Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik I

Florian Amrhein

Spannbäume mit kleinem Schnitt

8. Juni 2007

Seminararbeit im SS 2007

Zusammenfassung

In dieser Arbeit betrachten wir Spannbäume in einem Graphen, welche möglichst wenige Kanten jedes Schnitts aus einer Menge von in dem Graphen gegebenen Schnitten enthalten. Wir einen geeigneten Algorithmus, der solche Spannbäume erstellt, betrachten, und beweisen, dass dieser einen Baum erzeugt, bei dem gegenüber einer optimalen Lösung maximal $O(r \log n)$ so viele Kanten einen Schnitt durchkreuzen. r ist hier die Anzahl der Schnitte, in denen jede beliebige Kante maximal enthalten sein kann. Zudem werden wir beweisen, dass das Problem selbst dann NP-vollständig ist, wenn man vollständige Graphen betrachtet.

Inhaltsverzeichnis

1 Grundlagen

1.1 Spannbäume

Definition 1 Ein Spannbaum zu einem ungerichteten Graphen $G = (V, E)$ ist ein Baum, der alle Knoten $V(G)$ sowie eine Teilmenge der Kanten $E(G)$ enthält.

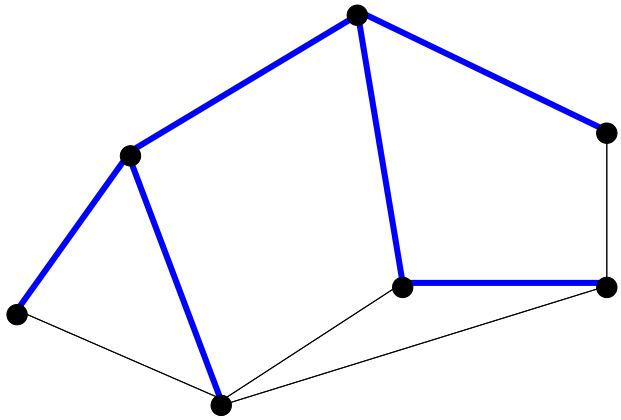

Abbildung 1: Ein Spannbaum.

Definition 2 Ein Spannbaum mit minimalem Grad minimiert den maximalen Grad der enthaltenen Knoten.

1.2 Schnitte

Definition 3 Ein Schnitt bezeichnet eine Menge von Kanten eines Graphen $G = (V, E)$, welche zwischen einer Teilmenge $W \subset V(G)$ und der Restmenge $V(G) \backslash W$ verlaufen.

2 Spannbäume mit kleinem Schnitt

Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit $|V(G)| = n$ sowie eine darin enthaltene Menge von m Schnitten $\mathcal{C} = \{C_1, \ldots, C_m\}$.

Wir suchen nun einen Spannbaum zum Graphen G, also einen Baum, der alle Knoten aus G enthält, wollen dabei jedoch ein besonderes Augenmerk auf die Schnitte legen:

Definition 4 Das Crossing eines Schnitts C in einem Graphen G ist definiert als $Cross(C, G) := |E(G) \bigcap C|$, gibt also die Anzahl der Kanten aus G an, die der Schnitt C enthält.

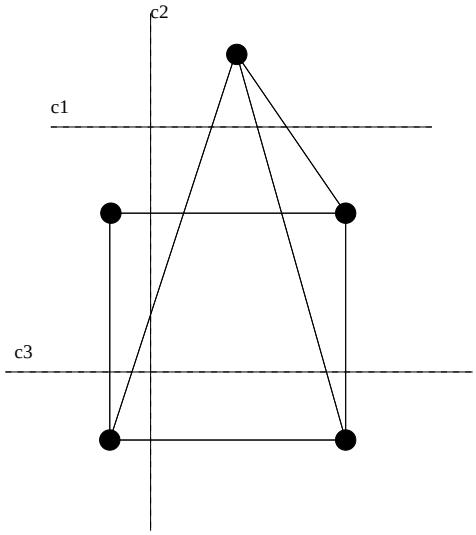

Abbildung 2: Schnitte in einem Graphen.

Definition 5 Das Crossing einer Menge von Schnitten $\mathcal{C}$ ist definiert als das maximale Crossing der einzelnen Schnitte $C \in \mathcal{C}$:

$$Cross(\mathcal{C}, G) := max_{C \in \mathcal{C}}\{Cross(C, G)\}$$

Definition 6 Ein *MCSP* (minimum crossing spanning tree) ist ein Spannbaum T, welcher das maximale Crossing aller Schnitte minimiert.

