Hochschule Aachen

JÜLICH
FORSCHUNGSZENTRUM

”Most propable pathes” - Analyse an plasma-chemischen Modellen

Bachelorarbeit
im Studiengang Scientific Programming
Fachbereich 9 - Medizintechnik und Technomathematik
Fachhochschule Aachen
12.08.2009

vorgelegt von
Oliver Schmidt

Referent:	Prof. Dr. Christof Schelthoff
Korreferent:	Bettina Küppers

Inhaltsverzeichnis

Abbildungsverzeichnis III

Tabellenverzeichnis IV

1 Einleitung 1

2 Problemstellung 3
2.1 Grundlage für das Modell . . . 3
2.2 Umsetzung des Modells im Computer . . . 5
2.3 Einführung zu Graphen . . . 6

3 Grundlagen zur Berechnung 8

4 Ausgangssituation 12

5 Optimierungen aufgrund der Ausgangssituation 15
5.1 Optimierung durch Einführung einer unteren Wahrscheinlichkeitsschranke 16
5.2 Optimierung der Wahrscheinlichkeits-Berechnung . . . 18
5.3 Versuch einer Zwischenspeicherung . . . 19
5.4 Versuch eines Abweich-Kriteriums . . . 20
5.5 Optimierung durch Beschränkung auf n Pfade . . . 21
5.6 Optimierung der Rekursion . . . 22

6 Vollständig rekursiver Ansatz 26
6.1 Begründung des Ansatzwechsels . . . 26
6.1.1 Versuch der Optimierung durch Verwerfen von Kanten . . . 28
6.2 Beschreibung und Vergleich der Ansätze . . . 29
6.3 Optimierung durch Zwischenspeicherung . . . 32

7 Probleme, Ausblick 36
7.1 Versuch der Optimierung durch Sortieren . . . 36
7.2 Zusammenfassung und Ausblick . . . 39

Abbildungsverzeichnis

2.1 Beispiele für Graphen 6

3.1 Graph der vollständigen Ionisierung von CH 9
3.2 Wahrscheinlichkeiten der Zerlegungen von CH_4, [?] 11

4.1 Struktogramm des ursprünglichen Programmablaufes 14

5.1 Auftragung der Einzelwahrscheinlichkeiten über der x-Achse 21
5.2 Vergleich der Optimierungen „untere Wahrscheinlichkeitsschranke" und „Beschränkung auf n Pfade" 23
5.3 Vergleich der Optimierungen „Beschränkung auf n Pfade" und Rekursions-Optimierung 24

6.1 Faktor c zwischen Knoten und Kanten, aufgetragen nach Gruppen CH_X, C_2H_X, C_3H_X 26
6.2 Vergleich der Laufzeiten des neuen Ansatzes mit der Rekursions-Optimierung des alten Ansatzes 31
6.3 Vergleich der Laufzeiten des neuen Ansatzes und der Optimierung des Ansatzes durch Zwischenspeicherung 33

7.1 Vergleich der Laufzeiten des neuen Ansatzes und der beiden Optimierungen des Ansatzes 37
7.2 Beispielgraph für die Konditionierung der Wahrscheinlichkeiten 38

Tabellenverzeichnis

3.1 Anzahl der Knoten und Kanten der Fragmentierungen von Kohlenwasserstoffen (ohne ionisierte), nur Standard-Konfiguration 10

5.1 Benchmark für die Einführung der unteren Wahrscheinlichkeitsschranke . 18
5.2 Benchmark für Optimierung „Einbettung der Wahrscheinlichkeitsberechnung" 19
5.3 Benchmark für Optimierung „Beschränkung auf n Pfade" 22
5.4 Benchmark für Optimierung der Rekursion 24

6.1 Faktor c zwischen Knoten und Kanten 27
6.2 Anzahl der Kanten-Updates in representativen Fällen 29
6.3 Benchmark zum vollständig rekursiven Ansatz 30
6.4 Benchmark der Optimierung durch Zwischenspeicherung 34

7.1 Benchmark der Optimierung durch Sortierung 36

1 Einleitung

Bereits seit Mitte des 20ten Jahrhunderts gehen Wissenschaftler und Forscher der Frage nach, ob es möglich ist, mittels Kernfusion eine großtechnische Lösung zur Energieversorgung zu gewinnen. Dafür werden Experimente mit teilweise erheblichen finanziellen und materiellen Ausmaßen durchgeführt. Die Klärung der Frage, ob es mit Plasmareaktoren auf der Erde möglich ist, eine positive Energiebilanz zu erhalten, steht dabei im Mittelpunkt. Erreicht werden soll dies durch ein ca. 10^8 °C heißes Wasserstoffplasma in Nachbildung der Bedingungen und Reaktionen, die im Inneren der Sonne und Sonnen anderer Sternensysteme stattfinden. Diese Plasmen beschäftigen Forscher in aller Welt, aktuell zum Beispiel in dem internationalen Forschungsprojekt *ITER*, dem ersten Versuch eines Plasmareaktors in Kraftwerksgröße (Baubeginn: 2008, in Cadarache, Frankreich).

Im *Institut für Energieforschung 4*, ehemals *Institut für Plasmaphysik*, des *Forschungszentrums Jülich* ist die Wechselwirkung des Plasmas mit den Wänden von Plasmareaktoren ein Gegenstand der Forschung. Die Forschung zum Wandaufbau wird in der theoretischen Abteilung ausschließlich mittels Computer-Simulation und Berechnungen unterstützt, die experimentelle Verifikation findet in anderen Abteilungen oder bei Forschungspartnern statt.

Kohlenstoff als gut bekanntes Material mit großer Wärmeresistenz wird im Institut darauf untersucht, ob es das geeignetste Material zur Beschichtung der besonders stark exponierten, aus Edelstahl hergestellten Bauteile ist, oder ob ein neues Material entwickelt werden muss [**?**].

Die derzeit untersuchten Plasmareaktoren basieren auf Plasmen, die aus Wasserstoff-Nukliden (Deuterium und Tritium) bestehen, da dieses Konzept das Erfolg versprechenste ist (größte Wirkungsquerschnitte für Fusion bei vergleichsweise niedrigen Temperaturen) [**?**]. Da ein Kontakt des Plasmas mit der Wand nicht vollständig ausgeschlossen werden kann, ist es notwendig, Reaktionen von Wand und Plasma zu untersuchen [**?**]. Wie verhalten sich die aus diesen Prozessen entstehenden Kohlenwasserstoffverbindungen? Haben sie negative Auswirkungen als Verunreinigungen im Plasma?

Einen Teil dieser Problemstellung bearbeitet das vom *Forschungszentrum Jülich* entwickelte, im Internet frei zugängliche Online-Tool *Hydkin* [**?**]. Mit diesem Werkzeug kann die Fragmentierung von Kohlenwasserstoff-Molekülen und deren Ionen in Wasserstoffplasmen untersucht werden. Dabei können unter anderem Ausgangsmoleküle, Energien und Temperaturen frei vom Benutzer gewählt werden.

Das im Rahmen dieser Bachelor-Arbeit erstellte Programm soll als Teil von *Hydkin* die wahrscheinlichsten Fragmentierungen mit Hilfe der bekannten Zerlegungsreaktionen be-

rechnen. Dabei soll, ausgehend von der bisherigen Implementierung, eine optimierte Version programmiert werden, die insbesondere die seitens der Nutzer an eine Internetanwendung gestellten Erwartungen erfüllen soll.

In der Arbeit wird in Kapitel 2 zunächst eine Beschreibung und Modellierung der Lösung des Problems vorgestellt. Die daran anschließenden Kapitel 3 und 4 beschreiben die Grundlagen und Motivation der Arbeit näher. Auf diesen aufbauend, wird in den Kapiteln 5 und 6 die Entwicklung der verschiedenen Optimierungen und Ansätze dargelegt. Kapitel 7 dient der Zusammenfassung und Beschreibung der noch vorhandenen Probleme und liefert einen kurzen Ausblick der im Anschluss an die Bachelorarbeit folgenden Weiterentwicklung.

Die Entwicklung der neuen Ansätze und Optimierung führt zu einer Lösung der Problemstellung, jedoch ist weitere Optimierung notwendig. Die Berechnungsergebnisse, die das vorher vorhandene Programm erzielt, sind in der neuen Implementierung reproduzierbar.

Die als Teil von *Hydkin* berechneten wahrscheinlichsten Fragmentierungspfade sollen unter anderem als Hilfe für andere Forschergruppen genutzt werden, um deren Programmen, die auch auf Supercomputern wie dem neuen *HPC FF* im *Forschungszentrum Jülich* mit z. T. Laufzeiten von Tagen und Wochen ausgeführt werden, eine Reduktion der chemischen Komplexität der Problemstellungen zu bieten. Erreicht wird dies, indem die Programme Fragmentierungen, die mit geringer Wahrscheinlichkeit auftreten, nicht mehr betrachten müssen und den Schwerpunkt auf die Verarbeitung der wahrscheinlichen Zerlegungen legen können.

2 Problemstellung

Um berechnen zu können, welche Fragmentierungen welche Wahrscheinlichkeiten unter welchen Bedingungen aufweisen, gilt es, eine Modellierung dieser Problemstellung zu finden, die ein Computer effektiv bearbeiten kann.

Die Abbildung der realen Fragmentierungspfade und ihrer Wahrscheinlichkeiten als Modell ist problembehaftet und unterliegt Regeln, die ebenfalls abgebildet werden müssen. Die momentan vorhandene Implementierung ist mit dieser Aufgabe aufgrund von Speicher- und Laufzeitproblemen teilweise überlastet.

Bei allen folgenden Ausführungen gilt, dass der Benutzer neben Temperaturen, Energien, etc. eine maximale Anzahl der wahrscheinlichsten Zerlegungspfade vor der Berechnung angeben kann. Die Aufgabenstellung definiert zusätzlich die Möglichkeit, zur maximalen Anzahl an Pfaden eine untere Wahrscheinlichkeitsschranke für einen Fragmentierungspfad anzugeben.

2.1 Grundlage für das Modell

Für dieses Modell sollen die folgenden Grundsätze gelten:

Ein benutzerdefiniertes Molekül wird durch chemische Reaktionen im Plasma bis zur atomaren oder molekularen Ebene fragmentiert. Moleküle, Atome und Ionen werden in der Literatur und innerhalb des Online-Tools *Hydkin* auch als *species* bezeichnet. Im weiteren soll dieser englische Begriff synonym zum deutschen Begriff *Spezies* genutzt werden.

Es gilt,

1. dass kein Molekül mit anderen Molekülen reagiert,
2. durch eine Reaktion keine größeren Moleküle entstehen können und
3. die Zerlegungs-Reaktionen der Moleküle bekannt sind.

Da es für jedes Molekül meist mehrere Zerlegungsmöglichkeiten gibt und auch die Fragmente wieder mehrere weitere Zerlegungen aufweisen können, gibt es dementsprechend viele unterschiedliche Möglichkeiten, ein Molekül zu zerlegen.

Durch die Einordnung des Ausgangsmoleküls als Grundzustand kann man alle weiteren Zerlegungen inklusive der Abbauprodukte als Zustandswechsel betrachten. Unterschiedliche Möglichkeiten der weiteren Zerlegung sind gleichbedeutend mit unterschiedlichen möglichen Zuständen, in die gewechselt werden kann.

Es gilt nach Regel 2, dass ein Zustand innerhalb eines Zerlegungs-Pfades nicht mehrfach erreicht werden kann, um endlose Schleifen von Zerlegungen auszuschließen.

Wahrscheinlichkeit eines Zustandswechels

In einem Zustand A sei das Molekül C_xH_y zerlegbar. Eine Zerlegung (ggf. eine der mehreren möglichen) bilde mit den Abbauprodukten den Zustand B1. Der der Zerlegung entsprechende Zustandswechsel von A nach B1 findet mit einer bestimmbaren Wahrscheinlichkeit statt. Die dem Zustandswechsel zu Grunde liegenden p Reaktionen weisen jeweils eine *Rate* r_{AB1_i} auf, sodass ein Zustandswechsel eine Gesamtrate

$$r_{AB1} = \sum_{i=1}^{p} r_{AB1_i} \qquad \left[\frac{1}{s}\right]$$

besitzt. Alle Raten r_{AB1_i} werden berechnet durch

$$r_{AB1_i} = \langle \sigma v \rangle_{AB1_i} \cdot n_{AB1_i},$$

wobei $\langle \sigma v \rangle_{AB1_i}$ als *Ratenkoeffizient* und n_{AB1_i} als *Dichte* bezeichnet wird. [?]

Die Wahrscheinlichkeit des Zustandswechsels von einem Zustand A in einen Zustand B1 für insgesamt s existente Zustandwechsel AB1, AB2, AB3, ..., ABs berechnet sich durch

$$W_{AB1} = \frac{r_{AB1}}{\sum_{j=1}^{s} r_{ABj}}.$$

Alle q Ratenkoeffizienten der q Reaktionen, die das Molekül C_xH_y zerlegen, werden von *Hydkin* durch ein *Ratengleichungssystem* (auch *chemisches Bilanzgleichungssystem*)

$$\vec{\dot{n}}_{C_xH_y} = \underbrace{\overleftrightarrow{A}}_{a_{ii}=\langle\sigma v\rangle_i} \cdot \underbrace{\vec{n}_{C_xH_y}}_{Dichten} + \underbrace{\vec{b}}_{influx}, \quad i = 1 \ldots n \tag{2.1}$$

bestimmt.

Für jede Dichte $n_{C_xH_y}$ der q Reaktionen gilt $n_{C_xH_y} \in \{n_e, n_p, n_H\}$. Auf diese Größen wird im nächsten Kapitel eingegangen (siehe S. 8).

Die Kenntnis der genauen physikalischen und chemischen Zusammenhänge sind zur Bearbeitung der Aufgabe nicht erforderlich.

Rahmenbedingungen

Das Modell und seine weitere Berechnung sind an die Rahmenbedingungen von Webservern und die Erwartungen von Internetnutzern anzupassen:

- Hardware
  - Es steht nur ein Prozessor zur Verfügung.
  - 1 Gigabyte RAM ist das Maximum an nutzbarem Arbeitsspeicher.
- Software
  - Basis des Online-Tools ist ein Apache-Webserver mit Perl-Interpreter.
  - Das Hauptprogramm ist in Perl implementiert, daher ist auch dieses Programm in Perl zu implementieren.
  - Es dürfen keine Anbindungen an Fortran, C, C++ oder Java geschaffen werden.
  - Es steht für alle Berechnungen nur ein Prozess und kein paralleles Rechnen zur Verfügung.
  - Die Gesamtrechenzeiten sind den Eingaben des Benutzers angemessen anzupassen.
    Richtmaß dafür ist die C_3H_8-Fragmentierung mit 10 angeforderten Pfaden, einer unteren Wahrscheinlichkeitsschranke von 1,0E-07 und einer maximalen Laufzeit von 120 Sekunden, besser 60 Sekunden.

