Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D – 70569 Stuttgart

Diplomarbeit Nr. 2972

Bicluster Visualisierung

Robert Seifert

Studiengang:	Informatik
Prüfer:	Prof. Dr. Daniel Weiskopf
Betreuer:	Dipl.-Inf. Julian Heinrich
begonnen am:	22.09.2009
beendet am:	24.03.2010
CR-Klassifikation:	I.3.8; I.5.3; J.3

Zusammenfassung

Die Analyse von Datensätzen mittels Biclustern gewann zunächst im Bereich der Genexpressionsdaten stark an Popularität und ist heute auch in anderen Bereichen, wie der Text-Analyse, Kunden-Systemen und der Analyse von Stoffwechsel-Erkrankungen ein starkes Werkzeug. Die Techniken zur Erzeugung von Expressionsdaten haben sich in letzter Zeit erheblich verbessert, wodurch eine Vielzahl solcher Daten immer schneller erstellt werden können. Auch die Algorithmen zum Auffinden von Biclustern in Datensätzen haben sich in den letzten zehn Jahren stark entwickelt. Die Visualisierung der berechneten Bicluster weist jedoch noch eine Vielzahl ungelöster Probleme auf. Für eine effiziente Analyse der gruppierten Datenwerte ist es unabdingbar, dass die Visualisierung in der Lage ist, die gefundenen Bicluster anzuzeigen und gegebenenfalls Abhängigkeiten zwischen den einzelnen Biclustern darzustellen.

In dieser Diplomarbeit wird eine Anwendung zur Visualisierung von Biclustern entwickelt. Die Anwendung ist darauf ausgelegt die gefundenen Bicluster auf dem Originaldatensatz darzustellen und den Verlauf einzelner Zeilen genauer herauszuarbeiten. Die Visualisierung basiert hauptsächlich auf einer Heatmap und einer Ansicht in Parallelen Koordinaten. Innerhalb der Heatmap und den Parallelen Koordinaten sollen die Bicluster als zusammenhängende Einheit präsentiert werden. Da dies nicht in allen Fällen durch alleiniges Vertauschen von Zeilen und Spalten realisiert werden kann, werden zusätzlich zur Vertauschung auch Zeilen und Spalten wenn nötig verdoppelt. Dabei wird darauf geachtet die Verdopplungsrate möglichst gering zu halten. Aufgrund der meist sehr großen Datensätze kann die Darstellung durch Überzeichnung unübersichtlich werden. Daher stellt die Anwendung dem Benutzer mehrere Interaktionsmöglichkeiten wie Selektion, (Neu-)Sortierung, Einfärbung sowie Scroll- und Zoom-Funktionen zur Verfügung. Weiterhin kann der Benutzer interaktiv Zeilen und Spalten in der Ansicht verdoppelt anzeigen lassen, um bestimmte Bicluster zusammenhängend darzustellen.

Zunächst wird eine Auflistung und Beschreibung der bisherigen Visualisierungsansätze gegeben und es wird untersucht, in wie weit sich die jeweiligen Ansätze zur Visualisierung von Biclustern eignen. Anschließend werden die Algorithmen für die Anordnung der Bicluster innerhalb der Datenmatrix erläutert und die wichtigsten Aspekte der Architektur und der Implementierung werden beschrieben. Abschließend wird die Anwendung bezüglich ihrer Eignung zur Bicluster-Visualisierung betrachtet und es werden mögliche Verbesserungen angeschnitten.

Danksagungen

Zunächst möchte ich meinem Betreuer Julian Heinrich für seine hilfreichen Tipps und Anregungen danken, die mir stets geholfen haben die Anwendung und Ausarbeitung zu verbessern. Danke Julian, ohne Deine Hilfe wäre die R-Umgebung wahrscheinlich immer noch nicht richtig in meine Anwendung eingebunden und die Pilotstudie hätte wohl nicht so schnell stattfinden können.

Bedanken möchte ich mich auch bei meinen Eltern und meiner Freundin, die mir während meines Studiums immer mit helfender Hand zu Seite standen. Ganz besonderer Dank gilt dabei meiner Mutter, die mir in unzähligen Stunden des Korrekturlesens geholfen hat, die Stolpersteine der deutschen Sprache zu überwinden.

Abschließend möchte ich noch allen Teilnehmern der Pilotstudie danken, durch deren Teilnahme wichtige Erkenntnisse für eine spätere Durchführung einer Benutzerstudie gesammelt werden konnten.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Motivation

1953 revolutionierten Francis Crick und James Watson die moderne Genetik, indem sie die Doppelhelixstruktur des DNS-Moleküls entdeckten. Heute reicht bereits ein einzelnes Haar aus, um den kompletten genetischen Code eines Menschen zu lesen. Ihn jedoch zu verstehen und damit Krankheiten, wie z. B. Krebs oder Diabetes frühzeitig zu erkennen oder sie sogar zu heilen, blieb bisher ein Wunschgedanke.

Immer noch ist man Jahre – vielleicht sogar Jahrzehnte – davon entfernt den DNA-Code des Menschen zu verstehen. Einen viel versprechenden Ansatz stellen derzeit DNA-Microarrays und die damit verbundenen Genexpressionsdaten dar. Dabei entstehen riesige Datenmengen, aus denen man versucht, das Verhalten einzelner Gene zu unterschiedlichen Bedingungen (z. B. verschiedene Zellzyklen oder schwankende Temperaturen) zu bestimmen oder Abhängigkeiten unterschiedlicher Gene zueinander zu ermitteln.

Die Genexpressionsdaten werden in Tabellenform gespeichert und müssen nun so aufbereitet werden, dass solche Zusammenhänge schnell erkannt werden können. Dabei kommen bei der Analyse dieser Tabellen immer öfter Biclusterverfahren zum Einsatz, durch die es möglich ist, Gene und Bedingungen in einen einfachen Zusammenhang zu stellen. Ein Bicluster stellt eine Subtabelle dar, in der sich die dazugehörigen Gene zu den beteiligten Bedingungen in bestimmter Weise gleich verhalten.

Alleine durch die immense Anzahl von Datensätzen und Biclustern ist es sehr schwer bzw. unmöglich alle Zusammenhänge auf einen Blick darstellen zu können. Eine weitere Schwierigkeit stellt die Tatsache dar, dass a priori nicht bekannt ist, was man sehen möchte. In dieser Diplom-

arbeit wird daher ein Werkzeug entwickelt, das zum einen in der Lage ist, möglichst viele gefundene Abhängigkeiten (Bicluster) zwischen den Zeilen und Spalten (Genen und Bedingungen) darzustellen und zum anderen die Möglichkeit bietet, bestimmte Abhängigkeiten genauer zu untersuchen.

1.2 Aufgabenstellung

Ziel dieser Diplomarbeit ist, Methoden zur Visualisierung von Biclustern mit Anwendungen in der Genexpressionsanalyse zu entwickeln. Dafür soll der Bearbeiter nach einem geeigneten Bicluster-Algorithmus recherchieren, der dann für die Visualisierung eingesetzt werden kann. Dabei sollte ein möglichst generischer Algorithmus mit Möglichkeit zur Qualitätsbestimmung gewählt werden. In der Diplomarbeit soll die Wahl des Algorithmus begründet und der Algorithmus beschrieben werden.

Ein weiterer wichtiger Teil der Diplomarbeit ist die Recherche nach vorhandenen Bicluster-Visualisierungen und -Interaktionen sowie die Identifikation möglicher Verbesserungen und Erweiterungen. Insbesondere Heatmaps und Parallele Koordinaten sollen daraufhin untersucht und die Ergebnisse beschrieben werden. Wichtig hierbei ist:

- Es müssen alle gefundenen Bicluster gleichzeitig darstellbar sein.
- Einzelne Bicluster müssen wählbar sein.
- Weitere Informationen zu Biclustern sollen dargestellt werden können.
- Eventuelle Parameter müssen geändert und der Algorithmus gestartet werden können.

Die Visualisierung soll in *SpRay* integriert werden. Deshalb muss die Implementierung in *Qt*/C++, als Plugin für *SpRay* erfolgen. Auf Folgendes muss dabei geachtet werden:

- Die Bicluster müssen in einer geeigneten Datenstruktur, welche Linking/Brushing erlaubt, dargestellt werden.
- Die Visualisierung soll interaktive Frameraten erlauben.
- Das Plugin sollte plattformunabhängig sein.

Für den Test und die Evaluation der Implementation sollen sowohl ein künstlicher Datensatz mit bekannten Biclustern als auch Daten einer Genexpressionsanalyse verwendet werden. Hierbei sollen die Änderungen/Verbesserungen zu existierenden Systemen herausgearbeitet werden. Außerdem soll eine Benutzerstudie durchgeführt werden, bei der die Interaktion hinsichtlich folgender Fragen untersucht werden soll:

- Wie gut lassen sich alle Bicluster darstellen und wie funktioniert die Filterung?
- Können Datenpunkte nach ihrer Bicluster-Zugehörigkeit gefunden werden?
- Was lässt sich über die Relation mehrerer Bicluster zueinander aussagen?
- Wie ist die Qualität/Unsicherheit einzelner Bicluster?

1.3 Vorgehensweise

Diese Arbeit beschränkt sich nicht nur auf die Visualisierung von Biclustern in Genexpressionsdaten, sondern soll versuchen einer Vielzahl von Anwendungsfällen gerecht zu werden. Das bedeutet, dass die Kernfunktionalitäten der Visualisierung sich in abstrakter Weise auf Datenwerte (hier: *double*) beziehen und die Funktionalitäten nicht auf Genen und Bedingungen, sondern auf den Datentypen implementiert werden sollen.

Heatmaps bilden einen weit verbreiteten Ansatz zum Visualisieren großer Datensätze in Tabellenform. Dabei werden die einzelnen Datenwerte auf Farbwerte abgebildet. Der Benutzer bekommt also keine Zahlen in den Zellen präsentiert, sondern erhält eine dem menschlichen Auge angepasste Ansicht. Es stehen nicht die exakten Werte im Vordergrund, sondern vielmehr der Überblick über den gesamten Datensatz. Daher wird auch in dieser Diplomarbeit die Kernvisualisierung auf Heatmaps basieren und soll dem Benutzer als Ausgangspunkt dienen.

Weiterhin soll es die Möglichkeit geben, die gefundenen Bicluster als zusammenhängende Subtabelle innerhalb einer Heatmap auf den Originaldaten darzustellen.[1] Bei überlappenden Biclustern kann es jedoch dazu kommen, dass sich die gleichzeitige Darstellungen der Bicluster durch alleiniges Vertauschen von Zeilen und Spalten gegenseitig ausschließen (vgl. Abschnitt

1 Allgemein gesprochen bedeutet dies, dass auf der Originaltabelle die Bicluster durch Zeilen- bzw. Spaltenvertauschungen dargestellt werden.

3.1 auf Seite 42). GROTHAUS et al. lösen in [Gro06] dieses Problem, indem sie erlauben Zeilen und Spalten der Originaltabelle zu verdoppeln. Hierbei ist darauf zu achten, dass die Verdopplungsrate der Zeilen und Spalten möglichst gering ist. Der in dieser Diplomarbeit entwickelte Algorithmus, der die Umsortierung der Zeilen und Spalten bestimmt und dabei bereits verdoppelte Zeilen bzw. Spalten mit beachtet, wird in Kapitel 3 vorgestellt. Dem Benutzer soll die Möglichkeit gegeben werden, zu entscheiden, ob er die verdoppelten Zeilen bzw. Spalten eingeblendet haben möchte oder ob die Bicluster, die verdoppelte Zeilen bzw. Spalten enthalten unter Umständen als nicht zusammenhängend dargestellt werden.

Um einzelne Zeilen bzw. Spalten miteinander zu vergleichen, soll eine Funktionalität bereitgestellt werden, einzelne bzw. mehrere Zeilen und Spalten auszuwählen und diese dann bezüglich deren Zugehörigkeit zu verschiedenen Bicluster darzustellen. Dazu müssen für die Menge der ausgewählten Zeilen und Spalten diejenigen Bicluster gefunden werden, die eine Übereinstimmung mit den ausgewählten Zeilen bzw. Spalten aufweisen. Diese sollen dann dem Benutzer präsentiert werden. Hierzu muss die Datenstruktur der Spalten und Zeilen so implementiert werden, dass jede Spalte und Zeile die Zugehörigkeit zu den entsprechenden Biclustern bestimmen kann.

Weiterhin sollen zur Visualisierung und Analyse der Bicluster Parallele Koordinaten benutzt werden, mit Hilfe derer es möglich ist, die Zeilen eines Bicluster auf der gesamten Breite aller Spalten zu verfolgen. Bei den Parallelen Koordinaten werden die einzelnen Dimensionen der Datenpunkte als parallele Achsen in einer Ebene aufgetragen. Ein Datenpunkt wird als Verbindungslinie zwischen den Achsen dargestellt. [Ins85]

Eines der Hauptprobleme bei Parallelen Koordinaten tritt bei großen Datenmengen auf. Durch die hohe Anzahl von Linien kommt es vermehrt zu Überlagerungen, wodurch der Verlauf der Linien unter Umständen unkenntlich werden kann. Durch eine Einfärbung einzelner Linien mittels Farbverläufen, einer Filterung und dem Hinzufügen von Transparenz [Art04] [Bar04] [Die09] [Rüb06] [Zho09] sowie dem von ZHOU et al. vorgestellten „Splatting“ [Zho09] kann diesem Problem entgegengewirkt werden. Auch in dieser Arbeit sollen Farben und Transparenz für eine bessere Übersichtlichkeit verwendet werden.

Weiterhin soll es möglich sein, mehrere Bicluster gleichzeitig in der Darstellung der Parallelen Koordinaten zu präsentieren. Eine Möglichkeit einen bestimmten Bicluster aus den Parallelen Koordinaten hervorzuheben bietet *BiVisu* [Che07b] und *BicOverlapper* [San08]. Die Hervorhebung wird dabei durch ein Einfärben der entsprechenden Linien bewirkt. Das Einfärben von mehreren Linien soll auch in dieser Arbeit als Mittel benutzt werden, um den aktuell ausgewählten Bicluster hervorzuheben. Damit alle Bicluster gleichzeitig visualisiert werden können, werden die einzelnen Liniensegmente zwischen den Achsen nur dann als eine einzige Gerade dargestellt, wenn keine der beiden Achsen (Spalten) zu einem Bicluster gehört. Andernfalls werden zwischen den Achsen Schwerpunkte (Zentroide) bestimmt. Die Berechnung erfolgt durch die Mittelwert-Bildung aller Zellen in den entsprechenden Zeilen und Spalten. Dabei sollen alle Linien, die zum selben Bicluster gehören, durch denselben Zentroiden verlaufen. [Luo10]

Damit eine Auswahl der Bicluster in der Heatmap auch in den Parallelen Koordinaten registriert und verarbeitet werden kann, muss ein gemeinsames Datenmodell zwischen den beiden Visualisierungsansätzen bestehen. Dieses wird durch das in Kapitel 3 vorgestellte Datenmodell BiclusterList realisiert. Das Modell bietet den einzelnen Visualisierungsmodulen die Möglichkeit auf neue oder sich verändernde Selektionen zu reagieren. Es wird auf Grundlage des „Listener-Patterns“ implementiert. Dadurch kann sich jedes Modul als „Listener“ bei dem Datenmodell anmelden und wird mittels den *Qt* typischen Signals und Slots über Änderungen bei der Selektion informiert.

Um einen möglichst generischen Ansatz für die Berechnung der Bicluster bereitzustellen, wird ein Interface definiert, mit dessen Hilfe sich dann zur Laufzeit verschiedene Berechnungsmethoden einbinden lassen. Die bereitgestellte Berechnung der Bicluster soll mit Hilfe des *R*-Project stattfinden. Hierzu wird das *R*-Paket *Biclust* [Kai09] benutzt, welches eine Implementierung des von Cheng und Church [Che00] vorgestellten Algorithmus bereitstellt. Das Interface besteht zum einen aus einer *Matrix*-Klasse, die als Eingabe benutzt wird und zum anderen aus einer BiclustModel-Klasse, die die Ausgabe widerspiegelt. Die BiclustModel-Objekte werden dann für die Erstellung der *BiclusterVisaulModel*-Objekte benutzt (vgl. Kapitel 3).

Die in dieser Diplomarbeit entwickelte Anwendung wird abschließend daraufhin untersucht, in wie weit sich mehrere Bicluster gleichzeitig darstellen lassen. Sie wird zusätzlich mit den in

Kapitel 2 besprochenen Ansätzen verglichen. Um die Interaktionsmöglichkeiten und die Benutzerfreundlichkeit der Anwendung zu überprüfen wird eine Pilotstudie durchgeführt.

1.3.1 Gliederung

Kapitel 2 beschreibt zunächst die Grundlagen der Genexpressionsdaten und erläutert die Methoden zur Erstellung solcher Daten. Weiterhin geht es auf die Grundlagen von Biclustern ein, gibt eine Zusammenfassung der Berechnungsverfahren von Biclustern nach [Mad04] und beschreibt den gewählten Algorithmus [Che00]. Anschließend werden die gefundenen, bereits existierenden Visualisierungsansätze aufgelistet und ihre Eignung zur Darstellung von Biclustern genauer betrachtet.

Kapitel 3 geht zunächst auf die Problemstellungen ein. Dabei werden die Punkte genauer erläutert, die bei den jeweiligen Darstellungen besonders zu beachten sind. Danach werden die Lösungsansätze herausgearbeitet und es werden die Algorithmen beschrieben, die für diese Diplomarbeit entwickelt wurden. Bei der Beschreibung der Algorithmen werden die wichtigsten Implementierungsaspekte genauer erläutert.

In Kapitel 4 wird die Anwendung daraufhin untersucht, in wie weit sie sich zur Darstellung von mehreren Biclustern eignet. Weiterhin beschreibt es die Pilotstudie, die für diese Diplomarbeit durchgeführt wurde und wertet die Ergebnisse der Studie aus. Abschließend wird auf die Aspekte eingegangen, die von dieser Anwendung nicht gelöst werden und es werden mögliche Erweiterungen genannt, die diese Aspekte behandeln könnten.

2 Grundlagen und verwandte Arbeiten

2.1 Genexpressionsanalyse

Dieser Abschnitt umfasst die biologischen Grundlagen der Genexpressionsanalyse. Eine Vielzahl der Bicluster-Verfahren und Visualisierungsmethoden fanden ihren Ursprung bei der Analyse von Genexpressionsdaten. Deshalb sollen in diesem Abschnitt die Grundlagen der Genexpressionsanalyse erläutert werden. Es wird dabei auf den Aufbau der DNA, deren Codierung, die Synthese von Proteinen und die damit verbundene Expression von Genen eingegangen. Genauere Informationen sind [Det09], [Dep06], [Bro99], [Leu03] und [Koh06] zu entnehmen.

2.1.1 Die Zelle, DNA, RNA, Gene und Proteine

Jedes höhere Lebewesen besteht aus mehreren Zellen, die die komplette Erbinformationen (Genom) speichern. Das Genom besteht aus mehreren Chromosomen (beim Menschen sind es 46), welche aus der sogenannten DNA[2] aufgebaut sind. Die DNA, die in einer Doppelhelix vorliegt, ist in Chromosomen in mehreren „Verpackungsstufen“ organisiert. Sie besteht aus den vier Basen (Nucleotide): Adenin, Cytosin, Guanin und Thymin, wobei jeweils die Basenpaarungen C-G und A-T auftreten (vgl. Abbildung 1). Bei allen Lebewesen (vom Einzeller bis zum Menschen) sind diese vier Basenpaare gleich. Der einzige mögliche Unterschied, der zwischen verschiedenen DNA-Sequenzen in einzelnen Zellen auftreten kann, ist die Abfolge, in der diese Paare vorliegen.

2 deoxyribonucleic acid (deutsch: DNS – Desoxyribonukleinsäure)

Erstaunlicherweise enthält jede Zelle eines Menschen fast dieselbe Abfolge von Basenpaaren in ihrer Erbinformation. Trotzdem unterscheiden sich aus verschiedenem Gewebe stammende Zellen sowohl in ihrem Aussehen als auch in ihrer Funktionsweise. [Det09]

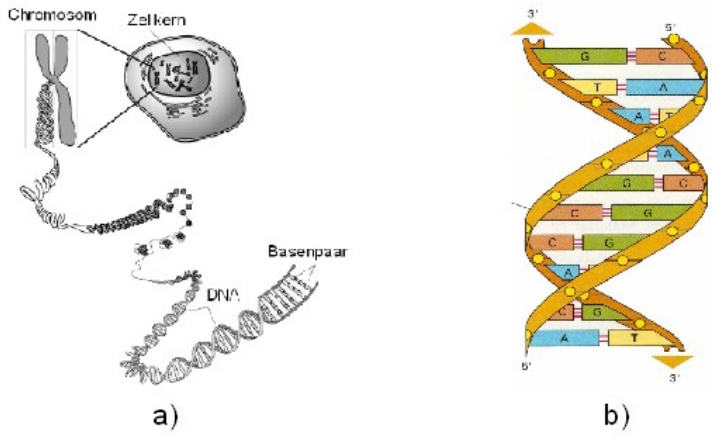


Abbildung 1: Die „Verpackungsstufen“ der Basen und die Doppelhelix
a) Zusammenspiel von Basen, DNA, Chromosomen und Zellen, b) Die Doppelhelixstruktur der DNA mit den Nucleotiden.
Quelle: a) [Koh06], b) [Alb04] auf Seite 224

Den Grundbaustein des Lebens bilden die Proteine. Sie bestimmen die Funktionsweise einer Zelle, indem sie innerhalb der Zelle biochemische Reaktionen durchführen. Insgesamt gibt es 20 Aminosäuren[3], die in Ketten von bis zu 300 Aminosäuren ein Protein bilden. Die wesentliche Funktionalität eines Proteins wird durch die Abfolge der Aminosäuren, die Temperatur und die Umgebung bestimmt. [Dep06]

Dem oben beschriebenen Phänomen geht die sogenannte Genexpressionsanalyse nach. Sie versucht herauszufinden warum bei manchen Zellen, trotz gleicher DNA unterschiedliche Gene aktiv sind und sich daraus die unterschiedlichen Funktionalitäten der verschiedenen Zellen ableiten lassen.

2.1.2 Genexpression

Die wesentliche Arbeit einer Zelle besteht in der Synthese von Proteinen. Dabei muss die in der DNA gespeicherte Information – Reihenfolge der Basenpaare – in eine Abfolge von Aminosäuren übersetzt werden. Diesen Vorgang nennt man Proteinsynthese oder auch Genexpression.

3 Eine Auflistung aller 20 Aminosäuren befindet sich in Anhang A

Es wird jedoch nicht jedes Basenpaar berücksichtigt. Das Genom unterteilt sich in „kodierende“ und „nicht-kodierende“ Abschnitte. Die „kodierenden“ Abschnitte – also diejenigen aus denen Proteine gebildet werden – nennt man auch Gene. [Koh06]

Die Proteinsynthese unterteilt sich in zwei Schritte: die Transkription und die Translation (vgl. Abbildung 2). Bei der Transkription wird die DNA in die RNA[4] aufgespalten. Der Aufbau der RNA ist mit dem der DNA fast identisch, außer dass die Base Thymin durch die Base **U**racil ersetzt wird. Aus den in der RNA gespeicherten Informationen werden dann in der Translation Proteine bzw. Aminosäuren gebildet. [Alb04] [Dep06]

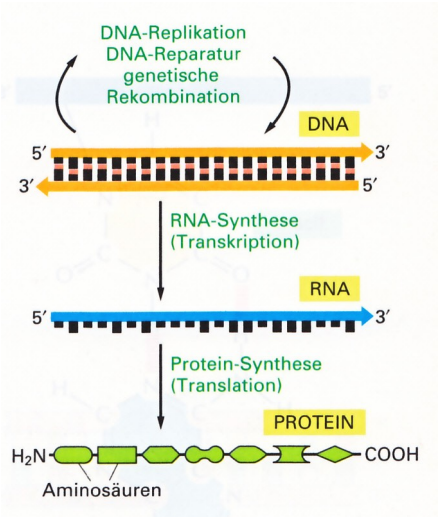


Abbildung 2: Das zentrale Dogma der Molekular-Biologie
Quelle: [Alb04] auf Seite 347

Die eigentliche Proteinsynthese findet an den Ribosomen der Zelle statt. Dabei bilden jeweils 3 Basen ein Codon. Aus einem Codon wird dann eine der 20 Aminosäuren gebildet. Für die Bildung eines Codons gibt es $4^3=64$ Möglichkeiten. Bei nur 20 Aminosäuren ergeben sich damit Redundanzen bzw. sind einige Kombinationen Start-, Stopp-Codierungen. Einen kompletten Überblick über die Codierung liefert Abbildung 3. [Dep06]

4 ribonucleid acid (deutsch: RNS – Ribonukleinsäure)

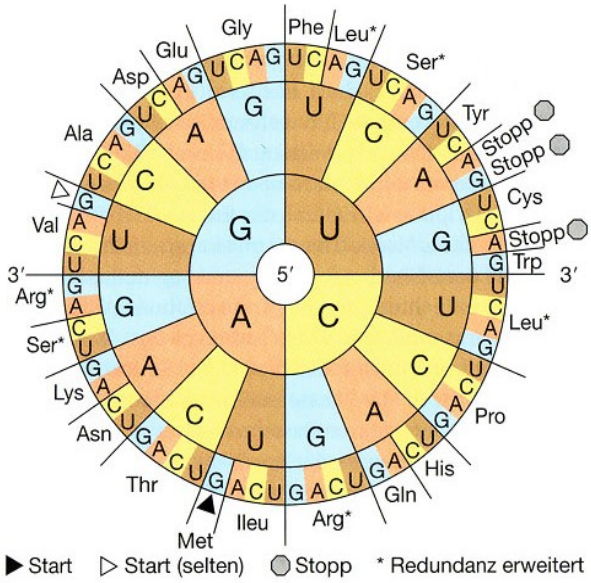


Abbildung 3: Die Gen-Sonne.
Quelle: [Dep06]

Jede Zelle eines Organismus besitzt, wie oben schon erwähnt, ein fast identisches Genom. Jedoch bestehen gravierende Unterschiede in der Funktionsweise mancher Zellen. Das ist damit zu begründen, dass in jeder Zelle unterschiedliche Gene des gemeinsamen Genoms exprimiert werden. [Koh06]

Die Genexpressionsanalyse versucht nun einen Zusammenhang zwischen Genom, unterschiedlichen Bedingungen in den Zellen und exprimierten Genen herzustellen. Dabei werden jedoch nur die exprimierten RNA-Gene bestimmt, mit deren Hilfe dann Rückschlüsse auf die von der Zelle produzierten Proteine gezogen werden können.

2.1.3 DNA-Microarrays

In den letzten Jahren haben sich vor allem die Microarrays zum Erstellen der Expressionsdaten ausgezeichnet. Sie treten besonders durch die folgenden Vorteile hervor: Sie sind billig, flexibel und universell, sehr schnell und benutzerfreundlich. Dabei wurden zwei grundlegende Verfahren entwickelt, die kommerziellen Gen-Chips der Firma AFFYMETRIX und ein frei zugängliches Produkt von der Universität Stanford, die sogenannten cDNA-Microarrays. Beide Verfahren ähneln sich in ihren Grundprinzipien und es wird im Folgenden nur auf die cDNA-Microarrays eingegangen. [Bro99] [Det09]

Ein Microarray stellt einen durchsichtigen Träger dar, der meist aus einer kleinen Glasplatte (etwa in der Größe eines Daumennagels) besteht. Auf diesen Träger werden alle Gene eines Genoms in Form von einzelsträngigen DNA-Molekülen spotweise aufgetragen. Pro Spot ist eine Vielzahl derselben einzelsträngigen DNA-Moleküle vorhanden. Die Moleküle in den Spots dienen dann den zu untersuchenden Genen als Verbindungsstelle. Die Spots trägt ein sogenannter „Arryer"-Roboter auf. Bei der Untersuchung des kompletten Genom einer Hefezelle[5] sind dies beispielsweise 6.200 Spots. [Bro99] [Koh06] [Det09]

Beim cDNA-Microarray Verfahren werden jeweils 2 Proben gleichzeitig betrachtet: eine Probe, deren Gene auf Expression untersucht werden sollen und eine andere Probe, die als Referenz dient. Die RNA wird aus beiden Proben extrahiert und mit Hilfe des Enzyms *Reverse-Transkriptase* wird daraus die cDNA[6] gewonnen. Die cDNAs werden anschließend mit einem fluoreszierenden Farbstoff (meist der rote Farbstoff *Cy5* für die zu untersuchende Probe und der grüne Farbstoff *Cy3* für die Referenzprobe) markiert. Danach werden die beiden cDNAs gemischt und auf den Träger aufgetragen. Die cDNAs hybridisieren mit den zuvor aufgetragenen DNAs auf dem Microarray, das heißt sie verbinden sich mit dem zu ihren Basen passenden DNA-Strang (vgl. Abbildung 4). [Leu03] [Koh06] [Bro99]

Nach der Hybridisierung werden die nicht hybridisierten cDNA Reste abgespült und das Microarray wird mit Hilfe von Lasern abgescannt. Dabei kommen 2 verschiedenfarbige Laser zum Einsatz, ein roter und ein grüner. Aus den beiden abgescannten Bildern wird dann – in einem additiven Verarbeitungsschritt – ein Bild errechnet. Dabei bedeuten rote Spots, dass die jeweiligen Gene der Probe stärker exprimiert wurden, eine grüne Färbung deutet auf eine stärkere Expression der Referenzprobe hin. Sind entsprechende Gene beider Proben gleich stark exprimiert, so erhält der jeweilige Spot eine gelbe Färbung. Bleibt das Bild schwarz, so hat keine Expression stattgefunden (vgl. Abbildung 4). [Leu03] [Koh06] [Bro99]

---

5 Bei *Saccharomyces cerevisiae* sind ca. 6.200 Gene bekannt

6 cDNA: complementary DNA, das Gegenstück zum vorliegenden RNA-Strang, welcher dann an die einzelsträngigen DNA Proben innerhalb der entsprechenden Spots andocken kann.

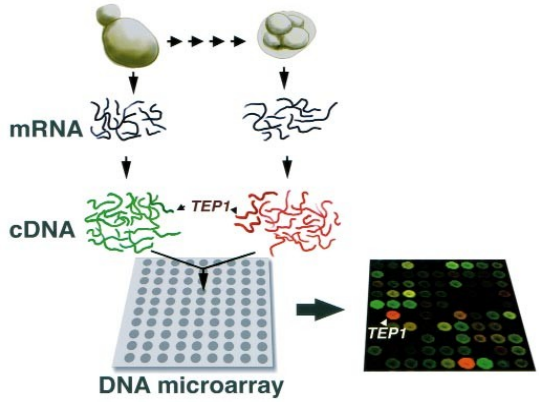


Abbildung 4: Schematische Darstellung des cDNA-Microarray-Verfahrens
Aus den Proben wird die messenger RNA (mRNA) extrahiert, daraus wird die cDNA gewonnen, diese wird mit einem Farbstoff modifiziert und auf das cDNA-Microarray aufgetragen. Eine rote Färbung deutet auf eine starke Expression der Probe hin, eine grüne Färbung auf eine starke Expression der Referenzprobe und eine gelbe Färbung auf eine etwa gleich starke Expression in beiden Proben.
Quelle: [Bro99]

Diese Farbwerte werden dann in normalisierte Zahlenwerte umgerechnet. Die Normalisierung findet statt, um Fehler zu glätten und unterschiedliche Ausgangsbedingungen verschiedener Experimente (z. B. unterschiedliche Microarrays) anzugleichen. [Koh06]

Visualisierungsansätze für Genexpressionsdaten

Als weit verbreitete Visualisierungsmethode haben sich die sogenannten Heatmaps ausgezeichnet. Heatmaps sind generell in der Lage zweidimensionale Datensätze zu visualisieren. Dabei wird einem bestimmten Bereich von Werten eine bestimmte Farbe zugeordnet. Bei der Genexpressionsanalyse entsprechen dann die Zeilen den einzelnen Genen, die Spalten entsprechen den einzelnen Bedingungen. Es entsteht damit eine tabellarische Ansicht, die die Expression der einzelnen Gene unter den verschiedenen Bedingungen (Temperatur, Salzgehalt, Zellzyklus, etc.) darstellt. Eine Weiterentwicklung stellen die Cluster Heatmaps dar, bei denen zusätzlich ein Abhängigkeitsbaum an den Reihen (oft auch über den Spalten) dargestellt wird. Dieses Clustering erleichtert es dem Benutzer Zusammenhänge zwischen den einzelnen Genen zu erkennen. Abbildung 5 zeigt eine solche Cluster Heatmap. Dabei wird eine hohe Genexpression rot, eine geringe Expression mit der Farbe Grün dargestellt. [Wil08]

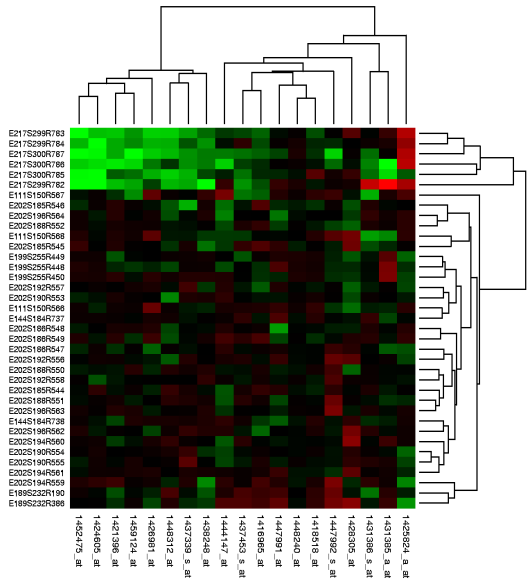


Abbildung 5: Beispiel einer Cluster Heatmap
Quelle: [Wil08]

Einen anderen Ansatz zur Visualisierung der Genexpressionsdaten sind die Parallelen Koordinaten. Hierbei werden die einzelnen Bedingungen als Achsen interpretiert, diese werden parallel zueinander in einer Ebene aufgezeichnet und die einzelnen Datenpunkte werden auf jeder Achse eingetragen. Dabei entspricht ein Datenpunkt auf einer Achse genau einem Eintrag in der Datenmatrix. Eine Zeile innerhalb der Datenmatrix (also ein Gen) entspricht dann einer Verbindungslinie der entsprechenden Datenpunkte. [Che07b]

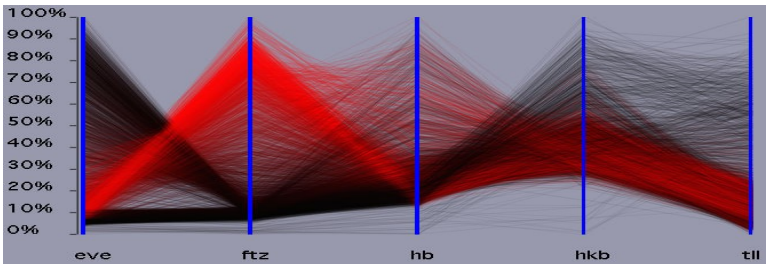


Abbildung 6: Beispielhafte Darstellung von Genexpression in Parallelen Koordinaten
Verbesserung der Darstellung durch Transparenz und Farbgebung
Quelle: [Rüb06]

Eine genauere Betrachtung der Heatmaps und der Parallelen Koordinaten findet sich in Abschnitt 2.5 auf Seite 28.

2.2 Bicluster und deren Berechnung

Ein einfaches Clustering kann die Datenmatrix über die gesamte Gen-Dimension (Zeilen) bzw. die gesamte Bedingungs-Dimension (Spalten) gruppieren. Bicluster hingegen ermöglichen es, bestimmte Gene zu bestimmten Bedingungen zu gruppieren. Dies hat den Vorteil, dass sich gleich verhaltende Gene bei gleichen Bedingungen schnell erkennen lassen. Beispielsweise können damit zelluläre Prozesse hervorgehoben werden, die nur zu bestimmten Bedingungen aktiv sind. [Mad04]

In den letzten Jahren wurde eine Vielzahl von neuen Biclusterverfahren entwickelt. Aufgrund des hohen Berechnungsaufwandes sind viele Verfahren Heuristiken und ergeben somit nur eine approximierte Lösung des Bicluster-Problems wieder. Algorithmen die ohne Heuristiken auskommen haben im schlimmsten Fall eine exponentielle Laufzeit. Dies ist damit zu begründen, dass das Finden eines maximalen Biclusters in einer binären Datenmatrix auf das NP-vollständige „maximum edge biclique problem“ zurückzuführen ist. [Mad04] [Pee00]

Im Folgenden werden eine Definition und eine Unterteilung für Bicluster gegeben, es werden einige der Methoden für die Berechnung von Biclustern vorgestellt und der in der Diplomarbeit verwendete Ansatz von CHENG und CHURCH wird genauer beschrieben. Die folgenden Abschnitte fassen im Wesentlichen Teile von [Mad04], [Kri09] und [Che00] zusammen.

Als Grundlage der Datenmatrix wird angenommen, dass diese n Zeilen und m Spalten enthält. Der Eintrag in der i-ten Zeile und der j-ten Spalte wird als a_{ij} bezeichnet.

2.2.1 Bicluster

Die folgende Definition basiert auf den Definitionen aus [Che00] und [Mad04]:

> *Sei X die Menge der Zeilen und Y die Menge der Spalten der Datenmatrix A, mit A=(X,Y). Sei* a_{ij} *der Eintrag von A in der i-ten Zeile und der j-ten Spalte. Sei weiterhin* $I \subset X$ *und* $J \subset Y$*. Die Submatrix* $A_{IJ}=(I, J)$ *wird dann als Bicluster bezeichnet.*

In der späteren computergestützten Verarbeitung der Datenmatrix, wird diese in einem zweidimensionalen Array gespeichert. Das Array wird über die Indexmengen $\tilde{X}$ (Zeilen) und $\tilde{Y}$ (Spalten) adressiert. Die Identifizierung eines Biclusters erfolgt dann nur noch über die definierten Teilmengen $\tilde{I} \subseteq \tilde{X}$ und $\tilde{J} \subseteq \tilde{Y}$. Ein Bicluster besteht somit nur noch aus einem 2-Tupel von

Indexmengen, die Teilmengen der Indexmengen der Datenmatrix sind. Die zugehörigen Einträge eines Biclusters können dann aus den Indexmengen und dem Array bestimmt werden.

Für eine genauere Betrachtung der unterschiedlichen Typen von Biclustern werden die Definitionen der Mittelwerte von Zeilen, Spalten und allen Werten eines Biclusters aus [Mad04] übernommen:

- Mittelwert der i-ten Zeile a_{iJ} :

$$a_{iJ} = \frac{1}{|J|}\sum_{j\in J} a_{ij} \tag{1}$$

- Mittelwert der j-ten Spalte a_{Ij} :

$$a_{Ij} = \frac{1}{|I|}\sum_{i\in I} a_{ij} \tag{2}$$

- Mittelwert aller Elemente eines Biclusters a_{IJ} :

$$a_{IJ} = \frac{1}{|I|\cdot|J|}\sum_{i\in I,\ j\in J} a_{ij} = \frac{1}{|I|}\sum_{i\in I} a_{iJ} = \frac{1}{|J|}\sum_{j\in J} a_{Ij} \tag{3}$$

Verschiedene Bicluster Arten

Die Einträge der Datenmatrix, die zu einem Bicluster gehören, müssen eine Gemeinsamkeit aufweisen. Dabei wird zwischen verschiedenen Arten von Gemeinsamkeiten unterschieden, die hier genauer betrachtet werden sollen.

