Referenzintervallgraphen
(probe interval graphs)

Studienarbeit
Universität Rostock, Fachbereich Informatik

vorgelegt von: Schulz, Hans-Jörg
geboren am 26. Februar 1979 in Rostock
Matrikelnummer: 099201929

Betreuer: PD Dr. Van Bang Le
Abgabedatum: 02.12.2003

Inhaltsverzeichnis

1 Einführende Begriffsklärungen

Definition 1 *Sei $G(V, E)$ ein Graph mit endlicher*[1] *Knotenmenge V und zugehöriger Kantenmenge E. Dann bezeichnet $G(X)$ den durch die Menge $X \subseteq V$* ***induzierten Teilgraphen*** *von $G(V, E)$.*

Definition 2 [5] *Eine* ***Clique*** *ist ein (Teil-)Graph, dessen Knoten vollständig durch Kanten verbunden sind. $(\forall x, y \in V \ \exists xy \in E)$*

Definition 3 [18] *Bezeichne $x \in V$ einen Knoten des Graphen $G(V, E)$. Dann ist die Menge $\{y \mid xy \in E\}$ die* ***Nachbarschaft*** *$N(x)$ von x. Die* ***abgeschlossene Nachbarschaft*** *$N[x]$ ist die Menge $\{y \mid xy \in E\} \cup x = N(x) \cup x$.*

Definition 4 [6] *Ein Graph $G(V, E)$ heißt* ***chordal*** *oder* ***trianguliert****, wenn G keinen Kreis der Länge ≥ 4 als induzierten Teilgraphen hat. $G(V, E)$ ist* ***schwach trianguliert***[2]*, wenn er und sein Komplement $\overline{G}$ keinen induzierten Teilgraphen enthalten, der isomorph zu einem sehnenlosen Kreis C_k mit $k > 4$ ist.*

Definition 5 [21] *Ein Graph $G(V, E)$ ist* ***perfekt****, wenn seine chromatische Zahl $\chi(G)$ (siehe z.B. [5, 8]) für alle induzierten Teilgraphen der jeweiligen maximalen Cliquengröße entspricht.*

Definition 6 [8] *Ein Graph $G(V, E)$ ist ein* ***Intervallgraph***[3]*, wenn es eine Menge reeller Intervalle $\{I_v \mid v \in V\}$ gibt, die einander genau dann* ***überlappen*** *$(I_u \cap I_v \neq \emptyset)$, wenn $uv \in E$ ist.*

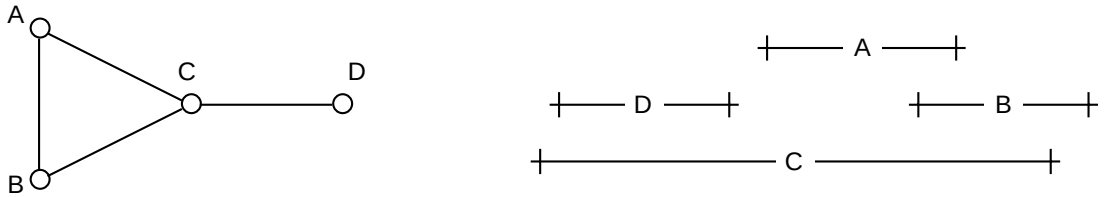

Abbildung 1: Beispiel eines Intervallgraphen und seiner Intervalldarstellung

Definition 7 [23] *Ein gerichteter Graph heißt* ***transitiv****, wenn er mit seinem transitiven Abschluß übereinstimmt, d.h. sind (e, f) und (f, g) Kanten des gerichteten Graphen G, dann ist auch (e, g) eine Kante von G. Ein ungerichteter Graph heißt* ***transitiv orientierbar****, falls er eine kreisfreie transitive Orientierung besitzt. Jede transitive Orientierung eines Graphen liefert eine eindeutige Reihen-*

[1] Sämtliche Ausführungen dieser Arbeit beschränken sich auf endliche Graphen.
[2] engl. weakly triangulated
[3] engl. interval graph

folge der Knoten (topologisches Sortieren [5, 23]). Graphen, die eine transitive Orientierung besitzen, gehören der Klasse der ***Vergleichbarkeitsgraphen***[4] *an. Graphen, die transitiv orientierbar sind und deren Komplement ebenfalls ein Vergleichbarkeitsgraph ist, heißen* ***Permutationsgraphen***[5]*.*

2 Motivation und historische Bemerkungen

Das menschliche Genom ist in seiner Vielgestaltigkeit der wohl komplexeste Datensatz der Natur. Die in ihm enthaltenen Erbinformationen codieren nicht nur äußere Erscheinungsmerkmale wie die natürliche Augen- und Haarfarbe, sondern z.B. auch das erblich bedingte Schlaganfallrisiko oder die erbliche Veranlagung zu Depressionen. Dies verleiht der Erforschung des Genoms eine beträchtliche Brisanz und wenn man der modernen Wissenschaft Glauben schenken darf, zählt sie sogar zu den vordringlichsten Aufgaben unsere Zeit.

Bereits im Jahr 1871 wurde die Desoxyribonukleinsäure[6] (kurz: DNS) erstmals vom Schweizer Biologen Friedrich Miescher (1844-1895) in Tübingen aus Zellkernen isoliert. Es dauerte dennoch bis 1943, daß der Bakteriologe Oswald Avery (1877-1955) und seine Mitarbeiter die ersten Hinweise entdeckten, wonach eben jenes von Miescher gefundene Makromolekül der Träger der Erbinformationen ist. Zehn Jahre später, im April des Jahres 1953, veröffentlichten James Watson (1928-) und Francis Crick (1916-) die räumliche Struktur der DNS [26]. Vier Basen (Adenin, Cytosin, Guanin und Thymin) verbinden dabei paarweise zwei gewundene Zucker-Phosphat-Stränge zu der weltberühmt gewordenen Doppelhelixstruktur. Für ihre Arbeiten über die Struktur der DNS erhielten Watson und Crick 1962 den Nobelpreis für Medizin.

Die unterschiedlichen Anordnungen der einzelnen Basen in diesem DNS-Strang codieren nun das Erbgut des zugehörigen Organismus'. Beim Menschen besteht das gesamte Genom aus ca. $3,2 \times 10^9$ Basenpaaren, die jeweils eine 2 Bit große Information codieren. Die menschliche DNS hat also ein Informationsvolumen von rund 763 MByte, also ungefähr einer 90-Minuten-CD. Wenn man allerdings in Betracht zieht, daß lediglich 3% davon spezifizierte, sinntragende Information ist, reduziert sich das Informationsvolumen auf nicht einmal 23 MByte. Um aus diesen 23 MByte jetzt aber abzulesen, ob nun z.B. ein erhöhtes Schlaganfallrisiko vorliegt oder nicht, müssen erst die semantischen Informationseinheiten des Genoms, die Gene, mit ihren Wechselwirkungen bekannt sein. Um sie zu finden, hält die moderne Informatik ausgeklügelte Data-Mining-Algorithmen bereit, die im Zusammenspiel mit dem KnowHow der Biotech-Industrie nahe-

[4] engl. comparability graph
[5] engl. permutation graph
[6] engl. deoxyribonucleic acid oder kurz: DNA

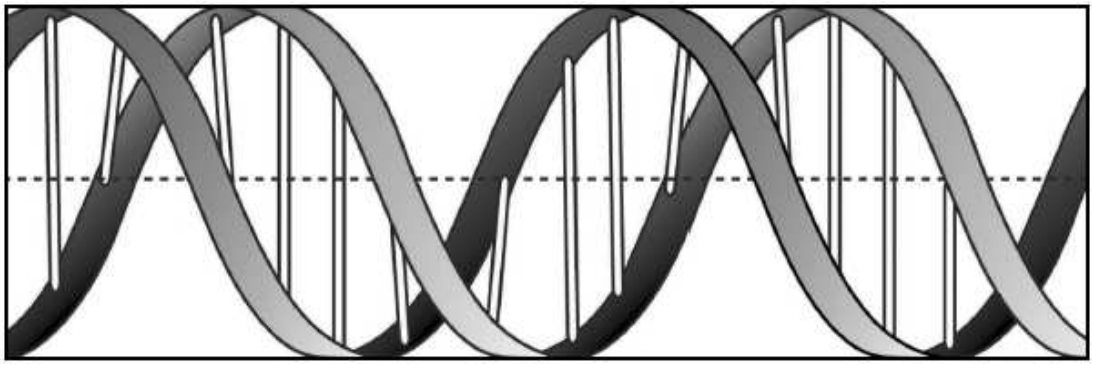

Abbildung 2: Die Doppelhelixstruktur der DNS

zu täglich neue Ergebnisse auf diesem Gebiet liefern.[7] Bevor man sich aber der Aufgabe der (halb-)automatischen Identifizierung von Genen in der Fülle der Erbinformationen zuwenden kann, müssen die als biochemische Verbindungen „gespeicherten“ Informationseinheiten, die Basenpaare, in ein maschinell zu verarbeitendes Format umgewandelt werden – der DNS-Strang wird „sequenziert“. Sequenzierautomaten übernehmen dies für Teilstücke der DNS mit Hilfe moderner Kapillarelektrophorese[8] - je nach Ausführung bis zu einer Länge von ca. 1000 Basenpaaren. Leider garantieren derartige Automaten lediglich einen Anteil von 98,5% korrekt sequenzierten Basenpaaren, so daß es sogar beim bekannten „Human Genom Projekt“, welches sich der Sequenzierung der menschlichen DNS widmet, noch immer unstimmige Teilsequenzen gibt. Diese werden in gewissen Abständen nochmal nachsequenziert, sobald die dafür nötige verbesserte Technik verfügbar ist. Auf Basis der Teilsequenzen wird ungefähr dreimal jährlich eine neue Gesamtsequenz berechnet.[9]

[7]Eine weiterführende Übersicht zu den verwendeten Methoden findet der interessierte Leser z.B. in der „Genomics, Proteomics and Bioinformatics Knowledge Base“ unter http://www.123genomics.com. Eine kurze Zusammenfassung der ersten Ergebnisse kann dem Artikel „Guide to the draft human genome“, Nature 409, 02/15/2001, 824-826, entnommen werden.

