT-Algorithmen) zur Lösung eines Anwendungsproblems aus dem Bereich der Molekularbiologie einsetzbar sind. Im einzelnen wird zunächst die molekularbiologische Problemstellung aus dem Bereich der physikalischen Genomkartierung vorgestellt; wie dann gezeigt wird, kann dieses kombinatorische Optimierungsproblem als spezielle Variante des „Traveling-Salesman Problem" (TSP) formuliert werden. Anschließend wird ausgeführt, wie die so erhaltenen speziellen TSP-Instanzen mithilfe einer einfachen Erweiterung der Iwama-Miyazaki-Kodierung (aus Kapitel 6) in MaxSAT-Instanzen transformiert und unter Verwendung entsprechender Verallgemeinerungen der in Kapitel 7 vorgestellten SAT-Verfahren prinzipiell gelöst werden können. Das Kapitel schließt mit einer Zusammenfassung und Diskussion der hier vorgeschlagenen Vorgehensweise zur Lösung des molekularbiologischen Problems.

9.1 Molekulargenetische Grundlagen

Die moderne Molekularbiologie beschäftigt sich mit den molekularen Grundlagen der Vererbung und des Stoffwechsels; von besonderem Interesse sind dabei die Steuerungs- und Kontrollmechanismen, welche auf molekularer Ebene alle Lebensvorgänge eines Organismus bestimmen. Eine wesentliche Rolle spielt in diesem Zusammenhang die genetische Information („Erbinformation") eines Organismus, die in ihrer Gesamtheit als *Genom* bezeichnet wird. Genome bestehen aus *DNA-Molekülen*, welche für höhere, mehrzellige Lebewesen größtenteils zu *Chromosomen* verdichtet im Zellkern vorliegen, wobei jedes Chromosom in der Regel genau ein DNA-Molekül enthält. Das menschliche Genom besteht im wesentlichen aus einem Satz von 46 Chromosomen pro Zelle.

Die DNA-Moleküle, welche als universelle Träger genetischer Information betrachtet werden, bestehen jeweils aus einer Kette von einfachen Elementen, den *Nukleotiden*. Diese werden mit den Buchstaben A,T,C und G bezeichnet (für Adenosin, Thymin, Cytosin und Guanin). Ein DNA-Molekül ist damit als eine (aufgrund von chemischen Eigenschaften) *gerichtete Kette aus Nukleotiden* aufzufassen. Die so realisierte Nukleotidsequenz (über dem vierelementigen Alphabet A,T,C,G) kodiert nahezu die gesamte genetische Information.

In Chromosomen liegt die DNA in Form von Doppelsträngen vor; dabei sind jeweils DNA-Moleküle in Form einer Doppelhelix miteinander verzwirnt. Diese Doppelstränge entstehen durch spezifische Paarungen zwischen den Nukleotiden der beiden Stränge (Wasserstoffbrückenbindungen). Dabei kann A lediglich mit T, G lediglich mit C verbunden werden; A und T sowie G und C werden daher auch als *komplementär* bezeichnet. Aufgrund ihrer chemischen Struktur wird eine Verbindung aus zwei komplementären Nukleotiden auch als *Basenpaar* bezeichnet; diese Einheit wird in der Regel verwendet, um die Länge von DNA-Sequenzen anzugeben. Innerhalb eines komplementären DNA-Doppelstrangs repräsentiert also jeder Einzelstrang die gesamte Information; dies bildet die Grundlage des Replikationsmechanismus, der eine identische Verdopplung von DNA-Molekülen bewirkt.

In einer lebenden Zelle wird die Nukleotidsequenz eines DNA-Moleküls abschnittsweise über eine Zwischenstufe (mRNA) in Proteine übersetzt (*Translation* und *Transkription*), welche durch ihre biochemischen Eigenschaften die Stoffwechselsvorgänge der Zelle realisieren. Die hierbei funktional bedeutsamen DNA-Abschnitte werden als *Gene* bezeichnet. Die DNA-Kodierung für Proteine und große Teile des Übersetzungsmechanismus sind für alle bekannten Organismen weitestgehend identisch.

Viele Methoden der modernen Molekularbiologie basieren auf der Kenntnis der Nukleotidsequenz von Genen, Chromosomen und vollständigen Genomen. Die Gewinnung dieser Sequenzinformation ist seit 1977 möglich und basiert auf biochemischen, molekularbiologischen und algorithmischen Methoden. Dieser als *DNA-Sequenzierung* bezeichnete Vorgang kann allerdings aufgrund von Problemen, die sich aus den physikalischen und chemischen Eigenschaften großer DNA-Moleküle ergeben, mit heutigen Mitteln nur für DNA-Stränge mit einer Länge von wenigen Tausend Basenpaaren direkt durchgeführt werden (dies entspricht der Größe kleinerer Virengenome). Da die Genome der meisten Organismen in Form von sehr viel längeren DNA-Molekülen vorliegen (ein menschliches Chromosom besteht aus ca. 10^8 Basenpaaren), müssen spezielle Methoden eingesetzt werden, um die gesamte Sequenzinformation solcher langen DNA-Stränge zu gewinnen.

Als Vorstufe einer kompletten Sequenzierung kann die *physikalische Genomkartierung* betrachtet werden; dabei versucht man, bestimmte charakteristische Sequenzmotive auf dem DNA-Strang zu lokalisieren. Die so gewonnenen Genomkarten dienen oft als Orientierungspunkte für detailliertere Untersuchungen einzelner Gene oder Genomabschnitte oder als Ausgangspunkt für die komplette Sequenzierung. Für die Genomkartierung gelten ähnliche Beschränkungen wie für die Sequenzierung: auch hier sind beispielsweise ganze Chromosome in der Regel nicht handhabbar.

Mithilfe verschiedener Techniken ist es in den letzten Jahren gelungen, die Genome einiger einfacher Organismen (z.B. der Bäckerhefe *Saccharomyces cerevisiae*) komplett zu sequenzieren. Darüberhinaus laufen gegenwärtig etliche Projekte zur Sequenzierung größerer Genome; das spektakulärste unter diesen ist das *Human Genome Project*, welches die komplette Sequenzierung des menschlichen Genoms zum Ziel hat. Aufgrund ihres immensen Umfangs sind derartige Vorhaben nur mithilfe weitestgehend automatisierter Kartierungs- bzw. Sequenzierungsverfahren durchführbar. Da hierbei spezielle algorithmische Methoden eine zentrale Rolle spielen, ergeben sich in diesem Zusammenhang vielfältige und komplexe Aufgabenstellungen für die Informatik.

9.2 Ein kombinatorisches Problem bei der DNA-Kartierung

Bei der Kartierung von DNA-Molekülen, die zu groß für die direkte Analyse sind, geht man üblicherweise folgendermaßen vor:

Schritt 1: Eine gewisse Anzahl der zu untersuchenden DNA-Moleküle (beispielsweise Chromosomen) werden isoliert.

Schritt 2: Mithilfe von biochemischen (z.B. Restriktionsenzyme) oder physikalischen Methoden (z.B. Ultraschall) werden die DNA-Stränge in Fragmente handhabbarer Größe zerlegt.

Schritt 3: Die so erzeugten DNA-Fragmente werden direkt analysiert.

Schritt 4: Mithilfe der so gewonnenen Informationen wird die Position jedes Fragments innerhalb des ursprünglichen Gesamtmoleküls bestimmt.

Die in Schritt 2 erzeugten Fragmente werden auch als *Klone* bezeichnet; diese besitzen, abhängig von der eingesetzten Methode und der Art der intendierten weiteren Untersuchung, eine Größe von 3000 bis 2 Millionen Basenpaaren. Derartige Klone können in der Regel leicht vervielfältigt oder als sogenannte *Genbank* aufbewahrt werden.

9.2.1 Die STS-Kartierungsstrategie

Die STS-Kartierungsstrategie (*Sequence Tagged Site mapping strategy*) ist eine für praktische Anwendungen weit verbreitete Variante dieser allgemeinen Vorgehensweise. Dabei werden zunächst kurze Einzelstrang-DNA-Fragmente aus der Gesamtsequenz isoliert (oft von den Enden bereits erzeugter Klone), deren Sequenz bekannt ist, und die durch spezielle Markierungstechniken (z.B. durch radioaktive Atome oder Fluoreszenzfarbstoffe) leicht nachweisbar sind. Diese Fragmente werden als *DNA-Sonden* bezeichnet, da sie unter geeigneten Bedingungen dazu eingesetzt werden können, Klone zu identifizieren, welche eine zur Sonde komplementäre Teilsequenz enthalten. Dabei wird die zu untersuchende Doppelstrang-DNA der Klone in Einzelstränge aufgetrennt, die sich dann mit geeigneten Sonden durch Bildung von Basenpaarungen zwischen Sonde und Klon verbinden kann; dieser Vorgang wird als *Hybridisierung* bezeichnet.

Die Länge der Sonden wird so gewählt, daß die Sequenz jeder Sonde mit hoher Wahrscheinlichkeit nur einmal auf dem gesamten zu untersuchenden DNA-Strang auftaucht. Ist dies gewährleistet, markiert jede Sonde durch ihre Sequenz eine eindeutige Position (*sequence tagged site*) der zu untersuchenden Gesamtsequenz. Für die Kartierung werden die Sonden mithilfe verschiedener Methoden gewonnen, um eine möglichst gleichmäßige Verteilung der Sondenpositionen über den gesamten DNA-Strang zu gewährleisten.

Hat man sich eine Menge von Klonen und Sonden verschafft, wird nun durch Hybridisierungsexperimente ermittelt, welche Sonde mit welchem Klon hybridisiert. Da bei der Erzeugung der Klone in der Regel eine Menge überlappender Sequenzfragmente entsteht, können aus dieser Hybridisierungsinformation Rückschlüsse über die Anordnung der Sondenpositionen innerhalb der Gesamtsequenz gezogen werden; im Idealfall kann auf diese Weise die Reihenfolge der Sonden und damit auch die Anordnung der Klone eindeutig bestimmt werden. Diese Aufgabenstellung, aus den Ergebnissen der Hybridisierungsexperimente auf die Anordnung der Klone zu schließen, kann als kombinatorisches Problem formuliert werden. Dabei wird folgendes Modell zugrundegelegt:

Definition 9.1 (Hybridisierungsmatrix)

Sei $C = \{c_1, \ldots, c_n\}$ eine Menge von n Klonen einer zu rekonstruierenden Gesamtsequenz, und $P = \{p_1, \ldots, p_m\}$ eine Menge von m Sonden der gleichen Sequenz.

Die Hybridisierungsmöglichkeiten zwischen den Klonen C und den Sonden P werden durch die (ideale) Hybridisierungsmatrix $H = (h_{i,j})$ *beschrieben; dies ist eine Matrix vom Typ $m \times n$ über der Menge $\{0, 1\}$, welche für jede Sonde aus P eine Zeile und für jeden Klon aus C eine Spalte besitzt. Die Einträge von H sind folgendermaßen definiert:*

$$h_{i,j} = \begin{cases} 1 & : \text{ Sonde } p_i \text{ kann mit Klon } c_j \text{ hybridisieren, d.h. die Sequenz von } p_i \\ & \text{ ist komplementär zu einer Teilsequenz von } c_j \\ 0 & : \text{ sonst} \end{cases}$$

Leider verlaufen Hybridisierungsexperimente in der Regel nicht fehlerfrei, sodaß deren Ergebnisse nicht mit den Einträgen der soeben definierten (idealen) Hybridisierungsmatrix H übereinstimmen. Im Rahmen dieser Betrachtung sollen zwei Fehlerarten unterschieden werden: irrtümliche Hybridisierungen (*false positives*) und fehlende Hybridisierungen (*false negatives*).[1]

Die experimentell ermittelte (und möglicherweise fehlerbehaftete) Hybridisierungsinformation wird ebenfalls in Form einer (empririschen) Hybridisierungsmatrix $\tilde{H}$ dargestellt, deren

[1] Es existieren weitere Fehlerquellen wie Chimärenbildung, Inversionen, Deletionen und Translokationen, auf die aber in diesem Zusammenhang nicht weiter eingegangen werden soll.