2.2 Umsetzung des Modells im Computer

Um ein Modell im Computer darzustellen, wird neben den das Modell bearbeitenden Algorithmen eine Datenstruktur benötigt, die das Problem möglichst gut beschreibt und effizient bearbeitet werden kann.

Die vollständige Zerlegung eines Moleküls entspricht einer Liste von Zuständen, die nacheinander eintreten. Alle Möglichkeiten der Fragmentierung sind jedoch nicht in einer einzelnen Liste darstellbar. Eine Liste ist daher kein geeignetes Mittel, um die zur Zerlegung a_1 eventuell weiteren vorhandenen Zerlegungen a_2, ..., a_n in einem Zustand zu modellieren.

Diese Verzweigungen an jedem Listenelement sind durch Bäume darstellbar. Jeder Zustand wird durch einen Knoten symbolisiert. Der Anfangszustand ist dabei durch die Wurzel und die Endzustände der vollständigen Fragmentierung durch die Blätter des Baumes hinreichend dargestellt.

Dennoch ist diese Struktur zur Abbildung des Modells ungeeignet, da sie einen Aspekt der Fragmentierung nur unter Speicherung redundanter Information abbilden kann: gegebenenfalls kann ein Zustand von unterschiedlichen vorhergehenden Zuständen erreicht werden. Durch die Voraussetzung, dass ein Knoten eines Baumes nur das Kind eines Elternknotens sein kann, ist eine Speicherung eines mehrfach auf verschiedene Arten erreichbaren Zustandes nur durch eine redundante Speicherung (an jeder Stelle im Baum,

an der dieser auftritt) möglich. Da unbekannt ist, wieviele solcher redundanten Speicherungen benötigt werden und wie groß der Teilbaum ist, der an dieser Stelle folgt, ist diese Datenstruktur ungeeignet.

Ein gerichteter und gewichteter Graph, in der Eigenschaft als Obermenge von Bäumen, liefert hingegen eine adäquate Modellierung des Problems, da hier mehrere „Elternzustände" auf einen neuen Zustand verweisen können.

2.3 Einführung zu Graphen

Diese kurze Einführung basiert auf den ausführlichen Abhandlungen über Graphen in [?, ?].

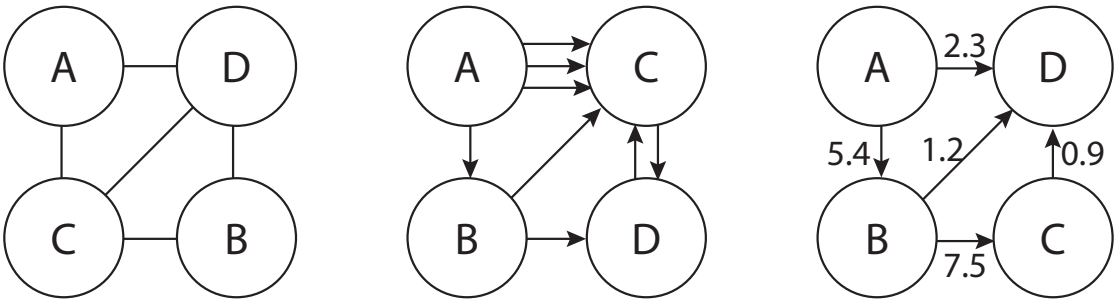

Abbildung 2.1: Beispiele für Graphen.
Links: ungerichteter Graph aus Knoten und Kanten
Mitte: gerichteter Graph
Rechts: gerichteter und gewichteter Graph

Die Abbildung 2.1 enthält Beispiele für die im Text erklärten Begriffe und Definitionen.

Anschaulich ist ein Graph ein Gebilde aus *Knoten* (*Ecken*) mit den verbindenden *Kanten*.

Ein aufeinander folgendes Besuchen von Knoten über verbindende Kanten, ausgehend von einem Startknoten A bis zu einem Endknoten B, nennt man einen *Weg* durch den Graphen von A nach B. Bezogen auf den linken Beispielgraph in Abbildung 2.1 wäre ein beispielhafter Weg von A nach B: $A \rightarrow C \rightarrow D \rightarrow B$.

Gerichtete Graphen haben die Eigenschaft, dass eine Kante einen Anfangsknoten A und einen Endknoten B hat und nur in Richtung von A nach B genutzt werden kann, wie in der Abbildung 2.1 im mittleren Beispiel zu sehen.

Zwischen den Knoten A und B können beliebig viele gerichtete Kanten existieren, sodass auch von B nach A Kanten verlaufen können. Im mittleren Beispielbild sind zwischen C und D diese wechselseitigen Kanten zu sehen.

Sind mehrere Kanten in einer Richtung von A nach B vorhanden, spricht man von *Mehrfachkanten*. Ein Beispiel dafür findet sich im mittleren Bild zwischen den Knoten A und C.

Erhält eine Kante zusätzlich zur bloßen Existenz ein sog. *Gewicht*, lassen sich *Kosten* für einen Weg angeben, die sich aus den Gewichten der Kanten des Weges ergeben. Die Berechnung dieser Kosten ist je nach Einsatzzweck unterschiedlich.

Ein gerichteter und gewichteter Graph, wie in der Modellierung eingesetzt, wird wie folgt definiert:

- Ein gerichteter Graph ist ein Tupel G = (V,E), wobei V die Menge aller Knoten und E die Menge aller Kanten ist.
- Die Menge E besteht aus Paaren e = (v,v'), welche die Verbindung eines Knotens v mit einem Knoten v' symbolisieren. Die Knoten v und v' sind *adjazent*. Für die Modellierung des zu bearbeitenden Problems gilt weiter: es existieren keine Kanten e = (v',v) und keine Zirkelbezüge, da in keiner Reaktion Moleküle größer werden.
- Ein Graph G heißt *gewichteter Graph* für $G_\gamma = (V, E, \gamma)$ mit $\gamma : E \to \mathbb{R}_0^+$.

Im Computer erfolgt die Speicherung dieser Informationen auf zwei verschiedene mögliche Arten, die im folgenden erklärt werden.

Adjazenzlisten

Adjazenzlisten sind dynamische Datenstrukturen, in welchen ein Knoten als Element einer Liste gespeichert wird. Die Kanten, die von diesem Knoten auf andere Knoten zeigen, werden in einer weiteren Liste gespeichert, die im Knoten-Element referenziert wird.

In einer Adjazenzliste müssen immer alle Kanten angegeben werden, die von einem Knoten auf einen anderen verweisen. Dies ist für gerichtete Graphen problemlos durch Eintragung zu lösen.

Eine Gewichtung der Kanten ist entweder in den Listenelementen oder in einer gesonderten Datenstruktur zu speichern.

Adjazenzmatrizen

Adjazenzmatrizen dagegen beruhen auf quadratischen Matrizen, in denen die Knoten in Zeilen und Spalten dargestellt werden. Ein Matrixelement a_{ij} entspricht der Definition einer Kante von Knoten I nach Knoten J.

- ungewichteter Graph: $a_{ij} \in \{0; 1\}$
- gewichteter Graph:
  - allgemein: $a_{ij} \in \mathbb{R}$
  - hier: $a_{ij} \in]0, 1]$

3 Grundlagen zur Berechnung

In diesem Kapitel wird zunächst beschrieben, auf welcher allgemeinen Grundlage die Berechnung fußt. Das folgende Kapitel beschäftigt sich damit, welche Ausgangssituation zu Beginn der Bachelor-Arbeit gegeben war. Daraufhin findet eine erste Analyse der Probleme dieser Situation statt, gefolgt von der Darstellung der Optimierung und ihrer Folgen.

Eine Modellierung der Zerlegungspfade als gerichteter und gewichteter Graph hat zur Folge, dass die Wahrscheinlichkeit einer vollständigen Fragmentierung durch die Berechnung der Wahrscheinlichkeit eines entsprechend den Regeln gültigen Weges durch den Graphen ausgedrückt werden kann.

Die in der Stochastik-Vorlesung vermittelten Eigenschaften von Wahrscheinlichkeitsbäumen können auf „Wahrscheinlichkeits-Graphen" übertragen werden:

1. Die Summe der einzelnen Wahrscheinlichkeiten der von einem Knoten weg weisenden Kanten muss 1 ergeben.
2. Das Produkt der einzelnen Wahrscheinlichkeiten der Kanten eines Weges entspricht der Gesamtwahrscheinlichkeit des Weges und damit der Wahrscheinlichkeit einer der möglichen Fragmentierungen eines Moleküls.
3. Die Summe der Wahrscheinlichkeiten aller Wege muss 1 ergeben.

Abbildung 3.1 zeigt einen solchen Graphen beispielhaft für das Ausgangsmolekül CH. Dies ist ein noch als Grafik darstellbarer Graph, der ähnlich zu den Fragmentierungsgraphen der größeren Kohlenwasserstoffe ist.

Alle Betrachtungen von Zerlegungen erfolgen, sofern nicht anders vermerkt, jeweils mit der Standardkonfiguration zur Berechnung durch *Hydkin*:

- Parameter, die pro Spezies gewählt werden können:
  - *initial condition*: beliebig, da nicht relevant für diese Analyse
  - *influx*: beliebig, da nicht relevant für diese Analyse
  - *losstime*: momentan beliebig
    - Die *losstime* wird in späteren Implementierungen eventuell dazu genutzt, eine weitere Reaktion mit der Rate $r_{loss} = \frac{1}{losstime} \cdot r$ einzubringen. Es entsteht dadurch für jede Spezies eine zeitliche Komponente für Analyse und Zerlegung. In den Zerlegungsgraphen entstehen neue Endknoten, die zu berücksichtigen sind.

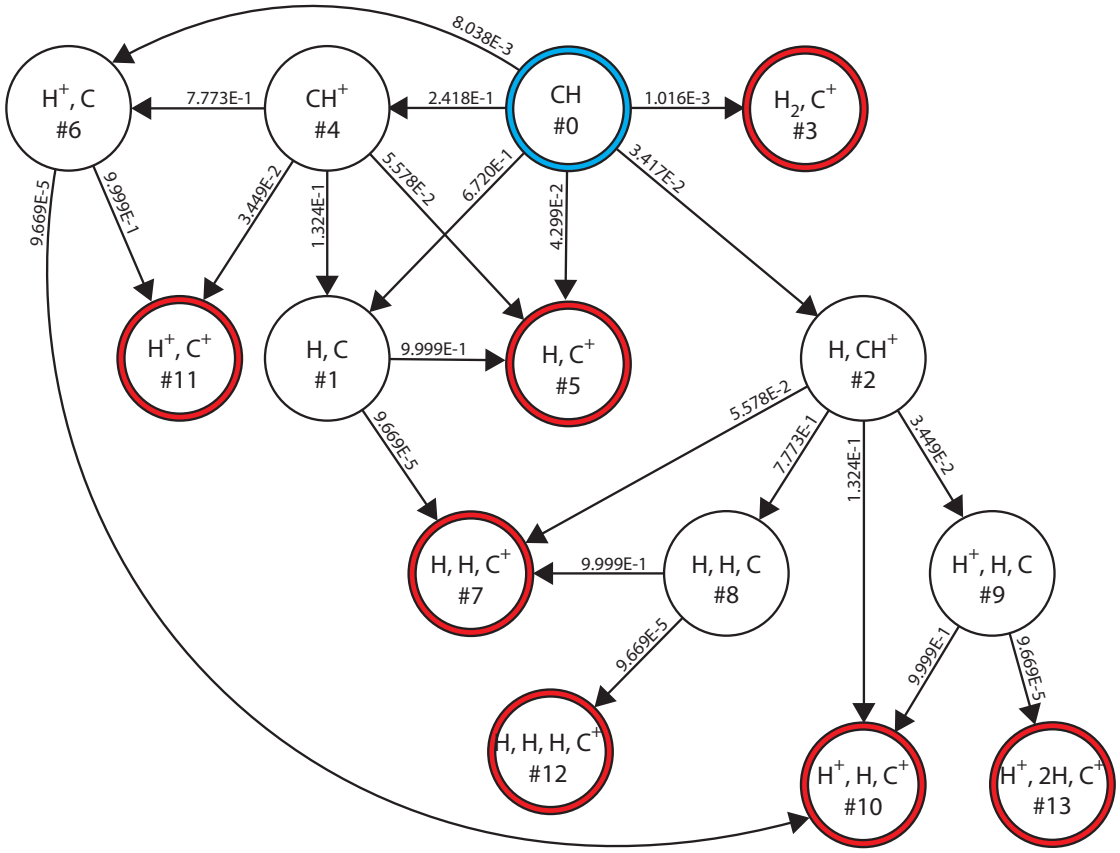

Abbildung 3.1: Vollständige Ionisierung von CH; zugehöriger Graph mit Zerlegungswahrscheinlichkeiten
Der blaue Kreis entspricht dem Ausgangsstoff und Beginn der Fragmentierung. Rote Kreise stellen Endknoten (∼zustände) dar.

- Parameter, die allgemein für alle Spezies angegeben werden:
  - Einfluss auf alle Ratenkoeffizienten $\langle \sigma v \rangle$:
    - E: 1 eV
    - $\mathrm{T}_e = \mathrm{T}_p = \mathrm{T}_H$: 10 eV
  - Einfluss auf alle Dichten n_X:
    - n_e: 1.0E13 $\frac{Anzahl\,Elektronen}{cm^3}$
    - n_p: 1.0E13 $\frac{Anzahl\,Protonen}{cm^3}$
    - n_H: 1.0E13 $\frac{Anzahl\,freier\,H-Atome}{cm^3}$
  - t_{max}: beliebig, da nicht relevant für diese Analyse

Zu den Parametern finden sich bei [**?**] nähere Erläuterungen zur Bedeutung. Der Algorithmus beachtet momentan keinen dieser Parameter. Die Abhängigkeit zwischen den Parametern und den Eingabedaten, auf denen der Algorithmus arbeitet, ist durch die Berechnung der Raten $r_X = \langle \sigma v \rangle_x \cdot n_X$ ein indirekter Einflussfaktor auf die Problemgröße (siehe unten).

In Tabelle 3.1 wird deutlich, welche Größe die Graphen der Fragmentierungen annehmen. Gleichzeitig liefern diese Zahlen eine Begründung dafür, warum bereits vor dieser Arbeit eine Optimierung der Speicherung der Kanten vorgenommen wurde (s. Kap. 4).