[8]Eine deutschsprachige Einführung in dieses Gebiet der analytischen Chemie inkl. einer Liste weiterführender Literatur ist unter http://www.kapillarelektrophorese.de verfügbar.

[9]Die Gesamtsequenzen für eine Reihe ausgewählter Organismen darunter die des Menschen und des SARS-Virus stehen unter http://genome.cse.ucsc.edu zur Einsicht und zum Download frei zur Verfügung.

Abbildung 3: Der 145.000 US$ teure Sequenzierautomat „ABI PRISM®3100 Genetic Analyzer“ der Firma Applied Biosystems sequenziert DNS-Stränge mit einer Länge von max. 750 Basenpaaren in 16 parallel arbeitenden Kapillaren [1].

3 Physikalische Kartierung

Der Begriff der „physikalischen Kartierung“[10] umfaßt die Positionierung von DNS-Fragmenten unterschiedlicher Größenordnungen bezüglich eines noch größeren, übergeordneten DNS-Abschnitts. Dabei werden genau wie bei der geographischen Kartierung unterschiedliche Detailstufen benötigt: sucht man ein Land, so nutzt man dafür einen Atlas - sucht man eine Stadt, so ist eine Landkarte geeigneter - und um schließlich eine bestimmte Straße zu finden, braucht man einen Stadtplan. Um bei der Kartierung der DNS ebenfalls mehrere Detailstufen zu erhalten, nutzt man folgende Aufteilungen der DNS:

- Das Genom ist bereits von sich aus auf eine Anzahl von Chromosomen verteilt. Diese bilden auf natürliche Weise die niedrigste Detailstufe der physikalischen Kartierung.

- Die Chromosomen werden weiterhin in Teilstränge geteilt. Diese Teilstränge haben immer noch einen Umfang von 500.000 bis 2.000.000 Basenpaaren.

[10] in der Literatur auch unter den Namen „physische Kartierung“ und „zytogenetische Kartierung“ zu finden; engl. physical mapping

- In der nächsten Stufe werden die besagten Teilstränge weiter in größere Fragmente mit rund 40.000 Basenpaaren zerteilt.

- Danach geschieht dies noch einmal, um endlich kleine Fragmente mit einer ungefähren Länge von 4.000 Basenpaaren zu erhalten.

- Zu guter Letzt werden diese kleinsten DNS-Fragmente sequenziert, um die höchste Detailstufe zu erhalten: die einzelnen Basenpaare.

Das Zerteilen eines größeren Stranges in kleinere Abschnitte wird dabei meist mit Hilfe von Enzymen, sogenannten Nukleasen, durchgeführt. Es ist einsichtig, daß die folgende Auswahl an Fragestellungen mit physikalischen Karten unterschiedlicher Detailstufen beantwortet werden kann:

- Auf welchem Chromosom liegt ein bestimmtes Gen? (engl. gene mapping)

- Welche Abschnitte der DNS enthalten überhaupt sinntragende Information? (engl. transcript mapping)

- An welchen Stellen brechen bestimmte Restriktionsenzyme den DNS-Strang auf? (engl. restriction mapping)

Zur Beantwortung solcher Fragen werden je nach Bedarf die unterschiedlichen Detailstufen zu einem Gesamtbild zusammengefügt[11]. Das Vorgehen entspricht insgesamt also einem „Teile-und-Herrsche“-Verfahren[12]. Leider ist der letztgenannte Schritt des Zusammensetzens aus mehreren Gründen ein äußerst problembehaftetes Unterfangen. Sämtliche bekannte Verfahren nutzen dazu Informationen über die Fragmentüberlappung, die entsteht, wenn mehrere Kopien eines Ausgangsstranges an unterschiedlichen Stellen aufgetrennt werden. Formal könnte die Beschreibung dieses **„Verknüpfungsproblems“** so lauten:

geg.: `n einander teilweise überlappende DNS-Fragmente`
`in einer Fragment-Bibliothek (engl. clone library)`

ges.: `die Positionen der Fragmente innerhalb eines übergeordneten`
`DNS-Stranges einer niedrigeren Detailstufe`

Dabei ist die besagte Ausgangssequenz meist nicht vorhanden, sondern soll eben erst durch korrektes Aneinanderreihen der Fragmente entstehen. Die Schwierigkeiten beim Lösen dieses Problems stecken mitunter im Detail:

[11] engl. sequence assembly
[12] engl. divide and conquer

- Aus den Fragmenten läßt sich zuweilen kein zusammenhängender Ausgangsstrang rekonstruieren. Es entstehen sogenannte „Inseln“, deren relative Entfernung und Position ungewiß bleibt.

- Erbinformationen bestehen zu einem beträchtlichen Teil aus sich wiederholenden Basensequenzen, sogenannter „repetitiver DNS“[13]. Beim Menschen machen diese Wiederholungen immerhin 45% des gesamten Erbmaterials aus. Dadurch ist es fast unmöglich eindeutige und konsistente Ausgangssequenzen zu berechnen.

- Zur Bestimmung der Informationen über die Fragmentüberlappung kommen häufig experimentelle Verfahren der analytischen Chemie zum Einsatz, die selbst bei genauer Beachtung der Randbedingungen Schwankungen und Meßfehlern unterworfen sind.

Der letzte Punkt soll nachfolgend am Beispiel des verbreiteten Hilfsmittels der Hybridisation, also das Zusammenlagern komplementärer DNS-Stränge verdeutlicht werden.

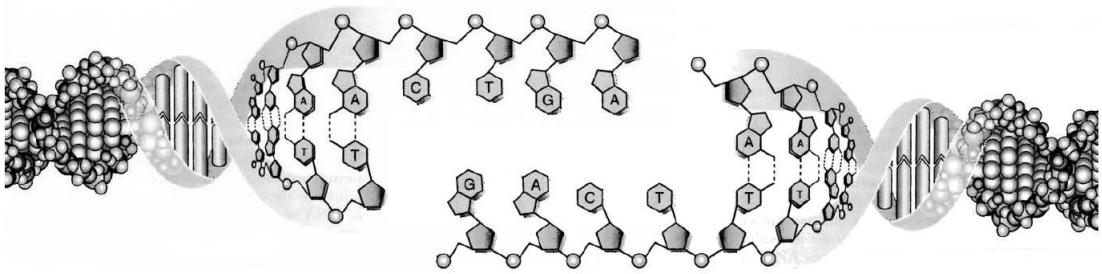

Abbildung 4: Schematische Darstellung zweier DNS-Abschnitte, die einander an den Enden überlappen, also miteinander hybridisieren würden.

Da es für große Fragment-Bibliotheken praktisch unmöglich ist, jedes Fragment mit jedem auf Überlappung zu testen, wird eine Teilmenge P der in der Bibliothek enthaltenen Fragmente[14] als sogenannte Referenzmenge[15] ausgewählt. Dann wird jedes Fragment, das nicht als Referenz gewählt wurde, darauf überprüft, ob es mit einer der Referenzen hybridisiert, also sich mit ihr zusammenlagert. Denn das ließe auf eine Überlappung der Referenz mit der getesteten Nicht-Referenz[16] schließen. Basierend auf den Versuchsergebnissen der Hybridisation, wird eine binäre Hybridisationsmatrix aufgestellt, deren Zeilen den getesteten Nicht-Referenzen und deren Spalten den Referenzen entsprechen.

[13]Die Bedeutung repetitiver DNS für die Evolution wird ausführlich in [13] diskutiert.
[14]engl. clones
[15]engl. probes
[16]engl. non-probe

Dies ergibt eine Matrix $A = [a_{ij}]$ mit

$$a_{ij} := \begin{cases} 1 & : \quad Referenz\ j\ hybridisiert\ mit\ Nichtreferenz\ i \\ 0 & : \quad sonst \end{cases}$$

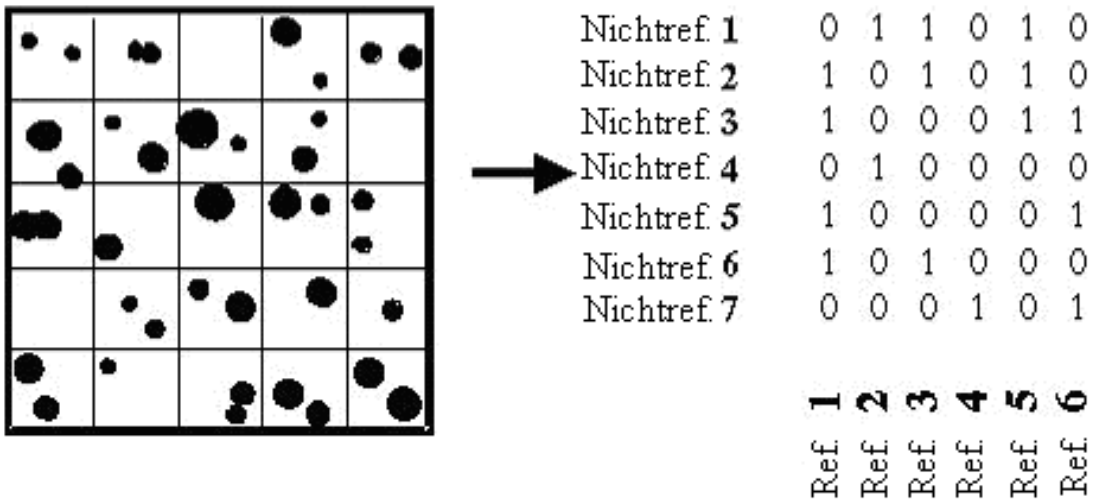

Abbildung 5: Die resultierende binäre Hybridisationsmatrix

Wie man leicht sieht, ist die Zuordnung der Hybridisationsergebnisse zu einem Matritzeneintrag alles andere als eindeutig und stark von dem gewählten Schwellwert abhängig, der bestimmt, ab wann ein Versuchsergebnis als erfolgreiche Hybridisation gewertet wird.