Gesucht ist also ein Spannbaum T, bei dem jeder Schnitt aus $\mathcal{C}$ möglichst wenige Kanten aus $E(T)$ enthalten soll.

3 Komplexität

Sei $\mathcal{C}$ eine Menge von Schnitten:

$$\mathcal{C} = \{(v, V \backslash \{v\}) : v \in V\}$$

Dadurch, dass jedem Knoten ein Schnitt zugeordnet wird, entspricht das Crossing jedes dieser Schnitte genau dem Knotengrad des dem Schnitt zugeordneten Knotens. Das Problem des Spannbaums mit minimalem Grad läßt sich also auf das MCST-Problem reduzieren. Da das Problem des Spannbaums mit minimalem Grad NP schwer ist [2], ist auch das MCST-Problem NP schwer.

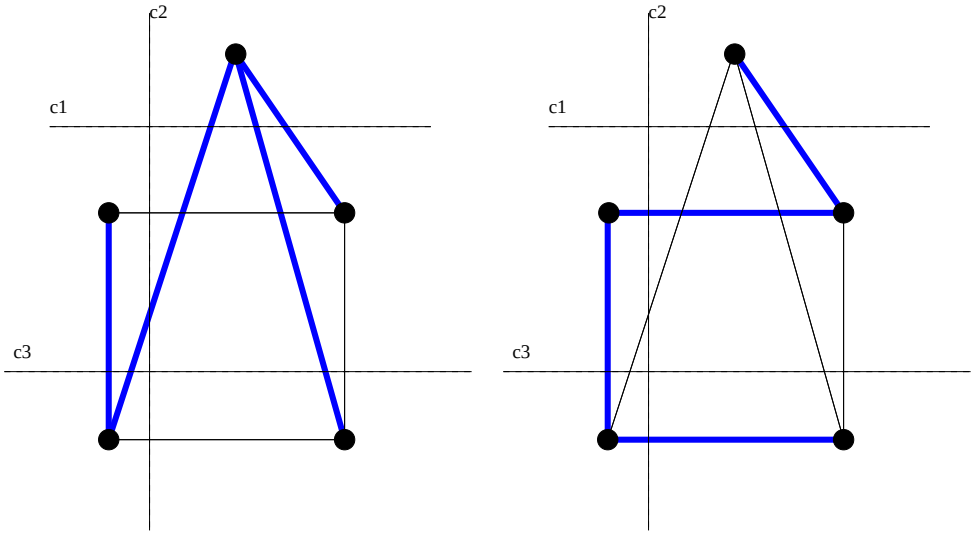

Abbildung 3: Spannbäume mit maximalem Crossing von 3 und 2.

Im Falle eines vollständigen Graphen ist das Problem des Spannbaums mit minimalem Grad trivial, was jedoch nicht für das MCST-Problem gilt. Dies werden wir in einem späteren Kapitel beweisen.

4 Greedy-Algorithmus

Der hier vorgestellte Greedy-Algorithmus liefert im Allgemeinen keine optimale Lösung. Er erzeugt einen Spannbaum, der einen maximalen Schnitt von $O(r \log n \cdot OPT)$ hat, wobei $r := \max_{e \in E(G)} |\{C \in \mathcal{C} : e \in C\}|$ ist, also die maximale Anzahl an Schnitten aus $\mathcal{C}$, in denen jede beliebige Kante aus G enthalten ist.

Der Greedy-Algorithmus startet mit einem leeren Baum und einer Menge von Zusammenhangskomponenten, wobei jeder Knoten aus G zunächst eine eigene Komponente bildet. Diese Komponenten werden nach und nach miteinander verknüpft. Nach Möglichkeit werden als erstes Komponenten mit Kanten verknüpft, ohne dass sich das maximale Crossing erhöht, und nur falls keine Kanten gefunden werden, die diese Eigenschaft erhalten, werden auch andere Kanten eingefügt.