Ausgangsstoff	Knoten	Kanten	Endknoten	Erstellzeit in sec.
CH	14	22	7	1
CH_2	48	122	20	1
CH_3	119	359	43	1
CH_4	276	1.033	86	1
C_2H	104	252	34	1
C_2H_2	504	1.566	146	1
C_2H_3	1.360	5.575	319	1
C_2H_4	4.097	20.377	801	5
C_2H_5	9.506	53.157	1.691	14
C_2H_6	26.881	161.430	4.553	50
C_3H	825	2.863	168	1
C_3H_2	3.408	15.169	570	4
C_3H_3	10.375	57.119	1.429	15
C_3H_4	27.606	176.620	3.449	52
C_3H_5	62.565	444.855	7.750	140
C_3H_6	149.328	1.121.802	18.832	384
C_3H_7	412.296	3.169.721	49.796	1.223
C_3H_8	$>$ 761.471	$>$ 6.245.319	unb.	$\geq$3.600

Tabelle 3.1: Anzahl an Knoten und Kanten der Fragmentierungen von Kohlenwasserstoffen (ohne ionisierte), nur Standard-Konfiguration
Für C_3H_8 gibt es keine absoluten Zahlen, da Speicherplatzbedarf und Laufzeit zu groß für eine Analyse wurden.

Bei vielen Fragmentierungen wie z. B. CH_4 oder C_2H_6 ist folgender Sachverhalt für die Wahrscheinlichkeiten W_i der $n \geq 500$ Wege zu beobachten: unter 5% der berechneten Wege weisen eine Wahrscheinlichkeit über 1.0E-2 auf. Auch bei relativ kleinen Fragmentierungen, wie beispielsweise der von CH_4, kann eine Nicht-Standardkonfiguration diesen Sachverhalt bedingen.

Anschaulich bedeutet dies, dass eine sehr große Anzahl von Pfaden mit einer sehr kleinen

Wahrscheinlichkeit einen großen Teil der summierten Wahrscheinlichkeiten ausmacht:

$$\sum_{i=5\%\,von\,n}^{n} W_i \gg \sum_{i=1}^{5\%\,von\,n} W_i$$

Eine kleine Anzahl von Pfaden, die einen großen Teil der summierten Wahrscheinlichkeit abdeckt, ist wünschenswert in Bezug auf die Laufzeit des Problems, da diese direkt von der Anzahl der Pfade abhängt.

Ein Beispiel mit der Fragmentierung von CH_4 ist in Abbildung 3.2 zu sehen. Das linke Bild zeigt die Zerlegungen bei $T_e = T_p = 1$ eV, $n_e = n_p =$ 2.0E13, das rechte für $T_e = T_p = 45$ eV, $n_e = n_p =$ 2.0E12. Für diese Konfiguration hat die Dichte keinen Einfluss auf das Ergebnis, da sie bei der Berechnung der Wahrscheinlichkeit der Kanten gekürzt wird.

Es wird deutlich, dass links alle eingezeichneten Pfade ca. 80% der möglichen Fragmentierungen abdecken, rechts jedoch für eine ähnlich große Abdeckung ∼500 Pfade notwendig sind. [**?**]

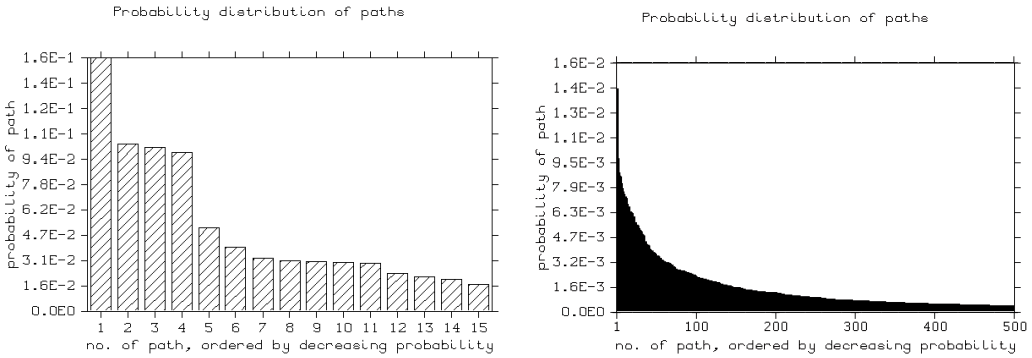

Abbildung 3.2: Wahrscheinlichkeiten der Zerlegungen von CH_4., [**?**]
Links: bei $T_e = T_p = 1$ eV, $n_e = n_p =$ 2.0E13 decken 15 Wahrscheinlichkeiten 80% aller Wege ab (summierte Wahrscheinlichkeit)
Rechts: bei $T_e = T_p = 45$ eV, $n_e = n_p =$ 2.0E12 decken ∼500 Wahrscheinlichkeiten 80% aller Wege ab

Mit Bezug auf die Zielsetzung, Aussagen über die Vernachlässigbarkeit von Pfaden treffen zu können und eine Reduktion der chemischen Komplexität von anderen Problemstellungen zu erhalten, bedeutet dieser Sachverhalt, dass für viele gleich gering wahrscheinliche Pfade keiner vernachlässigt werden kann und keine Reduktion möglich ist. Programme und Codes, die sich auf diese Aussage stützen, müssen also dennoch alle Pfade betrachten. Decken wenige Pfade mehr als 50 - 80% aller Pfade ab, sind alle weiteren, kleineren vernachlässigbar: die Problemstellungen anderer Projekte können in ihrer Komplexität reduziert werden.

4 Ausgangssituation

Vor Beginn dieser Arbeit ist bereits eine erste Implementierung zur Lösung des Problems realisiert worden. Diese sah vor, aus den bekannten Reaktionen und dem Ausgangsstoff eine spezielle Adjazenz-Struktur aufzustellen, die mit Hilfe von Breiten- und Tiefensuche rekursiv bearbeitet wurde. Es wurden dabei keine unteren Wahrscheinlichkeits-Schranken verwendet.

Die Implementierung wurde in der Interpreter-Skript-Sprache Perl vorgenommen, da der größte Teil von *Hydkin* in Perl implementiert ist (siehe Rahmenbedingungen) und diese Sprache für den Einsatz des Programms als online verfügbares Tool gut geeignet ist.

Die spezielle Adjazenz-Struktur entstand bereits vor dieser Arbeit vor folgendem Hintergrund: Perl arbeitet intern für Berechnungen mit doppelt genauen Fließkommazahlen[1] [**?**], die, wie in der IT-Grundlagen-Vorlesung vermittelt, in der Darstellung nach IEEE 754 einen Speicherbedarf von 64 Bit bzw. 8 Byte haben. Bei 1 Gigabyte (GB) Arbeitsspeicher, wie in den Rahmenbedingungen gefordert, ist es theoretisch möglich, ca. 134.217.728 doppelt genaue Fließkommazahlen zu speichern (ohne Abzüge an Speicher für Betriebssystem, Overhead des Programms und des Interpreters, Daten die das Programm erzeugt, etc.).

Eine Adjazenzmatrix für maximal $\sqrt{134.217.728} \approx 11.585$ Knoten würde dieses Speicherlimit bereits brechen, da für alle 11.585 Knoten auch 11.585 Zahlen für die Kanten zu speichern sind, die von diesem Knoten abgehen. Da auch eine nicht existierende Kante als Zahl 0 abgespeichert werden muss, ist der Speicheraufwand enorm.

Für die meisten Fragmentierungen wird diese kritische Grenze überschritten, wie der Tabelle 3.1 entnommen werden kann.

Vorangegangene empirische Tests der Software zeigten auf, dass die meisten Knoten nur wenige Kanten aufweisen. Das bedeutet, dass die überwiegende Zahl der Matrixeinträge $a_{ij} = 0$ sind. Die Matrix ist dadurch wahrscheinlich nur *dünn besetzt*[2] und es liegt nahe eine Ersatzmatrix aufzustellen, die die Positionen der Nicht-Null-Elemente speichert und wesentlich kleiner dimensioniert ist.

Die einzusetzende Programmiersprache Perl bietet jedoch mit Hashes als vorimplementierte, einfach nutzbare Datenstruktur ein einfacheres Konzept zur Realisierung. Dabei ist ein linearer Zugriff gewährleistet.

[1] Doppelt genaue Fließkommazahlen müssen in Perl nicht nach IEEE 754 formatiert sein. Es entspricht immer dem Fließkommazahlen-Format, das auf dem ausführenden Computer genutzt wird. (vgl. [**?**], http://perldoc.perl.org/functions/pack.html)

[2] (Lineare Algebra und Numerik-Vorlesung: *dünn besetzt* bedeutet min. 80% der Matrixelemente sind gleich 0)

Hashes sind, wie in der Vorlesung *Algorithmen und Datenstrukturen* vermittelt, durch eine Hashfunktion in der Lage, mit linearem Aufwand Elemente effizient zu speichern und wieder zu finden. In Perl wird diese Eigenschaft genutzt, um ein zu speicherndes Datum durch ein gehashtes Index-Datum zu referenzieren. Wird das gespeicherte Datum erneut benötigt, ist es mit linearem Zeitaufwand durch den gehashten Index auffindbar.

Es werden daher in der Software nicht die Knoten, sondern die Kanten in einem Hash gespeichert. Dabei bildet die Verknüpfung von „$Knoten\,A \sim Knoten\,B$" (= Kante) das Indexelement und das Kantengewicht das zu speichernde Datum.

Diese Art der Speicherung ist am ehesten mit Adjazenzlisten zu vergleichen. Sie ist jedoch wesentlich effizienter als diese, da viele Operationen auf Listen im Allgemeinen sehr aufwändig sind und das Indexelement ähnlich zu der Adressierung von Adjazenzmatrizen aufgebaut ist. [**?**, **?**]

Die Suche nach den Wegen erfolgt rekursiv, wobei alle Wege gespeichert werden. Das rekursive Durchlaufen des Graphen anhand der existierenden Kanten bis zu einem Endknoten vereinfacht die Implementierung des Such-Algorithmus. Nach einer Sortierung nach Wahrscheinlichkeit wird die gewünschte Anzahl Pfade ausgegeben. Abbildung 4.1 zeigt die ursprüngliche Bearbeitung anschaulich als Ablaufdiagramm.

Der im Abblaufdiagramm dritte Schritt „Endknoten und Wahrscheinlichkeiten in der Adjazenzstruktur bestimmen" zwischen Aufbau der Adjazenz-Struktur und Suche erklärt sich aus der Struktur der Datenbasis für die Fragmentierungen, die während des Aufbaus genutzt werden. Wie auf Seite 4 gezeigt, sind die Wahrscheinlichkeiten erst durch die Gesamtsumme der Raten berechenbar. Diese ist erst nach dem vollständigen Aufbau der Adjazenzstruktur bzw. der vollständigen Eintragung aller existierenden Kanten bekannt.

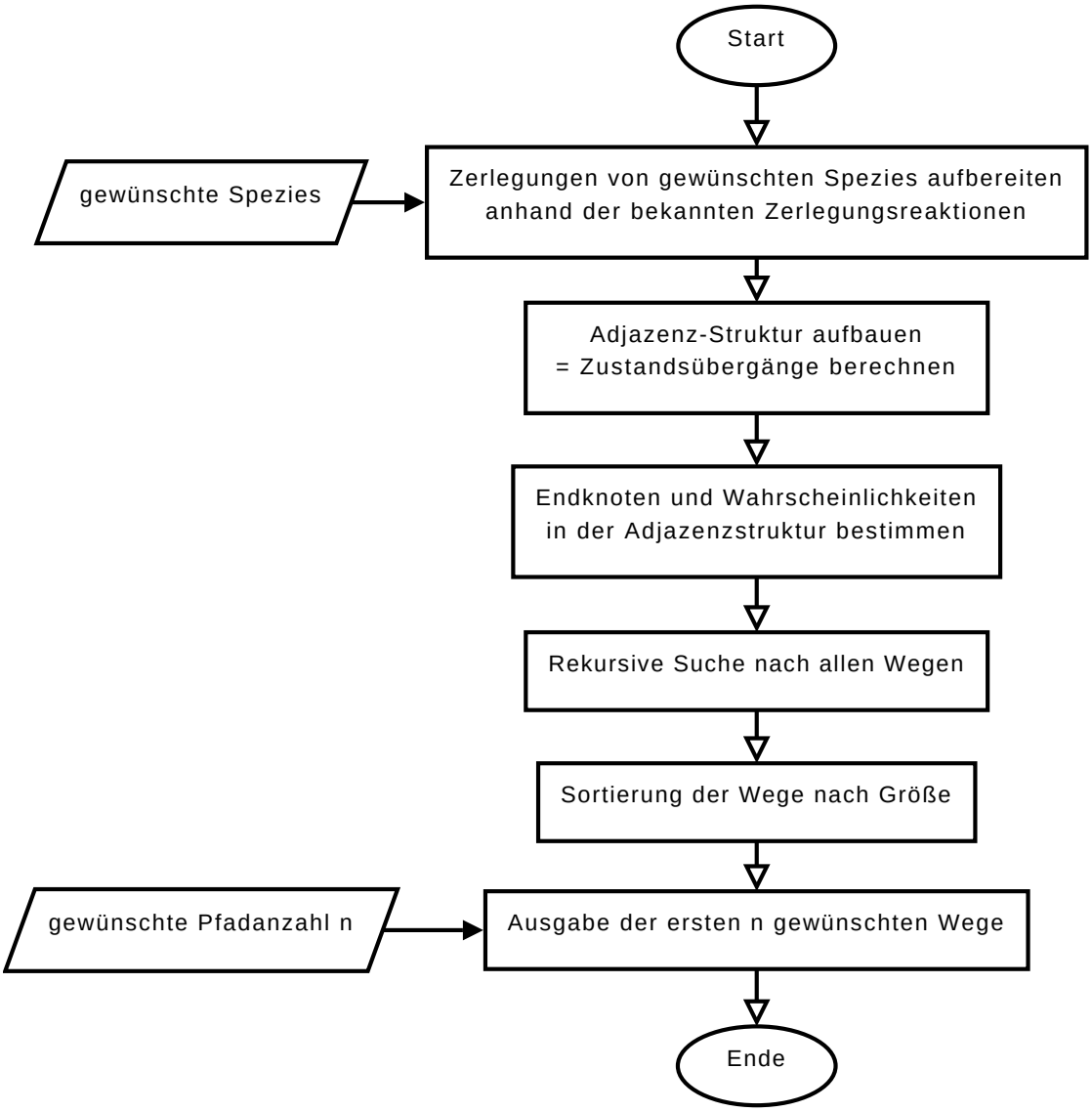

Abbildung 4.1: Struktogramm des ursprünglichen Programmablaufes

5 Optimierungen aufgrund der Ausgangssituation

Die folgenden Schwachpunkte des vorhandenen Algorithmus lassen sich durch Einfügen von Zeitmessungen und Codeanalysen für die einzelnen Programmabschnitte erkennen:

1. hohe Knotenanzahlen bei vielen Fragmentierungen (siehe Tabelle 3.1 auf Seite 10)
2. Teilabschnitte des Programms mit einem Aufwand von $O(n^2)$
3. Speicherung aller existenten Pfade, keine Ausnutzung der durch den Benutzer maximal vorgegebenen Anzahl
4. Rekursion aufwändig, Ursache zunächst unbekannt
5. Aufstellung der Adjazenzstruktur nimmt z. T. sehr viel Zeit in Anspruch (siehe Tabelle 3.1 auf Seite 10)
6. Speicherprobleme für C_3H_6, C_3H_7 und C_3H_8 vorhanden, da die Anzahl der Kanten den verfügbaren Speicherplatz übersteigt

Die Punkte 1 und 2 sind stark miteinander verknüpft, da die Laufzeit für eine große Anzahl von Knoten durch einen Aufwand $O(n^2)$ quadratisch anwächst.

