Ein generisches Programm für inklusionsminimale Teilmengen mit einer graphentheoretischen Anwendung

Rudolf Berghammer

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel
Olshausenstraße 40, D-24098 Kiel

Zusammenfassung Wir entwickeln formal ein generisches Programm für inklusionsminimale Teilmengen und betrachten Verfeinerungen davon. Eine Instantiierung der Endversion führt zu einer Verbesserung eines bekannten Verfahrens zur Approximation einer anzahlminimalen Knotenüberdeckung bei Graphen.

1 Einleitung

Neben der Berechnung von anzahlminimalen bzw. -maximalen Teilmengen eines gegebenen Universums M spielt auch die Berechnung von inklusionsminimalen bzw. -maximalen Teilmengen von M eine große Rolle bei Optimierungsproblemen. Einerseits fragt ein Problem oft direkt nach inklusionsminimalen bzw. -maximalen Teilmengen. Beispiele sind spannende Wälder und Knotenbasen von Graphen. Im ersten Fall sucht man inklusionsmaximale kreisfreie Kantenmengen und im zweiten Fall sucht man inklusionsminimale Knotenmengen, von denen aus alle anderen Knoten erreichbar sind. Weiterhin werden inklusionsminimale bzw. -maximale Teilmengen auch gerne als Startpunkte zur Berechnung von anzahlminimalen bzw. -maximalen Teilmengen verwendet um letztere zu beschleunigen. Matchings (das sind Mengen von paarweise nicht inzidierenden Kanten von Graphen) sind dazu ein gutes Beispiel. Man berechnet in der Regel zuerst sehr einfach und effizient ein inklusionsmaximales Matching und startet erst anschließend den relativ aufwendigen Prozeß der iterativen Suche nach den sogenannten vergrößernden alternierenden Ketten um ein anzahlmaximales Matching zu erhalten. Schließlich werden inklusionsminimale bzw. -maximale Teilmengen auch als Approximationen verwendet, wenn eine Berechnung von entsprechenden anzahlminimalen bzw. -maximalen Teilmengen aus komplexitätstheoretischen Gründen effizient nicht möglich zu sein scheint. Dies ist etwa bei transitiven Reduktionen von Graphen der Fall. Eigentlich ist man nämlich an Teilgraphen mit möglichst kleiner Kantenanzahl interessiert, die die gleiche reflexiv-transitive Hülle wie das Original besitzen. Nur aufgrund der NP-Vollständigkeit dieses Problems beschränkt man sich auf transitive Reduktionen, das sind Teilgraphen, die die gleiche reflexiv-transitive Hülle wie das Original besitzen und aus denen man keine Kante mehr entfernen darf, ohne daß diese Eigenschaft verletzt wird.

In diesem Artikel beschränken wir uns auf die Berechnung von inklusionsminimalen Teilmengen. Wir entwickeln zuerst in Abschnitt 2 formal mittels der Invariantentechnik ein generisches Programm, das zu einem endlichen Universum M, einer Teilmenge R von M und einem speziellen Prädikat $\mathcal{P}$ auf 2^M eine inklusionsminimale Teilmenge von R berechnet, für die $\mathcal{P}$ gilt. Im gleichen Abschnitt betrachten wir auch noch zwei Verfeinerungen dieses Programms, die es in vielen Fällen erlauben, die Effizienz bei einer konkreten Instantiierung teils beträchtlich zu verbessern. Abschnitt 3 demonstriert dann Instantiierungen der entwickelten Programme zur Approximation einer anzahlminimalen Knotenüberdeckung eines Graphen. Dies führt schließlich zu einer Verbesserung eines in der Literatur sehr bekannten Approximationsverfahrens für dieses Problem, was wir auch durch entsprechende Testdaten belegen. Wir schließen in Abschnitt 4 mit einigen Bemerkungen zu derzeitigen und geplanten Untersuchungen.