Bei perfekten konstanten Biclustern (vgl. Abbildung 7a) haben alle Elemente den gleichen Wert μ . Es gilt also: $a_{ij}=\mu$. μ wird als typischer Wert des Biclusters bezeichnet. Bei gemessenen Datensätzen tritt jedoch immer ein Rauschen auf. Damit kommt es bei fast allen Einträgen in der Datenmatrix zu Abweichungen des Idealwertes. Der Wert des Eintrages in der i-ten Zeile und j-ten Spalte lässt sich damit ausdrücken durch: $a_{ij}=\mu+n_{ij}$. Dabei ist n_{ij} der Wert des Rauschens. Um die Güte eines konstanten Biclusters zu bestimmen eignet sich daher die Varianz: $VAR(I,J)=\sum_{i\in I,\ j\in J}(a_{ij}-a_{IJ})^2$. Ein ideales (rauschfreies) konstantes Bicluster hat damit eine Varianz von 0. [Mad04]

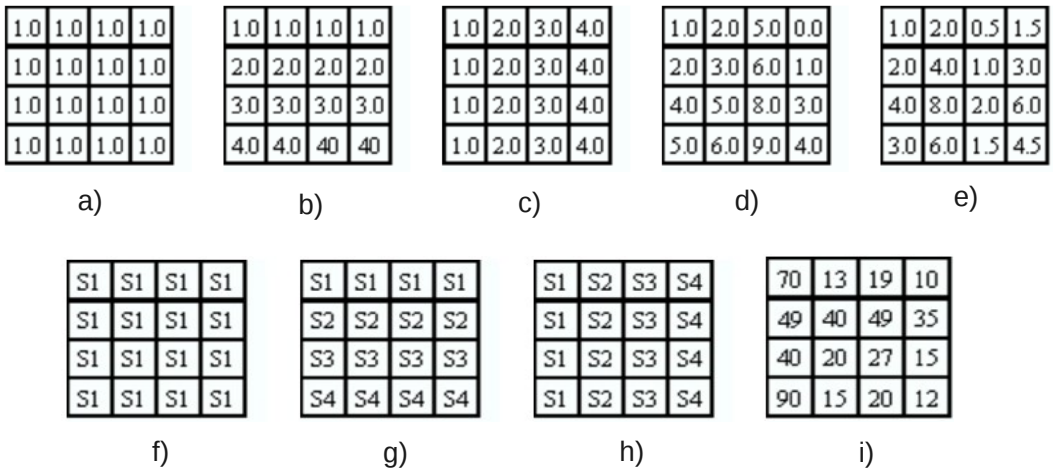


Abbildung 7: Beispiel verschiedener Bicluster Typen
a) konstante Bicluster, b) konstante Reihen, c) konstante Spalten, d) kohärente Werte – additives Modell, e) kohärente Werte – multiplikatives Modell, f) insgesamt-kohärente Evolution, g) kohärente Evolution in Zeilen, h) kohärente Evolution in Spalten, i) nummerisches Beispiel für kohärente Evolution
Quelle: nach [Mad04] und [Koh06]

Bicluster mit konstanten Werten in Zeilen bzw. Spalten (vgl. Abbildung 7b und c) sind in der Genexpressionsanalyse von großer Bedeutung. Ein Bicluster mit konstanten Zeilen deutet auf Gene hin, die bei bestimmten Bedingungen konstante Expressionswerte haben. Ein Bicluster mit konstanten Spalten zeigt Bedingungen, bei denen die Gene denselben Expressionswert besitzen. Ein perfektes Bicluster mit konstanten Reihen besitzt nur Werte für die mit dem Abgleich α_i der i-ten Zeile gilt:

- additives Modell: $a_{ij} = \mu + \alpha_i$ (4)
- multiplikatives Modell: $a_{ij} = \mu \cdot \alpha_i$ (5)

Für die Werte eines perfekten Biclusters mit konstanten Spalten und dem Abgleich β_j der j-ten Spalte gilt:

- additives Modell: $a_{ij} = \mu + \beta_j$ (6)
- multiplikatives Modell: $a_{ij} = \mu \cdot \beta_j$ (7)

Diese Art von Biclustern lässt sich nicht durch das alleinige Berechnen der Varianz bestimmen. Vielmehr müssen die Daten mit Hilfe des Zeilen- bzw. Spalten-Mittelwerts normalisiert werden. [Mad04]

Bicluster mit kohärenten Werten stellen eine Verallgemeinerung der Bicluster mit konstanten Zeilen bzw. Spalten dar. Gemäß den Notationen der vorherigen perfekten Bicluster lässt sich ein perfektes Bicluster mit kohärenten Werte wie folgt definieren:

- additives Modell: $a_{ij} = \mu + \alpha_i + \beta_j$ (8)
- multiplikatives Modell: $a_{ij} = \mu' \cdot \alpha_i' \cdot \beta_j'$ (9)

Aus dem multiplikativen Modell lässt sich auch das Additive bilden, wenn man wie folgt umdefiniert: $\mu = \log(\mu')$, $\alpha_i = \alpha_i'$ und $\beta_j = \beta_j'$. Die Verallgemeinerung lässt sich dadurch zeigen, dass man die Gleichungen der konstanten Zeilen bzw. Spalten erhält, indem man α_i bzw. β_j auf 0 setzt. [Mad04]

Ein Bicluster mit kohärenter Evolution in Zeilen und/oder Spalten weist eine gleiche Entwicklung der Werte auf (vgl. Abbildung 7 f-i). Dabei wird nur auf die Entwicklung der Werte geachtet, jedoch nicht auf ihren absoluten Betrag. In Abbildung 7i ist ein Zahlenbeispiel für eine kohärente Evolution in den Spalten dargestellt. In diesem Bicluster gilt für die Werte der i-ten Zeile: $a_{i4} \leq a_{i2} \leq a_{i3} \leq a_{i1}$. **Ben-Dor** et al. definierten ein Bicluster[7] mit kohärenter Evolution in den Spalten als eine Gruppe von Reihen, deren Werte eine lineare Ordnung über eine Teilmenge von Spalten erzeugen. [Mad04]

Bicluster Strukturen

Viele Bicluster-Berechnungsverfahren unterscheiden sich nicht nur in der Art der Bicluster, die sie auffinden können, sondern auch in der Struktur der Bicluster. Manche Algorithmen sind darauf ausgelegt nur einen Bicluster zu finden (vgl. Abbildung 8a), andere finden eine bestimme zuvor definierte Anzahl von Biclustern. Dabei ist nicht jeder Algorithmus im Stande, Bicluster in allen Strukturen zu finden. In diesem Abschnitt werden die von **Madeira** und **Oliveira** in [Mad04] beschriebenen Strukturen kurz vorgestellt (vgl. Abbildung 8).

Auch bei der Darstellung der Bicluster stellen unterschiedliche Strukturen unterschiedliche Anforderungen an das Visualisierungssystem. Möchte man die Bicluster auf der Originaltabelle darstellen – dieser Ansatz wird auch in dieser Diplomarbeit verfolgt – so ist dies durch einfache

7 Das Konzept bezeichneten sie als OPSM: Order Preserving Submatrix

Zeilen- und Spaltenvertauschungen nur bei den in Abbildung 8a-c dargestellten Strukturen im Allgemeinen möglich.

Bei exklusiven Biclustern gehören jede Zeile und jede Spalte genau einem Bicluster an. In Abbildung 8a und b also bei exklusiven Zeilen und Spalten Biclustern gehört sogar jede Zeile und jede Spalte maximal einem Bicluster an. Bei exklusiven Zeilen/Spalten Biclustern, gehört jede Zeile/Spalte maximal einem Bicluster an, während eine Spalte/Zeile zu mehren Biclustern gehören kann. Bei den in Abbildung 8b-e dargestellten Strukturen muss nochmals zwischen zwei Arten unterschieden werden: Zum einen sind dies die vollständigen Strukturen, die jeder Zeile und jeder Spalte mindestens ein Bicluster zuweisen und zum anderen die nicht-vollständigen Strukturen, die einzelnen Zeilen bzw. Spalten erlauben zu keinem Bicluster zugeordnet zu sein. [Mad04]

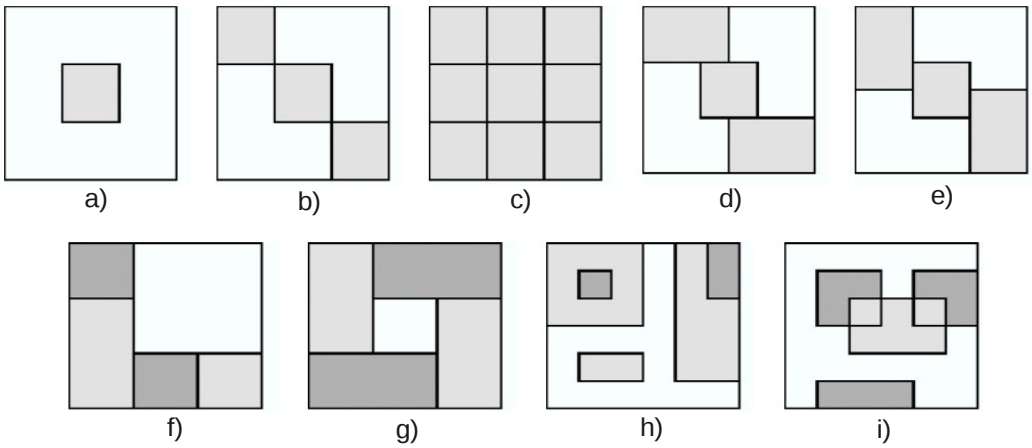


Abbildung 8: Bicluster Strukturen
a) ein Bicluster, b) exklusive Reihen- und Spalten-Bicluster, c) Schachbrett-Struktur, d) exklusive Reihen-Bicluster e) exklusive Spalten-Bicluster f) nicht-überlappende Bicluster mit Baum-Struktur g) nicht-überlappende, nicht-exklusive Bicluster, h) überlappende Bicluster mit hierarchischer Struktur, i) beliebig positionierte überlappende Bicluster.
Quelle: [Mad04]

Die in Abbildung 8f und g dargestellten Strukturen stellen nicht-exklusive Bicluster dar. Dabei ist es erlaubt, dass eine Zeile und eine Spalte zu mehreren Biclustern gehören darf. Auch hier gibt es wieder eine Unterteilung in vollständige und nicht-vollständige Strukturen. Allen Strukturen von a bis g ist jedoch gemein, dass sie keine Überlappungen zulassen. Dies würde für eine Genexpressionsanalyse bedeuten, dass kein Gen-Bedingung-Paar zu mehreren Biclustern gehören darf. Bei überlappenden Strukturen kann zwischen zwei Arten unterschieden werden (vgl.

Abbildung 8h und i). Bei hierarchischen Überlappungen treten nur Bicluster auf, bei denen eines der beiden überlappenden Bicluster eine Teilmenge des anderen darstellt. Den allgemeinsten Fall stellen die beliebig-positionierten, sich überlappenden Bicluster Strukturen dar. Ein für diese Diplomarbeit gewählter Bicluster-Algorithmus sollte in der Lage sein, Bicluster mit dieser Struktur zu finden. [Mad04]

2.2.2 Berechnungsmethoden

Die klassische Beschreibung für das Bicluster-Problem gab HARTIGAN 1972, indem er die Qualität eines Biclusters durch seine Varianz definierte[8]:

$$VAR(I,J) = \sum_{i \in I,\ j \in J} (a_{ij} - a_{IJ})^2 \tag{10}$$

Bei seinem Algorithmus – bekannt als Block Clustering – wird die Datenmatrix rekursiv in zwei Partitionen unterteilt. Das Varianz-Kriterium alleine birgt jedoch den Nachteil, dass die perfekte Unterteilung einer Datenmatrix aus nur 1x1 Biclustern bestehen würde. Diese hätten alle eine Varianz von 0. HARTIGAN definierte daher eine Metrik mit deren Hilfe die Güte von mehreren Biclustern zusammen bestimmt werden kann. Er geht von einer Unterteilung der Datenmatrix $A = (X, Y)$ in K Bicluster aus: $(I, J)_k$ für $k \in 1, \ldots, K$. Die gesamte Varianz der Unterteilung kann dann mit folgender Gütefunktion bestimmt werden:

$$VAR(I,J)_K = \sum_{k=1}^{K} \left(\sum_{i \in I,\ j \in J} (a_{ij} - a_{IJ})^2 \right) \tag{11}$$

Bei jedem Durchlauf wird die Unterteilung so vorgenommen, dass die Reduktion der Gütefunktion maximiert wird. Dies wird so lange durchgeführt, bis die Datenmatrix in K Submatrizen unterteilt ist. Dieser Ansatz ist für das Auffinden von konstanten Biclustern ausgelegt, kann aber durch eine Abänderung der Gütefunktion auch für das Auffinden von Biclustern mit konstanten Zeilen/Spalten oder für Bicluster mit kohärenten Werten benutzt werden. [Mad04] [Kri09]

Den einfachsten Ansatz um Bicluster mit konstanten Zeilen oder Spalten zu finden, stellt eine Normalisierung dar. Dadurch werden diese Art von Biclustern in konstante Bicluster transformiert. Er wird unter anderem vom „Coupled Two-Way Clustering (CTWC)“ verfolgt, bei dem

8 Notationen sind aus Abschnitt 2.2.1 übernommen.

in einem Vorverarbeitungsschritt jede Spalte der Datenmatrix durch ihren Mittelwert normalisiert wird. Einen anderen Ansatz verfolgen CALIFANO et al., die auf das Finden von sogenannten „maximal δ-valid ks-patterns“ abzielen. Dabei beschreibt k sie Anzahl der Zeilen und s die Anzahl der Spalten eines Biclusters. Dieses ist „ δ-valid“, falls der Abstand zwischen dem größten und kleinsten Element kleiner ist als ein δ. Ein Bicluster ist maximal, falls es durch das Hinzufügen einer neuen Spalte bzw. Zeile nicht mehr „ δ-valid“ ist. [Kri09] [Mad04]

CHENG und CHURCH waren die ersten, die eine Biclusterung für Genexpressionsdaten vorschlugen. Sie suchen in der Datenmatrix nach sogenannten δ-Biclustern. Dafür definierten sie für die Güte eines Biclusters den „mean squared residue“[9] Wert H, mit:

$$H(I,J)=\frac{1}{|I|\cdot|J|}\sum_{i\in I,\ j\in J}\left(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ}\right)^2 \tag{12}$$

Ein δ-Bicluster ist nun eine Submatrix (I,J) für deren Güte H gilt: $H(I,J)\leq\delta$ für ein gegebenes $\delta\geq 0$. Bei einem perfekten δ-Bicluster gilt: $\delta=0$. Dabei kann die Abweichung einer Spalte durch $a_{Ij}-a_{IJ}$, die einer Zeile durch $a_{iJ}-a_{IJ}$ angegeben werden. Daraus ergibt sich mit $\mu=a_{IJ}$, $\alpha_i=a_{iJ}-a_{IJ}$, $\beta_j=a_{Ij}-a_{IJ}$ und Gleichung (8): $a_{ij}=a_{iJ}+a_{Ij}-a_{IJ}$. Aufgrund von Rauschen tritt, wie bereits erwähnt, eine Abweichung des Wertes a_{ij} auf. Diese Abweichung bezeichnen Cheng und Church als „Rest“ $r(a_{ij})$. Damit gilt: $a_{ij}=r(a_{ij})+a_{iJ}+a_{Ij}-a_{IJ}$. Für den „Rest“ ergibt sich: $r(a_{ij})=a_{ij}-a_{iJ}-a_{Ij}+a_{IJ}$. Der „Rest“ eines Biclusters ist demnach ein Indikator für die Kohärenz und damit gilt: „je kleiner der Rest, desto stärker die Kohärenz“. Aus dem eben Beschriebenen lässt sich nun Gleichung (12) [10]herleiten:

$$H(I,J)=\frac{1}{|I|\cdot|J|}\sum_{i\in I,\ j\in J}\left(r(a_{ij})\right)^2=\frac{1}{|I|\cdot|J|}\sum_{i\in I,\ j\in J}\left(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ}\right)^2$$

Der Algorithmus basiert auf der Top-Down Strategie, was bedeutet, dass er initial auf den kompletten Spalten- und Zeilenmengen arbeitet. Es werden dann aus einer Submatrix (I,J) jeweils die Menge von Spalten und Zeilen heraus genommen, so dass sich $H(I,J)$ am stärksten reduziert. Dies wird so lange durchgeführt, bis $H(I,J)\leq\delta$ ist oder es sich nicht mehr verkleinert. Somit findet der Algorithmus in jedem Durchlauf genau ein δ-Bicluster. Da auch CHENG

9 Der Mittelwert der Quadrate über die Reste

10 Sie eignet sich zur Berechnung von Biclustern mit kohärenten Werten, siehe Gleichung (8)

und CHURCH ihren Algorithmus darauf ausgelegt haben K Bicluster zu finden, wird der Schleifendurchlauf K mal wiederholt. Um zu vermeiden, dass bereits gefundene Bicluster erneut bestimmt werden, werden die bereits gefundenen Bicluster durch Zufallszahlen „maskiert“, das bedeutet die entsprechenden Matrixeinträge werden durch diese Zufallswerte ersetzt. Dadurch wird es unwahrscheinlich, dass ein bereits einem Bicluster angehöriges Element einem andern Bicluster zugeordnet wird, es werden aber weiterhin alle Zeilen und Spalten bei der Berechnung weiterer Bicluster einbezogen. Dadurch ist es möglich sich überlappende Bicluster (vgl. Abbildung 8i) zu bestimmen. Eine genauere Beschreibung des Algorithmus findet sich in Abschnitt 2.3. [Che00] [Kri09] [Koh06] [Mad04]

Durch die „Maskierung“ von CHENG und CHURCH ist es zwar möglich überlappende Bicluster zu finden, es ist jedoch eher unwahrscheinlich, dass stark überlappende Bicluster erkannt werden. Dieses Problem versucht FLOC (FLexible Overlapping biClustering) zu lösen. Es basiert auf der Bicluster-Definition von CHENG und CHURCH, führt jedoch ein simultanes Clustering aus. Auch hier werden K Bicluster gefunden. Der Algorithmus besteht aus zwei Phasen. In der ersten Phase werden K initiale Bicluster generiert, indem zufällig (mit der unabhängigen Wahrscheinlichkeit p) jede Zeile und jede Spalte mit ihnen kombiniert werden. In der zweiten Phase werden dann die entstandenen Bicluster optimiert, indem bei jeder Zeile und Spalte geprüft wird, in wie weit sie den gesamten Mittelwert der „Reste“ verkleinern kann. Dabei wird nicht nur das Entfernen von Spalten und Zeilen betrachtet, sondern auch das Hinzufügen. Die Optimierungsphase endet, wenn keine Verbesserung der Gesamtqualität der Bicluster zu erwarten ist. [Yan05] [Kri09] [Mad04]

Einen Verfahren zur Bestimmung von Biclustern mit kohärenter Evolution in den Spalten stellt der von BEN-DOR et al. definierte Ansatz mit „Order Preserving Submatrix“ (OPSM) dar. Dabei wird ein Bicluster als Submatrix (I,J) beschrieben, bei der eine lineare Ordnung auf J existiert. Das Cluster-Modell (J,π) wird durch die Menge der Spalten J mit s Spalten und der Permutation $\pi=(c_{1,}\ldots,c_s)$ definiert. Eine Zeile $i\in I$ „unterstützt“ dieses Modell, falls für alle ihre Elemente gilt: $a_{i,\pi[k]}<a_{i,\pi[k+1]}$ für alle $1\leq k\leq s-1$ gilt. Der Algorithmus fängt zunächst mit kleinen Modellen an und erweitert iterativ die besten l Modelle. Dabei werden Modelle mit vielen „unterstützenden“ Zeilen bevorzugt. [Kri09] [Mad04]

2.3 Der Ansatz von Cheng und Church

Da in dieser Diplomarbeit eine konkrete Implementierung für die Visualisierung von Biclustern im Bereich der Genexpressionsanalyse angefertigt werden soll, sollte der Bicluster-Algorithmus an diesen Bereich angepasst sein. Der von CHENG und CHURCH entwickelte Algorithmus [Che00] berechnet eine gegebene Anzahl sogenannter δ -Bicluster (vgl. Abschnitt 2.2.2 auf Seite 19). Diese sind bei der Analyse von Genexpressionsdaten gut geeignet, da man Gene mit gleicher Expression sucht. Eine Menge von Genen mit etwa gleicher Expression kann auch als Menge von Datenpunkten angesehen werden, bei denen sich die einzelnen Datenpunkte um nicht mehr als einen bestimmten Schwellwert von ihrem Mittel unterscheiden. Genau solche Bicluster werden mit Hilfe der „mean squared residue"-Methode von CHENG und CHURCH berechnet.

Die im Folgenden benutzten Algorithmen basieren auf der Arbeit [Che00] und benutzen die Notationen aus Abschnitt 2.2. Die Notationen werden für eine bessere Lesbarkeit nochmals angegeben.

- X: Menge der Zeilen, Y: Menge der Spalten, Teilmengen: $I \subset X$, $J \subset Y$
- Gleichung (1) Mittelwert der i-ten Zeile: $a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}$
- Gleichung (2) Mittelwert der j-ten Spalte: $a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}$
- Gleichung (3) Bicluster-Mittelwert: $a_{IJ} = \frac{1}{|I| \cdot |J|} \sum_{i \in I,\ j \in J} a_{ij}$
- Gleichung (12) „mean squared residue" H: $H(I,J) = \frac{1}{|I| \cdot |J|} \sum_{i \in I,\ j \in J} \left(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ}\right)^2$

Die Berechnung der Bicluster erfolgt in drei Schritten: *Single Node Deletion*, *Multiple Node Deletion* und *Node Addition*. Diese drei Schritte sollen kurz beschrieben werden.

Algorithmus 1 (Single Node Deletion)

Als Eingabe erwartet der Algorithmus die Datenmatrix A mit der gesamten Zeilenmenge X und der gesamten Spaltenmenge Y und ein $\delta > 0$. Es werden für alle

Zeilen in $I=X$ die Werte a_{iJ} und alle Spalten in $J=Y$ die Werte a_{Ij} sowie die beiden Werte a_{IJ} und $H(I,J)$ berechnet. Der Algorithmus führt die folgenden Schritte so lange durch, bis die Bedingung $H(I,J)\leq\delta$ erfüllt ist:

1) Entferne die Zeile $i\in I$ aus I bzw. die Spalte $j\in J$ aus J, für die der Fehler $d(i)$ bzw. der Fehler $d(j)$ am größten ist.
2) Berechne die Werte a_{iJ}, a_{Ij}, a_{IJ} und $H(I,J)$ erneut.

Dabei gilt für die Fehler:

$$d(i)=\frac{1}{|J|}\sum_{j\in J}(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ})^2 \text{ und } d(j)=\frac{1}{|I|}\sum_{i\in I}(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ})^2$$

Algorithmus 2 (Multiple Node Deletion)

Die von dem Algorithmus erwartete Eingabe ist die Datenmatrix A mit der gesamten Zeilenmenge X und der gesamten Spaltenmenge Y, ein $\delta\geq 0$ und ein $\alpha>1$. Auch hier werden wieder für alle Zeilen in $I=X$ die Werte a_{iJ} und alle Spalten in $J=Y$ die Werte a_{Ij} sowie die beiden Werte a_{IJ} und $H(I,J)$ berechnet. Auch hier wird eine Schleife so lange durchlaufen, bis die Bedingung $H(I,J)\leq\delta$ erfüllt ist. Die Schritte bei jeder Iteration sind:

1) Entferne diejenigen Zeilen $i\in I$ aus I, für die gilt:

$$\frac{1}{|J|}\sum_{j\in J}(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ})^2 > \alpha H(I,J).$$

2) Berechne a_{Ij}, a_{IJ} und $H(I,J)$ erneut.
3) Entferne diejenigen Spalte $j\in J$ aus J, für die gilt:

$$\frac{1}{|I|}\sum_{i\in I}(a_{ij}-a_{iJ}-a_{Ij}+a_{IJ})^2 > \alpha H(I,J)$$

4) Falls keine Zeile und keine Spalte entfernt wurde führe Algorithmus 1 aus.

Algorithmus 3 (Node Addition)

Als Eingabe wird die Datenmatrix A mit der gesamten Zeilenmenge X und der gesamten Spaltenmenge Y, eine Zeilenmenge I und eine Spaltenmenge J erwartet. I und J identifizieren dabei ein Bicluster. Die Schleife wird so lange durchlaufen,

bis keine Zeile und keine Spalte mehr hinzugefügt werden kann. Dabei werden die folgenden Schritte ausgeführt:

1) Berechne für alle Zeilen $i \in X$ die Mittelwerte a_{iJ}, für alle Spalten $j \in Y$ die Mittelwerte a_{Ij}, sowie den Mittelwert a_{IJ} und $H(I,J)$.
2) Füge jede Spalte $j \notin J$ ein, mit:

$$\frac{1}{|I|}\sum_{i \in I}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I,J)$$

3) Berechne a_{iJ}, a_{IJ} und $H(I,J)$ erneut.
4) Füge alle Spalten $i \notin I$, mit:

$$\frac{1}{|J|}\sum_{j \in J}(a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I,J)$$

5) Füge für jede bisher noch nicht eingefügte Zeile $i \notin I$ ihre Inverse ein, falls:

$$\frac{1}{|J|}\sum_{j \in J}(-a_{ij} + a_{iJ} - a_{Ij} + a_{IJ})^2 \leq H(I,J)$$

Bevor die drei Algorithmen auf der Datenmatrix ausgeführt werden, müssen zunächst fehlende Daten (meist gekennzeichnet durch eine -1 oder ein NaN) in der Datenmatrix ersetzt werden. Dies machen **Cheng** und **Church**, indem sie für die fehlenden Werte Zufallszahlen einsetzen. Die Idee diese Ansatzes ist, dass die zufällig gewählten Zahlen keine Muster bilden und damit bei der *Node Deletion* mit hoher Wahrscheinlichkeit gelöscht werden.

Der Algorithmus findet eine zuvor definierte Anzahl von Biclustern. Er erwartet als Parameter die Datenmatrix A, ein $\alpha > 1$ (Parameter für die *Multiple Node Deletion*), ein $\delta \geq 0$ (Parameter für die Ungenauigkeit der Bicluster) und ein K für die zu findende Anzahl der Bicluster. Zunächst wird eine Kopie A' der Datenmatrix A erstellt und alle fehlenden Werte werden durch Zufallszahlen ersetzt. Dann wird Algorithmus 2 (*Multiple Node Deletion)* auf A' durchgeführt. Dieser ruft in Schritt 4, falls keine mehrfache Löschung erfolgt ist, Algorithmus 1 mit A' auf. Nach der *Node Deletion* ist der bisher gefundene Bicluster nicht maximal. Deshalb wird anschließend noch die *Node Addition* (Algorithmus 3) ausgeführt, in der der Bicluster erweitert wird. **Cheng** und **Church** führten jedoch Algorithmus 3 nur einmal aus, da eine Iteration der Schritte in Algorithmus 3 keine nennenswerten Verbesserungen ergab. Weiterhin ist zu beach-

ten, dass jede Ausführung von Algorithmus 1, 2 und 3 den Wert H reduziert. Nach Ausführung des Algorithmus 3 auf A ist der Bicluster vollständig berechnet und die Zeilen- und Spaltenmenge werden ausgegeben. Da der Algorithmus deterministisch abläuft, würde eine erneute Ausführung der drei Schritte wieder den gleichen Bicluster liefern. Deshalb werden nach Beendigung von Algorithmus 3 die Felder in A' mit Zufallszahlen „maskiert“, die dem gefundenen Bicluster angehören. Daher wird es sehr unwahrscheinlich, dass die bereits einem Bicluster angehörigen Felder einem anderen Bicluster zugeordnet werden. Um trotzdem überlappende Bicluster zu finden, wird Algorithmus 3 (*Node Addition*) mit der nicht „maskierten“ Matrix A aufgerufen. Es können daher bereits Biclustern zugehörige Felder einem neuen Bicluster hinzugefügt werden.

Die Gesamtlaufzeit des Algorithmus um K Bicluster in einer $(n \times m)$-Matrix zu finden beträgt: $O(nm \cdot (n+m) \cdot K)$. Dies liegt vor allem an Algorithmus 1, der maximal n+m Schleifendurchläufe benötigt und daran, dass eine Berechnung aller Mittelwerte in $O(nm)$ liegt. [Yan05] [Che00]

Die Wahl des eben beschriebenen Algorithmus ist vor allem darin zu begründen, dass er überlappende δ-Bicluster findet und sich damit besonders gut zur Analyse von Genexpressionsdaten eignet. Ein weiterer wichtiger Vorteil ist sie Tatsache, dass eine fertige Implementierung des Algorithmus bereits im *R*-Paket *Biclust* vorliegt. Der Algorithmus kann damit aus *R* gestartet werden, indem lediglich die Parameter übergeben werden (vgl. Abschnitt 2.4).

Ein einschränkender Nachteil des Algorithmus ist, dass er wegen der „Maskierung“ sich stark überlappende Bicluster nur mit einer sehr geringen Wahrscheinlichkeit auffindet. Eine Verbesserung stellt der von YANG et al. entwickelte FLOC-Algorithmus in [Yan05] dar. Auch für diesen Algorithmus gibt es eine Implementierung im *R*-Paket *BicARE*. Da der Algorithmus ein simultanes Clustering durchführt, ist er in der Lage auch stark überlappende Bicluster zu finden.

In dieser Diplomarbeit soll das Ausführen von Bicluster-Algorithmen dynamisch zur Laufzeit erfolgen. Deshalb wird eine abstrakte Klasse implementiert, wodurch leicht neue Bicluster-Algorithmen dem Programm hinzugefügt werden können. Eine genauere Beschreibung der Vorgehensweise enthält Kapitel 3.

2.4 R und die Pakete Biclust und BicARE

R stellt eine Umgebung zur statistischen Berechnung und Auswertung von Daten bereit. Es kann als eine Implementierung der Sprache S angesehen werden. In erster Linie stellt *R* ein Kommandozeilen-Interface bereit, indem sich Arrays und Matrizen definieren und bearbeiten lassen. Weiterhin bietet *R* die Möglichkeit Pakete einzubinden, die eine bestimmte Funktionalität zur Verfügung stellen. Ein Paket ist vom Grundsatz her eine Art Modul, das eine bestimmte Abfolge von Funktionsaufrufen beschreibt. Diese Abfolgen sind dann wiederum in Funktionen gekapselt, die sich innerhalb der *R* Umgebung aufrufen lassen.

Weiterhin stellt *R* eine Schnittstelle zur Sprache C bereit, die vorwiegend dazu verwendet wird, C-Code von *R* aus aufzurufen. Dies wird vor allem benutzt, um die Effizienz von Funktionen in *R* zu steigern. Unter anderem verwendet auch das Paket *Biclust* eine fertige Implementierung des Algorithmus von CHENG und CHURCH durch Aufruf des von ihnen bereitgestellten C-Codes. Da *R* standardmäßig nur ein Kommandozeilen-Interface bietet, gibt es auch die Möglichkeit *R* Kommandos aus C aufzurufen. So kann einer *R* Konsole eine grafische Benutzer-Schnittstelle hinzugefügt werden. Hierzu stehen einige Bibliotheken zur Verfügung, die als .h-, .so- beziehungsweise .dll-Dateien dem Code hinzugefügt werden. Diese bieten die Möglichkeit Werte zwischen dem C-Programm und *R* auszutauschen.

Eine genaue Beschreibung der Schnittstellen und der *R*-Funktionalitäten sind [R_D09a], [R_D09b], [R_D09c], [R_D09d] und [R_D09e] zu entnehmen.

Das Paket Biclust

Das Paket *Biclust* [Kai09] ist ein Modul, das mehrere Funktionen bereitstellt, mit deren Hilfe sich Bicluster berechnen lassen. Unter anderem bietet es die Möglichkeit den Algorithmus von CHENG und CHURCH auszuführen. Der Funktionsaufruf hat folgenden Aufbau:

$$biclust(x, method=BCCC(), delta=0.6, alpha=1.5, number=100)$$

Dabei ist *x* ein Tabellen-Objekt, der Parameter *method=BCCC()* veranlasst die Funktion biclust den Algorithmus von CHENG und CHURCH auszuführen, *delta* gibt einen Schwellwert für die

Güte eines Biclusters an, *alpha* gibt einen Faktor für die Beschleunigung der Multiple Node Deletion an und *number* definiert die zu findende Anzahl von Biclustern.

Die Funktion liefert als Rückgabe ein Objekt der Klasse Biclust. Die Biclust-Objekte haben unter anderem die Slots *RowxNumber* und *NumberxCol*. Diese sind logische Matrizen, wobei *RowxNumber* eine 1 in Zeile i und Spalte j enthält, wenn die i-te Zeile zum j-ten Bicluster gehört. *NumberxCol* enthält hingegen eine 1 in Zeile i und Spalte j wenn die j-te Spalte zum i-ten Bicluster gehört.

Das Paket BicARE

Das Paket BicARE [Ges09] stellt den von YANG et al. [Yan05] entwickelten FLOC (FLexible overlapping biClustering) Algorithmus zu Berechnung von Biclustern bereit. Der Funktionsaufruf hat folgenden Aufbau:

$$FLOC(x, k=20, pGene=0.5, pSample=0.5, r=NULL, N=8, M=6, t=100)$$

Dabei ist *x* die Datenmatrix, *k* gibt die zu findende Anzahl von Biclustern an, *pGene* ist die initiale Wahrscheinlichkeit, dass Gene einem Bicluster angehören, *pSample* ist die initiale Wahrscheinlichkeit dass Spalten einem Bicluster angehören. *N* und *M* geben die minimale Anzahl von Zeilen und Spalten an, die einem Bicluster angehören sollen, *r* ist ein Schwellwert für den „Rest“ eines Biclusters, ähnlich dem delta bei *Biclust* und *t* gibt die Anzahl der auszuführenden Iterationen an.

Als Resultat liefert die Funktion ein Objekt der Klasse biclustering zurück. Aus diesen Objekten kann mit Hilfe der Funktion bicluster aus einem biclustering-Objekt ein Bicluster ausgelesen werden. Der Funktionsaufruf hat folgenden Aufbau:

$$bicluster(biclustering, k=1, graph=false)$$

biclustering ist ein Objekt der Klasse biclustering, *k* gibt die Nummer des Biclusters innerhalb des biclustering-Objeckts an und *graph* veranlasst das Zeichnen des Biclusters, falls es auf true gesetzt wird.

2.5 Bicluster-Visualisierungsverfahren

Dieser Abschnitt befasst sich mit bereits existierenden Visualisierungsansätzen für Bicluster. Zunächst wird auf die Grundlagen für zwei weit verbreitete Ansätze – die Heatmaps und die Parallelen Koordinaten – eingegangen und im späteren Verlauf werden die gefundenen Anwendungen vorgestellt.

2.5.1 Heatmaps

Die Grundidee einer Heatmap ist es, eine $(m \times n)$ Datenmatrix auf ein $m \cdot n$ großes Raster zu übertragen, wobei die Datenwerte auf Farbwerte abgebildet werden. Dabei ist darauf zu achten, dass eine signifikante Unterscheidung der Datenwerte durch eine signifikante Unterscheidung in der Farbgebung dargestellt wird. Die einfachste Farbskala für einen solchen Zusammenhang ist eine Abbildung der Datenwerte auf Grautöne.

In der Genexpressionsanalyse hat sich eine Abbildung der Datenwerte auf eine Farbskala von Grün über Schwarz nach Rot durchgesetzt. Dies liegt vor allem an der Tatsache, dass bei der Gewinnung der Genexpressionsdaten die Abtastung der Microarrays mittels eines roten und grünen Lasers erfolgt (vgl. Abschnitt 2.1.3). Abbildung 9 zeigt eine solche Farbskala.

Abbildung 9: Farbskala aus *BiVisu*
Quelle: [BiV10]

Um die Position und Zusammengehörigkeit einzelner Zellen zu einem Bicluster zu verdeutlichen werden innerhalb der Heatmap die Zellen umsortiert und ein umschließendes Rechteck um die entsprechenden Zellen gezeichnet. Dabei beschränken sich die meisten Anwendungen darauf einen bzw. mehrere exklusive (also sich nicht überlappende) Bicluster in einer Heatmap darzustellen. Die Heatmaps werden durch Zeilen- und Spaltenvertauschungen so verändert, dass der/die anzuzeigende/n Bicluster oben links in der Heatmap auftreten.

GROTHAUS et al. stellen sich in [Gro06] dem Problem überlappende Bicluster in einer Heatmap darzustellen. Sie erlauben Zeilen und Spalten zu verdoppeln, falls Bicluster durch alleinige Zeilen- und Spaltenvertauschungen nicht zusammenhängend dargestellt werden können. Abbildung

10 zeigt eine solche Heatmap mit umschließenden Rechtecken um die entsprechenden Zellen. Dabei entspricht ein Bicluster genau einem Rechteck. Die Anwendung zum Erstellen solcher Heatmaps ist jedoch in ihrer Interaktionsmöglichkeit beschränkt. Der Benutzer kann lediglich durch Angabe der Bicluster (Zeilen und Spalten), die im Fokus seines Interesses stehen, diese hervorheben bzw. die anderen ausblenden lassen. Eine Interaktion mit der Maus ist nicht möglich, da die Anwendung nur statische Bilder generiert.

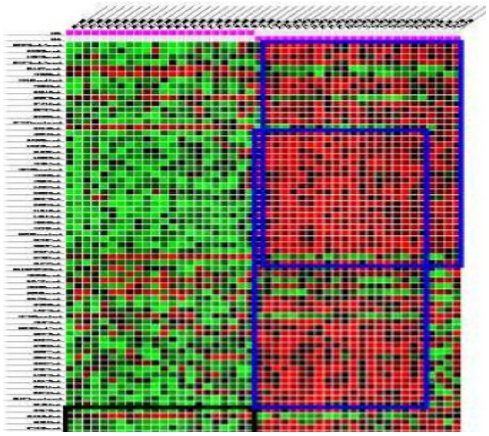


Abbildung 10: Umschließende Rechtecke innerhalb einer Heatmap
Quelle: [Gro06]

Diese Anwendung ist die einzige, welche in der Lage ist, überlappende Bicluster innerhalb einer Heatmap darzustellen. Auch in der Anwendung, die in dieser Arbeit entwickelt wird, soll das Prinzip benutzt werden, um überlappende Bicluster darzustellen.

2.5.2 Parallele Koordinaten

Parallele Koordinaten werden dazu benutzt, den Verlauf einer gesamten Zeile bzw. Spalte darzustellen. Dabei werden die einzelnen Datenpunkte auf parallele Achsen aufgetragen und mittels Liniensegmenten verbunden. Abbildung 11 zeigt eine solche Darstellung von Datenpunkten in Parallelen Koordinaten. [Ins85]

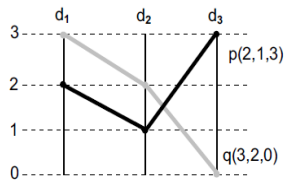


Abbildung 11: Beispielhafte Darstellung zweier Datenpunkte in Parallelen Koordinaten
Quelle: [Bar04]

Eines der Hauptprobleme, das beim Zeichnen der Parallelen Koordinaten für große Datenmengen entsteht, ist die Überzeichnung. Dabei kann es passieren, dass der Verlauf der zu Beginn gezeichneten Linien durch das spätere Zeichnen anderer Linien unkenntlich wird. Dieser Tatsache kann vor allem durch Transparenz und der Einfärbung von Linien entgegengewirkt werden. Ebenfalls kann durch Ausblenden der gerade nicht im Fokus stehenden Linien der Grad der Überlagerung reduziert werden. Abbildung 12 zeigt eine Verbesserung der Darstellung mittels Transparenz, Einfärbung und Ausblenden.

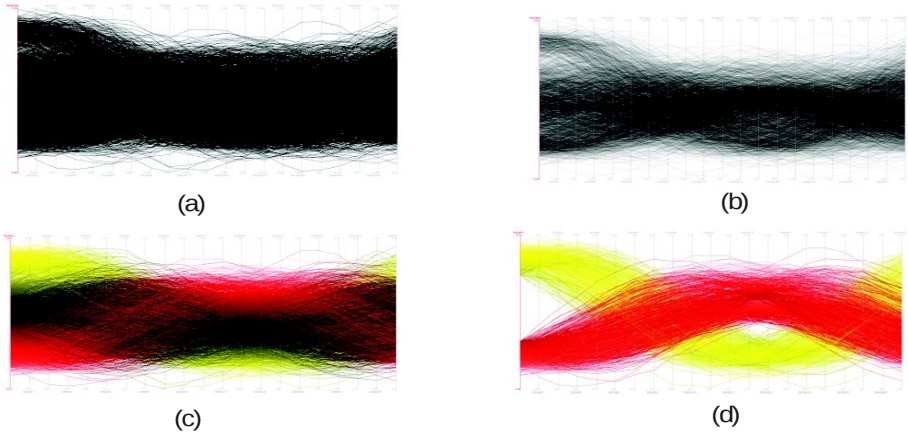


Abbildung 12: Verbesserung der Darstellung bei Parallelen Koordinaten
(a) einfaches Zeichnen der Linien, (b) mit Transparenz, (c) mit Transparenz und Einfärbung (d) mit Transparenz, Einfärbung und Ausblenden
Quelle: [Die09]

Anhand des Verlaufes der Linien in Parallelen Koordinaten lassen sich auch unterschiedliche Arten von Biclustern erkennen. Konstante Bicluster zeichnen sich dadurch aus, dass sich zwischen zwei Achsen nur eine waagrechte Linie befindet, sich also alle Linien des Biclusters überdecken. Bicluster mit konstanten Spalten werden ebenfalls durch nur eine Linie zwischen zwei Achsen dargestellt, diese ist jedoch nicht waagrecht. Bicluster mit konstanten Reihen werden durch nur waagrechte Linien abgebildet, jedoch sind diese zwischen den Achsen versetzt dargestellt. Abbildung 13 zeigt den Verlauf von kohärenten Biclustern, dabei zeigt (a) das additive (b) das multiplikative Modell.