Lemma 7 *Sei $\mathcal{S} \subset \mathcal{C}$ eine beliebige Teilmenge von $\mathcal{C}$, und sei k die Anzahl der Zusammenhangskomponenten, in die G zerfällt, nachdem alle Kanten aus $E(G)$, welche in den Schnitten aus $\mathcal{S}$ enthalten sind, entfernt wurden.*

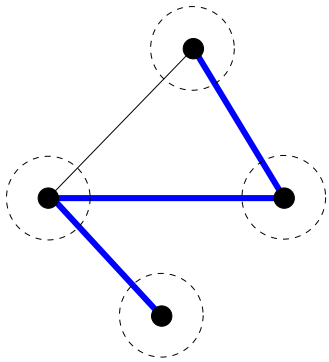

Abbildung 4: Komplexität: Jedem Knoten wird ein eigener Schnitt zugeordnet, um so das Problem des Spannbaums mit minimalem Grad auf das MCST-Problem zu reduzieren.

Algorithmus 4.1 Greedy-Algorithmus

$F := \emptyset$
while (V, F) ist kein Baum **do**
  Sei $e' \in E(G)$ eine Kante, die zwei bislang nicht zusammenhängende Komponenten verbindet, und $Cross\,(Fe, \mathcal{C})$ minimiert.
  $F \leftarrow F \cup e'$
end while

Dann gilt:

$$opt \geq \frac{k-1}{|\mathcal{S}|}$$

Beweis. Jeder Spannbaum für G benötigt mindestens $k-1$ Kanten, um diese k Komponenten wieder miteinander zu verbinden. Jede dieser $k-1$ Kanten ist in mindestens einem Schnitt aus $\mathcal{S}$ enthalten. Also gilt, dass das durchschnittliche Crossing eines solchen Schnitts aus $\mathcal{S}$ $\frac{k-1}{|\mathcal{S}|}$ beträgt. □

Theorem 8 *Der Greedy-Algorithmus liefert einen Spannbaum, welcher ein maximales Crossing von $O(r \log n \cdot OPT)$ aufweist, bei dem also das Crossing maximal $O(r \log n)$ so hoch ist wie bei einer optimalen Lösung.*

Beweis. Sei T_g der Spannbaum, welcher vom Greedy-Algorithmus erzeugt wird, und sei $l = Cross\,(T_g, \mathcal{C})$.

Wir fassen die einzelnen Schritte des Algorithmus nun zu l Phasen zusammen, wobei die i-te Phase des Algorithmus diejenige ist, in der gilt: $Cross(F, \mathcal{C}) = i$. k_i sei nun die Anzahl an Zusammenhangskomponenten in

F am Ende der i-ten Phase, und seien $\mathcal{M}_i \subset \mathcal{C}$ diejenigen Schnitte, welche von den i Kanten am Ende der i-ten Phase durchkreuzt werden. Sei außerdem $m_i := |\mathcal{M}_i|$.

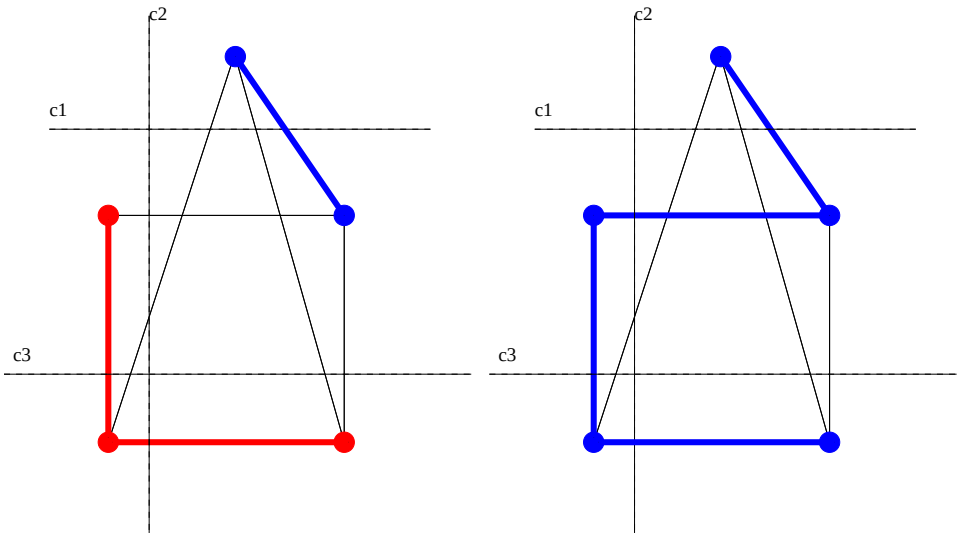

Abbildung 5: Zusammenhangskomponenten (blau und rot) am Ende der ersten und zweiten Phase. Nach der ersten Phase gilt $k_1 = 2, m_1 = 3$ und nach der zweiten $k_2 = 1, m_2 = 1$. Zudem ist $r = 3$, da jede Kante hier nur in maximal drei Schnitten enthalten ist (die die Kante von unten links nach oben). Die meisten Kanten sind jedoch nur in einem Schnitt enthalten.