Einträge genau dann 1 sind, wenn für die entsprechende Kombination aus Sonde und Klon tatsächlich eine Hybridisierung beobachtet wurde. Die hier betrachteten Fehler entsprechen gerade den Abweichungen von jeweils einzelnen Einträgen zwischen idealer und empirischer Hybridisierungsmatrix: In diesem Sinne entsprechen irrtümliche Hybridisierungen der Situation $h_{i,j} = 0$ und $\tilde{h}_{i,j} = 1$, während fehlende Hybridisierungen die duale Situation $h_{i,j} = 1$ und $\tilde{h}_{i,j} = 0$ bedingen. Im Modell werden diese Fehlerarten durch zwei Parameter beschrieben; ϵ_+ ist dabei die Wahrscheinlichkeit für das Auftreten von irrtümlichen Hybridisierungen und ϵ_- die Wahrscheinlichkeit für fehlende Hybridisierungen.

9.2.2 Das Anordnungsproblem

Bei der STS-Kartierungsstrategie ergibt sich das folgende Anordnungsproblem:

Definition 9.2 (Anordnungsproblem)

Zu einer DNA-Sequenz s sei eine Menge C von Klonen und eine Menge P von Sonden sowie die empirische Hybridisierungsmatrix $\tilde{H}$ gegeben. Das Anordnungsproblem *besteht darin, bei bekannten Fehlerraten ϵ_+ und ϵ_- eine möglichst gute Hypothese über die relative Anordnung der Klone innerhalb der Gesamtsequenz s zu finden.*

Da die Sonden so konstruiert wurden, daß jeder Sonde eine eindeutige Position innerhalb der Gesamtsequenz entspricht, müssen sich verschiedene Klone, die mit der gleichen Sonde (fehlerfrei) hybridisieren, in diesem Fall immer überlappen. Infolgedessen weist die ideale Hybridisierungsmatrix H immer die folgende Eigenschaft auf (*"consecutive ones property" für Spalten*):

> *Die Zeilen von H können so permutiert werden, daß in jeder Spalte alle Einträge mit Wert 1 einen einzigen, zusammenhängenden Block bilden.*

Beispiel: ...

Aus dieser Zeilenpermutation der Matrix H kann die Anordnung der Klone leicht abgelesen werden. Während im fehlerfreien Fall empirische und ideale Hybridisierungsmatrix übereinstimmen, die genannte Eigenschaft also folglich auch für das experimentell ermittelte $\tilde{H}$ gilt, wird die Situation im fehlerbehafteten Falle dadurch kompliziert, daß die empirische Hybridisierungsmatrix in der Regel *nicht* die "consecutive ones property" für Spalten aufweist. Das folgende Resultat (siehe [Kar93]) zeigt, daß das Anordnungsproblem im fehlerfreien Fall effizient lösbar ist, während es sich im fehlerbehafteten Fall um eine algorithmisch schwierige Problemstellung handelt.

Satz 9.1 Komplexität des Anordnungsproblems

Das Anordnungsproblem ist für den Fall $\epsilon_+ = \epsilon_- = 0$ in linearer Zeit lösbar; für $\epsilon_+ \neq 0$ oder $\epsilon_- \neq 0$ handelt es sich hingegen um ein $\mathcal{NP}$-hartes Problem.

In der Realität sind Hybridisierungsfehler bislang praktisch unvermeidlich; daher läßt sich die tatsächlich eingesetzte Vorgehensweise bei der STS-Kartierung als interaktiver Prozess aus Experimenten und automatisch generierten Hypothesen beschreiben; hierbei werden Inkonsistenzen oder Unsicherheiten im Datenmaterial iterativ aufgedeckt und ggf. durch zusätzliche Experimente behoben.

9.3 Algorithmische Ansätze zur STS-Kartierung

Die algorithmische Lösung des fehlerbehafteten Anordnungsproblems wurde in verschiedenen Arbeiten ausführlich untersucht (siehe [AKWZ94, Fin95]). Wie bereits im letzten Abschnitt erläutert, besteht das eigentliche Problem darin, eine Zeilenpermutation der empirischen Hybridisierungsmatrix $\tilde{H}$ derart zu finden, daß bei gegebenen Fehlerraten ϵ_+ und ϵ_- die zugrundeliegende ideale Hybridisierungsmatrix H mit möglichst hoher Wahrscheinlichkeit die "consecutive ones property" für Spalten besitzt. Somit handelt es sich um ein $\mathcal{NP}$-hartes kombinatorisches Optimierungsproblem.

9.3.1 Allgemeine Vorgehensweisen

Zur Lösung dieses Problems wurden unter anderem folgende Methoden vorgeschlagen (vergl. [Fin95]):

- Maximierung einer Maximum-Likelihood Zielfunktion mithilfe lokaler Suchverfahren und/oder Simulated Annealing;
- Minimierung einer geeigneten Energiefunktion mittels Simulated Annealing;
- Varianzanalytische Verfahren auf der Basis von Klon-Klon-Korrelationen;
- Abbildung auf ein Traveling-Salesman Problem (TSP) mit Hamming-Distanz-Metrik.

Die letztgenannte Vorgehensweise erscheint dabei besonders interessant, da sie durch eine geeignete Transformation des Problems eine Lösung mithilfe von bekannten und gut untersuchten Standardverfahren ermöglicht. Gleichzeitig konnte gezeigt werden, daß diese Vorgehensweise geeignet ist, um sehr schnell relativ brauchbare Lösungen des Anordnungsproblems zu finden. Darüberhinaus können auf diese Weise irrtümliche Hybridisierungen mit hoher

Effizient erkannt werden ("false positive screening"). Auf der Basis der dadurch ermöglichten Korrektur der empirischen Hybridisierungsdaten ist es möglich, bei leicht erhöhtem Zeitaufwand gute Lösungen des Anordnungsproblems zu finden (siehe [AKWZ94]).

In der Praxis werden die besten Resultate mithilfe einer Kombination verschiedener spezieller Verfahren erreicht; dabei wird eine Erkennung irrtümlicher Hybridisierungen mithilfe der beschriebenen TSP-Transformation realisiert.

9.3.2 Reduktion auf Hamming-Distanz TSP

Der TSP-Transformation des Anordnungsproblems liegt folgende Idee zugrunde: Betrachtet man für jede Spalte der idealen Hybridisierungsmatrix diejenigen Einträge mit Wert 1, so bilden diese unter einer bestimmten Zeilenpermutation einen einzigen, zusammenhängenden Block. Unter derselben Zeilenpermutation erscheinen in den Spalten der empirischen Hybridisierungsmatrix durch die betrachteten Fehlerarten Lücken. Eine gute Anordnung der Sonden entspricht nun einer Zeilenpermutation, welche die Anzahl dieser Lücken minimiert. Diese Minimierungsaufgabe kann folgendermaßen als als Traveling-Salesman Problem formuliert werden:

Definition 9.3 (Hamming-Distanz TSP (HD-TSP))

Zu einer DNA-Sequenz s sei eine Menge C von Klonen und eine Menge P von Sonden sowie die empirische Hybridisierungsmatrix $\tilde{H}$ gegeben. Die Matrix M entstehe aus $\tilde{H}$ durch Hinzufügen einer Zeile, deren Einträge sämtlichst den Wert 0 aufweisen. Sei weiterhin der vollständige, bewertete Graph $G = (V, E, c)$ über der Indexmenge der Klone definiert durch:

$$\begin{aligned} V &:= \{k \mid 1 \leq i \leq |P| + 1\} \\ E &:= \{(i, j) \mid i, j \in V\} \\ c(i, j) &:= d_H(M_i, M_j) \end{aligned}$$

Dabei ist $d_H(M_i, M_j)$ die Hamming-Distanz (d.h. die Anzahl der unterschiedlichen Einträge) der Zeilen i und j der Matrix M.

Das Hamming-Distanz TSP *besteht darin, einen Hamiltonschen Kreis im Graphen G zu finden, für den die Summe der Kantenkosten minimal ist.*

Anmerkung: Die Erweiterung der Matrix um eine „Null-Zeile" ist erforderlich, um eine korrekte Verbindung des letzten mit dem ersten Element der gesuchten Zeilenpermutation zu einem (kosten-minimalen) Kreis zu ermöglichen.

Da G ein vollständig verbundener Graph ist, besteht in diesem Fall die Schwierigkeit nicht darin, überhaupt einen Hamiltonschen Kreis in G zu finden, sondern vielmehr in der Minimierung der Kantenkosten. Es handelt sich bei dieser speziellen Variante des TSP um ein $\mathcal{NP}$-hartes Problem (vergl. [AKWZ94]).

Die Lösung des HD-TSP entspricht direkt der gesuchten optimale Zeilenpermutation der empirischen Hybridisierungsmatrix $\tilde{H}$; aus der so ermittelten Reihenfolge der Sondenposition innerhalb der Gesamtsequenz kann die gesuchte Anordnung der Klone leicht abgeleitet werden. Ein Nachteil der Methode ergibt sich allerdings daraus, daß die Zielfunktion des HD-TSP keinen Nutzen aus den bekannten Fehlerraten zieht; dennoch eignet sie sich, wie bereits erwähnt, für praktische Anwendungen (insbesondere zur Erkennung irrtümlicher Hybridisierungen).

9.4 MaxSAT-Kodierung von HD-TSP-Instanzen

Wie bereits in Abschnitt 6.1 erwähnt, kann man das Traveling-Salesman Problem als Verallgemeinerung des Hamilton-Kreis Problems betrachten. Aufgrund der Untersuchungen zur SAT-Reduktion von HCP-Instanzen (siehe Kapitel 6–8) liegt es nun nahe, die dabei verwendete Vorgehensweise auf das HD-TSP zu übertragen und somit letztendlich das Anordnungsproblem bei der STS-Kartierung auf eine sehr allgemeine logische Problemstellung zurückzuführen.

Tatsächlich kann dies mithilfe einer naheliegenden Modifikation der Iwama-Miyazaki-Transformation geleistet werden. Dabei besteht die zugrundeliegende Idee darin, anstelle einer CNF-Formel eine Formel der Penalty-Logik (vergl. Abschnitt 1.4) zu erzeugen und die Kostenbewertung der Kanten gerade auf die Gewichte der Klauseln abzubilden. Damit wird die gegebene TSP-Instanz in ein MaxSAT-Problem transformiert, welches mithilfe geeignet erweiterter SAT-Verfahren behandelt werden kann. Hierbei kann man folgenderweise vorgehen:

Zunächst wird das Minimierungsproblem in ein Maximierungsproblem transformiert, indem man den Graphen $G' = (V, E, c')$ betrachtet, der aus G entsteht, indem jede Kantenbewertung $c(i, j) = c$ durch $c'(i, j) = |V| + 1 - c$ ersetzt wird.[2]

Für den Graphen G' führt man eine modifizierte Iwama-Miyazaki-Transformation durch (vergl. Abschnitt 6.2.1), wobei jeweils penalty-logische Klauseln erzeugt werden. In den Schritten 3 und 4 des ursprünglichen Iwama-Miyazaki-Verfahrens wird für jedes Paar von Knoten, welche *nicht* durch eine Kante im Graphen verbunden sind, eine Klausel erzeugt; im modifizierten Verfahren wird für *jede* Kante (i, j) von G' die entsprechende Klausel erzeugt und mit dem Gewicht $c'(i, j)$ versehen. Damit wird erreicht, daß für jede zulässige Lösung (also für jede Rundtour im Graphen) die verwendeten Kanten einen Beitrag zum Gewicht des Modells liefern, der umgekehrt proportional zu den entsprechenden Kosten der Kanten in G ist. Die Summe der Gewichte aller in diesen Schritten erzeugten Klauseln sei mit $W_{3,4}$ bezeichnet.