Additive Bicluster sind durch parallele Verbindungslinien zwischen den Achsen zu erkennen. Multiplikative Bicluster weisen ein konstantes Verhältnis der Abstände auf den Achsen auf, die zu einem Bicluster gehören. Bildet man die Quotienten aus den Werten auf den Achsen C2 bzw.

C3 mit denen auf C1, so würde sich für die Quotienten C2/C1 bzw. C3/C1 ein für alle Linien konstanter Wert ergeben [Che07b]. Multiplikative Bicluster sind jedoch nur schwer auf den ersten Blick erkennbar, da ein konstantes Verhältnis der Werte auf den Achsen keiner signifikanten geometrischen Eigenschaft entspricht.

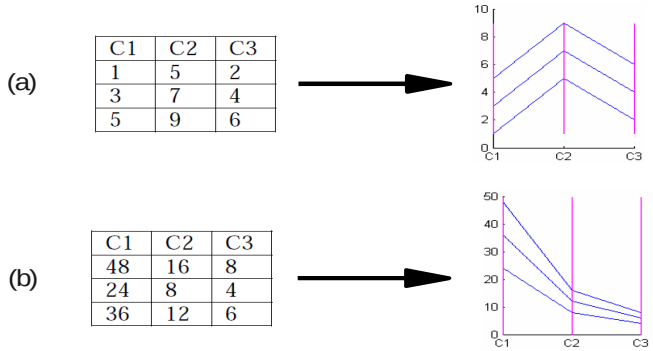


Abbildung 13: Signifikanter Verlauf von additiven und multiplikativen Biclustern in Parallelen Koordinaten
(a) additiver Bicluster, (b) multiplikativer Bicluster
Quelle: [Che07a]

Ein weiteres Problem bei der Erkennung von Biclustern stellt das Überdecken der Linien eines Biclusters durch andere Linien dar. Die Überlagerung wird mit zunehmender Größe des darzustellenden Datensatzes stärker. In Abbildung 12(a) sind dadurch Verläufe innerhalb eines Biclusters nicht erkennbar und damit wird die Zuordnung einzelner Linien bzw. Achsen zu einem bestimmten Bicluster unmöglich. Um dem Überzeichnen bei großen Datenmengen entgegenzuwirken werden bei vielen Visualisierungen im Wesentlichen zwei Ansätze verfolgt. Beim ersten Ansatz werden die Zeilen (Liniensegmente), die nicht dem aktuell ausgewählten Bicluster angehören, ausgeblendet. Es werden also nur die dem aktuell ausgewählten Bicluster zugehörigen Liniensegmente gezeichnet. Der andere Ansatz hebt die aktuell ausgewählten Liniensegmente (Zeilen eines Biclusters) durch Einfärben hervor. Dabei ist darauf zu achten, dass die eingefärbten Linien zuletzt gezeichnet werden, um eine Überlagerung zu vermeiden.

Auf eine genauere Beschreibung der Ansätze soll in diesem Abschnitt verzichtet werden. Sie werden in den folgenden Abschnitten entsprechend ihrer Benutzung in den einzelnen Visualisierungen betrachtet.

2.5.3 Visualisierung in R mit dem Biclust-Paket

Das *Biclust*-Paket ist eine Erweiterung für die *R*-Umgebung und kann mit Hilfe des Befehls *install.packages("biclust", repos="http://R-Forge.R-project.org")* eingebunden werden. Dieser Abschnitt bezieht sich auf [Kai09].

Neben den in Abschnitt 2.4 vorgestellten Berechnungsmethoden für Bicluster bietet das Paket eine Vielzahl von Visualisierungen für die berechneten Bicluster. Dies sind unter anderem: (a) Heatmaps, (b) Parallele Koordinaten und (c) Bubble-Plots. Die Visualisierungen sind in Abbildung 14 dargestellt.

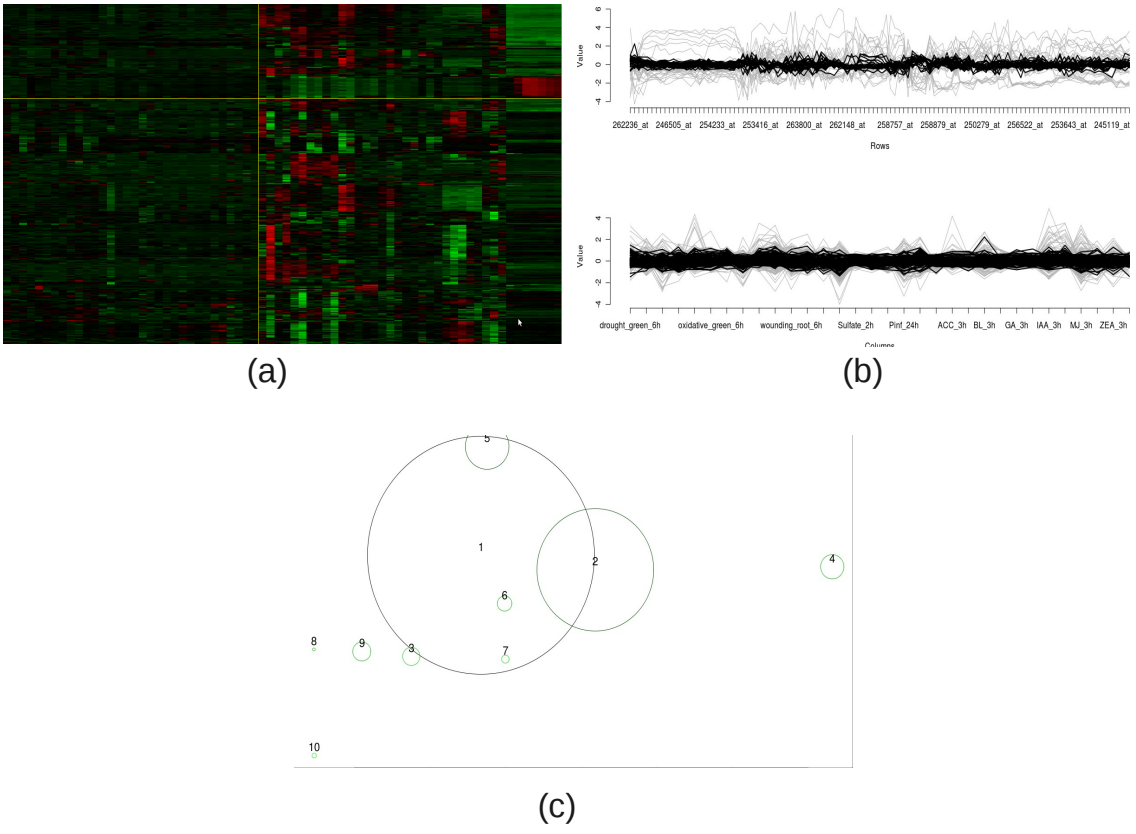


Abbildung 14: Bicluster-Visualisierungen aus dem *R*-Paket *Biclust*
(a) Heatmap, (b) Parallele Koordinaten, (c) Bubble-Plot
Quelle: Screenshots aus dem *Biclust*-Paket in *R* [Kai09]

Die Diagramme in Abbildung 14 beziehen sich alle auf dieselbe Menge gefundener Bicluster[11] wobei (a) und (b) jeweils nur den ersten gefundenen Bicluster darstellen. Alle erstellten Diagramme werden in separaten Fenstern angezeigt. In keinem der angezeigten Diagramme ist eine Interaktion mittels Maus oder Tastatur möglich. Änderungen an den Diagrammen müssen durch erneutes Aufrufen der entsprechenden Funktion mit geänderten Parametern durchgeführt werden.

Die Heatmap in Abbildung 14(a) wird dadurch erzeugt, dass die entsprechenden Zeilen und Spalten der Heatmap so umsortiert werden, dass sie sich oben links befinden. Damit ist gewährleistet, dass immer mindestens ein Bicluster zusammenhängend dargestellt ist. Es besteht weiterhin die Möglichkeit, dass auch mehrere Bicluster in der Darstellung auftreten, jedoch nur wenn es sich um exklusive Bicluster handelt (vgl. Abbildung 8(b) auf Seite 18) . Die Bicluster werden zusätzlich durch gelbe Linien von den anderen Zellen der Heatmap getrennt. Die Zellen, die nicht dem Bicluster angehören, können auch ausgeblendet werden. Überlappende Bicluster oder Bicluster mit nur gleichen Zeilen bzw. Spalten lassen sich in dieser Ansicht nicht darstellen. So bleibt die Bicluster-Analyse in einer Heatmap meist nur auf einen einzigen Bicluster beschränkt. Weiterhin kann der Benutzer keinen Einfluss auf die Sortierung der Zeilen und Spalten innerhalb der Heatmap nehmen. Das Heatmap-Diagramm wurde mit folgendem Befehl erstellt, wobei *x* die Datenmatrix und *bic* die Menge der Bicluster-Objekte sind:

$$drawHeatmap(x, bicResult = bic, number = 1, plotAll = TRUE, local = FALSE)$$

Die in Abbildung 14(b) dargestellten Parallelen Koordinaten haben im oberen Teil die Zeilen als parallele Achsen aufgezeichnet, im unteren Teil entsprechen die parallelen Achsen den Spalten. Dabei sind in beiden Teilen alle Spalten bzw. Zeilen in den Parallelen Koordinaten dargestellt. Die Spalten bzw. Zeilen, die nicht einem Bicluster angehören sind in Grau, die die dem Bicluster angehören sind in Schwarz dargestellt. Dabei werden die schwarzen Linien über die grauen Linien gezeichnet, um die „ausgewählten“ Linien ohne Überlagerung darstellen zu können. In beiden Teilen wird jedoch auf Transparenz verzichtet, wodurch der Verlauf bei sich überdeckenden schwarzen bzw. grauen Linien nicht erkennbar ist. Der Benutzer hat zusätzlich die Möglichkeit die grauen Linien auszublenden, um nur die dem Bicluster zugehörigen Zeilen zeichnen zu lassen. Es besteht jedoch keine Möglichkeit die Zeilen über alle Spalten darzustel-

11 *bic <- biclust(BicatYeast, method=BCCC(), delta=0.1, alpha=5, number=10)*

len und die Spalten nach Zugehörigkeit zu den einzelnen Biclustern zu gruppieren. Auch hier lässt sich pro Diagramm nur ein einziger Bicluster analysieren. Die Parallelen Koordinaten wurden mit folgender Anweisung erstellt:

$$parallelCoordinates(x, bic, number = 1, plotBoth = TRUE, compare = TRUE, bothLab = TRUE)$$

Abbildung 14(c) zeigt einen Bubble-Plot. In dieser Ansicht werden alle gefundenen Bicluster dargestellt. Es können bis zu drei Biclustermengen in einem Bubble-Plot dargestellt werden, wobei die Kreise (Bubbles) der ersten Biclustermenge in Grün, die der zweiten in Rot und die der dritten in Blau dargestellt werden. Dabei werden die Größe, Güte und Lage eines Biclusters durch visuelle Eigenschaften der Kreise ausgedrückt. Die Helligkeit eines Kreises drückt die Homogenität des entsprechenden Biclusters aus, die Größe eines Kreises gibt Aufschluss über die Größe des Biclusters und die Position des Mittelpunktes eines Kreises wird aus den Mittelwerten der Zeilen- und Spaltenindices eines Biclusters berechnet. Durch diese Darstellung erhält man einen Aufschluss über die gegenseitigen Abhängigkeiten der verschiedenen Bicluster. Auch kann diese Darstellung als Überblick über alle gefundenen Bicluster benutzt werden. Der Bubble-Plot wurde mit folgendem Aufruf erstellt:

$$bubbleplot(x, bic, showLabels = TRUE)$$

2.5.4 Visualisierung mit BicAT

BicAT ist eine Anwendung, die für die Untersuchung von biologischen Daten (z. B. Genexpressionsdaten) entwickelt wurde. Sie kann von der Webseite der ETH-Zürich [ETH10] heruntergeladen werden. *BicAT* ist im Gegensatz zu *R* keine auf Kommandozeilen basierende Anwendung, sondern kann mittels einer grafischen Benutzeroberfläche gesteuert werden. Die erstellten Screenshots basieren auf dem Datensatz, der im vorherigen Abschnitt für *Biclust* benutzt wurde.

Um die Screenshots aus *BicAT* zu erstellen, wurde zunächst der Datensatz geladen und anschließend ebenfalls der Bicluster-Algorithmus von CHENG und CHURCH ausgeführt. Die Parameter wurden wie folgt gesetzt: $\delta = 0.1$, $\alpha = 5$ und $number = 10$. Zusätzlich wurde der Saatwert (seed) auf 13 gesetzt. Die Screenshots sind in Abbildung 15 dargestellt, wobei sich (a) und (b) jeweils auf den zweiten gefundenen Bicluster beziehen (ID: 1).

Dem Benutzer werden alle gefundenen Bicluster in einer Liste (Abbildung 15(c)) angezeigt. Dadurch ist es dem Benutzer möglich durch die einzelnen Bicluster zu navigieren. Neben der ID werden zusätzlich die Größe und die Anzahl der Zeilen und Spalten des Biclusters angezeigt. Weiterhin besteht die Möglichkeit, alle berechneten Bicluster nach ihrer Zeilen- und Spaltenzugehörigkeit zu durchsuchen. Die passenden Bicluster werden dann im Zweig „Search Results“ angezeigt. Auch *BicAT* benutzt zur Visualisierung der Bicluster zum einen eine Heatmap und zum anderen eine Darstellung in Parallelen Koordinaten. Diese beiden Darstellungen passen sich im Gegensatz zu *Biclust* synchron dem aktuell ausgewählten Bicluster an. Eine Selektion der Bicluster innerhalb der Heatmap oder den Parallelen Koordinaten ist nicht möglich.

Auch in *BicAT* wird die Heatmap durch Zeilen- und Spaltenvertauschungen so verändert, dass der aktuell ausgewählte Bicluster oben links dargestellt ist (vgl. Abbildung 15a). Hierbei wird jedoch auf die zusätzliche Darstellung von anderen exklusiven Biclustern innerhalb der Heatmap verzichtet. Der Bicluster wird zusätzlich durch ein weißes umschließendes Rechteck hervorgehoben. Der Benutzer kann bei dieser Darstellung ebenfalls keinen Einfluss auf die Sortierung nehmen. Die Darstellung der Heatmap bietet – im Gegensatz zu der vom Paket *Biclust* – eine Scroll- und Zoom-Funktion. Somit kann der Benutzer Teile der Heatmap hervorheben, die im Fokus seines Interesses stehen. Zusätzlich kann sich der Benutzer durch Klicken auf eine Zelle den Zeilen- und Spaltennamen anzeigen lassen.

Wie bereits erwähnt entspricht die Darstellung in Parallelen Koordinaten (vgl. Abbildung 15(b)) dem aktuell ausgewählten Bicluster. Dabei werden alle Linien (Zeilen) die dem Bicluster nicht angehören ausgeblendet. Zusätzlich wird jedes Liniensegment einer Zeile in derselben Farbe eingefärbt. Durch die Einfärbung kann jede Linie gut von den anderen unterschieden werden. Auch in dieser Darstellung der Parallelen Koordinaten wird keine Transparenz verwendet, wodurch der Verlauf bei stark überlagernden Linien unkenntlich werden kann. Weiterhin werden bei Biclustern mit mehr als 25 Zeilen lediglich die ersten 25 Zeilen in den Parallelen Koordinaten dargestellt. Dies kann zu Fehlinterpretationen bei der Analyse der Bicluster führen.

In *BicAT* werden alle Spalten in den Parallelen Koordinaten dargestellt. Die Zugehörigkeit einzelner Spalten zu einem Bicluster wird dadurch hervorgehoben, dass die parallelen Achsen, die den Spalten eines Biclusters entsprechen, grau eingefärbt werden. Dadurch kann der komplette

Verlauf einer Zeile über alle Spalten verfolgt werden und die einzelnen Liniensegmente können dem Bicluster zugeordnet werden.

Auch hier wird lediglich ein Bicluster in den Parallelen Koordinaten dargestellt, wodurch ein Vergleich mehrerer Bicluster zueinander nicht möglich ist. Dies muss durch das Vergleichen mehrerer Darstellungen in Parallelen Koordinaten durchgeführt werden, wobei jede Darstellung genau ein Bicluster aufzeigt.

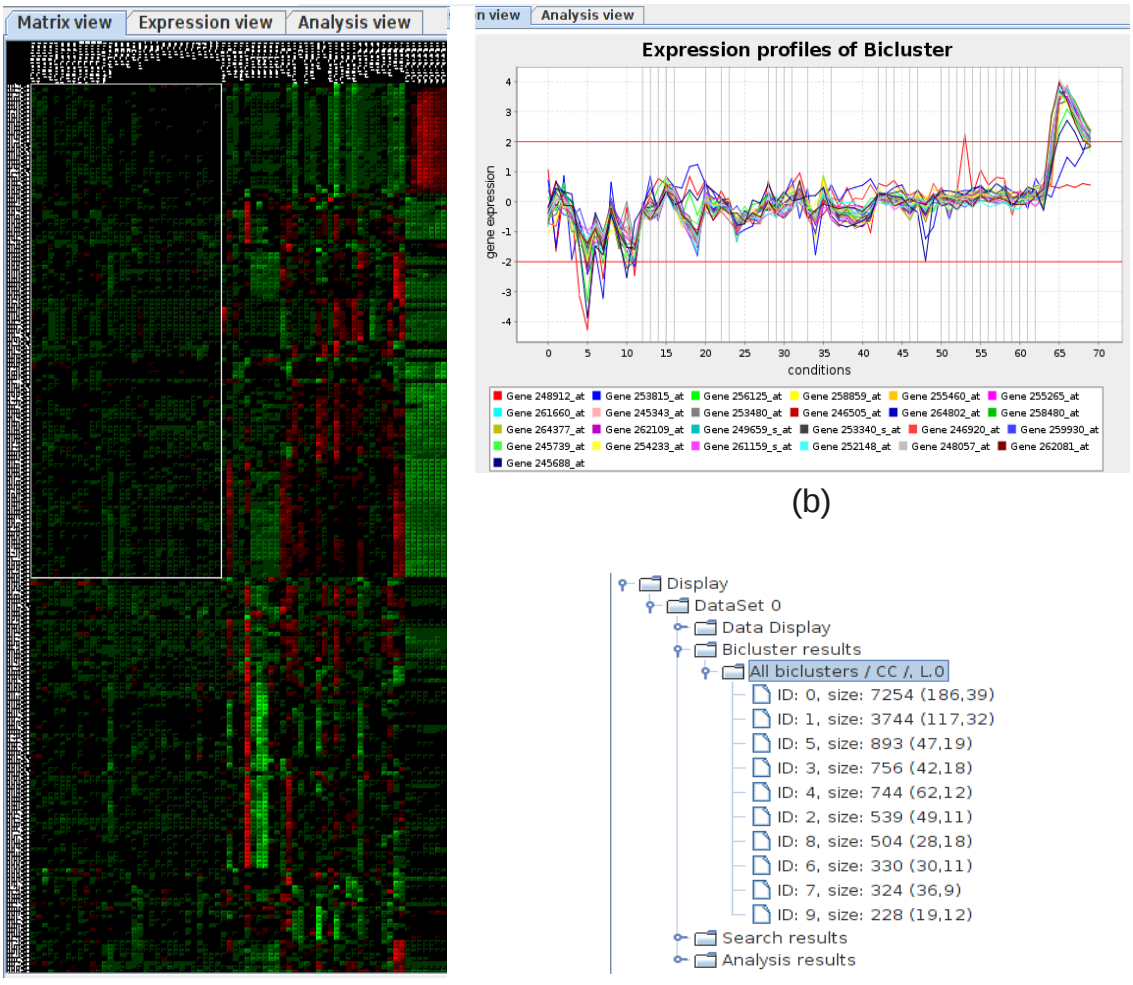


Abbildung 15: Bicluster-Visualisierung von *BicAT*
(a) Heatmap, (b) Parallele Koordinaten, (c) Navigationsliste
Quelle: Screenshots aus *BicAT* [ETH10]

2.5.5 Visualisierung mit BiVisu

BiVisu ähnelt in seiner Darstellung *BicAT*, da es ebenfalls Heatmaps und Parallele Koordinaten zur Visualisierung verwendet. *BiVisu* zielt jedoch nicht auf die Analyse biologischer Datensätze ab, sondern dient in erster Linie der Erkennung und der Darstellung von additiven und multiplikativen Biclustern. Die Anwendung ist in [Che07b] beschrieben und kann über [BiV10] bezogen werden. Die Ausführungen in diesem Abschnitt beziehen sich auf die auf der Web-Seite beschriebenen Beispiele und auf [Che07b].

Auch hier kann wieder ein Datensatz geladen werden. Der Benutzer kann anschließend auswählen, ob der Algorithmus additive oder multiplikative Bicluster berechnen soll. In beiden Fällen muss er zwei Parameter angeben, einen Rausch-Schwellwert und die minimale prozentuale Anzahl von Zeilen in einem Bicluster.

In jeder der in Abbildung 16 dargestellten Visualisierungen werden im linken Teil zusätzliche Informationen (Qualität, Index, Größe, Zeilen- und Spaltennamen) zu den Biclustern dargestellt. In allen Ansichten hat der Benutzer die Möglichkeit durch Zoomen und Scrollen bestimmte Teile genauer zu betrachten. Zusätzlich kann er im unteren Teil durch die einzelnen Bicluster navigieren. Die Darstellungen passen sich der aktuellen Auswahl des Biclusters an. Die hauptsächliche Visualisierung basiert auf den Parallelen Koordinaten (vgl. Abbildung 16a und b). Dabei werden jedoch nur die Spalten als parallele Achsen gezeichnet, die dem aktuell gewählten Bicluster angehören. Der Benutzer kann zwischen zwei verschiedenen Darstellungen wählen. In Abbildung 16a werden nur die Zeilen als Liniensegmente dargestellt, die dem Bicluster angehören (blau). Zusätzlich kann der Benutzer einzelne Zeilen und Spalten auswählen, die dann in der Darstellung hervorgehoben werden (rot). In Abbildung 16b werden ebenfalls nur die Spalten durch parallele Achsen dargestellt, die dem Bicluster angehören. Hier werden jedoch alle Zeilen gezeichnet und die dem Bicluster zugehörigen Zeilen werden in grün hervorgehoben. Der Benutzer kann einzelne Zeilen und Spalten auswählen und dadurch hervorheben. Es kann jedoch immer nur ein Bicluster gleichzeitig in der Darstellung der Parallelen Koordinaten angezeigt werden, wodurch eine Analyse der Bicluster zueinander nur mit Hilfe von mehreren Darstellungen möglich ist. Ebenfalls kann der Verlauf der Zeilen eines Biclusters über alle Spalten der Datenmatrix in den Parallelen Koordinaten nicht dargestellt werden, da nur die Spalten abgebildet werden, die einem Bicluster angehören.

BiVisu bietet zusätzlich eine Ansicht als Heatmap. Dabei werden die Zeilen und Spalten der Datenmatrix so vertauscht, dass der aktuell ausgewählte Bicluster oben links zusammenhängend dargestellt wird. Auch hier kann der Benutzer einzelne Zeilen und Spalten auswählen. Die Heatmap kann nur einen Bicluster darstellen. Somit ist auch hier eine Analyse von mehreren Biclustern in einer Darstellung nicht möglich.

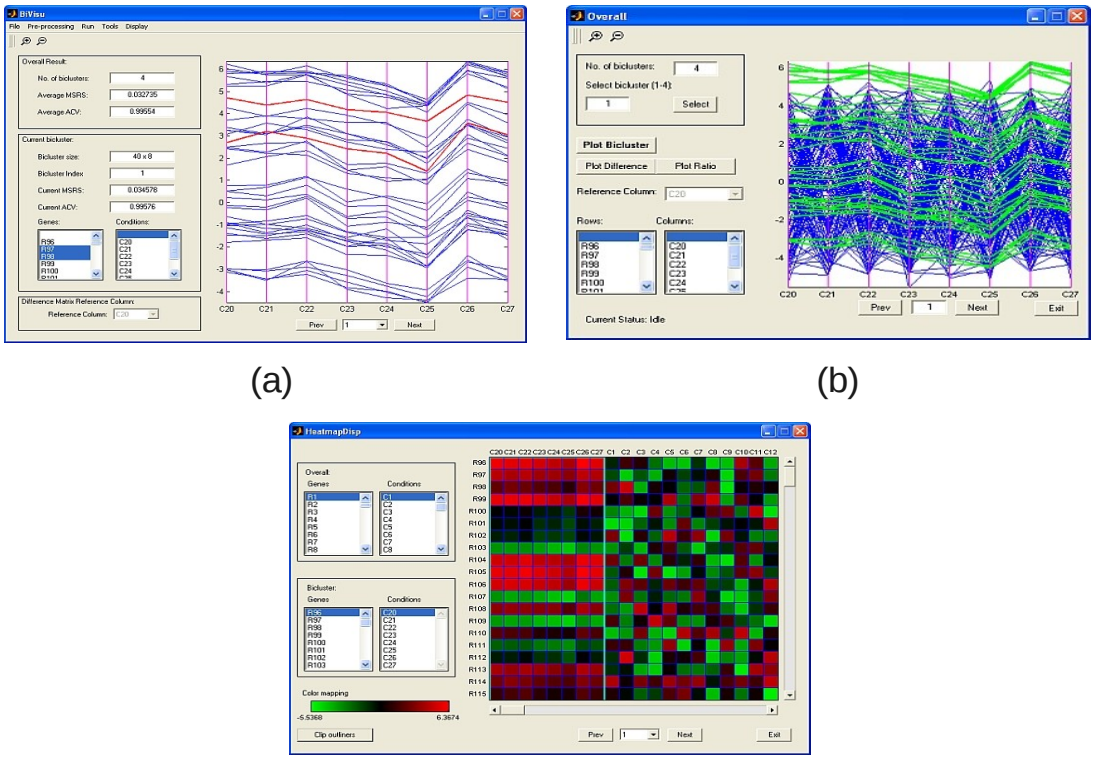


Abbildung 16: Screenshots aus *BiVisu*
(a) Anzeige eines Biclusters mit hervorgehobenen Zeilen, (b) Anzeige eines Biclusters (grün) mit allen Zeilen (blau) (c) Heatmap mit Bicluster oben links
Quelle: [BiV10]

2.5.6 Visualisierung mit BicOverlapper

BicOverlapper ist die bisher umfangreichste Anwendung für die Bicluster-Visualisierung. Sie ist auf die Analyse mehrerer Bicluster ausgelegt und bietet die Möglichkeit alle Bicluster gleichzeitig darzustellen. Neben den bereits beschriebenen Visualisierungsansätzen (Heatmap, Parallele Koordinaten und Bubble Plot) bietet sie außerdem einen bisher noch nicht beschriebenen Visualisierungsansatz: den *Overlapper*. Die Darstellungen werden in separaten Fenstern dem Benutzer präsentiert. Dieser Abschnitt fasst größtenteils die Ausführungen aus [San08] zusammen.

Einen großen Vorteil gegenüber den zuvor beschriebenen Ansätzen im Bezug auf die Interaktion bietet die Tatsache, dass alle Visualisierungen miteinander verlinkt sind. Das bedeutet, dass die Selektion eines Biclusters bzw. mehrerer Bicluster in einer Ansicht gleichzeitig ein Hervorheben des Biclusters bzw. der Bicluster in allen anderen Darstellungen bewirkt. Somit kann ein Bicluster-Ergebnis mit mehreren Biclustern besser analysiert werden. Die Selektion kann der Benutzer innerhalb der Darstellung bewirken und muss nicht in einer Navigationsliste nach dem entsprechenden Bicluster suchen.

Im Gegensatz zu den bisherigen Visualisierungen mit Heatmaps ist es mit *BicOverlapper* möglich, mehrere Zeilen bzw. Spalten durch Anklicken innerhalb der Heatmap zu selektieren. Aus Performance-Gründen wird jedoch nicht die komplette Heatmap gezeichnet. Es wird aber sichergestellt, dass selektierte Zeilen und Spalten – auch wenn die Selektion in anderen Ansichten erfolgt – oben links dargestellt werden. Unten und rechts werden dann zufällig weitere Zeilen und Spalten der Heatmap angefügt. Ist keine Spalte oder Zeile ausgewählt werden nur zufällig gewählte Zeilen und Spalten angezeigt. Einen weiteren Vorteil bietet die Anwendung dadurch, dass selektierte Zeilen und Spalten in allen Ansichten hervorgehoben werden. Z. B. bewirkt eine Selektion in den Parallelen Koordinaten eine Hervorhebung der entsprechenden Zeilen in der Heatmap.

Der Bubble-Plot bietet die Möglichkeit durch Zoomen und Verschieben der Ansicht bestimmte Teile des Plots genauer zu analysieren. Die Eigenschaften der Bicluster werden, wie bei *Biclust*, durch Größe, Farbe und Position visualisiert. Durch das generelle Verlinken der Ansichten un-

tereinander werden auch im Bubble-Plot automatisch die Bicluster hervorgehoben, die in anderen Darstellungen selektiert wurden.

Die Darstellung der Parallelen Koordinaten (vgl. Abbildung 17) bietet eine Neuerung im Vergleich zu den bisher vorgestellten Ansätzen. Es ist dem Benutzer möglich durch Verschieben der Pfeil-Icons einen Schwellwert zu setzen. Je nach Schwellwert werden dann alle Linien hervorgehoben, die innerhalb der angegebenen Schwellwerte verlaufen. Damit bietet *BicOverlapper* als einzige der untersuchten Anwendungen eine Interaktionsmöglichkeit innerhalb der Parallelen Koordinaten.

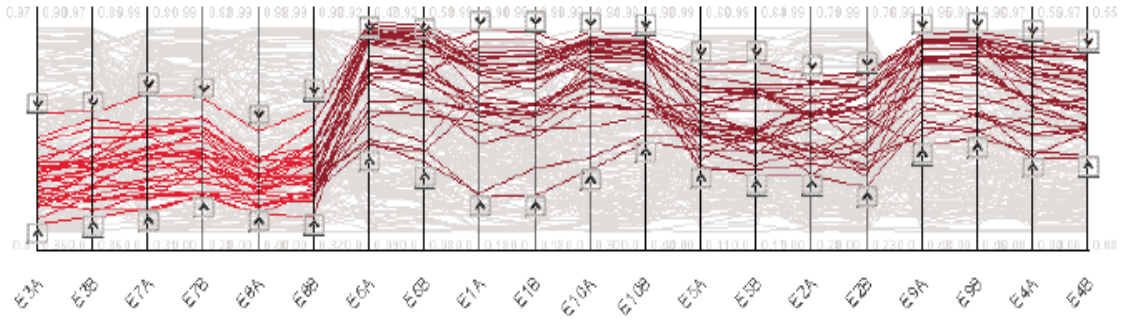


Abbildung 17: Parallele Koordinaten mit Interaktionsmöglichkeit aus *BicOverlapper*
Quelle: [San08]

Weiterhin zeigt die Darstellung in Abbildung 17 den Verlauf aller Linien über alle Spalten an. Bei einer Selektion eines Biclusters in einer der anderen Ansichten werden automatisch die entsprechenden Linien in den Parallelen Koordinaten in Rot hervorgehoben, die restlichen Linien werden in Grau gezeichnet. Die Schwellwerte werden entsprechend gesetzt. Die Spalten eines Biclusters kann man in dieser Darstellung ebenfalls gut erkennen, da die Achsen entsprechend der Spalten des Biclusters umsortiert werden. Sie werden im linken Teil der Darstellung angezeigt. Die Liniensegmente, die zu einem Bicluster gehören, werden in einem helleren Rot gezeichnet als die, die nicht dem Bicluster angehören. Bei einer Selektion von mehreren Biclustern fällt jedoch eine Zuordnung zu den einzelnen Biclustern schwer.

In Abbildung 18 ist das Prinzip des Overlappers dargestellt. Das gezeigte Beispiel bezieht sich auf Genexpressionsdaten, deshalb enthalten die Bicluster B_1 bis B_3 Gene (g_i) und Bedingungen (c_j). Der Overlapper präsentiert einen Graphen, in dem die Bicluster durch umschließende semitransparente Hüllen dargestellt werden. Die Gene und die Bedingungen bilden die Knoten des

Graphen, die Bicluster werden durch die Kanten abgebildet (vgl. Abbildung 18a). Ein Bicluster entspricht damit einer Menge von Kanten zwischen den entsprechenden Knoten. Die Anordnung der Knoten wird mittels eines kraftgerichteten Layouts bestimmt. Das Layout bewirkt, dass die Knoten, die mehreren Biclustern angehören, in der Mitte des Graphen angeordnet werden. Somit liegt der Knoten g_1 in der Mitte des Graphen, da er in allen Biclustern enthalten ist.

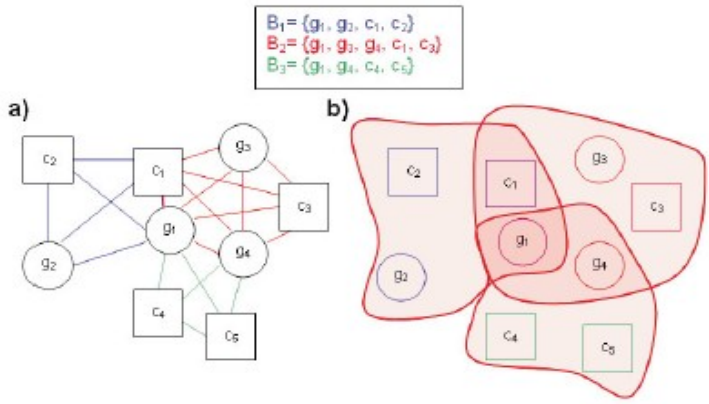


Abbildung 18: Overlapper Prinzip
a) Graph mit kompletter Kantenstruktur, Teilmengen entsprechen Biclustern
b) Kanten sind ausgeblendet, transparente Hüllen umschließen die Knoten eines Biclusters
Quelle: [San08]

Dem Benutzer werden in der späteren Darstellung die Kanten nicht angezeigt (vgl. Abbildung 18b). Die Bicluster werden durch umschließende Hüllen mit Transparenz dargestellt. Diese Hüllen werden um die am äußeren Rand liegenden Knoten des Biclusters gezeichnet. Durch die Transparenz wird unter anderem hervorgehoben, dass der Knoten g_1 zu mehr Biclustern gehört als die anderen Knoten. Durch diese Darstellung kann der Benutzer die Abhängigkeiten der verschiedenen Bicluster gut erkennen. Sie zeichnet sich vor allem dadurch aus, dass alle Bicluster visualisiert und die Überlappungen den einzelnen Zeilen (Gene) und Spalten (Bedingungen) zugeordnet werden können. Der *Overlapper* ist ebenfalls mit den anderen Darstellungen verbunden, wodurch sich eine Selektion einzelner Knoten bzw. Hüllen auf die anderen Darstellungen auswirkt.

3 Problemstellung und Lösungsansätze

3.1 Problemstellungen

Ziel dieser Diplomarbeit ist es, eine Anwendung zu entwickeln, welche in der Lage ist, Bicluster-Ergebnisse auf Genexpressionsdaten zu visualisieren. Vor allem Heatmaps und Parallele Koordinaten sollen so erweitert bzw. verändert werden, dass sie mehrere Bicluster anzeigen können. In Abschnitt 2.5 auf Seite 28 wurden die bisherigen Ansätze beschrieben. Dabei stellte sich heraus, dass vor allem die gleichzeitige Visualisierung von mehreren Biclustern Probleme verursacht und bisher nur von *BicOverlapper* eine gute Visualisierung für überlappende Bicluster gefunden wurde. *BicOverlapper* benutzt zur Darstellung der Bicluster jedoch Graphen, welche in dieser Arbeit nicht genauer untersucht werden.

3.1.1 Berechnung der Bicluster

Es soll ein möglichst generischer Algorithmus gefunden werden, der die Berechnung der Bicluster durchführt. Die Visualisierung soll unabhängig von dem gewählten Algorithmus durchgeführt werden können. Der Benutzer soll die Möglichkeit haben, den Algorithmus beliebig oft parametrisiert starten zu können. Zusätzlich soll der Benutzer die Parameter während der Ausführung der Anwendung ändern können. Dies ist vor allem dann wichtig, wenn er während der Analyse der Bicluster feststellt, dass die initial gewählten Parameter zu keinem befriedigenden Ergebnis geführt haben.

Generell kann gesagt werden, dass der „perfekte“ Bicluster-Algorithmus bisher noch nicht entwickelt wurde und es fraglich ist, ob er jemals entdeckt wird. Deshalb muss es möglich sein, unterschiedliche bzw. neue Bicluster-Algorithmen der Anwendung leicht hinzuzufügen. Daher

muss ein Mechanismus entwickelt werden, der unabhängig von der Visualisierung die Biclusterberechnung übernimmt. Das Einbinden neuer Bicluster-Algorithmen sollte keine erneute Kompilierung der gesamten Anwendung erfordern. Die Schnittstelle zwischen der Anwendung und dem Bicluster-Algorithmus muss daher sehr variabel aufgebaut sein und eine Möglichkeit der Parameterübergabe bereitstellen.

3.1.2 Heatmaps

Heatmaps eignen sich besonders gut, um die Gesamtheit der Daten aufzuzeigen. Zusätzlich bieten sie eine Übersicht darüber, wo – das bedeutet in welcher Zeile und Spalte – entsprechende Datenpunkte innerhalb der Datenmatrix zu finden sind. Die bisherigen betrachteten Ansätze zur Visualisierung von Biclustern auf einer Datenmatrix ermöglichen es nur exklusive Bicluster anzuzeigen. Dies liegt vor allem daran, dass es bei entsprechender Lage von überlappenden Biclustern nicht möglich ist, die Bicluster durch alleiniges Vertauschen von Zeilen und Spalten zusammenhängend darzustellen. Diese Tatsache wird an einem Beispiel erläutert.

Sei D eine (10 x 11) große Datenmatrix mit den Zeilenindices I und den Spaltenindices J:

$$I = \{0,1,2,3,4,5,6,7,8,9\} \text{ und } J = \{0,1,2,3,4,5,6,7,8,9,10\}$$

Weiterhin seien auf den Mengen I und J die drei Bicluster A, B, und C definiert, mit:

$$I_A = \{2,7,8\}, J_A = \{1,3,8,9\}$$

$$I_B = \{1,2,5,7\}, J_B = \{1,3,5,10\}$$

$$I_C = \{1,3,6,9\}, J_C = \{3,5,6,7,8\}$$

Die Bicluster A und B haben eine Überlappung in den Zeilen 2 und 7 und den Spalten 1 und 3. A und C haben die Spalten 3 und 8 gemeinsam. B und C haben eine Überlappung in der Zeile 1 und den Spalten 3 und 5.

Im Folgenden soll versucht werden, die Bicluster A, B und C zusammenhängend in eine Tabelle einzufügen, wobei Zeilen und Spalten nur vertauscht werden dürfen. Dabei werden die Bicluster in folgenden Farben dargestellt: A rot, B grün und C blau. Abbildung 19 zeigt ein Beispiel, bei dem die Bicluster A, B und C nicht ohne Verdopplung zusammenhängend dargestellt werden können.

In Abbildung 19a) ist dargestellt, wie die Bicluster A und B liegen könnten, so dass keine Zeilen und Spalten verdoppelt werden müssen. A und B überlappen sich in den Zeilen 2 und 7 und den Spalten 1 und 3, diese sind in der Mischfarbe von Rot und Grün (also Gelb) dargestellt. Die gegenseitige Lage von A und B ist fest vorgegeben, da sie sich Zeilen und Spalten teilen. Lediglich die Position der Überlappung ist variabel, das bedeutet, dass B auch links unten, rechts unten oder rechts oben an A eingefügt werden kann. Auch müssen die beiden Bicluster nicht mittig in der Tabelle liegen. Die beiden Bicluster müssen sich jedoch an ihren Rändern überlappen, da sie sonst nicht zusammenhängend dargestellt werden könnten oder Zeilen bzw. Spalten verdoppelt werden müssten.