2 Berechnung inklusionsminimaler Teilmengen

Gegeben seien eine endliche Menge M, ein Prädikat $\mathcal{P}$ auf der Potenzmenge 2^M und ein Element R dieser Potenzmenge. Von $\mathcal{P}$ nehmen wir an, daß es *nach oben vererbend* ist, was heißt, daß

$$X \subseteq Y \wedge \mathcal{P}(X) \implies \mathcal{P}(Y) \tag{1}$$

für alle $X, Y \in 2^M$ gilt. Eine unmittelbare Folgerung von Implikation (1) ist, daß die Negation $\neg\mathcal{P}$ von $\mathcal{P}$ sich *nach unten vererbt*, also die Implikation

$$X \supseteq Y \wedge \neg\mathcal{P}(X) \implies \neg\mathcal{P}(Y) \tag{2}$$

für alle $X, Y \in 2^M$ zutrifft. In diesem Abschnitt entwickeln wir zuerst formal ein imperatives generisches Programm *Min*, das für die Eingabe R eine inklusionsminimale Teilmenge von R berechnet, die $\mathcal{P}$ erfüllt.

Wenn wir die Variable A dazu verwenden, das Resultat von *Min* abzuspeichern, dann ist die Bedingung an A, die nach Beendigung des Programms zu gelten hat, also die *Nachbedingung* $\mathit{post}(R, A)$, formal gegeben als

$$\mathit{post}(R, A) :\iff \mathcal{P}(A) \wedge A \subseteq R \wedge \forall X \in 2^A : \mathcal{P}(X) \to X = A\,.$$

Zur formalen Entwicklung des Programs *Min* kombinieren wir die sogenannte Dijkstra-Gries-Methode (man vergleiche beispielsweise mit den Lehrbüchern [4, 5]) mit den Gesetzen der Mengentheorie und der Logik. Dabei schreiben wir im folgenden für eine Menge $X \in 2^M$ und ein Element $x \in M$ vereinfachend $X - x$ statt $X \setminus \{x\}$ und $X + x$ statt $X \cup \{x\}$ um die Lesbarkeit zu verbessern. Wir zielen auf ein Programm ab, das aus einer Initialisierung und einer sich daran anschließenden While-Schleife besteht. Die Herleitung eines solchen Programms geschieht üblicherweise in drei Teilen.

Zuerst haben wir die Nachbedingung zu einer *Schleifeninvariante* konjugiert mit der Bedingung zum Verlassen der While-Schleife zu verstärken. Hier folgen wir der oft sehr erfolgreichen Technik der Problemverallgemeinerung. Einzelheiten und auch viele Beispiele zu dieser Vorgehensweise findet man beispielsweise wiederum in [4, 5]. Die entsprechende Rechnung führt in unserem Fall eine neue Variable B ein und ist nachfolgend angegeben:

$$\begin{array}{ll} & \mathit{post}(R, A) \\ \Longleftrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge \forall X \in 2^A : \mathcal{P}(X) \rightarrow X = A \\ \Longleftrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge \forall X \in 2^A : X \neq A \rightarrow \neg\mathcal{P}(X) \\ \Longleftrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge \forall x \in A : \neg\mathcal{P}(A - x) \\ \Longleftarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge B = \emptyset \wedge B \subseteq A \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A - x) \,. \end{array}$$

Nur die Richtung "$\Leftarrow$" des dritten Schritts ist nicht trivial und bedarf einer Erklärung: Falls $X \subset A$ zutrifft, dann gibt es ein $x \in A$ mit $X \subseteq A - x$ und aus $\neg\mathcal{P}(A - x)$ folgt somit $\neg\mathcal{P}(X)$ wegen Eigenschaft (2). Geleitet von der gerade gezeigten Implikation definieren wir nun

$$\mathit{inv}(R, A, B) \;:\Longleftrightarrow\; \mathcal{P}(A) \wedge A \subseteq R \wedge B \subseteq A \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A - x)$$

als Schleifeninvariante $\mathit{inv}(R, A, B)$ und wählen $B = \emptyset$ als Abbruchbedingung für die While-Schleife. In Pascal-artiger Syntax und mit der Nachbedingung und der Schleifeninvariante als Kommentare sieht dann die bisher erreichte Programmstruktur folgendermaßen aus:

$$\begin{array}{l} \ldots \\ \{ \; \mathit{inv}(R, A, B) \; \} \\ \textbf{while } B \neq \emptyset \textbf{ do} \\ \quad \cdots \textbf{ od}; \\ \{ \; \mathit{post}(R, A) \; \} \,. \end{array}$$