Hat man z.B. durch geschickte Wahl der Referenzen[17] erst einmal eine einigermaßen verläßliche Hybridisationsmatrix erstellt, reduziert sich das Verknüpfungsproblem auf ein rein kombinatorisches – dem **Problem der Kopplungsanalyse**[18]:

geg.: `die Menge P der ausgewählten Referenzen,`
`binäre Hybridisationsmatrix mit Informationen zur Überlappung`

ges.: `die relativen Positionen aller Fragmente`

Es ist zu beachten, daß jetzt „nur noch“ die **relativen** Positionen der Fragmente gesucht sind. Durch die Reduktion des Problems auf (partielle!) Informationen

[17] engl. probe pre-selection
[18] engl. linkage mapping

über die Überlappungen, ist keine Information über das Ausmaß derselben vorhanden. Ob ein Fragment eine Referenz nur zu einem Viertel oder sogar zur Hälfte überlappt, ist unbekannt. Daher läßt sich höchstens eine Reihenfolge der Fragmente ermitteln. Doch auch diese relativen Fragmentpositionen wären wertvolle Ergebnisse, die zusammen mit einer Sequenzierung der Fragmente auf triviale Weise zu einer Gesamtsequenz verbunden werden könnten. Dennoch stellt sich auch dieses reduzierte Problem, eine Reihenfolge der Fragmente zu finden, als nicht trivial heraus. So ist es in der Praxis z.B. üblich, bestimmte Spalten der Hybridisationsmatrix im Nachhinein nach gewissen pragmatischen Gesichtspunkten wieder zu löschen, wenn sie die Komplexität und damit die Laufzeit der Lösungsalgorithmen stark negativ beeinflussen könnten.[19] Eine andere Variante, die allerdings noch nicht vollends ausgereift ist, versucht das Verknüpfungsproblem von vornherein zu vermeiden. Hierbei wird der DNS-Strang vor dem Zerlegen in einzelne Fragmente mit einer Art Geleehülle umgeben. Die DNS wird in dieser Geleehülle durch Enzyme aufgebrochen, ohne sich dabei aber zu verteilen. Die Geleehülle kann nun von einem Ende aus nach und nach aufgelöst werden und die Fragmente freigeben, welche dann exakt in der richtigen Reihenfolge vorliegen. Dieses Verfahren ist natürlich nur auf den unteren der beschriebenen Detailebenen zweckmäßig, da die Fragmente dort noch groß genug sind, um sie garantiert **hintereinander** freisetzen zu können.

Die Relevanz effizienter Algorithmen für die Lösung des Problems der Kopplungsanalyse wird deutlich, wenn man bedenkt, daß die Sequenzierung des menschlichen Genoms bereits im Juni 2000 abgeschlossen war – die zusammengesetzte Gesamtsequenz aber trotz modernster Rechentechnik erst im Februar 2001 veröffentlicht werden konnte [13].

4 Referenzintervallgraphen

In den nachfolgenden Absätzen wird davon ausgegangen, daß die Teilmenge der Referenzen bereits bestimmt wurde und eine verläßliche Hybridisationsmatrix für die zugehörige Fragment-Bibliothek vorliegt. In diesem Fall kann das Problem der Kopplungsanalyse auch mit graphentheoretischen Hilfsmitteln formalisiert werden[20]. Der Bioinformatiker Peisen Zhang beschrieb dazu 1994 in einem Manuskript eine Erweiterung der gut erforschten Klasse der Intervallgraphen: die Referenzintervallgraphen[21] [28]. Bereits im Jahr 1959 hatte Seymour Benzer bei der Arbeit mit DNS-Fragmenten eines bestimmten Virus' festgestellt, daß

[19] engl. postscreening

[20] Wenn hingegen „verrauschte“, unzuverlässige Eingangsdaten vorliegen, sollten die einschlägigen heuristischen und statistischen Verfahren den kombinatorischen Lösungsalgorithmen vorgezogen werden. Ein kurze Übersicht findet sich dazu in [27]

[21] engl. Probe Interval Graphs oder Interval Probe Graphs; Abkürzung: PIG

ihre Überlappungen einen Intervallgraphen bilden [3]. Seither nehmen Intervallgraphen und ihre zirkulären Verwandten, die analog definierten Kreisbogengraphen[22], einen festen Platz in der Modellierung genetischer Strukturen ein [25].

Definition 8 [28] *Sei P eine Teilmenge der Knotenmenge V eines ungerichteten Graphen. Dann ist der Graph $G(V, P, E)$[23] ein **Referenzgraph**[24], wenn die Kantenmenge $E \subset (P \times V)$ ist. Die Menge P heißt dann **Referenzmenge**; ihre Elemente werden **Referenzen** genannt.*

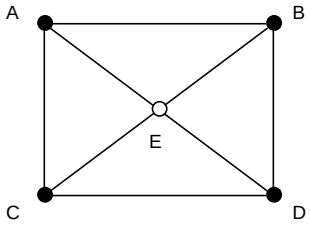

Abbildung 6: Beispiel für einen Referenzintervallgraphen. ($\bullet \in P$ = Referenzen, $\circ \in V \setminus P$ = Nicht-Referenzen)

Definition 9 [28] *Sei V eine endliche Menge von Intervallen des reellen Zahlenstrahls und $P \subseteq V$. Dann ist der Referenzgraph $G(V, P, E)$ mit der Knotenmenge V bezüglich der Referenzmenge P ein **Referenzintervallgraph**, wenn die Kantenmenge $E \subset (P \times V)$ nur aus Paaren (v_i, v_j) besteht, für die gilt: $v_i \cap v_j \neq \emptyset$.*

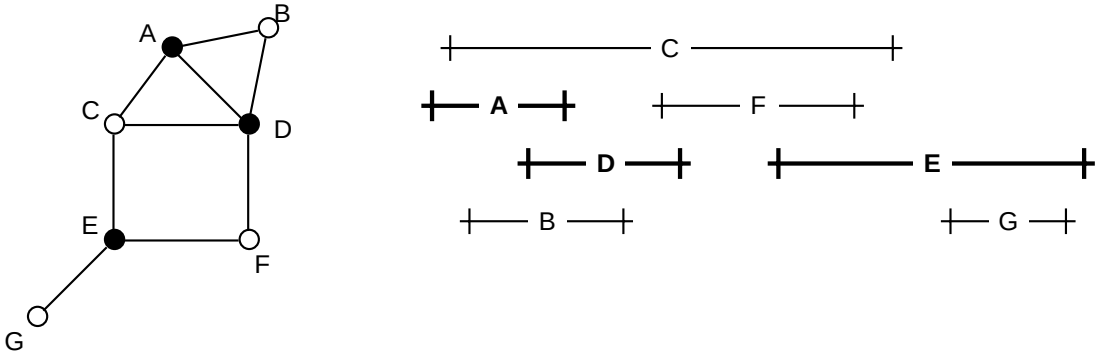

Abbildung 7: Beispiel für einen Referenzintervallgraphen und seine Intervalldarstellung. ($\bullet \in P$ = Referenzen, $\circ \in V \setminus P$ = Nicht-Referenzen)

[22] engl. circular arc graphs
[23] manchmal auch $G(P, N, E)$ mit $N = V \setminus P$ der Menge der Nicht-Referenzen
[24] engl. Probe Graph oder Interval Split Graph

Anmerkung: Wie man leicht nachprüft, ist der in Abbildung 6 dargestellte Referenzgraph kein Referenzintervallgraph, da keine lineare Intervalldarstellung für ihn existiert.

Definition 10 [28] *Sei $G(V, P, E)$ ein Referenzintervallgraph, bei dem für je zwei Nicht-Referenzen v_i und v_j, die von zwei einander nicht überlappenden Referenzen überlappt werden, eine zusätzliche Kante (v_i, v_j) hinzugefügt wird. Der so entstandene Graph heißt* ***Erweiterter Referenzintervallgraph****.*

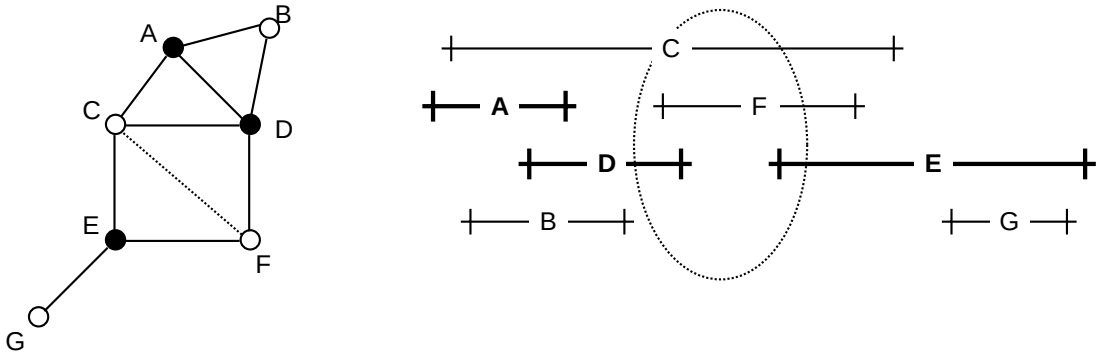

Abbildung 8: Das obige Beispiel als Erweiterter Referenzintervallgraph.

Es ist offensichtlich, daß Intervallgraphen eine spezielle Ausprägung von Referenzintervallgraphen sind. So ist jeder Intervallgraph auch ein Referenzintervallgraph mit $P = V$, wobei die Umkehrung nicht gilt. Der in folgender Abbildung gezeigte Graph „Domino“ ist bekanntermaßen kein Intervallgraph. Doch bei geeigneter Partitionierung sieht man leicht, daß er ein Referenzintervallraph ist. [10] kann dazu eine ausführliche Betrachtung der hierarchischen Ordnung von Intervallgraphen, Referenzintervallgraphen und sogenannten Toleranzgraphen[25] entnommen werden.

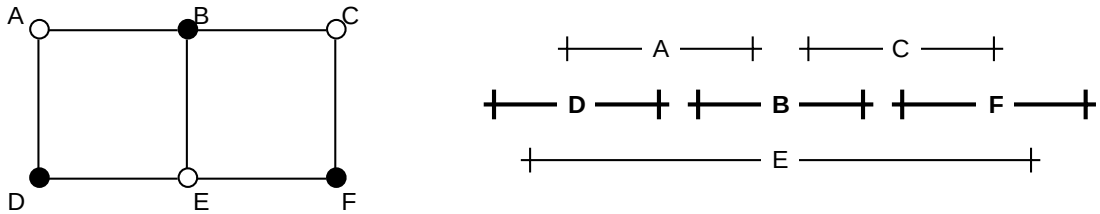

Abbildung 9: Der „Domino“ als Referenzintervallgraph.

