



# Outline

- 1 Einführung
  - Bekannte Algorithmen
  - Fitnessfunktionen
- 2 Randomisierte lokale Suche und (1+1) Evolutionärer Algorithmus
  - Randomisierte Lokale Suche
  - Evolutionäre Algorithmen
- 3 Lokale Veränderungen an Spanning Trees
  - Lemmata zu lokalen Veränderungen
- 4 Die Analyse von RLS und (1+1) EA für das MST-Problem
  - Konstruktion eines zshg. Graphen
  - Konstruktion eines Baums
- 5 Beispielgraph
  - Rechenwerkzeuge zur Betrachtung des Beispielgraphen
  - Satz zum Beispielgraphen
- 6 Weiterführende Gedanken


# Grundproblem

- Zshg. Graph  $G = (V, E)$  aus  $n$  Knoten und  $m$  nicht gerichteten Kanten
- Gesucht: Kantenmenge  $E'$ , sodass  $E'$  alle Knoten von  $G$  verbindet und dass das Gewicht von  $E'$  minimal ist
- $G$  nicht zshg.  $\rightarrow$  *Minimum Spanning Forest*

# Kruskal's Algorithmus

*Führe den folgenden Schritt so oft wie möglich aus: Wähle unter den noch nicht ausgewählten Kanten von  $G$  die kürzeste Kante, die mit den schon gewählten Kanten keinen Kreis bildet.<sup>1</sup>*

---

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

# 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.

# Definitionen

- 1 Eine *Fitnessfunktion* ist die Zielfunktion eines evolutionären Algorithmus, die als Mittel dient, Kandidaten zu vergleichen.

# Definitionen

- 1 Eine *Fitnessfunktion* ist die Zielfunktion eines evolutionären Algorithmus, die als Mittel dient, Kandidaten zu vergleichen.
- 2 Bekanntermaßen gibt es eine Bijektion  $\psi$  zwischen der Potenzmenge der Kantenmenge und  $S = \{0, 1\}^m$ . Für Elemente  $s \in S$  sprechen wir von einem *Suchpunkt*, den wir genauso gut mit einer Teilmenge der Kantenmenge eindeutig identifizieren könnten.

# Definitionen

- 1 Eine *Fitnessfunktion* ist die Zielfunktion eines evolutionären Algorithmus, die als Mittel dient, Kandidaten zu vergleichen.
- 2 Bekanntermaßen gibt es eine Bijektion  $\psi$  zwischen der Potenzmenge der Kantenmenge und  $S = \{0, 1\}^m$ . Für Elemente  $s \in S$  sprechen wir von einem *Suchpunkt*, den wir genauso gut mit einer Teilmenge der Kantenmenge eindeutig identifizieren könnten.

Bemerkung: Wir gehen von einer Gleichverteilung aus, d.h. für jede Kante ist es gleich wahrscheinlich, Element eines Suchpunkts zu sein.



# Fitnessfunktionen

- Sei  $w_i$  das Gewicht der Kante  $e_i$ . Sei weiter  $w_{\max}$  das maximale Gewicht.

# Fitnessfunktionen

- Sei  $w_i$  das Gewicht der Kante  $e_i$ . Sei weiter  $w_{\max}$  das maximale Gewicht.
- Dann:  $w_{\text{ub}} := n^2 \cdot w_{\max}$  eine obere Grenze für das Gewicht eines Kantensets.

# Fitnessfunktionen

- Sei  $w_i$  das Gewicht der Kante  $e_i$ . Sei weiter  $w_{\max}$  das maximale Gewicht.
- Dann:  $w_{\text{ub}} := n^2 \cdot w_{\max}$  eine obere Grenze für das Gewicht eines Kantensets.
- Zu  $s \in S$  seien nun:

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

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

# Fitnessfunktionen

- Sei  $w_i$  das Gewicht der Kante  $e_i$ . Sei weiter  $w_{\max}$  das maximale Gewicht.
- Dann:  $w_{\text{ub}} := n^2 \cdot w_{\max}$  eine obere Grenze für das Gewicht eines Kantensets.
- Zu  $s \in S$  seien nun:

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

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

wobei  $e(s)$  die Anzahl der Kanten im Graph sei und  $c(s)$  die Anzahl der Zusammenhangskomponenten.

# Algorithmus für Randomisierte Lokale Suche (RLS)

- 1 Wähle  $s \in \{0, 1\}^m$  zufällig.

