

Universität Dortmund  
Fachbereich Informatik Lehrstuhl 11 - Algorithm  
Engineering

**Ausarbeitung**  
Hypothesis Generation in Signaling Networks

**Seminar: Visualisierung in der Bioinformatik**

**von:** Philipp Büdenbender  
**gehalten am:** 25.5.2007

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
1.1	Motivation . . . . .	3
1.2	Analyse . . . . .	3
1.2.1	Das Constrained-Downstream-Problem . . . . .	3
1.2.2	Das Minimum-Knockout-Problem . . . . .	4
<b>2</b>	<b>Biologische Netzwerke</b>	<b>4</b>
2.1	Grundlagen für biologische Netzwerke . . . . .	4
2.1.1	Definition von Pathway-Graphen . . . . .	4
2.2	Der Algorithmus zum Entfernen . . . . .	6
2.3	Zusammenfassung . . . . .	7
<b>3</b>	<b>Das Constrained-Downstream-Problem</b>	<b>8</b>
3.1	Formale Definition: Pathway-Graphen . . . . .	8
3.2	Der Algorithmus für das Constrained-Downstream-Problem . . . . .	9
3.3	Theorem 1: Laufzeit . . . . .	10
3.4	Theorem 2: Korrektheit . . . . .	11
3.5	Zusammenfassung . . . . .	11
<b>4</b>	<b>Das Minimum-Knockout-Problem</b>	<b>12</b>
4.1	Die Formale Definition des Minimum-Knockout-Problems . . . . .	12
4.2	Die Heuristik für das MK-Problem . . . . .	13
4.3	Zusammenfassung . . . . .	13
<b>5</b>	<b>Ergebnisse</b>	<b>14</b>
<b>6</b>	<b>Fazit</b>	<b>15</b>

# 1 Einleitung

## 1.1 Motivation

Biologische Netzwerke (engl: biological signaling networks) beschreiben die chemischen Prozesse von Reaktionsketten, die beim Auflösen, Fortpflanzen und beim Verarbeiten von Signalen in Zellen beteiligt sind. Diese Netzwerke regeln eine Vielzahl von Zellaktivitäten, welche für die korrekte Funktion eines Organismus kritisch sind. Es gibt Störungen dieser Prozessketten wie z.B. Krebs, Herzkrankheiten, Kongenitale Abweichungen, Metabolische Störungen und Immunologische Abweichungen. Hierzu hat die Wissenschaft sehr große Datenmengen zusammengetragen. Diese Datensammlungen werden in so genannten Datenbanken organisiert und beinhalten Moleküle samt deren Beziehungen untereinander. Aufgrund dieser großen Datenmengen ist es sehr wichtig diese visuell aufzubereiten um sich in ihnen leichter zurechtzufinden. Um dieses bei der gegebenen Datenmenge zu bewältigen brauchen wir Techniken zum Filtern, Suchen und Validieren.

## 1.2 Analyse

Es gibt im Grunde drei große Bereiche der Analyse dieser Biologischen Netzwerke:

### 1. Visualisierung

Hier hat man in der letzten Zeit versucht mit Model Checking und Formaler Verifikation diese Netzwerke zu prüfen.

### 2. High-Level-parametric-simulation

Die Verwendung dieser Methode wird in der Realität sehr oft verwendet.

### 3. Detaillierte Simulation von kleinen Teilnetzwerken

Hierbei ist das Problem, dass man sehr oft sehr viele Parameter braucht, die man aber de facto nicht hat.

Hypothesis-generation-Tools sollen den Biologen helfen die Ergebnisse von Experimenten hervor zuzusagen. Sie sollen helfen Laborzeiten zu reduzieren. Hierbei ist oft der große Nachteil, dass Details benötigt werden, die man in der Realität oft nicht hat. Bei der von den Autoren des Papers vorgestellten Netzwerken braucht man nur zwei Dinge. Als erstes braucht man einen Reaktanten und dann noch die Produkte einer Reaktion. Dazu benutzten wir Graphen basierende Netzwerke. Der Vorteil hierbei ist, dass wir uns Graphen Algorithmen bedienen, die schon bekannt sind und wo schon sehr viel geforscht wurde.

#### 1.2.1 Das Constrained-Downstream-Problem

Dieses Problem wird benutzt beim Entwurf von Medikamenten. Es wird nach einer Menge von Reaktionen gesucht, die von einer Menge von Molekülen in eine andere Menge von Molekülen führen. Damit kann man nun gewisse Moleküle

hemmen, gleichzeitig aber den Signalfluss als ganzes erhalten, also zu konservieren. Ein biologischer Endpunkt wäre es, ein Molekül zu kennzeichnen, um eine starke Verbreitung zu hemmen und gleichzeitig die metabolischen Funktionen zu konservieren. Dazu werden wir später einen Algorithmus kennen lernen, der in polynomialzeit arbeitet und als korrekt bewiesen wurde.

