Evolutionäre Algorithmen für Minimale Spannbäume

Martin Bullinger
Technische Universität München

Zusammenfassung

Klassische Greedy-Algorithmen zum Auffinden Minimaler Spannbäume (MST) sind schon seit der Mitte des zwanzigsten Jahrhunderts bekannt. Neuere Erkenntnisse liefern alternative Herangehensweisen an dieses Problem. Dazu gehören *Randomisierte Lokale Suche* (RLS) und *Evolutionäre Algorithmen* (EA). Ziel des Artikels ist es, auf diesen beiden Verfahren basierende Algorithmen speziell für MST einzuführen und eine detaillierte Laufzeitanalyse durchzuführen. Dazu werden einige wichtige Rechenwerkzeuge wie die Chernoff-Ungleichungen eingeführt. Beim Erstellen wird sich in weiten Zügen am Artikel von Neumann und Wegener orientiert.

Inhaltsverzeichnis

1 Einleitung **2**
1.1 Beschreibung des MST-Problems . . . 2
1.2 Bekannte Algorithmen zum Finden Minimaler Spannbäume . . . 2
1.2.1 Kruskal's Algorithmus . . . 2
1.2.2 Prim's Algorithmus . . . 2
1.3 Definition der Fitness-Funktionen für EA . . . 2

2 Randomisierte lokale Suche und (1+1) Evolutionärer Algorithmus **3**

3 Lokale Veränderungen an Spannbäumen **3**
3.1 Lemma: Erzeugen eines MST aus einem beliebigen Spannbaum . . . 3
3.2 Lemma: Erzeugen eines MST mit konstanter Anzahl an Flips . . . 4
3.3 Lemma: Erzeugen eines MST aus einem zshg. Graphen . . . 4

4 Detaillierte Analyse von RLS und (1+1) EA für das MST-Problem **4**
4.1 Lemma: Laufzeit zur Konstruktion eines zshg. Graphen . . . 5
4.2 Lemma: Laufzeit zur Konstruktion eines Spannbaumes . . . 6
4.3 Beispielgraph zur Bestapproximation . . . 6
4.4 Vorbemerkungen zur Analyse des Beispielgraphen . . . 7
4.4.1 Wahrscheinlichkeit $1 - o(1)$ . . . 7
4.4.2 Chernoff-Ungleichungen . . . 7
4.4.3 Markovsche Ungleichung . . . 8
4.5 Satz: Grenze für die erwartete Laufzeit . . . 8
4.5.1 Behauptung: Eigenschaften nach der Initialisierung . . . 9
4.5.2 Behauptung: Finden des Suchpunktes mit einer ZHK . . . 10
4.5.3 Behauptung: Finden des Suchpunktes, der einen Baum darstellt . . . 11
4.5.4 Behauptung: Mindestzeit zum Finden des MST . . . 11

5 Weiterführende Gedanken: Andere Mutations-Operatoren **12**

1 Einleitung

1.1 Beschreibung des MST-Problems

Gegeben sei ein zusammenhängender (im Folgenden: zshg.) gewichteter Graph $G = (V, E)$ aus n Knoten und m nicht gerichteten Kanten. Gesucht ist eine Kantenmenge $E' \subset E$ mit der Eigenschaft, dass E' alle Knoten von G verbindet und dass das Gewicht von E' minimal ist. Dann heißt $T := (V, E')$ minimaler Spannbaum (engl.: *Minimum Spanning Tree*). Das zugehörige Problem nennt man MST.
Wenn G nicht zshg. ist, kann man alternativ nach einem Minimalen Spannwald (engl.: *Minimum Spanning Forest*) suchen.
Anwendung findet die Problemstellung beispielsweise beim Auffinden kostenminimaler Netzwerke zur Versorgung einer Menge an Haushalten mit Internetzugang, Strom oder Wasser.

1.2 Bekannte Algorithmen zum Finden Minimaler Spannbäume

Zwei Algorithmen zum Auffinden minimaler Spannbäume erfreuen sich besonderer Bekanntheit und sollen deswegen hier einführend genannt werden. Beide Algorithmen finden *einen* MST; falls dieser nicht eindeutig ist, kann mehrmaliges Ausführen zu verschiedenen Ergebnissen führen.

1.2.1 Kruskal's Algorithmus

„Führe den folgenden Schritt so oft wie möglich aus: Wähle unter den noch nicht ausgewählten Kanten von G (dem Graphen) die kürzeste Kante, die mit den schon gewählten Kanten keinen Kreis bildet."[1]

1.2.2 Prim's Algorithmus

Wähle einen beliebigen Knoten k in G aus. Finde eine minimale Kante e, die k mit einem neuen Knoten k' verbindet. Setze $T = (k \cup k', e)$. Solange es Knoten aus G gibt, die noch nicht in T sind, suche weitere minimale Kanten, die einen weiteren, neuen Knoten mit T verbinden.