# Algorithmus für Randomisierte Lokale Suche (RLS)

- 1 Wähle  $s \in \{0, 1\}^m$  zufällig.
- 2 Wähle  $b \in \{0, 1\}$  zufällig.
  - (a) Falls  $b = 0$ , wähle  $i \in \{1, \dots, m\}$  zufällig und definiere  $s'$ , indem die  $i$ -te Stelle von  $s$  geändert wird.
  - (b) Falls  $b = 1$ , wähle  $(i, j) \in \{(k, l) \mid 1 \leq k < l \leq m\}$  und definiere  $s'$ , indem die  $i$ -te und die  $j$ -te Stelle von  $s$  geändert wird.

# Algorithmus für Randomisierte Lokale Suche (RLS)

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

# Algorithmus für Randomisierte Lokale Suche (RLS)

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



# Mutations-Operator des (1+1) EA

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

# Mutations-Operator des (1+1) EA

- Jede Stelle von  $s$  wird mit der Wahrscheinlichkeit  $1/m$  geändert.
- Dieser Operator ersetzt bei EA Schritt 1 und 2 aus dem Algorithmus für RLS.

# Mutations-Operator des (1+1) EA

- Jede Stelle von  $s$  wird mit der Wahrscheinlichkeit  $1/m$  geändert.
- Dieser Operator ersetzt bei EA Schritt 1 und 2 aus dem Algorithmus für RLS.
- Bei beiden Algorithmen fehlt noch das Beende-Kriterium  $\rightarrow$  Laufzeit

# Lemma 1.1

Sei  $w_{opt}$  das Gewicht eines MST.

Sei  $s \in S$  ein Suchpunkt, der einen nicht-minimalen Spannbaum  $T$  beschreibt. Dann exist. ein  $k \in \{1, \dots, n - 1\}$  und  $k$  verschiedene akzeptierte 2-bit Flips, sodass die durchschnittliche 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  $k := |E(T^*) - E(T)|$ .

---

<sup>2</sup>Der Beweis der Existenz dieser Bijektion ist nicht trivial. Siehe hierzu: Mayr, Ernst W; Plaxton, C. Greg: On the spanning trees of weighted graphs. In: Combinatorica (1992), S. 433-447.

# Beweis

- Sei  $s^*$  ein Suchpunkt, der einen MST  $T^*$  beschreibt. Seien  $E(T)$  und  $E(T^*)$  die Kantenmengen von  $T$  und  $T^*$ . Sei  $k := |E(T^*) - E(T)|$ . Dann gibt es eine Bijektion<sup>2</sup>  $\alpha : E(T^*) - E(T) \rightarrow E(T) - E(T^*)$ , sodass:
1.  $\alpha(e)$  liegt auf dem Kreis in  $T$ , der kreierte wird, indem  $e$  in  $T$  eingefügt wird  $\forall e \in E(T^*) - E(T)$ .
  2.  $w(\alpha(e)) \geq w(e) \forall e \in E(T^*) - E(T)$ .

---

<sup>2</sup>Der Beweis der Existenz dieser Bijektion ist nicht trivial. Siehe hierzu: Mayr, Ernst W; Plaxton, C. Greg: On the spanning trees of weighted graphs. In: Combinatorica (1992), S. 433-447.

# Beweis

Sei  $s^*$  ein Suchpunkt, der einen MST  $T^*$  beschreibt. Seien  $E(T)$  und  $E(T^*)$  die Kantenmengen von  $T$  und  $T^*$ . Sei  $k := |E(T^*) - E(T)|$ .

Dann gibt es eine Bijektion<sup>2</sup>  $\alpha : E(T^*) - E(T) \rightarrow E(T) - E(T^*)$ , sodass:

1.  $\alpha(e)$  liegt auf dem Kreis in  $T$ , der kreierte 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  $w(s) - w_{\text{opt}}$ . □

---

<sup>2</sup>Der Beweis der Existenz dieser Bijektion ist nicht trivial. Siehe hierzu: Mayr, Ernst W; Plaxton, C. Greg: On the spanning trees of weighted graphs. In: Combinatorica (1992), S. 433-447.

## Lemma 1.2

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  $(w(s) - w_{\text{opt}})/n$  liegt.



## Lemma 1.2

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  $(w(s) - w_{\text{opt}})/n$  liegt.

- Vereinfachung von vorherigem Lemma.
- $n - k$  nicht-akzeptierte Flips werden hinzugefügt (Gewichtsverminderung bei 0)



## Lemma 1.3

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.

## Lemma 1.3

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 können wir auf  $T$  das soeben bewiesene Lemma anwenden. □

## Lemma 2.1: Erwartete Zeit für 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: Anzahl der Zusammenhangskomponenten (im Folgenden: ZHK) wird nie vergrößert.

