Strikte lokale Suche

Lokale Suche

Wir betrachten das allgemeine Minimierungsproblem

$$\min_{y} \quad f(x, y) \quad \text{so dass} \quad L(x, y).$$

Wir nehmen an, dass zu jeder Lösung y auch eine Nachbarschaft $\mathcal{N}(y)$ „benachbarter“ Lösungen gegeben ist.

Die strikte lokale Suche:

(1) Sei $y^{(0)}$ eine Lösung für die Eingabe x. Setze $i = 0$.

(2) Wiederhole solange, bis eine lokal optimale Lösung gefunden ist:

(2a) Bestimme einen Nachbarn $y \in \mathcal{N}(y^{(i)})$, so dass $f(x, y) < f(x, y^{(i)})$ und y eine Lösung ist.

(2b) Setze $y^{(i+1)} = y$ und $i = i + 1$.

Die k-Flip Nachbarschaft

Implementierung der lokalen Suche (1/2)

- Wie sind die **Nachbarschaften** $\mathcal{N}(y)$ zu definieren?
  - Wenn nur Elemente aus $\{0,1\}^n$ Lösungen sind, dann ist

$$\mathcal{N}_k(y) = \{y' \in \{0,1\}^n \quad | \quad y \text{ und } y' \text{ unterscheiden sich in in höchstens } k \text{ Positionen}\}$$

    die **k-Flip Nachbarschaft** der Lösung y.
  - Die k-Flip Nachbarschaft besteht aus $\binom{n}{k}$ Elementen und ist für große k zu groß. Deshalb wird häufig $k = 1$ oder $k = 2$ gewählt.
- Mit welcher **Anfangslösung** $y^{(0)}$ soll begonnen werden?
  - Neben der zufälligen Wahl einer Startlösung
  - werden auch Heuristiken eingesetzt, um mit einer guten Lösung $y^{(0)}$ zu beginnen.
    Achtung: $y^{(0)}$ darf nicht in der Nähe eines lokalen Minimums sein!

Implementierung der lokalen Suche (2/2)

- Mit welcher Nachbarlösung soll die Suche fortgesetzt werden?
  - Wenn die Nachbarschaft genügend klein ist, dann liegt die Wahl eines Nachbarn mit kleinstem Zielfunktionswert nahe.
  - Bei zu großen Nachbarschaften wählt man häufig benachbarte Lösungen mit Hilfe einer Heuristik, bzw. man wählt zufällig eine kleine Menge von Nachbarn.

- Die große Schwäche der lokalen Suche:

  Nur Abwärtsbewegungen sind erlaubt.

  Kurz über lang fällt eine Lösung in ein lokales Minimum mit großer Sogwirkung.
- **Auswege:**
  - Im **Metropolis Algorithmus** und in **Simulated Annealing** werden auch Aufwärtsbewegungen erlaubt.
  - Die **lokale Suche in variabler Tiefe** erlaubt ebenfalls Verschlechterungen.

Erfolgreiche lokale Suchalgorithmen

Matroide

Der Greedy-Algorithmus für Matroide ist ein lokaler Suchalgorithmus, der aber stets einen besten Nachbarn bestimmt.

Beispiel: Der Simplex Algorithmus

- Der Lösungsraum des linearen Programmierproblems

$$\min c^T \cdot y, \text{ so dass } A \cdot y \geqslant b \text{ und } y \geqslant 0$$

ist ein Durchschnitt von Halbräumen der Form $\{y \mid \alpha^T \cdot y \geqslant \beta\}$.

Deshalb wird das Optimum an einer „Ecke“ des Lösungsraums angenommen.

- Der **Simplex-Algorithmus** wandert solange von einer Ecke zu einer besseren, benachbarten Ecke, bis keine Verbesserung erreichbar ist.

Der Simplex-Algorithmus führt eine lokale Suche auf der Menge der Ecken durch.

Beispiel: Minimierung reellwertiger Funktionen

- Die Zielfunktion $f : \mathbb{R}^n \to \mathbb{R}$ des Minimierungsproblems $P = (\min, f, L)$ sei differenzierbar.
- Der Lösungsraum sei eine kompakte Teilmenge des $\mathbb{R}^n$.
- Wenn wir uns gegenwärtig in der Lösung $a \in \mathbb{R}^n$ befinden:
  - In welcher Richtung sollten wir nach besseren Lösungen suchen?
  - In der Richtung des negativen Gradienten! Ersetze a durch

$$a - \eta \cdot \nabla f(a)$$

für ein hinreichend kleines η.