Betrachten wir nun die i-te Phase des Algorithmus: Hier werden genau $k_{i-1} - k_i$ Kanten eingefügt, um nun k_i Komponenten zu erhalten. Jede neue Kante kann nur von maximal r Schnitten das jeweilige Crossing erhöhen, da laut Definition eine Kante nur in maximal r Schnitten enthalten ist. In jeder Phase müssen also $\frac{m_i}{r}$ Kanten eingefügt werden, wobei jede dieser Kanten die Zahl der Zusammenhangskomponenten um genau eins verringert. Wir erhalten somit:

$$k_{i-1} - k_i \geq \frac{m_i}{r} \tag{1}$$

Am Ende der i-ten Phase ist jede Kante zwischen den verbleibenden Komponenten in mindestens einem der Schnitte $\mathcal{M}_i$ enthalten. Mit Lemma 7 gilt nun:

$$\begin{aligned} opt &\geq \frac{k_i - 1}{m_i} \\ \Leftrightarrow m_i &\geq \frac{k_i - 1}{opt} \end{aligned} \tag{2}$$

$$\Leftrightarrow \frac{m_i}{r} \geq \frac{k_i - 1}{r \cdot opt}$$

mit (1) und (2) erhalten wir nun:

$$k_{i-1} - k_i \geq \frac{k_i - 1}{r \cdot opt} \tag{3}$$

Aus $k_i \geq 2$ für alle $i \leq l-1$ und $k_{l-1} > k_l$ folgt:

$$\begin{aligned} k_{i-1} - k_i &\geq \frac{k_i}{2r \cdot opt} \qquad (4)\\ \Leftrightarrow k_{i-1} &\geq k_i + \frac{k_i}{2r \cdot opt} \\ \Leftrightarrow k_{i-1} &\geq k_i \left(1 + \frac{1}{2r \cdot opt}\right) \\ \Rightarrow k_0 &\geq k_l \left(1 + \frac{1}{2r \cdot opt}\right)^l \end{aligned}$$

Da $k_0 = n$ und $k_l = 1$:

$$\begin{aligned} n &\geq \left(1 + \frac{1}{2r \cdot opt}\right)^l \qquad (5)\\ \Rightarrow \log n &\geq l \log \left(1 + \frac{1}{2r \cdot opt}\right) \end{aligned}$$

Wegen $\log(1+x) \geq x - \frac{x^2}{2}$ für $0 < x < 1$ und $r \cdot opt \geq 1$:

$$\begin{aligned} \log n &\geq l \left(\frac{1}{2r \cdot opt} - \frac{\left(\frac{1}{2r \cdot opt}\right)^2}{2} \right) \qquad (6)\\ &= l \left(\frac{1}{2r \cdot opt} - \frac{1}{8r^2 \cdot opt^2} \right) \\ &= l \left(\frac{1}{2r \cdot opt} \cdot \underbrace{\left(1 - \underbrace{\left(\frac{1}{4r \cdot opt} \right)}_{\leq \frac{1}{4}} \right)}_{\geq \frac{3}{4}} \right) \\ &\geq 3l \frac{1}{8r \cdot opt} \\ \Rightarrow l &\leq \frac{8}{3} r \log n \cdot opt \end{aligned}$$

Jeder vom Greedy-Algorithmus erzeugte Baum hat demnach ein maximales Crossing, welches höchstens $O(r \log n)$ mal so groß wie das einer optimalen Lösung ist.

□

5 MCST ist für vollständige Graphen NP schwer

In diesem Kapitel zeigen wir nun, dass das MCST-Problem sogar für vollständige Graphen in NP schwer ist. Hierzu werden wir 2-C1P auf MCST reduzieren.