Die Aufstellung der Adjazenzstruktur hängt nur linear von der Anzahl der Knoten ab, da die Anzahl der Kanten an einem Knoten wesentlich kleiner als die Anzahl der Knoten ist. Daher muss Punkt 2 getrennt von Punkt 5 betrachtet werden, wird aber auch durch Punkt 1 beeinflusst.

Die Speicherung aller Pfade (siehe Punkt 3) ist nicht effizient, wird jedoch beibehalten. Der Grund dafür liegt in der Überlegung, zunächst nur in kleinen Schritten zu optimieren, um die Gefahr von falschen Ergebnissen durch fehlerhafte Optimierungsschritte zu minimieren.

Aus Punkt 5 und 6 ergibt sich, dass die Adjazenzstruktur ggf. in der vorhandenen Implementierung ungeeignet ist, da sie sowohl zu groß als auch zu zeitintensiv bei der Aufstellung wird. Das Problem der Optimierung dieses Code-Abschnittes in den Punkten 5 und 6 (Aufstellung und Speicherprobleme) wird dennoch zunächst zurückgestellt, da dies erst für größere Ausgangsmoleküle (wie C_3H_5) problematisch ist und die anderen angesprochenen Probleme der Implementierung davon weitestgehend unabhängig sind.

Es wird zunächst die Implementierung einer unteren Wahrscheinlichkeitsschranke (wie in der Aufgabenstellung gefordert) und eine Optimierung des Punktes 2 (Aufwand $O(n^2)$) in den Vordergrund gestellt.

5.1 Optimierung durch Einführung einer unteren Wahrscheinlichkeitsschranke

Die vom Benutzer vorzugebende untere Wahrscheinlichkeitsschranke hat zum Ziel, Pfade, deren Gesamtwahrscheinlichkeit kleiner wird als diese, zu verwerfen und nicht weiter zu betrachten. Dies hat ein Ende der Rekursion zur Suche nach dem nun verworfenen Weg zur Folge. Die Einführung der Schranke erbringt signifikante Fortschritte in der Berechenbarkeit, speziell auch für größere Modelle.

Um diese Schranke optimal zu nutzen, sind folgende Überlegungen wichtig:

$$\forall x, y \in]0;1] \ \wedge \ x \leq y : \quad x * y \leq x \tag{5.1}$$

$$\forall x, y \in]0;1[\wedge x < y : \quad x * y < x \tag{5.2}$$

Beweis: (nur für (5.2), gilt für (5.1) analog)

$$x, y \in]0;1[\wedge x < y \tag{5.3}$$

Da $x > x * y$ für $x \to y$ ein Maximum aufweist, jedoch für $y \to x$ ein Minimum, ist dies abschätzbar durch:

$$\begin{aligned} & x > x * y > x * x = x^2 \\ \Leftrightarrow \quad & x > x^2 \\ \Leftrightarrow \quad & 1 > x \end{aligned}$$

Überträgt man diesen Sachverhalt auf den Fall, dass x das Produkt der Einzelwahrscheinlichkeiten des bisherigen Pfades und y die Einzelwahrscheinlichkeit der zu untersuchenden Kante ist, ergibt sich folgendes Kriterium:

Bei der Multiplikation der Einzelwahrscheinlichkeiten eines Pfades im Graphen wird die Gesamtwahrscheinlichkeit immer nur maximal so groß wie die kleinste Einzelwahrscheinlichkeit. Dies ergibt eine Bedingung für die Untersuchung einer Kante:

- ist das Kantengewicht bereits kleiner als die untere Schranke, muss die Kante nicht weiter betrachtet werden.

Wichtiger ist jedoch die Betrachtung der Gesamtwahrscheinlichkeit eines Pfades während der Suche. Ein Pfad kann Einzelwahrscheinlichkeiten aufweisen, die alle größer als die untere Schranke sind und dennoch aufgrund einer Unterschreitung der Schranke mit der Gesamtwahrscheinlichkeit verworfen werden.

- Die Gesamtwahrscheinlichkeit fällt aufgrund von Gleichung (5.1) monoton mit jeder hinzugefügten Kante. (Für alle Wahrscheinlichkeiten < 1 sogar streng monoton, siehe Gleichung (5.2))
- Unterschreitet die Wahrscheinlichkeit eines Weges im Graphen durch das Hinzufügen einer Kante die untere Schranke, ist diese Kante nicht weiter zu betrachten.

Die Exponenten der beiden Faktoren liefern ein Kriterium, wie schnell ein Pfad die kritische Grenze unterschreitet:

$$\begin{aligned} x &= a * 10^{-b} \\ y &= c * 10^{-d} \\ \Rightarrow x * y &= a * c * 10^{-(b+d)} \end{aligned}$$

- Je größer (b+d), desto kleiner wird die Wahrscheinlichkeit. Wird nun eine im Exponenten kleine Zahl y mit einer noch oberhalb der Schrank liegenden Zahl x multipliziert, wird die Wahrscheinlichkeit unter Umständen sprunghaft kleiner als die untere Schranke.

Beispiel mit unterer Schranke $1.0 * 10^{-7}$:

$$\begin{aligned} x &= 1.0 * 10^{-2} \\ y &= 1.0 * 10^{-6} \\ x * y &= 1.0 * 10^{-8} \\ &< 1.0 * 10^{-7} \end{aligned}$$

Die Geschwindigkeit, mit der eine Gesamtwahrscheinlichkeit unter die kritische Grenze fällt, ist ebenfalls ein wichtiger Aspekt im Hinblick auf den Algorithmus. Fällt die Gesamtwahrscheinlichkeit eines Pfades nach nur wenigen Rekursionsstufen unter die kritische Grenze, ist dies in Bezug auf Laufzeitverhalten und Speichernutzung besser als eine langsam, in vielen Rekursionsstufen unter die Schranke abfallende Gesamtwahrscheinlichkeit (entspricht einer schlechten Konditionierung des Problems), da viele neue Stufen und Betrachtungen ausgelassen werden können.

Es ist jedoch zu beachten, dass große Wahrscheinlichkeiten auch nur kleine Veränderungen im Exponenten bewirken, wenn sie multipiliziert werden. Eine Unterscheidung zwischen langen, aber gültigen Pfaden und nur schlecht konditionierten Teilproblemen, die unter die kritische Schranke fallen, ist daher ohne weitere Anhaltspunkte nur durch konkrete Berechnung möglich. Es ist daher auch für gültige Pfade mit Speicher- und Laufzeitproblemen zu rechnen.

Eine Berechnung für die meisten Spezies aus der Gruppe C_2H_X wird erst durch die Einführung dieser Schranke möglich - zuvor sprengte die Anzahl an Wegen und die lange Laufzeit der Berechnung den vorgegebenen Rahmen oder die Berechnung wurde durch *Timeouts*[1] vorzeitig abgebrochen.

[1] Timeout bedeutet, dass der Webserver das Perl-Skript aufgrund zu langer Laufzeit abgebrochen hat.

	CH_4			C_2H_6		
untere Schranke:	1.0E-3	1.0E-4	1.0E-7	1.0E-3	1.0E-4	1.0E-7
Laufzeit in Sekunden:	<1	<1	4	33	298	>600, Timeout
maximale Pfadzahl:	181	860	6.383	97	1.390	unbekannt

Tabelle 5.1: Benchmark für die Einführung der unteren Wahrscheinlichkeitsschranke. Die in Abschnitt 5.2 beschriebene Optimierung wurde zur Datenerhebung genutzt. Die Zeitangabe bezieht sich nur auf die Zeit zur Suche, ohne Aufstellung der Adjazenzmatrix.

Insbesondere zu erwähnen ist hier nochmals der auf Seite 10 beschriebene Sachverhalt: durch das Weglassen dieser großen Anzahl an kleinen Wegen ist eine Berechnung möglich innerhalb der Zeiten, die in den Rahmenbedingungen gefordert sind.

Dies ist andererseits auch als Schwachpunkt des optimierten Algorithmus zu sehen: ist die Schranke zu klein, sind zu viele Wege zu berechnen. Ist sie zu groß, verschwinden möglicherweise interessante Wege. An dieser Stelle ist Optimierung zur geschickten Findung einer solchen Schranke ausgehend von einer Startgröße notwendig.

5.2 Optimierung der Wahrscheinlichkeits-Berechnung

Die Problematik zur Berechnung der Wahrscheinlichkeiten der Kanten an jedem Knoten (dritter Schritt im Abblaufdiagramm auf Seite 14) gehört in die Klasse der Probleme mit Aufwand $O(n^2)$. Dieser Aufwand wird durch die nötigen Betrachtungen aller theoretisch möglichen Kanten an jedem Knoten generiert, auch wenn sie nicht existieren.

Bereits kleinere Problemstellungen wie C_2H_3 mit 1.360 Knoten und $1.360^2 = 1.849.600$ Schleifendurchläufen in der benötigten doppelt-geschachtelten Zählschleife können durch die notwendigen Überprüfungen auf Existenz der Kante und ggf. die Berechnung der Wahrscheinlichkeit eine erhebliche Rechenleistung fordern und den anfangs beschriebenen Rahmen der Anforderungen sprengen.

Zur Lösung dieses Problems wird die Berechnung der Wahrscheinlichkeit in die Rekursion zur Wegesuche eingebettet. Bevor eine Kante näher auf ihre Eignung für einen Pfad untersucht wird, wird ihre Wahrscheinlichkeit berechnet und in der Adjazenzstruktur gespeichert.

Tabelle 5.2 zeigt die Güte dieser Optimierung. Es ist jedoch zu beachten, dass eine Schranke von 1.0E-3 verwendet wurde, da für kleinere Schranken aufgrund der höheren Anzahl an Pfaden keine Ergebnisse vor einem Abbruch durch Timeout zu erhalten waren.

	Laufzeit in sec.	
Modus	C_2H_4	C_2H_6
separat + Suche	21 + 5	> 600 + unb.
eingebettet + Suche	5	33

Tabelle 5.2: Benchmark für Optimierung „Einbettung der Wahrscheinlichkeitsberechnung".
Die Zeitangabe bezieht sich nur auf die Berechnung der Wahrscheinlichkeiten und die Suche, ohne Zeitangabe für die Aufstellung der Adjazenzstruktur. Als untere Wahrscheinlichkeitsschranke wurde 1.0E-3 gesetzt.

5.3 Versuch einer Zwischenspeicherung

Wie in Abschnitt 2.2 beschrieben, wurde als Datenstruktur für das Modell ein gerichteter und gewichteter Graph gewählt. Diese Entscheidung wurde damit begründet, dass mehrere Zustände auf den gleichen Zustand verweisen können.

Es kann durch diesen Sachverhalt für das Laufzeitverhalten förderlich sein, statt bei jedem Besuchen dieses Knotens erneut die Übergänge zu berechnen, die bereits analysierten Übergänge und Teilwege zwischenzuspeichern. Dies würde die Analyse in späteren Rekursionsstufen für weitere, auf diesen Zustand verweisende Zustände und eventuell daraus resultierende Wege einsparen.

Alle m zwischengespeicherten Teilwege ab einem Zustand A zu den Endknoten haben eine Wahrscheinlichkeit x. Sinkt die Wahrscheinlichkeit bei der Findung dieser Teilwege unter die Wahrscheinlichkeitsschranke, wird der Teilweg verworfen.

Wird bei einer Wegsuche erneut in einer Rekursionsstufe der Zustand A erreicht, können m vollständige Wege ohne weitere Analysen berechnet werden:

$$\begin{aligned} Weg\, i &= (Teilweg\, bis\, A) + (i - ter\, Teilweg\, hinter\, A) \\ W_i &= W_A * W_{i_{Teil}} \end{aligned}$$

Ein Anfügen der Teilwege an den Weg bis zum Zustand A ergibt m vollständige Wege. Die Wahrscheinlichkeit des i. Weges ist dann das Produkt aus der Wahrscheinlichkeit des i. Teilweges und der Wahrscheinlichkeit des Weges bis zum Zustand A. Liegt dieses Produkt der Wahrscheinlichkeit oberhalb der Wahrscheinlichkeitsschranke, ist der i. Weg gültig und kann gespeichert werden.

Wenn Teilwege ab einem Knoten (Zustand) A gespeichert werden sollen, ist zu fragen, ab welchem Zustand ein Teilweg als zwischenzuspeichern zu betrachten ist.

Unzweckmäßig ist die Speicherung aller Analysen aller Wege, da dies keine Verkleinerung des Problems bewirkt und auch zu enormen Speicherplatzproblemen führen würde.

Ebenfalls ohne Reduzierung des Aufwandes wäre eine Betrachtung des Problems von den Endknoten in rückwärtige Richtung zum Anfangszustand. Der Aufwand, von dort aus

gültige Wege zu suchen, liegt in der gleichen Größenordnung wie die vorwärts gerichtete Suche für Knoten, da mindestens die gleiche Anzahl an Kanten zu betrachten ist. Ebenso ist zu fragen, bis zu welchem Punkt rückwärts gesucht werden muss, um eine Vereinfachung der Vorwärtssuche durch die zwischengespeicherten Teilwege zu gewährleisten.