[25] engl. tolerance graphs, eingeführt von Golumbic und Monma in „A generalization of interval graphs with tolerances“, Congressus Numeratium 35 (1982), 321-331.

Die offensichtliche Motivation dieser Graphenklasse liegt in der Möglichkeit der Modellierung partieller Überlappungsinformationen, wie sie durch die Hybridisationsmatrix gegeben sind: wenn eine Referenz von einer Nicht-Referenz überlappt wird, sind die entsprechenden Knoten des zugehörigen Referenzgraphen miteinander durch eine Kante verbunden, sonst nicht. Zu keinem Zeitpunkt wird eine Nicht-Referenz mit einer anderen Nicht-Referenz auf Überschneidung getestet, was das Fehlen sämtlicher Kanten zwischen den Knoten der Nicht-Referenzen erklärt. Es ist aber zu beachten und dies ist auch die entscheidende Neuerung gegenüber den herkömmlichen Intervallgraphen, daß nicht dennoch auch Überlappungen zwischen Nicht-Referenzen vorliegen können. Jeder Hybridisationsmatrix ist damit genau ein entsprechender Referenzgraph zugeordnet, dessen Adjazenzmatrix sie bildet. Das Problem der Kopplungsanalyse findet dabei seine Entsprechung in der Frage, ob ein gegebener Referenzgraph $G(V, P, E)$ ein Referenzintervallgraph ist, also ob er eine passende Intervalldarstellung besitzt oder nicht. Dabei ist man in erster Linie an konstruktiven Antworten interessiert, die eine oder besser noch gleich alle möglichen Intervalldarstellungen liefern, falls sich die Frage bejahen läßt. Denn aus diesen Intervalldarstellungen lassen sich sämtliche Informationen zur relativen Position der Fragmente ablesen, die indirekt durch die Hybridisationsmatrix für die gewählte Referenzmenge gegeben sind. Das Problem der Kopplungsanalyse sieht aus einer graphentheoretischen Perspektive damit so aus:

geg.: `ein Referenzgraph G(V,P,E)`

ges.: `alle möglichen Intervalldarstellungen für G`

Diese Form der Modellierung wurde bereits erfolgreich in der Praxis angewandt [29, 30], zumal es inzwischen auch für die Referenzgraphen effiziente Polynomialzeitalgorithmen gibt, die analog zum PQ-Baum-Algorithmus[26] [4] für Intervallgraphen eine lineare Intervalldarstellung finden, sofern eine existiert [18, 14]. Nach einem kurzen Überblick über einige interessante Eigenschaften der Klasse der Referenzintervallgraphen, sollen die auf sie basierenden Algorithmen zur Lösung des Problems der Kopplungsanalyse in den folgenden Abschnitten kurz vorgestellt werden.

[26] engl. PQ-Tree Algorithm

5 Eigenschaften von Referenzintervallgraphen

Satz 1 [28, 21] Alle erweiterten Referenzintervallgraphen sind chordal.

Definition 11 [28]

(a) *Eine **Quasiclique** C eines Referenzgraphen ist eine Menge von Knoten, deren Referenzen eine Clique bilden und deren Nicht-Referenzen zu jeder Referenz in C benachbart sind.*

(b) *Eine **quasimaximale Clique** ist eine Quasiclique, die wenigstens eine maximale Clique des Graphen G enthält.*

(c) *Eine **vollständige Menge quasimaximaler Cliquen** ist eine Menge quasimaximaler Cliquen, so daß jede maximale Clique des Graphen G in genau einer davon enthalten ist.*

Definition 12 [21] *Eine Familie von Teilgraphen $G_1, G_2, ..., G_n$ ist **aufeinanderfolgend geordnet**, wenn gilt:*

$$u \in V(G_i) \cap V(G_j), i \leq j \Rightarrow \forall\ i \leq k \leq j : u \in V(G_k).$$

Satz 2 [28, 21] Ein Referenzgraph $G(V, P, E)$ ist genau dann ein Referenzintervallgraph, wenn eine zugehörige vollständige Menge quasimaximaler Cliquen existiert, die aufeinanderfolgend geordnet werden kann.

Satz 3 [28, 21] Ist $G(V, P, E)$ ein Referenzintervallgraph, dann ist er schwach trianguliert.

Korollar 1 [21] Da nach [12] schwach triangulierte Graphen perfekt sind, sind Referenzintervallgraphen perfekt.

Definition 13 [21] *Sei $G(V, P, E)$ ein Referenzgraph mit der Referenzmenge $P \subseteq V$ und bezeichne $N(u)$ die Nachbarschaft der Nicht-Referenz u. Dann ist eine Clique C des Graphen G **immanent**[27], wenn eine der folgenden Bedingungen erfüllt wird:*

(a) *C ist eine maximale Clique von $G(P)$; oder*

(b) *C ist eine maximale Clique von $G(P \cap N(u))$ für eine beliebige Nicht-Referenz u.*

[27] engl. intrinsic

Definition 14 [21] *Die immanenten Cliquen eines Referenzgraphen sind* ***vollständig aufeinanderfolgend geordnet***[28]*, wenn sie gemäß Definition 7 aufeinanderfolgend geordnet sind und außerdem für alle maximalen Cliquen* C_i, C_j $(i \leq j)$ *des Graphen* $G(N(u))$ *einer beliebigen Nicht-Referenz u gilt:*

$$C_i \cup C_j \subseteq G(N(u)) \Rightarrow C_k \subseteq G(N(u)), \forall k \text{ mit } i \leq k \leq j$$

Satz 4[21] Ein Referenzgraph $G(V, P, E)$ ist genau dann ein Referenzintervallgraph, wenn seine immanenten Cliquen vollständig aufeinanderfolgend geordnet werden können.

6 Algorithmen für die Kopplungsanalyse

6.1 Ein erster einfacher Algorithmus

Basierend auf Satz 4 ist es nun nicht kompliziert, einen ersten heuristischen Erkennungsalgorithmus anzugeben. Dazu wird der Satz noch einmal umformuliert:

Definition 15 [21] *Eine 0-1-Matrix hat die* ***Spalteneigenschaft, daß alle Einsen aneinandergereiht sind***[29]*, wenn die Spalten derart permutiert werden können, daß alle von Null verschiedenen Einträge einer Zeile hintereinander erscheinen.*

Definition 16 [21] *Die* ***Knoten-Cliquen-Matrix*** $M = [m_{ij}]$ *eines Graphen G ist eine 0-1-Matrix, wobei die Spalten den maximalen Cliquen von G und die Zeilen den Knoten* $v \in V$ *entsprechen. Der Eintrag* $m_{ij} = 1$*, wenn der i-te Knoten in der j-ten Clique enthalten ist, sonst ist* $m_{ij} = 0$*.*

Aus Definition 12 folgt damit offensichtlich, daß genau dann eine aufeinanderfolgende Ordnung der maximalen Cliquen von $G(V, E)$ existiert, wenn die Knoten-Cliquen-Matrix die erwähnte Spalteneigenschaft besitzt.

Definition 17 [21] *Eine* ***gemischte 0-1-Matrix*** $M = [m_{ij}]$ *besteht aus den Einträgen 0,1 oder *, wobei * als „unbestimmt“*[30] *interpretiert wird. Eine gemischte 0-1-Matrix hat genau dann die Spalteneigenschaft, daß alle Einsen aneinandergereiht sind, wenn eine Permutation der Spalten existiert, daß bei geeigneter Belegung der unbestimmten Einträge mit den Werten 0 und 1 alle von Null verschiedenen Werte einer Zeile hintereinander auftreten.*

[28] engl. pan-consecutively ordered
[29] engl. consecutive 1‘s property for columns
[30] engl. indeterminate

Definition 18 [21] *Die **immanente Matrix** $S = [s_{ij}]$ eines Referenzgraphen $G(V, P, E)$ sei eine gemischte 0-1-Matrix deren i-te Zeile mit dem i-ten Knoten und deren j-te Spalte mit der j-ten immanenten Clique des Graphen G korrespondieren. Für den Fall, daß der i-te Knoten u_i eine Referenz ist, gilt:*

- $s_{ij} = 1$, *wenn* $u_i \in C_j$
- $s_{ij} = 0$, *wenn* $u_i \notin C_j$

Im anderen Fall, daß der Knoten u_i keine Referenz ist, gilt:

- $s_{ij} = 1$, *wenn C_j eine maximale Clique von $G(N(u_i))$ ist*
- $s_{ij} = *$, *wenn $C_j \subseteq G(N(u_i))$ aber keine maximale Clique von $G(N(u_i))$ ist*
- $s_{ij} = 0$, *wenn $C_j \nsubseteq G(N(u_i))$*

Mit diesen Hilfsmitteln läßt sich der Satz 4 nun unmittelbar wie folgt formulieren:

Satz 5 [21] Ein Referenzgraph $G(V, P, E)$ ist genau dann ein Referenzintervallgraph, wenn seine immanente Matrix die Spalteneigenschaft besitzt, daß alle Einsen aneinandergereiht sind.

Korollar 2 [21] Sei $G(V, P, E)$ ein Referenzgraph mit den immanenten Cliquen $C_1, ..., C_m$. Wenn nun für alle Nicht-Referenzen $u \in V \setminus P$ gilt, daß jede immanente Clique $C_i \subseteq G(N(u))$ eine maximale Clique in $G(N(u))$ ist, dann existiert ein Polynomzeitalgorithmus, der bestimmt, ob G ein Referenzintervallgraph ist oder nicht.