Als nächstes haben wir die (kollaterale) Initialisierung der beiden Variablen A und B zu betrachten. Wir tun dies in Kombination mit der Wahl einer günstigen *Vorbedingung* $\mathit{pre}(R)$, also einer Forderung an die Eingabe R, die hinreichend allgemein für das betrachtete Problem ist und außerdem sicherstellt, daß die Initialisierung die Schleifeninvariante etabliert. Da wir eine inklusionsminimale Teilmenge von R berechnen wollen, die $\mathcal{P}$ erfüllt, und zusätzlich $\mathcal{P}$ als sich nach oben vererbend angenommen haben, scheint es vernünftig zu sein,

$$\mathit{pre}(R) \;:\Longleftrightarrow\; \mathcal{P}(R)$$

als Vorbedingung zu wählen, d.h. zu fordern, daß die Eingabe R von *Min* ebenfalls $\mathcal{P}$ erfüllt. Dieser Ansatz funktioniert und liefert eine Initialisierung, die sowohl A als auch B mit R vorbesetzt. Hier ist die formale Rechtfertigung, daß diese die Schleifeninvariante etabliert:

$$\begin{array}{ll} & \mathit{pre}(R) \\ \Longleftrightarrow & \mathcal{P}(R) \wedge R \subseteq R \wedge \forall x \in \emptyset : \neg\mathcal{P}(R - x) \\ \Longleftrightarrow & \mathcal{P}(R) \wedge R \subseteq R \wedge R \subseteq R \wedge \forall x \in R \setminus R : \neg\mathcal{P}(R - x) \\ \Longleftrightarrow & \mathit{inv}(R, R, R) \,. \end{array}$$

Nachdem wir die Initialisierung entwickelt haben, bleibt als dritter Teil noch die Herleitung des Schleifenrumpfs, dessen Ausführung die Gültigkeit der Schleifeninvariante aufrechterhält und zur Terminierung führt. Dazu sei $B \neq \emptyset$ angenommen und b ein beliebig gewähltes Element von B (genauer: von seinem Wert). Wir unterscheiden zwei Fälle. Zuerst gelte $\mathcal{P}(A-b)$. Hier bekommen wir

$$
\begin{array}{ll}
 & inv(R,A,B) \\
\Longleftrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge B \subseteq A \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A-x) \\
\Longrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge B-b \subseteq A-b \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A-x) \\
\Longrightarrow & \mathcal{P}(A-b) \wedge A-b \subseteq R \wedge B-b \subseteq A-b \\
 & \qquad\qquad\qquad\qquad \wedge\ \forall x \in A \setminus B : \neg\mathcal{P}((A-b)-x) \\
\Longrightarrow & \mathcal{P}(A-b) \wedge A-b \subseteq R \wedge B-b \subseteq A-b \\
 & \qquad\qquad\qquad\qquad \wedge\ \forall x \in (A \setminus B)-b : \neg\mathcal{P}((A-b)-x) \\
\Longleftrightarrow & \mathcal{P}(A-b) \wedge A-b \subseteq R \wedge B-b \subseteq A-b \\
 & \qquad\qquad\qquad\qquad \wedge\ \forall x \in (A-b) \setminus (B-b) : \neg\mathcal{P}((A-b)-x) \\
\Longleftrightarrow & inv(R,A-b,B-b)\,.
\end{array}
$$

Diese Rechnung verwendet im dritten Schritt die Gültigkeit von $\mathcal{P}(A-b)$ und Eigenschaft (2). Es bleibt noch der Fall, daß $\mathcal{P}(A-b)$ nicht wahr ist. Hier rechnen wir wie nachfolgend angegeben, wobei die Gültigkeit von $\neg\mathcal{P}(A-b)$ im dritten Schritt verwendet wird:

$$
\begin{array}{ll}
 & inv(R,A,B) \\
\Longleftrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge B \subseteq A \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A-x) \\
\Longrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge B-b \subseteq A \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A-x) \\
\Longleftrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge B-b \subseteq A \wedge \forall x \in (A \setminus B)+b : \neg\mathcal{P}(A-x) \\
\Longleftrightarrow & \mathcal{P}(A) \wedge A \subseteq R \wedge B-b \subseteq A \wedge \forall x \in A \setminus (B-b) : \neg\mathcal{P}(A-x) \\
\Longleftrightarrow & inv(R,A,B-b)\,.
\end{array}
$$