# Beweis

- In akzeptierten Schritten: Anzahl der Zusammenhangskomponenten (im Folgenden: ZHK) wird 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.

# Beweis

- In akzeptierten Schritten: Anzahl der Zusammenhangskomponenten (im Folgenden: ZHK) wird 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.
- 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
  - mindestens  $\frac{1}{e} \cdot \frac{k-1}{m}$  für (1+1) EA

# Begründung der Wahrscheinlichkeit für RLS

- Mit einer Wahrscheinlichkeit von  $1/2$  ist die Variable  $b = 0$ .
- Es muss eine der  $k - 1$  von  $m$  Kanten gewählt werden.



# Begründung der Wahrscheinlichkeit für (1+1) EA

Vorfaktor  $1/e$ ?  $\rightarrow$  Wahrscheinlichkeit  $p$ , dass genau eine Kante geflippt wird. Es gilt:

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

# Begründung der Wahrscheinlichkeit für (1+1) EA

Vorfaktor  $1/e$ ?  $\rightarrow$  Wahrscheinlichkeit  $p$ , dass genau eine Kante geflippt wird. Es gilt:

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

# Begründung der Wahrscheinlichkeit für (1+1) EA

Vorfaktor  $1/e$ ?  $\rightarrow$  Wahrscheinlichkeit  $p$ , dass genau eine Kante geflippt wird. Es gilt:

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

# Begründung der Wahrscheinlichkeit für (1+1) EA

Vorfaktor  $1/e$ ?  $\rightarrow$  Wahrscheinlichkeit  $p$ , dass genau eine Kante geflippt wird. Es gilt:

$$\begin{aligned} p &= \binom{m}{1} \cdot \left(\frac{1}{m}\right)^1 \cdot \left(\frac{m-1}{m}\right)^{m-1} = \left(\frac{m-1}{m}\right)^{m-1} \\ &= \frac{1}{(m/(m-1))^{m-1}} = \frac{1}{\left(\frac{m-1+1}{m-1}\right)^{m-1}} \end{aligned}$$

# Begründung der Wahrscheinlichkeit für (1+1) EA

Vorfaktor  $1/e$ ?  $\rightarrow$  Wahrscheinlichkeit  $p$ , dass genau eine Kante geflippt wird. Es gilt:

$$\begin{aligned} p &= \binom{m}{1} \cdot \left(\frac{1}{m}\right)^1 \cdot \left(\frac{m-1}{m}\right)^{m-1} = \left(\frac{m-1}{m}\right)^{m-1} \\ &= \frac{1}{(m/(m-1))^{m-1}} = \frac{1}{\left(\frac{m-1+1}{m-1}\right)^{m-1}} = \frac{1}{\left(1 + \frac{1}{m-1}\right)^{m-1}} \end{aligned}$$

# Begründung der Wahrscheinlichkeit für (1+1) EA

Vorfaktor  $1/e$ ?  $\rightarrow$  Wahrscheinlichkeit  $p$ , dass genau eine Kante geflippt wird. Es gilt:

$$\begin{aligned} p &= \binom{m}{1} \cdot \left(\frac{1}{m}\right)^1 \cdot \left(\frac{m-1}{m}\right)^{m-1} = \left(\frac{m-1}{m}\right)^{m-1} \\ &= \frac{1}{(m/(m-1))^{m-1}} = \frac{1}{\left(\frac{m-1+1}{m-1}\right)^{m-1}} = \frac{1}{\left(1 + \frac{1}{m-1}\right)^{m-1}} > \frac{1}{e} \end{aligned}$$

# Begründung der Wahrscheinlichkeit für (1+1) EA

Vorfaktor  $1/e$ ?  $\rightarrow$  Wahrscheinlichkeit  $p$ , dass genau eine Kante geflippt wird. Es gilt:

$$\begin{aligned} p &= \binom{m}{1} \cdot \left(\frac{1}{m}\right)^1 \cdot \left(\frac{m-1}{m}\right)^{m-1} = \left(\frac{m-1}{m}\right)^{m-1} \\ &= \frac{1}{(m/(m-1))^{m-1}} = \frac{1}{\left(\frac{m-1+1}{m-1}\right)^{m-1}} = \frac{1}{\left(1 + \frac{1}{m-1}\right)^{m-1}} > \frac{1}{e} \end{aligned}$$

Für  $m \rightarrow \infty$  ist die Wahrscheinlichkeit  $1/e$  der Grenzwert, der von oben angenähert wird.