Die Menge der in den übrigen Schritten 1 und 2 des ursprünglichen Verfahrens erzeugten Klauseln kodiert gerade eine hinreichende Bedingung dafür, daß jedes Modell der Formel

[2] Dieser Wert ist immer positiv, da $|V|$ gerade die maximale Hamming-Distanz zweier Zeilen der Matrix M ist.

einer Knotenpermutation von G' entspricht, also eine Rundtour kodiert. Jeder der hier erzeugten Klauseln wird im modifizierten Verfahren das Gewicht $W_{3,4} + 1$ zugeordnet; die Summe der Gewichte aller so erzeugten Klauseln sei mit $W_{1,2}$ bezeichnet.

Damit gilt für die so erzeugte Gesamtformel Φ: Jedes $W_{3,4}$-Modell B von Φ erfüllt alle in Schritt 1 und 2 erzeugten Klauseln (vergl. Abschnitt 1.4) und kodiert somit eine Rundtour in G'. Weiterhin entsprechen die maximalen Modelle von Φ gerade den kostenminimalen Rundtouren in G und somit den Lösungen des gegebenen Hamming-Distanz-TSP. Insgesamt wird auf diese Weise jede gegebene HD-TSP-Instanz über einem Graphen mit n Knoten in eine MaxSAT-Instanz über $(n-1)\lceil \log_2 n - 1 \rceil$ Variablen transformiert.

9.5 Algorithmische Lösung kodierter HD-TSP-Instanzen

Die in Kapitel 3 vorgestellten GenSAT-Verfahren könne in der Regel leicht zu MaxSAT-Verfahren erweitert werden. Für GSAT entspricht dies im wesentlichen der Verwendung von Klauselgewichten (vergl. Abschnitt 2.3), welche mit den Gewichten der penalty-logischen Klauseln initialisiert werden und während des Verfahrens nicht modifiziert werden. Die lokale Suche verläuft dann im wesentlichen wie im SAT-Fall; da jedoch viele MaxSAT-Probleme keine Modelle im aussagenlogischen Sinne besitzen (dies gilt insbesondere für die kodierten HD-TSP-Instanzen), wird am Ende jedes Versuchs das beste bislang gefundene Modell gespeichert, sofern dessen Penalty-Wert die gegebene Penalty-Schranke P nicht überschreitet. Die Güte der so gefundenen Lösungen ist ein wichtiges Kriterium zur Bewertung von MaxSAT-Algorithmen.

Da das zur MaxSAT-Kodierung der HD-TSP-Instanzen verwendete Verfahren sich kaum von der SAT-Kodierung von HCP-Instanzen nach Iwama und Miyazaki unterscheidet, gelten die Aussagen über die Struktur der so erzeugten Formeln (siehe Abschnitt 6.2.3) auch in diesem Fall. Somit ergeben sich für den Einsatz der MaxSAT-Varianten von GenSAT-Verfahren ähnliche Schwierigkeiten wie für die SAT-kodierten HCP-Instanzen (vergl. Abschnitt 7.1). Glücklicherweise können die in Kapitel 7 entwickelten Mechanismen zur Überwindung dieser Schwierigkeiten und zur Verbesserung der Performanz unter Ausnutzung der analogen Problemstruktur auf den Fall der MaxSAT-kodierten TSP-Instanzen übertragen werden. Auf diese Weise kann insbesondere ein zum GHC-Verfahren analoger Algorithmus GTSP zur Lösung des HD-TSP gewonnen werden, welcher durch implizite Klauselkodierung einen quadratischen Platzbedarf (bezüglich der Größe n des Graphen) aufweist.

Um die prinzipielle Durchführbarkeit dieser Vorgehensweise zur Lösung von HD-TSP-Instanzen zu demonstrieren, wurde der GTSP-Algorithmus im Rahmen dieser Arbeit implementiert und auf einigen künstlich erzeugten HD-TSP-Instanzen experimentiell evaluiert. Die Versuchsergebnisse deuten darauf hin, daß unter Verwendung der Random-Walk-Strategie für steigende maxSteps-Werte die mittlere Lösungsgüte gegen das globale Optimum konvergiert. Angesichts der approximativen Vollständigkeitsresultate für SAT-kodierte HCP-Instanzen (siehe Abschnitt 8.8.2) enspricht diese Beobachtung der Erwartung; da aus Zeitgründen allerdings nur wenige Experimente auf kleinen Probleminstanzen durchgeführt werden konnten, ist die Aussagekraft der experimentellen Resultate jedoch stark beschränkt.

9.6 Zusammenfassung

In diesem Kapitel wurde gezeigt, wie das Anordnungsproblem der STS-Kartierung — ein $\mathcal{NP}$-hartes kombinatorisches Optimierungsproblem aus dem Bereich der Molekularbiologie — über ein spezielles Traveling-Salesman Problem auf das MaxSAT-Problem der Penalty-Logik reduziert werden kann. Dabei wird im wesentlichen das effiziente Verfahren zur SAT-Kodierung von Instanzen des Hamilton-Kreis Problems nach Iwama und Miyazaki verwendet. Aufgrund der daraus resultierenden strukturellen Ähnlichkeit zwischen SAT-kodierten HCP-Instanzen und MaxSAT-kodierten TSP-Instanzen sind die in Kapitel 7 entwickelten Verfahren auf diese erweiterte Problemstellung übertragbar.

Somit ist die Behandlung des biologischen Problems mit allgemeinen logischen Verfahren prinzipiell möglich; erste Ergebnisse einer experimentellen Untersuchung deuten darauf hin, daß einige der für SAT-kodierte HCP-Instanzen nachgewiesenen Performanzresultate der entsprechenden Algorithmen sich auf die hier betrachtete Problemstellung übertragen lassen. Unklar ist allerdings, inwieweit die hier aufgezeigte Vorgehensweise zur Lösung realer Probleminstanzen aus dem biologischen Anwendungsbereich geeignet ist. Es erscheint jedoch unwahrscheinlich, daß die hier betrachtete Methode zur Lösung des HD-TSP mit den bekannten TSP-Algorithmen konkurrieren kann. Dennoch ist deren Untersuchung von theoretischem Interesse, da sich der zugrundeliegende Ansatz der Transformation auf eine sehr allgemeine und gut untersuchte Klasse logischer Probleme auf viele speziellen Anwendungsprobleme übertragen läßt.

Resumé und Ausblick

Aufbauend auf zwei neueren Entwicklungen, dem GSAT-Verfahren zur konstruktiven Lösung des aussagenlogischen SAT-Problems einerseits und der effizienten SAT-Kodierung von Hamilton-Kreis Problemen in Graphen nach Iwama und Miyazaki andererseits, wurde in dieser Arbeit untersucht, inwieweit sich ein Anwendungsproblem aus dem Bereich der Molekularbiologie effizient auf eine Variante des SAT-Problems in der Aussagenlogik transformieren und mithilfe GSAT-ähnlicher Verfahren praktisch lösen läßt.

Im ersten Teil der Arbeit (Kapitel 1–5) wurde neben den benötigten theoretischen Grundlagen die einheitliche Darstellung einer Familie von SAT-Verfahren mithilfe des GenSAT-Rahmens (aufbauend auf [GW93, BAH$^+$94a]) behandelt. Weiterhin wurde die Evaluierung von GenSAT-Verfahren ausführlich diskutiert; hierbei wurden insbesondere Aspekte der Evaluierung anhand experimenteller Untersuchungen betrachtet, Probleme in diesem Bereich aufgezeigt und schließlich eine experimentelle Vorgehensweise entwickelt, welche die Grundlage für den experimentellen Teil dieser Arbeit bildete. Abschließend wurden Möglichkeiten zur Parallelisierung von GenSAT-Verfahren systematisch betrachtet; da parallele Verarbeitung eine immer größere Rolle in allen Bereichen der Informatik spielt, können die hier diskutierten Ansätze einerseits zur Bewertung von GenSAT-Verfahren (im Bezug auf ihre Parallelisierbarkeit) herangezogen werden; andererseits können sie jedoch auch als Grundlage zur Entwicklung und Untersuchung paralleler GenSAT-Implementierungen dienen.

Die praktische Anwendbarkeit von SAT-Verfahren zur Lösung eines entsprechend kodierten molekularbiologischen Anwendungsproblems wurde im zweiten Teil der Arbeit (Kapitel 6–9) untersucht. Sehr schnell wurde dabei deutlich, daß bereits die zugrundegelegte SAT-Kodierung von HCP-Instanzen zu erheblichen Problemen im Zusammenspiel mit den betrachteten GSAT-Varianten führte. Hieraus ergab sich die Notwendigkeit einer eingehenderen Untersuchung in diesem Bereich; dabei wurden folgende inhaltliche Schwerpunkte bearbeitet:

- Charakterisierung struktureller Eigenschaften der CNF-Formeln, welche mithilfe der variableneffizienten SAT-Kodierung nach Iwama & Miyazaki aus HCP-Instanzen gewonnen wurden;
- Analyse der Auswirkungen dieser Eigenschaften auf die Performanz von GSAT-ähnlichen Verfahren;

- Problemspezifische Anpassung von GSAT an das HCP (GHC,GSHC);
- Experimentelle Untersuchung des Laufzeitverhaltens von GSAT, GHC und GSHC auf verschieden schwierigen Klassen von HC-Problemen (HC_1 und HC_2);
- Experimentelle Untersuchung von GSAT-Varianten (adaptive Klauselgewichte, Random Walk);
- Experimentelle Untersuchung von RBR-Erweiterungen der kodierten HC-Probleme (Präkonditionierung durch eine spezielle Form der Resolution);
- Experimentelle Untersuchung der Auswirkungen verschiedener Parametereinstellungen (maxSteps, walkProb) von GSAT, GHC und GSHC auf die Performanz der Verfahren.

Bezüglich der Formelstruktur der kodierten HCP-Instanzen konnten folgende Resultate erzielt werden:

- Iwama-Miyazaki-kodierte HCP-Instanzen besitzen eine Formelstruktur, die für GSAT besonders ungünstig ist (die für die Performanz bedeutsame „score-by-need"-Strategie ist hier aufgrund von Variablenabhängigkeiten nicht effizient).
- Mithilfe der Methode der impliziten Klauselkodierung (motiviert durch ein von Selman & al. in anderem Kontext verwendetes Konzept) kann der durch die Kodierung verursachte und für die Praxis problematische, kubische Platzbedarf für GSAT vermieden werden.

Aufgrund dieser Resultate wurde das allgemeine GSAT-Verfahren in zwei Schritten an die spezielle Struktur von SAT-kodierten HC-Problemen angepaßt; hieraus entstanden die Algorithmen GHC und GSHC. Dabei basiert GHC auf einer Idee (Erweiterung der Nachbarschaftsrelation), die prinzipiell auf den Fall allgemeiner SAT-Probleme übertragbar ist. GSHC ist dagegen ein sehr eng am HC-Problem orientiertes Verfahren.

Die Untersuchung des Performanzverhaltens dieser Verfahren und verschiedener GSAT-Varianten auf SAT-kodierten HC-Problemen ergab folgende Resultate:

- Beim Performanz-Vergleich der Verfahren auf identischen Mengen von Testproblemen liegt GSHC vor GHC und GHC deutlich vor GSAT.
- Obwohl es sich bei der betrachteten Problemklasse um stark strukturierte Probleme handelt, konnte weder für die Verwendung von adaptiven Klauselgewichten („clause weights") noch für die eingeschränkte Resolventen-Erweiterung (RBR) der SAT-kodierten HCP-Instanzen ein signifikanter Einfluß auf die Performanz von GSAT nachgewiesen werden.

- Die Random-Walk-Strategie hat hingegen einen qualitativen Einfluß auf die Performanz aller betrachteten Verfahren. Bei festem Wert für den Parameter maxSteps konnten für alle untersuchten Verfahren optimale walkProb-Werte auf den jeweiligen Problemklassen bestimmt werden.

- Für alle Verfahren konnte auf HC_2-Problemen (bei optimaler Einstellung des walkProb-Parameters) der gleiche, problemgrößenunabhängige Zusammenhang zwischen der maximalen Anzahl der Schritte (maxSteps) und der Erfolgsquote festgestellt werden (sr/ms-Charakteristik).