1.3 Definition der Fitness-Funktionen für EA

Bekannterweise gibt es eine Bijektion zwischen der Potenzmenge der Kantenmenge E zur Menge $S = \{0, 1\}^{|E|}$. Man kann also jede Teilmenge der Kantenmenge als eine Bitfolge aus 0- und 1-Bits auffassen, bei der jede Position für eine Kante steht und 1-Bits bedeuten, dass die Kante Element der Teilmenge ist. Jedes $s \in S$ nennen wir *Suchpunkt*.
Bei der Suche müssen nicht zshg. Graphen und Graphen, die keine Bäume sind, ausgeschlossen werden (*bad fitness*). Dazu stellen wir Fitness-Funktionen auf. Sei w_i das Gewicht der Kante e_i. Sei weiter w_{max} das maximale Gewicht. Dann ist $w_{ub} := n^2 \cdot w_{max}$ eine obere Grenze für das Gewicht einer Kantenmenge. Zu $s \in S$ seien nun:

$$w(s) := (c(s) - 1) \cdot w_{ub}^2 + (e(s) - (n-1)) \cdot w_{ub} + \sum_{i|s_i=1} w_i$$

$$w'(s) := (c(s) - 1) w_{ub} + \sum_{i|s_i=1} w_i$$

[1] Vgl. [1] Kruskal 1956.

wobei $e(s)$ die Anzahl der Kanten im von s beschriebenen induzierten Subgraphen von G sei und $c(s)$ die Anzahl der Zusammenhangskomponenten (im Folgenden: ZHK) von s.
Es enthält also $w'(s)$ eine Information weniger, nämlich die Information, ob s einen Baum beschreibt. Man hat jedoch eine Information weniger im Input und muss weniger Rechnungen durchführen, falls der Algorithmus auch mit dieser Fitness-Funktion funktioniert.

2 Randomisierte lokale Suche und (1+1) Evolutionärer Algorithmus

Generell geht die Formulierung *Wähle* $i \in \{1, \ldots, m\}$ *zufällig* im Folgenden von einer Gleichverteilung aus.
Es werden im Folgenden andere Algorithmen eingeführt, die man nutzen kann, um einen Minimalen Spannbaum zu finden. Diese werden detailliert auf ihre Laufzeit hin untersucht. Zunächst werden die beiden Algorithmen betrachtet.
Randomisierte Lokale Suche wird wie im Folgenden Algorithmus 1 charakterisiert:

Algorithmus 1

1. Wähle $s \in \{0,1\}^m$ zufällig.
2. Wähle $b \in \{0,1\}$ zufällig.
   (a) Falls $b = 0$, wähle $i \in \{1, \ldots, m\}$ zufällig und definiere s', indem die i-te Stelle von s geändert wird.
   (b) Falls $b = 1$, wähle $(i,j) \in \{(k,l) | 1 \leq k < l \leq m\}$ und definiere s', indem die i-te und die j-te Stelle von s geändert wird.
3. Ersetze s durch s', falls $w(s') \geq w(s)$.
4. Wiederhole 2 und 3 für immer.

Bislang fehlt das Beende-Kriterium. Dies muss sich nach der erwarteten Laufzeit richten, die der Algorithmus braucht, um einen MST zu finden. Schritt 2 des Algorithmus beschreibt den Mutationsoperator bei RLS. Dieser wird bei $(1+1)$ EA[2] folgendermaßen gewählt:

„Jede Stelle von s wird mit der Wahrscheinlichkeit $1/m$ geändert."

3 Lokale Veränderungen an Spannbäumen

Sei w_{opt} das Gewicht eines MST. Nun sollen Auswirkungen von Veränderungen am Kantenset in Bezug auf die Veränderung des Gewichtes untersucht werden. Dabei wird die Differenz $w(s) - w_{opt}$ betrachtet.

3.1 Lemma: Erzeugen eines MST aus einem beliebigen Spannbaum

Sei $s \in S$ ein Suchpunkt, der einen nicht-minimalen Spannbaum T beschreibt. Dann existiert ein $k \in 1, \ldots, n-1$ und k verschiedene akzeptierte 2-bit Flips, sodass die durchschnittliche

[2]Der (1+1) Evolutionäre Algorithmus ist ein Algorithmus, der sich an der Evolution orientiert. Dies kann man sich folgendermaßen vorstellen: Es gibt immer Elterngenerationen, aus denen nach einem bestimmten Mutations-Operator Kindergenerationen erzeugt werden. In unserem speziellen Fall gibt es immer ein Element in der Elterngeneration und ein Element in der Kindergeneration. Daher der Name $(1+1)$ EA. Allgemein besitzt ein $(\lambda+\mu)$ EA λ Elemente in der Elterngeneration und μ Elemente in der Kindergeneration.

Gewichtsverminderung dieser Flips bei mindestens $(w(s) - w_{opt})/k$ liegt.

Beweis
Sei s^* ein Suchpunkt, der einen MST T^* beschreibt. Seien $E(T)$ und $E(T^*)$ die Kantenmengen von T und T^*. Sei $k := |E(T^*) - E(T)| = |\{e \in E : e \in E(T^*) \wedge e \notin E(T)\}|$.
Dann gibt es eine Bijektion[3] $\alpha : E(T^*) - E(T) \to E(T) - E(T^*)$, sodass:
1. $\alpha(e)$ liegt auf dem Kreis in T, der kreiert wird, indem e in T eingefügt wird $\forall e \in E(T^*) - E(T)$.
2. $w(\alpha(e)) \geq w(e)$ $\forall e \in E(T^*) - E(T)$.