# Erwartete Gesamtzeit

Aufsummieren für Gesamtzeit:

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





## Lemma 2.2: Erw. Zeit für die Konstruktion eines Baums

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 durch  $\mathcal{O}(m \log n)$  beschränkt.

# Beweis

Die Fitness-Funktion  $w$  ist so definiert, dass nur zshg. Graphen aus einer ZHK akzeptiert werden. Es enthalte  $s$  nun  $N \leq m$  Kanten. Dann enthält  $s$  einen Spannbaum aus  $n - 1$  Kanten.

# Beweis

Die Fitness-Funktion  $w$  ist so definiert, dass nur zshg. Graphen aus einer ZHK akzeptiert werden. Es enthalte  $s$  nun  $N \leq m$  Kanten. Dann enthält  $s$  einen Spannbaum aus  $n - 1$  Kanten.

Es treten folgende Fälle auf:

- 1  $s$  enthält  $n - 1$  Kanten. Dann beschreibt  $s$  bereits einen Spannbaum.

# Beweis

Die Fitness-Funktion  $w$  ist so definiert, dass nur zshg. Graphen aus einer ZHK akzeptiert werden. Es enthalte  $s$  nun  $N \leq m$  Kanten. Dann enthält  $s$  einen Spannbaum aus  $n - 1$  Kanten.

Es treten folgende Fälle auf:

- 1  $s$  enthält  $n - 1$  Kanten. Dann beschreibt  $s$  bereits einen Spannbaum.
- 2  $s$  enthält  $n - 1 < N \leq m$  Kanten.

## Fall 2

Im zweiten Fall gibt es mind.  $N - (n - 1)$  Kanten, deren Streichen die Anzahl der Kanten verringern würde ohne den Zusammenhang zu zerstören.

## Fall 2

Im zweiten Fall gibt es mind.  $N - (n - 1)$  Kanten, deren Streichen die Anzahl der Kanten verringern würde ohne den Zusammenhang zu zerstören.

worst case:  $m - (n - 1)$  Kanten.

## Fall 2

Im zweiten Fall gibt es mind.  $N - (n - 1)$  Kanten, deren Streichen die Anzahl der Kanten verringern würde ohne den Zusammenhang zu zerstören.

worst case:  $m - (n - 1)$  Kanten.

Wegen der Wahrscheinlichkeit  $\frac{1}{e} \cdot \frac{m - (n - 1)}{m}$  für (1+1) EA folgt wie in Lemma 2.1 (Erwartete Zeit für die Konstruktion eines zshg. Graphen) als obere Grenze:

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

□

# Unterschied zwischen RLS und (1+1) EA bezüglich $w'$

RLS: es werden höchstens zwei Kanten verändert und es kann deswegen in einem Schritt nicht ohne Gewichtszunahme zu einer Erhöhung der Anzahl der Kanten kommen! Dies ist bei (1+1) EA möglich.



# Darstellung des Beispielgraphen

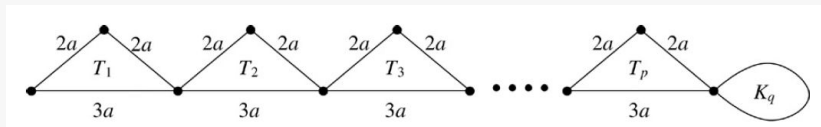


Abbildung: Beispielgraph<sup>3</sup>

<sup>3</sup>Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theoretical Computer Science 378 (2007) 32-40.

# Beschreibung des Beispielgraphen

- Graph besteht aus  $p$  Dreiecken und einem vollständigen Graphen aus  $q$  Knoten

# Beschreibung des Beispielgraphen

- Graph besteht aus  $p$  Dreiecken und einem vollständigen Graphen aus  $q$  Knoten
- Für Knoten und Kanten gilt also:  $n = 2p + q$  und  $m = 3p + q(q - 1)/2$

# Beschreibung des Beispielgraphen

- Graph besteht aus  $p$  Dreiecken und einem vollständigen Graphen aus  $q$  Knoten
- Für Knoten und Kanten gilt also:  $n = 2p + q$  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  $a := n^2$

# Wahrscheinlichkeit $1 - o(1)$

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

# Wahrscheinlichkeit $1 - o(1)$

- Unser Algorithmus soll vor allem für große Graphen gut sein, d.h. für Graphen mit einer großen Anzahl an Knoten  $n$ .
- Uns reicht für alle der folgenden Schritte eine Fehlerwahrscheinlichkeit von  $o(1)$

# Wahrscheinlichkeit $1 - o(1)$

- Unser Algorithmus soll vor allem für große Graphen gut sein, d.h. für Graphen mit einer großen Anzahl an Knoten  $n$ .
- Uns reicht für alle der folgenden Schritte eine Fehlerwahrscheinlichkeit von  $o(1)$
- Konkret:  
 $\forall c > 0 \exists n_0: \forall n \geq n_0$ : Die Wahrscheinlichkeit, dass nicht das gewünschte Ergebnis eintritt ist kleiner gleich  $c \cdot 1 = c$ .
- Die Fehlerwahrscheinlichkeit konvergiert also gegen 0.