Eventuell wird der Aufwand durch die Rückwärtssuche sogar größer. Es können rückwärtig Teilwege untersucht werden, deren Wahrscheinlichkeit vom Ende der Wege ausgehend analysiert nicht unter die Wahrscheinlichkeitsschranke fällt, von vorne jedoch aufgrund der Schranke niemals betrachtet werden würden. Die Folge ist, dass nicht die gleiche Anzahl, sondern mehr Kanten betrachtet werden, da die Rückwärtssuche nicht alleinige Betrachtung sein kann (andernfalls wäre sie nur eine umgedrehte Vorwärtssuche), sondern zusätzlich eine Vorwärtssuche durchgeführt werden muss.

Aus der Anzahl an Kanten, die auf einen Zustand verweisen, kann kein Kriterium abgeleitet werden, da unbekannt ist, wieviele dieser Kanten auch tatsächlich als Teilwege genutzt werden.

Eine Reduzierung der Speicherung auf Teilwege, die nicht bis zu Endzuständen laufen, jedoch mehrfach Teile von Wegen sind, ist nicht möglich. Aufgrund der Unvorhersagbarkeit vor der Rekursion und durch die Tatsache, das während der Rekursion zur Beurteilung einer Speicherung der Weg schon gespeichert werden muss, ist dies nicht durchführbar.

Die Optimierungs-Idee wurde daher als momentan nicht realisierbar fallen gelassen.

5.4 Versuch eines Abweich-Kriteriums

Die Gesamtwahrscheinlichkeit eines Weges sinkt rapide, wenn der Exponent sprunghaft mit Hinzufügen einer Kante kleiner wird (siehe Abschnitt 5.1). Trägt man alle Einzelwahrscheinlichkeiten der Reihenfolge nach über der X-Achse eines Koordinatensystems auf, erhält man Funktionsgraphen wie in Abbildung 5.1. Dabei sind in der linken Abbildung starke Schwankungen in den Werten der Einzelwahrscheinlichkeiten zu sehen, bei der rechten jedoch ein relativ glatter Verlauf.

Diese Schwankungen ließen sich nutzen, indem eine neue Betrachtung der Wahrscheinlichkeiten eingeführt wird: die Abweichung der Werte von einem Mittelwert der Einzelwahrscheinlichkeiten um mehr als eine zugelassene Toleranzgrenze. Diese Schwankungen korrelieren mit den Sprüngen, die den Exponenten der Wahrscheinlichkeit stark verkleinern und damit die kritische Grenze unterschreiten lassen. (siehe Abschnitt 5.1)

Zur Bestimmung eines möglichst gut passenden Toleranzbereiches müssten neben einer zu definierenden groben Startabweichung die Grenzen bereits gefundener Wege in die Bestimmung eingehen, um der Komplexität und Konditionierung des Problems Rechnung zu tragen. Lösbar wäre dies beispielsweise durch die Bildung eines Mittelwertes über alle Toleranzgrenzen aller Wege.

Bei der Untersuchung einer möglichen neuen Kante eines Weges ist daher neben der Wahrscheinlichkeitsschranke entscheidend, ob die neue Einzelwahrscheinlichkeit einen To-

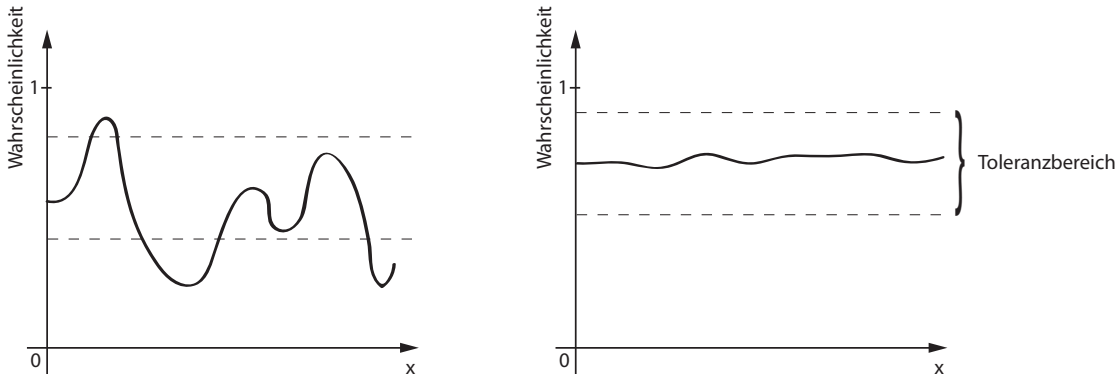

Abbildung 5.1: Auftragung der Einzelwahrscheinlichkeiten über der X-Achse
Links: starke Schwankungen über den Toleranzbereich hinaus in einem unwahrscheinlichen Zerlegungspfad
Rechts: alle Werte innerhalb des Toleranzbereichs in einem gültigen Zerlegungspfad

leranzbereich um den Mittelwert aller bisherigen Einzelwahrscheinlichkeiten der Kanten des Weges über- bzw. unterschreitet.

Gegen dieses Kriterium spricht jedoch die Auflage, dass Wege tatsächlich die Wahrscheinlichkeitsschranke unterschreiten und damit *unmöglich* sein müssen und nicht nur *unwahrscheinlich* aufgrund des Verlassens des Toleranzbereichs. Der Ansatz ist daher zu verwerfen.

5.5 Optimierung durch Beschränkung auf n Pfade

Die bereits rudimentär implementierte Beschränkung auf die n vom Benutzer gewünschten wahrscheinlichsten Pfade soll ersetzt werden. Die vorherige Version speicherte alle Pfade und gab n aus. Die neue Implementierung speichert alleinig die geforderten n Pfade und setzt die daraus resultierenden Möglichkeiten zur Vereinfachung des Problems ein.

Die Suche der Wege ist weiterhin rekursiv implementiert. Die Speicherung der Pfade erfolgt in einem Feld. Statt wie bislang alle ohne Bewertung einzufügen, werden nun n Pfade anhand ihrer Wahrscheinlichkeit absteigend sortiert in das Feld eingefügt und gespeichert.

Dies hat gleichzeitig den Vorteil, dass, sobald die n Pfade gefunden sind, nur noch Pfade betrachtet und eingefügt werden müssen, die eine größere Wahrscheinlichkeit als der Pfad mit der kleinsten Wahrscheinlichkeit aller gespeicherten Pfade haben. Nach jedem Einfügen ist der $(n+1)$. überflüßige Pfad im Feld zu löschen und die untere Schranke für die Suche nach Pfaden auf den Wert des an letzter Stelle stehenden Weges anzuheben.

		Laufzeit der Rekursion in Sekunden				
		CH_4			C_2H_6	
	untere Schranke:	1.0E-3	1.0E-4	1.0E-7	1.0E-3	1.0E-4
	max. Pfadzahl:	184	872	6.383	97	1.390
Modus	Pfadanzahl					
ohne Beschränkung (vgl. Tabelle 5.2)	(beliebig viele)	<1	<1	4	33	298
mit Beschränkung	10	<1	<1	<1	27	49
	50	<1	<1	<1	30	178
	100	<1	<1	2	31	137
	250	<1	1	2	-	195
	500	-	2	5	-	253
	1.000	-	2	11	-	311
	2.500	-	-	38	-	306
	5.000	-	-	61	-	-

Tabelle 5.3: Benchmark für Optimierung „Beschränkung auf n Pfade"
Für C_2H_6 und eine untere Schranke <1.0E-4 können aufgrund von Timeouts keine Daten erhoben werden.

Es ist gegenüber dem letzten Benchmark zur Einbettung der Wahrscheinlichkeitsbildung (siehe Tabelle 5.2 und Vergleiche in Abbildung 5.2) ein deutlicher Unterschied und Fortschritt zu erkennen. Dennoch ist das Ergebnis nicht zufrieden stellend, da die Wahrscheinlichkeitsschranken immer noch relativ hoch sind gegenüber der geforderten unteren Schranke 1.0E-7. Tests mit Schranken < 1.0E-4 führten z. T. zu Timeouts und sind daher hier nicht aufgeführt.

Die auffällige Laufzeit für CH_4 mit einer unteren Wahrscheinlichkeitsschranke 1.0E-7 wurde zunächst nicht weiter beachtet, da für größere Moleküle (wie C_2H_6) diese Wahrscheinlichkeitsschranke nicht ohne Laufzeit- oder Speicherprobleme erreicht werden konnte (siehe dazu: Abschnitt 6.2 auf Seite 32).

5.6 Optimierung der Rekursion

Die Suche der Wege benötigt teilweise eine Laufzeit von über 3 Minuten (siehe Tabelle 5.3). Insbesondere für Fragmentierungen größerer Moleküle wie z. B. C_2H_6 und einer unteren Wahrscheinlichkeitsschranke von 1.0E-4, die den in den Rahmenbedingungen geforderten 1.0E-7 in keiner Weise entspricht, sind hohe Laufzeiten zu verzeichnen. Wie bereits unter Punkt 4 am Anfang des Kapitels angesprochen, war die Ursache für diese Zeitdauer zunächst unbekannt.

Es bestand zunächst die Idee, eine Priorisierung der Betrachtung der Kanten einzuführen. Dabei sollen die Kanten ihrer Wahrscheinlichkeit nach bearbeitet werden, beginnend mit der größten Wahrscheinlichkeit.

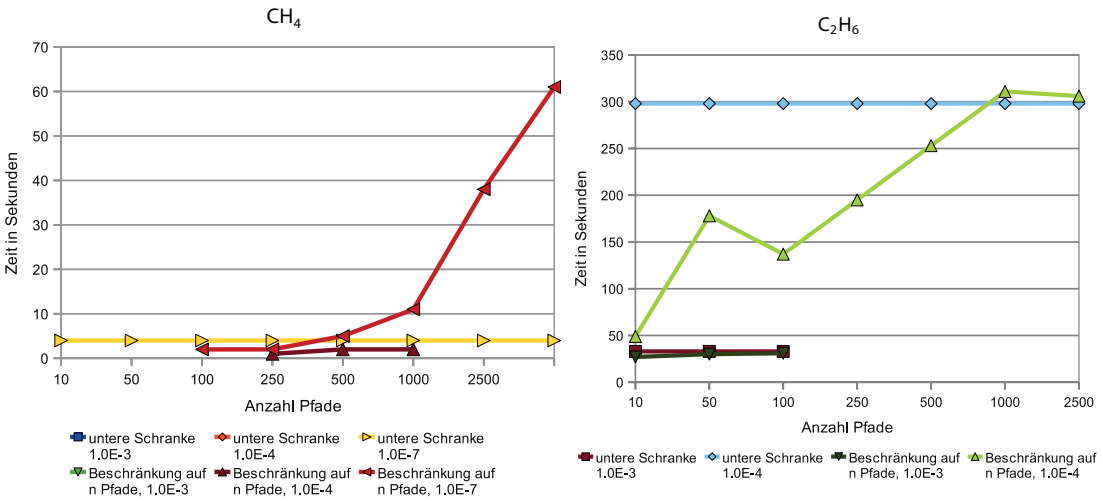

Abbildung 5.2: Vergleich der Optimierungen „untere Wahrscheinlichkeitsschranke" und „Beschränkung auf n Pfade".
Zeiten < 1 Sekunde sind nicht verzeichnet. Deutlich zu erkennen ist, dass für CH_4 die Beschränkung auf n Pfade suboptimal ist, für C_2H_6 hingegen sehr förderlich.
Links: Vergleich für Fragmentierungen von CH_4
Rechts: Vergleich für Fragmentierungen von C_2H_6

Um abzuschätzen, welchen Aufwand das notwendige absteigende Sortieren nach sich zieht, wurde repräsentativ für den Fall C_2H_6 eine Analyse durchgeführt. Es wurde dabei ermittelt, wieviele Kanten tatsächlich maximal an den Knoten vorhanden sind. Für C_2H_6 sind für alle 26.881 Knoten maximal 30 Kanten zu finden, die von einem Knoten abgehen. Der zu erwartende Aufwand für eine Sortierung mit einem Sortieralgorithmus der Aufwandsklasse $O(n * log(n))$ wird sich demzufolge in einem vertretbaren Umfang bewegen.

Gleichzeitig ist durch dieses Missverhältniss von Anzahl der real existierenden Kanten und Anzahl der Knoten der große Aufwand in der Rekursion zur Wegesuche erklärbar: die Zählschleife innerhalb der Rekursion, die über alle theoretisch möglichen Kanten des zu bearbeitenden Knotens iteriert, benötigt durch die vielen nicht existenten Kanten einen erheblichen Teil der gemessenen Rechenzeit zur Prüfung der Existenz.

Dieser Overhead ist durch eine Speicherung der tatsächlich existenten Kanten pro Knoten durch einen *Kanten-Index* zu lösen, über den die Zählschleife iteriert.

	Laufzeit der Suche in sec.					
	CH_4			C_2H_6		
untere Schranke:	1.0E-3	1.0E-4	1.0E-7	1.0E-3	1.0E-4	1.0E-7
max. Pfadzahl:	184	872	6.383	97	1.390	unbekannt
Pfadanzahl						
10	<1	<1	<1	<1	<1	<1
50	<1	<1	<1	<1	<1	1
100	<1	1	1	<1	1	2
250	1	1	1	-	1	13
500	-	1	10	-	2	28
1.000	-	1	10	-	2	45
2.500	-	-	37	-	2	297
5.000	-	-	51	-	-	478

Tabelle 5.4: Benchmark für Optimierung der Rekursion

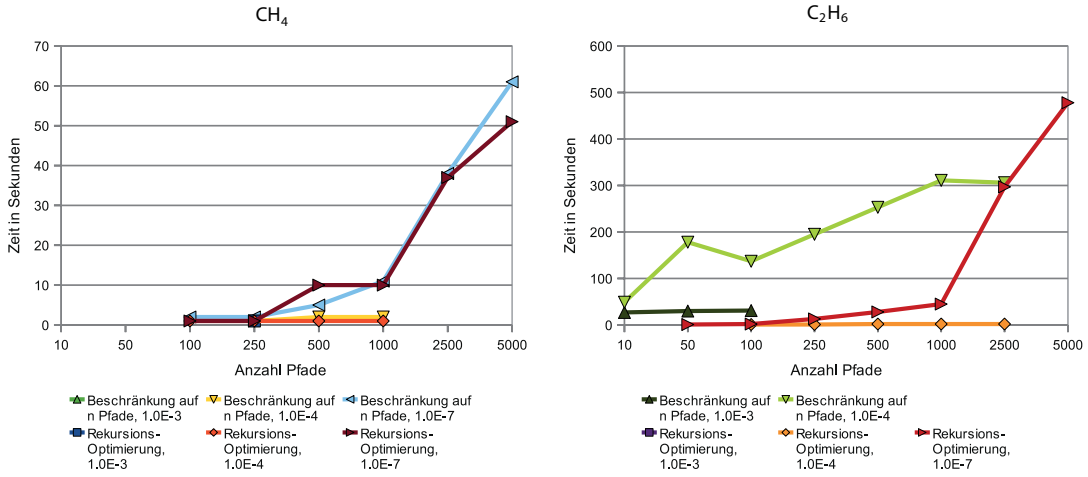

Abbildung 5.3: Vergleich der Optimierungen „Beschränkung auf n Pfade" und Rekursions-Optimierung.
Zeiten < 1 Sekunde sind nicht verzeichnet. Zu beachten ist die nun mögliche Berechnung von C_2H_6 mit einer unteren Schranke 1.0E-7, es existieren jedoch keine Vergleichswerte.
Links: Vergleich für Fragmentierungen von CH_4
Rechts: Vergleich für Fragmentierungen von C_2H_6

In Tabelle 5.4 und Abbildung 5.3 wird deutlich, welche Laufzeitoptimierung dieses Verfahren ermöglicht. Der Mehraufwand zur Speicherung des Kantenindex liegt in einem vertretbaren Rahmen, bezogen auf den Laufzeitvorteil. Die Messungen ergeben im Vergleich zur letzten Optimierung eine Erhöhung des Speicherbedarfs um etwa 1% von 4 auf

5%.