Wähle die k 2-bit Flips, die durch Ändern von e und $\alpha(e)$ $\forall e \in E(T^*) - E(T)$ gefunden werden. Dadurch wird T in T^* überführt bei einem Gewichtsverlust von $w(s) - w_{opt}$. □

Diesen Satz kann man vereinfachen, indem in Lemma 3.1 immer der gleiche Parameter k gewählt wird. Dazu erlaubt man nicht-akzeptierte 2-bit Flips mit einer Gewichtsverminderung definiert als 0. Wenn man zu den k 2-bit Flips aus Lemma 3.1 also $n-k$ nicht-akzeptierte Flips hinzufügt, hat man immer genau n Flips. Der Gesamtgewichtsverlust liegt dann mindestens bei $w(s)-w_{opt}$, d.h. der durchschnittliche Gewichtsverlust pro Flip bei mindestens $(w(s)-w_{opt})/n$.

3.2 Lemma: Erzeugen eines MST mit konstanter Anzahl an Flips

Sei s ein Suchpunkt, der einen Spannbaum T beschreibt. Dann existiert eine Menge von n 2-bit Flips, sodass die durchschnittliche Gewichtsabnahme für diese Flips bei $(w(s) - w_{opt})/n$ liegt.

In einem dritten Lemma wollen wir sogar nicht-Spannbäume, aber zshg. Graphen akzeptieren, deren Gewicht durch 1-bit Flips verbessert wird, wenn diese einen Kreis zerstören. Wie anfangs angekündigt, werden nicht zshg. Graphen übergangen, da man in diesem Falle jede ZHK einzeln betrachten kann.

3.3 Lemma: Erzeugen eines MST aus einem zshg. Graphen

Sei s ein Suchpunkt, der einen zshg. Graphen beschreibt. Dann gibt es eine Menge von n 2-bit Flips und eine Menge von $m - (n - 1)$ 1-bit Flips, sodass die durchschnittliche Gewichtsabnahme dieser Flips bei mindestens $(w(s) - w_{opt})/(m + 1)$ liegt.

Beweis
Zunächst finden wir $m-(n-1)$ 1-bit Flips, mit denen wir den zshg. Graphen zu einem Baum T verringern. Danach wenden wir auf T Lemma an. □

4 Detaillierte Analyse von RLS und (1+1) EA für das MST-Problem

Zunächst berechnen wir einige wesentliche Laufzeiten. Mithilfe der Laufzeiten kann man zeigen, dass RLS und (1+1) EA Spannbäume effizient konstruieren. Andererseits beweisen wir, dass die angegebene Grenze wirklich angenommen wird, die errechneten Laufzeiten also eine Bestabschätzung sind. Alle Laufzeiten sind im Folgenden erwartete Laufzeiten.

[3]Der Beweis der Existenz dieser Bijektion ist keinesfalls elementar, vgl. [2] Mayr, Plaxton 1992.

4.1 Lemma: Laufzeit zur Konstruktion eines zshg. Graphen

Die erwartete Zeit, bis RLS oder (1+1) EA mithilfe einer der beiden Fitness-Funktionen w oder w' einen zshg. Graphen konstruiert haben, ist $\mathcal{O}(m \log n)$.

Beweis
In akzeptierten Schritten wird die Anzahl der ZHK nie vergrößert. Sei s eine Kantenmenge mit k ZHK. Dann gibt es mindestens $k-1$ Kanten, deren Hinzufügen die Anzahl der ZHK um 1 reduziert. (Ansonsten wäre der ursprüngliche Graph G nicht zshg.)

Die Wahrscheinlichkeit, die Anzahl der Komponenten in einem Schritt um 1 zu verringern, beträgt mindestens $\frac{1}{2} \cdot \frac{k-1}{m}$ für RLS und $\frac{1}{e} \cdot \frac{k-1}{m}$ für (1+1) EA. Dies kann man sich folgendermaßen klarmachen. Nur im schlechtesten Fall wird der Graph um genau eine ZHK verringert. Die Wahrscheinlichkeit ist also definitiv höher, als wenn in einem Schritt der Graph um genau eine ZHK verringert wird. Dafür sind die angegebenen Wahrscheinlichkeiten untere Grenzen. Es ist in RLS zu einer Wahrscheinlichkeit von 1/2 die Variable aus RLS $b = 0$ (d.h. Wahrscheinlichkeit für genau einen Flip ist 1/2). Nun muss eine der $k-1$ Kanten gewählt werden. Dafür ist die Wahrscheinlichkeit $\frac{k-1}{m}$. Insgesamt erhalten wir also die für RLS angegebene Wahrscheinlichkeit als Mindestwahrscheinlichkeit.
Für (1+1) EA ist die Berechnung ein wenig komplizierter. Wir machen uns vor allem den Vorfaktor $1/e$ klar. Dieser resultiert aus der Wahrscheinlichkeit p, dass genau eine Kante geplippt wird. Dafür gilt nach unserer Variante von (1+1) EA:

$$p = \binom{m}{1} \cdot \left(\frac{1}{m}\right)^1 \cdot \left(\frac{m-1}{m}\right)^{m-1} = \left(\frac{m-1}{m}\right)^{m-1} = \frac{1}{(m/(m-1))^{m-1}} =$$

$$= \frac{1}{\left(\frac{m-1+1}{m-1}\right)^{m-1}} = \frac{1}{\left(1+\frac{1}{m-1}\right)^{m-1}} > \frac{1}{e}$$

Für $m \to \infty$ ist die Wahrscheinlichkeit $1/e$ der Grenzwert, der von oben angenähert wird.
Für den Erwartungswert müssen somit mindestens das Inverse der Wahrscheinlichkeiten an Durchläufen durchgeführt werden. Da uns nur die Größenordnung interessiert und $\frac{k-1}{m} = \Theta(\frac{1}{e} \cdot \frac{k-1}{m})$, werfen wir RLS und (1+1) EA in einen Topf, indem wir beides durch e abschätzen. Um die erwartete Gesamtzeit zu erhalten, bis der Graph nur noch aus einer ZHK besteht, müssen wir noch aufsummieren:

$$e \cdot m \cdot \left(1 + \cdots + \frac{1}{n-1}\right) = \mathcal{O}(m \log n)^4$$

□

Wir haben jetzt also eine Abschätzung für die Laufzeit, um einen Suchpunkt mithilfe von RLS oder (1+1) EA in einen Suchpunkt eines zshg. Graphen zu verwandeln. Aus diesem Graphen wollen wir nun einen Spannbaum konstruieren.