Im Hinblick auf die eben hergeleiteten zwei Implikationen haben wir bei jedem Durchlauf der While-Schleife den Wert von B in $B-b$ abzuändern und den Wert von A in $A-b$; letzteres jedoch nur, falls $\mathcal{P}(A-b)$ gilt. Diese Veränderungen der Variablenwerte kann man offensichtlich mit Hilfe einer bedingten Anweisung erreichen. Wir erhalten dann das nachfolgend angegebene vollständige Programm *Min*, in dem wir zusätzlich annehmen, daß der Aufruf *elem*(B) der primitiven Operation *elem* ein Element von B liefert:

$$
\begin{array}{l}
\{\ pre(R)\ \} \\
A,B := R,R; \\
\{\ inv(R,A,B)\ \} \\
\textbf{while}\ B \neq \emptyset\ \textbf{do} \\
\quad b := elem(B); \\
\quad \textbf{if}\ \mathcal{P}(A-b)\ \textbf{then}\ A,B := A-b, B-b \\
\qquad\qquad\qquad\ \ \textbf{else}\ \ B := B-b\ \textbf{fi od}; \\
\{\ post(R,A)\ \}\,.
\end{array}
$$

Die Terminierung von Min ist offensichtlich. Seine Laufzeit wird hauptsächlich von zwei Faktoren bestimmt, nämlich dem Aufwand zur Auswertung von $\mathcal{P}$ und der Anzahl der Durchläufe der While-Schleife. In vielen konkreten Fällen kann der Aufwand zur Auswertung von $\mathcal{P}$ dadurch verringert werden, daß man ein Prädikat $\mathcal{Q}$ auf $2^M \times M$ sucht, das einerseits einfacher auszuwerten ist als $\mathcal{P}$ und andererseits für alle $X \in 2^M$ und $x \in X$ die Eigenschaft

$$\mathcal{P}(X - x) \iff \mathcal{P}(X) \wedge \mathcal{Q}(X, x) \tag{3}$$

erfüllt. Weil $\mathcal{P}(A)$ ein Teil der Schleifeninvariante von Min ist und Auswahl von b durch die Zuweisung $b := elem(B)$ deren Gültigkeit nicht verändert, verfeinert sich unter der Annahme (3) das bisherige Programm offensichtlich korrekterweise zu dem folgenden:

$$
\begin{array}{l}
\{\ pre(R)\ \}\\
A, B := R, R;\\
\{\ inv(R, A, B)\ \}\\
\textbf{while}\ B \neq \emptyset\ \textbf{do}\\
\quad b := elem(B);\\
\quad \textbf{if}\ \mathcal{Q}(A, b)\ \textbf{then}\ A, B := A - b, B - b\\
\qquad\qquad\qquad \textbf{else}\ \ B := B - b\ \textbf{fi od};\\
\{\ post(R, A)\ \}\,.
\end{array}
$$

Wir bezeichnen diese Verfeinerung von Min mit Min_1. Je effizienter die Auswertung von $\mathcal{Q}$ im Vergleich zu der von $\mathcal{P}$ möglich ist, um so effizienter ist natürlich auch das Programm Min_1 im Vergleich zu Min.

Die Anzahl der Schleifendurchläufe von Min und damit auch von Min_1 wird durch die Kardinalität der Eingabe R bestimmt. Im Hinblick auf eine weitere Effizienzverbesserung bietet sich deshalb an, die Initialisierung von A und B nicht mit R erfolgen zu lassen, sondern mit einer möglichst kleinen Teilmenge von R, die ebenfalls $\mathcal{P}$ erfüllt. Geschieht diese Vorberechnung durch ein Programmstück $\gg$Vorberechnung$\ll$, das in einer weiteren Variablen C die gesuchte Menge abspeichert, so führt, wegen der Äquivalenz von $\mathcal{P}(C) \wedge C \subseteq R$ und $inv(R, C, C)$, dieser Ansatz zur nachfolgenden korrekten Verfeinerung Min_2 von Min_1:

$$
\begin{array}{l}
\{\ pre(R)\ \}\\
\gg \text{Vorberechnung} \ll;\\
\{\ \mathcal{P}(C) \wedge C \subseteq R\ \}\\
A, B := C, C;\\
\{\ inv(R, A, B)\ \}\\
\textbf{while}\ B \neq \emptyset\ \textbf{do}\\
\quad b := elem(B);\\
\quad \textbf{if}\ \mathcal{Q}(A, b)\ \textbf{then}\ A, B := A - b, B - b\\
\qquad\qquad\qquad \textbf{else}\ \ B := B - b\ \textbf{fi od};\\
\{\ post(R, A)\ \}\,.
\end{array}
$$

Zusätzlich zu der oben gemachten Bemerkung beim Übergang von *Min* zu Min_1 gilt nun offensichtlich: Je einfacher die Vorberechnungsphase und je kleiner die Kardinalität ihres Resultats im Vergleich zu der von R ist, um so effizienter ist das Programm Min_2 im Vergleich zu Min_1.