5.1 2-C1P

Definition 9

Gegeben sei eine 0-1-Matrix A. Das „2 Aufeinanderfolgende Einsen Problem“ (2-C1P) sucht eine Permutation π der Zeilen der Matrix, welche die Eigenschaft hat, dass sich in jeder Spalte der resultierenden Matrix jeweils maximal zwei zusammenhängende Blöcke mit 1-Einträgen befinden.

r1	1	1	1	1	0
r2	0	1	1	1	0
r3	1	0	1	1	1
r4	0	1	0	0	1
r5	1	0	1	0	0

r1	1	1	1	1	0
r3	1	0	1	1	1
r2	0	1	1	1	0
r4	0	1	0	0	1
r5	1	0	1	0	0

Abbildung 6: Links eine Matrix, rechts eine Permutation (zweite und dritte Zeile vertauschen), die die 2-C1P-Eigenschaft erfüllt.

5.2 Reduktion von 2-C1P auf MCST

Das Problem zu entscheiden, ob eine Permutation existiert, die die 2-C1P-Eigenschaft erfüllt, ist laut [4] NP schwer.

Sei A eine $n \times m$ Matrix. Ein dazu gehöriger vollständiger Graph $G = (V, E)$ wird folgendermaßen konstruiert: Zu jeder zeile r_i wird ein Knoten $v_i \in V(G)$ erzeugt, sowie zusätzlich ein Knoten s.

Nun wird eine Menge an Schnitten $\mathcal{C}$ angelegt, in dem zu jeder Spalte der Matrix ein Schnitt $C_j \in \mathcal{C}$ erzeugt wird. Der Schnitt separiert dabei die Knoten des Graphen, deren Eintrag in der Matrix 0 ist, mit den Knoten

r1	1	1	1	1	0
r2	1	0	1	1	1
r3	0	1	1	1	0
r4	0	1	0	0	1
r5	1	0	1	0	0

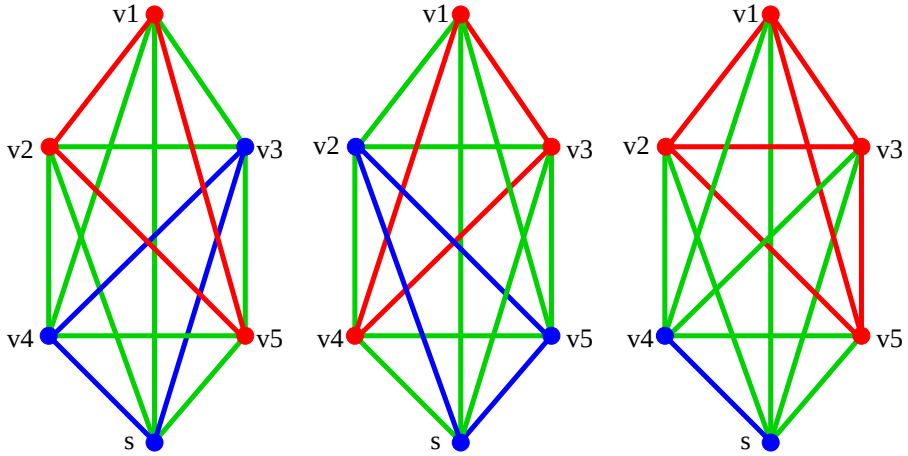

Abbildung 7: Beispiel für eine Matrix, der daraus resultierende Graph mit dem zusätzlichen Knoten s, sowie den Schnitten, die aus den ersten drei Spalten resultieren: In rot die 1-Knoten sowie die Kanten zwischen diesen, in blau die 0-Knoten und deren Kanten und in grün die im Schnitt enthaltenen Kanten.

deren Eintrag 1 ist. Diese Schnitte nennen wir *Spalten-Schnitte*.

Zusätzlich zu den Spalten-Schnitten werden einelementige sogenannte *Knoten-Schnitte* in $\mathcal{C}$ eingefügt: Diese enthalten jeweils einen Knoten aus $V(G)$, welcher damit von den verbleibenden Knoten getrennt wird: $C_v = (\{v\}, V \backslash \{v\})$. Außerdem wird für jedes Knotenpaar u und v aus G ein weiterer Schnitt in $\mathcal{C}$ zugefügt, der diese beiden Knoten von den verbleibenden Knoten separiert: $C_{uv} = (\{u, v\}, V \backslash \{u, v\})$.