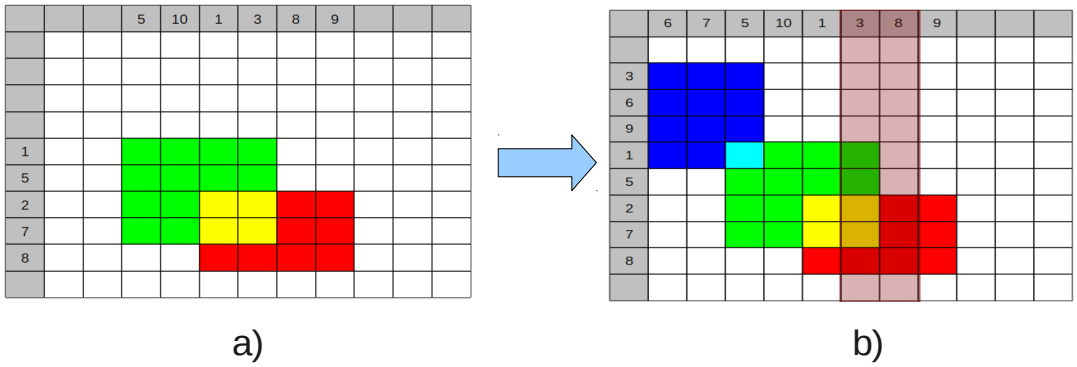


Abbildung 19: Beispiel für das Einfügen ohne Verdopplung
a) A und B sind in die Tabelle eingefügt, die Überlappung ist in Gelb dargestellt.
b) C wird in die Tabelle eingefügt, kann jedoch ohne Verdopplung nicht zusammenhängend dargestellt werden; die dazu benötigten Zeilen sind in transparentem Rot hervorgehoben

An welcher Position die Überlappung stattfindet ist jedoch im Allgemeinen nicht von Bedeutung. Alle folgenden Ausführungen können ebenfalls für eine der anderen Positionierungen begründet werden.

Zunächst wird festgehalten, dass die Spalten 1 und 3 immer nebeneinander auftreten müssen, da sich A und B in diesen Spalten überlappen. Es sind lediglich Vertauschungen dieser Abfolge erlaubt, also (1,3) und (3,1). Wird diese Abfolge von Spalten nicht eingehalten, so können die Bicluster A und B nicht zusammenhängend ohne Verdopplung dargestellt werden. Diese Tatsache wird in Aussage (I) allgemein formuliert:

> **Aussage (I)**: *Werden die Bicluster A mit den Spaltenindices $J_A = \{a_1, a_2, \ldots a_m\}$ und B mit den Spaltenindices $J_B = \{b_1, b_2, \ldots, b_n\}$ zusammenhängend ohne Verdopplung dargestellt und haben A und B die Spaltenindices $J_C = \{c_1, c_2, \ldots, c_k\}$ gemeinsam (d. h. $J_C = J_A \cap J_B$), dann muss jede Abfolge von Spaltenindices die Teilfolge $(\pi(c_1), \pi(c_2), \ldots, \pi(c_k))$ besitzen, wobei π eine beliebige Permutation auf J_C beschreibt, mit $\pi : J_C \rightarrow J_C$.*

In Abbildung 19b) wird nun Bicluster C eingefügt. Sie zeigt die Anordnung von A, B und C, bei der am wenigsten Spalten verdoppelt werden müssen. C enthält die Spalten 3, 5, 6, 7 und 8. Die folgende Argumentation bezieht sich auf die Spalten 3, 5 und 8. Es wird gezeigt, dass diese drei Spalten nicht zusammenhängend dargestellt werden können ohne dass Aussage (I) widersprochen oder ein anderer Bicluster nicht zusammenhängend dargestellt wird. Zusammenhängend bedeutet bei Bicluster C, dass zwischen diesen Spalten lediglich die Spalten 6 und 7 auftreten dürfen. Der Widerspruch wird einmal für die Teilfolge (1,3) und anschließend für (3,1) hergeleitet.

In Abbildung 19a) sind bereits die Spalten 3 und 8 nebeneinander dargestellt. In diese Abfolge dürfen nur die Spalten 5, 6 und 7 zwischen 3 und 8 eingefügt werden. Rechts an Spalte 8 kann nur Spalte 9 angefügt werden, da sie die letzte bisher noch nicht beachtete Spalten in Bicluster A ist und nicht zwischen 3 und 8 eingefügt werden darf. Links neben Spalte 3 muss nach Aussage (I) Spalte 1 vorkommen. Dies ergibt die Teilfolge (1,3,8,9) und somit können die Spalten 3, 5 und 8 nicht zusammenhängend dargestellt werden.

Es wird nun angenommen, dass in Abbildung 19a) nicht die Abfolge (1,3) sondern (3,1) bei der Überlappung von A und B auftritt. Die Darstellung bleibt gleich, es werden lediglich die Spaltenindices vertauscht. An Spalte 3 kann links Spalte 5 angefügt werden. Dies ergibt die Teilfolge (5,3,1). Links neben Spalte 5 kann jedoch nur noch Spalte 10 angefügt werden, da dies die letzte noch nicht beachtete Spalte in Bicluster B ist. Es ergibt sich die Teilfolge (10,5,3,1) und somit können auch in diesem Fall die Spalten 3, 5 und 8 nicht zusammenhängend dargestellt werden.

Genau zwei überlappende Bicluster lassen sich immer zusammenhängend ohne Verdopplung darstellen. Bei drei überlappenden Biclustern kann im Allgemeinen nicht garantiert werden, dass die Bicluster ohne Verdopplung zusammenhängend dargestellt werden können. Das gleiche Problem kann auch bei exklusiven Zeilen- oder Spalten-Biclustern (vgl. Abschnitt 2.2.1) auftreten.

3.1.3 Parallele Koordinaten

Eines der Hauptprobleme der Parallelen Koordinaten ist die Überzeichnung bei großen Datensätzen. Dadurch kann man unter Umständen die einzelnen Linien nicht immer anhand ihres Verlaufs zu einem bestimmten Bicluster zuordnen. Da ein Bicluster – im Gegensatz zu einem Cluster – nicht über alle Spalten verläuft, ist es ebenfalls sehr schwer die Spalten (parallele Achsen) einem bestimmten Bicluster zuzuordnen.

Die Untersuchung aus Kapitel 2 hat ergeben, dass es bisher keinen Ansatz gibt, mehrere Bicluster in einer Darstellung der Parallelen Koordinaten zu untersuchen. Vor allem das Gegenüberstellen von mehreren Biclustern in Parallelen Koordinaten ist ein bisher ungelöstes Problem. Zwar werden viele Ansätze für die Darstellung eines Biclusters aufgezeigt, jedoch sind diese meist nur in sehr beschränkter Weise für mehrere Bicluster praktikabel.

Durch Einfärben der entsprechenden Liniensegmente können Linien einem Bicluster zugeordnet werden. Werden jedoch viele Bicluster in einer Darstellung der Parallelen Koordinaten angezeigt, wird eine genaue Zuordnung der einzelnen Farben zu den Biclustern schwierig. Dies liegt zum einen daran, dass eine signifikant unterscheidende Farbgebung meist für nur 15 Bicluster erkennbar ist, zum anderen können die Farben durch starke Überlagerung der Linien unkenntlich werden.

Auch das Ausblenden von Linien und Achsen, die einem Bicluster nicht angehören, ist ebenfalls nur für die Darstellung eines Biclusters geeignet. Bei mehreren Biclustern ist fraglich, ob die angezeigten Linien und Achsen den einzelnen Biclustern wieder zugeordnet werden können. Lediglich das Ausblenden der Linien ist ein guter Ansatz dem Überzeichnen entgegenzuwirken.

Eine weitere wichtige Anforderung an die Darstellung besteht darin, einzelne Achsen (Spalten) den Biclustern zuzuordnen. Auch für diese Aufgabe eignen sich das Einfärben und Ausblenden nur sehr beschränkt. Da dieselbe Achse mehreren Biclustern angehören kann, ermöglicht das Einfärben bzw. Ausblenden keine eindeutige Zuordnung zu den einzelnen Biclustern. Vor allem den Zusammenhang zwischen den einzelnen Achsen und Linien zu verdeutlichen ist ein bisher ungelöstes Problem innerhalb der Darstellung der Parallelen Koordinaten.

3.2 Lösungsansätze, Implementierung und Architektur

In diesem Abschnitt wird ein Überblick über die in dieser Diplomarbeit entstandene Anwendung geben, sie wird im Folgenden als *Bicluster Viewer* bezeichnet. Es wird grob die Architektur der Anwendung beschreiben und die einzelnen Lösungsansätze werden kurz erläutern. Für eine genauere Erklärung der Ansätze wird auf die entsprechenden Abschnitte verweisen. Zusätzlich geht dieser Abschnitt genauer auf das in *Bicluster Viewer* verwendete Datenmodell ein.

Der *Bicluster Viewer* besteht hauptsächlich aus drei Visualisierungsmodulen: die Heatmap, die Parallelen Koordinaten und der Informationsrahmen. Alle Module sind gegen das Datenmodell gelinkt, wodurch eine Selektion in einem der Module automatisch in den anderen Modulen registriert und angezeigt wird. Abbildung 20 zeigt einen Screenshot von *Bicluster Viewer*. Für diesen Screenshot wurden der Yeast Datensatz und die von CHENG und CHURCH gefundenen Bicluster geladen [Har10]. Einige der Bicluster wurden exemplarisch hervorgehoben.

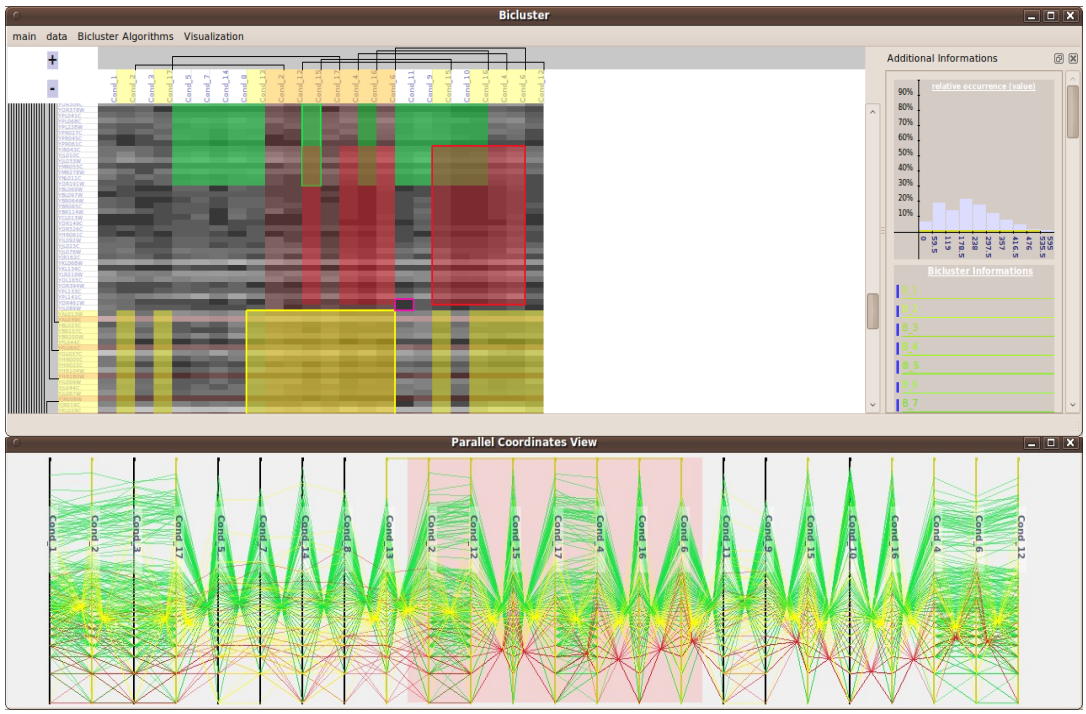


Abbildung 20: Screenshot aus *Bicluster Viewer*
Angezeigt wird der Yeast Datensatz und die von CHENG und CHURCH darauf gefundenen Bicluster.

Die Datenwerte werden durch ein *Matrix*-Objekt repräsentiert. Diese *Matrix*-Klasse wird sowohl von den Visualisierungsmodulen als auch von dem Berechnungsmodul verwendet. Sie speichert die Datenwerte und die Zeilen- und Spaltennamen. Die Werte und Namen können dann durch Angabe der entsprechenden Zeilen- und Spaltenindices wieder ausgelesen werden. Ein *Matrix*-Objekt wird im *Bicluster Viewer* durch das Einlesen einer Datei erstellt. Beim Einlesen kann der Benutzer festlegen, ob die Datei Zeilen- und Spaltennamen enthält und mit welchem Symbol die Datenwerte voneinander getrennt werden. Nach dem Einlesen der Datenmatrix werden Referenzen zu dem erstellten *Matrix*-Objekt allen Visualisierungsmodulen bekannt gemacht.

Die Berechnung der Bicluster soll möglichst variabel erfolgen, weshalb ein Mechanismus entwickelt wurde, der die Bicluster-Algorithmen erst zur Laufzeit in das Programm einbindet. Der Benutzer hat die Möglichkeit die Parameter des Algorithmus zu ändern und die Berechnung erneut zu starten. Eine genauere Erklärung des Mechanismus ist in Abschnitt 3.3 auf Seite 54 gegeben.

Bicluster werden in einer Liste bestehend aus *BiclusterVisualModel*-Objekten gespeichert. Die Liste wird durch ein *BiclusterList*-Objekt repräsentiert. Diese Klasse übernimmt die Verwaltung der Bicluster-Objekte. Deshalb muss allen Visualisierungsmodulen nur eine Referenz auf das *BiclusterList*-Objekt bekannt gemacht werden. Weiterhin übernimmt die Klasse die Bekanntmachung neuer bzw. sich ändernder Selektionen. Sie ist nach dem Listener-Pattern implementiert, daher müssen sich alle Module, die diesen Selektionsmechanismus verwenden, bei dem *BiclusterList*-Objekt anmelden. Eine genauere Beschreibung der Datenmodelle wird in Abschnitt 3.2.1 auf Seite 51 gegeben.

Abbildung 21 zeigt schematisch den hierarchischen Aufbau des *Bicluster Viewers*. Die drei Visualisierungsmodule sind direkt mit dem Hauptmodul (abgeleitet aus der *Qt* Klasse *QMainWindow*) verbunden. Die Bekanntmachung neuer Daten- oder Bicluster-Objekte erfolgt von der Wurzel in Richtung der Blätter. Die Visualisierung wird von den in den Blättern positionierten Modulen durchgeführt. Dabei übermitteln die Module *Heatmap*, *Parallele Koordinaten* und *Information Dock* Änderungen, die der Benutzer in den Dialogen des Hauptmoduls vornimmt. So werden z. B. Änderungen bei der Sortierung oder bei den Farben der Bicluster an die in den Blättern positionierten Module gemeldet.

Durch das in der *BiclusterList*-Klasse implementierte Listener-Pattern, muss bei einer Änderung der selektierten Bicluster nicht der komplette Baum durchlaufen werden. Die Module in den Blättern melden sich direkt bei dem *BiclusterList*-Objekt an und werden somit direkt über Selektionsänderungen informiert.

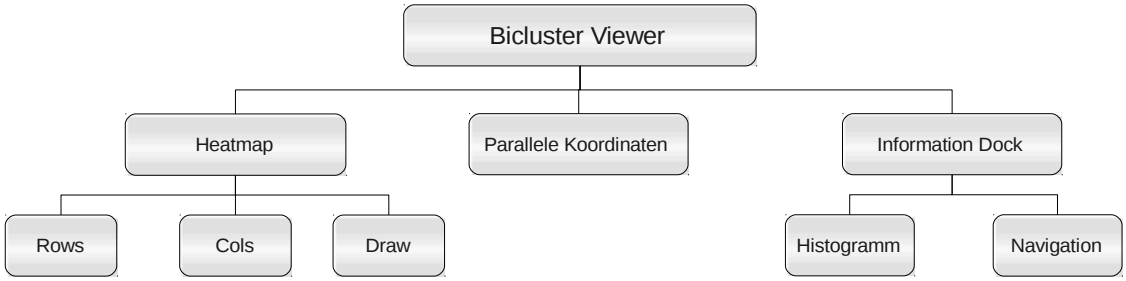


Abbildung 21: Hierarchischer Aufbau von *Bicluster Viewer*

Die Heatmap dient dem Benutzer als Übersicht über alle gefundenen Bicluster. Dabei wird versucht die Bicluster möglichst zusammenhängend ohne Verdopplung zu zeichnen. Dies ist jedoch im Allgemeinen nicht immer möglich (vgl. Abschnitt 3.1.2). Um diesem Problem entgegenzuwirken werden die Bicluster nach einer bestimmten Reihenfolge in die Heatmap eingefügt. Falls nötig werden verdoppelte Zeilen und Spalten der Heatmap hinzugefügt. Bicluster werden innerhalb der Heatmap durch farbige Rechtecke gekennzeichnet. Die verdoppelten Zeilen und Spalten werden zunächst ausgeblendet, da die Darstellung sonst sehr unübersichtlich wird. Weiterhin kann es bei der automatischen Anordnung der Bicluster dazu kommen, dass Bicluster, die der Benutzer untersuchen möchte, nicht zusammenhängend dargestellt werden. Deshalb ist die Anwendung vor allem auf die Benutzer-Interaktion ausgerichtet. Der Benutzer kann die Reihenfolge festlegen, in der die Bicluster in die Heatmap eingefügt werden, kann bestimmen welcher Bicluster durch Verdopplung angezeigt werden soll und welche Bicluster ständig hervorgehoben werden. Eine genauere Beschreibung der Funktionalitäten der Heatmap und der für die Darstellung entwickelten Algorithmen erfolgt in Abschnitt 3.4 auf Seite 57.

Da die Darstellung der Bicluster innerhalb der Heatmap bei vielen überlappenden Biclustern eine eindeutige Zuordnung zu den einzelnen Biclustern oft unmöglich macht, enthält der *Bicluster Viewer* das *Information Dock*. In dieser Ansicht gibt es eine Navigationsliste aller berechneten Bicluster (siehe Abbildung 22a). Der Benutzer kann Bicluster durch Anklicken der entsprechenden Bicluster-ID selektieren. Selektierte Bicluster werden gelb hinterlegt. Somit erhält er

Aufschluss über die ID des gerade in der Heatmap selektieren Biclusters. Weiterhin kann er in der Navigationsliste einen Dialog öffnen, in dem er die Farbe und ID des Biclusters einstellen kann und in dem die Größe des Biclusters angezeigt wird. Neben den Bicluster-IDs werden in der Navigationsliste blaue Säulen gezeichnet. Diese stellen die relative Qualität der Bicluster dar. Eine große Säule verdeutlicht, dass die Qualität dieses Biclusters besser ist als die eines Biclusters mit einer kleinen Säule. Somit erkennt der Benutzer auf einen Blick, welche Bicluster von hoher Qualität sind. Die exakte Qualität eines Biclusters wird dem Benutzer als ToolTip angezeigt, wenn sich die Maus über der entsprechenden ID befindet. In Abbildung 22a erkennt man durch die Säulendarstellung leicht, dass Bicluster B_4 eine bessere Qualität als Bicluster B_2 hat.

Der Benutzer hat zusätzlich die Möglichkeit die Navigationsliste entsprechend der Qualität oder der IDs der einzelnen Bicluster zu sortieren. Hiermit kann auch die Heatmap neu sortiert werden. Die Bicluster werden dann entsprechend der Sortierung der Navigationsliste in die Heatmap eingefügt.

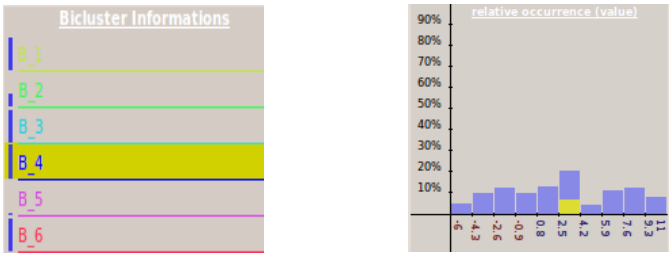


Abbildung 22: Navigationsliste und Histogramm im *Bicluster Viewer*
Bicluster B_4 ist in beiden Darstellungen hervorgehoben.

Viele Bicluster-Algorithmen erwarten Parameter, die sich nach den absoluten Datenwerten in der Datenmatrix richten. In der Heatmap werden jedoch die Datenwerte durch Grautöne repräsentiert, wodurch eine Zuordnung zu den absoluten Datenwerten nicht mehr gewährleistet ist. Deshalb wird zusätzlich ein Histogramm im *Information Dock* angezeigt. Das Histogramm unterteilt den kompletten Wertebereich in gleich große Intervalle, deren Anzahl vom Benutzer festgelegt werden kann. Es zeigt dann die relative Häufigkeit der Datenwerte entsprechend ihrer Zugehörigkeit zu den Intervallen an. Zusätzlich werden entsprechend der Datenwerte innerhalb der

selektierten Bicluster die Säulen mit Gelb markiert. Die Höhe der Markierung entspricht ebenfalls der relativen Häufigkeit. Abbildung 22b zeigt ein solches Histogramm.

Auch der Verlauf der Zeilen innerhalb der Heatmap ist meist nur schwer erkennbar. Deshalb besitzt der *Bicluster Viewer* zusätzlich eine Darstellung in Parallelen Koordinaten. Auch dieses Modul ist an das *BiclusterList*-Objekt angemeldet und erfährt somit ebenfalls von Änderungen in der Selektion. Die Linien sowie die parallelen Achsen, die dem aktuell selektierten Bicluster angehören, werden in Gelb hervorgehoben. Zusätzlich hat der Benutzer die Möglichkeit Bicluster dauerhaft hervorzuheben[12]. Ein dauerhaft hervorgehobener Bicluster wird in den Parallelen Koordinaten dargestellt, indem alle Linien, die dem Bicluster angehören, in der Farbe des Bicluster gezeichnet werden. Eine zusätzliche Hervorhebung wird durch das späte Zeichnen der Linien eines hervorgehobenen Biclusters bewirkt. Die Linien der selektierten Bicluster werden zuletzt dargestellt. Zusätzlich werden zwischen den Achsen, die einem Bicluster angehören, sogenannte Zentroide gezeichnet, wodurch Linienverläufe den einzelnen Achsen besser zugeordnet werden können. Außerdem werden die Achsen der Parallelen Koordinaten entsprechend den Spalten der Heatmap sortiert und verdoppelt. Somit kann der Benutzer auch Bicluster innerhalb der Parallelen Koordinaten zusammenhängend darstellen lassen. Eine genauere Beschreibung der Parallelen Koordinaten wird in Abschnitt 3.5 ab Seite 79 gegeben.

3.2.1 Das Datenmodell

Bei der kompletten Implementierung wurde darauf geachtet das MVC-Modell möglichst vollständig umzusetzen. Deshalb besitzen alle Visualisierungmodule nur Referenzen auf dieselben Objekte. Somit wird eine Änderung in den Daten-Objekten in allen Visualisierungmodulen sichtbar. Die Daten werden im Wesentlichen in drei Klassen gehalten: *Matrix*, *BiclusterList* und *IndexMapping*. Diese Klassen werden im Folgenden beschrieben.

- **Matrix**: Diese Klasse dient zur Speicherung der Datenwerte. Sie speichert alle *double* Werte in einem eindimensionalen Array sowie die Zeilen und Spaltennamen in jeweils einer Liste. Die Visualisierungsmodule greifen nur in lesender Weise darauf zu. Für die komplette Visualisierung wird nur ein Objekt dieser Klasse verwendet. Allen Visuali-

12 Dauerhaft hervorgehobene Bicluster werden in der Heatmap dargestellt, indem alle dem Bicluster zugehörigen Zellen in der Farbe des Biclusters transparent ausgefüllt werden.

sierungsmodulen wird eine Referenz auf das Objekt bekanntgemacht. Ein Update der Referenz wird nur vorgenommen, wenn ein neuer Datensatz geladen wird. Zusätzlich wird das *Matrix*-Objekt benutzt, um dem auszuführenden Bicluster-Algorithmus die Datenmatrix zu übergeben.

- **BiclusterList**: Diese Klasse speichert alle berechneten Bicluster in einer Liste. Die Bicluster werden durch die Klasse *BiclusterVisualModel* repräsentiert. Ein Bicluster wird mit allen Attributen (ID, Zeilen- und Spaltenindices, Qualität und Farbe) in einem *BiclusterVisualModel*-Objekt gekapselt. Zusätzlich kann aus dem Objekt ausgelesen werden, ob der Bicluster gerade selektiert ist, ob er zusammenhängend dargestellt werden kann, ob er immer hervorgehoben werden soll, ob er gerade mit Hilfe verdoppelter Zeilen und Spalten dargestellt ist und ob bzw. wie er mit anderen Biclustern in Konflikt steht. Das in allen Visualisierungsmodulen bekanntgemachte *BiclusterList*-Objekt übernimmt die Verwaltung dieser Objekte. An das Objekt können sich die Visualisierungsmodule als „Listener“ anmelden und werden somit über alle Selektionsänderungen der *BiclusterVisualModel*-Objekte informiert. Der Informationsaustausch wird über das *Qt* typische Signal-Slot-Prinzip durchgeführt, weshalb alle „Listener“ einen öffentlichen Slot implementieren müssen: *SLOT_selectionChanged*. Zusätzlich können aus der Liste Bicluster durch Angabe der ID und des Zeilen- oder Spaltenindex ausgelesen werden.

- **IndexMapping**: Die Zeilen und Spalten der Datenmatrix sollen beliebig vertauscht werden können. Die Vertauschungen werden nicht auf der Datenmatrix (*Matrix*-Objekt) durchgeführt, sondern werden durch zwei surjektive Abbildungen m_r und m_c realisiert, mit $m_r : \tilde{X} \to \tilde{X}$ und $m_c : \tilde{Y} \to \tilde{Y}$ (vgl. Abschnitt 2.2.1 auf Seite 14). Auch hier wird allen Visualisierungsmodulen für die gesamte Visualisierung nur ein Objekt bekannt gemacht. Ein *IndexMapping*-Objekt speichert nicht nur die aktuelle Sortierung der Indexmengen, sondern auch an welchen Stellen verdoppelte Zeilen bzw. Spalten vorkommen. Somit können die einzelnen Visualisierungsmodule über das Aufrufen von öffentlichen Methoden steuern, welche Zeilen und Spalten verdoppelt dargestellt werden. Weiterhin können sie abfragen, welche Zeilen und Spalten bereits verdoppelt dargestellt sind. Um die Visualisierung zu beschleunigen, wird die aktuelle Sortierung (mit Verdopplung) der Zeilen und Spalten zwischengespeichert. Zusätzlich wird zu jedem Bicluster seine

„Startposition“ in der aktuellen Sortierung vermerkt. Eine genaue Beschreibung zum Aufbau und der Erstellung der Indexstruktur wird in Abschnitt 3.4 ab Seite 57 gegeben.

3.3 Realisierung der Bicluster-Berechnung

Damit die Berechnung der Bicluster möglichst variabel durchgeführt werden kann, wurde ein Mechanismus entwickelt, der es dem Benutzer erlaubt, den Bicluster-Algorithmus erst zur Laufzeit festzulegen. Zusätzlich kann er den Algorithmus mit beliebigen Parametern starten. Weiterhin hat er damit die Möglichkeit den Bicluster-Algorithmus zur Laufzeit beliebig auszutauschen. In Anhang B wird an einem Beispiel beschrieben, wie neue Bicluster-Algorithmen erstellt werden können.

Der Mechanismus basiert darauf, dynamisch zur Laufzeit sogenannte „shared object“ Dateien (so-Datei) zu binden. Eine konkrete Implementierung eines Bicluster-Algorithmus ist demnach eine einzelne so-Datei. Damit die Anwendung mit dem Algorithmus kommunizieren kann, muss dieser dafür ein fest definiertes Interface implementieren. Das Interface wird durch die abstrakte Klasse *AbstractBiclusterAlgorithm* definiert. Für die Verwendung von *R*-Methoden bei der Biclusterberechnung wurde speziell die abstrakte Klasse *AbstractRBiclusterAlgorithm* definiert. Sie implementiert Methoden, mit denen Daten zwischen *R* und dem Bicluster-Algorithmus ausgetauscht werden können. Zusätzlich beinhaltet sie Methoden mit denen Kommandos in *R* ausgeführt werden können. Sie erbt von der Klasse *AbstractBiclusterAlgorithm* und definiert damit dasselbe Interface wie die Super-Klasse.

Im Folgenden wird eine Auflistung der wichtigsten Methoden gegeben:

- **compute**: Diese Methode erwartet als Übergabeparameter eine Datenmatrix und eine Parameterliste. Sie dient zur Berechnung der Bicluster, die als Liste aus Bicluster-Objekten zurückgegeben werden.
- **getParameterDef**: Mit dieser Methode definiert ein Bicluster-Algorithmus, welche Parameter er erwartet.
- **quality**: Diese Methode berechnet die Qualität eines bestimmten Biclusters. Als Parameter erwartet sie einen Bicluster für den die Qualität bestimmt werden soll. Der Rückgabewert ist ein *double*-Wert $q \geq 0$, es gilt: Je kleiner q, desto besser ist die Qualität eines Biclusters. Ein perfekter Bicluster hat damit die Qualität 0.

Die Berechnung der Bicluster soll aus der Anwendung heraus parametrisiert gestartet werden können. Deshalb besitzt das Interface die *compute*-Methode, die als Übergabeparameter zum einen die Datenmatrix und zum anderen eine Parameterliste erwartet. Die Parameterliste besteht aus Schlüssel-Wert Paaren, wodurch die Namen der Parameter beliebig in einer konkreten Implementierung eines Bicluster-Algorithmus definiert werden können. Als Rückgabewert liefert die Methode einen Vektor aus *BiclustModel*-Objekten zurück. Diese Objekte werden dann von der Anwendung in die *BiclusterVisualModel*-Objekte übersetzt. Die Entscheidung, für Bicluster zwei unterschiedliche Klassen zu definieren, wurde deshalb getroffen, da eine Implementierung eines Bicluster-Algorithmus unabhängig von den in der Visualisierung verwendeten *Qt*-Bibliotheken sein soll. Die *BiclusterVisualModel*-Objekte benutzen Klassen aus den *Qt*-Bibliotheken. Würde also die *compute*-Methode *BiclusterVisualModel*-Objekte zurückgeben, müssten *Qt*-Bibliotheken bei der Implementierung eines Bicluster-Algorithmus eingebunden werden.

Der Verfasser einer konkreten Implementierung soll möglichst viele Freiheiten bei der Wahl der Parameter haben. Deshalb kann er in der Methode *getParameterDef* definieren, welche Parameter sein Algorithmus erwartet und welche davon gesetzt werden müssen. Dabei kann er den Namen und den Typ des Parameters definieren. Ihm stehen folgende Typen zur Auswahl: *boolean, natural, real* und *text*. Um es dem Benutzer zu ermöglichen, die in einer konkreten Implementierung definierten Parameter zu setzen, passt sich der Parameter-Dialog automatisch an den aktuell geladenen Bicluster-Algorithmus an. Weiterhin überprüft der Dialog (entsprechend der Parameter-Typen) die Benutzereingaben auf Korrektheit. Ein Parameter vom Typ *boolean* wird mittels einer *QCheckBox* dargestellt, womit eine Überprüfung auf Korrektheit nicht durchgeführt werden muss. Die weiteren Parameter-Typen werden mittels *QLineEdit* Feldern dargestellt. Ist der Parameter vom Typ *text*, so darf eine beliebige Zeichenkette eingegeben werden, ist er vom Typ *natural*, dürfen nur Natürliche Zahlen eingegeben werden. Beim Typ *real* dürfen beliebige Reelle Zahlen eingegeben werden. Die Eingaben des Benutzers werden in einer Parameter-Liste gespeichert und bei Aufruf der *compute*-Methode an den Bicluster-Algorithmus gesendet.

Die *quality*-Methode wird benötigt, da die Anwendung im Voraus nichts über die Biclusterberechnung weiß. Die Qualität eines Biclusters hängt sehr stark davon ab, welche Bicluster der Algorithmus zu finden versucht. Deshalb muss für jede Ausführung eines Bicluster-Algorithmus

ebenfalls die *quality*-Methode implementiert werden, damit die Anwendung Aufschluss über die Güte eines bestimmten Biclusters erhält.

Der gesamte Mechanismus wird gegenüber der Anwendung gekapselt. Das Laden und Ausführen von Bicluster-Algorithmen wird durch eine *Executer*-Klasse durchgeführt. Dieser muss nur der Pfad zu der auszuführenden so-Datei mitgegeben werden. Über einen Flag wird ihr mitgeteilt, ob die auszuführenden Bicluster-Algorithmen die *R*-Umgebung benötigen. Das Starten und Beenden der *R*-Umgebung wird von der Executer Klasse übernommen. Zum Ausführen eines Bicluster-Algorithmus öffnet sie die angegebene so-Datei und ruft die *create*-Methode auf. Diese C-Methode müssen alle erstellten Implementierungen anbieten. Sie dient dazu eine neue Instanz der in der Implementierung definierten Klasse zu erzeugen. Weiterhin muss jede Implementierung über eine *destroy*-Methode verfügen, um das erstellte Objekt wieder zu löschen. Eine genaue Beschreibung dieser beiden Methoden befindet sich in Anhang B auf Seite 103.

In dieser Diplomarbeit werden zusätzlich zwei konkrete Implementierungen erstellt. Sie werden im Folgenden kurz beschrieben:

- **ChengChurch**: Sie implementiert den Algorithmus von CHENG und CHURCH (vgl. Abschnitt 2.3). Um diesen zu realisieren wird das *R*-Paket *Biclust* verwendet. In diesem Paket ist die Methode *biclust* implementiert, die von dieser Implementierung aufgerufen wird. Da diese Implementierung die *R*-Umgebung benutzt erbt sie von der abstrakten Klasse *AbstractRBiclusterAlgorithm*. In der *quality*-Methode ist die Güte-Funktion H „mean squared residue“ implementiert.

- **LoadStoredBicluster**: Diese Implementierung liest lediglich gespeicherte Bicluster aus einer Datei aus, weshalb sie die *R*-Umgebung nicht benötigt. Sie erbt von der abstrakten Klasse *AbstractBiclusterAlgorithm*. Die Datei enthält in der ersten Zeile die Anzahl der Zeilen und Spalten der Datenmatrix. Beide Zahlen sind durch ein Leerzeichen voneinander getrennt. In den folgenden Zeilen ist jeweils ein Bicluster pro Zeile definiert. In jeder Zeile werden zunächst die Zeilenindices und dann die Spaltenindices aufgelistet. Die Indices sind durch Leerzeichen getrennt. Die Indexmengen werden durch ein „#“ voneinander getrennt. Die *quality*-Funktion implementiert ebenfalls die Gütefunktion H von CHENG und CHURCH.

3.4 Realisierung der Heatmap

Das Grundprinzip einer Heatmap ist, Datenwerte auf Farbwerte abzubilden und diese an entsprechender Stelle in einem Raster darzustellen. Genexpressionsdaten besitzen in der Regel zwischen 20 und 80 Spalten und unter Umständen mehrere Tausend Zeilen. Deshalb kann die Heatmap meist nicht komplett in einer Ansicht dargestellt werden. Dem Benutzer werden in der hier entwickelten Darstellung Zoom- und Scroll-Funktionen zur Verfügung gestellt. Somit kann er Teile der Heatmap, die im Fokus seines Interesses stehen, genauer betrachten oder er kann sich durch Herauszoomen einen Überblick über die Heatmap anzeigen lassen.

In dieser Darstellung werden die Datenwerte auf Grautöne abgebildet. Dies wird erreicht, indem zwischen dem kleinsten und dem größten Datenwert innerhalb der Datenmatrix linear interpoliert wird. Der Grauton g wird anhand des RGB-Farbmodells berechnet. Dabei gilt für den Grauton g, für den Datenwert x, mit dem maximalen Wert x_{max} und dem minimalen Wert x_{min}:

$$g = RGB(c, c, c), \text{ mit } c = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Ziel der Darstellung ist es, mehrere auch überlappende Bicluster darzustellen. Dies soll erreicht werden, indem Zeilen und Spalten innerhalb der Datenmatrix vertauscht und falls nötig verdoppelt werden. Bicluster werden dabei durch umschließende Rechtecke innerhalb der Heatmap dargestellt. Weil die Bicluster unter Umständen nicht zusammenhängend dargestellt werden können, kann ein Bicluster auch aus mehreren umschließenden Rechtecken bestehen. Die Rechtecke werden in unterschiedlichen Farben gezeichnet[13], um die Zuordnung zu den einzelnen Biclustern zu erleichtern. Die Start- und Endindices für jedes Rechteck, sowie die Farben der einzelnen Bicluster werden in den *BiclusterVisualModel*-Objekten gespeichert. In der Standard-Ansicht wird nur das erste in einem Bicluster-Objekt gespeicherte Rechteck gezeichnet. Der Benutzer kann auswählen, ob er jeden Bicluster nur durch sein „Hauptrechteck“ angezeigt haben möchte oder ob alle Rechtecke eines Biclusters angezeigt werden. Werden alle Rechtecke ange-

13 Die Farben werden gemäß der „Rainbow-Map“ den einzelnen Biclustern zugeordnet, wobei ein reines Gelb ausgeschlossen wird. Da die Farben sich ab 15 Biclustern nicht mehr gut unterscheiden lassen, kann der Benutzer die Farben der einzelnen Bicluster zu jedem Zeitpunkt ändern.

zeigt, erhält der Benutzer eine Darstellung, in der auch die Überlappung nicht zusammenhängender Bicluster dargestellt ist. Nur das „Hauptrechteck“ eines Biclusters wird mit durchgezogenen Linien dargestellt. Alle anderen Rechtecke werden nur mit gestrichelten Linien gezeichnet.

Zusätzlich kann der Benutzer durch Anklicken eines Rechtecks den entsprechenden Bicluster auswählen. Ist ein Bicluster selektiert, werden alle Zellen des Biclusters in einem transparenten Gelb eingefärbt. Auch mehrere Bicluster können selektiert werden, wodurch eine Darstellung entsteht, in der Zellen, die zu vielen Biclustern gehören in einem starken Gelb, Zellen, die zu wenigen Biclustern gehören, in einem schwachen Gelb gezeichnet werden. Der Benutzer hat weiterhin die Möglichkeit Bicluster permanent einfärben zu lassen. Einfärben bedeutet in diesem Fall, dass alle Zellen, die einem Bicluster angehören in der dem Bicluster zugeordneten Farbe hervorgehoben werden.

In Abbildung 23 sind die verschiedenen Anzeigemodi der Heatmap an einem Beispiel mit sechs Biclustern dargestellt. In a) ist die Standard-Ansicht dargestellt, b) zeigt alle Rechtecke der Bicluster, in c) sind drei Bicluster markiert, wodurch Überlappungen in einem stärkeren Gelb zu erkennen sind und in d) sind exemplarisch zwei Bicluster dauerhaft hervorgehoben.

Um die Bicluster den einzelnen Zeilen und Spalten besser zuordnen zu können, werden über der Heatmap die Spaltennamen, links neben ihr die Zeilennamen eingeblendet. Die Namensanzeige ändert sich gemäß der aktuellen Zoom- und Scroll-Einstellung des Benutzers. Ist ein Bicluster selektiert, so werden die entsprechenden Zeilen und Spalten in der Namensanzeige ebenfalls hervorgehoben. Weil die Bicluster unter Umständen nicht zusammenhängend gezeichnet werden können, ist die Frage, welche Bicluster in einer bestimmten Zeile oder Spalte liegen, meist nur schwer zu beantworten. Der Benutzer kann deshalb durch Selektieren einer bzw. mehrerer Zeilen oder Spalten, alle Bicluster selektieren, die in den entsprechenden Zeilen bzw. Spalten liegen.