Zum Schluß dieses Abschnitts sollte noch erwähnt werden, daß unser Vorgehen Ähnlichkeiten hat mit dem von J. Ravelo in [8] gewählten Ansatz. In [8] wird im Verfeinerungskalkül (siehe beispielsweise [7] für Einzelheiten) formal ein generisches Programm zur Berechnung inklusionsmaximaler Teilmengen hergeleitet und anschließend zur Lösung von zwei graphentheoretischen Problemen (spannende Wälder und inklusionsmaximale unabhängige Knotenmengen) instantiiert. Es benutzt in seiner Endversion aus Effizienzgründen ebenfalls die Idee des Fortschreibens eines Prädikats, was unserer Annahme (3) entspricht. Die Vorberechnungsphase ist hingegen nicht vorhanden. Ein weiterer Unterschied zu unserer Lösung ist die Verwendung von Mengenkomplementen in [8], insbesondere in der Abbruchbedingung einer While-Schleife, was bei den gängigen Mengenimplementierungen in der Regel zu ineffizienteren Instantiierungen führt.

3 Eine graphentheoretische Anwendung

Wir betrachten nun eine Anwendung der bisher entwickelten generischen Programme auf ein graphentheoretisches Problem. Dazu setzen wir voraus, daß der Leser mit den Grundlagen der Graphentheorie vertraut ist. Andernfalls verweisen wir auf entsprechende Literatur, beispielsweise das Lehrbuch [6].

Gegeben sei für den Rest des Abschnitts ein (ungerichteter) Graph $g = (V, E)$ mit endlicher Knotenmenge V und Kantenmenge E, wobei jede Kante eine zweielementige Teilmenge von V ist. Weiterhin nehmen wir für die zugrundeliegende Programmiersprache neben den schon im letzten Abschnitt verwendeten Mengenoperationen noch Implementierungen der beiden Operationen

$$neighb(x) = \{y \in V : \{x, y\} \in E\} \qquad incid(e) = \{f \in E : e \cap f \neq \emptyset\}$$

an, die zu einem Knoten x von g die Menge $neighb(x)$ seiner Nachbarknoten bzw. zu einer Kante e von g die Menge $incid(e)$ der mit ihr inzidierenden Kanten berechnen.

Eine Teilmenge C von V heißt eine *Knotenüberdeckung* von g, falls $C \cap e \neq \emptyset$ für alle $e \in E$ gilt, also jede Kante von g mit mindestens einem Knoten aus C inzidiert. (Man sagt dann auch, daß E von C *überdeckt* wird.) Offensichtlich ist das Prädikat $\mathcal{P}$ auf 2^V, festgelegt durch

$$\mathcal{P}(X) :\Longleftrightarrow \forall\, e \in E : X \cap e \neq \emptyset\,, \tag{4}$$

nach oben vererbend und es gilt $\mathcal{P}(V)$. Seine triviale operationelle $O(|E|)$-Realisierung führt damit durch die Instantiierung von *Min* mit der Ersetzung von sowohl M als auch R durch V sofort zu dem folgenden Programm, genannt *MinCover*, das in $O(|V| * |E|)$ elementaren Rechenoperationen eine inklusionsminimale Knotenüberdeckung von g berechnet:

$\{\ \mathit{True}\ \}$
$A, B := V, V;$
$\{\ \mathcal{P}(A) \wedge B \subseteq A \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A - x)\ \}$
while $B \neq \emptyset$ **do**
  $b := \mathit{elem}(B);$
  if $\mathcal{P}(A - b)$ **then** $A, B := A - b, B - b$
      else $B := B - b$ **fi od**;
$\{\ A$ inklusionsminimale Knotenüberdeckung von $g\ \}$.