Die Methode des Gradientenabstiegs

Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine zweimal im Punkt $a \in \mathbb{R}^n$ stetig differenzierbare Funktion mit $\nabla f(a) \neq 0$. Dann gibt es ein $\eta > 0$ mit

$$f\left(a - \eta \cdot \nabla f(a)\right) < f(a),$$

wobei $\nabla f(a)$ der Gradient von f an der Stelle a ist.

Warum funktioniert der Gradientenabstieg?

- Approximiere die Funktion f durch ihr lineares Taylor-Polynom:

$$f(a+z) = f(a) + \nabla f(a)^T \cdot z + O(z^T \cdot z).$$

  gilt für hinreichend kleines z.
- Für $z = -\eta \cdot \nabla f(a)$ bei hinreichend kleinem $\eta > 0$ ist

$$f(a - \eta \cdot \nabla f(a)) = f(a) - \eta \cdot \ ||\nabla f(a)||^2 + \eta^2 \cdot O(||\nabla f(a)||^2).$$

- Für hinreichend kleines $\eta < 1$ wird somit $f(a - \eta \cdot \nabla f(a))$ kleiner als $f(a)$ sein.

Die Gefahr schlechter lokaler Optima

- Wir betrachten das **Vertex Cover** Problem für einen Graphen $G = (V, E)$.
  - Die Nachbarschaft $\mathcal{N}(U)$ einer Teilmenge $U \subseteq V$ besteht aus allen Überdeckungen, die durch Hinzufügen oder Entfernen eines Elementes aus der Überdeckung U entstehen.
- Wir beginnen die lokale Suche mit der Lösung $U = V$ und betrachten einige Beispiele.
  - **Der Graph ohne Kanten:** Die lokale Suche entfernt in jedem Schritt einen Knoten und findet die leere Menge.
  - **Der Sterngraph:**
    - ⋆ Wird im ersten Schritt das Sternzentrum entfernt, dann wird das schlechte lokale Minimum der „Satelliten“ gefunden.
    - ⋆ Ansonsten führt lokale Suche zwangsläufig auf das Sternzentrum.
  - **Der Weg 0** $\rightarrow$ **1** $\rightarrow$ **2** $\rightarrow \cdots \rightarrow$ **n** $-$ **2** $\rightarrow$ **n** $-$ **1**: Für gerades n ist $\{0, 2, 4, \ldots, n-2\}$ optimal. Wie sehen die lokalen Minima aus?

Die Klasse PLS:
Wann lassen sich lokale Minima effizient bestimmen?

Die Klasse PLS

Die Klasse PLS aller **polynomiellen Suchprobleme** besteht aus allen NP-Optimierungsproblemen (opt, f, L) mit:

- Die Funktion f nimmt nur ganzzahlige Werte an.
- Es gibt einen effizienten Algorithmus A, der für jede Instanz x_0 eine Anfangslösung y_0 berechnet.
- Es gibt einen effizienten Algorithmus B, der für jede Instanz x_0 und jede Lösung y mit $L(x_0, y)$ entscheidet, ob y ein lokales Minimum ist.

  Ist dies nicht der Fall, dann bestimmt B einen besseren Nachbarn.

Wir beschreiben das Suchproblem durch (opt, f, L, A, B).

Lokale Minima lassen sich doch schnell bestimmen?!

Für jedes polynomielle Suchproblem $(\mathrm{opt}, f, L, A, B)$ führe den folgenden **Standardalgorithmus** aus:

(1) Berechne $y = A(x_0)$.

(2) Solange y kein lokales Optimum ist, setze $y = B(x_0, y)$.

- Wenn f nur polynomiell w viele Werte annimmt, dann terminiert das Suchverfahren nach höchstens w Schritten.
- Aber was passiert, wenn f extrem viele Werte annehmen kann?

  Ist es in diesem Fall überhaupt möglich, **irgendein** lokales Optimum effizient zu bestimmen?

Berechne ein bestimmtes oder irgendein lokales Optimum?

In den **Übungen**:

- Wenn der Standardalgorithmus für alle Probleme in PLS effizient simulierbar ist, dann ist das Erfüllbarkeitsproblem effizient lösbar.
  - Die Bestimmung des vom Standardalgorithmus berechneten lokalen Optimums ist im Allgemeinen „NP-hart“.
  - Der Standard Algorithmus lässt sich nicht effizient simulieren.
- Angenommen, wir dürfen einen Algorithmus kostenfrei anwenden, der **irgendein** lokales Optimum bestimmt. Wenn das Erfüllbarkeitsproblem effizient lösbar ist, dann gilt $\mathrm{NP} = \mathrm{coNP}$.
  - Die Bestimmung irgendeines lokalen Optimums ist hochwahrscheinlich nicht „NP-hart“.

Und nun?