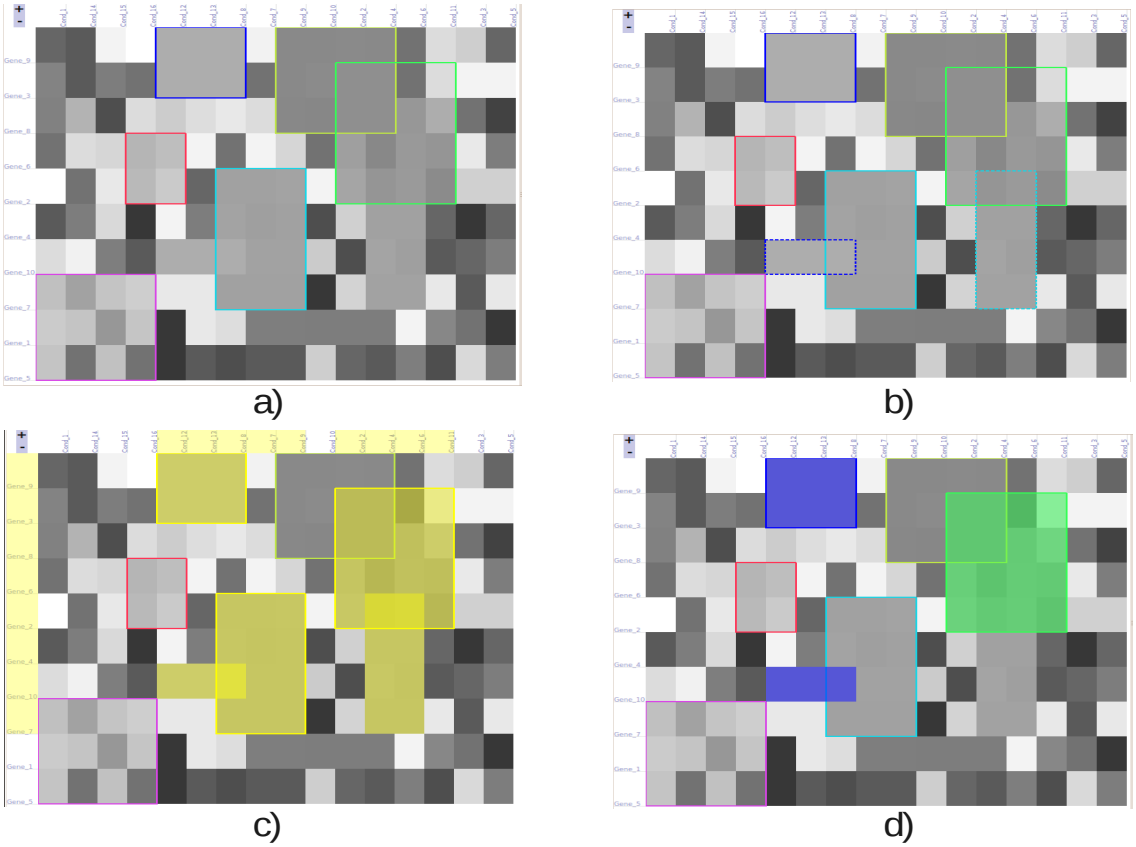


Abbildung 23: Verschiedene Darstellungsmodi innerhalb der Heatmap
a) Standard-Ansicht: jeder Bicluster wird nur durch sein „Hauptrechteck“ dargestellt
b) Darstellung aller Rechtecke eines Biclusters
c) Darstellung mit drei selektierten Biclustern
d) Darstellung mit permanent hervorgehobenen Biclustern

Wie in Abschnitt 3.1.2 auf Seite 43 bereits gezeigt wurde, ist es unter Umständen unmöglich alle Bicluster zusammenhängend ohne Verdopplung anzuzeigen. Lediglich für zwei Bicluster kann im Allgemeinen garantiert werden, dass beide Bicluster ohne Verdopplung zusammenhängend dargestellt werden können. Dies sind bei dem in dieser Diplomarbeit entwickelten Algorithmus (Abschnitt 3.4.1) die ersten beiden Bicluster, die in die Heatmap eingefügt werden. Deshalb wird dem Benutzer die Möglichkeit gegeben Zeilen und Spalten verdoppelt darzustellen, so dass ein Bicluster (basierend auf der aktuellen Sortierung) zusammenhängend in der Heatmap angezeigt wird. Dabei kann der Benutzer immer nur einen Bicluster auf diese Weise darstellen lassen. Es werden immer nur die Zeilen und Spalten verdoppelt dargestellt, die benötigt werden, um den vom Benutzer gewählten Bicluster zusammenhängend darzustellen.

In Abbildung 24 ist im linken Teil der cyan-gefärbte Bicluster nicht zusammenhängend dargestellt (zu erkennen an dem gestrichelten Rechteck). Im rechten Teil werden die Spalten so verdoppelt angezeigt, dass er zusammenhängend dargestellt wird. Die verdoppelten Zeilen und Spalten werden mit einem transparenten Rot hervorgehoben. Um die verdoppelten Zeilen und Spalten besser den originalen Zeilen und Spalten zuordnen zu können, werden zusätzlich Linien zwischen den originalen und verdoppelten Elementen gezeichnet (unterer Teil der Abbildung). Der Benutzer kann entscheiden, ob diese Linien eingeblendet werden. Sind die Linien eingeblendet, kann der Benutzer durch Anklicken einer verdoppelten Zeile bzw. Spalte alle entsprechenden Linien hervorheben. Die Bicluster-Selektion über die Zeilen- und Spaltennamen wird in diesem Anzeigemodus deaktiviert.

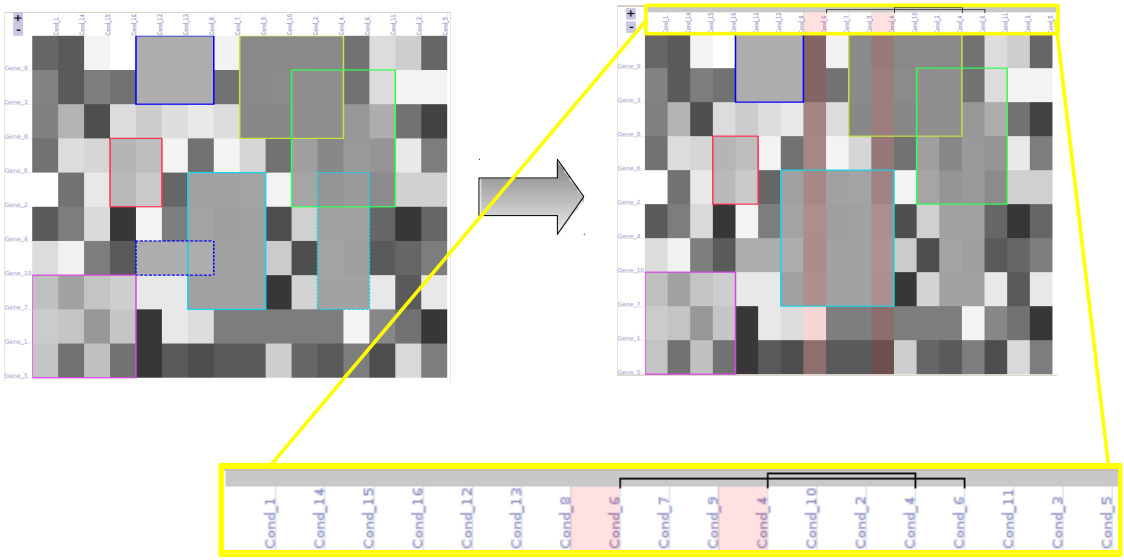


Abbildung 24: Verdopplung von Zeilen und Spalten

Für die in dieser Diplomarbeit entwickelte Darstellung der Bicluster innerhalb der Heatmap ist die Reihenfolge, in der sie eingefügt werden, relevant. Daher ist ein Algorithmus entworfen worden, der die Sortierung der Bicluster bestimmt. Dieser ist darauf ausgelegt, dass Bicluster, die viele gemeinsame Zeilen oder Spalten besitzen, in der Sortierung nahe beieinander liegen. Somit wird beispielsweise gewährleistet, dass der zweite Bicluster, der in die Heatmap eingefügt wird, derjenige ist, der mit dem zu Beginn eingefügten Bicluster die meisten Zeilen und Spalten gemeinsam hat. Im Folgenden soll die Grundidee der Sortierung nur kurz skizziert werden, eine genaue Beschreibung der Anordnung von Zeilen und Spalten wird in Abschnitt 3.4.1 gegeben.

In jedem Schritt der Berechnung der Anordnung der Zeilen und Spalten wird genau ein Bicluster in die Heatmap eingefügt. Dabei werden die jeweiligen Zeilen- und Spaltenindices des gerade eingefügten Biclusters mit den Mengen I und J vereint, welche zu Beginn der Berechnung leer sind. Somit sind in I alle Zeilenindices der bisher eingefügten Bicluster enthalten, in J alle Spaltenindices. Als nächster Bicluster wird derjenige in die Heatmap eingefügt, der die größte Überschneidung mit den bisher eingefügten Biclustern aufweist. Das bedeutet, es wird derjenige Bicluster B_i mit den Indexmengen I_i und J_i in die Heatmap eingefügt, für den die Überschneidung S maximal ist, mit $S = |I \cap I_i| + |J \cap J_i|$ [14]. Für die Standard-Anordnung wird das Einfügen mit demjenigen Bicluster begonnen, der mit den anderen Biclustern die stärkste Überschneidung aufweist. Somit ist die Reihenfolge in der die Bicluster in die Heatmap eingefügt werden fest und damit werden unter Umständen auch immer nur dieselben Bicluster zusammenhängend ohne Verdopplung dargestellt. Daher wird dem Benutzer die Möglichkeit angeboten, den Bicluster zu bestimmen, mit dem das Einfügen begonnen werden soll. Somit kann er den Bicluster, den er aktuell untersuchen möchte, als Ausgangspunkt der Anordnung der Zeilen und Spalten festlegen. Die restlichen Bicluster werden sozusagen um den von ihm bestimmten Bicluster angeordnet. Beim Einfügen der Bicluster gilt der Grundsatz: *Je früher ein Bicluster in die Heatmap eingefügt wird, desto größer ist die Wahrscheinlichkeit, dass er ohne Verdopplung zusammenhängend dargestellt werden kann.*

In Abbildung 25 ist im linken Teil die Standard-Anordnung dargestellt, das Einfügen beginnt mit dem grünen Bicluster. Der cyan-gefärbte Bicluster kann in dieser Anordnung nur mittels Verdopplung zusammenhängend dargestellt werden. Im rechten Teil der Abbildung wurde das Einfügen mit dem cyan-gefärbten Bicluster begonnen. Dieser ist nun ohne Verdopplung zusammenhängend dargestellt. Weiterhin ist der Bicluster (grün) mit der stärksten Überschneidung ebenfalls zusammenhängend ohne Verdopplung dargestellt.

Der Benutzer kann zusätzlich das Überschneidungskriterium festlegen. Während bei der Standard-Anordnung sowohl die Überschneidungen in Zeilen als auch in Spalten beachtet werden, hat er ebenfalls die Möglichkeit zu bestimmen, dass bei der Berechnung der Bicluster-Reihenfolge nur Zeilen oder nur Spalten betrachtet werden sollen.

---

14 Mit dem Betrag einer Menge S ist die Anzahl der Elemente der Menge S gemeint.

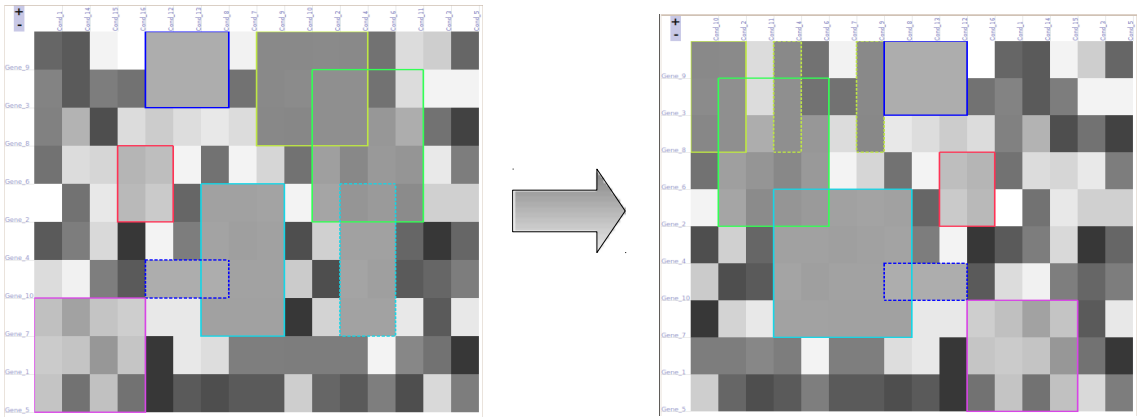


Abbildung 25: Umsortierung der Heatmap mit neuem Start-Bicluster

Bisher wurde bei der Anordnung der Bicluster nur die Überschneidung als Kriterium für die Bicluster-Reihenfolge beachtet. Dem Benutzer wird nun zusätzlich die Möglichkeit angeboten, die Bicluster-Reihenfolge komplett vorzugeben. Die Bicluster-Reihenfolge kann von der Sortierung der Navigationsliste (vgl. Seite 50) übernommen werden. Somit hat er beispielsweise die Möglichkeit, die Bicluster gemäß ihrer Qualität in die Heatmap einzufügen. Damit steigt die Wahrscheinlichkeit, dass Bicluster mit hoher Qualität zusammenhängend dargestellt werden. In Abbildung 26 ist links die Standard-Anordnung der Heatmap dargestellt. In dieser werden eher Bicluster mit starker Überschneidung zusammenhängend dargestellt. Im rechten Teil der Abbildung sind die Bicluster gemäß der Qualität in die Heatmap eingefügt worden. In dieser Anordnung werden eher Bicluster mit hoher Qualität (B_4, B_1 und B_6) zusammenhängend dargestellt.

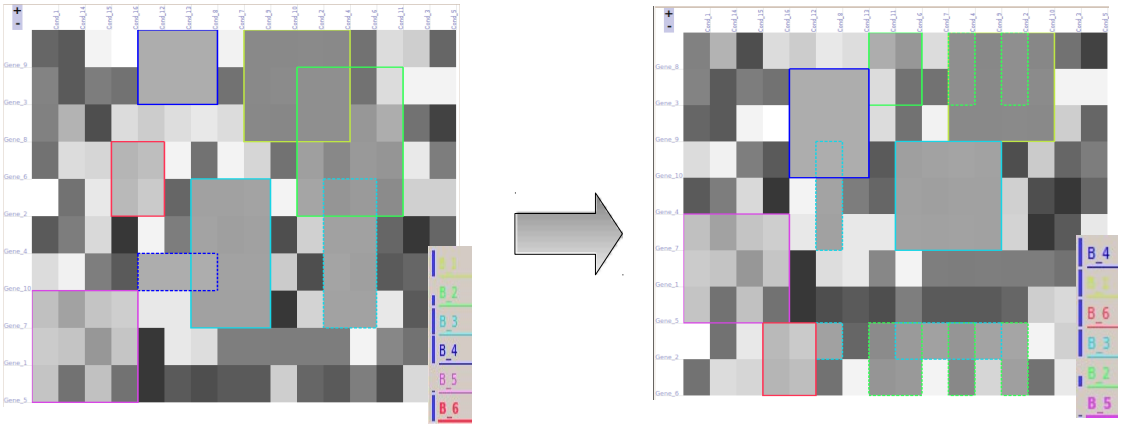


Abbildung 26: Bicluster-Reihenfolge entsprechend der Qualität

3.4.1 Berechnung des Zeilen- und Spalten-Layouts

Eine gute Anordnung der Zeilen und Spalten bildet die Voraussetzung für eine gute Darstellung der Bicluster innerhalb einer Heatmap. Die Berechnung der Anordnung und der Verdopplungen wird von der Klasse *IndexMapping* übernommen. Die Heatmap stellt nur die Ergebnisse der Berechnungen aus dieser Klasse und die Datenwerte innerhalb des *Matrix*-Objekts dar. Die Start- und Endindices der umschließenden Rechtecke der Bicluster werden ebenfalls von der Klasse *IndexMapping* berechnet. Für eine Datenmatrix und eine Menge von darauf berechneten Biclustern existiert immer genau ein Objekt dieser Klasse. Das Objekt wird allen Visualisierungsmodulen bekannt gemacht, wodurch alle Module ihre Visualisierung auf Grundlage derselben Sortierung von Zeilen und Spalten berechnen. Die aktuell berechnete Sortierung wird in dem *IndexMapping*-Objekt zwischengespeichert, wodurch eine erhebliche Beschleunigung in den Visualisierungen erreicht wird. Die Visualisierungsmodule können daher schnell die benötigten Zeilen- und Spaltenindices zu einem bestimmten Bicluster aus diesem Objekt auslesen. Die Hauptaufgabe der *IndexMapping*-Klasse ist das Berechnen der Anordnung der Zeilen und Spalten. Der dafür entwickelte Algorithmus soll im Folgenden beschrieben werden.

Berechnung der Bicluster Reihenfolge

Um die Reihenfolge der Bicluster festzulegen, in der sie in die Heatmap eingefügt werden, wird ein sogenannter Konfliktgraph aufgebaut. Der Konfliktgraph ist ein stark zusammenhängender, gewichteter Graph, dessen Knoten die Bicluster mit ihren Zeilen- und Spaltenindices repräsentieren. Zwischen je zwei Knoten existiert eine Kante, die die Überschneidung (Konflikte) zwischen den einzelnen Biclustern darstellt. Dabei ist das Kantengewicht g einer Kante e wie folgt definiert:

> *Seien B_1 und B_2 zwei Bicluster mit den Zeilenindices I_1 und I_2 und den Spaltenindices J_1 und J_2, dann gilt für das Kantengewicht g der Kante e zwischen B_1 und B_2:*
> $g(e) = |I_1 \cap I_2| + |J_1 \cap J_2|$.

Die Reihenfolge in der die Bicluster in die Heatmap eingefügt werden, wird anhand des Konfliktgraphen bestimmt. Dabei werden die Knoten des Konfliktgraphen in der Menge B miteinander verschmolzen. Für das Kantengewicht der Kanten zwischen den bereits verschmolzenen Knoten und einem bisher noch nicht verschmolzenen Knoten gilt:

Sei B die Menge der i verschmolzenen Knoten, mit $B=\{B_1, B_2, \ldots, B_i\}$ *und* B_j *ein bisher noch nicht verschmolzener Knoten, mit* $B_j \notin B$, *dann gilt für das Kantengewicht g der Kante e zwischen B und* B_j*:* $g(e)=|I_B \cap I_j|+|J_B \cap J_j|$, *mit* $I_B = \bigcup_{1 \le k \le i} (I_k)$ *und* $J_B = \bigcup_{1 \le k \le i} (J_k)$.

Die Idee des Algorithmus ist, immer den Bicluster in die Heatmap einzufügen, welcher mit den bereits eingefügten Biclustern B am stärksten in Konflikt steht. Er wird im Folgenden beschrieben:

Algorithmus zur Berechnung der Reihenfolge der Bicluster B_1, B_2, … , B_n:

Sei B die Menge der bereits in die Heatmap eingefügten Bicluster und sei R die Menge der bisher noch nicht eingefügten Bicluster. Sei B_m der Bicluster, bei dem die Summe der Kantengewichte maximal ist.

Zu Beginn gilt: $B=\emptyset$ und $R=\{B_1, B_2, \ldots, B_n\}$. Füge B_m in die Heatmap ein und setzte: $B=\{B_m\}$ und $R=R \setminus \{B_m\}$.

Wiederhole die folgenden Schritte so lange gilt $R \neq \emptyset$:

1) Finde denjenigen Knoten B_i dessen Kante zwischen B und B_i das größte Gewicht hat.
2) Füge B_i in die Heatmap ein.
3) Verschmelze B_i mit B. Das heißt: $B=B \cup \{B_i\}$ und $R=R \setminus \{B_i\}$.

Der Algorithmus soll für ein besseres Verständnis anhand eines Beispiels durchgeführt werden. Sei beispielhaft eine Menge R von 6 Biclustern gegeben, mit $R=\{B_1, B_2, B_3, B_4, B_5, B_6\}$:

$B_1: I_1=\{2,7,8\}, J_1=\{1,3,8,9\}$ $B_2: I_2=\{1,2,5,7\}, J_2=\{1,3,5,10\}$

$B_3: I_3=\{1,3,6,9\}, J_3=\{3,5,6,7,8\}$ $B_4: I_4=\{2,8,9\}, J_4=\{7,11,12\}$

$B_5: I_5=\{0,4,6\}, J_5=\{0,13,14,15\}$ $B_6: I_6=\{1,5\}, J_6=\{11,15\}$

Die ausgeführten Schritte des Algorithmus zeigt Abbildung 27 auf Seite 65. Kanten mit dem Kantengewicht 0 sind durch gestrichelte Linien dargestellt. Zusätzlich sind die Zeilenindexmenge (R) und Spaltenindexmenge (C) der Bicluster bei jedem Knoten eingezeichnet. Die Knoten sind entsprechend den Biclustern gewählt. Zu Beginn wird für jeden Knoten die Summe über

alle seine Kantengewichte gebildet. B_2 wird als erster Knoten in B eingefügt, da die Summe seiner Kantengewichte mit 10 maximal ist (vgl. (1)). Damit ergibt sich für die Bicluster-Menge B: $B=\{B_2\}$; für die Restmenge R ergibt sich: $R=\{B_1, B_3, B_4, B_5, B_6\}$. Dies ist in a) dargestellt. Als nächster Knoten wird derjenige gewählt, der mit B die Kante mit dem größten Gewicht hat. Dies ist gemäß a) Knoten B_1. Somit ergibt sich für die Biclustermenge B und die Restmenge R: $B=\{B_2, B_1\}$ und $R=\{B_3, B_4, B_5, B_6\}$. Der entstandenen Konfliktgraph ist in b) dargestellt. Im nächsten Schritt wird der Knoten B_3 dessen Kante zu B mit 4 das maximale Gewicht hat eingefügt. In c) ist der entstandene Konfliktgraph dargestellt, es gilt: $B=\{B_2, B_1, B_3\}$ und $R=\{B_4, B_5, B_6\}$. In den nächsten Schritten werden schließlich die Knoten B_4 und B_6 in B eingefügt. Die dadurch entstehenden Konfliktgraphen sind in d) und e) dargestellt. Als letzter Knoten wird B_5 eingefügt. Nach dieser Verschmelzung ist die Restmenge R leer und der Algorithmus wird beendet. Damit ergibt sich die Reihenfolge der sechs gegebenen Bicluster: $B_2 \rightarrow B_1 \rightarrow B_3 \rightarrow B_4 \rightarrow B_6 \rightarrow B_5$.

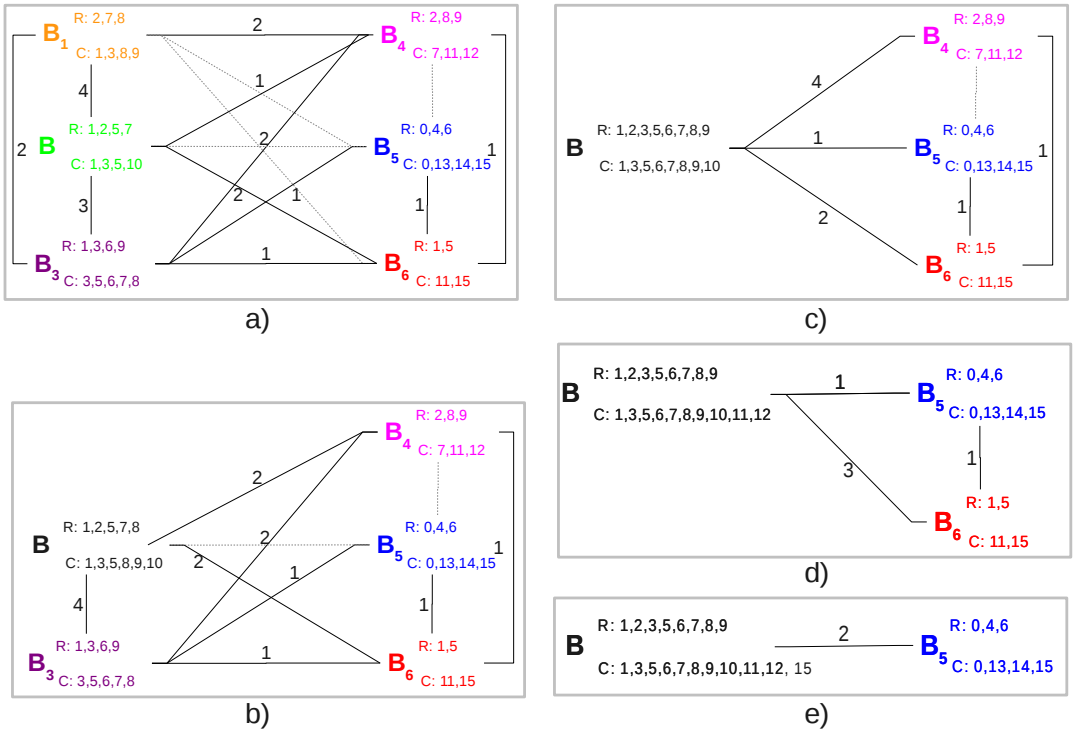


Abbildung 27: Beispielhafte Durchführung der Berechnung der Bicluster-Reihenfolge
(1) B_2 wird als erster Knoten gewählt, $B=\{B_2\}$. (2) B_1 wird als zweiter Knoten in B eingefügt $B=\{B_2, B_1\}$. (3) B_3 wird in B eingefügt, $B=\{B_2, B_1, B_3\}$. (4) B_4 wird in B eingefügt, $B=\{B_2, B_1, B_3, B_4\}$. (5) B_6 wird in B eingefügt, $B=\{B_2, B_1, B_3, B_4, B_6\}$.

Berechnung der Anordnung der Zeilen und Spalten

Ist die Reihenfolge der Bicluster bestimmt, werden die Zeilen und Spalten der Heatmap so angeordnet, dass möglichst wenige verdoppelt werden müssen. Der Algorithmus zum Anordnen der Spalten läuft immer auf die gleiche Art und Weise ab. Der Benutzer kann keinen Einfluss auf die Anordnung nehmen. Die Einflussnahme des Benutzers beschränkt sich auf die Bicluster-Reihenfolge. Diese entscheidet letztendlich darüber, wie viele Zeilen und Spalten verdoppelt werden müssen und welche Bicluster zusammenhängend dargestellt werden können. Im Folgenden wird beschrieben, wie die Anordnung der Zeilen und Spalten der Heatmap aus der Bicluster-Reihenfolge berechnet wird.

Die Grundidee des Algorithmus ist ähnlich der Idee des Widerspruch-Beispiels auf Seite 44. Zeilen und Spalten dürfen innerhalb eines Bicluster zunächst frei vertauscht werden. Der Bicluster bleibt damit immer noch zusammenhängend. Die Anordnung der Zeilen und Spalten innerhalb eines Biclusters wird erst durch eine Überschneidung mit einem anderen Bicluster eingeschränkt. Dieser Sachverhalt wird auf einer Datenstruktur abgebildet und auf dieser wird dann versucht die Bicluster so anzuordnen, dass möglichst wenig Zeilen und Spalten verdoppelt werden müssen. Deshalb wird versucht bereits eingefügte Indices für die Darstellung der neu einzufügenden Bicluster zu verwenden. Ziel der Anordnung ist es, Bicluster zusammenhängend darzustellen. Da dies im Allgemeinen nicht möglich ist, werden zusätzlich verdoppelte Zeilen- und Spaltenindices eingefügt. Die verdoppelten Indices werden nur dann in der Heatmap angezeigt, wenn ein Bicluster nicht zusammenhängend ist und der Benutzer diesen zusammenhängend darstellen möchte.

Die Datenstruktur besteht zu Anfang aus einer, später aus mehreren Listen, die wiederum aus *IndexNode*-Objekten (Knoten) bestehen. Jeder dieser Knoten repräsentiert einen Bicluster. Die Zeilen und Spalten werden unabhängig voneinander angeordnet, daher wird die Datenstruktur aus zwei *IndexTree*-Objekten gebildet, eines für die Zeilen und eines für die Spalten der Datenmatrix. Ein solches *IndexTree*-Objekt speichert mehrere der Listen sowie die gesamte Indexmenge der Zeilen und Spalten der Datenmatrix und übernimmt das Einfügen der Knoten in die Listen. In jedem Knoten einer Liste gibt es insgesamt fünf weitere Listen, die die Abhängigkeiten in der Abfolge der Indices speichern. Weiterhin hat jeder Knoten alle Indices des Biclusters ge-

speichert, den er repräsentiert. Der schematische Aufbau eines solchen Knotens für den Bicluster B_i ist in Abbildung 28 dargestellt.

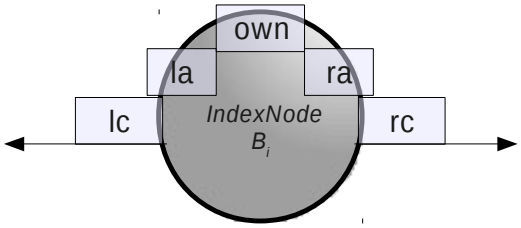


Abbildung 28: Schematischer Aufbau eines *IndexNode*-Knotens

Die fünf Listen innerhalb eines Knotens heißen: *leftCommon*, *leftAttached*, *own*, *rightAttached* und *rightCommon*. Die Listen *leftCommon* und *rightCommon* speichern jeweils die Indices, die ein Knoten mit seinem linken bzw. rechten Nachbarn gemeinsam hat. Die Liste *own* speichert die Indices, die bisher nur diesem Knoten zugeordnet sind. In diesen drei Listen ist die Anordnung der Indices egal, das bedeutet sie dürfen innerhalb der Listen beliebig vertauscht werden. Die Listen *leftAttached* und *rightAttached* dienen dem Anheften bestimmter Indices, wenn diese über mehrere Nachbarn hinweg benötigt werden, um einen Bicluster zusammenhängend darzustellen. In diesen Listen darf die Anordnung der Indices nicht mehr vertauscht werden. Durch Auslesen der Listen von links nach rechts ergibt sich dann die Anordnung der Indices. Somit wird beispielsweise gewährleistet, dass die Indices der *leftAttached* Liste immer nach denen der *leftCommon* Liste in der Anordnung vorkommen.

Die Aufgaben der Listen wird im folgenden Beispiel erklärt. Es werden in dem Beispiel nur die Spaltenindices betrachtet, das Einfügen der Zeilen verläuft analog. Gegeben seien die Bicluster B_1, B_2 und B_3 mit den Spaltenindices $J_1 = \{2,7,8\}$, $J_2 = \{1,2,7\}$, $J_3 = \{1,2,3,6\}$. In Abbildung 29 ist in (1) die entstandene Liste dargestellt, nachdem B_1 und B_2 eingefügt worden sind. In den jeweiligen *Common*-Listen sind die Indices eingetragen, welche die beiden Bicluster gemeinsam haben. In den *own* Listen von B_1 und B_2 sind die Indices, welche die beiden Bicluster noch nicht „teilen“ mussten. In (2) wird die Liste gezeigt, nachdem B_3 eingefügt worden ist. Da B_3 mit B_1 den Index 1 gemeinsam hat, rutscht dieser aus der *own* Liste von B_2 in die entsprechenden *Common*-Listen. Die Indices 3 und 6 werden von keinem anderen Bicluster benötigt und somit bleiben diese in der *own* Liste von B_3. B_3 benötigt jedoch zusätzlich noch den Index 2, welcher bereits in die *Common*-Listen von B_1 und B_2 eingefügt worden ist. Dieser ist von B_3 aus

ohne einen dazwischenliegenden anderen Index erreichbar, wenn die 2 rechts neben der 7 auftritt. Dies ist jedoch innerhalb der *Common*-Listen nicht gewährleistet, da die Reihenfolge in diesen Listen nicht beachtet wird. Deshalb wird die 2 aus den *Common*-Listen gelöscht und in die leftAttached Liste von B_2 eingetragen. Somit ist gewährleistet, dass B_1 immer noch die 2 erreichen kann, diese jedoch rechts von der 7 auftritt, so dass sie von B_3 ebenfalls für dessen Darstellung verwendet werden kann und nicht verdoppelt werden muss. Für die Anordnung der Indices ergibt sich die Reihenfolge: $8 \rightarrow 7 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 6$.

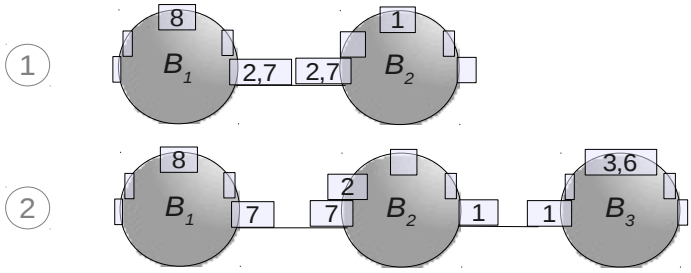


Abbildung 29: Beispielhaftes Einfügen von Biclustern zur Verdeutlichung der Listen
(1) B_1 und B_2 sind eingefügt worden
(2) B_3 ist eingefügt worden

Die Anordnung wird berechnet, indem die Listen der *IndexTree*-Objekte gemäß der Bicluster-Reihenfolge aufgebaut werden. Dabei wird eine Restmenge verwaltet, die die bisher noch nicht eingefügten Zeilen- bzw. Spaltenindices speichert. Ein neuer Knoten wird so eingefügt, dass die Anzahl der verdoppelten Zeilen bzw. Spalten möglichst gering ist und kein anderer bereits eingefügter Bicluster seine zusammenhängenden Indices verliert. Dabei gilt die grundlegende Regel: *zwischen zwei nicht verdoppelten bereits eingefügten Indices darf ein neuer Index nur verdoppelt eingefügt werden*. Verdoppelte Indices werden in den Knoten dadurch hervorgehoben, dass sie negativ[15] (also mit -1 multipliziert) sind. Dadurch wird verhindert, dass ein bereits zusammenhängender Bicluster innerhalb der Liste durch das Einfügen neuer Indices nicht mehr zusammenhängend ist. Das Einfügen verdoppelter Indices verletzt dies nicht, da die verdoppelten Indices bei der Standard-Darstellung ausgeblendet werden.

Zunächst muss der Knoten bestimmt werden, der für das Einfügen eines neuen Biclusters die wenigsten Verdopplungen benötigt. Dieser wird gefunden, indem jeder Knoten nach der benö-

15 Die 0 muss bei dieser Kennzeichnung besonders behandelt werden. Sie wird bei einer Verdopplung nicht zu -0, sondern erhält bei insgesamt n Indices $\{0,1,2,\ldots,n-1\}$ den Wert -n.

tigten Anzahl von Verdopplungen gefragt wird, wenn er den neuen Knoten bei sich anhängen würde. Der Algorithmus zur Bestimmung der Verdopplung von Indices wird im Folgenden erklärt.

Je nach Lage der Knoten innerhalb der Liste, können die Knoten die Indices unterschiedlich einfügen. Dabei sind zwei Fälle zu unterscheiden: der Knoten liegt am Rand der Liste und der Knoten liegt innerhalb der Liste. Liegt der Knoten am linken Rand der Liste, so kann der neue Knoten links oder rechts an den Knoten angehängt werden. Liegt er am rechten Rand oder innerhalb der Liste, so kann der neue Knoten nur rechts angehängt werden. Dabei ist zu beachten, dass Indices nur dann nicht verdoppelt eingefügt werden dürfen, wenn der neue Knoten am Rand der Liste angehängt werden kann und die benötigten Indices bisher noch nicht in einem anderen Knoten eingefügt wurden.

Zunächst wird der Fall betrachtet, wenn der Knoten, an den der neue Knoten angehängt werden soll, am Rand liegt. Hier wird nur der Fall beschrieben, wenn sich der Knoten am linken Rand befindet, für den rechten Rand gelten die Ausführungen analog. Für jeden Knoten werden die Listen von links nach rechts überprüft, ob sie passend für die neu einzufügenden Indices sind. Eine Liste wird dann als passend bezeichnet, wenn die in ihr befindlichen nicht verdoppelten Indices alle in den neu einzufügenden Indices des Biclusters enthalten sind. Zunächst wird geprüft, ob die Liste *own* für die neuen Indices passend ist. Ist dies der Fall, wird als nächstes die Liste *leftAttached* geprüft. Ist diese ebenfalls passend wird die *leftCommon* Liste untersucht, ob auch diese passend ist. Sind alle drei Listen passend für die neuen Indices, wird der rechte Nachbarknoten für die Berechnung der Verdopplung betrachtet. Bei diesem muss allerdings die Liste *leftCommon* nicht mehr daraufhin geprüft werden, ob sie passend ist, da sie dieselben Indices wie die *rightCommon* Liste des aktuellen Knotens enthält. Passen alle Listen innerhalb des rechten Nachbarn (geprüft von links nach rechts) so wird dessen Nachbar ebenfalls in die Betrachtung mit einbezogen. Die Einbeziehung der rechten Nachbarknoten ist in Abbildung 30 dargestellt. Dabei ist der blau umrandete Knoten derjenige, an den der neue Knoten links angehängt werden soll. Dessen Listen *leftAttached* und *leftCommon* sind leer, da er keinen linken Nachbar besitzt.

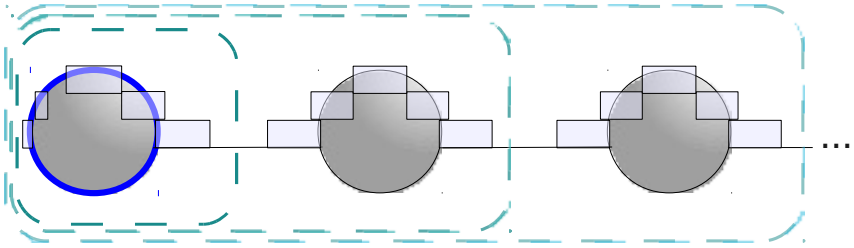


Abbildung 30: Einbeziehung der rechten Nachbarknoten bei der Berechnung der Verdopplung am linken Rand der Liste

Für die Berechnung der Verdopplung werden die verschiedenen Listen unterschiedlich betrachtet, falls sie für die neuen Indices nicht mehr passend sind. Wird bei der Überprüfung der Listen ein nicht verdoppelter Index entdeckt, der nicht in der Menge der neu einzufügenden Indices liegt, wird die Überprüfung bei den *Attached*-Listen sofort abgebrochen. Die anderen Listen werden in diesem Fall noch bis zum Ende durchsucht, ob sich in ihnen noch ein Index befindet, der in der Menge der neu einzufügenden Indices liegt. Nach der Betrachtung der entsprechenden Nachbarn hat der Knoten eine Liste von unterstützenden Indices, die er für die neu einzufügenden Indices nicht mehr verdoppeln muss, da sie über die Nachbarn zusammenhängend erreicht werden können. Als unterstützend wird ein Index bezeichnet, wenn er von einem Knoten benötigt wird und direkt von diesem Knoten aus erreichbar ist, also ohne dazwischenliegende nicht verdoppelte Indices, die nicht von diesem Knoten benötigt werden.

Anschließend prüft der Knoten wie viele Verdopplungen er benötigen würde, um den neuen Knoten links bzw. rechts an ihn anzuhängen. Bei der Berechnung der Verdopplung beim Anhängen links an den linken Rand bzw. rechts an den rechten Rand der Liste betrachtet der Knoten die Restmenge der verbleibenden Indices. Es müssen nur die Indices verdoppelt werden, die von seinen rechten Nachbarn nicht unterstützt werden und sich nicht in der Restmenge befinden. Beim Anhängen an die rechte Seite werden nur die unterstützten Indices seiner Nachbarn zusätzlich betrachtet, Indices aus der Restmenge dürfen in diesem Fall nur verdoppelt eingefügt werden. Damit ergibt sich für die Anzahl der Verdopplungen beim Einfügen am Rand der Listen: $Verdopplung = Anzahl_{neue\ Indices} - Anzahl_{unterstützende\ Indices} - Anzahl_{passende\ Restindices}$. Für die Anzahl der Verdopplungen innerhalb gilt: $Verdopplung = Anzahl_{neue\ Indices} - Anzahl_{unterstützende\ Indices}$.

Befindet sich der Knoten am rechten Rand der Liste werden anstatt der rechten Nachbarknoten, die linken Nachbarknoten auf unterstützende Indices untersucht. Beim Berechnen der Ver-

dopplung werden für das Anhängen an die rechte Seite des Knotens die Indices der Restmenge betrachtet. Beim Anhängen an die linke Seite des Knotens dürfen die Indices der Restmenge nur verdoppelt eingefügt werden.

Befindet sich der Knoten, für den die Verdopplung berechnet wird, nicht am Rand der Liste, läuft die Berechnung ähnlich ab. Bei einem Knoten innerhalb der Liste wird auch mit dessen *own* Liste begonnen. Hier werden ebenfalls nur die rechten Nachbarknoten auf unterstützende Indices untersucht. Dies liegt an der Tatsache, dass der aktuell betrachtete Knoten bereits als rechter Nachbarknoten seines linken Nachbarknotens betrachtete wurde. Somit sind die unterstützenden Indices, die eine Betrachtung des linken Nachbarknotens ergeben würden, bereits bei der Berechnung des vorherigen Knotens untersucht worden. Ein neuer Knoten kann nur an der rechten Seite des Knoten angehängt werden. Die Indices der Restmenge dürfen in diesem Fall auch nur verdoppelt eingefügt werden. Damit ergibt sich für die Verdopplung eines Knotens innerhalb der Liste: $Verdopplung = Anzahl_{neue\ Indices} - Anzahl_{unterstützende\ Indices}$.

Das Einfügen der Knoten verläuft nach einem ähnlichen Prinzip wie die Berechnung der Verdopplung. Auch hier wird wieder unterschieden, ob ein Knoten am Rand der Liste oder innerhalb der Liste eingefügt werden kann. Ob der neue Knoten links oder rechts an den Knoten angehängt werden soll, wird bereits bei der Berechnung der Verdopplung für jeden Knoten mit Hilfe einer Flag-Variablen gespeichert. Zusätzlich zu den Knoten, wird auch die Restmenge betrachtet. Der Knoten wird an den Knoten eingefügt, an der am wenigsten Verdopplungen benötigt werden. Dieser wird bereits bei der Berechnung der Verdopplung gespeichert. Wird für das Einfügen keine Verdopplung benötigt, wird die Suche nach dem besten Knoten abgebrochen, da er in diesem Fall bereits gefunden wurde.