Aus der speziellen Instantiierung des Prädikats $\mathcal{P}$ mittels (4) folgt aber unmittelbar auch für alle $X \in 2^V$ und $x \in X$ die Äquivalenz

$$\mathcal{P}(X - x) \iff \mathcal{P}(X) \wedge \mathit{neighb}(x) \subseteq X\,. \tag{5}$$

Wählen wir nun den rechten Teil der Konjunktion von (5) als $\mathcal{Q}(X, x)$, so gilt (3) und wir erhalten dadurch die nachfolgende Instantiierung von Min_1 zur effizienteren Berechnung einer inklusionsminimalen Knotenüberdeckung von g:

$\{\ \mathit{True}\ \}$
$A, B := V, V;$
$\{\ \mathcal{P}(A) \wedge B \subseteq A \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A - x)\ \}$
while $B \neq \emptyset$ **do**
  $b := \mathit{elem}(B);$
  if $\mathit{neighb}(b) \subseteq A$ **then** $A, B := A - b, B - b$
      else $B := B - b$ **fi od**;
$\{\ A$ inklusionsminimale Knotenüberdeckung von $g\ \}$.

Wir bezeichnen dieses Programm mit $\mathit{MinCover}_1$. Stellt man g durch Adjazenzlisten dar (siehe etwa [3]), so führt seine Abarbeitung zu $O(\max(|V|, |E|))$ elementaren Rechenoperationen, denn die gewählte Graphdarstellung erlaubt, alle dabei auftretenden Inklusionen $\mathit{neighb}(b) \subseteq A$ mittels insgesamt $O(|E|)$ Tests auf das Enthaltensein in einer Menge auszuwerten.

Das Problem, eine anzahlminimale Knotenüberdeckung eines beliebigen Graphen zu berechnen, ist NP-vollständig. Einen Beweis findet man etwa in [3]. In diesem Buch wird auch ein auf F. Gavril und M. Yannakakis zurückgehender Approximationsalgorithmus formuliert, der in unserer Programmnotation zur Eingabe E wie folgt aussieht und $O(|E|)$ elementare Rechenoperationen benötigt:

$\{\ \mathit{True}\ \}$
$F, C := E, \emptyset;$
$\{\ C$ Knotenüberdeckung von $g_F = (V, E \setminus F)\ \}$
while $F \neq \emptyset$ **do**
  $e := \mathit{elem}(F);$
  $F, C := F \setminus \mathit{incid}(e), C \cup e$ **od**;
$\{\ C$ Knotenüberdeckung von $g \wedge |C| \leq 2 * c_{opt}\ \}$.

Bezeichnet c_{opt} die Kardinalität einer anzahlminimalen Knotenüberdeckung von g, so berechnet dieses Programm – wir nennen es im folgenden *Gavril* – eine Knotenüberdeckung C von g mit $|C| \leq 2 * c_{opt}$. Man nennt den Faktor 2 auch die *Güteratio* der Approximation. Die Eigenschaft, daß C eine Knotenüberdeckung von g ist, verifiziert man sofort mit Hilfe der angegebenen Schleifeninvariante, indem man nachweist, daß diese durch die Initialisierung und den Schleifenrumpf von *Gavril* etabliert bzw. aufrechterhalten wird. Zum Beweis von $|C| \leq 2 * c_{opt}$ ergänzt man die Schleifeninvariante von *Gavril* durch Konjunktion von

$$\mathit{inv}'(F,C) \; :\Longleftrightarrow \; \begin{cases} \exists\, M \in 2^E \; : M \text{ Matching von } g \wedge \\ \qquad\qquad\quad |C| \leq 2 * |M| \wedge \\ \qquad\qquad\quad \forall\, f \in F, m \in M : f \cap m = \emptyset \end{cases}$$

und zeigt, daß $\mathit{inv}'(F,C)$ durch die Initialisierung etabliert und durch den Schleifenrumpf aufrechterhalten wird. (Anschaulich werden in M genau die durch die Zuweisung $e := \mathit{elem}(F)$ ausgewählten Kanten aufgesammelt; der dritte Teil des Rumpfs von $\mathit{inv}'(F,C)$ ist notwendig, um die Matching-Eigenschaft von $M + e$ mittels der Matching-Eigenschaft von M und $e \in F$ zu beweisen.) Nach Beendigung der Schleife von *Gavril* ist also C eine Knotenüberdeckung von g und es gibt ein Matching M von g mit $|C| \leq 2*|M|$. Die Abschätzung $|C| \leq 2*c_{opt}$ folgt nun sofort aus $|M| \leq c_{opt}$, denn jede Knotenüberdeckung – also insbesondere auch jede anzahlminimale – muß das Matching M überdecken.