5.2.1 Von 2-C1P zu MCST

Als erstes zeigen wir, dass falls eine Permutation π für die Zeilen einer Matrix A existiert, welche die C1P-Eigenschaft aufweist, es auch in dem aus $\pi(A)$ resultierenden Graphen einen Spannbaum mit einem maximalen Crossing von vier gibt.

Sei H ein Hamiltonweg im Graphen G, der im Knoten s startet, und danach jeden Knoten in der durch π vorgegebenen Reihenfolge besucht. Ein Spalten-Schnitt wird von H dabei genau dann durchkreuzt, wenn in $\pi(A)$ ein eintrag von 0 nach 1 oder von 1 nach 0 wechselt. Da die Einsen, gemäß der 2-C1P-Eigenschaft in genau zwei Blöcken angeordnet sind, kann es nur maximal vier solche Wechsel geben. Dass dabei in s, statt in einem der regulären Knoten, gestartet wird, ändert dabei nichts, da man diesen Knoten als neue erste Zeile in $\pi(A)$ betrachten kann, welche vollständig mit Nullen belegt ist. Daraus folgt, dass H jeden der Spalten-Schnitte maximal vier mal durchkreuzt. Es ist ferner klar, dass H die einelmentigen Knoten-Schnitte maximal zwei mal durchkreuzt, sowie die zweielementigen Knoten-Schnitte maximal vier mal. Da H ein Baum ist, existiert also zu der Permutation π ein MCST, der jeden Schnitt aus $\mathcal{C}$ maximal vier mal durchkreuzt.

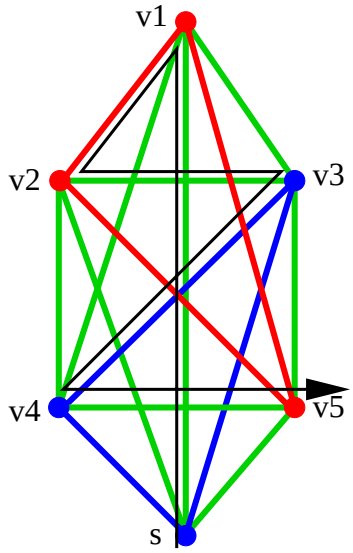

Abbildung 8: Ein Hamiltonweg, der startend mit s der Reihe nach alle Knoten besucht. Dabei wird ein Schnitt, der die blauen Knoten von den roten Knoten trennt, drei mal gekreuzt.

5.2.2 Von MCST zu 2-C1P

Nun ist noch zu zeigen, dass es zu jedem MCST T mit maximalem Crossing von vier eine Permutation π gibt, welche die 2-C1P-Eigenschaft erfüllt.

Lemma 10 *Ein MCST T zum Graphen G und Schnitten C mit maximalem Crossing von vier ist ein Hamiltonweg*

Beweis.

1. Da jedem Knoten aus $V(T)$ ein Knoten-Schnitt zugeordnet ist, kann kein Knoten einen Grad größer als vier haben. Andernfalls würde T mehr als vier Schnitte durchkreuzen, was die Voraussetzung verletzt.

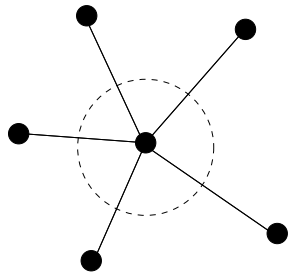

Abbildung 9: Baum, in dem ein Knoten den Grad 5 hat, und der diesem Knoten zugeordnete Knotenschnitt.

2. Angenommen es gibt einen Knoten $u \in V(T)$ mit Grad vier: Ist $|V(T)| > 5$, dann existiert ein Knoten $v \in V(T)$ welcher nicht zu u adjazent wäre. In diesem Fall würde jedoch der Schnitt C_{uv} mindestens fünf mal durchkreuzt werden. Der Knotengrad kann also drei nicht übersteigen.

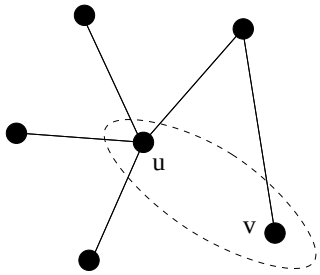

Abbildung 10: Knoten u hat Grad 4, Schnitt C_{uv} wird fünf mal durchkreuzt.