[4] Die Abschätzung mit der Landau-Notation kann man sich folgendermaßen klarmachen: Wenn man die Kurve $f(x) = 1/x$ betrachtet, dann erhält man die Abschätzung: $1/2 + ... + 1/n \geq \ln(n)\ \forall n \geq 2$ (Integral!).
Für $n \geq 3$ gilt $\ln(n) > 1$ und damit:

$$e \cdot m \cdot \left(1 + ... + \frac{1}{n-1}\right) \leq e \cdot m \cdot (1 + ln(n-1)) \leq e \cdot m \cdot (ln(n) + ln(n-1)) \leq e \cdot m \cdot (ln(n) + ln(n))$$

$$\leq 2 \cdot e \cdot m \cdot ln(n) = \mathcal{O}(m \ln n)\ \forall m \in \mathbb{N}\ \forall n \in \mathbb{N},\ n \geq 3.$$

4.2 Lemma: Laufzeit zur Konstruktion eines Spannbaumes

Es beschreibe s einen zshg. Graphen. Dann ist die erwartete Zeit bis RLS oder (1+1) EA einen Spannbaum mithilfe der Fitness-Funktion w konstruiert hat nach oben beschränkt durch $\mathcal{O}(m \log n)$.

Beweis
Die Fitness-Funktion w ist so definiert, dass nur zshg. Graphen aus einer Zshgskomponente akzeptiert werden. Es enthalte s nun $N \leq m$ Kanten. Dann enthält s (da der durch s definierte Graph zshg. ist) einen Spannbaum aus $n-1$ Kanten.
Fall 1:
s enthält $n-1$ Kanten. Dann beschreibt s einen Spanning Tree.
Fall 2:
s enthält $n-1 < N \leq m$ Kanten. Dann gibt es mindestens $N-(n-1)$ Kanten, deren Streichen die Anzahl der Kanten verringern würde. Im worst case sind das also $m-(n-1)$ Kanten. Wegen der Wahrscheinlichkeit $\frac{1}{e} \cdot \frac{m-(n-1)}{m}$ für (1+1) EA folgt wie in Lemma 4.1 als obere Grenze:

$$e \cdot m \cdot \left(1 + \cdots + \frac{1}{m-(n-1)}\right) = \mathcal{O}(m \cdot \log(m-(n-1))) = \mathcal{O}(m \log n)$$

□

Man sollte sich einen wichtigen Unterschied zwischen RLS und (1+1) EA bezüglich w' klarmachen. Bei RLS werden höchstens zwei Kanten verändert und es kann deswegen in einem Schritt nicht ohne Gewichtszunahme zu einer Erhöhung der Anzahl der Kanten kommen (auch bzgl. w', obwohl w' nicht die Anzahl der Kanten evaluiert!). Dies ist bei (1+1) EA möglich.

4.3 Beispielgraph zur Bestapproximation

Wir betrachten den in Abb. 1 dargestellten Beispielgraphen, mit dem wir beweisen, dass die erwartete Zeit bestmöglich abgeschätzt wird.

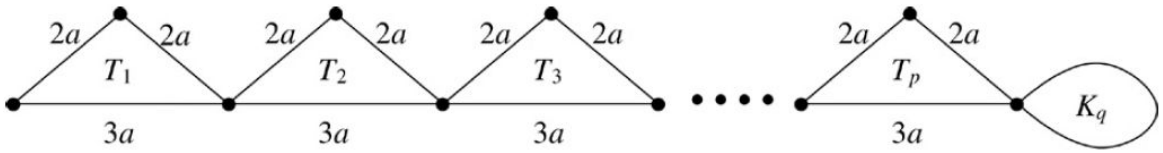

Abbildung 1: Beispielgraph, zitiert aus [3] Neumann, Wegener 2007.

Ein Graph bestehe aus p Dreiecken und am Ende aus einem vollständigen Graphen aus q Knoten (vgl. Abb. 1). Es gelten also für die Anzahl der Knoten und der Kanten: $n = 2p + q$ und $m = 3p + q(q-1)/2$.
Wir betrachten den Fall $p = n/4$ und $q = n/2$, d.h. $m = n^2/8 + n/2$; folglich gilt $m = \Theta(n^2)$. Die Kanten des vollständigen Graphen haben jeweils das Gewicht 1. Es sei $a := n^2$.

Über den Graph kann folgendes noch gesagt werden:

1. Der Graph enthält $3n/4$ Kanten im Dreieckteil und $(n+2)\frac{n}{8}$ Kanten im Cliquen-Teil.

2. Die Gesamtzahl der Kanten im Dreiecksteil liegt in der Größenordnung $\Theta(n)$, die Gesamtzahl der Kanten liegt aufgrund des Cliquenteils in der Größenordnung $\Theta(n^2)$. Dies werden wir später verwenden! (wir gehen von der späteren Definition von n schon hier aus.)