Bereits ohne eine Sortierung der Kanten vorzunehmen, wirkt sich diese Optimierung stark positiv auf das Laufzeitverhalten aus. Theoretisch sind mit dieser Optimierung auch Fragmentierungspfade der großen Moleküle wie C_3H_8 berechenbar. Problematisch ist nun nicht mehr unbedingt die Laufzeit der rekursiven Suche, sondern die zu Anfang des Kapitels genannten Punkte 5 und 6: diese beschreiben, dass die großen Moleküle nicht berechenbar sind aufgrund der Speicherprobleme bei der Aufstellung der Adjazenzstruktur. Darauf soll im nächsten Kapitel näher eingangen werden.

6 Vollständig rekursiver Ansatz

Die im letzten Kapitel angesprochenen Speicher- und Laufzeitprobleme bei der Aufstellung der Adjazenzstruktur, insbesondere für die Problemstellung C_3H_8, erfordern einen grundlegend neuen Ansatz. In diesem Kapitel soll zunächst dargelegt werden, aus welchen Gründen der vorherige Ansatz verworfen werden muss. Daran schließt sich die knappe Betrachtung des neuen Algorithmus an, zusammen mit einer Betrachtung der Unterschiede und Gemeinsamkeiten beider Ansätze. Eine Laufzeitanalyse ist darin eingeschlossen.

6.1 Begründung des Ansatzwechsels

Bei der Aufstellung der Adjazenzstruktur wird für jeden Knoten des Graphen eine Analyse der in diesem Zustand möglichen Zerlegungen durchgeführt. Dadurch entsteht kein Aufwand $O(n^2)$ für die Aufstellung, jedoch ist der Speicheraufwand für die Kanten um einen Faktor c größer als die Anzahl der Knoten. Tabelle 6.1 und Abbildung 6.1 zeigen, dass mit der Größe des zu zerlegenden Moleküls auch der Faktor c innerhalb der Molekül-Gruppe linear steigt.

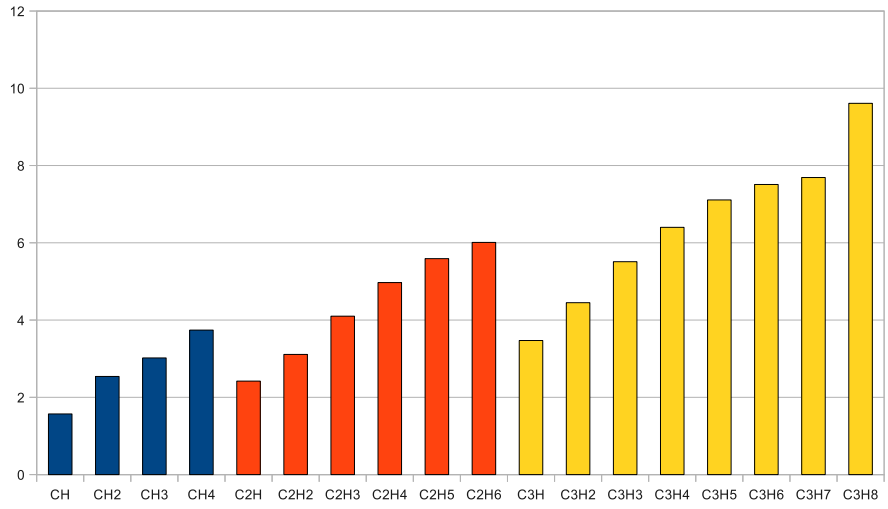

Abbildung 6.1: Faktor c zwischen Knoten und Kanten, aufgetragen nach Gruppen CH_X, C_2H_X, C_3H_X.
Ein ungefähr linearer Anstieg des Faktors ist erkennbar.

Ausgangsstoff	Knoten	Kanten	Faktor c
CH	14	22	1,57
CH_2	48	122	2,54
CH_3	119	359	3,02
CH_4	276	1.033	3,74
C_2H	104	252	2,42
C_2H_2	504	1.566	3,11
C_2H_3	1.360	5.575	4,10
C_2H_4	4.097	20.377	4,97
C_2H_5	9.506	53.157	5,59
C_2H_6	26.881	161.430	6,01
C_3H	825	2.863	3,47
C_3H_2	3.408	15.169	4,45
C_3H_3	10.375	57.119	5,51
C_3H_4	27.606	176.620	6,40
C_3H_5	62.565	444.855	7,11
C_3H_6	149.328	1.121.802	7,51
C_3H_7	412.296	3.169.721	7,69
C_3H_8	650.000	6.245.319	9,61

Tabelle 6.1: Faktor c zwischen Knoten und Kanten als linearer Faktor zur Abschätzung des Speicheraufwandes.
Für C_3H_8 können nur die Zahlen angegeben werden, die zum Zeitpunkt der letztmöglichen Iteration bekannt waren.

Durch die notwendige Betrachtung aller Knoten auf ihre Zerlegbarkeit und die eventuell folgende Analyse ist der Zeitaufwand hoch. Trotz des innerhalb einer Molekül-Gruppe linear wachsenden Rechenaufwandes sind die notwendigen Analysen als Teilprogramm für jeden Knoten zeitintensiv: für die Aufstellung der Adjazenzstruktur von C_2H_6 werden 50 Sekunden benötigt - vergleiche Tabelle 3.1 auf Seite 10. Da die Problemstellungen durch unterschiedliche Konfigurationsmöglichkeiten eine große Variabilität aufweisen, fehlt für eine verlässliche, generelle Aussage zur Abschätzung der Problemgrößen die Datenbasis.

Der Speicher- und Rechenaufwand reduziert sich mit einer Reduktion der Kantenanzahl bei der Aufstellung. Dazu gibt es zwei Möglichkeiten:

1. Durch Reduktion der Knotenzahl wird die Anzahl der zu analysierenden Kanten kleiner, da die Kanten der nicht betrachteten Knoten eingespart werden. Gleichzeitig müssen diese Kanten nicht mehr gespeichert werden, eventuell inklusive der folgenden Kanten und Knoten, sofern nicht andere Zustände auf diese verweisen. Allerdings gibt es aufgrund des Algorithmusdesigns keine direkte Möglichkeit, Knoten wegzulassen.

2. Durch Verwerfen von Kanten, deren Wahrscheinlichkeit die untere Schranke unter-

schreitet und damit das Einzelwahrscheinlichkeitskriterium verletzen. Zwar ist hier eine Analyse nötig, doch kann damit Speicherplatz eingespart werden.

Im bisherigen Ansatz wurde diese Optimierung versuchsweise eingebracht, jedoch nicht mit dem erhofften Erfolg.

6.1.1 Versuch der Optimierung durch Verwerfen von Kanten

Als Kriterium für das Verwerfen einer Kante wurde bereits oben die Einzel- und die Gesamtwahrscheinlichkeit genannt.

Bei der Aufstellung der Adjazenzstruktur ist es nicht möglich, die Gesamtwahrscheinlichkeit eines Weges für die Kante zu nutzen, da die Anzahl und die Wahrscheinlichkeiten der Wege zu dieser Kante unbekannt sind. Die Wege können erst nach dem vollständigen Aufbau der Struktur gefunden werden, da bei der Analyse eines Zustandes nicht bekannt ist, welche anderen, noch zu analysierenden Zustände auf diesen Zustand verweisen. Es findet daher nur die Prüfung mittels der Einzelwahrscheinlichkeit Anwendung.

Da die Anzahl der Reaktionen, die einen Zustandswechel hervorrufen, erst nach der vollständigen Analyse bekannt ist, kann eine Kante während der Analyse nicht unberücksichtigt bleiben. Vielmehr gilt folgendes Schema:

- Für jede Kante, die theoretisch existiert, wird die Summe aller Raten um die Rate erhöht. In der Adjazenzstruktur werden weiterhin die Summen, nicht die Wahrscheinlichkeiten hinterlegt.
- Ist eine Kante anzulegen, die in ihrer Wahrscheinlichkeit kleiner als die untere Schranke ist, wird die Kante in einem temporären Speicher abgelegt, da sie größer werden und dann die Schranke übersteigen kann.
- Erfährt die gleiche Kante (gleicher Zustandswechsel) ein Update im Sinne der Zusammenfassung einer Mehrfachkante durch Addition der Rate, wird die veränderte Wahrscheinlichkeit betrachtet. Im Falle, dass diese größer ist als die untere Schranke, wird die Kante in die Adjazenzstruktur übernommen. Die in der Adjazenzstruktur vorhandenen Kanten sind darauf zu prüfen, ob sie das Einzelwahrscheinlichkeitskriterium noch erfüllen.
- Erfährt eine Kante, die bereits Teil der Adjazenzstruktur ist, ein Update, oder wird eine gänzlich neue Kante in der Adjazenzstruktur angelegt, werden alle anderen in der Adjazenzstruktur hinterlegten Kanten darauf überprüft, ob sie die untere Wahrscheinlichkeitsschranke noch übersteigen. Ist dies nicht der Fall, werden sie aus der Struktur entfernt und in den temporären Speicher überführt. Die im temporären Speicher befindlichen Kanten brauchen keine Überprüfung, da die Wahrscheinlichkeit kleiner wird durch die Division der Rate mit der Summe der Raten, die durch das Update gewachsen ist.

Dadurch erhält man die Möglichkeit, Kanten und die eventuell dahinter liegenden Teilgraphen zu entfernen, sofern kein anderer Zustand auf den verworfenen verweist.

Da jedoch sehr häufig eine Prüfung der Kanten, die bereits angelegt worden sind, vorgenommen werden muss, wurde zunächst eine Analyse durchgeführt, wie oft in repräsentativen Fällen Updates durchgeführt werden. Updates stellen im Gegensatz zur reinen Anzahl an Kanten (Maximum für C_2H_6: 30 Kanten) den größeren Overhead an durchzuführenden Prüfungen dar.

	C_2H_6	C_3H_5
Kantenzahl gesamt:	161.430	444.855
	Anzahl der Kanten	
1 Update	12.324	34.490
2 Updates	3	5.045
4 Updates	4.308	9.592
9 Updates	449	1.001
14 Updates	-	109
Summe der Kanten:	17.084	50.237
% aller Kanten:	10,5829	11,2929

Tabelle 6.2: Anzahl der Kanten-Updates in repräsentativen Fällen.
Die Anzahl ist wichtig zur Abschätzung der Größe des Overheads für die Überprüfung des Einzelwahrscheinlichkeitskriteriums.

Wie aus der Tabelle 6.2 erkennbar ist, werden nur 10-11% der Kanten aktualisiert. Der Aufwand für die Überprüfungen im Vergleich zum erwarteten Gewinn an Geschwindigkeit und Speicherersparnis ist daher tolerierbar.

Bei der praktischen Anwendung des theoretisch sinnvoll erscheinenden Ansatzes stellte sich jedoch heraus, dass seine Vorteile marginal waren. Ein Entfernen der Kanten aufgrund des Einzelwahrscheinlichkeitskriteriums ist nicht effektiv, da nur wenige Kanten dieses Kriterium nicht erfüllen. Die Optimierung wurde daher als ineffektiv verworfen.

Um das oben ausgeschlossene Gesamtwahrscheinlichkeitskriterium nutzen zu können, das bei Aussortieren von Wegen effizienter als das Einzelwahrscheinlichkeitskriterium ist, ist daher ein neuer Ansatz nötig. Dieser muss die Problematik der Aufstellung der Adjazenz-Struktur umgehen und gleichzeitig das Kriterium zur Anwendung bringen können.

6.2 Beschreibung und Vergleich der Ansätze

Der neue Ansatz zur Lösung der Speicher- und Laufzeitprobleme bei der Aufstellung der Adjazenzstruktur soll von der bereits rekursiv implementierten Wegesuche so viel wie möglich übernehmen.

Eine erste Analyse ergab, dass die Zuordnungen von Zerlegungen zu Knotennummern und umgekehrt für die Berechnungen primär nicht relevant sind.

Die Einträge für C_2H_6 weisen im Durchschnitt eine Länge von 30 Zeichen (Byte) auf. Bei 2x ∼26.000 Einträgen für beide Zuordnungen, ist dies zu vernachlässigen. Für C_3H_8 ist

- a) ein höherer Durchschnitt aufgrund der größeren Menge von Zerlegungsprodukten zu erwarten und
- b) die Anzahl der Knoten mit mindestens 700.000 ein Multiplikator, der nicht vernachlässigbar ist.

Dennoch ist deutlich, dass in der Abhängigkeit von der Anzahl der Kanten der entscheidende Nachteil des bisherigen Ansatzes liegt. Eine Betrachtung der Gesamtwahrscheinlichkeit zeitgleich zur Analyse der Zerlegungen eines Zustandes ist jedoch nur möglich, wenn die Berechnung der Fragmentierungszustände erst stattfindet, sobald der Weg selbst gesucht wird.

Das bedeutet, dass die Wegesuche um das Zerlegen und Analysieren von Zuständen erweitert wird. Gleichzeitig können dadurch die wechselseitigen Zuordnungen von Nummern und Zerlegungen entfallen. Alle Zerlegungen, deren Gesamtwahrscheinlichkeiten des bisherigen Weges unter die Wahrscheinlichkeitsschranke fallen, werden nicht mehr Teil der Betrachtung, solange kein anderer Weg nochmals auf die gleiche Zerlegung stößt.