### 1.2.2 Das Minimum-Knockout-Problem

Das zweite große Problem, welches wir kennen lernen ist das Minimum-Knockout-Problem. Es ist ein sehr wichtiges Problem für die Krebsforschung. Das Ziel dieses Problems ist es, die Minimale Menge von Molekülen zu finden, die geblockt werden müssen um die Netzfunktionen zu stören. Das Problem bei der Klassischen Chemotherapie ist, daß Zellen aus bösartigem, wie auch aus normalem Gewebe getötet werden. In den letzten Jahren hat man Medikamente entwickelt, die auf Moleküle zielen, die abweichend in Krebszellen sind. Untersuchungen haben ergeben, dass diese Medikamente am besten mit anderen Medikamenten zusammen arbeiten. Von daher versucht man nicht ein Medikament gegen Krebs zu finden, sondern kombiniert verschiedene, die dann gewisse Funktionen hemmen sollen.

## 2 Biologische Netzwerke

### 2.1 Grundlagen für biologische Netzwerke

Die Standard Modelle von biologischen Netzwerken umfassen, wie bei uns auch, Interaktionen von Molekülen, welche in Zellen und deren Membran auftreten (vgl. Abb. 1). Diese Netzwerke beinhalten dann immer zwei Komponenten. Das erste sind die Moleküle. Diese sind in der Zeichnung die runden Knoten. Als zweites haben wir dann die Interaktionen. Sie werden bei uns als Rechtecke gezeichnet. Auf die Details werden wir später noch eingehen.

Eine Auffallende Eigenschaft von diesen Biologischen Netzwerken sind allerdings das Auftreten von Schleifen (s.h. Abb. 1 ,  $B \rightarrow C \rightarrow D \rightarrow B \rightarrow \dots$ ). Ferner gibt es dann noch positive und negative Rückkopplungsschleifen. Negative Rückkopplungsschleifen verringern z.B. die Tätigkeit. .

#### 2.1.1 Definition von Pathway-Graphen

Ein Pathway-Graph ist ein gerichteter Graph  $G = (V^\circ, V^\square, E)$  Dabei sind:

$V^\circ \doteq$  Molekülknoten (s.h. die Kreise in Abb. 1)

$V^\square \doteq$  Interaktionsknoten (s.h. die Rechtecke in Abb. 1)

1.  $V^\circ \cap V^\square = \emptyset$
2. Für jedes  $(u, v) \in E$  sind  $u$  und  $v$  nicht vom gleichen Typ.

Die erste Eigenschaft impliziert, daß jeder Knoten im Graphen entweder ein Molekül- oder ein Interaktionsknoten ist. Die zweite Eigenschaft drückt aus, daß es Biologisch gesehen nicht möglich ist, ohne Reaktion von einem Molekül zu einem anderen zu kommen. Ebenso ist natürlich auch die Umkehrung gültig,

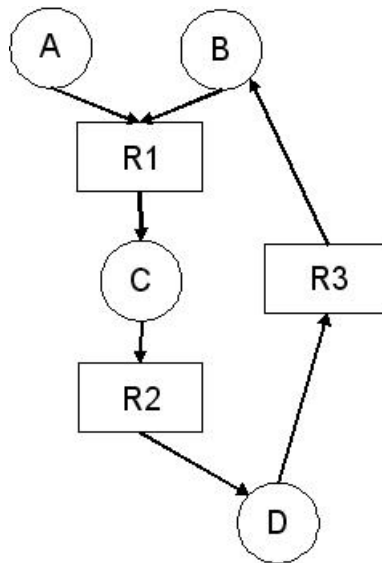


Abbildung 1: Ein Biologisches Netzwerk

daß wir ohne Moleküle nicht von einer Reaktion in die nächste gelangen können.

Nun kommen wir zu einem interessanten Teil der Pathway-Graphen, dem Effekt des Entfernens. Dieses simuliert die Blockierung von großen Teilen in unserem Netzwerk. In Pathway-Graphen hat das Entfernen oft den Effekt, daß sich das Entfernen von Knoten auf andere Teile des Pathways auswirkt. Dieses hängt auch oft mit der Struktur biologischer Netze zusammen, aber dazu sehen wir später noch mehr. Wenn wir nun ein Molekül entfernen, kann dieses eine ganze Kette nach sich ziehen. Also die Interaktionen in die das entfernte Molekül beteiligt ist können nicht statt finden. Dieses hat nun wiederum zur Folge, dass auch die daraus entstehenden Moleküle nicht produziert werden. Nun fehlen diese wieder in deren beteiligten Reaktionen und so weiter. Somit können oft große Teile in Graphen entfernt werden auch wenn wir nur wenige Moleküle entfernen wollten.