Der Beweis hierfür ist einsichtig: da alle immanenten Cliquen maximale Cliquen in $G(N(u))$ sind, enthält die immanente Matrix keine unbestimmten Einträge und ist damit eine echte 0-1-Matrix. Während das Problem des Testens einer gemischten 0-1-Matrix auf die Spalteneigenschaft, daß alle Einsen aneinandergereiht sind, als NP-vollständig bekannt ist [11, 15], kann für echte 0-1-Matrizen der bereits erwähnte PQ-Baum-Algorithmus eingesetzt werden, der sich durch lineare Laufzeit auszeichnet [4]. Um jedoch auch für den Fall, daß nicht alle immanenten Cliquen gleichzeitig maximale Cliquen sind, eine praktikable Zeitschranke zu erhalten, nutzt man die Beobachtung, daß Referenzen einander in der Praxis nur sehr selten komplett überlappen, um das Problem einzuschränken. Dies kann im Zweifelsfall immer durch geeignete Vorauswahl der Referenzen erreicht werden. Der auf dieser Beobachtung basierende heuristische Algorithmus zur Erkennung von Referenzintervallgraphen in dieser eingeschränkten Menge von Referenzgraphen kann der 1998 erschienenen Arbeit des Bioinformatikers Zhang entnommen werden [21]. Auf eine Darstellung desselben soll an dieser Stelle verzichtet werden, da es inzwischen weit elegantere Polynomzeitalgorithmen gibt, die ohne Einschränkungen auf allen Referenzgraphen arbeiten.

6.2 Ein quadratischer Lösungsalgorithmus

2001 veröffentlichten die Graphentheoretiker Johnson und Spinrad erstmals solch einen allgemeinen Polynomzeitalgorithmus, der das Erkennungsproblem von Referenzintervallgraphen in $O(n^2)$ Zeitschritten in Abhängigkeit von der Größe des zu testenden Referenzgraphen löste [14]. Sie benutzen als grundlegende Datenstruktur ebenfalls die besagten PQ-Bäume:

Definition 19 [4] *Die Klasse der* ***PQ-Bäume*** *über der Menge* $U = \{a_1, a_2, ..., a_m\}$ *ist folgendermaßen definiert:*

1. *Jedes Element* $a_i \in U$ *ist für sich ein PQ-Baum, dessen Wurzel und gleichzeitig einziges Blatt es ist.*

2. *Sind* $T_1, T_2, ..., T_k$ *alles PQ-Bäume, dann ist der neue Baum, der aus einem sogenannten* ***P-Knoten*** *als Wurzel und den PQ-Bäumen* $T_1, T_2, ..., T_k$ *als dessen Kinder entsteht, ebenfalls ein PQ-Baum. (P-Knoten üblicherweise als Kreise dargestellt.)*

3. *Sind* $T_1, T_2, ..., T_k$ *alles PQ-Bäume, dann ist der neue Baum, der aus einem sogenannten* ***Q-Knoten*** *als Wurzel und den PQ-Bäumen* $T_1, T_2, ..., T_k$ *als dessen Zweige entsteht, ebenfalls ein PQ-Baum. (Q-Knoten werden im Allgemeinen als Rechtecke dargestellt.)*

Abbildung 10: Darstellung von P- und Q-Knoten

Während die Kinder eines P-Knotens ungeordnet sind und in beliebiger Reihenfolge stehen können, ist die Reihenfolge bei den Q-Knoten festgelegt. Lediglich die Orientierung, also ob die Knoten von vorne nach hinten oder umgekehrt aufgezählt werden, ist bei einem Q-Knoten noch frei wählbar. Diese Eigenschaften machen PQ-Bäume zu einer idealen Datenstruktur für Permutationen der Menge U und in der Tat wurden sie just zu diesem Zweck entwickelt. Ein einzelner PQ-Baum kann die gesamte Menge aller denkbaren Permutationen von U (ein

einzelner P-Knoten mit allen Elementen $a_i \in U$ als Blätter), eine einzige ausgewählte Permutation (ein einzelner Q-Knoten mit allen Elementen aus U als Blätter) oder jede beliebige Form der Ordnung zwischen diesen beiden Extremen darstellen.

In dem Ausgangs-PQ-Baum, mit dem der generelle PQ-Baum-Algorithmus beginnt, werden nun nach und nach durch geschickte Manipulation zusätzliche Nebenbedingungen eingearbeitet, die die Zahl der durch ihn dargestellten Permutationen verringern. Entweder bricht der Algorithmus irgendwann ab, weil eine hinzuzufügende Nebenbedingung unvereinbar mit dem bisher berechneten Baum ist, oder aber der Algorithmus liefert einen PQ-Baum, der alle Permutationen darstellt, die den gestellten Bedingungen genügen. Dadurch findet der Algorithmus z.B. eine Permutation der Spalten einer 0-1-Matrix, so daß die Einsen in allen Zeilen hintereinander angeordnet sind. Hier durchläuft der Algorithmus Zeile für Zeile der Matrix und beschränkt die möglichen Permutationen der Spalten derart, daß nun nur noch solche auftreten, in denen wirklich alle Einsen der gerade bearbeiteten Zeile aufeinanderfolgen. Würde solch eine Beschränkung einer bereits getroffenen Reduktion widersprechen, ist die Lösungsmenge offensichtlich leer und der Algorithmus bricht ab. Die Manipulation des PQ-Baums ist in aller Ausführlichkeit in [4] beschrieben.

Bevor nun der Algorithmus von Johnson und Spinrad dargestellt wird, folgen einige notwendige Begriffsklärungen:

Definition 20 [6] *Jede Menge $M \subseteq V$ eines Graphen G(V,E) heißt genau dann **Modul**, wenn jeder Knoten $v \notin M$ entweder mit keinem oder mit allen Knoten in M verbunden ist. Da jeder Graph die Module $\emptyset, \{v\}$ mit $v \in V$ und V selbst enthält, werden diese als **triviale Module** bezeichnet. Nicht triviale Module nennt man auch **homogene Mengen**.*

Definition 21 [6] *Ein Graph $G(V, E)$ kann durch folgendes Verfahren der **modularen Dekomposition** zerlegt werden:*

- *Ist G unzusammenhängend, wird G durch einen **parallelen Dekompositionsschritt**[31] in seine Zusammenhangskomponenten zerlegt.*
- *Ist $\overline{G}$, das Komplement von G, unzusammenhängend, wird G durch einen **seriellen Dekompositionsschritt** in die Zusammenhangskomponenten von $\overline{G}$ zerlegt.*
- *Sind G und $\overline{G}$ zusammenhängend, wird G durch einen **primen Dekompositionsschritt** in seine maximalen Module zerlegt.*

[31] Diese Arbeit folgt hier der Bezeichnung von Johnson&Spinrad.

Mit den durch die Zerlegung entstandenen Teilgraphen wird analog verfahren, bis der komplette Graph in seine einzelnen Knoten zerlegt wurde. Der auf natürliche Weise durch dieses Vorgehen entstehende Baum heißt ***modularer Dekompositionsbaum****.*

Zur Realisierung des modularen Dekompositionsbaums eines Graphen existieren die verschiedensten Polynomialzeit- und Linearzeitalgorithmen [7, 19]. Angenommen, solch ein Dekompositionsbaum des induzierten Teilgraphen, den die Referenzen P des zu testenden Referenzgraphen $G(V, P, E)$ bilden, wurde bereits mit Hilfe eines Linearzeitalgorithmus bestimmt. Dann konstruiert der Algorithmus von Johnson und Spinrad zuerst einen Start-PQ-Baum aus diesem modularen Dekompositionsbaum. Dabei wird jeder Knoten $p \in P$, der im Dekompositionsbaum noch als ein einzelnes Blatt codiert war, nun durch 2 Blätter symbolisiert - p_L als der linke Rand des dem Knoten zugeordneten Intervalls (Startmarke), p_R als der rechte Rand (Endmarke). In der Praxis sieht das auf der untersten Ebene des Baumes dann folgendermaßen aus:

- Ein serieller Dekompositionsknoten, der nur Blätter als Kinder hat, repräsentiert eine Clique. Solch eine Clique kann nur als Folge übereinander geschichteter Intervalle dargestellt werden. Die Reihenfolge der Endpunkte ist dabei beliebig.

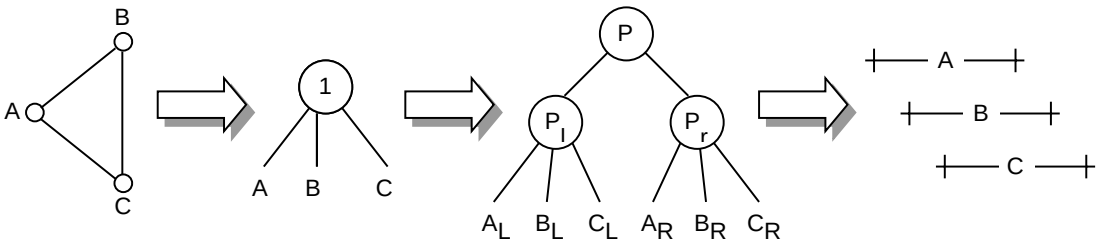

Abbildung 11: Eine Clique, ihre Repräsentation im modularen Dekompositionsbaum, dessen Entsprechung im PQ-Baum und eine daraus resultierende mögliche Anordnung der zugehörigen Intervalle. (von links nach rechts)

- Ein paralleler Dekompositionsknoten, der nur Blätter als Kinder hat, repräsentiert eine unabhängige Menge. Eine solche Menge kann nur als Folge einander nicht überlappender Intervalle dargestellt werden, deren Reihenfolge nach Belieben variiert werden kann.