Werden für den neuen Knoten die wenigsten Verdopplungen benötigt, wenn er in eine neue Liste eingefügt wird, wird eine neue Liste mit nur einem Knoten dem *IndexTree*-Objekt hinzugefügt. Dies ist vor allem beim Einfügen des ersten Knotens der Fall, da in der Restmenge alle Indices enthalten sind. Wird ein Knoten in eine neue Liste eingefügt, enthält seine *own* Liste alle benötigten Indices. Ob diese verdoppelt werden, hängt davon ab, welche Indices in der Restmenge noch enthalten sind. Beim Einfügen des ersten Knotens sind in seiner *own* Liste nur nicht verdoppelte Indices. Die nicht verdoppelten Indices, die in die *own* Liste des Knotens eingefügt wurden, werden aus der Restmenge entfernt.

Wird ein Knoten links an den linken Rand der Liste bzw. rechts an den rechten Rand der Liste angehängt, werden ebenfalls Indices nur dann nicht verdoppelt eingefügt, wenn diese in der Restmenge noch enthalten sind. Auch in diesem Abschnitt wird nur der Fall beschrieben, wenn ein Knoten links an den linken Rand der Liste angehängt wird. Der Fall, wenn der Knoten rechts an den rechten Rand der Liste angehängt wird verläuft analog. Dabei ist zu beachten, dass die *leftAttached* bzw. *rightAttached* Liste des Knotens leer ist. Der neu einzufügende Knoten wird im Folgenden mit **new**, der Knoten, an den der neue Knoten angehängt wird, wird mit **act** bezeichnet. Zunächst werden **new** in seine *own* Liste alle benötigten Indices eingefügt. Benötigte Indices sind in diesem Zusammenhang alle Zeilen- bzw. Spaltenindices, die der entsprechende Bicluster enthält. Es werden danach alle unterstützenden Indices aus der *own* Liste von **new** entfernt, die in der *own* Liste von **act** enthalten sind. Diese Indices werden in die *rightCommon* Liste von **new** und die *leftCommon* Liste von **act** geschrieben. Ist die *own* Liste von **act** passend für die benötigten Indices, wird als nächstes die *rightAttached* Liste von **act** überprüft. Auch hier wird abgebrochen, falls ein nicht verdoppelter Index entdeckt wird, der nicht in der benötigten Indexmenge ist. Alle unterstützenden Indices werden aus der *own* Liste von **new** entfernt. Ist die *rightAttached* Liste von **act** passend, wird die *rightCommon* Liste von **act** untersucht. Alle Indices die in dieser Liste in den benötigten Indices enthalten sind, werden aus der *own* Liste von **new** entfernt. Weiterhin werden alle unterstützenden Indices aus der *rightCommon* Liste von **act** entfernt und in dessen *rightAttached* Liste hinten angehängt. Hat **act** einen rechten Nachbarknoten, werden alle unterstützenden Indices aus dessen *leftCommon* Liste gelöscht. Ist die *rightCommon* Liste von **act** passend, wird der rechte Nachbarknoten auf unterstützende Indices untersucht. Das Einbeziehen der rechten Nachbarknoten verläuft nach demselben Prinzip, welches für die Berechnung der Verdopplung benutzt wird (vgl. Abbildung 30). Alle unterstützenden Indices der Nachbarknoten werden aus der *own* Liste von **new** entfernt. Die unterstützenden Indices der Nachbarknoten müssen in deren Listen ebenfalls entsprechend vertauscht werden. Dies wird durch zwei Methoden realisiert: *insertLeftIndices* und *insertRightIndices*. Diese beiden Methoden sind auf jedem der Knoten implementiert und verschieben die benötigten Indices aus den *own* und *Common*-Listen in die entsprechenden *Attached*-Listen.

Abschließend werden die verbleibenden Indices der *own* Liste von **new** mit der Restmenge verglichen. Alle Indices, die in der Restmenge nicht enthalten sind, werden in der *own* Liste von **new** als verdoppelt markiert. Danach werden alle verdoppelten Indices aus der *own* Liste von **new** gelöscht, die in der Indexmenge des entsprechenden Biclusters von **act** sind. Die gelöschten Indices werden den Listen *rightCommon* von **new** und *leftCommon* von **act** angehängt. Dieser letzte Schritt muss durchgeführt werden, da durch das Verschieben der Indices nicht mehr alle Indices eines Biclusters in der *own* Liste des entsprechenden Knotens sind.

Befindet sich der Knoten, an den der neue Knoten angehängt werden soll, innerhalb der Liste, wird immer rechts an diesen Knoten angehängt. Hier ist zu beachten, dass die *rightCommon* und die *rightAttached* Liste unter Umständen nicht leer sind (diese sind beim Anhängen am Rand der Liste immer leer). Weiterhin dürfen neue Indices nur verdoppelt in die Knoten eingefügt werden. Auch hier wird der neue Knoten, der angehängt wird, mit **new**, der Knoten, an den angehängt wird, wird mit **act** bezeichnet.

Zuerst werden alle benötigten Indices des Biclusters in die *own* Liste von **new** geschrieben. Dann werden die Indices in der Liste *rightAttached* von **act** überprüft. Ist diese Liste für die benötigten Indices passend, werden alle unterstützenden Indices aus der *own* Liste von **new** gelöscht. Als nächstes wird dann die *own* Liste von **act** betrachtet. Alle unterstützenden Indices werden aus der *own* Liste von **new** und der *own* Liste von **act** gelöscht und in die *rightAttached* Liste von **act** vorne angehängt. Ist die *own* Liste von **act** ebenfalls passend, wird anschließend die *leftAttached* Liste von **act** betrachtet. Auch hier werden alle unterstützenden Indices aus der *own* Liste von **new** gelöscht, wenn sie passend für die benötigten Indices ist. Abschließend werden dann die Indices der *rightCommon* Liste von **act** untersucht. Die Indices dieser Liste werden auf die *rightAttached* Liste von **act** und die *leftAttached* Liste dessen früheren rechten Nachbarknotens verteilt. Dabei wird ein Index aus der *rightCommon* Liste von **act** in dessen rightAttached Liste hinten angehängt, wenn er sich ebenfalls in der *own* Liste von **new** befindet. Ist dies der Fall wird der entsprechende Index aus der *own* Liste von **new** gelöscht und in die *leftAttached* Liste von **new** vorne angehängt. Befindet sich der Index nicht in der *own* Liste von **new**, so wird er in die *leftAttached* Liste des früheren rechten Nachbarknotens von **act** eingefügt. Die

rightCommon Liste von **act** und die *leftCommon* Liste dessen früheren rechten Nachbarknotens werden anschließend geleert. Ist die rightCommon Liste von **act** passend für die benötigten Indices, so werden die unterstützenden Indices der rechten Nachbarknoten von **act** ermittelt. Dies geschieht auf die gleiche Weise wie bei der Berechnung der Verdopplung. Auch hier wird die auf den Knoten implementierte Methode *insertLeftIndices* aufgerufen, um die Listen der rechten Nachbarknoten anzupassen.

Die Anordnung der Zeilen und Spalten kann nun aus den einzelnen Listen ermittelt werden, indem diese von links nach rechts durchlaufen werden. Je nachdem, ob der Benutzer bestimmte Zeilen oder Spalten verdoppelt angezeigt haben möchte, werden die als verdoppelt gekennzeichneten Indices interpretiert oder nicht. Wie letztendlich aus der Index-Anordnung die Heatmap-Anordnung generiert wird, wird in Abschnitt 3.4.2 erläutert.

Abschließend wird der Algorithmus an dem bereits vorgestellten Beispiel der Bicluster-Sortierung aus dem vorherigen Abschnitt durchgeführt und auf die einzelnen Schritte wird eingegangen. Die Datenmatrix besteht aus 10 Zeilen und 16 Spalten. Damit ergeben sich die Indexmengen: $\{0,1,2,3,4,5,6,7,8,9\}$ und $\{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15\}$. Die Bicluster seien wie im vorherigen Beispiel definiert:

$B_1: I_1 = \{2,7,8\}, J_1 = \{1,3,8,9\}$ $B_2: I_2 = \{1,2,5,7\}, J_2 = \{1,3,5,10\}$
$B_3: I_3 = \{1,3,6,9\}, J_3 = \{3,5,6,7,8\}$ $B_4: I_4 = \{2,8,9\}, J_4 = \{7,11,12\}$
$B_5: I_5 = \{0,4,6\}, J_5 = \{0,13,14,15\}$ $B_6: I_6 = \{1,5\}, J_6 = \{11,15\}$

Als Bicluster-Reihenfolge, wird die berechnete Reihenfolge aus dem vorherigen Beispiel genommen: $B_2 \rightarrow B_1 \rightarrow B_3 \rightarrow B_4 \rightarrow B_6 \rightarrow B_5$. Somit wird als erstes B_2 in die *IndexTree*-Objekte (eines für die Zeilen und eines für die Spalten) eingefügt. Dies kann ohne Verdopplung durchgeführt werden, da die Restmenge noch alle Indices enthält. Im nächsten Schritt wird B_1 links an B_2 in beiden Listen angehängt. Dies kann ebenfalls ohne Verdopplung durchgeführt werden. Die Listen für Zeilen und Spalten sowie die entstandenen Restmengen sind in Abbildung 31 dargestellt. Der aktuell eingefügt Knoten ist dick umrandet.

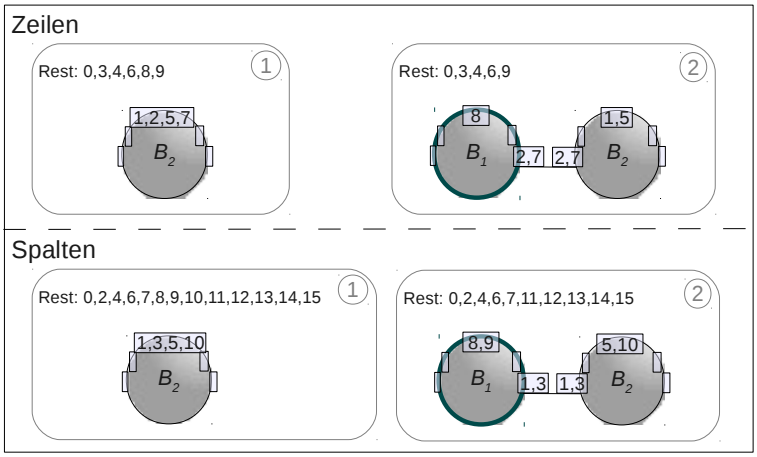


Abbildung 31: Zeilen- und Spalten-Listen nach Einfügen der ersten beiden Bicluster

Im nächsten Schritt wird B_3 in die Listen eingefügt. In der Zeilen-Liste wird er rechts an B_2 ohne Verdopplung angehängt. In der Spalten-Liste wird er links an B_1 angehängt, hierbei werden zwei Verdopplungen benötigt. Die Indices 5 und 3 werden als verdoppelt markiert. Würde er beispielsweise rechts an B_1 angehängt werden, müssten insgesamt drei Spalten verdoppelt werden, da innerhalb der Liste nur verdoppelt angefügt werden darf. 8 und 3 werden in die *Common*-Listen der Knoten eingefügt, da die Bicluster diese Indices gemeinsam haben.

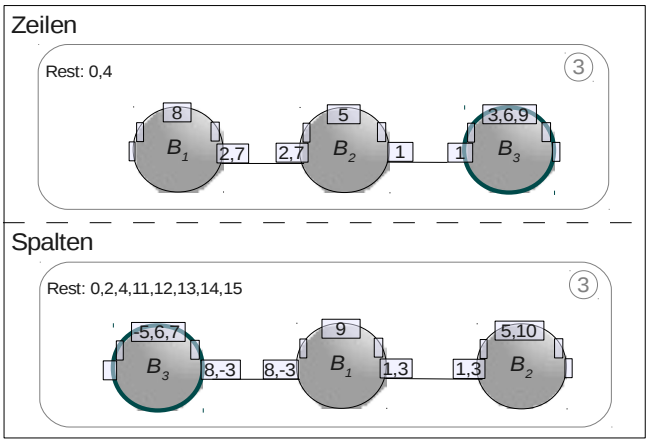


Abbildung 32: Zeilen- und Spalten-Listen nach Einfügen des dritten Biclusters

Im nächsten Schritt wird B_4 in beiden Listen links angehängt. In der Zeilen-Liste muss die 9 als verdoppelt markiert werden, da sie nicht mehr in der Restmenge ist. Die 8 wird dabei aus der *own* Liste von B_1 in die *Common*-Listen von B_4 und B_1 verschoben. Zusätzlich wird die 2 in die *rightAttached* Liste von B_1 verschoben, da sie ebenfalls von B_4 benötigt wird und somit vor der 7 kommen muss. In der Spalten-Liste kann B_4 ohne Verdopplung eingefügt werden.

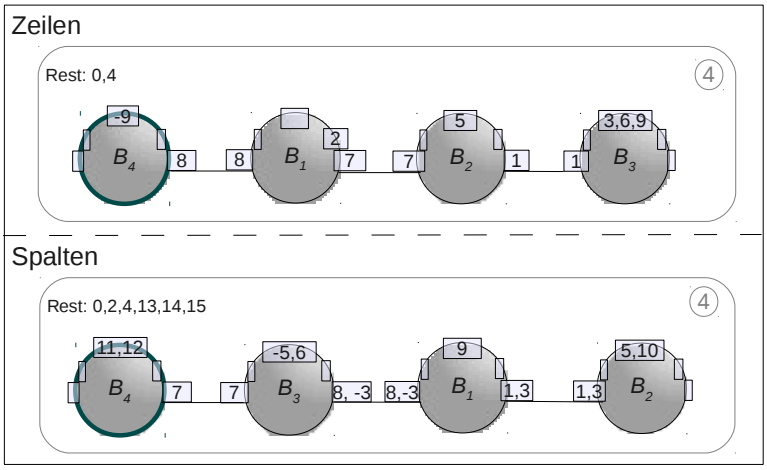


Abbildung 33: Zeilen- und Spalten-Listen nach Einfügen des vierten Biclusters

Als nächstes wird B_6 eingefügt. Dies kann in der Spalten-Liste ohne Verdopplung durchgeführt werden. In der Zeilen-Liste wird B_6 rechts an B_2 angehängt. Dies führt dazu, dass die *rightCommon* Liste von B_2 und die *leftCommon* Liste von B_3 aufgeteilt werden muss. Die 1 wird in die *leftAttached* Liste von B_6 eingefügt, die 5 aus der *own* Liste von B_2 kommt in die *rightAttached* Liste von B_2. In der Spalten-Liste wird B_6 an den linken Rand der Liste angehängt, die 11 aus der *own* Liste von B_4 wird in die entsprechenden *Common*-Listen von B_4 und B_6 geschrieben.

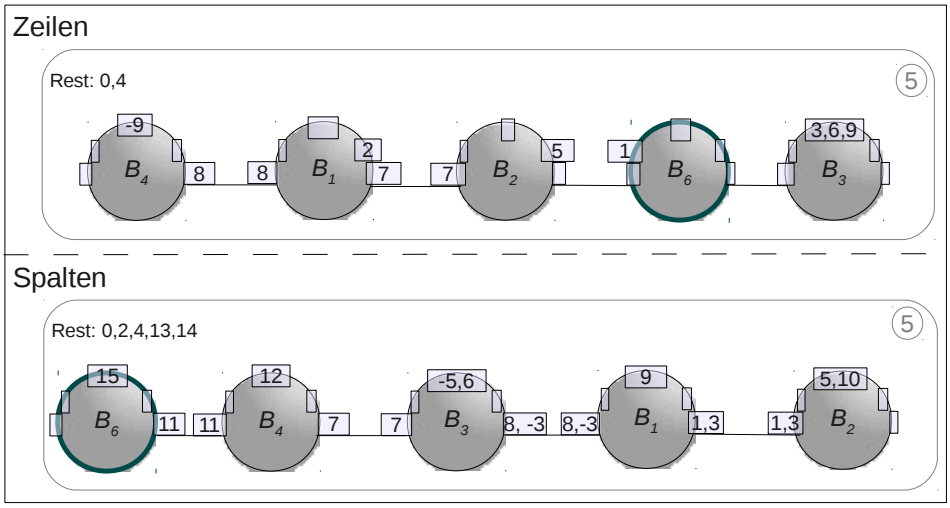


Abbildung 34: Zeilen- und Spalten-Listen nach Einfügen des fünften Biclusters

Zuletzt wird Bicluster B_5 in die Listen eingefügt. Dies kann in beiden Listen ohne Verdopplung durchgeführt werden. B_5 wird in der Zeilen-Liste rechts und in der Spalten-Liste links ange-

hängt. Die Restmenge ist in der Zeilen-Liste leer, in der Spalten-Liste bleiben die Indices 2 und 4 in der Restmenge.

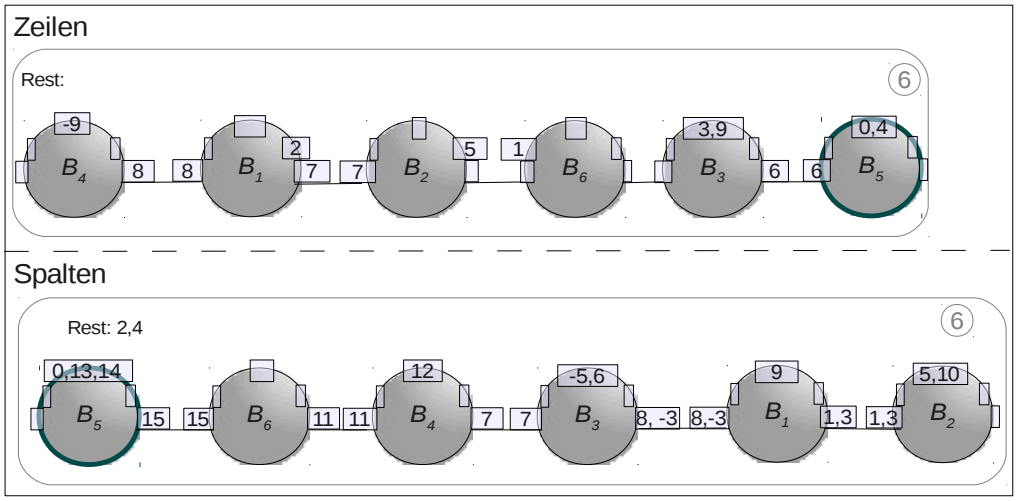


Abbildung 35: Die fertigen Listen nach Einfügen des letzten Biclusters

Aus der fertig aufgebauten Liste kann die Anordnung der Zeilen und Spalten bestimmt werden. Dabei werden die Listen von links nach rechts durchlaufen und die Indices in der entsprechenden Reihenfolge notiert. Die Indices in der Restmenge werden am Schluss der Reihenfolge hinzugefügt. Damit ergibt sich für die Zeilen:

$$\text{Zeilensortierung: } [-9,8,2,7,5,1,3,9,6,0,4]$$

Für die Spalten ergibt sich damit:

$$\text{Spaltensortierung: } [0,13,14,15,11,12,7,-5,6,8,-3,9,1,3,5,10,2,4]$$

Durch das Anhängen der in der Restmenge verbleibenden Indices werden automatisch die Zeilen und Spalten, die zu keinem Bicluster gehören, rechts bzw. unten in der Heatmap positioniert. Soll ein Bicluster durch verdoppelte Zeilen zusammenhängend dargestellt werden, müssen nur die entsprechenden (negativen) Indices in der Sortierung in die Heatmap eingefügt werden. Um die richtigen Indices zu bestimmen, kann aus den *IndexTree*-Objekten ein Offset für die entsprechenden Bicluster bestimmt werden. Dabei werden die entsprechenden Listen von links nach rechts durchlaufen und die Anzahl der Indices aufsummiert, bis der entsprechende Knoten (Vergleich der ID) gefunden wurde. Die Anzahl der Indices der *leftCommon* und der *leftAttached* Listen des gefundenen Knotens werden ebenfalls mitgezählt. Für den Offset von B_1 ergibt sich damit für die Zeilen ein Offset von 2 und für die Spalten 11.

3.4.2 Die Index-Liste der Heatmap und der Parallelen Koordinaten

Die entstandene Index-Liste ist darauf ausgelegt, dass der Benutzer beliebige Bicluster mit verdoppelten Zeilen und Spalten darstellen lassen kann, ohne dass die komplette Liste neu aufgebaut wird. Deshalb wird die Heatmap nicht anhand der entstandenen Listen der *IndexTree*-Objekte erstellt. Das *IndexMapping*-Objekt übernimmt zuvor noch die Verwaltung der verdoppelten Indices. Dabei muss es für den verdoppelt dargestellten Bicluster die als verdoppelt markierten Indices interpretieren, die in der IndexListe vom *IndexTree*-Objekt enthalten sind. Außerdem muss es die Zeilen- und Spalten-Offsets aller Bicluster neu berechnen. Die Index-Liste des *IndexTree*-Objekts verändert sich durch das Anzeigen verdoppelter Zeilen oder Spalten nicht und muss daher nicht neu berechnet werden.

Je nachdem welchen Bicluster der Benutzer verdoppelt dargestellt haben möchte, müssen verdoppelte Indices beachtet werden oder nicht. Alle nicht verdoppelt angezeigten Indices werden somit von dem aus den *IndexTree*-Objekten bestimmten Offset abgezogen. Würde beispielsweise B_3 verdoppelt angezeigt werden, so ergäben sich die Index-Abfolgen: [8, 2, 7, 5, 1, 3, 9, 6, 0, 4] für die Zeilen und [0, 13, 14, 15, 11, 12, 7, **5**, 6, 8, **3**, 9, 1, 3, 5, 10, 2, 4][16] für die Spalten. Da in den Zeilen die -9 ausgeblendet wird, ändert sich der Zeilen-Offset für B_1 zu 1, sein Spalten-Offset bleibt gleich, da alle negativen Indices dargestellt werden. Abschließend muss der Offset unter Umständen nach links verschoben werden. Dies liegt daran, dass der Offset zunächst so bestimmt wird, dass er auf die *own* Liste des entsprechenden Knotens „zeigt“. Da die Listen *leftAttached* und *leftCommon* und sogar die Listen der linken Nachbarn Indices enthalten können, die im Bicluster definiert sind, ist der bestimmte Offset im Allgemeinen zu groß. Deshalb wird die bisher bestimmte Index-Liste ausgehend vom aktuellen Offset so lange nach links durchlaufen, bis ein Index gefunden wird, welcher nicht im Bicluster definiert ist. Dies bedeutet für das oben beschriebene Beispiel, dass der Zeilen-Offset von B_1 zu 0 wird (8 ist ein Zeilenindex von B_1) und der Spalten-Offset wird zu 9.

Da sich die Parallelen Koordinaten auf dasselbe *IndexMapping*-Objekt beziehen, werden die Indices auf dieselbe Wiese bestimmt. Die Parallelen Koordinaten verwenden jedoch nur die Spaltenindices.

16 Verdoppelt angezeigte Indices sind fett geschrieben. In der Heatmap werden diese ebenfalls hervorgehoben.

3.5 Realisierung der Parallelen Koordinaten

Damit sich die Darstellung der Parallelen Koordinaten synchron zur Heatmap verhält, verwendet die Darstellung ebenfalls Referenzen auf die drei Objekte des Datenmodells: *Matrix*, *BiclusterList* und *IndexMapping*. Somit werden dieselben Daten angezeigt, die Darstellung kann auf veränderte Selektionen reagieren und die Achsen in den Parallelen Koordinaten sind in derselben Anordnung wie die Spalten der Heatmap. Dadurch wird ebenfalls gewährleistet, dass verdoppelt angezeigte Spalten in den Parallelen Koordinaten an der gleichen Position angezeigt werden wie in der Heatmap.

Zunächst werden die Zeilen der Datenmatrix in die Parallelen Koordinaten als Linien eingezeichnet, die keinem Bicluster angehören. Diese werden in einem transparenten Schwarz gezeichnet, wodurch Bereiche, in denen sich mehrere Linien überlagern, erkennbar sind. Dabei sind die Achsen stets in derselben Reihenfolge angeordnet wie die Spalten der Heatmap. Um nicht beim Zeichnen von jeder Linie alle Bicluster überprüfen zu müssen, wird bei der Bekanntmachung einer neuen Bicluster-Menge ein binärer Suchbaum aufgebaut, wodurch das Überprüfen im Mittel nicht mehr in quadratischer Zeit durchgeführt werden muss. Anschließend werden alle Linien gezeichnet, die einem Bicluster angehören, welcher nicht selektiert oder permanent hervorgehoben ist. Diese Linien werden ebenfalls in einem transparenten Schwarz dargestellt. Alle schwarzen Linien verlaufen grundsätzlich nicht durch Zentroide.

Als nächstes werden die Linien der Bicluster gezeichnet, die permanent hervorgehoben und nicht selektiert sind. Dabei werden die auf den Biclustern gespeicherten Farben für die entsprechenden Linien verwendet. Abschließend werden dann alle Linien der selektierten Bicluster in gelb gezeichnet. Dadurch sind die Linien in denselben Farben dargestellt wie die entsprechenden Rechtecke in der Heatmap, wodurch dem Benutzer die Zuordnung der einzelnen Linien zu den Biclustern erleichtert wird.

Dadurch dass die Linien in unterschiedlicher Reihenfolge gezeichnet werden, wird die Überlagerung der Linien vermindert, die im Fokus des Benutzers stehen. Linien eines selektierten Biclusters werden nur von anderen Linien eines (anderen) selektierten Biclusters überdeckt. Linien, die für einen nicht selektierten, permanent hervorgehobenen Biclusters gezeichnet werden,

können nur von Linien anderer selektierter oder permanent hervorgehobener Bicluster überdeckt werden. Die schwarz gezeichneten Linien stehen immer im Hintergrund der Darstellung. Diese können vom Benutzer ausgeblendet werden, um die Hervorhebung der Linien der Bicluster noch zu verstärken.

Die Zuordnung der Achsen zu den entsprechenden Spalten der Datenmatrix wird zum einen dadurch unterstützt, dass die Achsen in derselben Reihenfolge gezeichnet werden, wie sie in der Heatmap auftreten. Zum anderen kann der Benutzer optional die Spaltennamen auf den entsprechenden Achsen einblenden lassen.

Dadurch dass Heatmap und Parallele Koordinaten die Spalten-Anordnung synchron darstellen und die Rechtecke und Linien in denselben Farben dargestellt sind, ist es dem Benutzer möglich, die Linien der einzelnen Bicluster den Achsen in den Parallelen Koordinaten zuzuordnen (vgl. Abbildung 36). Dies kann jedoch sehr beschwerlich sein und viel Zeit in Anspruch nehmen, da die Bicluster unter Umständen nicht zusammenhängend dargestellt sind. Weiterhin wird die Zuordnung erschwert, wenn die Datenmatrix viele Spalten enthält und die Zuordnung über das Abzählen von Spalten vorgenommen wird und nicht über das Einblenden der Spaltennamen auf den Achsen.

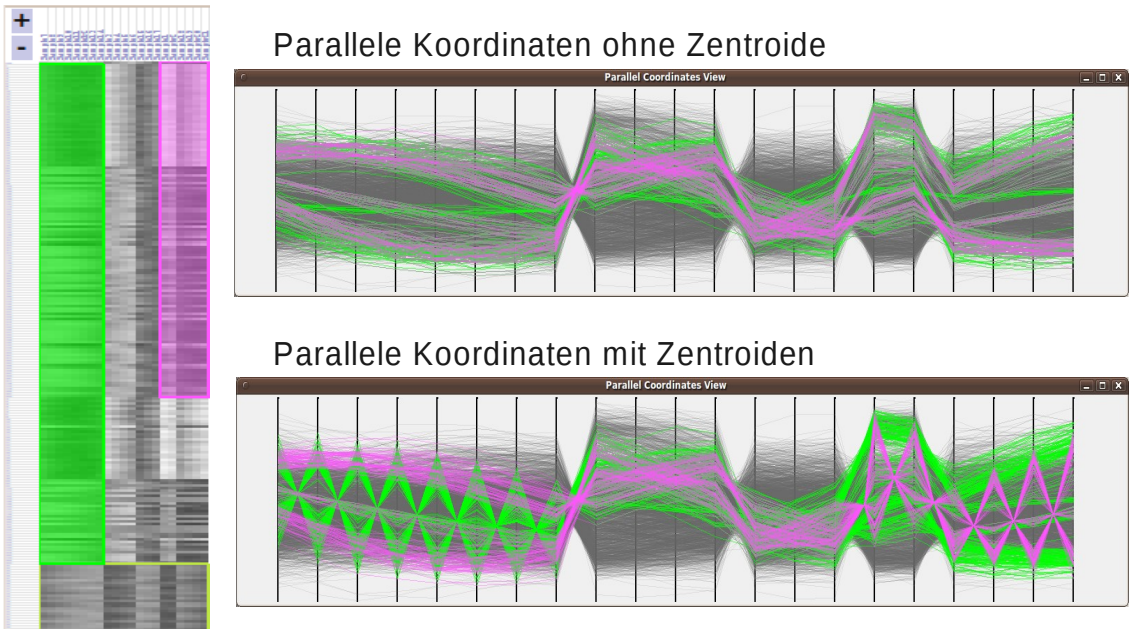


Abbildung 36: Zuordnung der Spalten mit Hilfe der Heatmap und Zentroiden

Um die Zuordnung zu beschleunigen und die Linien auch geometrisch in den Parallelen Koordinaten zu gruppieren, wird dem Benutzer zusätzlich die Möglichkeit angeboten, die Linien zwischen den Achsen zu „bündeln“ (vgl. Abbildung 36). Es wird dabei der in [Luo10] beschriebene Ansatz verwendet (auf einen kurvenlinigen Verlauf wird in dieser Darstellung verzichtet). Dabei werden für alle Linien zwischen jeweils zwei Achsen die Mittelwerte (Zentroide) gebildet. Alle Linien verlaufen dann durch diese Zentroide und werden somit „gebündelt“. In dieser Darstellung werden jedoch nicht zwischen allen Achsen Zentroide bestimmt und gezeichnet (Clustering), sondern nur für diejenigen Achsen, die dem darzustellenden Bicluster angehören (Biclustering). Es muss weiterhin unterschieden werden, ob beide Achsen diesem Bicluster angehören oder ob nur eine der beiden Achsen zu dem Bicluster gehört. Sind beide Achsen im Bicluster enthalten, wird der Zentroid mittig zwischen den beiden Achsen gezeichnet, ist nur eine Achse im Bicluster enthalten, so wird der Zentroid näher an die entsprechende Achse gelegt. Beispielsweise gilt für den x-Wert des Zentroiden (x_z) zwischen den benachbarten Achsen i und j, wobei nur die Achse j dem Bicluster angehört:

$$x_z = x_i + \frac{3}{4}(x_j - x_i), \text{ mit } x_j > x_i$$

Für den y-Wert des Zentroiden (y_z) werden die Mittelwerte der y-Werte der Linien auf den zwei benachbarten Achsen bestimmt und diese werden nochmals gemittelt. Damit gilt für y_z zwischen den Achsen i und j mit jeweils n Linien und den y-Werten auf i $\{y_{i,1}, y_{i,2}, \dots, y_{i,n}\}$ und den y-Werten auf j $\{y_{j,1}, y_{j,2}, \dots, y_{j,n}\}$:

$$y_z = \frac{1}{2} \cdot \left(\frac{1}{n} \sum_{k=1}^{n} (y_{i,k}) + \frac{1}{n} \sum_{k=1}^{n} (y_{j,k}) \right)$$

Durch die Darstellung der Zentroide zwischen den Achsen kann weiterhin der Verlauf des Mittelwerts des Biclusters gut verfolgt werden. In Abbildung 36 kann in der Darstellung ohne Zentroide nur schwer erkannt werden, wie sich die Bicluster über die Spalten hinweg verhalten. In der Darstellung mit Zentroiden kann man beispielsweise gut erkennen, dass die Werte des grünen Biclusters von links nach rechts stetig abnehmen, beim violetten Bicluster ist zu sehen, dass seine Werte im Mittel zunächst ansteigen und dann stark abfallen.

Einen großen Vorteil, den die synchrone Darstellung in Form der Heatmap und der Parallelen Koordinaten bietet, ist das zusammenhängende Darstellen von Biclustern durch Verdopplung. In

Abbildung 37 und Abbildung 38 sind aus dem Yeast-Datensatz zwei der 100 von CHENG und CHURCH gefundenen Biclustern hervorgehoben, einer durch permanentes Einfärben in Rot und einer durch Selektieren. Der rot eingefärbte Bicluster erstreckt sich über alle Spalten der Datenmatrix, der selektierte ist durch die Anordnung der 100 Bicluster in der aktuellen Sortierung nicht zusammenhängend dargestellt (Abbildung 37). Dadurch sind die Achsen des Biclusters in der Darstellung der Parallelen Koordinaten nicht nebeneinander gezeichnet. Das Verfolgen der Werte der Linien auf den Achsen, die zum Bicluster gehören, ist zwar möglich, jedoch aufwendig. Auch der Vergleich der beiden Bicluster an den Stellen, an denen sie sich überlappen, ist schwer zu erfassen.

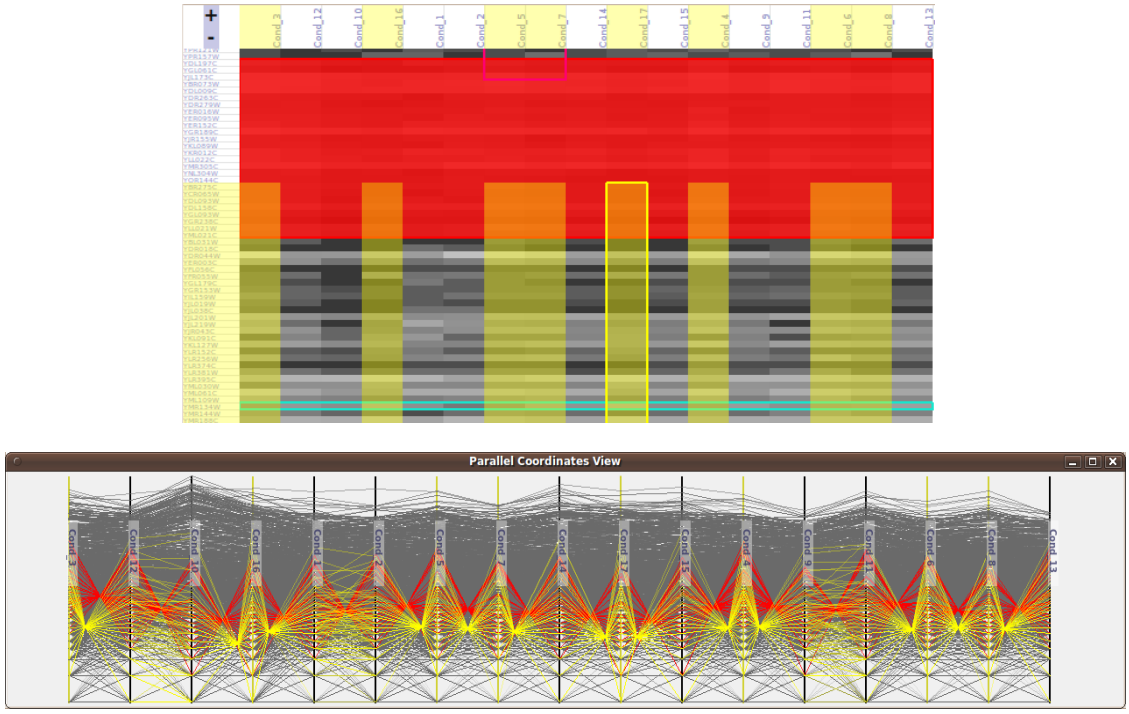


Abbildung 37: Yeast Datensatz mit zwei hervorgehobenen Biclustern ohne Verdopplung in Darstellung der Heatmap und der Parallelen Koordinaten mit Zentroiden

In Abbildung 38 ist der selektierte Bicluster durch Einfügen von verdoppelten Spalten (und Zeilen) in der Heatmap zusammenhängend dargestellt und somit liegen die Achsen, die dem Bicluster angehören in der Darstellung der Parallelen Koordinaten nebeneinander. Dadurch ist beispielsweise besser erkennbar, dass die Werte des selektierten Biclusters im Mittel unterhalb derer des rot eingefärbten Biclusters liegen. Weiterhin ist zu erkennen, dass sich die Bicluster in

etwa gleich verhalten. Verändern sich die Werte des roten Biclusters, so ändern sich die Werte des selektierten Biclusters in der gleichen bzw. abgeschwächten Weise.

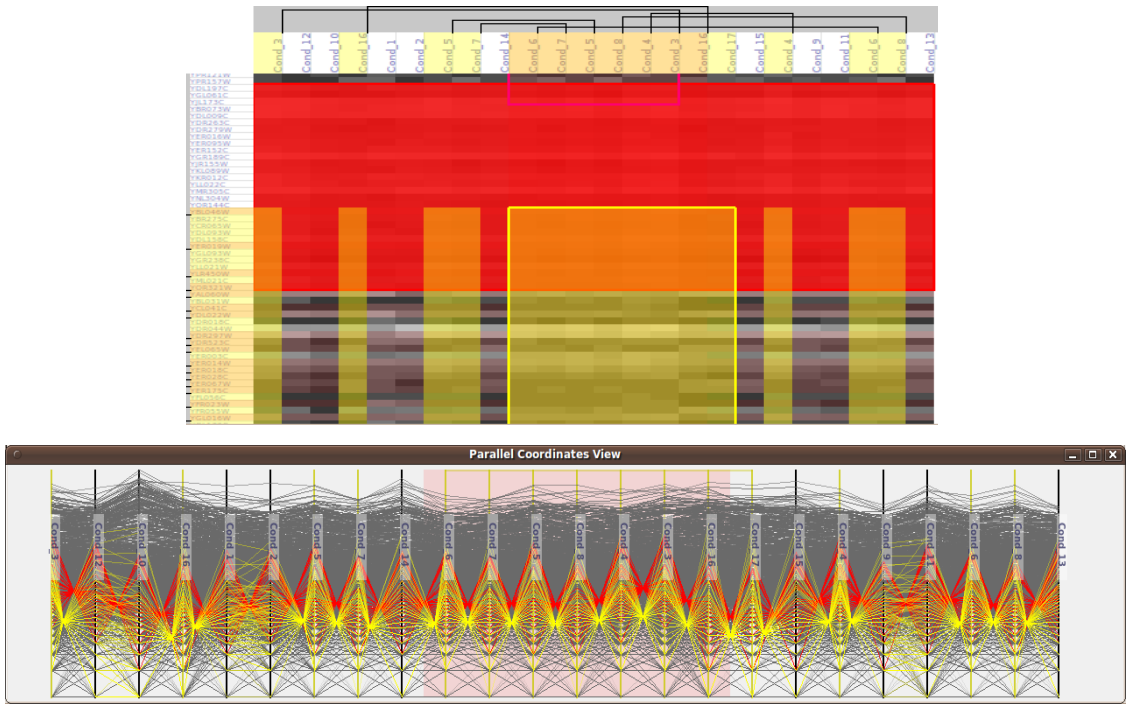


Abbildung 38: Yeast Datensatz mit zwei hervorgehobenen Biclustern mit Verdopplung in Darstellung der Heatmap und der Parallelen Koordinaten mit Zentroiden

Die verdoppelten Achsen werden in den Parallelen Koordinaten ebenfalls in einem transparenten Rot hervorgehoben. Die Einfärbung erfolgt in gleicher Weise wie die in der Heatmap. Sind über den Spaltennamen innerhalb der Heatmap die Abhängigkeiten zwischen den verdoppelten und originalen Spalten eingeblendet, so kann der Benutzer diese auf die Parallelen Koordinaten übertragen. Auf eine gesonderte Darstellung dieser Abhängigkeiten innerhalb der Parallelen Koordinaten wird jedoch verzichtet.

4 Diskussion

4.1 Die Pilotstudie

Am 16. März 2010 wurde im Technologiezentrum in Stuttgart-Vaihingen eine Pilotstudie mit insgesamt 4 Teilnehmern durchgeführt. Sie sollte dazu dienen, erste Erkenntnisse über die Bedienbarkeit der Anwendung und die Aussagekraft der Visualisierung der Heatmap zu gewinnen. Die Darstellung der Parallelen Koordinaten war nicht Bestandteil dieser Studie. Im Fokus der Studie standen vor allem die Fragestellungen:

- Wie gut sind Bicluster in der Darstellung zu erkennen und zu unterscheiden?
- Wie gut lassen sich Zellen den einzelnen Biclustern zuordnen?
- Wie gut lassen sich Überlappungen der Bicluster erkennen?
- Wie intuitiv ist die Bedienung der Anwendung? In wie weit müssen Bedienung und Darstellung genauer erklärt werden?