<p><b>Algorithmus : Remove</b></p> <p><b>Eingabe :</b> <math>G(V^{\circ}, V^{\square}, E)</math>, <math>v</math>, <math>Y</math></p> <p>1 <b>wenn</b> <math>v \in Y</math> <b>dann</b></p> <p>2     Return</p> <p>3 <math>X \subseteq V^{\circ} \cup V^{\square}</math> ist die Menge der Kinder von <math>v</math></p> <p>4 <b>für</b> jedes <math>x \in X</math> <b>tue</b></p> <p>5     Entferne Kante <math>(v, x)</math> aus <math>E</math></p> <p>6     <b>wenn</b> <math>x \in V^{\circ}</math> und <math>\text{Eingangsgrad}(x) = 0</math> <b>dann</b></p> <p>7         Remove <math>(G, x, Y)</math></p> <p>8     <b>wenn</b> <math>x \in V^{\square}</math> <b>dann</b></p> <p>9         Remove <math>(G, x, Y)</math></p> <p>10 Entferne <math>v</math> aus <math>V^{\circ} \cup V^{\square}</math></p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Algorithmus 1** : Remove[1]

## 2.2 Der Algorithmus zum Entfernen

Kommen wir nun also zu dem Algorithmus, um Knoten in unserem Pathway-Graphen zu entfernen (siehe Algorithmus 1). Der Algorithmus wird mit  $\text{Remove}(G, v, Y)$  aufgerufen. Dabei steht  $G$  für den Graphen aus dem wir einen Knoten entfernen wollen. Beim  $v$  handelt es sich um den Knoten, der entfernt werden soll. Dabei kann es sich um einen Molekül oder einen Interaktionsknoten handeln. Schließlich haben wir noch die Menge  $Y$ . In dieser Menge stehen jene Moleküle die wir explizit vor dem Entfernen schützen wollen, da wir z.B. etwas erhalten wollen.

Nach dem Aufruf des Algorithmus arbeitet sich dieser Rekursiv durch den gegebenen Graphen. Im ersten Schritt wird einfach nur überprüft, ob es sich bei gegebenem Knoten um einen Knoten aus unserer Menge  $Y$  handelt. Wenn ja, dann ist die Rekursion an dieser Stelle beendet. Wenn nein, fährt der Algorithmus mit dem Entfernen fort. In Schritt 2 und 3 werden nun alle Kinder des zu entfernenden Knoten betrachtet. Als erstes wird immer die Kante zu dem nachfolgenden Knoten gelöscht. Handelt es sich bei dem nachfolgenden Knoten um einen Molekülknoten und hat dieser keine andere eingehende Kante so wird dieser ebenfalls rekursiv gelöscht (Aufruf durch  $\text{Remove}(G, \text{Nachfolger}, Y)$ ). Falls es sich beim dem Nachfolger um eine Interaktionsknoten handelt, wird dieser auf jeden Fall gelöscht. Wenn nun alle Nachfolger auch wieder deren Nachfolger gelöscht haben kehrt der Algorithmus aus der Rekursion zurück und löscht am Ende den Knoten selber aus dem Graphen.

Dieser Algorithmus beschreibt im Grunde den Verarbeitungseffekt in Pathway-Graphen. Wenn nur ein Knoten gelöscht wird, so kann dieses große Auswirkungen auf den ganzen Graphen haben. Diesem Effekt versucht man nun mit der Menge  $Y$  vorzubeugen um somit gewisse Teile des Graphen zu schützen. Eine Annahme, die bei dem Algorithmus getroffen wird ist, daß alle Reaktanten in eine Reaktion eingehen müssen, damit diese stattfinden kann. Dieses ist in der Realität nicht immer so. Oft haben Moleküle nur die Eigenschaft die Geschwin-

digkeit einer Reaktion zu beeinflussen. Hier haben wir also eine Vergrößerung der Realität.