# Chernoff-Ungleichungen

Sei  $X_1, X_2, \dots, X_n$  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  $\forall \delta > 0$  :

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



# Chernoff-Ungleichungen

Sei  $X_1, X_2, \dots, X_n$  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  $\forall \delta > 0$  :

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

②  $\forall \delta \in [0, 1]$ :

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

# Bedeutung

- Wir führen RLS oder  $(1+1)$  EA  $k$  Mal durch
- Chernoff-Ugl. liefern eine Abschätzung für die Abweichung vom Erwartungswert und damit Grenzen für die Abweichung von der Fehlerwahrscheinlichkeit

# Bedeutung

- Wir führen RLS oder  $(1+1)$  EA  $k$  Mal durch
- Chernoff-Ugl. liefern eine Abschätzung für die Abweichung vom Erwartungswert und damit Grenzen für die Abweichung von der Fehlerwahrscheinlichkeit
- Für beliebiges  $\delta$  liegt die Wahrscheinlichkeit für eine Abweichung in  $o(1)$
- Chernoff-Ugl. sind Mittel zum Erreichen der Fehlerwahrscheinlichkeit  $o(1)$

# Markovsche Ungleichung

Sei  $X$  nichtnegative, integrierbare Zufallsvariable und  $a > 0$ , deren Erwartungswert  $E(X)$  existiert. Dann gilt

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

# Satz zum Beispielgraphen

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

---

<sup>4</sup>Im Vortrag wird nur  $\Omega(m^2 \log n)$  gezeigt.

# Notation

- Der Graph  $G$  werde partitioniert in den Dreiecksteil  $T$  und den Cliqueteil  $C$ . Der Suchpunkt  $x$  beschreibe die ihm zugehörige Kantenmenge.

---

<sup>5</sup>Jedes komplette Dreieck ist also einzeln betrachtet vollständig.

# Notation

- Der Graph  $G$  werde partitioniert in den Dreiecksteil  $T$  und den Cliqueteil  $C$ . Der Suchpunkt  $x$  beschreibe die ihm zugehörige Kantenmenge.
- $d(x)$ : Anzahl der *getrennten* Dreiecke („disconnected“)
- $b(x)$ : Anzahl der *schlechten* Dreiecke („bad“), d.h. genau eine 2a-Kante und die 3a-Kante
- $g(x)$ : Anzahl der *guten* Dreiecke („good“), d.h. genau die beiden 2a-Kanten
- $c(x)$  die Anzahl der *kompletten*<sup>5</sup> Dreiecke („complete“)

---

<sup>5</sup>Jedes komplette Dreieck ist also einzeln betrachtet vollständig. 

# Notation

- Der Graph  $G$  werde partitioniert in den Dreiecksteil  $T$  und den Cliqueteil  $C$ . Der Suchpunkt  $x$  beschreibe die ihm zugehörige Kantenmenge.
- $d(x)$ : Anzahl der *getrennten* Dreiecke („disconnected“)
- $b(x)$ : Anzahl der *schlechten* Dreiecke („bad“), d.h. genau eine 2a-Kante und die 3a-Kante
- $g(x)$ : Anzahl der *guten* Dreiecke („good“), d.h. genau die beiden 2a-Kanten
- $c(x)$  die Anzahl der *kompletten*<sup>5</sup> Dreiecke („complete“)
- $con_G(x)$ ,  $con_T(x)$  und  $con_C(x)$ : Anzahl der ZHK im Graphen und in den beiden Teilen der Partition
- $T(x)$ : Menge der Kanten aus dem Dreiecksteil

---

<sup>5</sup>Jedes komplette Dreieck ist also einzeln betrachtet vollständig. 



# Beweisskizze

- 1 Mit der Initialisierung des Programms gilt bereits zu einer Wahrscheinlichkeit von  $1 - o(1)$  für das edge set  $x$ :  $b(x) = \Theta(n)$  und  $con_C(x) = 1$ .