- Ein primer Dekompositionsknoten, der nur Blätter als Kinder hat, repräsentiert ein nicht-triviales Modul M. Die Menge der maximalen Cliquen von M kann nun derart geordnet werden, so daß für jeden Knoten $v \in M$

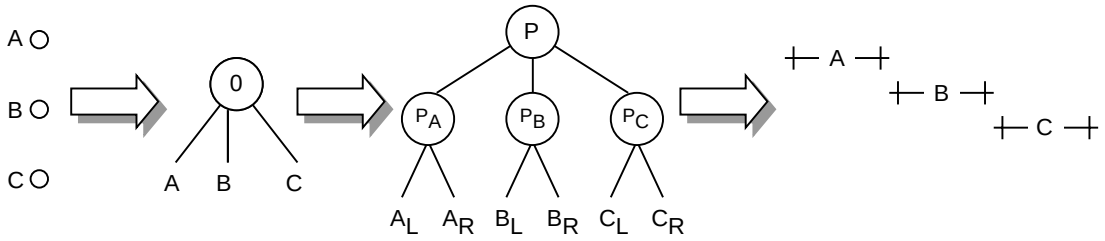

Abbildung 12: Eine unabhängige Menge, ihre Repräsentation im modularen Dekompositionsbaum, dessen Entsprechung im PQ-Baum und eine daraus resultierende mögliche Anordnung der zugehörigen Intervalle. (von links nach rechts)

die Cliquen, in denen er vorkommt, hintereinander stehen. Dies geschieht wiederum in Linearzeit mittels des PQ-Baum-Algorithmus'. Die folgende Abbildung zeigt ein Beispiel für solch ein nicht-triviales Modul mit den maximalen Cliquen $C_1 = \{A, C, F\},\ C_2 = \{C, F, B\},\ C_3 = \{F, B, D\}, C_4 = \{D, E\}$, wobei die Indizes der Cliquen bereits die erwähnte Ordnung wiederspiegeln. Zusätzlich wird jetzt jeder Clique C_i eine Menge S_i zugeordnet, die genau die Startmarken jener Knoten enthält, die erstmals in dieser Clique auftauchen, also nicht Teil einer der vorangegangenen Cliquen sind. So enthält z.B. $S_1 = \{A_L, C_L, F_L\}$. Analog wird die Menge F_i definiert, welche genau die Endmarken derjenigen Knoten aufnimmt, die zum letzten Mal in der Clique C_i auftauchen und damit in keiner der nachfolgenden Cliquen mehr vorkommen. Somit gilt im gezeigten Beispiel: $F_4 = \{D_R, E_R\}$. Alle nichtleeren Mengen S und F werden nun als P-Knoten in der Reihenfolge der Indizes an einen Q-Knoten gehängt, um die PQ-Baum-Repräsentation des primen Dekompositionsknotens zu erhalten.

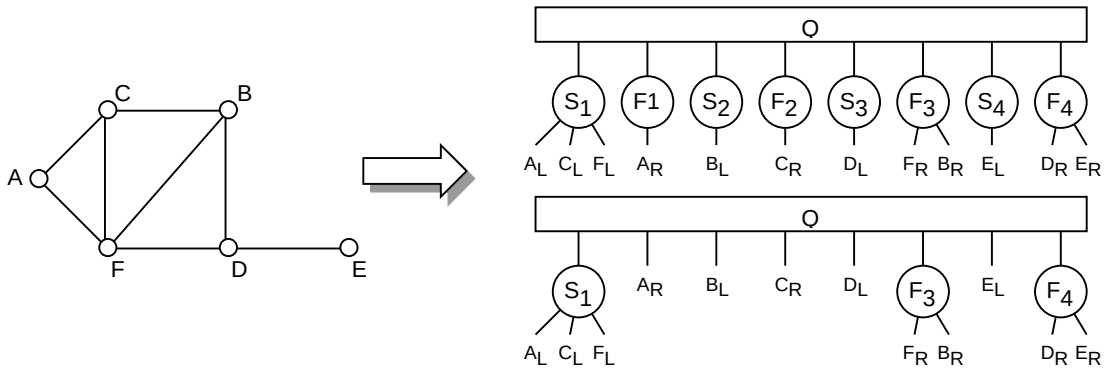

Abbildung 13: Ein nicht-triviales Modul und seine Repräsentation im PQ-Baum. (oben: normal, unten: vereinfacht, da P-Knoten mit einem Blatt sinnlos sind)

Wurde auf diese Art und Weise erstmal die unterste Ebene des modularen Dekompositionsbaums in PQ-Bäume umgewandelt, arbeitet sich der Algorithmus Ebene für Ebene bis zur Wurzel hoch und integriert nach bestimmten Vorschriften die Kinder wiederum zu einem PQ-Baum. Die dabei durchzuführenden Baummanipulationen reichen von einfachen Substitutionen bis hin zu komplexen Integrationsverfahren und sollen als technische Details hier nicht von Interesse sein. Auch in [14], der Originalveröffentlichung zu diesem Algorithmus, werden lediglich zwei der insgesamt sieben möglichen Vater-Kind-Knotenbeziehungen eines modularen Dekompositionsbaums mit ihren Integrationsalgorithmen betrachtet. Nach Abschluß des Integrationsprozeß' hat man also erstmal den Start-PQ-Baum, der alle möglichen Intervalldarstellungen der Referenzen codiert. Nun folgt der eigentliche Algorithmus, der sukzessive alle Nicht-Referenzen betrachtet und deren Nachbarschaftsbeziehungen zu den Referenzen als Nebenbedingungen in die Struktur des Baumes mit einfließen läßt; also wieder genau nach dem bereits beschriebenen Grundmuster eines jeden PQ-Baum-Algorithmus' arbeitet. Zwei Knoten sind benachbart, wenn eine Kante $e \in E$ existiert, die diese beiden Knoten verbindet. Für jede Nicht-Referenz $v \in V \setminus P$ eines Referenzgraphen $G(V, P, E)$ werden nun also alle Knoten des PQ-Baumes entsprechend ihrer Nachbarschaftsbeziehung zu v markiert. Die Blätter des Baumes mit: (N), falls Nachbarschaft zu v vorliegt - (O), falls dies nicht der Fall ist. Die inneren Knoten des Baumes mit: (N), falls alle Kinder ebenfalls mit (N) markiert wurden - (O), falls alle Kinder mit (O) markiert wurden - (S) sonst. Wurde die Markierung von den Blättern bis zur Wurzel komplett vorgenommen, werden die durch den Baum repräsentierten Permutationen nun derart eingeschränkt, daß Knoten, die zu v benachbart sind, hintereinander auftreten. Ein Beispiel:

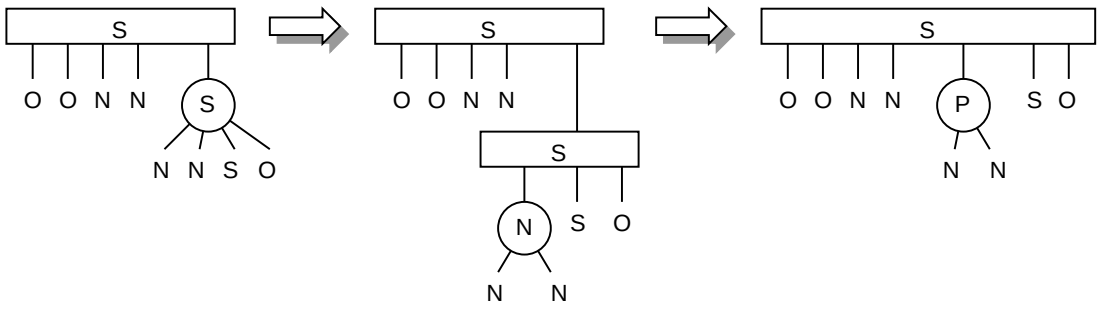

Abbildung 14: Ein Beispiel für die Beschränkung der möglichen Permutationen durch Einfügen der Nachbarschaftsinformationen

Man sieht, daß die Kinder des P-Knotens nach Einfügen der Nachbarschaftsmarken so angeordnet wurden, daß nun alle benachbarten Knoten nebeneinander stehen. Damit diese Festlegung auch erhalten bleibt und nicht von der nächsten Nicht-Referenz wieder rückgängig gemacht wird, ändert der Algorithmus an dieser Stelle die Struktur des Baumes. Zuerst wird durch Einführung eines neuen

Q-Knotens die gefundene Reihenfolge gesichert, denn ein Q-Knoten kann ja nicht mehr frei permutiert werden. Dann wird dieser Q-Knoten noch in den übergeordneten Q-Knoten integriert, damit auch eine Umkehrung der Reihenfolge, also ein Lesen von Rechts nach Links, ausgeschlossen wird. So gibt es für alle möglichen Fälle, die beim Durchlaufen der Nicht-Referenzen auftreten können, detaillierte Manipulationsanweisungen, die den PQ-Baum und damit auch die Anzahl der möglichen Permutationen beschränken. Lassen sich die zu einer Nicht-Referenz benachbarten Knoten des Baumes einmal nicht hintereinander anordnen, dann ist der betrachtete Referenzgraph kein Referenzintervallgraph und der Algorithmus bricht ab. Nachdem nun alle Nicht-Referenzen bearbeitet wurden, beginnt der Algorithmus wieder von vorn - solange, bis keine der Nicht-Referenzen mehr eine Änderung des PQ-Baumes hervorruft. Die Zeitkomplexität des gesamten Algorithmus' wird mit $O(n^3)$ angegeben, wobei man aber durch geschicktes Sortieren der Nicht-Referenzen erreichen kann, daß jede nur genau einmal betrachtet werden muß und die Zeitschranke dadurch im schlimmsten Fall quadratisch wird.

6.3 Ein Linearzeitalgorithmus

Der folgende Abschnitt geht o.B.d.A. davon aus, daß in der Intervalldarstellung an keiner Stelle zwei oder mehr Intervalle in ein und demselben Punkt enden. Da der zugrunde liegende Graph endlich ist, kann dies vermieden werden, indem die betroffenen Endpunkte geringfügig verschoben werden.

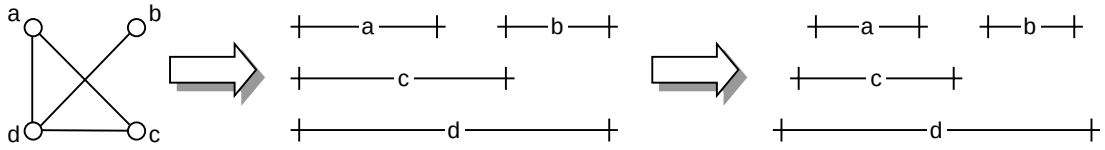

Abbildung 15: Zwei Intervalldarstellungen desselben Intervallgraphen, rechts die gewünschte ohne direkt aneinandergrenzende Intervallenden

Diese Einschränkung ermöglicht eine eindeutige Codierung der Intervalldarstellung als Zeichenkette, indem für jeden Endpunkt eines Intervalls sein Bezeichner an die entsprechende Stelle gesetzt wird. Dadurch erhält man eine Codierung, in der jeder Intervallbezeichner genau zweimal auftritt – je einmal für den linken und rechten Endpunkt. Die Intervalldarstellung des obigen Beispiels würde demnach (bis auf Umkehrung) eindeutig als DCAACBBD codiert werden. Da sich aus solch einer Darstellung als Zeichenkette die zugehörige Intervalldarstellung ohne weiteres wieder rekonstruieren läßt, kommt es im folgenden Abschnitt nur noch darauf an, eine passende Zeichenkette[32] zu einem gegebenen Referenzgraphen zu finden.