### **2.3 Zusammenfassung**

Abschließend wollen wir noch einmal eine kurze Zusammenfassung des Kapitels geben. Wir haben Pathway-Graphen kennen gelernt. Dies benutzen wir zur Beschreibung von Aktivitäten von Molekülen. Bei Pathway-Graphen haben wir zwei Knotenarten. Zum einen die Molekülknoten und zum anderen die Interaktionsknoten. Auf einen Molekülknoten muss immer ein Interaktionsknoten folgen und umgekehrt. Ferner haben wir nun einen Algorithmus um Knoten im Graphen zu Entfernen. Dieser kann nach sich ziehende Aktivitäten ebenfalls löschen und eine Menge von Knoten vor dem Löschen bewahren. Für unsere Graphen brauchen wir kaum Details, haben aber teilweise eine Vergrößerung der Realität

### 3 Das Constrained-Downstream-Problem

Das Constrained-Downstream-Problem (Deutsch: "Das abwärts bewegendes Problem") behandelt, das Problem in gegebenem Pathway-Graphen einen Weg von einer Start Knoten Mengen in eine Zielknoten Menge zu finden. Dabei hat man zwei Mengen. Eine Menge enthält Knoten, die nicht auf dem Weg liegen dürfen. Die andere Menge ist für das Gegenteil. Sie enthält Knoten, die auf jedem Weg von S(ource) nach T(arget) besucht werden müssen. Die Biologie hat dieses Problem. In Pathway-Graphen sind alle Moleküle die von einer Menge von Molekülen aus erreichbar sind, abhängig von diesen. Dieses hat zur Folge, dass Schleifen diese Mengen sehr groß machen können. Um diese Menge zu reduzieren sucht man also nach einer Suche über diese Knoten. Man benutzt dieses um gewisse Dinge zu hemmen und gleichzeitig andere zu bewahren. Dies könnte wiederum durch andere Medikamente erfolgen.

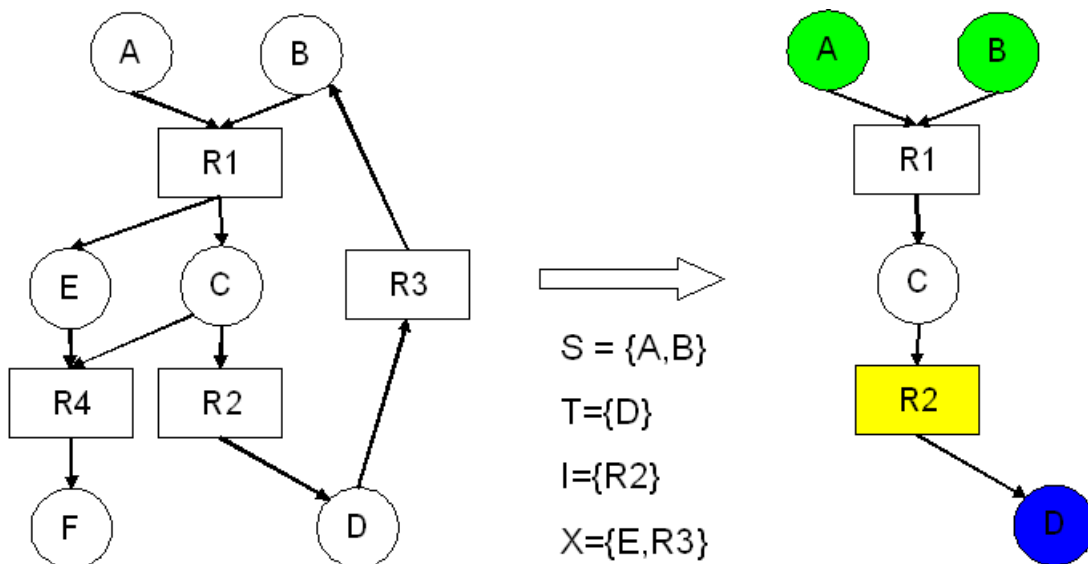


Abbildung 2: Ein Beispiel für einen Constrained-Downstream-Problem

#### 3.1 Formale Definition: Pathway-Graphen

Nun kommen wir zu der Formalen Definition von Pathway-Graphen.

Eingabe: Pathway Graph  $G = (V^{\circ}, V^{\square}, E)$ , vier Mengen  $S, T \subset V^{\circ}$  und  $I, X \subset (V^{\circ} \cup V^{\square})$

Ausgabe: Subgraph  $G' = (U^{\circ}, U^{\square}, E')$  wobei

1.  $U^{\circ} \subseteq V^{\circ}$ ,  $U^{\square} \subseteq V^{\square}$  und  $E' \subseteq E$ ;
2.  $\forall u \in (U^{\circ} \cup U^{\square}), \exists (s \in S, t \in T)$ , so dass  $s \rightarrow u$  und  $u \rightarrow t$ ;

3.  $(U^\circ \cup U^\square) \cap X = \emptyset$ ;
4. Jeder Pfad von S nach T führt durch einen Knoten aus I.
5.  $G'$  ist der maximale Subgraph, der 1-4 erfüllt

Wir haben also als Eingabe unseren Graphen  $G$  und wie bereits erwähnt die vier Mengen  $S$ ,  $T$ ,  $I$  und  $X$ . Die ersten beiden Mengen beschreiben unsere Start und Zielmengen.  $I$  und  $X$  sind wie bereits erwähnt die Mengen für die Knoten, die auf jedem Weg liegen müssen ( $I$ ) und die Knoten, die auf keinem Weg liegen dürfen ( $X$ ). Ein Graph  $G'$  der nun also die oben beschriebenen Eigenschaften erfüllt, hat also das Constrained-Downstream-Problem gelöst.