- Diese sr/ms-Charakteristik kann durch eine einfache Funktion approximiert werden. Auf diese Weise ist einerseits eine Prognose der Erfolgsraten für beliebige maxSteps-Werte möglich; andererseits kann für jede vorgegebene Erfolgrate der benötigte maxSteps-Wert ermittelt werden.

- Aufgrund dieser Charakteristik sind GSAT, GHC und GSHC approximativ vollständig in dem Sinne, daß lösbare HCP-Instanzen mit beliebig hoher Wahrscheinlichkeit gelöst werden können (experimentelles Resultat). Für HC_1-Probleme wurde gezeigt, daß diese Eigenschaft nur mithilfe der Random-Walk-Strategie erreicht werden kann.

- Die sr/ms-Charakteristik ist außerdem so beschaffen, daß in einem weiten Bereich eine Verdopplung des maxSteps-Werts die gleiche Wirkung zeigt, wie zweimalige Ausführung des Verfahrens. Daraus kann geschlossen werden, daß in diesem Bereich der Ausgangspunkt der lokalen Suche keinen wesentlichen Einfluß auf die Erfolgsrate hat (in dieser Situation bringen folglich Optimierungen bei der Initialisierung der lokalen Suche keinen Vorteil).

- Bei optimaler walkProb-Einstellung ergibt sich aus den experimentellen Daten eine exponentielle Abhängigkeit des für eine beliebig, aber fest gewählte Erfolgsquote benötigten maxSteps-Wertes von der Problemgröße. Die hieraus gewonnenen Prognosen über die Lage der sr/ms-Charakteristiken für beliebige Problemgrößen konnten an Stichproben bestätigt werden.

- Aus diesem Zusammenhang kann unter Verwendung der sr/ms-Charakteristik der Verlauf der Erfolgsquote für beliebige Problemgrößen bei festem maxSteps-Wert approximiert werden.

Auf der Grundlage dieser Resultate ist die Laufzeit der untersuchten Verfahren für gegebene Problemgröße und Erfolgsquote nach oben abschätzbar; daraus kann eine Abschätzung der maximalen Größe praktisch lösbarer Probleme gewonnen werden.

Insgesamt deuten die experimentellen Resultate darauf hin, daß die Iwama-Miyazaki-Kodierung von HCP-Instanzen trotz der geringen Anzahl der benötigten Aussagenvariablen für die praktische Lösung von HC-Problemen mithilfe GSAT-ähnlicher Verfahren kaum geeignet ist. Die Ursache hierfür liegt vermutlich in der Struktur der erzeugten Formeln; auch durch besondere Maßnahmen wie RBR-Erweiterung oder adaptive Klauselgewichte, mit deren Hilfe die Performanz GSAT-ähnlicher Verfahren für andere Klassen strukturierter SAT-Probleme erheblich verbessert werden kann, läßt sich für die betrachteten Problemklassen und Lösungsverfahren offensichtlich keine signifikante Verbesserung erzielen. Dies kann als Indiz dafür aufgefaßt werden, daß es sich bei den Iwama-Miyazaki-kodierten HC-Problemen um eine Klasse von SAT-Problemen handelt, welche für GSAT-ähnliche lokale Suchverfahren eine extreme Schwierigkeit aufweisen; folglich bietet sich deren Verwendung als Benchmark-Probleme zur Bewertung von SAT-Verfahren an.

Obwohl die qualitative Abstufung der Schwierigkeit der einzelnen Problemklassen zunächst für alle Algorithmen der Erwartung entspricht (die prinzipiell linear lösbaren HC_1-Probleme sind wesentlich leichter lösbar als die $\mathcal{NP}$-harten HC_2-Probleme), zeigen die erzielten Resultate deutliche Hinweise darauf, daß die untersuchten Algorithmen (zusammen mit der Iwama-Miyazaki-Kodierung) nicht als problemspezifisch adäquate Verfahren für das HCP (und damit auch nicht als adäquate Verfahren für das allgemeine SAT-Problem) betrachtet werden können.

In Kapitel 9 wurde (ausgehend von verschiedenen Resultaten) gezeigt, wie das $\mathcal{NP}$-harte Anordnungsproblem bei der STS-Methode zur physikalischen Genomkartierung durch eine Modifikation der Iwama-Miyazaki-Transformation variableneffizient auf ein MaxSAT-Problem reduziert werden kann, welches prinzipiell mithilfe einfacher Erweiterungen der zuvor betrachteten SAT-Verfahren lösbar ist. Aufgrund der Resultate zur Anwendbarkeit von GSAT-Varianten auf Iwama-Miyazaki-kodierte HCP-Instanzen erscheint jedoch die praktische Anwendbarkeit von SAT-Verfahren zur Lösung dieses molekularbiologischen Problems eher fraglich.

Offene Fragen und Ausblick

Aus den im Rahmen dieser Arbeit erzielten Resultaten ergibt sich eine Fülle von weiterführenden Fragestellungen. Einige der interessantesten unter diesen sind im folgenden zusamengestellt:

- Läßt sich für den Lösungsaufwand (Schritte pro erfolgreichem Versuch) eine ähnliche Charakteristik wie für die Erfolgsquote nachweisen? (In diesem Falle wären zuverlässige Prognosen für die reale Laufzeit des Verfahrens möglich.)

- Ist es möglich, funktionale Zusammenhänge für Erfolgsquote und/oder Lösungsaufwand bei variabler Random-Walk-Wahrscheinlichkeit zu finden? (In diesem Falle könnte das Verhalten der HC_1-Verfahren im gesamten Parameterraum prognostiziert werden.)

- Auf welche Weise können die beobachteten funktionalen Abhängigkeiten zwischen Performanz und Parametrisierung der betrachteten Verfahren theoretisch begründet werden und welche Schlüsse lassen sich ggf. daraus ziehen?

- Sind die zentralen Methoden und Ergebnisse der vorliegenden Untersuchung auf andere Problemklassen (Formelklassen) übertragbar? (Die Untersuchung dieser Fragestellung könnte dabei helfen, aus den beobachteten Resultaten auf grundlegende Eigenschaften bestimmter Kodierungsverfahren bzw. SAT-Algorithmen zu schließen.)
- Welches Performanzverhalten zeigen vollständige SAT-Verfahren (z.B. die neueren, effizienten Varianten des Davis-Putnam-Verfahrens) auf SAT-kodierten HC-Problemen? (Bislang ist weitgehend unklar, für welche Problemklassen lokale Suchverfahren wie GSAT „klassischen" Backtrack-Verfahren ebenbürtig oder überlegen sind.)
- Welche Ergebnisse lassen sich auf der Grundlage der nun bekannten Eigenschaften der Kodierung nach Iwama & Miyazaki für die ursprünglich betrachtete, molekularbiologisch motivierte Aufgabenstellung und andere Optimierungsprobleme aus dem Anwendungsbereich erzielen?

Die genannten Fragestellungen berühren zwei hoch aktuelle Bereiche, die innerhalb der Intellektik von erheblicher Bedeutung sind: Entwicklung und Analyse leistungsfähiger und universeller SAT-Verfahren sowie die Untersuchung effizienter Transformationen von praktisch relevanten Anwendungsproblemen auf allgemeine logische Problemstellungen wie SAT. Die vorliegende Arbeit zeigt, daß SAT-Algorithmen und SAT-Transformationsverfahren nicht isoliert voneinander betrachtet werden sollten, da Eigenschaften des Transformationsverfahren (über die Struktur der erzeugten Formeln) die Performanz der eingesetzten SAT-Algorithmen unter Umständen erheblich beeinflussen. Aufgrund der Tatsache, daß zu den Eigenschaften der meisten SAT-Verfahren kaum theoretische Ergebnisse vorliegen, erscheint es besonders interessant, die in dieser Arbeit gezeigten Eigenschaften von GSAT-Varianten genauer zu untersuchen. Dabei besteht meines Erachtens durchaus die Hoffnung, auf der Grundlage einer genaueren Charakterisierung und theoretischen Fundierung dieser Eigenschaften zu Verbesserungen der untersuchten Verfahren zu gelangen.

Ein wichtiger, hier kaum berührter Aspekt für die Beurteilung von SAT-Algorithmen ist deren Verhalten auf unerfüllbaren Formeln. Man sollte erwarten, durch ein genauere Untersuchung von Struktureigenschaften des Raums aller unerfüllbaren Formeln (oder — in der dualen Sicht — der Tautologien) weitreichende Ergebnisse erzielen zu können. Neuere Resultate legen nahe, daß auf diese Weise allgemeine, mächtige Kriterien zur Beurteilung von Verfahren zur Lösung aussagenlogischer Probleme erhalten werden können (vergl. [BE94, Bib94]).

Letztendlich stellt sich die Frage, inwieweit Kombinationen aus problemspezifischen Verfahren zur SAT-Transformation und allgemeinen SAT-Verfahren gefunden werden können, welche als adäquate Lösungsverfahren für bestimmte Anwendungsprobleme angesehen werden können. Statt des SAT-Problems kann man in einer allgemeineren Formulierung auch andere universelle logische Problemstellungen als Ziel der Transformation betrachten. Obwohl diese Fragestellung zur Zeit lediglich ansatzweise untersucht werden kann, ist sie doch von besonderer Bedeutung für die Einsetzbarkeit der in der Intellektik untersuchten universellen, logik-basierten Problemlösungsmechanismen in den verschiedensten Anwendungsbereichen.

— FiniS —

Anhang

Dieser Anhang enthält die in Kapitel 8 vorgestellten und diskutierten experimentellen Ergebnisse in Form von Diagrammen.

In den Diagrammen werden folgende Bezeichnungen verwendet:

n	Problemgröße (Knoten im Graphen)
ms	maximale Anzahl der Schritte pro Versuch (Parameter maxSteps)
wp	Walk-Wahrscheinlichkeit (Parameter walkProb)
sr	Erfolgsrate (Anzahl Lösungen / Anzahl Versuche) in %
ss	mittlere Anzahl der Schritte pro erfolgreichem Versuch
$fp(x)$	$= 100 \cdot (1 - 2^{-x})$ (Approximationsfunktion für die sr/ms-Charakteristik)

Soweit nicht explizit anders angegeben, wurden die Versuche mit Random-Walk unter Verwendung der für das jeweilige Verfahren und die Problemklasse als optimal ermittelten Walk-Wahrscheinlichkeit (siehe Abschnitt 8.7) ausgeführt.