Die bisherige Beurteilung von Knoten als Endknoten eines Weges wird weitestgehend beibehalten, ebenso wie die sortierte Speicherung der Wege und Anhebung der unteren Schranke, wenn die n gewünschten Wege gefunden wurden. Einzige Änderung ist, dass die Summe der Raten nicht mehr als Kriterium für einen Endknoten genutzt wird (Summe gleich 0). Stattdessen gilt, dass keine weiteren Zerlegungen für den aktuell untersuchten Zustand existieren dürfen. Ebenfalls übernommen wird die Idee des Kanten-Indexes: die (meist nur wenigen) existierenden Kanten sind effektiv in einem Perl-Hash zusammen mit den Gewichten der Kanten speicher- und untersuchbar.

Pfadanzahl	Laufzeit in sec.			
	CH_4	C_2H_6	C_3H_5	C_3H_8
10	<1	9	10	98
50	1	14	21	163
100	4	21	42	246
250	4	49	134	733
500	12	37	457	703
1.000	12	88	278	1.955
2.500	32	451	831	3.781
5.000	65	997	1.168	4.233
Speicherbedarf:	1,8%	2,0%	2,4%	2,4%

Tabelle 6.3: Benchmark zum vollständig rekursiven Ansatz mit unterer Schranke 1.0E-7 (Gesamtlaufzeit, keine Teilbetrachtungen).
Speicherbedarf in Prozent von 1 GB RAM. C_3H_5 ist ausser Konkurrenz hier aufgeführt, der Fall ist für Testzwecke genutzt worden.

Wie aus der Tabelle 6.3 erkennbar ist, benötigt der neue Ansatz nur einen Bruchteil des Speichers, der für die Aufstellung der Adjazenzstruktur und die Speicherung der Wege

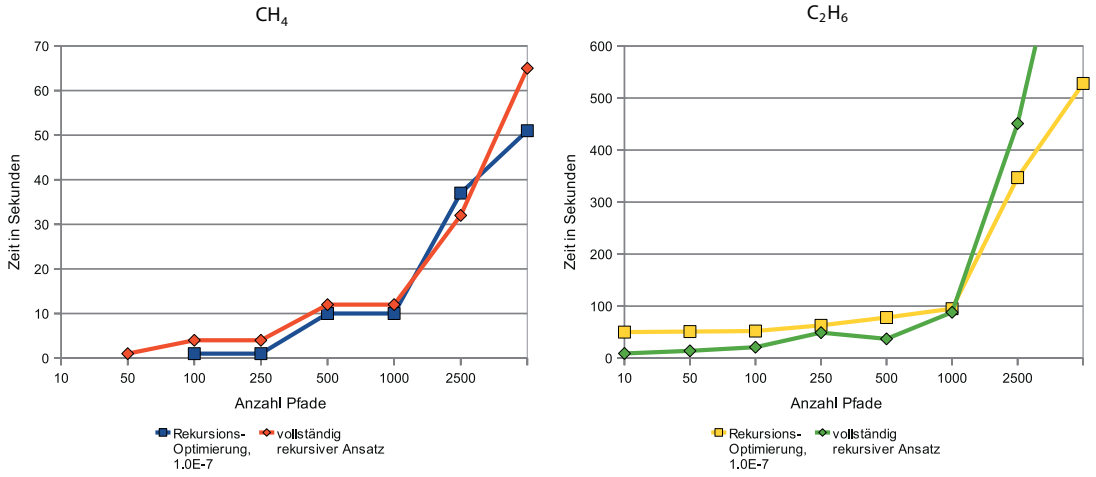

Abbildung 6.2: Vergleich der Laufzeiten des neuen Ansatzes mit der Rekursions-Optimierung des alten Ansatzes.
Zeiten < 1 Sekunde sind nicht verzeichnet. Zu beachten ist ausserdem, dass die Zeit für die Aufstellung der Adjazenzstruktur zur Zeit für die rekursive Suche addiert wurde.
Links: Vergleich für Fragmentierungen von CH_4
Rechts: Vergleich für Fragmentierungen von C_2H_6; Es wurde der Vergleich für 5.000 Pfade beschnitten, um die Vergleichbarkeit der Graphen zu erhöhen.

nötig war. Speziell für kleinere Anzahlen von Pfaden weist der neue Ansatz eine bessere Performance als der vorherige Ansatz auf (siehe dazu als Vergleich: Abbildung 6.2).

Wurde für 10 Pfade bei C_2H_6 mit der besten Optimierung im bisher verwendeten Ansatz eine Gesamtlaufzeit von etwa 50 Sekunden für Aufstellung der Adjazenzstruktur und Berechnung der Pfade benötigt, sind es nun nur noch insgesamt 9 Sekunden. Dies entspricht einer Laufzeitverringerung von ca. 80%. Erst ab 1.000 zu berechnenden Pfaden ist der Laufzeitunterschied beider Algorithmen marginal. Der bisherige Ansatz war für 5.000 Pfade besser geeignet (vergleiche Tabelle 5.4 auf Seite 24).

Auffällig bleiben die Laufzeiten, die für größere Pfadzahlen bei allen in der Tabelle aufgeführten Spezies auftreten. Besonders zu hinterfragen erscheint die Erhöhung der Laufzeit für 5.000 Pfade der Fragmentierung von CH_4 von 4 Sekunden in der Ausgangsimplementierung auf 65 Sekunden, sowohl in der optimierten Version des bisher verfolgten Ansatzes als auch im neuen Ansatz. Die folgenden Überlegungen sollen die Möglichkeit aufzeigen, dass durch Übernahme der Optimierungen für die Beurteilung auf Vollständigkeit eines Pfades und dessen Einsortierung diese Laufzeitprobleme entstehen könnten.

Die Einsortierung hat einen linearen Aufwand $O(n)$, wobei n die Anzahl der gespeicherten Pfade ist. Das Einfügen eines neuen Weges und seiner Wahrscheinlichkeit benötigt mehrere Operationen. Für m bereits gespeicherte Pfade gilt:

1. Zuerst muss anhand der Wahrscheinlichkeit eine Position a aus $m+1 \leq n$ möglichen Positionen gefunden werden, an der der Weg in das Feld eingefügt werden soll. Es wird immer eine solche gefunden, da die Wahrscheinlichkeit des Weges bereits vorher darauf geprüft wurde, ob sie größer als die untere Wahrscheinlichkeitsschranke ist (daher werden keine Wege mit zu kleinen Wahrscheinlichkeiten einsortiert).
2. Sofern die Position $a < m$ ist und zwischen den bereits gespeicherten Wegen eingefügt werden soll, müssen die Wege ab der Position a um 1 Element verschoben werden. Das Verschieben wird im Arbeitsspeicher durch Kopieren der Wege ab der Position a auf ihre neue, verschobene Position gelöst.
3. Sind bereits n Wege vor dem Einfügen gespeichert gewesen, erfolgt zuletzt noch die Löschung des überflüssigen $(n + 1)$. Weges.

Für den Fall, dass die Suche nach der Einfügeposition häufig für einen gefundenen Weg bis weit nach hinten im sortierten Feld läuft, ist der Aufwand für diese Suche groß, da immer alle Elemente vor der Position a geprüft werden müssen. Im Gegenzug ist die Anzahl der zu verschiebenden Elemente kleiner. Soll der Pfad näher zum Anfang des Feldes eingefügt werden, sind sehr viele Feldelemente (Wege) im Speicher zu verschieben.

Diese Überlegungen sind in der weiteren Entwicklung zu verifizieren. Ein erster Ansatz zur Verbesserung der Positionssuche wäre ein *Binary-Search-Algorithmus* mit einem Aufwand $O(log(n))$ statt eines linearen Durchlaufens (Aufwand $O(n)$) des Feldes. Für das Speichern der Wege ohne aufwändige Verschiebenoperationen bietet sich die Realisierung in einer zweifach verketteten Liste an.

Es ist jedoch zu beachten, dass Binary-Search nicht ohne weiteres in einer zweifach verketteten Liste funktioniert, da die Elemente der Liste nicht, wie bei Feldern der Fall, nebeneinander im Speicher angeordnet sind. Darum muss an dieser Stelle ein Kompromiss gefunden werden, der beide Aspekte gleichermaßen berücksichtigt.

Zu beachten ist außerdem, dass diese Problematik eventuell auch in den folgenden Optimierungen Einfluss auf die Benchmark-Ergebnisse nimmt.

6.3 Optimierung durch Zwischenspeicherung

Auch bei der vollständig rekursiven Bearbeitung der Zerlegungen besteht das Problem, dass Zustände mehrfach analysiert werden müssen, wenn sie Teil mehrerer Wege sind. Es ist daher eine sinnvolle Optimierung, diese Zerlegungen zu speichern.

Gespeichert werden muss der Zustand und welche Kanten mit welchem Gewicht von diesem aus existieren. Es stellt sich jedoch die gleiche Frage wie bei dem bislang verwendeten Ansatz: mit welchem Kriterium kann beurteilt werden, ob ein Zustand und alle Kanten dieses Zustandes gespeichert werden sollen?

Ergibt die rekursive Suche weitere gültige Wege zu einem oder mehreren Endknoten, ist es sinnvoll einen Knoten und die zugehörigen Kanten zu speichern. Da die Rekursionen

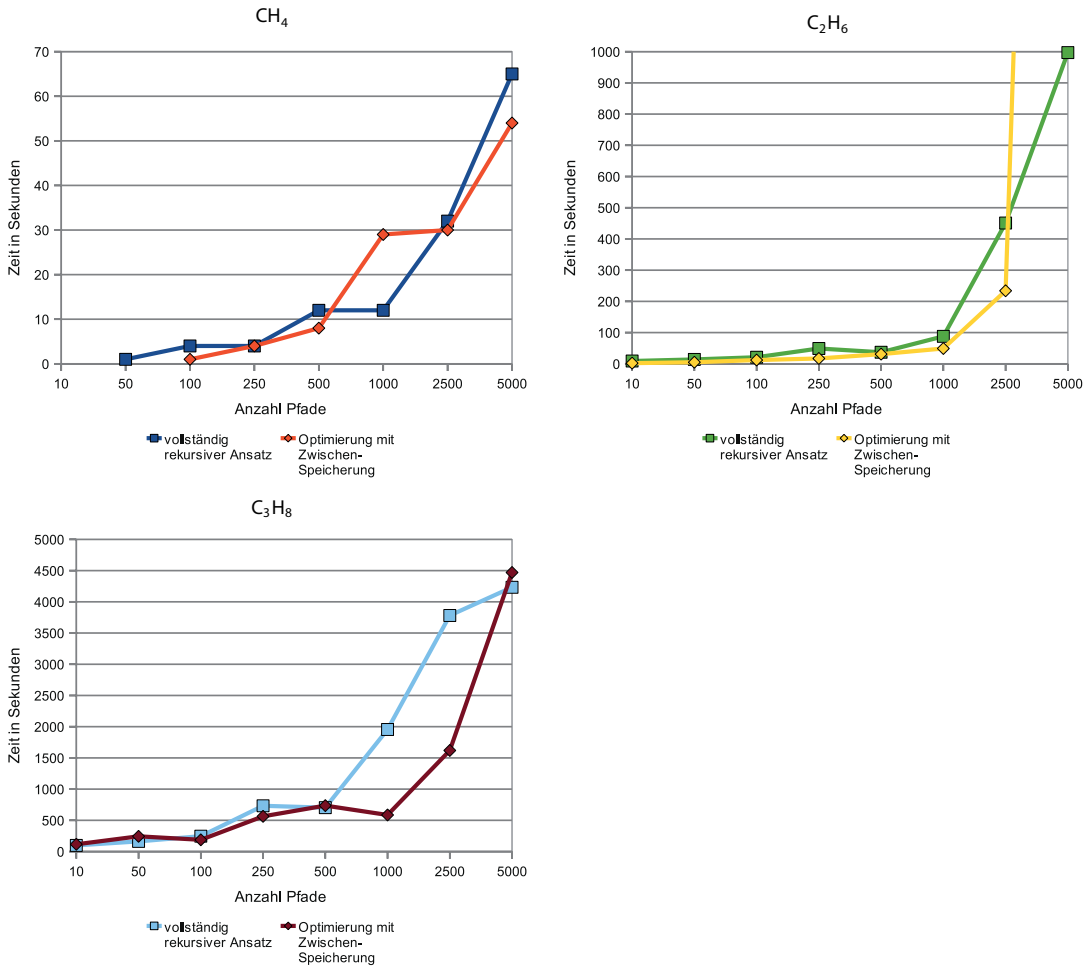

Abbildung 6.3: Vergleich der Laufzeiten des neuen Ansatzes und der Optimierung des Ansatzes durch Zwischenspeicherung.
Zeiten < 1 Sekunde sind nicht verzeichnet.
Links oben: Vergleich für Fragmentierungen von CH_4
Rechts oben: Vergleich für Fragmentierungen von C_2H_6; Es wurde der Vergleich für 5.000 Pfade beschnitten, um die Vergleichbarkeit der Graphen zu erhöhen.
Links unten: Vergleich für Fragmentierungen von C_3H_8

Pfadanzahl	Laufzeit in sec.		
	CH_4	C_2H_6	C_3H_8
10	<1	2	116
50	<1	5	244
100	1	12	188
250	4	17	564
500	8	31	736
1.000	29	49	586
2.500	30	234	1.619
5.000	54	6.152	4.470
Speicherbedarf:	1,8%	2,0%	2,4%

Tabelle 6.4: Benchmark der Optimierung durch Zwischenspeicherung mit unterer Schranke 1.0E-7.
Speicherbedarf in Prozent von 1 GB RAM.

nicht parallel ausgeführt werden können und das Design des Algorithmus so ausgelegt ist, dass das tiefere Eindringen in den Graphen bevorzugt wird, kann ein einzuführender Rückgabewert in der Rekursion darüber Auskunft geben, wie erfolgreich eine neue Rekursionsstufe war.

Sollte eine neue Rekursionsstufe von einem Zustand aus zu einem gültigen Weg führen, ist der Zustand zwischenzuspeichern. Der Zustand kann von späteren Rekursionen, die von anderen Knoten aus erneut auf diesen verweisen, ohne erneute Analyse genutzt werden.

Aus Tabelle 6.4 ist ersichtlich, dass diese Optimierung verglichen mit den Werten aus Tabelle 6.3 auf Seite 30 eine leichte Laufzeitverbesserung zu nicht oder nur geringfügig erhöhten Speicherkosten bietet. Abbildung 6.3 zeigt diesen Sachverhalt graphisch. Die Suche im Fall C_2H_6 für 5.000 Pfade ist eine gravierende Ausnahme; diese ist mit der neuen Optimierung aufwändiger als ohne die Optimierung.