In unserem Beispiel (vgl. Abbildung 2) sehen wir also einen solchen Graphen. Die grüne Menge sind unsere Knoten der Menge  $S$ . Der blaue Knoten ist die Menge  $T$ . Ferner haben wir dann noch die Menge  $I$ , die den Knoten  $R2$  enthält und die Menge  $X$  mit den Knoten  $E$  und  $R3$ . Der Graph auf der rechten Seite erfüllt nun alle Punkte der Formalen Definition von Pathway-Graphen. Dem Leser bleibt es an dieser Stelle überlassen dieses Nachzuprüfen.

### 3.2 Der Algorithmus für das Constrained-Downstream-Problem

**Algorithmus** : FindDownstream( $G=(V^\circ, V^\square, E), S, T, I, X$ )  
**Eingabe** :  $G(V^\circ, V^\square, E), S, T, I, X$   
**Ausgabe** :  $G$

- 1  $G = \text{Remove}(G, X, T)$
- 2  $\forall t \in T, \text{Visited}[t] = 1$
- 3  $\forall t \notin T, \text{Visited}[t] = 0$
- 4  $\forall t \in T, \text{OnPath}[t] = 1$
- 5  $\forall t \notin T, \text{OnPath}[t] = 0$
- 6  $\forall v \in V^\circ \cup V^\square, \text{AboveInclude}[v] = 0$
- 7  $G' = (\emptyset, \emptyset, \emptyset)$
- 8 **für jedes**  $s \in S$  **tue**
- 9      $\lfloor$  CalcDownstream( $G, s, I, \emptyset, G'$ )
- 10 Return  $G'$

**Algorithmus 2** : FindDownstream[1]

```

Algorithmus : CalcDownstream( $G, v, I, P, G'$ )
Eingabe :  $G(V^\circ, V^\square, E), v, I, P, G'$ 

1 wenn  $Visited[v] == 0$  dann
2   |  $Visited[v] = 1$ 
3   |  $C = \text{Kinder von } v$ 
4   | Für jedes  $c \in C$   $\text{CalcDownstream}(G, c, (p_1, \dots, p_k, v))$ 
5 sonst
6   | wenn  $OnPath[v] == 0$  dann
7   |   |  $\text{Return}$ 
8   |  $\forall p \in P, OnPath[p] = 1$ 
9   | wenn  $AboveInclude[v] == 1$  dann
10  |   |  $VG' = VG' \cup P$ 
11  |   |  $EG' = EG' \cup (p_1, p_2), \dots, (p_{k-1}, p_k)$ 
12  |   |  $\forall p \in P, AboveInclude[p] = 1$ 
13  | sonst wenn  $P \cap I \neq \emptyset$  dann
14  |   |  $VG' = VG' \cup P$ 
15  |   |  $EG' = EG' \cup (p_1, p_2), \dots, (p_{k-1}, p_k)$ 
16  |   |  $\forall p \in P, AboveInclude[p] = 1$ 
17  | sonst
18  |   |  $\text{Return}$ 
19  |

```

**Algorithmus 3** : CalcDownstream[1]

Unter Algorithmus 2 und 3 sehen wir den vorgestellten Algorithmus in zwei Teilen. Der erste Teil (Algo 2) wird mit FindDownstream ( $G, S, T, I, X$ ) aufgerufen. Als erstes werden mittels des Remove Algorithmus, den wir ja bereits kennen gelernt haben, alle Knoten aus der Menge  $X$  aus dem Graphen entfernt. Zu beachten ist hierbei, dass wir unsere Zielknoten sichern. Diese sollen auf keinen Fall gelöscht werden. Als nächstes werden Hilfsvariablen die wir für den Algorithmus brauchen initialisiert. Der Algorithmus ist eine Implementierung einer Tiefen Suche in unserem gegebenen Graphen. Die Menge Visited beinhaltet alle Knoten, die bereits besucht wurden. OnPath beschreibt das die Knoten bereits auf einem Weg zur Zielmenge mit Knoten  $I$  liegen. Dieses erspart uns einige Aufrufe des Algorithmus. Insgesamt hat der Algorithmus eine Laufzeit von  $O(|V^\circ \cup V^\square|)$ , da wir eine Tiefensuche machen.