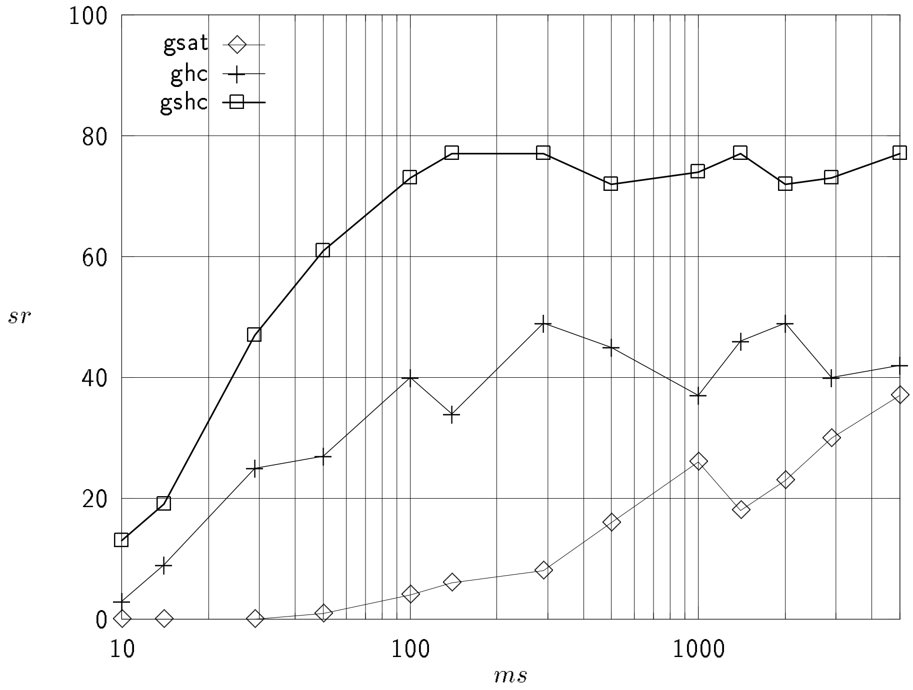

Abb.1-1a: Erfolgsquote vs. maxSteps für HC_1-10, Verfahren ohne Random Walk

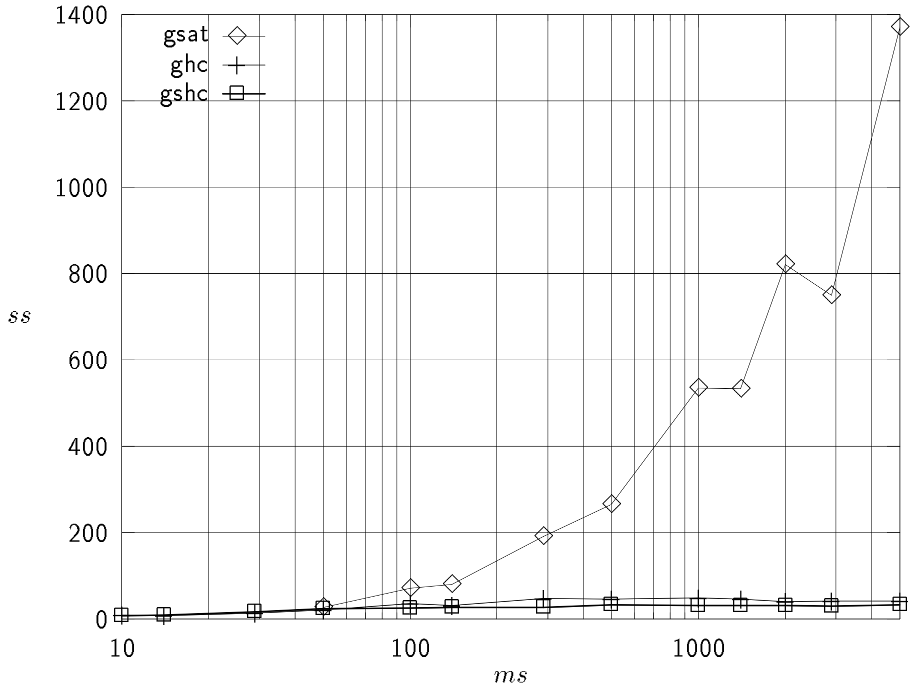

Abb.1-1b: Lösungsaufwand vs. maxSteps für HC_1-10, Verfahren ohne Random Walk

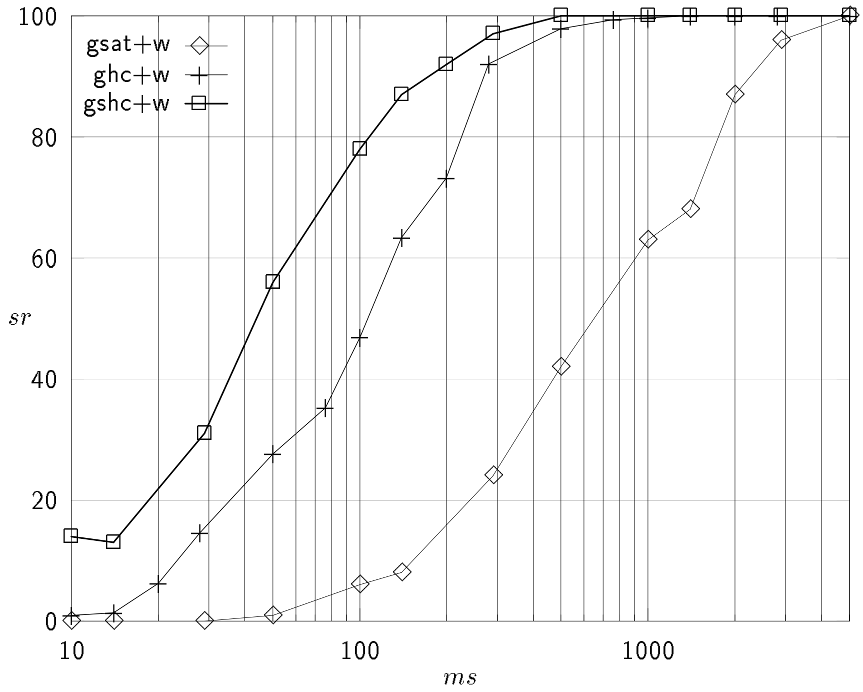

Abb.1-2a: Erfolgsquote vs. maxSteps für HC_1-10, Verfahren mit Random Walk

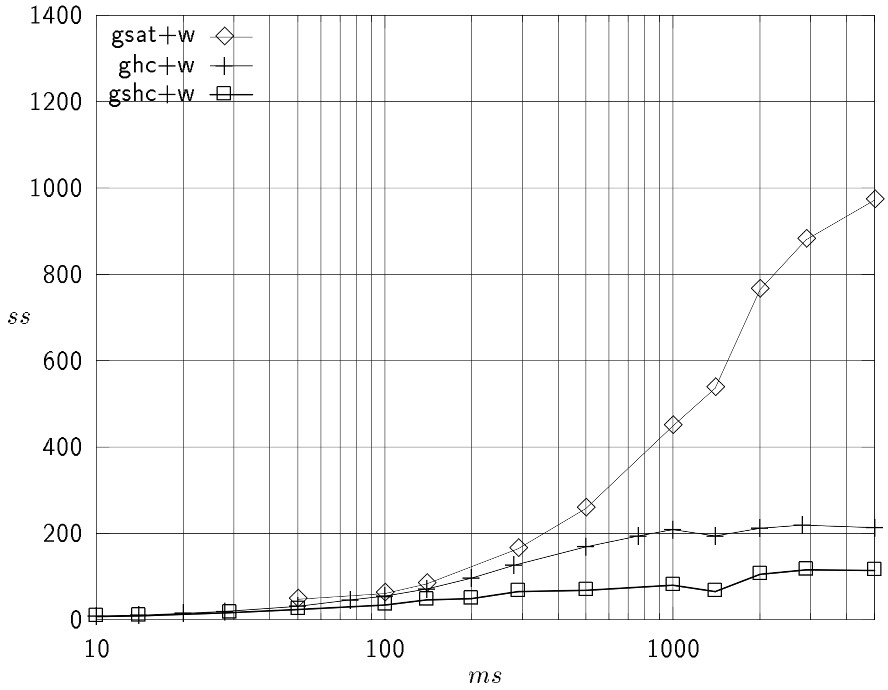

Abb.1-2b: Lösungsaufwand vs. maxSteps für HC_1-10, Verfahren mit Random Walk

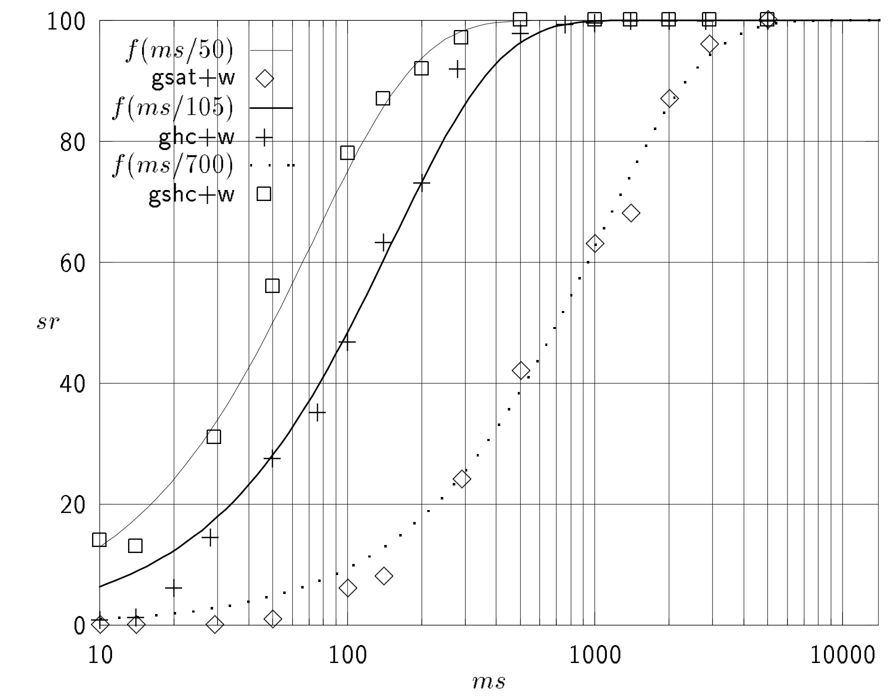

Abb.1-3: Approximation der Erfolgsquote vs. maxSteps für HC_1-10

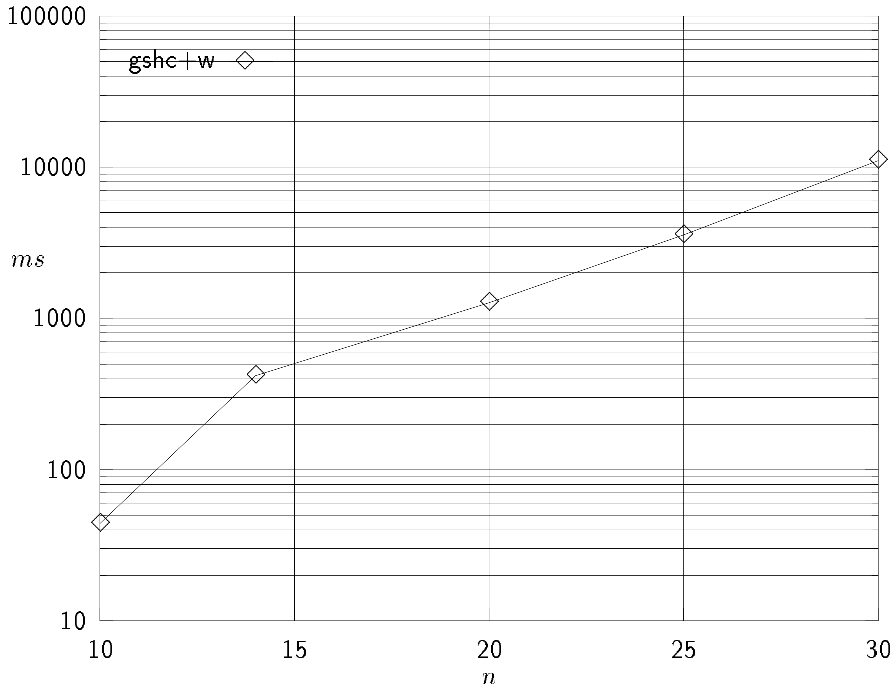

Abb.1-4: Für eine Erfolgsquote von 50% erforderlicher maxSteps-Wert vs. Problemgröße für HC_1-n

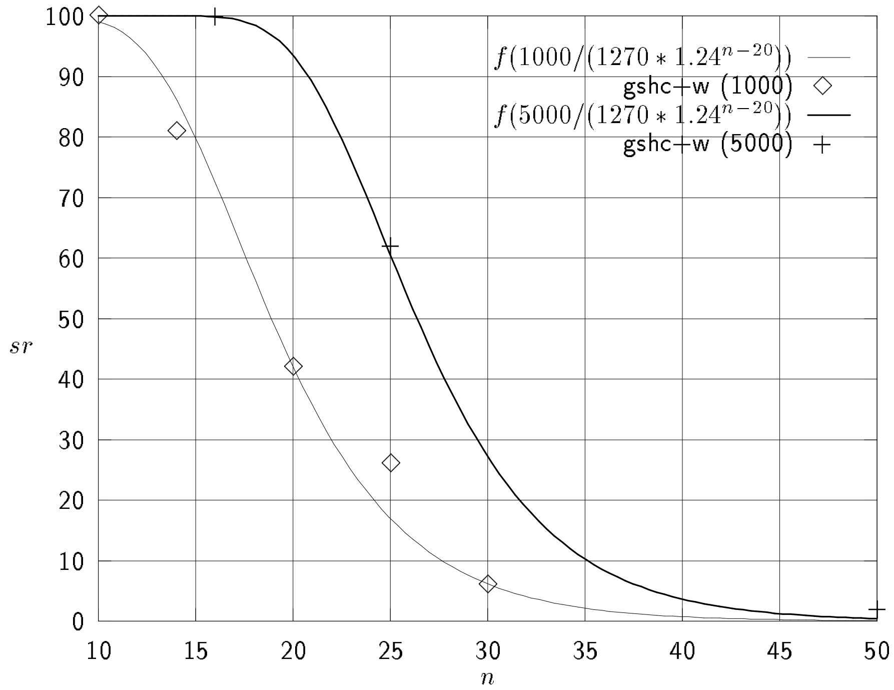

Abb.1-5: Approximation der Erfolgsquote vs. Problemgröße für HC_1-n bei festem maxSteps von 1000 bzw. 5000