Vergleicht man die beiden zuletzt vorgestellten Programme $\mathit{MinCover}_1$ und *Gavril* im Hinblick auf die Approximation einer anzahlminimalen Knotenüberdeckung von g, so besitzen beide Vor- und Nachteile. Der große Vorteil von *Gavril* ist, daß die Größe seines Resultats bewiesenermaßen durch $2 * c_{opt}$ beschränkt ist; bei $\mathit{MinCover}_1$ kann die Approximation hingegen, wie man sich etwa sofort für „sternförmige“ Graphen klarmacht, sehr schlecht werden. Ein gewisser Nachteil von *Gavril* ist hingegen, daß das berechnete Resultat in der Regel unnötige Knoten enthält, also nicht inklusionsminimal ist. Dies wurde im Rahmen von mehr als 1000 Tests mit zufällig erzeugten Graphen bestätigt, die alle mittels relationaler Versionen der beiden Programme und dem Computersystem RelView (siehe etwa [1, 2]) durchgeführt wurden. Beispielsweise lieferte die relationale Version von *Gavril* bei einer Testreihe von 180 Zufallsgraphen $g_{ij} = (V, E_{ij})$, $1 \leq i \leq 9, 1 \leq j \leq 20$, mit $|V| = 100$ und $|E_{ij}| = i * 1000$ für alle j nur ein einziges Mal eine Knotenüberdeckung ohne unnötige Knoten.

Die mit RelView durchgeführten Tests zeigten auch, daß $\mathit{MinCover}_1$ in der Regel Knotenüberdeckungen berechnet, die weniger Knoten haben als die von *Gavril* berechneten. Zur Illustration sind in den nachfolgenden drei Tabellen die Anfänge von drei RelView-Testreihen angegeben. Die linke Tabelle zeigt dabei die Kardinalitäten von den berechneten Knotenüberdeckungen, wobei die zufällig erzeugten Graphen genau 250 Knoten und 5 % aller möglichen Kanten besitzen. In der mittleren und der rechten Tabelle sind ebenfalls die Kardinalitäten von den berechneten Knotenüberdeckungen angegeben; hier haben alle zufällig erzeugten Graphen genau 100 Knoten und 10 % aller möglichen Kanten bzw. genau 500 Knoten und 20 % aller möglichen Kanten.

$MinCover_1$	$Gavril$
218	242
218	244
217	244
218	242
217	244
216	240
218	242
217	244
218	242
217	244
$\vdots$	$\vdots$

$MinCover_1$	$Gavril$
83	96
83	98
86	96
86	96
85	94
83	96
84	96
84	98
83	98
86	94
$\vdots$	$\vdots$

$MinCover_1$	$Gavril$
487	498
487	498
487	498
490	496
488	498
489	498
488	496
488	496
488	496
485	498
$\vdots$	$\vdots$

Die Verbesserung von $MinCover_1$ gegenüber *Gavril* beträgt im linken Fall etwa 11 %, im mittleren Fall etwa 12 % und im rechten Fall etwa 2 bis 3 %.

Aufbauend auf diese Tests scheint es nun vernünftig zu sein, beide Programme zu kombinieren, also eine Instantiierung von Min_2 mit *Gavril* als im Vergleich zu $MinCover_1$ zusätzlichen Vorberechnungsphase zu verwenden. Dies führt unmittelbar zum nachfolgend angegebenen Programm $MinCover_2$:

$\{\ \mathit{True}\ \}$
$F, C := E, \emptyset;$
$\{\ C$ Knotenüberdeckung von $g_F = (V, E \setminus F) \wedge \mathit{inv}'(F, C)\ \}$
while $F \neq \emptyset$ **do**
  $e := \mathit{elem}(F);$
  $F, C := F \setminus \mathit{incid}(e), C \cup e$ **od**;
$\{\ C$ Knotenüberdeckung von $g \wedge |C| \leq 2 * c_{opt}\ \}$
$A, B := C, C;$
$\{\ \mathcal{P}(A) \wedge B \subseteq A \wedge \forall x \in A \setminus B : \neg\mathcal{P}(A - x) \wedge |A| \leq 2 * c_{opt}\ \}$
while $B \neq \emptyset$ **do**
  $b := \mathit{elem}(B);$
  if $\mathit{neighb}(b) \subseteq A$ **then** $A, B := A - b, B - b$
    else $B := B - b$ **fi od**;
$\{\ A$ inklusionsminimale Knotenüberdeckung von $g \wedge |A| \leq 2 * c_{opt}\ \}$.