### 3.3 Theorem 1: Laufzeit

Die Aussage ist, daß der FindDownstream-Algorithmus in Zeit  $O(|V^\circ \cup V^\square|)$  läuft und  $V^\circ$  und  $V^\square$  Knoten des Input Graphen sind. Um dieses zu zeigen, genug es zu zeigen, daß der CalcDownstream ein DFS Algorithmus ist und die Laufzeit von  $O(|V^\circ \cup V^\square|)$  braucht. Schließlich brauchen alle Aufrufe um einen Knoten zu benennen  $O(|V^\circ \cup V^\square|)$ . Von daher ergibt sich die bereits oben erwähnte Laufzeit.

### 3.4 Theorem 2: Korrektheit

Theorem: Sei  $(G, S, T, I, X)$  eine Eingabe für den Algorithmus FindDownstream und  $G'$  die Ausgabe.

Um die Korrektheit zu zeigen muss man nun die Formalen Bedingungen, die wir bereits kennen gelernt haben beweisen. Es werden nur Knoten aus  $G$  in  $G'$  eingefügt (Bed. 1).  $G'$  erfüllt die Bedingungen 1-5 die ein solcher Graph haben muss. Wir rufen in dem Algorithmus jeden Knoten aus  $S$  auf und jeder dieser Knoten führt zu einem Knoten aus  $T$  (Bed. 2). Die dritte Bedingung ist ebenfalls erfüllt, da am Anfang ja alle Knoten aus  $X$  gelöscht werden. Es wird ferner gewährleistet, dass jeder Knoten in  $G'$  durch mindestens einen Knoten aus  $I$  führt. (Bed.4) Der Maximale Graph wird per Indirektem Beweis bewiesen. (Falls es größeren gibt, so müsste es einen Knoten vom Start zu diesem geben und einen von diesem zum Ende und somit wäre er auch in unserem Graph). Der genaue Beweis kann im Paper nachgelesen werden.

### 3.5 Zusammenfassung

Nun sind wir auch schon am Ende des Kapitels, daß sich mit dem Constrained-Downstream-Problem befasst hat. Abschließen wollen wir noch einmal kurz Zusammenfassen, was wir haben. Wir haben das Constrained-Downstream-Problem kennen gelernt. Man sucht einen Weg von einer Startmenge  $S$  in eine Zielmenge  $T$ . Dabei gibt man eine Menge von Knoten  $I$  und  $X$  vor. Die Knoten aus der Menge  $I$  müssen auf den Wegen von  $S$  nach  $T$  enthalten sein und keiner der Knoten aus der Menge  $X$  darf auf dem Weg besucht werden. Dazu haben wir eine Formale Definition kennen gelernt und einen haben nun einen polynomialzeit Algorithmus. Dieser wurde sowohl in seiner Laufzeit, als auch in seiner Korrektheit bewiesen.

## 4 Das Minimum-Knockout-Problem

Das Minimum-Knockout-Problem ist ein Problem, bei dem eine Minimale Menge von Knoten gesucht wird, so dass das Entfernen dieser Knoten eine gegebene Menge von Molekülen von einer anderen Menge trennt. In der Krebsforschung ist dieses ein sehr zentrales und wichtiges Problem. Das Problem der traditionellen Chemotherapie ist ja, daß diese gute und böse Zellen, die sich schnell vermehren, abtötet. So muss man also die minimale Menge der zu blockierenden Knoten finden um Krebs effektiv zu bekämpfen. Somit sind wir also bei unserem Problem. Dieses wollen wir nun einmal formal definieren.

### 4.1 Die Formale Definition des Minimum-Knockout-Problems

Eingabe: Pathway-Graph  $G = (V^\circ, V^\square, E)$ , zwei Mengen  $S, T \subset V^\circ$  und pos. Int  $Q$

Frage: Gibt es eine Menge  $U \subseteq ((V^\circ \cup V^\square) - (S \cup T))$  mit  $|U| \leq Q$ , so dass  $\text{Remove}(G, U, S \cup T)$  einen Graphen  $G'$  erzeugt in dem für jedes  $s \in S$  und  $t \in T$   $s \not\rightarrow t$  gibt.

Dieses Problem wird im originalen Paper als NP-hard bewiesen. Es kann durch eine Reduzierung auf das Minimum-Set-Cover-Problem bewiesen werden. Von daher ist der Ansatz hier eine gute Heuristik zu finden. Diese wollen wir jetzt einmal vorstellen. Vorher wollen wir aber kurz noch ein Beispiel für ein Minimum-Knockout-Problem zeigen.