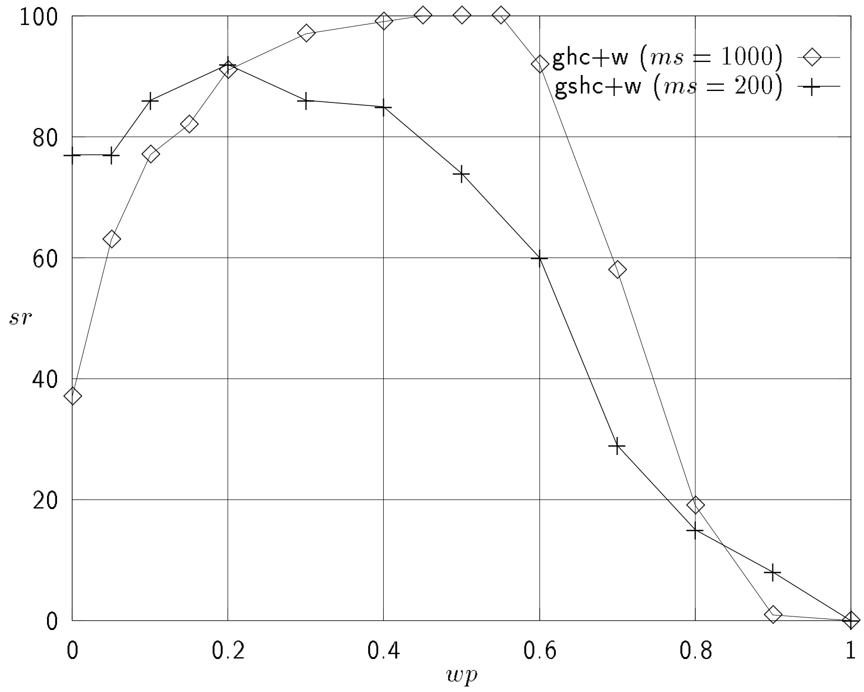

Abb.1-6a: Erfolgsquote vs. walkProb für HC_1-10, maxSteps fest

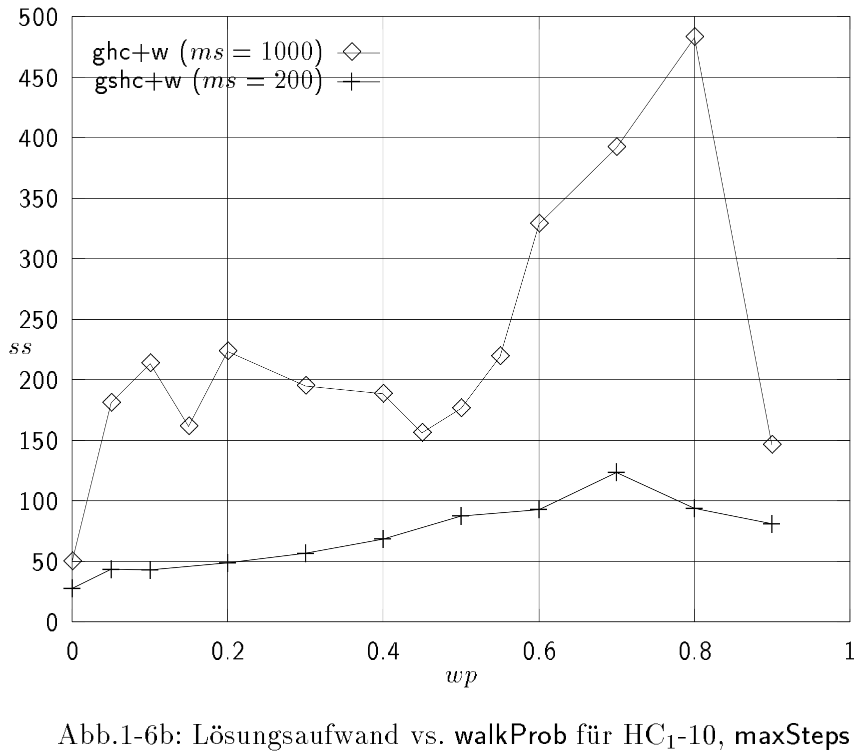

Abb.1-6b: Lösungsaufwand vs. walkProb für HC_1-10, maxSteps fest

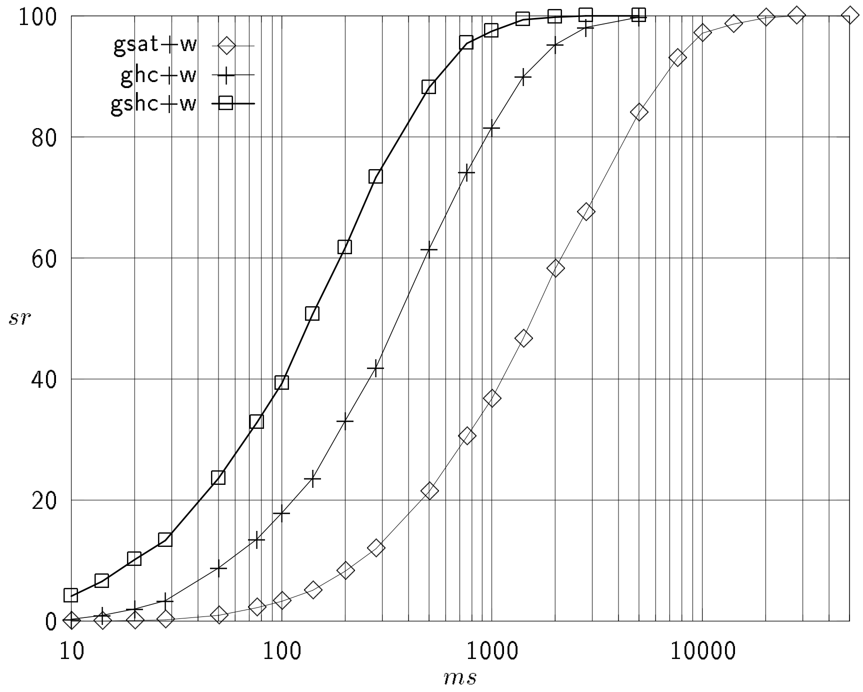

Abb.2-2a: Erfolgsquote vs. maxSteps für HC_2-10, Verfahren mit Random Walk

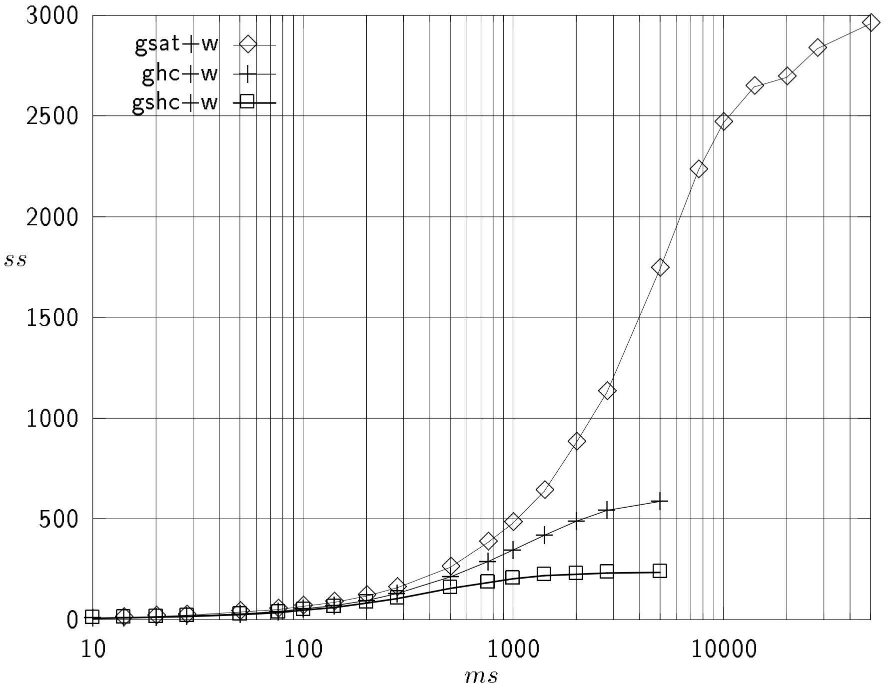

Abb.2-2b: Lösungsaufwand vs. maxSteps für HC_2-10, Verfahren mit Random Walk

gsat+w

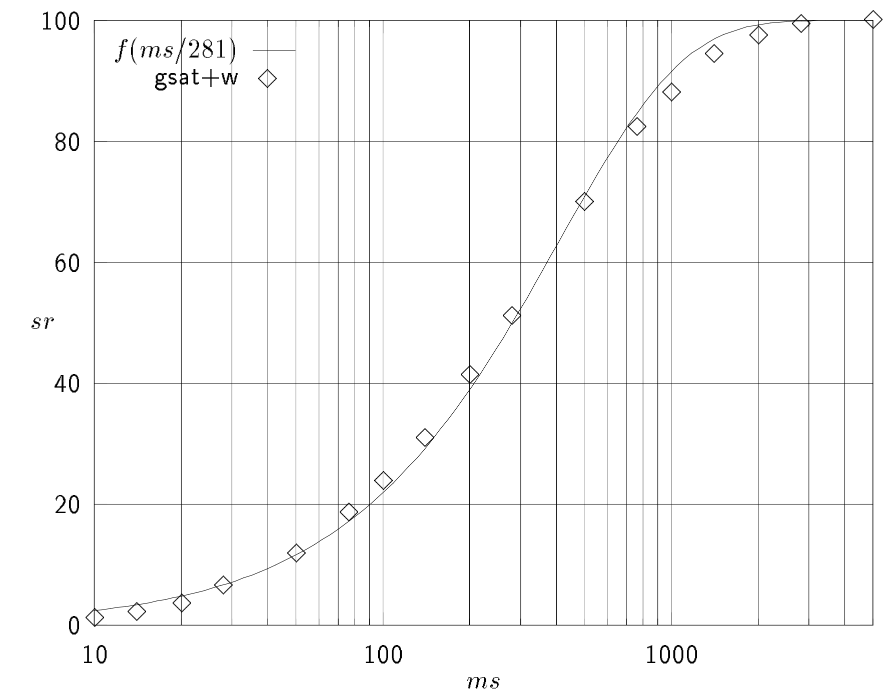

Abb.2-2c: Erfolgsquote vs. maxSteps für HC_2-8

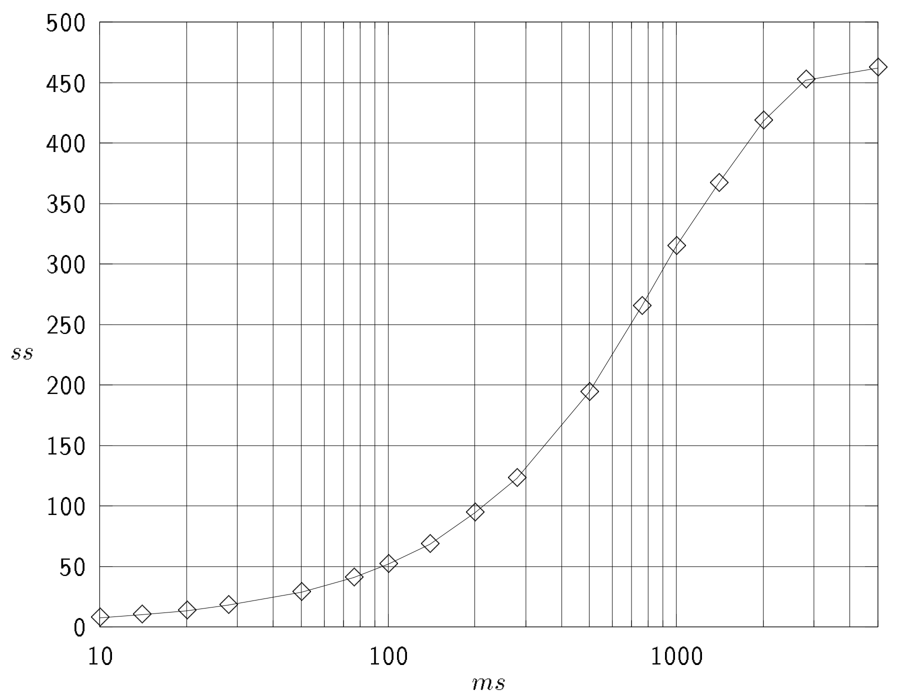