# Beweisskizze

- 1 Mit der Initialisierung des Programms gilt bereits zu einer Wahrscheinlichkeit von  $1 - o(1)$  für das edge set  $x$ :  $b(x) = \Theta(n)$  und  $con_C(x) = 1$ .
- 2 Wir betrachten eine Phase der Länge  $n^{5/2}$  und erzeugen einen Suchpunkt  $y$  mit  $b(y) = \Theta(n)$  und  $con_G(y) = 1$ .

# Beweisskizze

- 1 Mit der Initialisierung des Programms gilt bereits zu einer Wahrscheinlichkeit von  $1 - o(1)$  für das edge set  $x$ :  $b(x) = \Theta(n)$  und  $con_C(x) = 1$ .
- 2 Wir betrachten eine Phase der Länge  $n^{5/2}$  und erzeugen einen Suchpunkt  $y$  mit  $b(y) = \Theta(n)$  und  $con_G(y) = 1$ .
- 3 Wir kreieren einen Suchpunkt  $z$  mit  $b(z) = \Theta(n)$  und  $con_G(z) = 1$  und  $T(z)$  ist ein Baum.

# Beweisskizze

- 1 Mit der Initialisierung des Programms gilt bereits zu einer Wahrscheinlichkeit von  $1 - o(1)$  für das edge set  $x$ :  $b(x) = \Theta(n)$  und  $con_C(x) = 1$ .
- 2 Wir betrachten eine Phase der Länge  $n^{5/2}$  und erzeugen einen Suchpunkt  $y$  mit  $b(y) = \Theta(n)$  und  $con_G(y) = 1$ .
- 3 Wir kreieren einen Suchpunkt  $z$  mit  $b(z) = \Theta(n)$  und  $con_G(z) = 1$  und  $T(z)$  ist ein Baum.
- 4 Wir schätzen die erwartete Zeit bis ein MST gefunden wird auf  $\Omega(n^4 \log n)$ .

# Behauptung 3.1

Nach der Initialisierung gelten:

- $b(x) = \Theta(n)$
- $con_C(x) = 1$ .

# Beweis

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

# Begründung der Anwendung der Chernoff-Ungleichungen

- Die Wahrscheinlichkeit, dass ein schlechtes Dreieck kreiert wird, beträgt  $p = 1/4$

# Begründung der Anwendung der Chernoff-Ungleichungen

- Die Wahrscheinlichkeit, dass ein schlechtes Dreieck kreiert wird, beträgt  $p = 1/4$
- Da es  $n/4$  Dreiecke gibt, wird das „Experiment“ aus der Chernoff-Ungleichung  $n/4$  mal durchgeführt.



# Begründung der Anwendung der Chernoff-Ungleichungen

- Die Wahrscheinlichkeit, dass ein schlechtes Dreieck kreiert wird, beträgt  $p = 1/4$
- Da es  $n/4$  Dreiecke gibt, wird das „Experiment“ aus der Chernoff-Ungleichung  $n/4$  mal durchgeführt.
- Es ist der Erwartungswert für die Dreiecke  $p \cdot n = \frac{1}{4} \cdot \frac{n}{4} = \frac{n}{16} = \Theta(n)$ . Dieser wird mit einer Wahrscheinlichkeit von  $1 - o(1)$  (wegen der Chernoff-Ungleichungen) angenommen!

$$\text{con}_C(x) = 1$$

- Sei  $v$  ein Knoten in  $C$ .  $v$  hat  $n/2 - 1$  mögliche Nachbarn

$$\text{con}_C(x) = 1$$

- Sei  $v$  ein Knoten in  $C$ .  $v$  hat  $n/2 - 1$  mögliche Nachbarn
- Laut Chernoff-Ungleichung:  $v$  mit mindestens  $n/6$  Knoten benachbart

$$\text{con}_C(x) = 1$$

- Sei  $v$  ein Knoten in  $C$ .  $v$  hat  $n/2 - 1$  mögliche Nachbarn
- Laut Chernoff-Ungleichung:  $v$  mit mindestens  $n/6$  Knoten benachbart
- Für jeden weiteren Knoten ist die Wahrscheinlichkeit, mit keinem dieser  $n/6$  Knoten verbunden zu sein:  $(1/2)^{n/6}$ . (d.h.  $o(1)$ )



## Definition von $k$ -Schritten

Sei nun ein  $k$ -Schritt ein Schritt, in dem  $k$  Dreieckskanten getauscht werden. Sei  $p_k$  die Wahrscheinlichkeit für einen  $k$ -Schritt.