[32]engl. realizer

Die folgenden Betrachtungen gehen davon aus, daß bereits eine gültige Intervalldarstellung vorliegt und die nachfolgenden Definitionen beziehen sich der Einfachheit halber erstmal lediglich auf Intervallgraphen und werden dann anschließend für Referenzintervallgraphen verallgemeinert:

Definition 22 [18] *Die Kantenmenge E eines Intervallgraphen $G(V, E)$ symbolisiert Überlappungen in der Intervalldarstellung. Wir unterscheiden dabei zwischen der* ***teilweisen Überlappung*** *und der* ***vollständigen Überlappung****, bei der ein Intervall komplett in einem anderen enthalten ist. Die Kantenmenge E kann offensichtlich in die zwei disjunkten Teilmengen E_T und E_1 partitioniert werden, wobei die Menge E_1 genau diejenigen Kanten enthält, die eine vollständige Überlappung und E_T demnach die Kanten die eine teilweise Überlappung darstellen. $\overline{E}$, die Kantenmenge des Komplementgraphen $\overline{G}(V, \overline{E})$ enthält alle Nicht-Kanten von G und wird daher nachfolgend auch als E_0 bezeichnet. Die durch E_T, E_1 und E_0 induzierten Graphen werden analog mit G_T, G_1 und G_0 benannt, wobei mehrere Indizes die Vereinigung der angegebenen Graphen bedeuten. So ist z.B. $G_{1T0} = G(V, E_1 \cup E_T \cup E_0)$ der vollständige Graph $K_{|V|}$.*

Satz 6 Für jeden Intervallgraphen G sind G_1, G_0, G_{T0} und G_{1T0} transitiv orientierbar.

Beweis:

- G_1 ist Permutationsgraph (siehe z.B. [24]) $\rightleftarrows$ G_1 und $\overline{G_1} = G_{T0}$ sind Vergleichbarkeitsgraphen.
- G_0 ist das Komplement eines Intervallgraphen und damit ebenfalls ein Vergleichbarkeitsgraph [17].
- Cliquen wie G_{1T0} sind Permutationsgraphen.

Definition 23 [18] *Die Kantenmenge E eines Intervallgraphen $G(V, E)$ kann folgendermaßen in eine gerichtete Kantenmenge A, eine sogenannte* ***Intervallorientierung****, transformiert werden:*

$A = \{(a, b) : ab \in E$ *und das rechte Intervallende von a kommt vor dem von b*$\}$

Die Umkehrung aller Orientierungen in A wird durch A^T angezeigt. Entsprechend der Teilmengen E_1, E_T und E_0 sind dadurch auch die Mengen A_1, A_T und A_0 definiert. Die zugehörigen ***gerichteten Graphen*** *werden mit D_1, D_T und D_0 bezeichnet, wobei natürlich auch hierbei wieder mehrere Indizes möglich sind, um eine Vereinigung von zwei oder drei dieser Teilmengengraphen zu kennzeichnen.*

Beobachtung 1 [18] A_1, A_T, A_{T0} und A_{1T0} sind transitive Orientierungen der Graphen G_1, G_T, G_{T0} und G_{1T0}. Denn wenn z.B. Intervall a vor b endet und Intervall b vor c, dann muß auch Intervall a vor c enden.

Beobachtung 2(a) [18] Die eindeutige Knotenfolge, die die Vereinigung $A_{1T0} = A_1 \cup A_T \cup A_0$ liefert, entspricht genau der Reihenfolge $r(v_1), r(v_2), ..., r(v_n)$ der rechten Intervallendpunkte.

Beobachtung 2(b) [18] Die eindeutige Knotenfolge, die die Vereinigung $(A_1)^T \cup A_T \cup A_0$ liefert, entspricht genau der Reihenfolge $l(v_1), l(v_2), ..., l(v_n)$ der linken Intervallendpunkte.

Beobachtung 3 [17] Die Zeichenkettencodierung einer Intervalldarstellung läßt sich wie folgt aus den beiden Einzelsequenzen der linken und rechten Endpunkte ermitteln: Man nimmt die Teilsequenz der linken Intervallenden $l(v_1), l(v_2), ..., l(v_n)$ und fügt einzeln die rechten Intervallenden der zweiten Sequenz $r(v_1), r(v_2), ..., r(v_n)$ in ihrer gegebenen Reihenfolge ein. Die jeweils richtige Stelle für den rechten Endpunkt eines Intervalls v_i in der linken Einzelsequenz ist die erste Position von links, für die folgende Bedingungen erfüllt sind:

- sie befindet sich rechts vom zuvor eingefügten Intervallende $r(v_{i-1})$
- sie befindet sich rechts von allen linken Intervallenden, der Nachbarn von v_i

Nachfolgend ist zur Illustration der erläuterten Sachverhalte ein Beispiel angegeben:

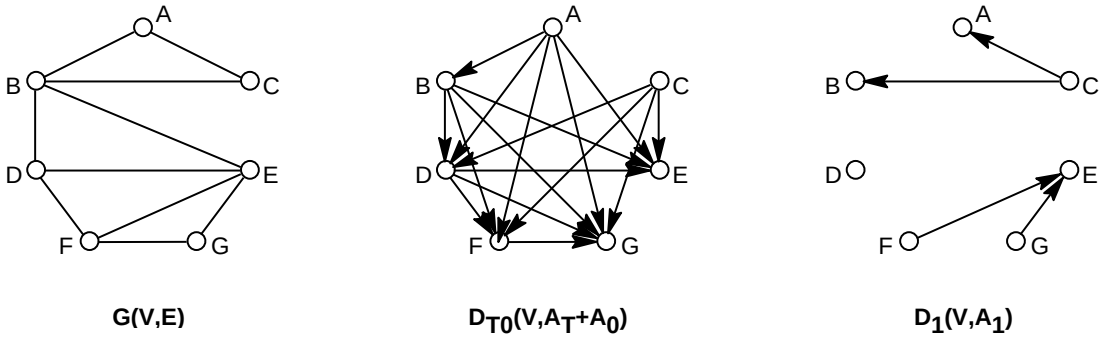

Abbildung 16: Transitive Orientierungen der Graphen G_{T0} und G_1 zu einer gegebenen Intervalldarstellung

Der Prozess des topologischen Sortierens der Graphen $D_{1T0} = G(V, A_1 \cup A_T \cup A_0)$ und $D_{1^TT0} = G(V, (A_1)^T \cup A_T \cup A_0)$ liefert die beiden Knotenfolgen CABDFGE und ABCDEFG, wobei die erste gemäß Beobachtung 2(a) genau die Reihenfolge der rechten Intervallenden angibt und die zweite nach 2(b) genau die der linken. Um die gesamte Zeichenkettencodierung zu erhalten, müssen die beiden Zeichenfolgen noch wie in Beobachtung 3 beschrieben miteinander verknüpft werden:

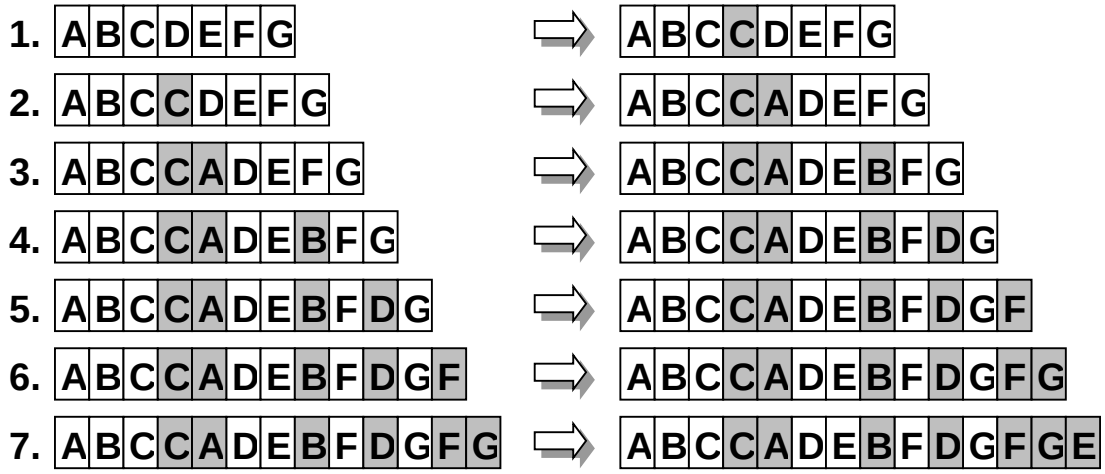

Abbildung 17: Die Verknüpfung der linken Teilfolge ABCDEFG mit der rechten Teilfolge CABDFGE liefert: ABCCADEBFDGFGE

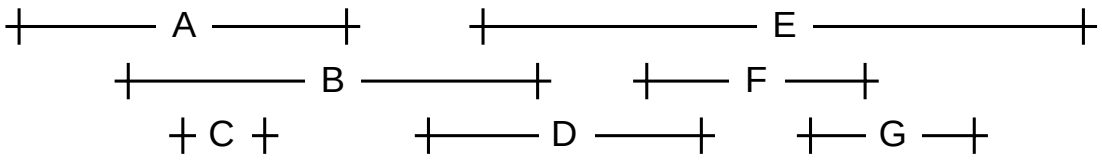

Abbildung 18: Die zu Abb.17 korrespondierende Intervalldarstellung

Da bisher die Intervalldarstellung als vorhanden betrachtet wurde, war es kein Problem, die zugehörige Intervallorientierung für G_{T0} und G_1 zu finden und im Nachhinein daraus wieder die Originalsequenz zu ermitteln. Nun ist im Allgemeinen die Intervalldarstellung ja aber noch unbekannt und soll erst durch den Algorithmus gefunden werden. Dazu wird ein rückwärtiges Vorgehen genutzt: man versucht zuerst, die Mengen E_1 und E_{T0} zu bestimmen. Dafür vereinfacht man das Problem schrittweise, indem man:

1. alle isolierten Knoten löscht. Diese können ganz am Ende des Algorithmus' als disjunkte Intervalle einfach an die errechnete Intervalldarstellung angehängt werden.