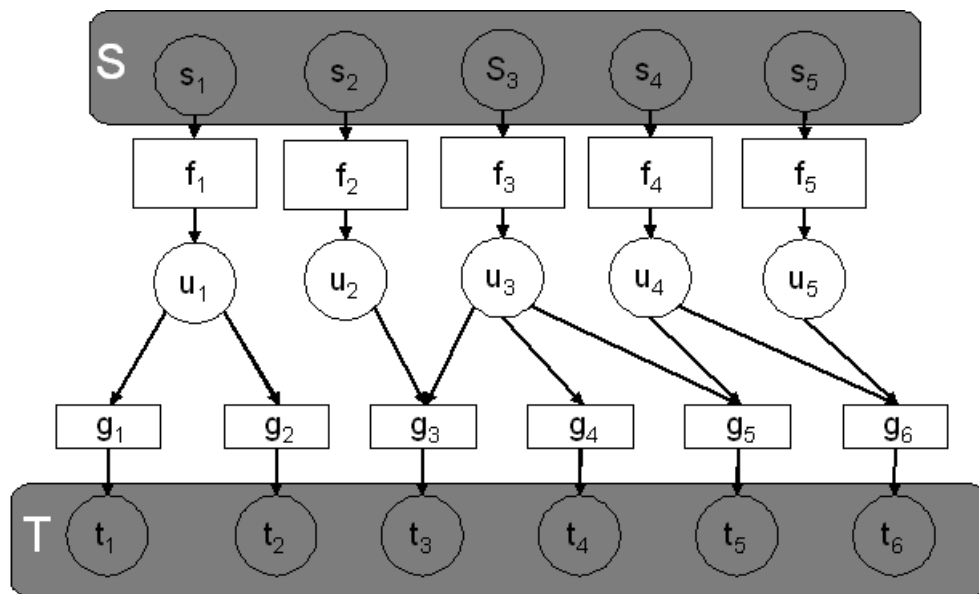


Abbildung 3: Ein Beispiel für ein Constrained-Downstream-Problem [1]

Wenn wir uns die Abbildung 3 anschauen sehen wir ein solches Problem. Wir haben als Startmenge die Knoten  $s_1, \dots, s_5$ . Die Knoten unserer Zielmenge sind die Knoten  $t_1, \dots, t_6$ . Dazwischen haben wir nun einige Knoten. Wir suchen nun

also die minimale Menge der Knoten, die wir blockieren müssen, damit es keinen Weg mehr von S nach T gibt. In diesem Beispiel ist die Lösung die Menge  $u_1$ ,  $u_3$  und  $u_5$ . Dieses ist hier leicht zu finden. In großen Graphen sieht dieses wiederum anders aus. Dazu wollen wir nun eine geeignet Heuristik kennen lernen.

## 4.2 Die Heuristik für das MK-Problem

Im folgenden Abschnitt lernen wir eine Heuristik für das Minimum-Knockout-Problem kennen. Die Idee dieser Heuristik basiert auf dem zufälligen Entfernen von Knoten aus dem gegebenen Graphen.

**Algorithmus** : Minimum-Knockout (G,S,T,m)

**Eingabe** : G, S, T, m

**Ausgabe** : maximaler G'

```

1 U = FindDownstream(G,S,T,∅,∅)
2  $U^\circ = U \cap V^\circ$ 
3  $U^\square = U \cap V^\square$ 
4  $C = \{u \in U : u \in U^\circ \vee u \in (\text{Children}(S) \cap U^\square)\}$ 
5 für  $i = 1$  bis  $m$  tue
6   |  $G' = G$ 
7   |  $S_i = \emptyset$ 
8   | solange  $S \rightsquigarrow^{G'} T$  tue
9   |   |  $c \in (C - S_i)$ 
10  |   |  $G' = \text{Remove}(G',c,S,T)$ 
11  |   |  $S_i = S_i \cup c$ 
12  $J = \text{argmax}_i |S_i|$ 
13 Return  $S_j$ 

```

**Algorithmus 4** : Minimum-Knockout[1]

Unter Algorithmus 4 sehen wir den Algorithmus für unser Problem. Im ersten Schritt des Algorithmus werfen wir alle Knoten raus, die nicht zu den Wegen zwischen S und T gehören. Der folgende Teil des Algorithmus ist relativ einfach. Es folgt solange eine Entfernung von Knoten, bis man keinen Weg mehr von S nach T hat. Ein Knoten wird zufällig ausgewählt und entfernt. Am Ende wird die kleinste gefundene Menge zurückgegeben. Der Algorithmus benutzt den FindDownstream Algorithmus, den wir ja bereits im letzten Kapitel kennen gelernt haben. Der Algorithmus hat keine untere Schranke und arbeitet sehr schnell.