- Es gilt dann für RLS:

$$p_1 = \Theta(n^{-1}), p_2 = \Theta(n^{-2}) \text{ und } p_k = 0 \quad \forall k \geq 3.$$

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

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

## Definition von $k$ -Schritten

Sei nun ein  $k$ -Schritt ein Schritt, in dem  $k$  Dreieckskanten getauscht werden. Sei  $p_k$  die Wahrscheinlichkeit für einen  $k$ -Schritt.

- Es gilt dann für RLS:

$$p_1 = \Theta(n^{-1}), p_2 = \Theta(n^{-2}) \text{ und } p_k = 0 \quad \forall k \geq 3.$$

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

# Wahrscheinlichkeit von $k$ -Schritten bei $(1+1)$ EA

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

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

## Behauptung 3.2

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

- $b(y) = \Theta(n)$
- $con_G(y) = 1$

produziert.



# Beweis

Laut Lemma 2.1 (Erwartete Zeit zum Finden eines zshg. Graphen) wissen wir, dass die Wahrscheinlichkeit, einen zshg. Graphen zu finden, hoch genug ist.

Es gilt nämlich wegen  $m = n^2/8 + n/2$ :

$$\mathcal{O}(m \log n) = \mathcal{O}((n^2/8 + n/2) \log n) = \mathcal{O}(n^2 \log n) = o(n^{5/2})$$

Es gibt also einen ersten Suchpunkt  $y$  mit  $con_G(y) = 1$ .

# 1- und 2-Schritte mit schlechten Dreiecken

Alle 2-Schritte können den  $b$ -Wert höchstens um  $\mathcal{O}(n^{1/2})$  verringern.  
Welche Möglichkeiten hat ein 1-step, ein schlechtes Dreieck zu zerstören?

# 1- und 2-Schritte mit schlechten Dreiecken

Alle 2-Schritte können den  $b$ -Wert höchstens um  $\mathcal{O}(n^{1/2})$  verringern.  
Welche Möglichkeiten hat ein 1-step, ein schlechtes Dreieck zu zerstören?

- 1 Kante eines schlechten Dreiecks entfernen bedeutet Erhöhung von  $con_G(y)$ , d.h. Erniedrigung des Wertes  $con_C$ , aber da  $con_C(y) = 1$ : nicht möglich!

# 1- und 2-Schritte mit schlechten Dreiecken

Alle 2-Schritte können den  $b$ -Wert höchstens um  $\mathcal{O}(n^{1/2})$  verringern.  
Welche Möglichkeiten hat ein 1-step, ein schlechtes Dreieck zu zerstören?

- 1 Kante eines schlechten Dreiecks entfernen bedeutet Erhöhung von  $con_G(y)$ , d.h. Erniedrigung des Wertes  $con_C$ , aber da  $con_C(y) = 1$ : nicht möglich!
- 2 Fehlende Kante des Dreiecks einfügen bedeutet  $con_C(y)$  verringern, d.h. Erhöhung des Gewichts des Graphen

$$b(y) = \Theta(n)$$

- Erhöhung des  $con_C$ -Werts muss notwendigerweise Erniedrigung des  $con_T$ -Werts um mindestens denselben Wert zur Folge haben

$$b(y) = \Theta(n)$$

- Erhöhung des  $con_C$ -Werts muss notwendigerweise Erniedrigung des  $con_T$ -Werts um mindestens denselben Wert zur Folge haben
- Für jeden 1-Schritt wird dann aber das Gewicht des Graphens erhöht und der Schritt würde von keiner der Fitness-Funktionen akzeptiert werden

$$b(y) = \Theta(n)$$

- Erhöhung des  $con_C$ -Werts muss notwendigerweise Erniedrigung des  $con_T$ -Werts um mindestens denselben Wert zur Folge haben
- Für jeden 1-Schritt wird dann aber das Gewicht des Graphens erhöht und der Schritt würde von keiner der Fitness-Funktionen akzeptiert werden
- Es werden nur schlechte Dreiecke durch  $\mathcal{O}(n^{1/2})$  2-Schritte hinzugefügt, d.h. insgesamt  $\Theta(n) + \mathcal{O}(n^{1/2}) = \Theta(n)$  schlechte Dreiecke im Suchpunkt  $y$



## Behauptung 3.3

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

- $b(z) = \Theta(n)$
- $con_G(z) = 1$
- $T(z)$  ein Baum

produziert.



# Beweis

- Wegen Behauptung 3.1 (erster Beweisschritt) wissen wir, dass  $d(z) = 0$ .