2. alle universellen Knoten des übriggebliebenen Restgraphen löscht. Diese können ebenfalls zum Ende als große, alles überlappende Intervalle wieder in die gefundene Intervalldarstellung eingefügt werden.

3. alle Module, die Cliquen sind, zu einem einzigen „Platzhalter-Knoten“ zusammenzieht. Clique-Module können in linearer Zeit durch lexikalisches Sortieren der Adjazenzliste gefunden werden [17]. Zum Schluß können diese Clique-Module durch Substitution von einander überlappenden Intervallen an Stelle des „Platzhalter-Intervalls“ wieder in die Intervalldarstellung integriert werden. Bestehende Nachbarschaftsbeziehungen (Überlappungen) mit angrenzenden Intervallen sind dabei auf alle der eingefügten Intervalle zu übertragen.

Dieser reduzierte Graph wird nachfolgend mit G' bezeichnet. Durch die angegebenen Vereinfachungen gibt es keine zwei benachbarten Knoten mehr, deren abgeschlossene Nachbarschaften identisch zueinander sind. Daraus folgt:

Satz 7 [18] Seien p und q zwei Knoten des reduzierten Graphen $G'(V', E')$ und bezeichne $N[x] = \{N(x) \cup x\}$ mit $N(x) = \{y \in V' | xy \in E'\}$ die abgeschlossene Nachbarschaft. So gilt $N[p] \subsetneq N[q]$ genau dann, wenn das dem Knoten p zugeordnete Intervall komplett von dem Intervall des Knotens q überlappt wird.

Durch Betrachten der abgeschlossenen Nachbarschaften der Knoten eines reduzierten Graphen lassen sich also bereits vorab folgende Eigenschaften der gesuchten Intervalldarstellung ermitteln:

- ob ein Intervall total in einem anderen enthalten ist, also die zugehörige Kante der Menge E_1 zugeordnet werden muß;

- welches der beiden jeweils betrachteten Intervalle in dem anderen enthalten ist, also die Orientierung der zugehörigen Kante in A_1 und damit auch in G_1.

Es bleibt also, eine Intervallorientierung für G_{T0} zu finden. Dies ist komplizierter, als es auf den ersten Blick aussieht, da die üblichen Algorithmen zur Bestimmung transitiver Orientierungen nicht in jedem Fall eine Intervallorientierung gemäß Definition 23 liefern. An dieser Stelle kommen besonders angepaßte Verfahren der modularen Dekomposition zum Einsatz, die aber nicht Gegenstand dieser Arbeit sind.

Zu guter Letzt soll jetzt noch auf die Änderungen eingegangen werden, die nötig sind, wenn der vorliegende Graph kein Intervallgraph, sondern ein Referenzintervallgraph ist [18]:

- Die Menge E_0 enthält jetzt nur noch diejenigen Nicht-Kanten, die wenigstens eine Referenz als Endpunkt besitzen. Das ist logisch, da eine Nicht-Kante zwischen zwei Nicht-Referenzen bekanntlich nichts über deren Verhältnis zueinander besagt, da ja Nicht-Referenzen nicht gegenseitig auf Überlappung getestet werden.

- Bei der Erzeugung des reduzierten Graphen G' ist folgendes zu beachten: der „Platzhalter-Knoten“, der ein Cliquenmodul ersetzt ist eine Referenz und alle Module aus der Menge der Nicht-Referenzen werden ebenfalls zu einem „Platzhalter-Knoten“ zusammengezogen, der als Nicht-Referenz eingefügt wird.

- Beim darauffolgenden Test auf Zugehörigkeit zu E_1 müssen jetzt die folgenden zwei Fälle unterscheiden werden:

  1. Seien p und q zwei Referenzen und pq eine Kante des reduzierten Graphen $G'(V', P', E')$. So gilt $N[p] \subsetneq N[q]$ genau dann, wenn das dem Knoten p zugeordnete Intervall komplett von dem Intervall des Knotens q überlappt wird.
  2. Seien p und q zwei Knoten (davon eine Referenz und eine Nicht-Referenz) und pq eine Kante des reduzierten Graphen $G'(V', P', E')$. So gilt $N[p] \cap P' \subsetneq N[q] \cap P'$ genau dann, wenn das dem Knoten p zugeordnete Intervall komplett von dem Intervall des Knotens q überlappt wird.

- Bei Referenzintervallgraphen muß bei der Suche nach einer transitiven Orientierung nicht nur darauf geachtet werden, daß diese eine zulässige Intervallorientierung ist, sondern sie muß noch zusätzlichen Eigenschaften genügen. Diese können der Originalpublikation [18] entnommen werden und führen zu dem Begriff der **erweiterbaren Intervallorientierung**[33].

[33] engl. extensible orientation

7 Fazit und offene Fragen

Aus den vorangegangenen Absätzen ist deutlich geworden, daß der direkte Praxisbezug der Klasse der Referenzintervallgraphen zu einer raschen Entwicklung leistungsfähiger Algorithmen geführt hat. z.T. konnten bereits bestehende Techniken wie der PQ-Baum-Algorithmus, modulare Dekomposition oder transitive Orientierungen sehr erfolgreich an die Referenzintervallgraphen angepaßt werden. Dabei bleiben zwei Fragen weiterhin offen [21]:

- Wie hoch ist die algorithmische Komplexität des Testens eines beliebigen Graphen – also ohne eine vorgegebene Partitionierung in Referenzen und Nicht-Referenzen – darauf, ob er ein Referenzintervallgraph ist?

- Kann die Klasse der Referenzintervallgraphen durch eine Menge verbotener Teilgraphen charakterisiert werden? Und wenn ja, durch welche?

Literatur

[1] Applied Biosystems, Store Catalog, ABI PRISM®3100 Genetic Analyzer - Product Information;
http://www.appliedbiosystems.com/catalog/myab/StoreCatalog/products/CategoryDetails.jsp?hierarchyID=101&category3rd=111905&trail=no

[2] O. Avery, The Oswald T. Avery Papers, 1909-1998, Modern Manuscripts Collection, History of Medicine Division, National Library of Medicine, Bethesda, Maryland, USA;
digitalisiert verfügbar unter http://profiles.nlm.nih.gov/CC/

[3] S. Benzer, On the Topology of the Fine Structure, Proceedings of the National Academy of Sciences U.S.A., Band 45 (1959), 1607-1620

[4] K. Booth, G. Lueker, Testing for the Consecutive Ones Properts, Interval Graphs, and Graph Planarity using PQ-Tree Algorithms, Journal of Computer and System Sciences 13 (1976), 335-379

[5] Andreas Brandstädt, Graphen und Algorithmen, Teubner, 1994

[6] A. Brandstädt, V.B. Le, Effiziente Graphenalgorithmen II (Graphendekomposition), Skript zueiner Vorlesung im WS 2000/2001, Fachbereich Informatik, Universität Rostock

[7] A. Cournier, M. Habib, A new linear algorithm for modular decomposition, LIRMM 1995, University Montpellier

[8] R. Diestel, Graphentheorie, 2. Auflage, Springer, 2000

[9] M. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, 1980

[10] M. Golumbic, M. Lipshteyn, On the Hierarchy of Interval, Probe and Tolerance Graphs, Congressus Numeratium 153 (2001), 97-106

[11] M. Golumbic, A. Wassermann, Complexity and algorithms for graph and hypergraph sandwich problems, Technical Report 96,03, Bar-Ilan University, Ramat-Gan, Israel

[12] R. Hayward, Weakly Triangulated Graphs, Journal of Combinatorial Theory, Series B 39 (1985), 200-209

[13] International Human Genome Sequencing Consortium, Initial sequencing and analysis of the human genoms, Nature Vol. 409 (02/15/2001), 860-921

[14] J. Johnson, J. Spinrad, A Polynomial Time Recognition Algorithm for Probe Interval Graphs, Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (2001), 477-486

[15] B. Klinz, R. Rüdolf, G.J. Woeginger, Permuting matrices to avoid forbidden submatrices, Discrete Applied Mathematics 60 (1989), 223-248

[16] R. McConnell, An $O(n^2)$-incremental algorithm for modular decomposition of graphs and 2-structures, Algorithmica 14 (1995), 229-248

[17] R. McConnell, Linear-time recognition of circular-arc graphs, Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (2001)

[18] R. McConnell, J. Spinrad, Construction of Probe Interval Models, Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (2002), 866-875

[19] R. McConnell, J. Spinrad, Linear-time modular decomposition and efficient transitive orientation of comparability graphs, 5th. Annual ACM-SIAM Symposium on Discrete Algorithms (1994), Arlington, Virginia, 536-543

[20] R. McConnell, J. Spinrad, Modular decomposition and transitive orientation, Discrete Mathematics 201 (1999), 189-241

[21] F. McMorris, C. Wang, P. Zhang, On probe interval graphs, Discrete Applied Mathematics 88 (1998), 312-324

[22] J. Muller, J. Spinrad, Incremental modular decomposition, Journal of the ACM 36 (1989), 1-19

[23] V. Turau, Algorithmische Graphentheorie, Addison-Wesley, 1996

[24] Universität Rostock, Information System on Graph Class Inclusion; http://wwwteo.informatik.uni-rostock.de/isgci/

[25] M. Waterman, Introduction to Computational Biology, Chapman & Hall, 1995

[26] J. Watson, F. Crick, A structure for Deoxyribose Nucleic Acid, Nature Vol. 171 (04/25/1953), 737-738

[27] P. Zhang, Method of Mapping DNA Fragments, United States Patent No. 5667970, 05/10/94; http://www.cc.columbia.edu/cu/cie/techlists/patents/5667970.htm

[28] P. Zhang, Probe Interval Graph and Its Applications to Physical Mapping of DNA, Manuskript, 1994

[29] P. Zhang, E. Schon, S. Fischer, E. Cayanis, J. Weiss, S. Kistler, P. Bourne, An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA, CABIOS 10 (1994), 309-317

[30] P. Zhang, X. Ye, L. Liao, J. Russo, S. Fischer, Integrated mapping package - a physical mapping software tool Kit, Genomics 55 (1999), 78-87

Index