3. Sei $u \in V(T)$ ein Knoten mit Grad drei, hat also genau drei adjazente Knoten $V_a \subset V(T)$, alle anderen Knoten $V_n \subset V(T)$ sind zu u damit nicht adjazent. Da T jedes für jedes $v \in V_n$ den Schnitt C_{uv} maximal vier mal durchkreuzen darf, muss der Grad jedes dieser v eins sein. Damit können höchstens noch die drei zu u adjazenten Knoten V_a einen Grad von zwei oder drei haben. Die Gesamtsummer der Grade in T beträgt damit maximal $3 + 3 \cdot 3 + (|V(T)| - 4) = |V(G)| + 8$. Die Summe der Grade in jedem Baum beträgt jedoch genau $2n-2$, so dass gelten müsste: $2 \cdot |V(T)| - 2 \leq |V(T)| + 8$, welches für $|V(T)| \geq 11$ jedoch falsch ist. Der Knotengrad des Baums T beträgt damit maximal zwei, zumindestens für $V(T) \geq 11$.

□

Theorem 11 *Sei T ein MCST zum Graphen G und den Schnitten $\mathcal{C}$ mit maximalem Crossing von vier. Sei der Hamiltonweg*

$$H = (v_1, ..., v_k, s, v_{k+1}, ..., v_n)$$

in T, wobei die Knoten v_i den Zeilen r_i einer Matrix A zugeordnet sind. Sei ferner die Permutation $\pi(A) = (r_{k+1}, ..., r_n, r_1, ..., r_k)$. Dann gilt für A die 2-C1P-Eigenschaft.

Beweis. Nehmen wir an, es gäbe eine Spalte c_1 welche drei Einerblöcke enthält. Ein Hamiltonkreis, welcher sich aus dem Hamiltonweg ergibt, wenn man v_1 und v_n miteinander verbindet, würde dann den zu c_1 gehörenden Schnitt mindestens fünf mal durchkreuzen. Da in jedem Kreis jeder Schnitt eine gerade Anzahl oft geschnitten wird, würde der Schnitt sogar sechs mal durchkreuzt. Wenn man diesen Kreis auseinanderschneidet, um wieder den Hamiltonweg zu erhalten, dann kann eine dieser Durchkreuzungen wegfallen, so dass der Schnitt mindestens fünf mal durchkreuzt wird. Dies steht jedoch im Widerspruch zur Voraussetzung, dass das Crossing von T maximal vier beträgt. □

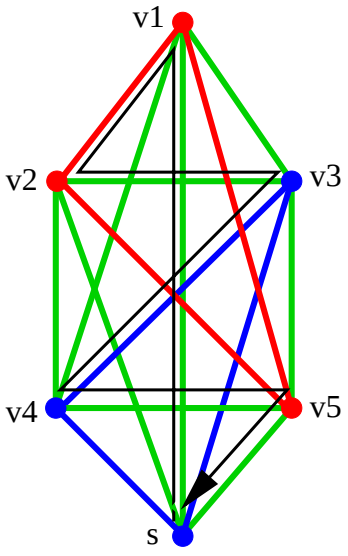

Abbildung 11: Ein Hamiltonkreis der, im Gegensatz zu einem Hamiltonweg, die grünen Schnittkanten vier mal durchkreuzt.

6 Fazit

Es gibt einen Algorithmus für das MCST-Problem, welcher Spannbäume erzeugt, deren Crossing maximal $O(r \log n \cdot OPT)$ beträgt, wobei r die maximale Anzahl an Schnitten ist, die eine Kante durchkreuzen kann. Zudem wurde bewiesen, dass das Problem nicht nur im allgemeinen Fall NP schwer ist, sondern auch in dem Spezialfall, dass der zugrunde liegende Graph vollständig ist.

Literatur

[1] V. Bilo, V. Goyal, R. Ravi, and M. Singh. on the crossing spanning tree problem. *LECTURE NOTES IN COMPUTER SCIENCE*, ISSU 3122:51–60, 2004.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, 1979.

[3] V. Goyal, V. Bilo, and M. Singh. On the crossing spanning tree.

[4] D. S. Greenberg and S. Istrail. Physical mapping by sts hybridization: Algorithmic strategies and the challenges of software evaluation. *Journal of Computational Biology*, 2(2):219–273, 1995.