# Beweis

- Wegen Behauptung 3.1 (erster Beweisschritt) 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.

# Beweis

- Wegen Behauptung 3.1 (erster Beweisschritt) 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, d.h. nur noch gute und schlechte Dreiecke

# Beweis

- Wegen Behauptung 3.1 (erster Beweisschritt) 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, d.h. nur noch gute und schlechte Dreiecke
- Ein 1-Schritt kann nur dann akzeptiert werden, wenn er ein komplettes Dreieck in ein gutes oder schlechtes Dreieck verwandelt.

# Beweis

- Wegen Behauptung 3.1 (erster Beweisschritt) 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, d.h. nur noch gute und schlechte Dreiecke
- Ein 1-Schritt kann nur dann akzeptiert werden, wenn er ein komplettes Dreieck in ein gutes oder schlechtes Dreieck verwandelt.
- Erhöhung der Anzahl der kompletten Dreiecke nicht möglich. 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.

# Elimination der kompletten Dreiecke und Produktion des Dreiecks-Baums

Sei nun  $c(x) = l$ , dann ist die Wahrscheinlichkeit, den  $c$ -Wert zu verringern mindestens  $3l/(e \cdot m)$ . Wie in Lemma 2.1 folgt für die erwartete Zeit, bis alle kompletten Dreiecke eliminiert, sind  $\mathcal{O}(m \log n) = \mathcal{O}(n^2 \log n)$ .

# Elimination der kompletten Dreiecke und Produktion des Dreiecks-Baums

Sei nun  $c(x) = l$ , dann ist die Wahrscheinlichkeit, den  $c$ -Wert zu verringern mindestens  $3l/(e \cdot m)$ . Wie in Lemma 2.1 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 3.2 wissen wir, dass die Anzahl der Durchläufe ausreicht, um einen Baum zu finden.

# Elimination der kompletten Dreiecke und Produktion des Dreiecks-Baums

Sei nun  $c(x) = l$ , dann ist die Wahrscheinlichkeit, den  $c$ -Wert zu verringern mindestens  $3l/(e \cdot m)$ . Wie in Lemma 2.1 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 3.2 wissen wir, 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.





## Behauptung 3.4

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

- Wahrscheinlichkeit, dass ein schlechtes in ein gutes Dreieck umgewandelt wird, liegt bei  $\Theta(n^{-4})$ .

# Beweis

- 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 ( $\Theta(n^{-2})$ ), Wahrscheinlichkeit, dass die beiden Kanten gewählt werden ( $\Theta(n^{-1})$ ). Damit liegt die erwartete Zeit für das Umwandeln eines schlechten in ein gutes Dreieck bei  $\Theta(n^4)$ .

# Beweis

- 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 ( $\Theta(n^{-2})$ ), Wahrscheinlichkeit, dass die beiden Kanten gewählt werden ( $\Theta(n^{-1})$ ). Damit liegt die erwartete Zeit für das Umwandeln eines schlechten in ein gutes Dreieck bei  $\Theta(n^4)$ .
- Um das  $b$ -te schlechte Dreieck in ein gutes Dreieck umzuwandeln, wird die Zeit  $\Theta(\frac{n^4}{b})$  benötigt.

# Beweis

- 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 ( $\Theta(n^{-2})$ ), Wahrscheinlichkeit, dass die beiden Kanten gewählt werden ( $\Theta(n^{-1})$ ). Damit liegt die erwartete Zeit für das Umwandeln eines schlechten in ein gutes Dreieck bei  $\Theta(n^4)$ .
- Um das  $b$ -te schlechte Dreieck in ein gutes Dreieck umzuwandeln, wird die Zeit  $\Theta(\frac{n^4}{b})$  benötigt.
- Insgesamt ist die erwartete Zeit, um alle schlechten Dreiecke in gute Dreiecke umgewandelt sind bei:

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

# Veränderung des Mutations-Operators

- 1 RLS: falls ein 2-bit Flip stattfindet, wird zufällig je eine neue Kante eingefügt und eine vorhandene Kante gelöscht.

# Veränderung des Mutations-Operators

- 1 RLS: falls ein 2-bit Flip stattfindet, wird zufällig je eine neue Kante eingefügt und eine vorhandene Kante gelöscht.
- 2 (1+1) EA: beinhatet der Suchpunkt  $s$  insgesamt  $k$  Kanten, so wird jede Kante mit der Wahrscheinlichkeit  $1/k$  und jede nicht enthaltene Kante mit der Wahrscheinlichkeit  $1/(m - k)$  geflippt.

# Literatur



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



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