4.4 Vorbemerkungen zur Analyse des Beispielgraphen

Für den Satz 4.5 brauchen wir einige Rechenwerkzeuge und wollen uns im Vorhinein schonmal einige der wesentlichen Gedanken beim folgenden Beweis klarmachen.

4.4.1 Wahrscheinlichkeit $1 - o(1)$

Uns interessieren für den Algorithmus vor allem sehr große Graphen, denn für kleine Graphen könnte man prinzipiell ja auch einen relativ schlecht implementierten Algorithmus verwenden, da die Rechenzeit aufgrund der Größe des Graphen ohnehin klein sein wird.

Unser Algorithmus soll also vor allem für große Graphen gut sein, d.h. für Graphen mit einer großen Anzahl an Knoten n.

Deswegen reicht uns für alle der folgenden Schritte eine Fehlerwahrscheinlichkeit von $o(1)$. Dies bedeutet konkret:

$\forall\, c > 0\, \exists\, n_0$, sodass $\forall\, n \geq n_0$: Die Wahrscheinlichkeit, dass bei bei n Durchläufen durch RLS oder (1+1) EA nicht das gewünschte Ergebnis eintritt ist kleiner gleich $c \cdot 1 = c$. Für große n wird also jede mögliche Grenze unterschritten und die Trefferwahrscheinlichkeit erfolgt mit nahezu Sicherheit (genau mit $1 - o(1)$) eben.
In anderen Worten gilt für den Fehler $f(n)$, dass $lim_{n\to\infty} f(n) = 0$.

Mit diesem Hintergedanken wollen wir nun versuchen, dass in allen Schritten unseres Beweises immer die Wahrscheinlichkeit, nicht das gewünschte Ergebnis zu erreichen $o(1)$ ist.

4.4.2 Chernoff-Ungleichungen

Ein wichtiges Mittel für das Unterschreiten der Grenze $o(1)$ sind die Chernoff-Ungleichungen, die hier als wichtiges Mittel für den kommenden Beweis genannt, aber nicht bewiesen werden sollen:
Sei $X_1, X_2, \ldots, X_n$ eine Folge von n unabhängigen Bernoulli-Experimenten mit $P[X_i = 1] = p$ und $P[X_i = 0] = 1 - p$. Es ist $p \cdot n$ die erwartete Anzahl an Erfolgen ($X_i = 1$) des Experiments. Dann gelten:

1. $\forall \delta > 0$:

$$P\left[\sum X_i \geq (1+\delta) \cdot p \cdot n\right] \leq \exp\left(-\frac{\min\{\delta, \delta^2\}}{3} p \cdot n\right)$$

2. $\forall \delta \in [0, 1]$ gilt:

$$P\left[\sum X_i \leq (1-\delta) \cdot p \cdot n\right] \leq \exp\left(-\frac{\delta^2}{2} p \cdot n\right)$$

In unserem Fall wird es so sein:
Wir führen RLS oder (1+1) EA n mal durch und haben eine gewisse Wahrscheinlichkeit dafür,

dass der nächste Schritt, den wir uns gerade vornehmen, eintrifft. Die Chernoff-Ungleichungen liefern uns dann eine Abschätzung für die Abweichung vom Erwartungswert.

Für beliebiges $\delta > 0$ liegt die Wahrscheinlichkeit für eine Abweichung in $o(1)$. Somit haben wir mit den Chernoff-Ungleichungen also ein Mittel, zu zeigen, dass unser Schritt in der im Abschnitt 4.4.1 behandelten Größenordnung liegt.

4.4.3 Markovsche Ungleichung

Sei X nichtnegative, integrierbare Zufallsvariable und $a > 0$. Außerdem bezeichne $E(X)$ den Erwartungswert. Dann gilt

$$P[X \geq a] \leq \frac{E(X)}{a}.$$

Wir werden auch auf diese Ungleichung später zurückkommen, ohne sie jetzt hier zu beweisen.

4.5 Satz: Grenze für die erwartete Laufzeit

Die erwartete Optimierungszeit bis RLS oder (1+1) EA einen MST finden ist für den Beispielgraphen $\Theta(m^2 \log n) = \Theta(n^4 \log n)$.

Beweis der Untergrenze[5]

Zunächst definieren wir einige Begriffe für den weiteren Verlauf des Beweises:
Der Graph G werde partitioniert in den Dreiecks-Teil T und den Clique-Teil C. Jeder Suchpunkt x beschreibt eine Kantenmenge.