Ein Vorteil des Programms $MinCover_2$ im Hinblick auf die Approximation einer anzahlminimalen Knotenüberdeckung von g ist, daß die bezüglich der Kantenanzahl lineare Laufzeitschranke und die Güteratio 2 wie beim Programm *Gavril* beibehalten werden. Zusätzlich vermeidet es unnötige Knoten im Resultat. Konkret heißt dies, daß es für Graphen mit bis zu etwa 50 % aller möglichen Kanten vorherrschend Knotenüberdeckungen mit noch weniger Knoten als die erste Verfeinerung $MinCover_1$ von *MinCover* liefert. Die in der Praxis vorkommenden Graphen enthalten üblicherweise viel weniger als 50 % aller Kanten. Bei „dicken“ Graphen ergibt sich hingegen normalerweise keine Verbesserung mehr, da hier, wie schon die rechte der obigen Tabellen andeutet, $MinCover_1$ und *Gavril* eigentlich immer gleichgroße Resultate berechnen. Viele mit dem RelView-System durchgeführte Tests mit Zufallsgraphen haben diese Aussagen bestätigt.

4 Abschließende Bemerkungen

In dem vorliegenden Artikel haben wir zuerst formal ein generisches Programm zur Berechnung von inklusionsminimalen Teilmengen entwickelt, es dann zweimal verfeinert um die Effizienz zu steigern und schließlich eine Anwendung bei der Approximation anzahlminimaler Knotenüberdeckungen von Graphen demonstriert. Letzteres beinhaltete die Kombination mit einem aus der Literatur bekannten Approximationsverfahren. Diese wurde motiviert durch dessen Vergleich mit einer Zwischenversion unseres Endprogramms anhand der zahlreichen mittels des Computersystems RelView erzielten Daten und führte zu einer Verbesserung der Approximation.

Derzeit laufende und geplante Arbeiten betreffen weitere Anwendungen der generischen Programme von Abschnitt 2 und Variationen davon. Auch die in Abschnitt 3 erwähnten relationalen Versionen sind Ursprung für weitere Untersuchungen. Ihre bisherigen formalen Verifikationen verwenden nämlich neben relationenalgebraischen Beweisschritten auch noch komponentenbehaftete, falls Kardinalitäten von Relationen ins Spiel kommen. Aus methodischen Gründen ist in den letzten Fällen ebenfalls eine Algebraisierung wünschenswert; eine entsprechende Erweiterung der Relationenalgebra um eine Kardinalitätsoperation und geeignete Axiome ist aber noch zu entwickeln.

Schließlich möchte ich mich noch bei Markus Müller-Olm für wertvolle Hinweise bei der Erstellung dieses Artikels bedanken.

Literatur

1. Behnke R. et al.: Applications of the RelView system. In: Berghammer R., Lakhnech Y. (eds.): Tool support for system specification, development and verification. Advances in Computing, Springer, 33-47 (1999)
2. Berghammer R., von Karger B., Ulke C.: Relation-algebraic analysis of Petri nets with RelView. In: Margaria T., Steffen B. (eds.): Proc. 2nd Workshop on Tools and Applications for the Construction and Analysis of Systems, LNCS 1055, Springer, 49-69 (1996)
3. Cormen T.H., Leiserson C.E., Rivest R.L.: Introduction to algorithms. The MIT Press (1990)
4. Dijkstra E.W.: A discipline of programming. Prentice-Hall (1976)
5. Gries D.: The science of computer programming. Springer (1981)
6. Jungnickel D.: Graphen, Netzwerke und Algorithmen. BI Wissenschaftsverlag, 3. Auflage (1994)
7. Morgan C, Vickers T. (eds.): On the refinement calculus. Formal Approaches to Computing and Information Technology, Springer (1992)
8. Ravelo J.: Calculating with relations for graph algorithmics. In: Jaoua A. (ed.): Proc. 3rd Int. Seminar on Relational Methods in Computer Science, Universität Tunis II, 291-300 (1997)