Abb.2-2d: Lösungsaufwand vs. maxSteps für HC_2-8

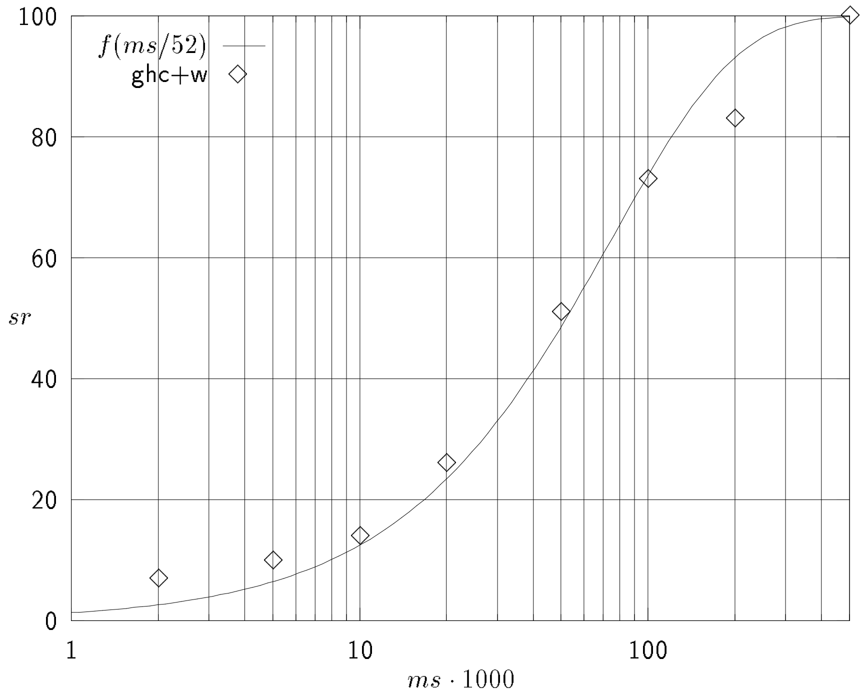

Abb.2-2e: Erfolgsquote vs. maxSteps für HC_2-20

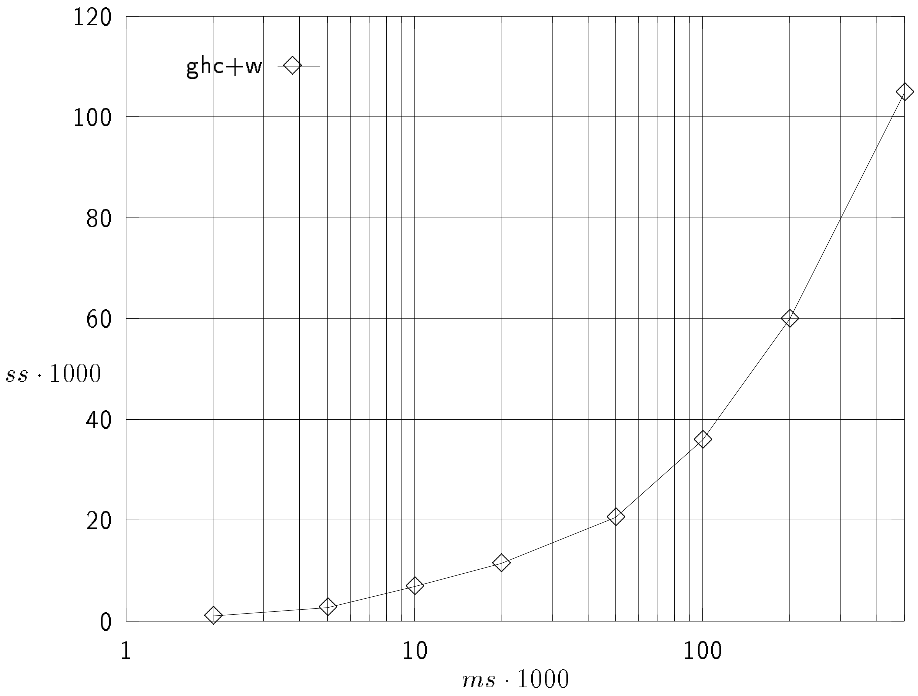

Abb.2-2f: Lösungsaufwand vs. maxSteps für HC_2-20

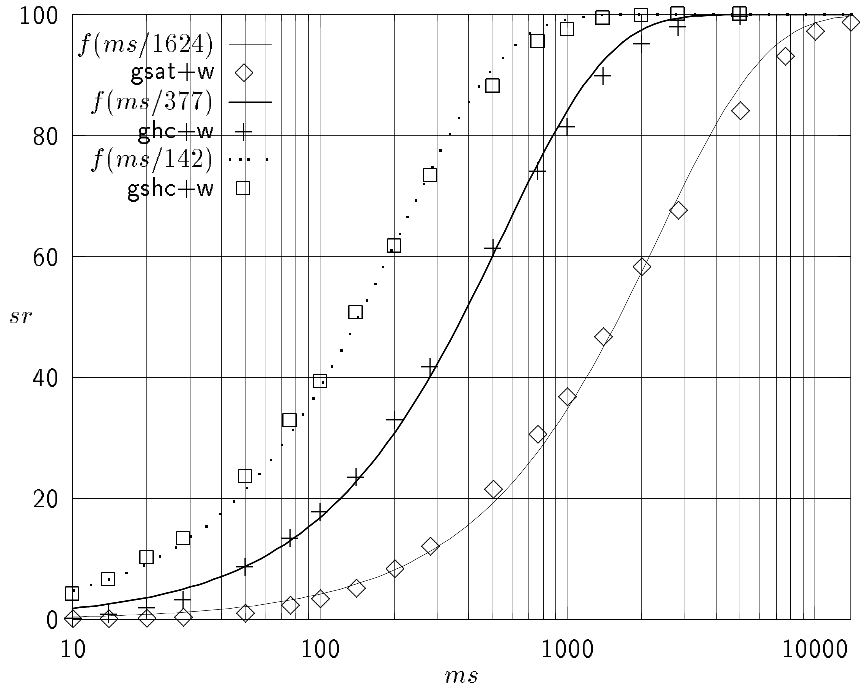

Abb.2-3: Approximation der Erfolgsquote vs. maxSteps für HC_2-10

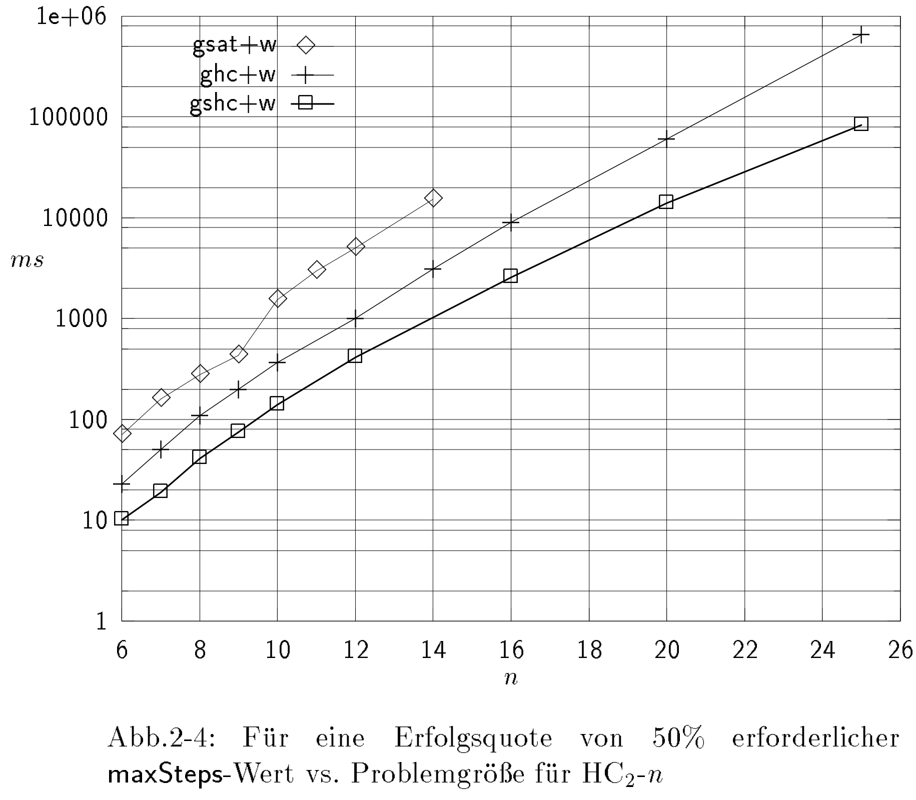

Abb.2-4: Für eine Erfolgsquote von 50% erforderlicher maxSteps-Wert vs. Problemgröße für HC_2-n

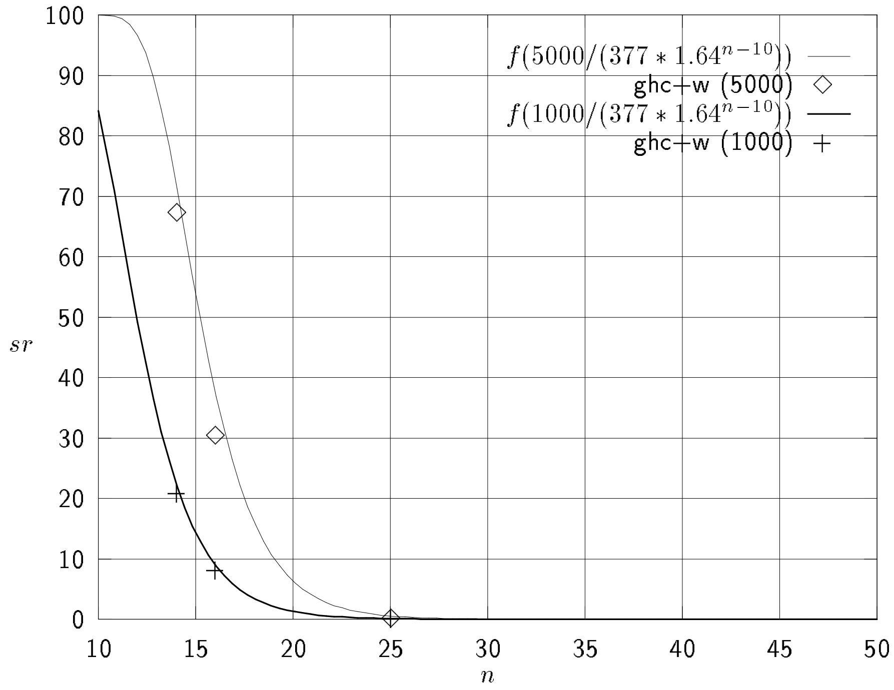

Abb.2-5: Approximation der Erfolgsquote vs. Problemgröße für HC_2-n bei festem maxSteps von 1000 bzw. 5000