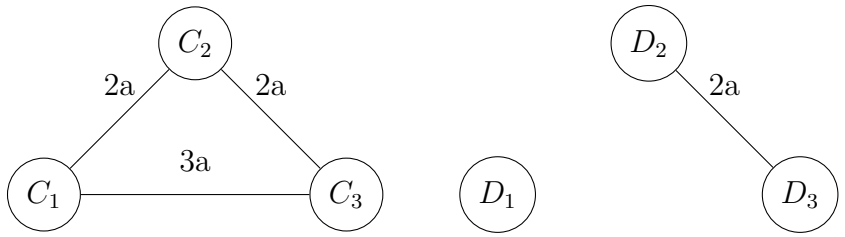

Abbildung 2: Beispiel für ein komplettes (links) und ein getrenntes Dreieck (rechts)

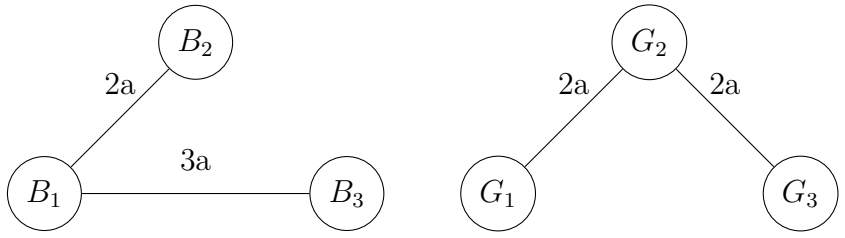

Abbildung 3: Beispiel für ein schlechtes (links) und ein gutes Dreieck (rechts)

Seien $d(x)$ die Anzahl der *getrennten* Dreiecke („disconnected“), $c(x)$ die Anzahl der *kompletten* Dreiecke („complete“), d.h. alle drei Kanten (vgl. Abb. 2), $b(x)$ die Anzahl der *schlechten*

[5]Es wird sich in diesem Rahmen auf den Beweis der Untergrenze beschränkt, obwohl es natürlich genauso wichtig ist, zu erkennen, dass man einen MST tatsächlich in der gewünschten Zeit findet. Eine Obergrenze von $\mathcal{O}(m^2(\log n + \log w_{\max}))$ wird in [3] Neumann, Wegener 2007 bewiesen. Für polynomiell beschränktes $w_{\max}$ entschspricht dies einer Obergrenze von $\mathcal{O}(m^2 \log n)$.

Dreiecke („bad“), d.h. genau eine $2a$-Kante und die $3a$-Kante, $g(x)$ die Anzahl der *guten* Dreiecke („good“), d.h. genau die beiden $2a$-Kanten (vgl. Abb. 3).

Weiter bezeichnen $con_G(x)$, $con_T(x)$ und $con_C(x)$ die Anzahl der ZHK im Graphen und in den beiden Teilen der Partition. $T(x)$ sei die Menge der Kanten aus dem Dreiecks-Teil, die durch x gewählt werden.

Wir führen den Beweis des Satzes nun in mehreren Schritten, die man folgendermaßen skizzieren kann:

1. Wir zeigen, dass mit der Initialisierung des Programms bereits zu einer Wahrscheinlichkeit von $1 - o(1)$ für den Suchpunkt x folgendes gilt: $b(x) = \Theta(n)$ und $con_C(x) = 1$.
2. Wir betrachten eine Phase der zunächst willkürlich erscheinenden Länge $n^{5/2}$ (die gewählte Länge wird später begründet). Dann wird, falls $b(x) = \Theta(n)$ und $con_C(x) = 1$ gelten, ein Suchpunkt y kreiert mit $b(y) = \Theta(n)$ und $con_G(y) = 1$. D.h. wir verringern die Zusammenhangskomponenten des gesamten Graphen auf eine ZHK.
3. Wenn $b(y) = \Theta(n)$ und $con_G(y) = 1$ gelten, so wird in einer Phase der Länge $n^{5/2}$ ein Suchpunkt z kreiert mit $b(z) = \Theta(n)$ und $con_G(z) = 1$ und $T(z)$ ist ein Baum (die Kanten des Dreiecks bilden bereits einen (beliebigen) Baum).
4. Seien $b(z) = \Theta(n)$ und $con_G(z) = 1$ und $T(z)$ ist Baum gegeben. Dann ist die erwartete Zeit bis ein MST gefunden wird $\Omega(n^4 \log n)$.

Dabei achten wir immer auf eine Fehlerhöchstwahrscheinlichkeit von $o(1)$ um so die gewünschte Trefferwahrscheinlichkeit für große n zu finden.

4.5.1 Behauptung: Eigenschaften nach der Initialisierung

Nach der Initialisierung gelten: $b(x) = \Theta(n)$ und $con_C(x) = 1$.

Beweis
Die Wahrscheinlichkeit, dass ein Dreieck schlecht ist, beträgt $1/4$. Da es $n/4$ Dreiecke gibt, folgt wegen der Chernoff-Ungleichung $b(x) = \Theta(n)$.

Diese Folgerung mit der Chernoff-Ungleichung wollen wir uns noch etwas ausführlicher überlegen. Es gilt also $p = 1/4$ für die Wahrscheinlichkeit, dass ein schlechtes Dreieck kreiert wird. (Für jedes Dreieck ist jede der drei Kanten zu 50 % Wahrscheinlichkeit im Graphen. Insgesamt gibt es 8 gleichwahrscheinliche Möglichkeiten, von denen zwei zu einem schlechten Dreieck führen.) Da es $n/4$ Dreiecke gibt, wird das *Experiment* aus der Chernoff-Ungleichung $n/4$ mal durchgeführt. (Vorsicht, hier ist n ein wenig schlecht gewählt; für das $n =: \bar{n}$ aus dem Satz der Chernoff-Ungleichung gilt also $\bar{n} = n/4$.) Somit ist der Erwartungswert für die Dreiecke $p \cdot n = \frac{1}{4} \cdot \frac{n}{4} = \frac{n}{16} = \Theta(n)$. Dieser wird mit mit einer Wahrscheinlichkeit von $1 - o(1)$ (wegen den Chernoff-Ungleichungen) angenommen.
Mit derselben Argumentation wird im Folgenden vielfach hantiert, ohne weiterhin so ausführliche Begründungen zu liefern.