Zunächst wurde den Teilnehmern eine etwa fünf- bis achtminütige Einführung in Bicluster und die Bedienung der Anwendung anhand eines Beispieldatensatzes gegeben. Anschließend konnten sie die Funktionen der Anwendung selbst testen. Die Fragen zu den zu bearbeitenden Datensätzen wurden den Teilnehmern nur mündlich gestellt und deren Antworten wurden notiert. Die Teilnehmer wurden angewiesen ihre Vorgehensweise und Eindrücke während der Bearbeitung der Aufgaben zu äußern. Eine genaue Beschreibung der Durchführung der Studie, sowie die Aufgabenstellungen und Erklärungen befinden sich in Anhang B ab Seite 103.

Die Heatmap sollte vor allem darauf getestet werden, ob die Darstellung die Lage der Bicluster gut repräsentieren kann. Eine Überprüfung in wie weit die Darstellung die Art der Bicluster

verdeutlichen kann war nicht Bestandteil der Studie. Deshalb wurden in der Studie vorwiegend zufällig generierte Datensätze mit zufällig darauf verteilten homogenen Biclustern untersucht. Die Datensätze befinden sich auf beiliegender CD im Ordner „Pilotstudie/data“.

Für die Generierung der zufälligen Datensätze wurde zunächst eine (1000 x 20)-Datenmatrix mit normalverteilten Datenpunkten erstellt. Danach wurden mit unterschiedlichem Überlappungsgrad Bicluster auf der *Matrix* definiert. Anschließend wurde für jeden definierten Bicluster der Mittelwert ermittelt. Dieser wurde dann um einen zufällig definierten Wert verschoben und die Datenwerte innerhalb des Biclusters wurden homogenisiert. Zusätzlich wurde bei der Homogenisierung der Datenwerte ein Rauschen auf die Datenpunkte verteilt. Bei Datenpunkten, die zu mehreren Biclustern gehören, wurde die Homogenisierung mit dem Mittelwert aller entsprechenden Bicluster durchgeführt.

Aufgabe 1 bis 3 dienen zur Überprüfung, wie gut sich Bicluster in den einzelnen Darstellungsmodi (standard, show all biclusters, select all) der Heatmap voneinander unterscheiden lassen und wie gut die Überlappungen der Bicluster in den Darstellungen wiedergegeben werden. Zusätzlich sollte geprüft werden, in wie weit die Heatmap einen Überblick über alle Bicluster bereitstellt. Alle 3 Aufgaben beziehen sich auf dieselben zufällig erstellten Datensätze. Aufgabe 4 befasst sich mit der Analyse von nicht überlappenden Biclustern und der Zuordnung der Bicluster zu einzelnen Spalten. Aufgabe 5 basiert auf überlappenden Biclustern und behandelt die Fragestellung, in wie weit die Standard-Anordnung ausreichend ist, Überlappungen den einzelnen Biclustern und Zellen zuzuordnen und in wie weit erkannt wird, dass Bicluster nicht zusammenhängend dargestellt werden. Aufgabe 6 soll überprüfen, in wie weit der Benutzer die Funktionen der Anwendung richtig benutzt und nicht zusammenhängende Bicluster richtig erkannt werden.

4.1.1 Auswertung der Studie

Die Ergebnisse der Studie sind in Abbildung 39 dargestellt. Die Balken stellen die Summe der korrekt gegebenen Antworten aller Teilnehmer pro Fragestellung dar. Die maximale Anzahl an korrekten Antworten ist daher 4. Dabei wurde eine Antwort nur dann als korrekt gewertet, wenn sie ohne Fehler ist. Teilweise richtige Antworten wurden somit als nicht korrekt gewertet. Die rote Linie zeigt die durchschnittliche Zeit in Sekunden an, die pro Frage benötigt wurde. Da an dieser Studie nur vier Teilnehmer beteiligt waren, können keine statistischen Aussagen über die

Darstellung bzw. die Anwendung getroffen werden. Jedoch können aus den Zeiten wichtige Erkenntnisse über die Dauer der Aufgaben gewonnen werden und aus der Summe der korrekten Antworten können für die unterschiedlichen Aufgabentypen Trends abgelesen werden. Zusätzlich bilden die notierten Kommentare der Teilnehmer eine gute Basis für eine erste Analyse der Bedienbarkeit der Anwendung.

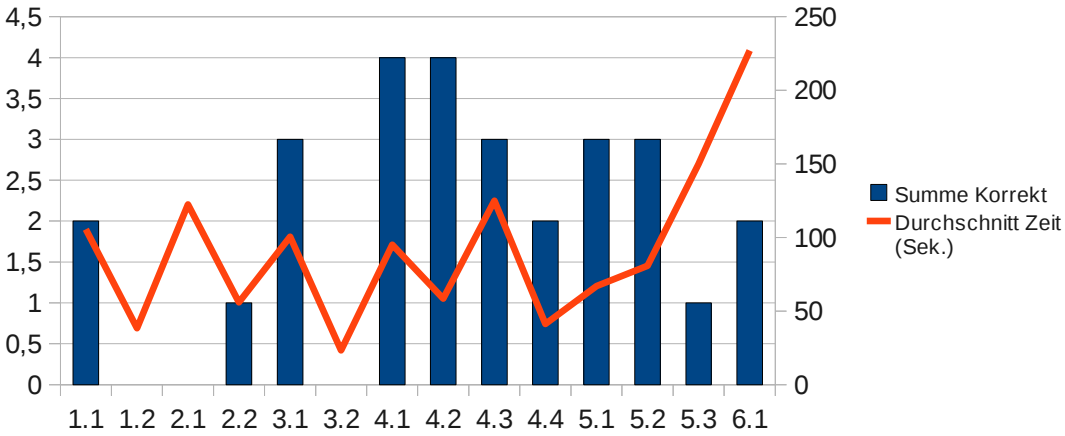


Abbildung 39: Diagramm mit korrekten Antworten und der durchschnittlichen Bearbeitungszeit

Aus den Ergebnissen aus den Aufgaben 1 bis 3 lassen sich bereits Trends für die einzelnen Darstellungen ableiten. In diesen Aufgaben erhielt der Teilnehmer jeweils einen Ausdruck einer Darstellung von mehreren Biclustern: in Aufgabe 1 die Standard-Darstellung, in Aufgabe 2 eine Darstellung, in der auch die gestrichelten Rechtecke eingeblendet sind und in Aufgabe 3 die Standard-Darstellung, in der alle Bicluster selektiert wurden. Die Fragen waren in allen drei Aufgaben gleich. In Frage 1 sollte er alle Bicluster zählen und in Frage 2 sollte er abschätzen in welcher Darstellung die gesamte Überlappung am stärksten ist.

Zunächst kann festgehalten werden, dass sich die Darstellung mit den gestrichelten Rechtecken wohl nicht zum Unterscheiden bzw. Abzählen (Frage 2.1) der einzelnen Bicluster eignet (keine korrekte Antwort), da die Darstellungen zu überladen präsentiert werden. Überraschend ist hingegen, dass der Überlappungsgrad in dieser Darstellung auch weitestgehend falsch zugeordnet wurde. Dies liegt vor allem daran, dass die Teilnehmer die zusammenhängenden Überlappungen stärker gewichtet haben, als die Überlappungen, die durch die gestrichelten Rechtecke verdeutlicht werden. Weiterhin ist zu erkennen, dass sich die Darstellungen aus Aufgabe 1 und 3

(standard, select all) gut zum unterscheiden der Bicluster eignen. Die falschen Antworten beruhen darauf, dass die Darstellung falsch interpretiert wurde. Deshalb ist für eine nächste Studie eine genauere Erläuterung der einzelnen Darstellungen zu empfehlen. Vor allem auf die Farbgebung der einzelnen Bicluster muss bei der Einführung genauer eingegangen werden, da mehrere Teilnehmer die Bicluster nach Farben gezählt haben. Auch hier ist der Trend zu erkennen, dass sich die Darstellungen nicht zum Erkennen des gesamten Überlappungsgrades der Bicluster eignen. Die Antworten aus Frage 1.2 sind erwartungsgemäß alle nicht korrekt, was die Vermutung untermauert, dass sich die Standard-Ansicht nicht zur Darstellung des gesamten Überlappungsgrades eignet. Ein sehr unerwartetes Ergebnis bilden jedoch die Antworten aus Frage 3.2. Hier wurde der Überlappungsgrad der Bicluster von keinem Teilnehmer richtig erkannt. Dies kann daran liegen, dass die Hervorhebung der selektierten Bicluster in Gelb in ausgedruckter Form nicht deutlich erkennbar ist, wodurch die sehr kurze Bearbeitungszeit (ca. 25 Sekunden) zu erklären ist. Die meisten Teilnehmer haben auch hier nur die Überlappungen der Hauptrechtecke gewertet. Für eine erneute Studie ist zu überlegen, ob die Darstellungen auf einem Bildschirm präsentiert werden sollten, um diese mögliche Fehlerquelle auszuschließen. Für die Bearbeitungszeit der einzelnen Aufgaben kann im Mittel etwa drei Minuten pro Aufgabe veranschlagt werden.

Die Antworten aus Aufgabe 4 sind weitestgehend korrekt und Aufgabe 4 kann insgesamt mit ca. 7 Minuten veranschlagt werden. In dieser Aufgabe wurde der Yeast-Datensatz mit nicht überlappenden Biclustern in die Anwendung geladen. Die Teilnehmer sollten folgende Fragestellungen dazu beantworten. Frage 1: *„Welcher ist der größte Bicluster?“*. Frage 2: *„Gibt es überlappende Biclsuter? Wenn ja, welche?“*. Frage 3: *„Welche Spalte enthält die meisten Bicluster?“*. Frage 4: *„Gibt es Spalten, die keine Bicluster enthalten?“*. Die Varianz bei der Bearbeitungszeit der einzelnen Teilnehmer ist jedoch sehr hoch. Dies liegt vor allem daran, dass einige Teilnehmer die Navigationsliste nicht beachtet haben und somit durch ständiges Zoomen für die Bearbeitung der Aufgabe mehr Zeit benötigt haben. Deshalb sollte bei einer erneuten Studie die Navigationsliste genauer erläutert und hervorgehoben werden, um die Bearbeitungszeit zu homogenisieren.

Für Aufgabe 5 wurde ein zufällig erstellter Datensatz mit zufällig darauf verteilten, überlappenden Biclustern in die Anwendung geladen. Die Fragen 1 *„Mit welchem Bicluster hat Biclus-*

ter B_8 die meisten Überlappungen?“ und 2 *„Finden Sie eine Zelle, die in mindestens drei Biclustern liegt und geben Sie die IDs der Bicluster an.“* wurden ebenfalls weitestgehend richtig beantwortet, dies lag vor allem daran, dass die Fragen mit der Standard-Visualisierung beantwortet werden konnten und die Teilnehmer dazu nicht viele Funktionen der Anwendung benutzen mussten. Hierbei wurde oft die Kritik geäußert, dass sich die Selektionen in den Zeilen- und Spaltennamen nicht akkumulieren und somit die Zuordnung der Zellen zu den Zeilen- und Spaltennamen nicht optimal unterstützt wird. In Frage 3 sollten die Teilnehmer einen bestimmten Bicluster (B_9, dieser wird in der Standard-Darstellung nicht zusammenhängend angezeigt) genauer untersuchen und alle Bicluster finden, mit denen dieser überlappt. Zusätzlich sollten sie die Zeilen- und Spaltennamen der Zellen angeben, in denen die Überlappung stattfindet. Bei der Bearbeitung von Frage 3 fiel auf, dass die meisten Teilnehmer die Funktionen der Anwendung falsch oder gar nicht benutzt haben und somit die Frage nicht korrekt beantworten konnten. Die Vielzahl der Teilnehmer hat die zu benutzenden Funktionen (show all biclusters, resort starting with, show duplicated) nicht mehr beachtet. Um dieser Tatsache entgegenzuwirken, sollten die Teilnehmer bei der Einführung der Anwendung bereits Aufgaben selbstständig bearbeiten, um die Funktionalitäten besser anwenden zu können. Aufgabe 5 weist vor allem bei den Fragen 2 und 3 eine große Varianz in der Bearbeitungszeit auf, dies ist jedoch auf die unterschiedliche Benutzung der Funktionen der Anwendung zurückzuführen. Es ist zu untersuchen, in wie weit sich eine genauere und interaktive Einführung der Teilnehmer homogenisierend auf die Bearbeitungszeit auswirkt. Durch die hohe Varianz in der Bearbeitungszeit ist schwer abzuschätzen, wie lange die Bearbeitung von Aufgabe 5 im Mittel benötigt. Generell ist jedoch erkennbar, dass die Bearbeitung von Frage 3 am meisten Zeit benötigt.

Die Frage aus Aufgabe 6 (auch hier wurde ein Zufallsdatensatz mit zufällig darauf verteilten, überlappenden Biclustern geladen) benötigt mit ca. vier Minuten erwartungsgemäß die längste Bearbeitungszeit. Hier sollten die Teilnehmer denjenigen Bicluster finden, der mit den meisten anderen Biclustern mindestens eine Zelle gemeinsam hat. Diese Fragestellung wird von keiner der Funktionen in der Anwendung komplett unterstützt. Es sollte vielmehr getestet werden, in wie weit die Interkationsmöglichkeiten und das Benutzen mehrerer Funktionen der Anwendung ausreichen, um diese Aufgabe zu lösen. Dazu müssen alle Bicluster auf Überlappung untersucht werden. Dies kann durch Umsortierung der Heatmap oder durch das Anzeigen verdoppelter Zei-

len und Spalten erreicht werden. Auch hier ist eine große Varianz in der Bearbeitungszeit festzustellen. Die große Varianz und die nicht korrekten Antworten sind in erster Linie auf nicht bzw. falsch benutzte Funktionen der Anwendung zurückzuführen. Auffallend ist weiterhin, dass alle Teilnehmer nur die in der Standard-Darstellung zusammenhängend dargestellten Bicluster untersucht haben. Eine genaue Bearbeitung der Fragestellung würde daher weitaus länger dauern.

Weiterhin kann festgehalten werden, dass die Einführungs- und Einarbeitungszeit von ca. zehn Minuten viel zu kurz gewählt wurde. Viele der falschen Antworten sind auf ein falsches Verständnis der Darstellung und auf nicht benutzte Funktionen der Anwendung zurückzuführen, wodurch die Ergebnisse einer Benutzerstudie verfälscht werden könnten. Weiterhin sollte den Teilnehmern vor Bearbeitung der Aufgaben ein kleiner Datensatz gegeben werden, zu dem sie selbstständig Aufgaben bearbeiten sollten, um die Funktionen der Anwendung für die Durchführung der Studie besser nutzen zu können.

Es hat sich als sehr nützlich und zeitsparend erwiesen, dass die Teilnehmer die Fragen nur mündlich beantworten mussten. Dadurch konnten sich die Teilnehmer auf die Bearbeitung der Aufgaben konzentrieren und bei möglichen Missverständnissen in der Fragestellung konnte nochmals näher auf die Frage eingegangen werden. Ebenfalls ergaben die nebenher geäußerten Kommentare großen Aufschluss darüber, in wie weit die Darstellungen und Funktionen der Anwendung richtig verstanden und angewendet wurden. Weiterhin konnten einige Erkenntnisse zur Benutzerfreundlichkeit der Anwendung und der Darstellung gesammelt werden.

Die Hauptkritikpunkte waren:

- die x- und y-Achsen sind nicht unterschiedlich skalierbar
- die Selektionen in den Zeilen- und Spaltennamen sollten akkumuliert werden
- die unterschiedliche Einfärbung der Bicluster ist kaum wahrnehmbar
- die Labels sollten zentriert dargestellt werden
- bei hell eingefärbten Biclustern kann man deren ID in der Navigationsliste nicht lesen
- „ich weiß nicht, wie mich das Programm dabei unterstützen soll“.

Die Pilotstudie hat gezeigt, dass die Darstellung der Bicluster in einer Heatmap prinzipiell in der Lage ist, mehrere Bicluster anzuzeigen, die Bicluster den einzelnen Zeilen und Spalten zuzuordnen und bestimmte Bicluster genauer zu untersuchen. In wie weit diese Untersuchungen effektiv und in kurzer Zeit ausgeführt werden können, konnte aufgrund der geringen Teilnehmerzahl und der zu kurzen Einarbeitungszeit nicht überprüft werden und sollte in einer größeren Benutzerstudie untersucht werden. Weiterhin ist durch die Studie erkennbar, dass die Untersuchung von Überlappungen bestimmter Bicluster schneller durchzuführen ist als die Untersuchung einer bestimmten Überlappung (z. B. die größte) innerhalb der gesamten Heatmap.

Die angesetzte Zeit von 30 Minuten pro Teilnehmer für die Einführung und die Bearbeitung der Aufgaben wurde bei jedem der Teilnehmer mit bis zu 10 Minuten überschritten. Für eine größere Benutzerstudie mit einer genaueren Einarbeitung sollten also mindestens 50 Minuten pro Teilnehmer veranschlagt werden.

4.2 Vergleich zu den bisherigen Visualisierungsansätzen

Ein Vergleich zu den bisherigen Visualisierungsansätzen wird dadurch erschwert, dass die Visualisierungen auf unterschiedlichen Datensätzen realisiert werden müssen, da die für diese Anwendung benutzten Datensätze nicht in allen Visualisierungen geladen werden konnten. Außerdem stellen die Visualisierungen unterschiedliche Berechnungsmethoden für die Bicluster bereit. Lediglich für die Visualisierungen von *Biclust* in *R* konnte derselbe Datensatz mit denselben darauf berechneten Biclustern verwendet werden. Bei den anderen Visualisierungsansätzen wurde versucht, ähnliche Daten zu laden und mit vergleichbaren Berechnungsmethoden in Bicluster zu unterteilen. Die zu vergleichenden Daten wurden geladen und der Bicluster-Algorithmus wurde mit annähernd denselben Parametern wie in den anderen Anwendungen ausgeführt. Abbildung 40 zeigt die entstandenen Darstellungen. Dies sind zum einen die Heatmap links und zum anderen die Parallelen Koordinaten rechts. Die Abbildungen aus Abschnitt 2.5 zeigen ähnliche Bicluster auf vergleichbaren Datensätzen.

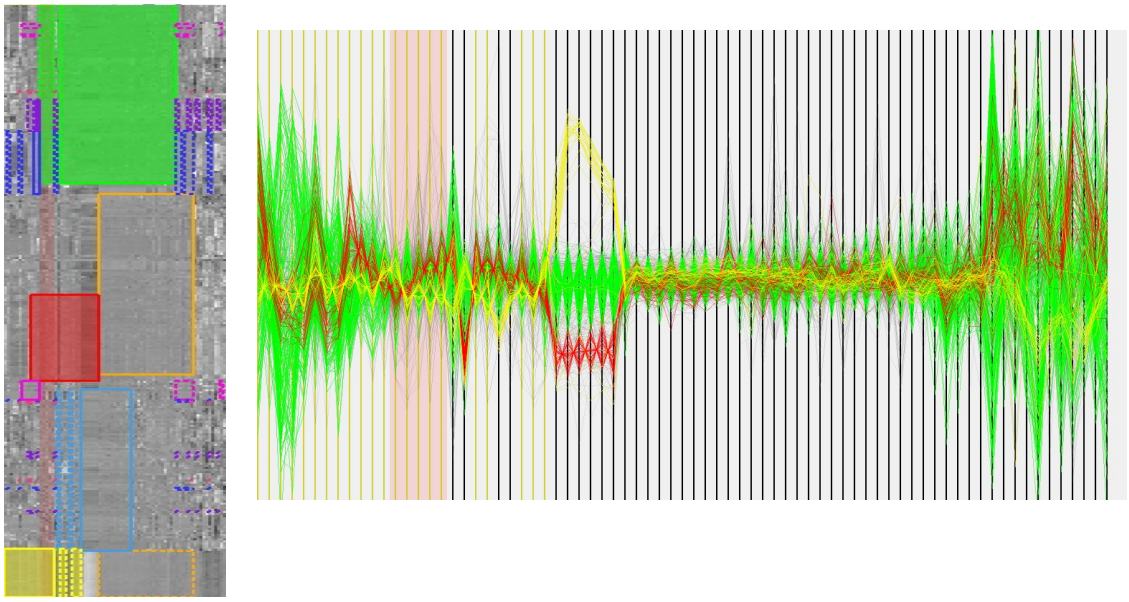


Abbildung 40: Bicluster-Visualisierung in *Bicluster Viewer* mit Heatmap und Parallelen Koordinaten

Der in den anderen Darstellungen gezeigte Bicluster (Bicluster-ID: 1) ist hier grün eingefärbt. Zusätzlich sind noch weitere Bicluster in der Darstellung angezeigt. Exemplarisch wurde ein Bicluster in rot eingefärbt, ein anderer wurde selektiert. Diese Darstellung der Heatmap bietet

den großen Vorteil, dass alle Bicluster dargestellt werden und auf ihnen Interaktionen ausgeführt werden können. So können z. B. bestimmte Bicluster dauerhaft hervorgehoben werden; andere können selektiert werden. Durch die gleichzeitige Darstellung der Bicluster in der Heatmap ist zusätzlich erkennbar, dass die Bicluster nicht überlappen und lediglich nur Zeilen oder nur Spalten gemeinsam haben. Dies ist vor allem in den Visualisierungen von *Biclust*[17], *BicAT* und *BiVisu* überhaupt nicht erkennbar, da sie in der Heatmap und den Parallelen Koordinaten nur einen Bicluster darstellen. Nur die Anwendung von GROTHAUS et al. [Gro06] besitzt ebenfalls die Funktionalität mehrere Bicluster in einer Heatmap darzustellen. Alle anderen Visualisierungsansätze beschränken sich darauf, einen oder mehrere exklusive Bicluster in einer Heatmap darzustellen. Der Visualisierungsansatz von GROTHAUS et al. bietet jedoch innerhalb der Darstellung keine Interaktionsmöglichkeiten und somit ist sie meist durch eine Vielzahl von verdoppelten Zeilen und Spalten überladen, wenn alle Bicluster angezeigt werden sollen.

Die in dieser Diplomarbeit entwickelte Anwendung zeigt zu Beginn keine verdoppelten Zeilen oder Spalten an, wodurch die Bicluster unter Umständen nicht zusammenhängend dargestellt werden. Der Benutzer hat jedoch die Möglichkeit Bicluster durch Verdopplung zusammenhängend darzustellen. Eine weitere wichtige Funktion die diese Darstellung der Heatmap anbietet, ist das Neusortieren der Zeilen und Spalten der Heatmap. Damit ist es dem Benutzer möglich ausgewählte Bicluster zusammenhängend darzustellen und diese auf ihre Überlappung zu anderen Biclustern anzeigen zu lassen. Damit bietet diese Anwendung als einzige die Möglichkeit, die Anordnung der Bicluster innerhalb der Heatmap interaktiv zu verändern und somit bestimmte Bicluster genauer auf Überlappungen zu untersuchen.

Auch *BicOverlapper* zeigt in der Darstellung der Heatmap nur einen Bicluster an bzw. es sind bei mehreren Biclustern die Zuordnungen der Zeilen und Spalten zu den einzelnen Biclustern nicht mehr möglich. Durch die Darstellungen im *Overlapper* sind jedoch Überlappungen bzw. keine Überlappungen gut erkennbar und deutlich hervorgehoben. Der dabei angezeigte Graph weist große Vorteile gegenüber der Darstellung in der Heatmap auf, wenn die Gesamtheit der Überlappungen aller Bicluster zu untersuchen ist. Da Genexpressionsdaten sehr viele Zeilen und nur wenig Spalten haben, skaliert die Darstellung der Heatmap im *Bicluster Viewer* für viele

17 Der Bubble-Plot zeigt mit den überschneidenden Kreisen nicht die Überlappungen der Bicluster an. Die Kreismittelpunkte entsprechen der gemittelten Lage der Bicluster, der Radius deren Größe.

stark überlappende Bicluster nicht. Dies liegt vor allem daran, dass beim Anordnen der Zeilen und Spalten schnell alle Spalten „verbraucht“ sind und die Bicluster in die bereits fest angeordneten Spalten eingefügt werden müssen. Somit entstehen bei der Standard-Darstellung, bei der viele Bicluster angezeigt werden sollen, meist nur (1x1)-große Rechtecke, wodurch die Überlappungen nur schwer erkennbar sind. Zwar kann man alle (gestrichelten) Rechtecke einblenden lassen, jedoch wird die Darstellung bei einer großen Anzahl von Biclustern sehr unübersichtlich. Die entwickelte Darstellung eignet sich daher eher zur Untersuchung der Überlappung eines bestimmten Biclusters als zur Darstellung der gesamten Überlappungen. Dies zeigen auch die Ergebnisse der Pilotstudie.

Da sich die Spalten in der Heatmap entsprechend den zu untersuchenden Biclustern anpassen lassen, können in der Darstellung der Parallelen Koordinaten mehrere zusammenhängende Bicluster angezeigt werden (vgl. Abbildung 40). Die meisten Anwendungen (*Biclust*, *BicAt*, *BiVisu*) zeigen in ihren Darstellungen der Parallelen Koordinaten nur einen Bicluster an, wodurch ein Vergleich mehrerer Bicluster innerhalb der Parallelen Koordinaten nicht möglich ist. Alle Anwendungen heben die Linien des Biclusters hervor und färben die dem Bicluster zugehörigen Achsen ein (*BicAT*) oder blenden die Achsen, die dem Bicluster nicht angehören aus (*Biclust*, *BiVisu*). Auch hier bietet *BicOverlapper* den einzigen Ansatz zur Darstellung mehrerer Bicluster. Es werden alle Linien der selektierten Bicluster rot eingefärbt und die Achsen werden entsprechend umsortiert. Liniensegmente, die zu den Achsen der selektierten Bicluster gehören, werden durch eine hellere Einfärbung hervorgehoben. Es ist jedoch fraglich, in wieweit die einzelnen Linien den Achsen der Bicluster zugeordnet werden können, wenn mehrere Bicluster dargestellt werden.

Durch die synchrone Darstellung der Bicluster und der Parallelen Koordinaten in *Bicluster Viewer* ist es dem Benutzer möglich die einzelnen Achsen der Parallelen Koordinaten eindeutig einem Bicluster zuzuordnen. Zusätzlich werden zwischen den Achsen Zentroide gezeichnet, wodurch die Zuordnung erleichtert wird. Mit Hilfe der Zentroide kann der Benutzer ebenfalls schnell den Mittelwert aller Zeilen innerhalb einer Spalte erfassen und somit den mittleren Verlauf der Zeilen eines Biclusters schnell ermitteln. Einen weiteren Vorteil, den die entwickelte Darstellung liefert, ist, dass Bicluster zusammenhängend dargestellt werden können. Somit können die Werte mehrerer Bicluster bei ihren Überschneidungen in den Spalten miteinander vergli-

chen werden. Der Überzeichnung der Bicluster kann entgegengewirkt werden, indem einzelne Bicluster unterschiedlich eingefärbt werden. Zusätzlich wird der aktuell gewählte Bicluster immer als letztes gezeichnet, wodurch seine Lininenverläufe nicht durch Linien anderer Bicluster überdeckt werden können. Eine solche Funktionalität innerhalb der Parallelen Koordinaten wurde bei keiner der untersuchten Anwendungen gefunden, zumal viele Anwendungen durch das Zeichnen eines einzigen Biclusters diese nicht benötigen.

4.3 Diskussion

Ziel dieser Diplomarbeit war es, eine Anwendung zu entwickeln, mit deren Hilfe es möglich ist, mehrere Bicluster gleichzeitig darzustellen und Abhängigkeiten zwischen den Biclustern zu verdeutlichen. Dafür wurden zwei miteinander verlinkte Darstellungen implementiert: die Heatmap und die Parallelen Koordinaten. Die Grundidee der Visualisierung ist es, dem Benutzer mit Hilfe der Heatmap einen Überblick über die Lage und Größe der Bicluster zu vermitteln. Weiterhin können mit Hilfe der Heatmap die einzelnen Zeilen und Spalten den Biclustern zugeordnet werden. Auch Überlappungen zwischen den Biclustern können innerhalb der Heatmap angezeigt werden. Aus der Gesamtheit der Bicluster kann sich der Benutzer somit für einen oder einige Bicluster entscheiden, die er genauer untersuchen möchte.

Durch die starke Verlinkung der beiden Ansichten passen sich diese immer synchron zueinander an. Somit sind Selektionen in der Heatmap stets in den Parallelen Koordinaten erkennbar. Zusätzlich ist es möglich die Anordnung der Zeilen und Spalten der Heatmap so anzuordnen, dass mehrere Bicluster zusammenhängend dargestellt werden können. Die Anordnung der Spalten der Heatmap wird ebenfalls von den Parallelen Koordinaten übernommen. Somit können sowohl Bicluster innerhalb der Heatmap als auch den Parallelen Koordinaten auf Gemeinsamkeiten untersucht werden.

Der Nutzen dieser Funktion soll an einem Beispiel demonstriert werden. Dieses Beispiel soll lediglich zeigen, wie Bicluster zueinander angeordnet werden können, um Gemeinsamkeiten und Unterschiede der Bicluster zu erkennen. Es soll keine biologischen Ergebnisse liefern. Im Beispiel wird der Yeast-Datensatz mit den 100 von CHENG und CHURCH gefundenen Biclustern [Har10] untersucht. Leider sind zu diesem Datensatz nur die Bezeichnungen der Gene (Zeilen) gefunden worden, die der Bedingungen (Spalten) bleiben unbekannt und werden für eine Zuordnung mit einer eindeutigen ID versehen: Spalte i erhält den Bezeichner *Cond_i*.

Zunächst wurde nach einem Bicluster mit signifikanter Ausprägung gesucht. Dabei wurde bei Bicluster 87[18] festgestellt, dass alle enthaltenen Gene bei Bedingung *Cond_10* einen wesentlich geringeren Expressionswert haben. Anschließend wurde nach weiteren Biclustern gesucht, die

18 Die Bicluster sind nach ihrer Anordnung in der Datei yeast.bicluster nummeriert.

einen ähnlichen Einbruch bei den Expressionswerten unter Bedingung *Cond_10* aufweisen. Der Selektionsmechanismus erwies sich dabei als ein sehr hilfreiches Werkzeug. Es wurden zusätzlich die Bicluster 91 und 6 gefunden. Bei diesen Biclustern stellte sich heraus, dass manche ihrer Gene einen deutlich geringeren Expressionswert aufweisen, andere hingegen einen deutlich höheren. Dies ist daran zu erkennen, dass die Bäuche der Bicluster 6 und 91 zwischen den Zentroiden bei *Cond_10* deutlich auseinander gehen.

Bicluster 6 erstreckt sich über alle Spalten der Datenmatrix, während die Bicluster 87 und 91 nur einige der Spalten beinhalten. Bicluster 87 und 91 können zunächst nicht gut miteinander verglichen werden, da sie nicht zusammenhängend innerhalb der Datenmatrix dargestellt sind. Durch ein Umsortieren der Zeilen und Spalten innerhalb der Heatmap können jedoch alle drei Bicluster zusammenhängend in der Datenmatrix angeordnet werden. Dies wird erreicht, indem die Sortierung der Bicluster vorgegeben wird. Es ergibt sich die in Abbildung 41 gezeigte Ansicht der Parallelen Koordinaten.

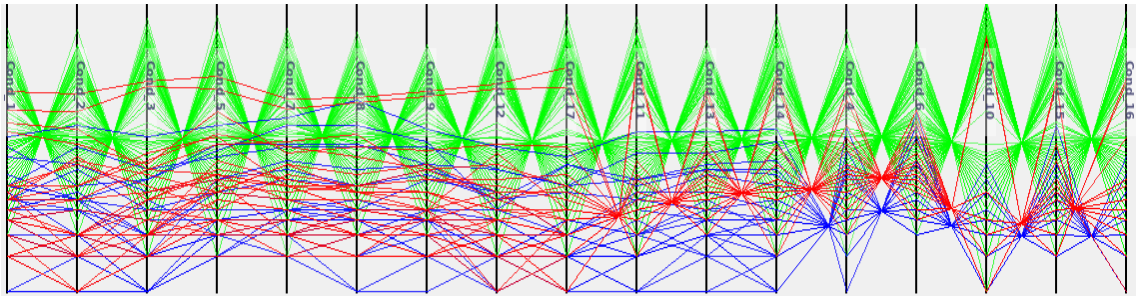


Abbildung 41: Zusammenhängende Bicluster durch Neusortierung der Zeilen und Spalten in Parallelen Koordinaten mit Zentroiden
Bicluster 6: grün, Bicluster 87: blau, Bicluster 91: rot

Aus der Ansicht der Heatmap ist ersichtlich, dass sich die Bicluster in keiner Zeile überlappen. Damit ist jedes der dargestellten Gene genau einem der drei Bicluster zugeordnet. Aus der Darstellung ist ersichtlich, dass sich die Bicluster 87 und 91 innerhalb der Spalten, die sie gemeinsam haben, im Mittel ähnlich verhalten, während Bicluster 6 über den gemeinsamen Verlauf im Mittel immer denselben Wert hat. In wieweit diese Gemeinsamkeiten von biologischer Bedeutung sind, bleibt zu klären.

Das Anwendungsbeispiel und die Pilotstudie haben gezeigt, dass sowohl die Heatmap als auch die Parallelen Koordinaten in der Lage sind, mehrere (auch überlappende) Bicluster interaktiv darzustellen. Durch den hohen Interaktionsgrad und die starke Verlinkung der Darstellungen ist

es dem Benutzer möglich, Bicluster in beiden Darstellungen so anzuordnen, dass sie zusammenhängend untersucht werden können.

Bicluster Viewer ist die bisher einzige Anwendung, die die Darstellungen von Heatmap und Parallelen Koordinaten in dieser Weise miteinander vereint. In den untersuchten Visualisierungen werden zwar ebenfalls Heatmaps und Parallele Koordinaten angeboten, jedoch sind die Interaktionsmöglichkeiten in den einzelnen Visualisierungen stark beschränkt oder nicht vorhanden. *Bicluster Viewer* bietet weiterhin die einzige Visualisierung in Parallelen Koordinaten an, die mehrere Bicluster in einer Darstellung anzeigen kann. Zusätzlich besteht die Möglichkeit die Bicluster in unterschiedlichen Farben hervorzuheben, wodurch eine Zuordnung der einzelnen Linien zu den Biclustern gewährleistet wird. Durch das Einzeichnen der Zentroide können sogar die einzelnen Achsen den Biclustern zugeordnet werden und der Mittelwert der entsprechenden Spalten kann abgelesen werden. Weiterhin stellt *Bicluster Viewer* Funktionalitäten bereit, mit deren Hilfe die Achsen innerhalb der Parallelen Koordinaten so umsortiert werden können, dass mindestens zwei Bicluster zusammenhängend dargestellt werden können. Somit wird ein Vergleich der Werte mehrerer Bicluster erheblich erleichtert.

Durch den entwickelte Mechanismus zum Einbinden neuer Bicluster-Algorithmen ist *Bicluster Viewer* nicht auf ein bestimmtes Berechnungsverfahren beschränkt. Es können leicht neue Bicluster-Algorithmen erstellt werden und diese zur Laufzeit der Anwendung hinzugefügt werden. Dadurch kann *Bicluster Viewer* schnell an unterschiedliche Aufgabenbereiche angepasst werden und ist somit nicht nur für die Analyse von Genexpressionsdaten ausgelegt.

Bicluster Viewer ist darauf ausgelegt Bicluster mit kohärenten Werten zu visualisieren. Bei Biclustern mit kohärenter Evolution in den Zeilen oder Spalten dürfen je nach definierter Ordnung Zeilen oder Spalten nicht vertauscht werden. Diese Tatsache wird allerdings bei der Anordnung von Zeilen und Spalten nicht beachtet und somit kann die definierte Ordnung unter Umständen zerstört werden. Weiterhin werden hierarchische Bicluster nicht gesondert bei der Anordnung der Zeilen und Spalten betrachtet. Diese werden auf die gleiche Art und Weise in die Sortierung eingefügt wie alle anderen Bicluster. Dadurch werden jedoch die Freiheitsgrade bei der Anordnung der Zeilen und Spalten eingeschränkt. Bei Experimenten, die die Daten zu unterschiedlichen Zeitpunkten erfassen, ist eine komplette Neuanordnung der Spalten unter Umständen sehr irreführend. Solche Einschränkungen in der Anordnung der Zeilen und Spalten werden

in der aktuellen Version von *Bicluster Viewer* nicht berücksichtigt und sollten Bestandteil einer möglichen Weiterentwicklung werden.

Ein weitere negative Auswirkung bei der Anordnung der Zeilen und Spalten ist die Tatsache, dass bei der Anordnung immer alle Bicluster betrachtet werden und der Benutzer nur die Reihenfolge der Bicluster vorgeben kann. Deshalb sollte es eine Möglichkeit geben, bestimmte Bicluster aus der berechneten Bicluster-Menge auszublenden oder zu löschen. Bisher kann dies nur dadurch erreicht werden, indem bestimmte Bicluster aus einer Datei ausgelesen werden.

Bei Biclustern mit sehr ähnlichen Werten kann es in der Darstellung der Parallelen Koordinaten weiterhin vorkommen, dass sich die Linien der einzelnen Bicluster stark überdecken und somit der Verlauf der Linien nicht mehr erkennbar ist. Das Überdecken der einzelnen Linien wird bei ähnlichen Mittelwerten durch die Zentroide noch verstärkt. Weiterhin ist keine Interaktion innerhalb der Parallelen Koordinaten realisiert. Die Auswahl der Bicluster muss über die Heatmap oder die Navigationsliste erfolgen. Dadurch ist es dem Benutzer nicht möglich, Bicluster anhand bestimmter Linienverläufe oder Vorgabe bestimmter Datenwerte zu selektieren. Einen guten Ansatz für eine solche Auswahlmöglichkeit stellt die Interaktion von *BicOverlapper* innerhalb der Parallelen Koordinaten dar. Hier kann der Benutzer Bicluster selektieren, indem er Schwellwerte auf den einzelnen Achsen definiert. Inwieweit dieser Ansatz für die Darstellung der Parallelen Koordinaten mit Zentroiden umsetzbar ist, bleibt zu untersuchen.

Beim Benutzen *Bicluster Viewer* (mit Expressionsdaten) fiel auf, dass manche Bicluster für eine Analyse nicht von Bedeutung sind, da sie Datenwerte gruppieren, in denen keine nennenswerten Veränderungen auftreten. Deshalb ist ein Interaktionsmechnismus wünschenswert, mit dessen Hilfe der Bicluster-Algorithmus bestimmte Bicluster nicht beachtet oder nur Bicluster auf bestimmten Datenwerten findet. Weiterhin sollten bei der Berechnung von Biclustern bestimmte Zeilen oder Spalten von der Berechnung ausgeschlossen werden können. Die Interaktion sollte nicht auf der Übergabe von Parametern basieren, sondern vielmehr in den einzelnen Darstellungen vorgenommen werden können. Jedoch muss ein für dieses Vorhaben verwendeter Bicluster-Algorithmus in der Lage sein, die definierten Kriterien zu interpretieren. Inwieweit der Mechanismus zum Einbinden neuer Bicluster-Algorithmen verändert werden müsste, ist schwer abzuschätzen.