## 4.3 Zusammenfassung

Wir haben in diesem Kapitel das Minimum-Knockout-Problem kennen gelernt. Wir benutzen es um von einer Startmenge S alle Wege in eine Zielmenge T zu blockieren. Es ist ein wichtiges Problem in der Forschung zur Bekämpfung von Krebs. Wir haben erfahren, dass dieses Problem NP-hart ist und haben eine gute Heuristik für dieses Problem kennen gelernt. Sie liefert gute Ergebnisse bezüglich unserer Eingabe. Sie ist recht schnell und sehr einfach zu implementieren. Sie

baut auf den bereits gelernten Dingen auf und benutzt diese. Im folgenden Kapitel wollen wir uns die Ergebnisse der Heuristik einmal angucken.

## 5 Ergebnisse

Die Autoren des Papers haben zu Testzwecken einen Graphen aus dem EGFR Netzwerk genommen. Dabei haben sie 100 Iterationen von (S, T) Paaren ausprobiert und sind zu den folgenden Ergebnissen gekommen:

- Die Heuristik liefert immer eine minimale Knockout Menge von 1, solange eine minimale Knockout Menge die Kardinalität ähnlich 1 hat.
- Sehr oft taucht eine minimale Knockout Menge 1 auf. Dieses ist durch die Connectivität des Netzes zu erklären
- In Abbildung 4 ist zu sehen, dass über die Hälfte aller Knoten eine sehr umfangreiche Verbindung haben
- Remove-Operation entfernt oft große Teile
- Die Heuristik findet immer eine Minimale-Knockout-Menge
- Eine Eigenschaft des Netzwerkes ist es, dass wir eine 50% Wahrscheinlichkeit haben einen Knoten zufällig auszuwählen, der bis zu 400 Knoten verbindet
- Praktisch ist die Heuristik aber sehr gut, da sie viele Fälle im EGFR Netzwerk gut löst
- Sehr schnell (ca. 1 Sekunde)

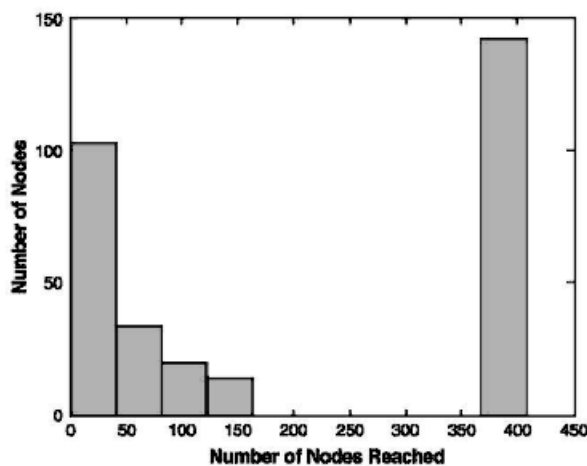


Abbildung 4: Der Aufbau von Pathway-Graphen [1]

Somit haben wir alle wesentlichen Punkte der Heuristik aufgezählt. Leider wurden hier nicht mehrere Graphen getestet. Somit müssten natürlich noch einige der oben genannten Eigenschaften weiter untersucht werden.

## 6 Fazit

Wir haben eine graphische Modellierung von Biologische Netzwerke kennen gelernt. Somit können wir nun also Biologische Netzwerke visualisieren. Wir brauchen für unsere Graphen nur wenige biologische Details. Es wurden zwei wichtige Fragen in der Biologie aufgegriffen. Dazu haben wir gute Algorithmen kennen gelernt. Ferner haben wir noch für unser NP-hartes Problem des Minimum-Knockout-Problems eine gute Heuristik kennen gelernt.

Die hier vorgestellten Algorithmen sind im Grunde aber nichts Neues. Es wird für das Entfernen ein naiver Algorithmus benutzt. Der Algorithmus für das Lösen des Constrained-Downstream-Problems ist die im wesentlichen eine Tiefensuche im gegebenem Graphen. Die vorstellte Heuristik für das Minimum-Knockout-Problem ist im im Kern eine Naive randomisierte Lösung. Leider wurde diese nur unzureichend getestet und somit ist es natürlich schwierig aufgrund dieser Tatsache hier ein Fazit zu ziehen.

## Liste der Algorithmen

1	Remove[1]	6
2	FindDownstream[1]	9
3	CalcDownstream[1]	10
4	Minimum-Knockout[1]	13

## Abbildungsverzeichnis

1	Ein Biologisches Netzwerk	5
2	Ein Beispiel für einen Constrained-Downstream-Problem	8
3	Ein Beispiel für einen Constrained-Downstream-Problem [1]	12
4	Der Aufbau von Pathway-Graphen [1]	14

## Literatur

- [1] Derek A. Ruths, Luay Nakhleh, M. Sriram Iyengar, Shrikanth A.G. Reddy, and Prahlad T. Ram. Hypothesis generation in signaling networks. *Journal of Computational Biology*, 2006.