Sei v ein Knoten in C. v hat $n/2-1$ mögliche Nachbarn. Laut der Chernoff-Ungleichung ist die Wahrscheinlichkeit sehr hoch (sehr hoch bedeutet in unserem Fall immer $1-o(1)$), dass v mit mindestens $n/6$ Knoten verbunden ist. Für jeden weiteren Knoten ist die Wahrscheinlichkeit, mit keinem dieser $n/6$ Knoten verbunden zu sein bei $(1/2)^{n/6}$. Es ist also sehr unwahrscheinlich (d.h. Wahrscheinlichkeit ist $o(1)$), dass es mehr als eine ZHK in der Clique gibt. Damit ist der erste Abschnitt des Satzes im Rahmen der gewünschten Wahrscheinlichkeit gezeigt. □

Im Folgenden werden die Schritte von RLS und (1+1) EA dadurch entschieden, wie viele der Dreiecks-Kanten verändert werden. Wegen den Fitness-Funktionen tritt der Fall nicht ein, dass durch das Entfernen einer Kante aus dem Cliquen-Teil mehrere ZHK entstehen.

Sei nun ein k-Schritt ein Schritt, in dem k Dreieckskanten getauscht werden. (d.h. es müssen i.a. nicht notwendigerweise nur k Kanten bei einem k-Schritt verändert werden). Sei p_k die Wahrscheinlichkeit für einen k-Schritt. Es gilt dann für RLS:
$p_1 = \Theta(n^{-1})$, $p_2 = \Theta(n^{-2})$ und $p_k = 0\ \forall k \geq 3$. Diese Wahrscheinlichkeiten kann man sich folgendermaßen klarmachen:

Es gibt $3n/4 = \Theta(n)$ Dreieckskanten, außerdem insgesamt $\Theta(n^2)$ Kanten, d.h.

$$p_1 = \frac{1}{2} \cdot \Theta\left(\frac{n}{n^2}\right) = \Theta(n^{-1})$$

Analog gilt $p_2 = \frac{1}{2} \cdot \left(\Theta\left(\frac{n}{n^2}\right)\right)^2 = \Theta(n^{-2})$.

Für (1+1) EA und konstantes k berechnet sich die Wahrscheinlichkeit nach einem Bernoulli-Experiment:

$$p_k = \binom{3n/4}{k} \left(\frac{1}{m}\right)^k \left(1 - \frac{1}{m}\right)^{3n/4-k} = \Theta(n^k m^{-k}) = \Theta(n^{-k})$$

Nun betrachten wir die Phase der Länge $n^{5/2}$.

4.5.2 Behauptung: Finden des Suchpunktes mit einer ZHK

Sei $b(x) = \Theta(n)$ und $con_C(x) = 1$. In einer Phase der Länge $n^{5/2}$ wird ein Suchpunkt y mit $b(y) = \Theta(n)$ und $con_G(y) = 1$ produziert.

Beweis
Laut Lemma 4.1 wissen wir, dass die Wahrscheinlichkeit, einen zusammenhängenden Graphen zu finden, hoch genug ist.

Es gilt nämlich wegen $m = n^2/8+n/2$, also $\mathcal{O}(m \log n) = \mathcal{O}((n^2/8+n/2) \log n) = \mathcal{O}(n^2 \log n) = o(n^{5/2})$[6] die genannte Abschätzung.

Sei nun also y der erste Suchpunkt, sodass $con_G(y) = 1$. Zu zeigen bleibt also, dass $b(y) = \Theta(n)$. Alle 2-Schritte können den b-Wert höchstens um $\mathcal{O}(n^{1/2})$ verringern. (vgl. Vorbemerkungen)

Welche Möglichkeiten hat ein 1-step, ein schlechtes Dreieck zu zerstören?
1. Es könnte die Kante eines schlechten Dreiecks entfernt werden. Dies würde $con_G(y)$ erhöhen. Gleichzeitig müsste also der con_C-Wert erniedrigt werden. Dies ist wegen $con_C(y) = 1$ nicht möglich.
2. Es könnte die fehlende Kante des Dreiecks eingefügt werden, folglich also $con_c(y)$ wieder erniedrigt werden. Dies ist auch nicht möglich, da sich das Gewicht des Graphen nur erhöhen würde.

Ein Schritt, der den con_C-Wert erhöht, muss notwendigerweise den con_T-Wert um mindestens denselben Wert erniedrigen. Für jeden 1-Schritt wird dann aber das Gewicht des Graphens

[6]Das letzte Gleichheitszeichen kann man nachrechnen, indem man $\lim_{n\to\infty} \frac{n^2 \log(n)}{n^{5/2}} = 0$ zeigt.

erhöht und der Schritt würde von keiner der Fitness-Funktionen akzeptiert werden.

Es werden also nur schlechte Dreiecke durch $\mathcal{O}(n^{1/2})$ 2-Schritte hinzugefügt. Dies impliziert, dass wir insgesamt $\Theta(n) + \mathcal{O}(n^{1/2}) = \Theta(n)$ schlechte Dreiecke im Suchpunkt y haben.

□

4.5.3 Behauptung: Finden des Suchpunktes, der einen Baum darstellt

Sei $b(y) = \Theta(n)$ und $con_G(y) = 1$. In einer Phase der Länge $n^{5/2}$ wird ein Suchpunkt z mit $b(z) = \Theta(n)$, $con_G(z) = 1$ und $T(z)$ ein Baum produziert.