Die markanten Unterschiede in den Laufzeit-Tests dieser Optimierung sind zum einen durch die zuvor beschrieben Probleme für das Einfügen von Pfaden erklärbar. Andererseits ist durch diese Optimierung nicht mehr eindeutig vorhersagbar, wieviele Analysen tatsächlich stattfinden. Steigt die Anzahl der zu speicherenden Pfade, so kann auf der einen Seite die Anzahl der tatsächlich nötigen Analysen durch die Zwischenspeicherung sinken, was eine Einsparung von Rechenzeit bedeutet. Gleichzeitig ist jedoch die Möglichkeit gegeben, dass die untere Schranke, die bei Erreichen der gewünschten Anzahl auf das Niveau der kleinsten Wahrscheinlichkeit der gespeicherten Pfade angehoben wird, für größere Anzahlen von zu speichernden Pfaden so klein ist, dass zuvor unberücksichtigte Wege nun analysiert werden müssen. Die Zwischenspeicherung ist für einen solchen Fall nicht mehr als gute Optimierung anzusehen, da die Pfade a) analysiert werden müssen und b) eventuell der Vorteil der Zwischenspeicherung wegen des einmaligen Besuchens von Knoten unwirksam wird.

Ein gutes Beispiel für diese markanten Laufzeitunterschiede ist die Analyse von C_2H_6:

wurden mit der grundlegenden Implementierung des rekursiven Ansatzes für die Analyse von 5.000 Pfade 997 Sekunden benötigt, steigt die Rechenzeit mit der Optimierung um den Faktor 6,17 auf 6.152 Sekunden. Für die Analyse von 1.000 Pfaden benötigte der nicht optimierte Algorithmus 88 Sekunden, der neue hingehen 49 Sekunden.

Für weitere Optimierungen bedeutet dies eine zusätzliche Unsicherheit, welche Ursachen bei auffälligen Laufzeitverhalten zu Grunde liegen.

7 Probleme, Ausblick

In diesem Kapitel soll ein weiterer Optimierungsansatz und die daraus verursachten Probleme angesprochen werden; es folgt eine kurze Zusammenfassung des Entwicklungsstandes und eine Ausblick auf die weitere Entwicklung.

7.1 Versuch der Optimierung durch Sortieren

Da die Gesamtwahrscheinlichkeit in Abhängigkeit von der Einzelwahrscheinlichkeit einer neuen Kante schneller oder langsamer sinken kann, wäre eine bevorzugte Betrachtung der Kanten mit hohen Wahrscheinlichkeiten sinnvoll.

In Perl werden die Indizes eines Hashes durch die Funktion keys() in einem zufällig geordneten Feld hinterlegt. Dieses Feld wird dazu benutzt über die vorhandenen Kanten zu iterieren und ggf. eine neue Rekursionsstufe anzustoßen. Deshalb werden im bisherigen Programm die Kanten in zufälliger Reihenfolge betrachtet.

Dieses Feld kann jedoch auch nach der Größe der Wahrscheinlichkeit der Kante sortiert werden, zu der der im Feld gespeicherte Index gehört. Die in Perl vorhandene Funktion sort(), die einen Merge-Sort-Algorithmus [**?**] mit einem Aufwand $O(n * log(n))$ implementiert, lässt sich dazu nutzen.

Pfadanzahl	Laufzeit in sec.		
	CH_4	C_2H_6	C_3H_8
10	<1	1	363
50	1	2	631
100	1	124	1.039
250	13	9	1.131
500	4	35	1.235
1.000	13	75	1.441
2.500	36	857	2.277
5.000	56	586	5.976
Speicherbedarf:	1,8%	2,0%	2,4%

Tabelle 7.1: Benchmark der Optimierung durch Sortierung.
Speicherbedarf in Prozent von 1 GB RAM. Es wird im Vergleich mit Tabelle 6.4 auf Seite 34 deutlich, dass die Optimierung nicht unbedingt von Vorteil ist.

Diese zunächst als sinnvoll zu betrachtende Sortierung führt nicht unbedingt zu einer verbesserten Laufzeit. In Tabelle 7.1 wird deutlich, dass für manche Fälle die Suche mit Sortierung eine erhebliche Laufzeitverlängerung erbringt. Abbildung 7.1 zeigt hier nochmals die Kontroverse zwischen den verschiedenen Optimierungen und dem nicht optimierten Ansatz.

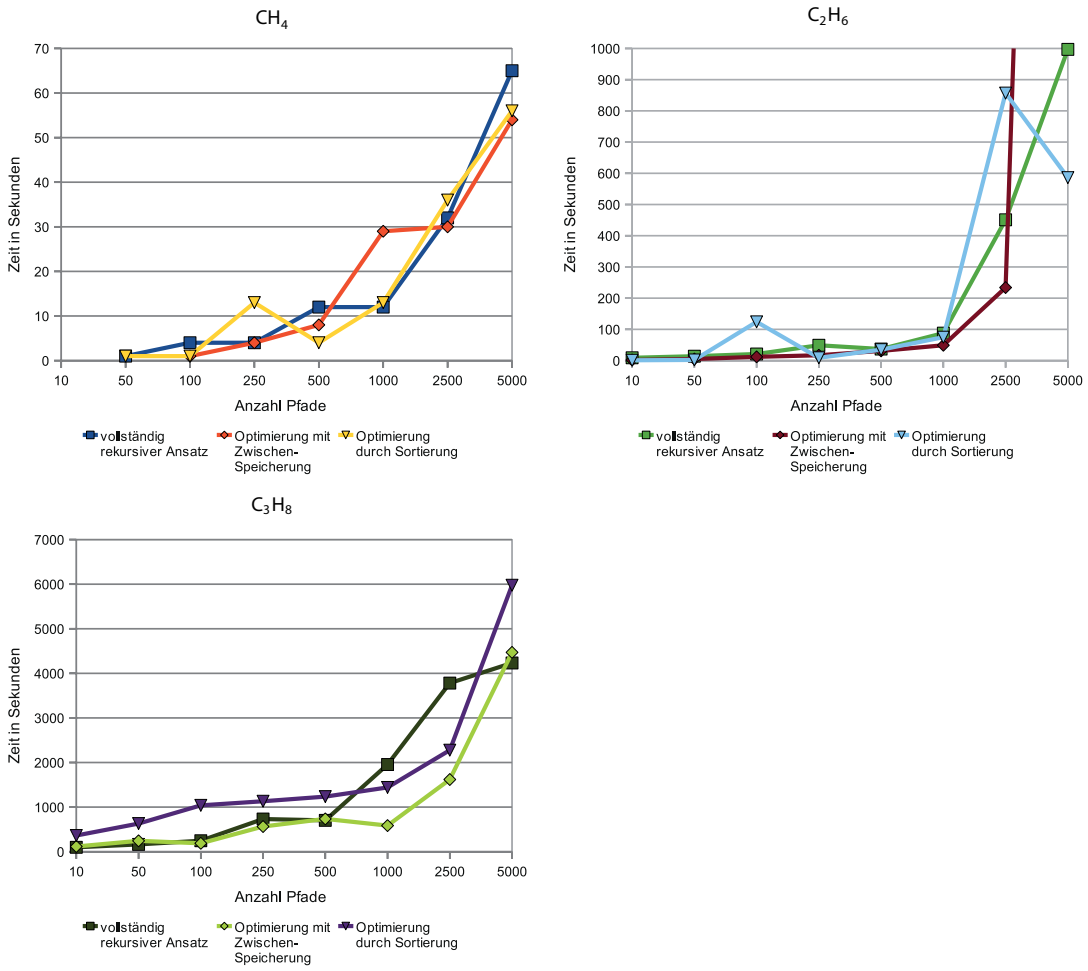

Abbildung 7.1: Vergleich der Laufzeiten des neuen Ansatzes und der beiden Optimierungen des Ansatzes.
Zeiten < 1 Sekunde sind nicht verzeichnet.
Links oben: Vergleich für Fragmentierungen von CH_4
Rechts oben: Vergleich für Fragmentierungen von C_2H_6. Es wurde der Vergleich für 5.000 Pfade beschnitten, um die Vergleichbarkeit der Graphen zu erhöhen.
Links unten: Vergleich für Fragmentierungen von C_3H_8

Ein Erklärungsansatz ist, dass diese Sortierung vorgenommen wird, obwohl über die

Kondition des Graphenproblems nichts bekannt ist. Eine Bevorzugung von Kanten mit großer Wahrscheinlichkeit kann sinnvoll sein, es ist jedoch unbekannt über wieviele weitere Kanten der Weg nach dieser Kante führt und ob die Wahrscheinlichkeit ggf. im nächsten Schritt rapide abfällt. Somit wäre eine Betrachtung der Kanten mit kleinerer Wahrscheinlichkeit, aber ohne sprunghafte Abstiege, eine Verbesserung. Ein Beispiel für eine solche Entwicklung zeigt Bild 7.2.

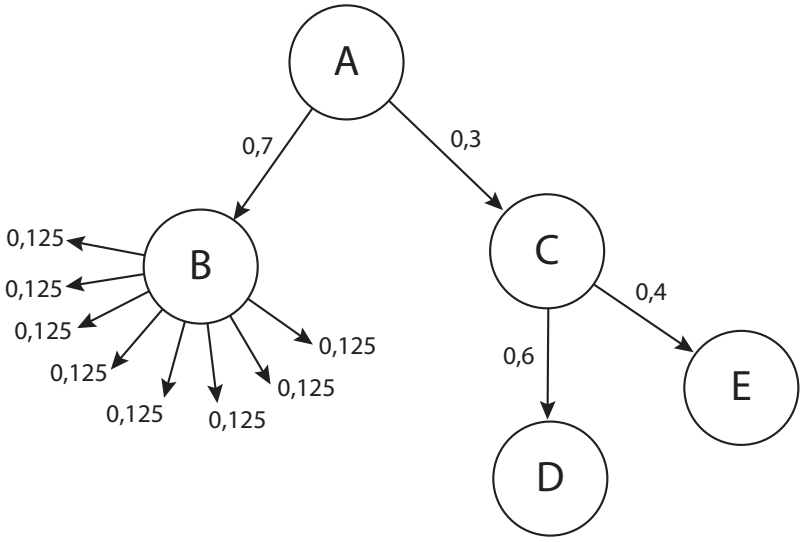

Abbildung 7.2: Beispielhafter Graph mit Kantengewichten im Sinne von Wahrscheinlichkeiten.
Die Wahrscheinlichkeit nimmt hinter dem Knoten B drastisch ab, wohingegen die Wahrscheinlichkeit über den Knoten C größer bleibt und somit die besseren Wege bietet.

Der Weg $A \rightarrow B \rightarrow ...$ erscheint am Knoten A mit einer Wahrscheinlichkeit von $0,7$ der geeignetere zu sein, jedoch beträgt die Gesamtwahrscheinlichkeit des Weges nach Knoten B nur noch maximal $0,7 * 0,125 = 0,0875$. Die Wege $A \rightarrow C \rightarrow D$ und $A \rightarrow C \rightarrow E$ weisen für $A \rightarrow C$ eine kleinere Wahrscheinlichkeit als der Weg $A \rightarrow B$ auf, doch liegt die Gesamtwahrscheinlichkeit mit $0,3 * 0,4 = 0,12$ für $A \rightarrow C \rightarrow E$ und $0,3 * 0,6 = 0,18$ für $A \rightarrow C \rightarrow D$ insgesamt höher.

Sind die Konditionen der Teilgraphen am Knoten A unbekannt, ist es nicht möglich, mittels Sortierung der Wahrscheinlichkeiten für alle möglichen Fälle eine Laufzeitverbesserung zu erhalten. Im Anschluss an diese Bachelor-Arbeit ist daher zu untersuchen, ob ein Kriterium gefunden werden kann, das es erlaubt, eine dynamisch anzuwendende Sortierung einzusetzen, um in allen Fällen eine Laufzeitverbesserung zu erhalten.

7.2 Zusammenfassung und Ausblick

Im Rahmen der Bachelorarbeit wurden Verbesserungen an der vorhandenen Implementierung vorgenommen und ein neuer Ansatz entwickelt. Es wurde damit eine Berechnung der wahrscheinlichsten Pfade des größten vorhandenen Zerlegungsproblems C_3H_8 ermöglicht. Die Berechnung der kleineren Fragmentierungsprobleme wurde wesentlich beschleunigt.

Die Berechnungen des neuen Ansatzes konnten für kleinere Zerlegungen (z. B. CH und CH_4) mit der bisherigen Implementierung verglichen werden. Da die Verifikation übereinstimmende Werte ergab, erscheint der hier erstellte Algorithmus als korrekt.

Die Optimierung ist damit jedoch nicht abgeschlossen. Weitere Tests müssen ergeben, wie stabil und fehlerkritisch der Algorithmus arbeitet. Die Überlegungen auf Seite 32 zum varierenden Laufzeitverhalten sind ebenfalls weiter zu verfolgen (siehe Tabelle 6.3, S. 30).

Die in der Problemstellung geforderten 120 Sekunden Laufzeit für 10 Zerlegungspfade von C_3H_8 mit einer unteren Wahrscheinlichkeitsschranke von 1.0E-7 wurden erreicht. Mittels Implementierung der Optimierung durch Sortieren wurde dieser Rahmen durch die Laufzeit von 363 Sekunden jedoch weit überschritten. Es ist daher zu untersuchen, wie sehr das Sortieren die Konditionierung des weiteren Teilgraphen nicht beachtet und/oder ob eventuell das Sortieren selbst ein Problem darstellt. Die Betrachtung der maximal pro Knoten vorhandenen Kanten wurde nur für C_2H_6 durchgeführt, da C_3H_8 mit dem vorher genutzten Ansatz in keiner Weise berechenbar ist.

Erstrebenswert ist die Reduktion der Laufzeit von C_3H_8 auf die in den Rahmenbedingungen angesprochenen 60 Sekunden. Zu diesem Zweck werden weitere Untersuchungen notwendig sein, die auch die Eignungsanalyse anderer Verfahren, beispielsweise aus dem *Operations Research*, einschließen. Weiterhin ist in anschließenden Analysen die Betrachtung von Laufzeiten mit Nicht-Standard-Konfigurationen durchzuführen. Im Rahmen der Qualitätssicherung sind vor einer Freigabe für die Produktionsversion der Software gesondert Tests zu absolvieren.