5 Literaturverzeichnis

[**Alb04**]: B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P.Walter. **Molekularbiologie der Zelle**. *4. Auflage* . Seiten: 223-228 und 345-420 WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim(2004)

[**Art04**]: A. O. Artero, M. C. Ferreira de Oliveira, H. Levkowitz. **Uncovering Clusters in Crowded Parallel Coordinates Visualizations**. *IEEE Symposium on Information Visualization* . Seiten: 81-88 (2004)

[Bar04]: N. Barlow, L. J. Stuart. **Animator: A Tool for the Animation of Parallel Coordinates**. *Proceedings of the Eighth International Conference on Information Visualisation*, IEEE (2004)

[BiV10]: BiVisu, http://www.eie.polyu.edu.hk/~nflaw/Biclustering/ (Stand: 1.3.2010)

[**Bro99**]: P. O. Brown, D. Botstein. **Exploring the new world of the genome with DNA microarrays**. *nature genetics suppleent* **21**. Seiten: 33-37 (1999)

[**Che00**]: Y. Cheng, G. M. Church. **Biclustering of expression data**. *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB '00)* . Seiten: 93-103 (2000)

[Che07a]: K.O. Cheng, N.F. Law, W.C. Siu, A. W.-C. Liew . **Biclusters visualization and detection using parallel coordinate plots**. *Proceedings of the International Symposium on Computational Models for Life Sciences*, American Institute of Physics (2007)

[**Che07b**]: K.O. Cheng, N.F. Law, W.C. Siu, T.H. Lau. **BiVisu: Software Tool for Bicluster Detection and Visualization**. *BMC Bioinformatics* **23**.17 Seiten: 2342-2344 (2007)

[**Dep06**]: P. Depping. **Biologische Grundlagen**. Universität Ulm http://theorie.informatik.uni-ulm.de/Lehre/SS6/BioGrundLagen_Depping2.pdf (2006)

[**Det09**]: M. Dettling. **Microarrays**. Eth Zürich http://stat.ethz.ch/~dettling/Microarray.pdf

(2009)

[**Die09**]: J. Dietzsch, J. Heinrich, K. Nieselt, D. Bartz. **SpRay: A Visual Analytics Approach for Gene Expression Data**. *IEEE Symposium on Visual Analytics Science and Technology* . Seiten: 179-186 (2009)

[ETH10]: ETH Zürich, http://www.tik.ethz.ch/sop/bicat/?page=bicat.php (Stand: 1.3.2010)

[**Ges09**]: P. Gestraud, I. Brito, E. Barillot. **BicARE : Biclustering Analysis and ResultsExploration** v0.1.0 (2009) http://www.bioconductor.org/packages/2.4/bioc/manuals/BicARE/man/BicARE.pdf (Stand: Januar 2010)

[**Gro06**]: G. A. Grothaus, A. Mufti, T.M. Murali. **Automatic Layout and Visualization of Biclusters**. *Algorithms for Molecular Bioligy* **1**.15 (2006)

[Har10]: Harvard Molecular Technology Group, http://arep.med.harvard.edu/biclustering/ (Stand: 10.2.2010)

[**Ins85**]: A. Inselberg. **The plane with parallel coordinates**. *The Visual Computer* **1**.2 Seiten: 69-91 Springer Berlin / Heidelberg(1985)

[**Kai09**]: S. Kaiser, R. Santamaría, R. Therón, L. Quintales, F. Leisch. **BiCluster Algorithms** v0.8.1 (2009) http://cran.r-project.org/web/packages/biclust/biclust.pdf (Stand: Dezember 2009)

[**Koh06**]: Hyung-Won Koh (B. Reusch) . **Evolutionäre Algorithmen für das Biclustering-Problem**. Universität Dortmund, Fachbereich Informatik (2006)

[**Kri09**]: H.-P. Kriegel, P. Kröger, A. Zimek. **Clustering on High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering**. *ACM Transaction on Knowledge Discovery from Data* **3**.1 (2009)

[**Leu03**]: Y. F. Leung, D. Cavalieri. **Fundamentals of cDNA microarraydata analysis**. *TRENDS in Genetics* **19**.11 Seiten: 649-659 (2003)

[Luo10]: Y. Luo, J. Heinrich, D. Weiskopf, H. Zhang, A. E.Kirkpatrick. **Cluster Visualization in Parallel Coordinates Using Curve Bundles**. *Unpublished manuscript* (2010)

[**Mad04**]: S.C. Madeira, A.L. Oliveira. **Biclustering Algorithms for Biological DataAnalysis: A Survey**. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **1**.1 Seiten: 24 – 45 (2004)

[**Pee00**]: R. Peeters. **The maximum edge biclique problem is NP-complete.** *Tilburg University* (2000)

[**R_D09a**]: R Development Core Team. **R Installation and Afdministration** v2.10.1 (2009) http://cran.r-project.org/doc/manuals/R-admin.pdf (Stand: Januar 2010)

[**R_D09b**]: R Development Core Team. **R Data Import/Export** v2.10.1 (2009) http://cran.r-project.org/doc/manuals/R-data.pdf (Stand:)

[**R_D09c**]: R Development Core Team. **Writing R Extensions** v2.10.1 (2009) http://cran.r-project.org/doc/manuals/R-exts.pdf (Stand: Januar 2010)

[**R_D09d**]: R Development Core Team. **R Internals** v2.9.1. (2009) http://cran.r-project.org/doc/manuals/R-ints.pdf (Stand: Januar 2010)

[**R_D09e**]: R Development Core Team. **An Introduction to R** v2.10.1 (2009) http://cran.r-project.org/doc/manuals/R-intro.pdf (Stand: Januar 2010)

[**Rüb06**]: O. Rübel, G.H. Weber, S.V.E. Keränen, C.C. Fowlkes, C.L. Luengo Hendriks, Lisa Simirenko, N.Y. Shah, M.B. Eisen, M.D. Biggin, H. Hagen, D. Sudar, J. Malik, D.W. Knowles, B. Hamann. **PointCloudXplore: Visual Analysis of 3D Gene Expression Data Using Physical Views and Parallel Coordinates.** *Eurographics/ IEEE-VGTC Symposium on Visualization* (2006)

[**San08**]: R. Santamaría, R. Therón, L. Quintales. **A visual analytics Approach for understanding Biclustering Results from Microarray Data.** *BMC Bioinformatics* **9**.247 (2008)

[**Wil08**]: L. Wilkinson, M. Friendly. **The History of the Cluster Heat Map.** *to appear in The American Statistician* . (2008)

[**Yan05**]: J. Yang, H. Wang, W. Wang, P. Yu. **An improved biclustering method for analyzing gene expression.** *International Journal on Artificial Tools* **14**.5 Seiten: 771-789 (2005)

[**Zho09**]: H. Zhou, W. Cui, H Qu, Y. Wu, X. Yuan, W. Zhuo. **Splatting the Lines in Parallel Coordinates.** *Eurographics/ IEEE-VGTC Symposium on Visualization* **28**.3 (2009)

A Die Aminosäuren

Die 20 Aminosäuren heißen:

ALA:	Alanin
ARG:	Arginin
ASN:	Asparagin
ASP:	Asparaginsäure
CYS:	Cystein
GLN:	Glutamin
GLU:	Glutaminsäure
GLY:	Glycin
HIS:	Histidin
ILE:	Isoleucin
LEU:	Leucin
LYS:	Lysin
MET:	Methionin
PHE:	Phenylalanin
PRO:	Prolin
SER:	Serin
THR:	Threonin
TRP:	Tryptophan
TYR:	Tyrosin
VAL:	Valin

B Das Erstellen von Bicluster-Algorithmen

In diesem Abschnitt möchte ich erläutern, welche Schritte durchgeführt werden müssen, um eine neue Implementierung eines Bicluster-Algorithmus zu erstellen. Um eine so-Datei mit einer neuen Implementierung zu erstellen, muss bei der Kompilierung gegen die Bibliothek *libAbstractBiclusterAlgorithm.so* gelinkt werden (dieselbe Datei muss sich im Library-Path des Systems befinden, um die gesamte Anwendung auszuführen). Die so-Datei kann aus den Quell- und Header-Dateien im Ordner *BiclusterAlgorithm* (beiliegende CD) erstellt werden. Soll die neue Implementierung eine Erweiterung der AbstractRBiclusterAlgorithm-Klasse sein, muss zusätzlich gegen die libR.so (von *R* mitgelieferte Library) gelinkt werden.

Der Bicluster-Algorithmus wird mit Hilfe der Funktion *dlopen* aus der Bibliothek *dlfcn* geladen. Da die Methode dafür ausgelegt ist C-Funktionen über Symbole zu laden, muss es innerhalb der Implementierung zwei C-Funktionen geben[19]:

- **AbstractBiclusterAlgorithm* create(void)**: Die Methode „create“ wird aufgerufen um ein neues Bicluster-Algorithmus Objekt (*AbstractBiclusterAlgorithm*) zu erzeugen. Innerhalb der Methode „create“ sollte also der Konstruktor der neu implementierten Klasse aufgerufen werden.
- **void destroy(AbstractBiclusterAlgorithm *p)**: Die Methode „destroy“ wird aufgerufen, wenn ein bereits erstelltes Bicluster-Algorithmus Objekt (hier p) wieder gelöscht werden soll. Innerhalb der Funktion sollte daher nur die Anweisung „delete p“ vorkommen.

Auf den nachfolgenden Seiten wird eine beispielhafte Implementierung eines Bicluster-Algorithmus beschrieben und erläutert. Der Algorithmus erwartet den Parameter „myBool“ vom Type *boolean. Er* gibt lediglich einen Bicluster zurück, der entweder alle Zeilen und alle Spalten oder keine Zeile und kein Spalte der Datenmatrix enthält. Dies richtet sich nach dem Wert des

19 Die beiden Funktionen sind nicht Teil der zu implementierenden Klasse, sondern sind außerhalb der Klassen-Definition implementiert. Sie befinden sich im unteren Teil der Beispiel-Implementierung.

Parameters „myBool“. Im Folgenden werden alle zu implementierenden Funktionen aufgeführt und kurz erklärt:

- **getName**: Diese Methode gibt lediglich eine Zeichenkette mit dem Namen des implementierten Algorithmus zurück.
- **quality**: Diese Methode gibt einen double-Wert zurück. Die Qualität ist höher, je kleiner der Rückgabewert ist. Die beste Qualität wird mit 0.0 erreicht. Die Methode sollte also keine negativen Zahlen zurückgeben.
- **getParameterDef**: Diese Methode gibt eine Liste von Parameter-Definitionen zurück. Hier wird ein Parameter namens „myBool“ definiert. Er ist vom Typ *boolean* und ist nicht optional. Die Parameter-Liste kann mit Hilfe der Methode *getParameter* ausgelesen werden. Ist ein Parameter nicht in der Liste enthalten, so gibt *getParameter* einen leeren String zurück.
- **checkParameter**: Diese Methode dient der Überprüfung der übergebenen Parameter-Werte. Sie wird vor der *compute*-Methode von der Klasse *Executer* (vgl. Abschnitt 3.3 auf Seite 54) aufgerufen. Sind Parameter in der Parameter-Liste falsch oder nicht gesetzt soll eine *BiclustException* geworfen werden.
- **compute**: Diese Methode wird ebenfalls von der Klasse *Executer* aufgerufen. Sie soll die Bicluster aus der Datenmatrix und den übergebenen Parametern berechnen. Ein *Biclust-Model*-Objekt kann durch zwei Bool'sche Listen erzeugt werden. Dabei ist in jeder der Listen an der i-ten Stelle ein *true* eingetragen, falls die i-te Zeile bzw. Spalte dem Bicluster angehört. Ansonsten befindet sich an der i-ten Stelle ein *false*. Die Länge der Liste muss der Zeilen- bzw. Spaltenanzahl der Datenmatrix entsprechen.

Für weitere Informationen zu den einzelnen Typen und Methoden möchte ich auf die Dokumentation und die Standard-Implementierung (Ordner *BiclusterAlgorithmImpl*) auf beiliegender CD verweisen.

Auf der nächsten Seite wird eine beispielhafte Implementierung der Klasse *DummyBicluster* präsentiert.

```
class DummyBicluster: public AbstractBiclusterAlgorithm {
public:
	bool _myParameterValue;
	DummyBicluster() : AbstractBiclusterAlgorithm() {}
	vector<BiclustModel*> compute(Matrix* m, ParameterList* p) throw (BiclustException*)
	{	vector<BiclustModel*> biclusters;
		int rowCount = m->getRowCount();
		int colCount = m->getColCount();
		vector<bool> rows;
		vector<bool> cols;
		for (int i = 0; i < rowCount; i++){
			rows.push_back(_myParameterValue);
		}
		for (int i = 0; i < colCount; i++){
			cols.push_back(_myParameterValue);
		}
		biclusters.push_back(new BiclustModel(rows, cols));
		return biclusters;
	}
	string getName()
	{	return "This is a dummy implementation";
	}
	vector<ParameterDef> getParameterDef()
	{	vector<ParameterDef> defs;
		ParameterDef myDef;
		myDef.name = "myBool";
		myDef.type = Type_boolean;
		myDef.optional = false;
		defs.push_back(myDef);
		return defs;
	}
	double quality(Matrix *matrix, BiclustModel *bicluster)
	{	return 0.0;
	}
	void checkParameter(ParameterList *param) throw (BiclustException*)
	{
		if (param->getParameter("myBool") == "")
			throw new BiclustException("myBool must be set");
		else if (param->getParameter("myBool") == Parameter::WAHR)
			_myParameterValue = true;
		else
			_myParameterValue = false;
	}
}

extern "C" AbstractBiclusterAlgorithm* create() {return new DummyBicluster();}

extern "C" void destroy(AbstractBiclusterAlgorithm* p) {delete p;}
```

Beispielhafte Implementierung eines Bicluster-Algorithmus

C Pilotstudie

Die Studie wurde am 16. März 2010 in Stuttgart durchgeführt. Insgesamt nahmen vier Teilnehmer an der Studie teil. Die Studie wurde an einem Notebook mit einem Intel Core2 Duo mit 2 GHz und 2GB Arbeitsspeicher durchgeführt. Den Teilnehmern wurden jeweils einzeln die Funktionsweise der Anwendung vorgestellt und sie erhielten eine kurze Erklärung über (überlappende) Bicluster. Die Funktionsweise der Anwendung und der Ablauf der Studie werden ebenfalls in den folgenden Informationsblättern beschrieben. Der Inhalt der Informationsblätter und der durchgeführten Einführung ist gleich. Den Teilnehmern wurde nach der Einführung ein wenig Zeit eingeräumt die Anwendung zu bedienen und falls nötig Fragen zu stellen.

Die Teilnehmer wurden angewiesen ihre Gedanken und Vorgehensweisen während der Bearbeitung der Aufgaben laut zu äußern. Die Aufgaben wurden den Teilnehmern nur mündlich gestellt, ihre Antworten, die geäußerten Gedanken und ihre Vorgehensweisen wurden notiert. Ebenfalls wurde pro Frage die Zeit gestoppt, die ein Teilnehmer für die Beantwortung der Frage benötigte. Die Aufgaben sind auf den folgenden Seiten aufgeführt, dabei ist eine bzw. mehrere richtige Lösung pro Aufgabe eingetragen. Die Fragen sind mit denen, die in der Studie nur mündlich gestellt wurden, identisch.

Zunächst wurden dem Benutzer nacheinander Ausdrucke der auf den Aufgabenblättern 1 bis 3 dargestellten Abbildungen vorgelegt. Zu den jeweiligen Ausdrucken wurden dem Teilnehmer dann die Fragen gemäß des entsprechenden Aufgabenblatts gestellt. Danach wurden die Aufgaben 4 bis 6 vom Teilnehmer unter Zuhilfenahme der Anwendung gelöst. Auch hier wurden die Fragen nur mündlich gestellt und die Antworten wurden für den Teilnehmer notiert.

Die jeweiligen Datensätze, die in die Anwendung geladen wurden, sind auf den jeweiligen Aufgabenblättern notiert. Die Datensätze selbst befinden sich auf der beiliegenden CD im Ordner „Pilotstudie/data“.

Hinweise zur Pilotstudie

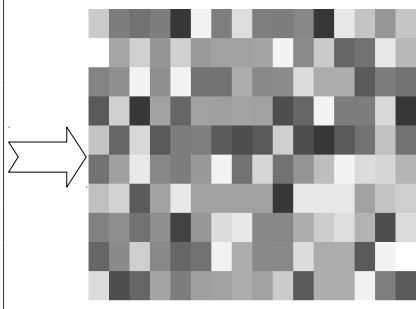
Ziel dieser Pilotstudie ist es, ein für die Bicluster-Visualisierung entwickeltes Software-Werkzeug zu testen. Dabei werden Ihnen unterschiedliche Datensätze vorgelegt. Ihre Aufgabe wird es sein, Bicluster, die auf diesen Datensätzen definiert wurden, zu untersuchen. Dabei soll vor allem getestet werden:

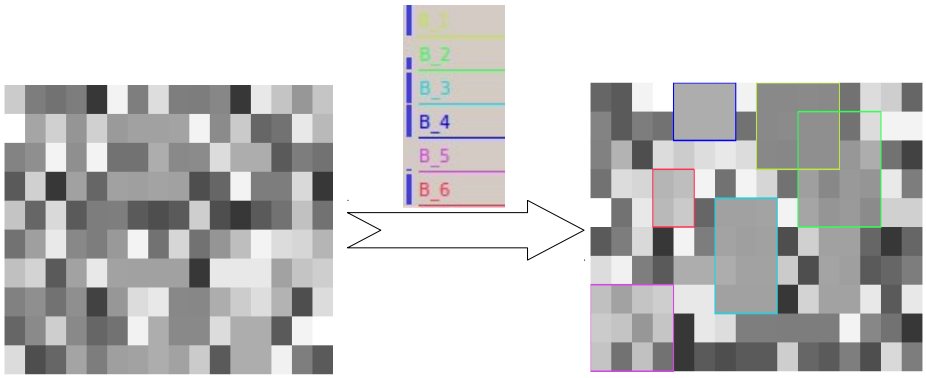
- wie gut lassen sich Bicluster unterscheiden
- wie präzise lassen sich Überlappungen bei Biclustern erkennen
- wie genau können Bicluster den einzelnen Zeilen und Spalten zugeordnet werden
- wie genau können einzelne Bicluster untersucht werden

Hinweise zur Funktionsweise der Anwendung

Die Bicluster-Visualisierung wird auf einer Heatmap realisiert, bei der Datenwerte auf Graustufen abgebildet werden. Dabei wird der kleinste Wert auf Schwarz, der größte auf Weiß abgebildet. Um die Bicluster zusammenhängend darzustellen wird die Heatmap durch Zeilen- und Spaltenvertauschungen umsortiert. Die Bicluster werden anschließend durch umschließende Rechtecke gekennzeichnet. Jedem Bicluster wird eine unterschiedliche Farbe zugeordnet. Die nächste Abbildung zeigt Ihnen eine Darstellung einer solchen Heatmap mit Biclustern.

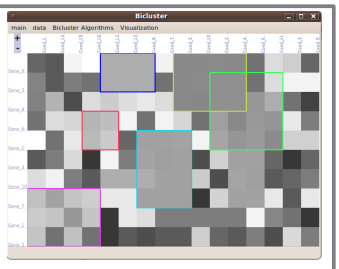
6.6	0	-1	0	-6	10	0.1	8.1	0.1	0	0.8	-6	9	6	2.2	6.1
11	3.4	7	2	7	2.5	3	3	3.2	10	1.2	6.5	-2	-1	9	5.1
0.5	1.5	10	10	-1	-1	4	1	1.2	8	4	4	4	-3	0	-1
-3	7	-6	3	-2	3	2.9	3.2	3	-4	-2	10	0	0	7.8	-6
6	-2	7.8	-3	0	0.1	-3	-4	-3	6.9	-4	-6	-3	-1	5.8	-1
-1	2.9	9.2	0.9	0	2.4	10	-1	7.5	-1	2	5.5	10	8	7.6	4.8
5.7	7.3	-3	3.2	9	3.1	3	3	3	-6	9	9	9	3.1	6	6.9
0.3	1.5	-1	1.5	-5	2.2	8	9	0.9	0.8	4	6.7	8.1	4.6	-1	8
-2	1.1	6.9	1	-2	-1	9.9	4	1	1	8	4	4	-1	10	11
8	-4	-2	3.3	0	2.9	3.1	4	3.1	6.6	-3	4	4	9.8	0.1	-3





Informationsblatt 1 aus der Pilotstudie

Bedienung der Anwendung



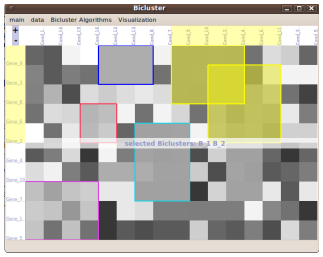
Rechts ist ein Screenshot der Oberfläche dargestellt. Die Menü-Einträge, die Sie in dieser Studie benutzen sollen finden Sie unter *Visualization*. Dabei sind nur die ersten beiden Einträge *setup fit policy* und *show all biclusters* für diese Studie von Bedeutung. Alle anderen Funktionen haben keine Auswirkungen auf den von Ihnen zu testenden Teil der Anwendung.

Navigation innerhalb der Heatmap

In der Darstellung werden Ihnen im oberen Teil die Spaltennamen angezeigt, die Zeilennamen befinden sich links neben der Heatmap. Im linken oberen Teil befinden sich ein + und ein - Symbol. Diese dienen zur Einstellung des Zoom-Faktors. Der Zoom-Faktor kann ebenfalls durch Drücken der STRG-Taste und Bewegen des Mausrads verstellt werden. Durch einen Rechtsklick auf das + oder - Symbol wird der Zoom-Faktor wieder auf 1 gestellt.

Kann die Heatmap nicht komplett dargestellt werden, werden automatisch Scroll-Balken eingeblendet mit denen Sie die Ansicht verschieben können. Sie können ebenfalls (bei nicht gedrückter STRG-Taste) mit dem Mausrad in vertikaler Richtung scrollen, bei gedrückter SHIFT-Taste können Sie mit dem Mausrad in horizontaler Richtung scrollen.

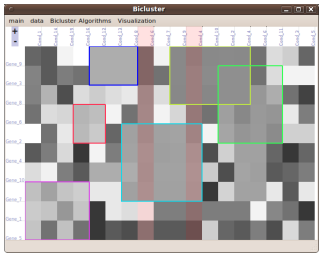
Selektion von Biclustern



Durch einen Linksklick innerhalb eines Rechtecks wird der bzw. werden die entsprechende/n Bicluster selektiert. Selektierte Bicluster werden immer in Gelb dargestellt. Dabei werden alle Zellen, die dem Bicluster angehören mit einem transparenten Gelb überzeichnet. Überlappen sich in einer Zelle mehrere Bicluster so wird diese Zelle in einem stärkeren Gelb dargestellt (additive Farbmischung). Die entsprechenden Zeilen bzw. Spalten werden ebenfalls in Gelb hervorgehoben. Zusätzlich wird kurz eine Information eingeblendet welche Bicluster gerade selektiert wurden. Bei gedrückter STRG-Taste können mehrere Bicluster ausgewählt werden. Wird bei gedrückter STRG-Taste ein selektierter Bicluster ausgewählt, so wird die Selektion wieder aufgehoben.

Die Selektion kann ebenfalls über das Anklicken von Zeilen- bzw. Spaltennamen erfolgen. Wird auf eine Zeile bzw. Spalte im linken bzw. oberen Teil der Oberfläche geklickt, so werden alle Bicluster selektiert, die sich in der entsprechenden Zeile bzw. Spalte befinden.

Verdoppeln von Zeilen und Spalten

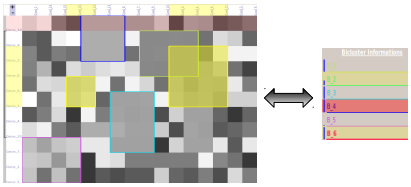


Da es in manchen Fällen unmöglich ist alle Bicluster zusammenhängend darzustellen, gibt es die Möglichkeit entsprechende Zeilen und Spalten einzelner Bicluster zu verdoppeln. Um einen bestimmten Bicluster zusammenhängend darzustellen wird mittels eines Rechtsklicks innerhalb des entsprechenden Rechtecks ein Kontextmenü geöffnet. Durch Auswahl des Punktes *expand duplicated* werden die benötigten Zeilen und Spalten verdoppelt dargestellt. Die verdoppelten Zeilen und Spalten werden in einem transparenten Rot hervorgehoben.

Mit Hilfe der Leertaste kann über den Spalten- bzw. links neben den Zeilennamen eine Markierung für die verdoppelten Spalten bzw. Zeilen ein- und ausgeblendet werden. Dabei werden die verdoppelten Spalten bzw. Zeilen mit den entsprechenden originalen Spalten bzw. Zeilen verbunden. Die Bicluster-Selektion über die Spalten bzw. Zeilen wird dann deaktiviert.

Informationsblatt 2 aus der Pilotstudie

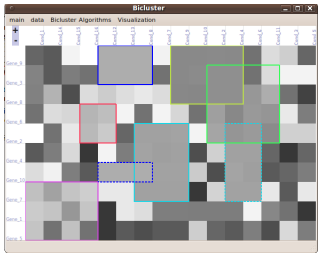
Die Bicluster-Navigationsliste



Die Navigationsliste im rechten Teil der Anwendung dient als Übersicht über alle in der Heatmap dargestellten Bicluster. Sie zeigt weiterhin alle selektierten Bicluster an, indem sie den Hintergrund in Gelb einfärbt. Zusätzlich bietet sie eine Übersicht, welcher Bicluster gerade verdoppelt dargestellt ist. Dieser ist dann mit Rot hinterlegt.

Es ist ebenfalls möglich Bicluster innerhalb der Liste zu selektieren. Durch einen Linksklick wird ein Bicluster selektiert, bei gedrückter STRG-Taste können Sie auch hier wieder mehrere Bicluster auswählen. Mit der rechten Maustaste können Sie das Kontextmenü öffnen, mit dem Sie Bicluster durch Verdopplung zusammenhängend darstellen können. Die Selektion innerhalb der Heatmap und der Navigationsliste sind synchron.

Die Menü-Einträge

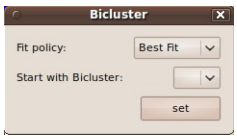


show all biclusters

Da unter Umständen nicht alle Bicluster gleichzeitig zusammenhängend dargestellt werden können, können mit Hilfe des Menü-Eintrags *show all biclusters* gestrichelte Rechtecke eingeblendet werden. Dabei wird um alle Zellen, die einem Bicluster angehören, ein gestricheltes Rechteck (in der entsprechenden Farbe) gezeichnet. Durch erneutes Auswählen des Menü-Eintrags werden diese Rechtecke wieder ausgeblendet.

setup fit policy

Das Layout der Heatmap wird automatisch bestimmt. Dabei kann es vorkommen, dass bestimmte Bicluster nicht ohne Verdopplung zusammenhängend dargestellt werden können. Durch Auswahl des Menü-Eintrags *setup fit policy* öffnet sich ein Dialog mit dem auf das Layout (die Sortierung der Zeilen und Spalten) Einfluss genommen werden kann. Die Sortierung der Heatmap wird aufgebaut, indem nach und nach alle Bicluster in die Heatmap eingefügt werden.



Der Dialog bietet zum einen die Möglichkeit auf das Einfügekriterium Einfluss zu nehmen und zum anderen kann bestimmt werden, welcher Bicluster zuerst in die Heatmap eingefügt werden soll. Für das Einfügekriterum gibt es drei Zustände: *Best Fit*, *Column Fit* und *Row Fit*. *Best Fit* bedeutet, dass beim Einfügen eines neuen Biclusters in die Heatmap sowohl auf Zeilen- als auch auf Spaltenüberlappungen geachtet wird. Bei *Column* bzw. *Row Fit* wird lediglich auf Spalten- bzw. Zeilenüberlappungen geachtet.

Da das Layout durch Einfügen der einzelnen Bicluster in die Heatmap berechnet wird, ist gewährleistet, dass die ersten beiden Bicluster, die eingefügt werden, zusammenhängend dargestellt werden können. Wird bei der Umsortierung zum Beispiel mit Bicluster B_1 begonnen, so wird dieser auf den originalen Zeilen und Spalten dargestellt. Als zweiter Bicluster wird dann der genommen, der gemäß des Einfügekriteriums die meisten Zeilen und/oder Spalten mit B_1 gemeinsam hat und entsprechend in die Heatmap eingefügt.

Wichtiger Hinweis für das Standard-Layout:

Beim Standard-Layout (beim Eintrag *Start with Bicluster* ist kein Bicluster ausgewählt) wird der Bicluster mit den meisten Überlappungen zuerst eingefügt. Danach wird derjenige Bicluster in die Heatmap eingefügt, der mit den bereits eingefügten Biclustern die meisten Überlappungen (gemäß des Einfügekriteriums) aufweist. Das Einfügekriterium ist beim Standard-Layout auf *Best Fit* gesetzt.

Informationsblatt 3 aus der Pilotstudie

Ablauf der Studie

Zunächst dürfen Sie sich mit der Funktionsweise der Anwendung vertraut machen. Bitte scheuen Sie sich nicht bei Unklarheiten Fragen zu stellen. Die Informationsblätter dürfen Sie während der Einarbeitung und den später zu bearbeitenden Aufgaben benutzen. Auch während der Bearbeitung der Aufgabe dürfen Sie gerne Fragen zur Funktionsweise der Anwendung stellen.

Sie erhalten anschließend 6 Aufgaben. Aufgabe 1, 2 und 3 sind ohne Zuhilfenahme der Anwendung zu bearbeiten, die restlichen Aufgaben (4 bis 6) beziehen sich auf das Arbeiten mit der Anwendung. Bitte versuchen sie die Aufgaben möglichst präzise zu beantworten und versuchen Sie nicht, möglichst schnell die Aufgaben abzuschließen. Ihnen werden bei jeder Aufgabe Fragen gestellt, die Sie nur mündlich beantworten sollen.

Bitte fühlen Sie sich ermutigt während der Bearbeitung der Aufgabe Kommentare zu Ihrer Vorgehensweise und den Funktionen der Anwendung einzubringen. Dabei spielt es keine Rolle, ob diese Kommentare positiv oder negativ ausfallen. Ein Feedback zur Benutzerfreundlichkeit der Anwendung ist zu jedem Zeitpunkt angebracht.

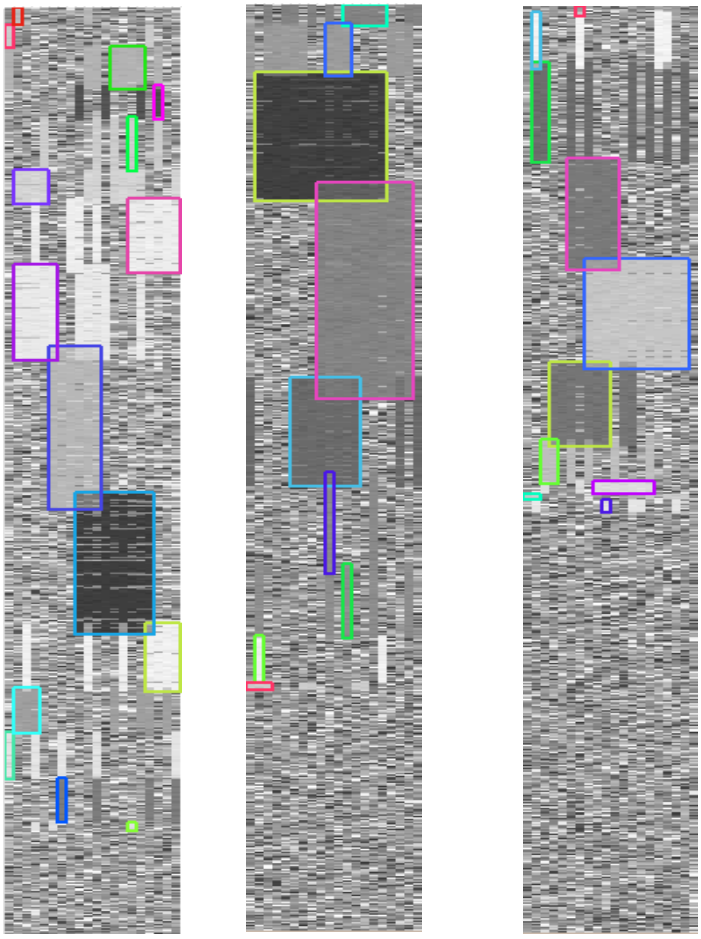
Die Zeit, die Sie für das Bearbeiten einer Aufgabe benötigen wird gestoppt. Bitte lassen Sie sich durch diese Tatsache nicht verunsichern und bearbeiten Sie ihre Aufgabe in Ruhe weiter. Die Zeitnahme dient lediglich einer späteren Auswertung.

Ich möchten mich bei Ihnen für die Teilnahme an dieser Studie bedanken und hoffe Sie finden an den nun folgenden Aufgaben ein wenig Vergnügen.

Informationsblatt 4 aus der Pilotstudie

Aufgabe 1: a) Rand2 b) Rand c) Rand1

Im Folgenden sind Bicluster-Ergebnisse auf drei verschiedenen Datensätzen dargestellt.



Frage 1

Wie viele Bicluster gibt es in den jeweiligen Darstellungen?

a) 15 b) 9 c) 10

Frage 2

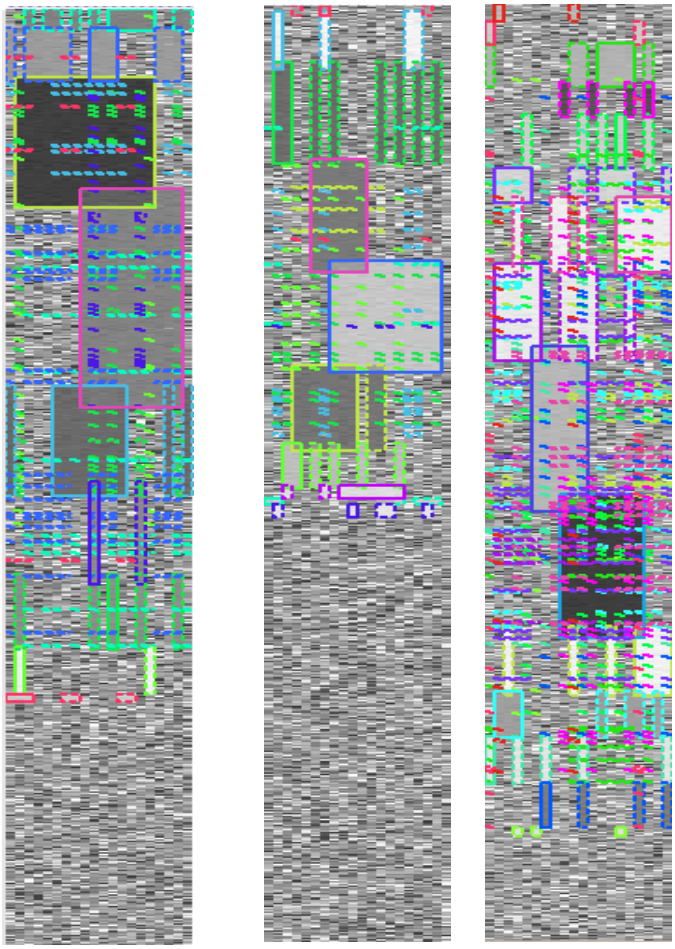
Bei welcher der Darstellungen ist die Überlappung am stärksten, das heißt, in welcher Darstellung gibt es die meisten Zellen, die zu mehreren Biclustern gehören?

Darstellung mit den meisten überlappenden Biclustern: a)

Bearbeitungsblatt 1 aus der Pilotstudie

Aufgabe 2: a) Rand b) Rand1 c) Rand2

Auch hier sind Bicluster-Ergebnisse auf drei verschiedenen Datensätzen dargestellt. Diesmal ist die Option *show all biclusters* gesetzt.



Frage 1

Wie viele Bicluster gibt es in den jeweiligen Darstellungen?

a) 9 b) 10 c) 15

Frage 2

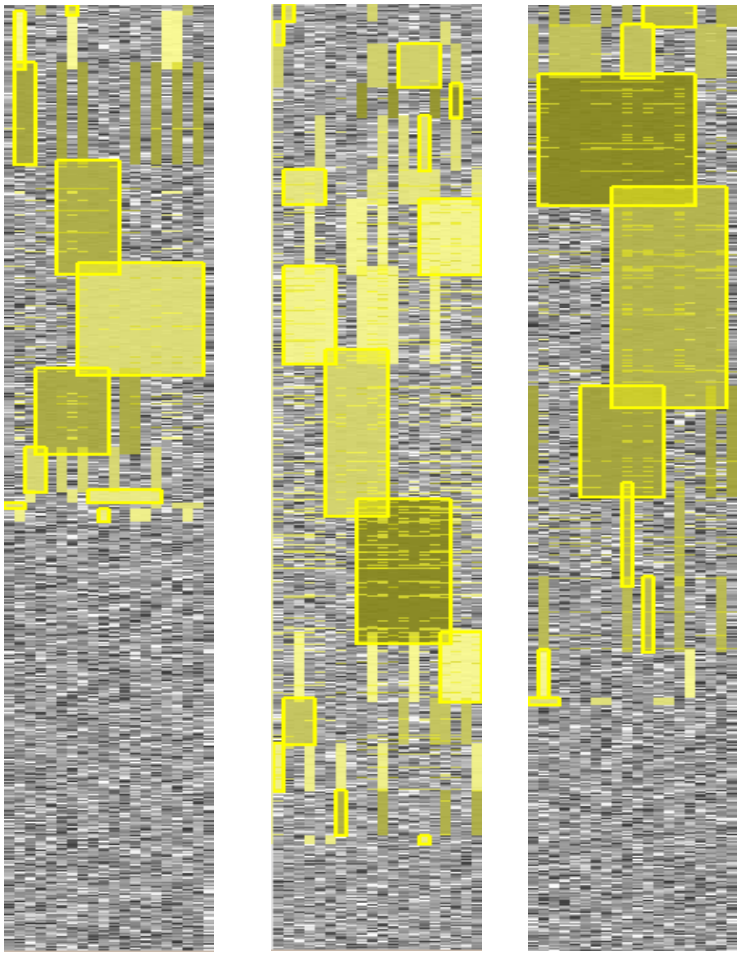
Bei welcher der Darstellungen ist die Überlappung am stärksten, das heißt, in welcher Darstellung gibt es die meisten Zellen, die zu mehreren Biclustern gehören?

Darstellung mit den meisten überlappenden Biclustern: c)

Bearbeitungsblatt 2 aus der Pilotstudie

Aufgabe 3: a) Rand1 b) Rand2 c) Rand

Auch hier sind Bicluster-Ergebnisse auf drei verschiedenen Datensätzen dargestellt. Diesmal ist die Option *show all biclusters* nicht gesetzt, aber alle Bicluster wurden selektiert.



Frage 1

Wie viele Bicluster gibt es in den jeweiligen Darstellungen?

a) 10 b) 15 c) 9

Frage 2

Bei welcher der Darstellungen ist die Überlappung am stärksten, das heißt, in welcher Darstellung gibt es die meisten Zellen, die zu mehreren Biclustern gehören?

Darstellung mit den meisten überlappenden Biclustern: b)

Bearbeitungsblatt 3 aus der Pilotstudie

Aufgabe 4 (Datensatz: yeast.dat, Bicluster: yeast.bic)

In der rechten Abbildung finden Sie eine Auflistung aller angezeigten Bicluster mit den entsprechenden Einfärbungen. Bitte geben Sie zur Beantwortung der Fragen, die Bicluster ID (z. B. B_11) und die Zeilen- bzw. Spaltennamen an.

Frage 1

Welcher ist der größte Bicluster?

B_1

Frage 2

Gibt es überlappende Bicluster? Wenn ja welche?

nein

Frage 3

Welche Spalte enthält die meisten Bicluster?

Cond_9 : 16 Bicluster

Frage 4

Gibt es Spalten, die keine Bicluster enthalten? Wenn ja, welche?

Cond_1, Cond_3

Bearbeitungsblatt 4 aus der Pilotstudie

Aufgabe 5 (Datensatz: Rand.dat, Biclsuter: Rand.bic)

Auch hier sind wieder alle Bicluster mit den entsprechenden Farben rechts dargestellt. Bitte beantworten Sie auch hier die Fragen durch Angabe der Bicluster ID bzw. den Zeilen- und Spaltennamen.

Frage 1

Mit welchem Bicluster hat Bicluster B_8 die meisten Überlappungen?

B_1 (13 x 8) = 104:
Zeilen (G): 171, 179, 188, 383, 391, 439, 491, 568, 631, 649, 714, 747, 861
Spalten (C): 2, 3, 8, 15, 20, 16, 12, 4
B_5 (15 x 6) = 90:
Zeilen (G): 17, 127, 141, 244, 539, 589, 664, 667, 713, 813, 814, 896, 922, 941, 976
Spalten (C): 2, 3, 8, 15, 20, 5

Frage 2

Finden Sie eine Zelle (Angabe von Zeile UND Spalte), die in mindestens 3 Biclustern liegt. Geben Sie dann die IDs der Bicluster an, in denen die Zelle liegt.

Bsp: G_913, C_3: B_3, B_5, B_7
G_477, C_3: B_3, B_6, B_6, B_7
G_747, C_4: B_1, B_2, B_8

Frage 3

Es soll nun ein bestimmter Bicluster (B_9) genauer untersucht werden. Bestimmen Sie alle Bicluster mit denen B_9 Zeilen UND Spalten gemeinsam hat und bestimmen Sie zu den jeweiligen Biclustern die gemeinsamen Zeilen- und Spaltennamen.

B_1: G(356, 706) x C(6, 11, 16, 17, 19, 20)
B_3: G(356) x C(19)
B_6: G(497) x C(6, 11, 7)

Bearbeitungsblatt 5 aus der Pilotstudie

Aufgabe 6 (Datensatz: Rand1.dat, Bicluster: Rand1.bic)

Auch hier sind wieder alle Bicluster mit den entsprechenden Farben rechts dargestellt. Bitte beantworten Sie auch hier die Fragen durch Angabe der Bicluster ID bzw. den Zeilen- und Spaltennamen.

Frage 1

Finden Sie denjenigen Bicluster, der mit den meisten anderen Biclustern eine Überlappung aufweist. Geben Sie die IDs der Bicluster an.

B_6: B_1, B_2, B_3, B_4, B_5, B_7, B_9

Bearbeitungsblatt 6 aus der Pilotstudie