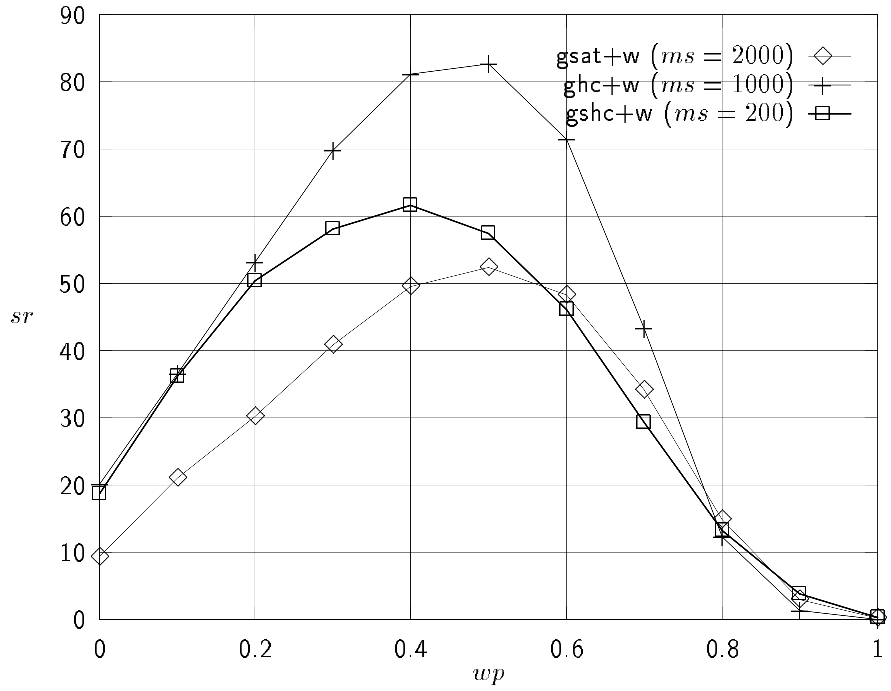

Abb.2-6a: Erfolgsquote vs. walkProb für HC_2-10, maxSteps fest

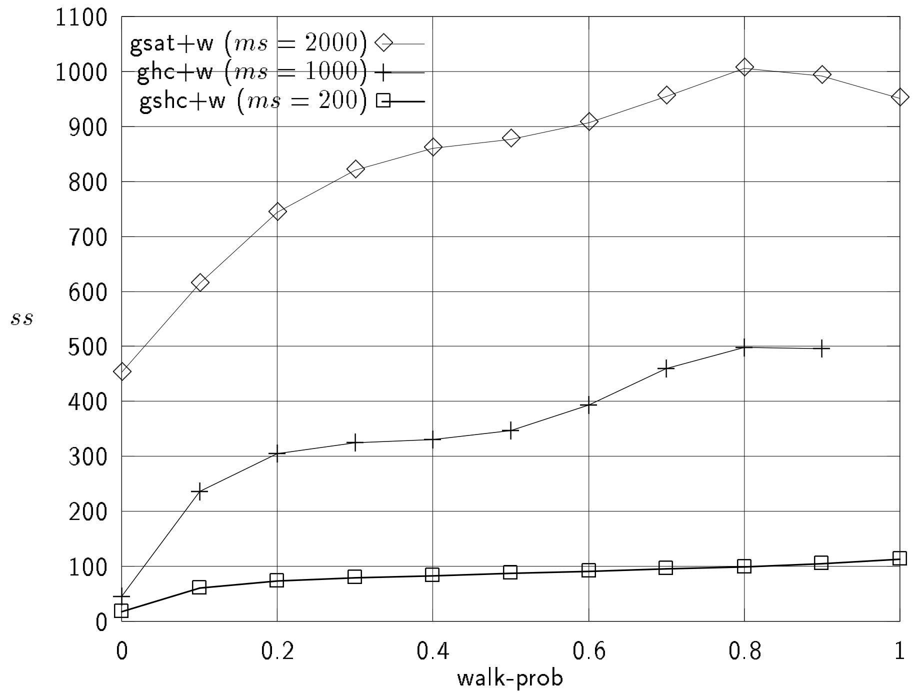

Abb.2-6b: Lösungsaufwand vs. walkProb für HC_2-10, maxSteps fest

Literaturverzeichnis

[Adl94] L.M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

[Adl95] L.M. Adleman. On constructing a molecular computer. Unpublished Manuscript, January 1995.

[AIM93] Yuichi Asahiro, Kazuo Iwama, and Eiji Miyano. Random generation of test instances with controlled attributes. In *2nd Dimacs Challenge Workshop*, 1993.

[AKWZ94] Farid Alizadeh, Richard M. Karp, Deborah K. Weiser, and Geoffry Zweig. Physical mapping of chromosomes using unique probes. In *ACM-SODA*. ACM, 1994.

[BAH+94a] A. Beringer, G. Aschemann, H.H. Hoos, M. Metzger, and A. Weiß. Gsat versus simulated annealing – a comparison. Technical Report AIDA-94-01, TH Darmstadt, 1994.

[BAH+94b] A. Beringer, G. Aschemann, H.H. Hoos, M. Metzger, and A. Weiß. GSAT versus Simulated Annealing. In A. G. Cohn, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 130–134. John Wiley & Sons, 1994.

[BE94] W. Bibel and E. Eder. Decomposition of tautologies into regular formulas and strong completeness of connection-graph resolution. *Journal of the ACM*, submitted, 1994.

[Bib92] Wolfgang Bibel. *Deduktion: Automatisierung der Logik*. Oldenbourg Verlag, München, 1992.

[Bib94] Wolfgang Bibel. Structure in the space of tautologies. Unpublished Manuscript, October 1994.

[BL93] N. Boissin and J.-L. Lutton. A parallel simulated annealing algorithm. *Parallel Computing*, 19:859–872, 1993.

[BS79] I.N. Bronstein and K.A. Semendjajew. *Taschenbuch der Mathematik*. B. G. Teubner Verlag, Stuttgart and Nauka Verlag, Moskau, 24th edition, 1979.

[CA93] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 21–27. MIT press, 1993.

[CB94] James M. Crawford and Andrew B. Baker. Experimental results on the application of satisfiability. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 1092–1097 (Volume 2). AAAI Press / The MIT Press, 1994.

[CI95] Byungki Cha and Kazuo Iwama. Performance test of local search algorithms using new types of random CNF formulas. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 304–310, 1995.

[CL88] S.A. Cook and M. Luby. A simple parallel algorithm for finding a satisfying truth assignment to a 2-CNF formula. *Information Processing Letters*, 27:141–145, 1988.

[DABC93] O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. SAT versus UNSAT. In *2nd DIMACS Implementation Challenge*, 1993.

[DG84] W. F. Dowling and J. H. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.

[DLL62] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[DP60] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[Fin95] Andreas Fink. Algorithmische Methoden zur Kartierung von DNA-Sequenzen. Diplomarbeit, TH Darmstadt, 1995.

[GG84] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[Gla63] R.J. Glauber. Time-dependent statistics of the ising model. *Journal of mathematical physics*, 4:294–307, 1963.

[GM94] M. Ginsberg and D. McAllester. GSAT and dynamic backtracking. In *Proceedings of the International Conference on Principles of Knowlege Representation and Reasoning*, pages 226–237, 1994.

[Gu95] Jun Gu. Parallel algorithms for satisfiability (sat) problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 22:105–161, 1995.

[GW93] Ian P. Gent and Toby Walsh. Towards an understanding of hill–climbing procedures for SAT. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 28–33. MIT press, 1993.

[HD94] Jin-Kao Hao and Raphael Dorne. A new poulation-based method for satisfiability problems. In A.G. Cohn, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 136–139. John Wiley & sons, 1994.

[HHSW94] Steffen Hölldobler, Holger H. Hoos, Antje Strohmaier, and Andreas Weiß. The GSAT/SA-Familiy – Relating greedy satisifability testing to simulated annealing. Technical Report AIDA-94-17, TH Darmstadt, 1994.

[HW92] Bernhard Heinemann and Klaus Weihrauch. *Logik für Informatiker*. Teubner Verlag, Stuttgart, 2nd edition, 1992.

[HW93] Tad Hogg and Colin P. Williams. Solving the really hard problems with co-operative search. In *Working notes of the AAAI Spring Symposium Series, AI and NP–Hard Problems*, pages 78–84, 1993.

[HW94] Tad Hogg and Colin P. Williams. Expected gains from parallelizing constraint solving for hard problems. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, volume 1, pages 331–336, 1994.

[IM94] Kazuo Iwama and Shuichi Miyazaki. SAT-variable complexity of hard combinatorial problems. In *Proceedings of the World Computer Congress of the IFIP*, pages 253–258 (Volume 1). Elsevier Science B.V., North Holland, 1994.

[JKS95] Yuejun Jiang, Henry Kautz, and Bart Selman. Solving problems with hard and soft constraints usings a stochastic algorithm for MAX-SAT. In *1st International Workshop on Artificial Intelligence and Operations Research*, 1995.

[Jun90] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen*. B.I. Wissenschaftsverlag, 2nd edition, 1990.

[Kar93] Richard M. Karp. Mapping the genome: Some combinatorial problems arising in molecular biology. In *25th ACM-STOC*, pages 279–285, 1993.

[KD95] Kalev Kask and Rina Dechter. SAT and local consistency. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 612–622, 1995.

[KGV82] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. Technical Report Research Report RC 9335, IBM Thomas j. Watson Center, Yorktown Heights, N.Y., 1982.

[Kon94] Kurt Konolige. Easy to be hard: difficult problems for greedy algorithms. In *Proceedings of the International Conference on Principles of Knowlege Representation and Reasoning*, pages 374–378, 1994.

[KR90] Richard M. Karp and Vijaya Ramachandran. Parallel Algorithms for Shared-Memory Machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 17, pages 869–941. Elsevier Science, 1990.

[KS92] Henry Kautz and Bart Selman. Planning as satisfiability. In *Proceedings of the European Conference on Artificial Intelligence*, pages 359–363, 1992.

[Lip94] Richard J. Lipton. Speeding up computations via molecular biology. Unpublished Manuscript, December 1994.

[Lip95] Richard J. Lipton. DNA solution of hard computational problems. *Science*, 268:525–545, 1995.

[Lov78] D.W. Loveland. *Automated theorem proving: A logical basis*. North Holland, Amsterdam, 1978.

[Meh84] Kurt Mehlhorn. *Data-Structures and Algorithms 2: Graph Algorithms and NP-Completeness*. Springer-Verlag, Berlin, 1984.

[Mor93] Paul Morris. The breakout method for escaping from local minima. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 40–45. MIT press, 1993.

[MRR+53] W. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of states calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[Pin92] Gadi Pinkas. *Logical Inference in Symmetric Connectionist Networks*. PhD thesis, Washington University, Sever Institute of Technology, St. Louis, Missouri, 1992.

[Rei90] Karl Rüdiger Reischuk. *Einführung in die Komplexitätstheorie*. Teubner Verlag, Stuttgart, 1990.

[Rei94] Gerhard Reinelt. *The Traveling Salesman — Computational Solutions for TSP Applications*. Springer Verlag, 1994.

[Sel94] Bart Selman. Domain-specific complexity tradeoffs. In A.G. Cohn, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 416–420. John Wiley & sons, 1994.

[SK93] Bart Selman and Henry A. Kautz. An empirical study of greedy local search for satisfiability testing. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 46–51. MIT press, 1993.

[SKC93] Bart Selman, Henry A. Kautz, and B. Cohen. Local search strategies for satisfiabiliy testing. Technical report, AT&T Bell Laboratories, Murray Hill, NJ, 1993.

[SKC94] Bart Selman, Henry A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, volume 1, pages 337–343, 1994.

[SLM92] Bart Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 440–446. MIT press, 1992.

[Spe93] W.M. Spears. Simulated Annealing for hard satisfiability problems. Technical report, Naval Research Laboratory, Washington D.C., 1993.