Beweis
Wegen Behauptung 4.5.1 wissen wir, dass $d(z) = 0$. Wir müssen zeigen, dass $b(z) = \Theta(n)$ und dass z innerhalb der gewünschten Anzahl an Schritten gefunden wird.
Damit $T(z)$ ein Baum wird, müssen alle kompletten Dreiecke eliminiert werden. Es darf also nur noch gute und schlechte Dreiecke geben. Ein 1-Schritt kann nur dann akzeptiert werden, wenn er ein komplettes Dreieck in ein gutes oder schlechtes Dreieck verwandelt. (Es kann wegen der Fitness-Funktionen nicht die Anzahl der ZHK erhöht werden.) Weiterhin kann die Anzahl der kompletten Dreiecke nicht erhöht werden. Um ein komplettes Dreieck zu erhalten, muss gleichzeitig ein schon vorhandenes komplettes Dreieck in ein gutes oder schlechtes Dreieck umgewandelt werden, damit die Gewichtszunahme kompensiert wird.

Sei nun $c(x) = l$, dann ist die Wahrscheinlichkeit, den c-Wert zu verringern mindestens $3l/(e \cdot m)$ (vgl. Lemma 4.1). Wie in Lemma 4.1 folgt für die erwartete Zeit, bis alle kompletten Dreiecke eliminiert, sind $\mathcal{O}(m \log n) = \mathcal{O}(n^2 \log n)$.

Analog zur Behauptung 4.5.2 wissen wir also, dass die Anzahl der Durchläufe ausreicht, um einen Baum zu finden.
Außerdem wissen wir analog zur vorherigen Behauptung, dass die Anzahl der schlechten Dreiecke in den $\mathcal{O}(n^{1/2})$ 2-Schritten verringert werden kann, was $b(z) = \Theta(n)$ impliziert. □

4.5.4 Behauptung: Mindestzeit zum Finden des MST

Sei $b(z) = \Theta(n)$, $con_G(z) = 1$, $T(z)$ ein Baum. Die erwartete Zeit, um einen MST zu finden, ist dann $\Omega(n^4 \log n)$.

Beweis
Die Wahrscheinlichkeit, dass ein schlechtes in ein gutes Dreieck umgewandelt wird, liegt bei $\Theta(n^{-4})$. Dies setzt sich zusammen aus: Wahrscheinlichkeit für einen 2-Schritt ($\mathcal{O}(n^{-2})$), Wahrscheinlichkeit, dass die beiden Kanten gewählt werden ($\mathcal{O}(n^{-1})$). Damit liegt die erwartete Zeit für das Umwandeln eines schelchten in ein gutes Dreieck bei $\mathcal{O}(n^4)$.
Um das b-te schlechte Dreieck in ein gutes Dreieck umzuwandeln, wird die Zeit $\mathcal{O}(\frac{n^4}{b})$ benötigt. (Wahrscheinlichkeit mit b multiplizieren); insgesamt ist die erwartete Zeit, um alle schlechten Dreiecke in gute Dreiecke umgewandelt sind, da b von n bis 1 laufen muss, bei:

$$\Theta\left(n^4 \sum_{1 \leq b \leq \Theta(n)} (1/b)\right) = \Theta(n^4 \log n).^7$$

[7] Man muss sich jetzt eigentlich noch Gedanken machen, dass es keine schnellere Möglichkeit gibt, die schlechten Dreiecke zu eliminieren, indem man zeigt, dass k-Schritte für $k \neq 2$ die schlechten Dreiecke nicht schneller eliminieren.

Das Abschätzen der Summe mit dem Logarithmus läuft analog zu Lemma 4.1.

□

Damit ist Satz 4.5 bewiesen.

5 Weiterführende Gedanken: Andere Mutations-Operatoren

Die Algorithmen sind bis jetzt noch relativ einfach gehalten und es ist natürlich eine Vielzahl an Variationen denkbar. Wie man an den Lemmata 4.1 und 4.2 sieht, ist die Konstruktion eines Spannbaumes relativ einfach.
Sobald ein Spannbaum konstruiert ist, ist es sinnvoll, wenn man Kindgenerationen (im Sinne von EA) verwendet, die die Anzahl der Kanten nicht verändern. Dazu könnten Mutations-Operatoren folgendermaßen aussehen:

1. RLS: falls ein 2-bit Flip stattfindet, wird zufällig ja ein 0-bit und ein 1-bit getauscht.
2. (1+1) EA: beinhaltet die Kantenmenge s insgesamt k 1-bits, so wird jeder 1-bit mit der Wahrscheinlichkeit $1/k$ und jeder 0-bit mit der Wahrscheinlichkeit $1/(m-k)$ getauscht.

Literatur

[1] **Kruskal, Joseph**: On the shortest spanning subtree and the traveling salesman problem. In: Proceedings of the American Mathematical Society. 7 (1956), S. 48–50.

[2] **Mayr, Ernst W.; Plaxton, C. Greg**: On the spanning trees of weighted graphs. In: Combinatoria 12 (1992), S. 433-447.

[3] **Neumann, Frank; Wegener, Ingo**: Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In: Theoretical Computer Science 378 (2007), S. 32-40.

Dazu muss man für alle anderen Schritte den Einfluss auf $o(n^4 \log n)$ limitieren.