Wie schwierig ist die Bestimmung **irgendeines** lokalen Optimums?

- Gibt es schwierigste polynomielle Suchprobleme?
- Wir führen einen Reduktionsbegriff ein.

PLS-vollständige Probleme

Schwierigste polynomielle Suchprobleme

- $P_1 = (\text{opt}_1, f_1, L_1, A_1, B_1)$ und $P_2 = (\text{opt}_2, f_2, L_2, A_2, B_2)$ sind polynomielle Suchprobleme.
- P_1 ist auf P_2 PLS-reduzierbar ($P_1 \leqslant_{\text{PLS}} P_2$), wenn es effiziente Transformationen Φ und Ψ gibt mit:
  (a) Wenn x eine Instanz von P_1 ist, dann ist $\Phi(x)$ eine Instanz von P_2.
  (b) Wenn y ein lokales Optimum für P_2 und Instanz $\Phi(x)$ ist, dann ist $\Psi(x, y)$ ein lokales Optimum für P_1 und Instanz x.
- P heißt **PLS-vollständig**, wenn P ein polynomielles Suchproblem ist und wenn $Q \leqslant_{\text{PLS}} P$ für jedes polynomielle Suchproblem Q gilt.

P sei PLS-vollständig.

Wenn irgendein lokales Optimum für jede Instanz von P effizient berechnet werden kann, dann lassen sich (irgendwelche) lokale Optima für alle polynomiellen Suchprobleme effizient bestimmen.

MINIMUM BALANCED CUT

Für einen ungerichteten Graphen $G = (V, E)$ mit gewichteten Kanten bestimme eine Knotenmenge $W \subseteq V$ mit $|W| = \frac{1}{2} \cdot |V|$, so dass das Gewicht

$$f(W) = \sum_{e \in E, |\{e\} \cap W| = 1} \text{gewicht}(e)$$

kreuzender Kanten minimal ist.

- Eine Anwendung: Der Entwurf von VLSI-Layouts.
  - Bestimme rekursiv Layouts für $(W, \{e \in E \mid e \subseteq W\})$ und $(V \setminus W, \{e \in E \mid e \subseteq V \setminus W\})$.
  - Wenige kreuzende Kanten erlauben eine platz-sparende Kombination der beiden Layouts.
- Wähle die 2-Flip Nachbarschaft: Die Nachbarschaft $N(W)$ ist also

$$N(W) = \{ U \subseteq V \mid |U| = \frac{|V|}{2} \text{ und } |U \oplus W| = 2 \}.$$

PLS-vollständige Probleme (ohne Beweis)

Die folgenden polynomiellen Suchprobleme sind PLS-vollständig:

(a) Die Bestimmung eines lokalen Minimums für TSP, wenn eine Rundreise mit allen Rundreisen benachbart ist, die sich durch einen Austausch von höchstens k Kanten ergeben. (Die Konstante k muss hinreichend groß gewählt werden.)

(b) Die Bestimmung eines lokalen Minimums für MINIMUM BALANCED CUT, wenn zwei Knotenmengen hälftiger Größe genau dann benachbart sind, wenn sie durch die Ersetzung eines Knotens auseinander hervorgehen.

(c) Die Bestimmung eines Nash-Gleichgewichts für Congestion Spiele.

Approximative lokale Optima

Was bedeutet PLS-Vollständigkeit für uns?

- Nichts, wenn wir wissen, dass das vorliegende Problem nur polynomiell viele verschiedene Werte annimmt.
- Bei vielen verschiedenen Werten müssen wir vorsichtig sein: Wir sollten möglicherweise approximative lokale Optima berechnen!

- $(\min, f, L, A, B)$ sei ein polynomielles Suchproblem und y sei eine Lösung für die Instanz x.
- Dann ist y ein ε-approximatives lokales Minimum, wenn

$$f(x, y) \leqslant (1 + \varepsilon) \cdot f(x, y')$$

für alle benachbarten Lösungen $y' \in N_x(y)$ gilt.

Die Berechnung approximativer lokaler Minima (1/2)

Gibt es stets ein volles polynomielles Approximationsschema für die Berechnung eines ε-approximativen lokalen Optimums?

- Betrachte **lineare kombinatorische Optimierungsprobleme**:
  - Ein Universum U und eine Gewichtung $w_u \in \mathbb{N}$ für alle Elemente $u \in U$ des Universums ist gegeben ebenso wie eine Klasse $\mathcal{Q}$ von Teilmengen von U.
  - Bestimme eine leichteste Teilmenge $T \in \mathcal{Q}$, wobei

$$\text{gewicht}(T) = \sum_{u \in T} w_u$$

  das Gewicht von T ist.
- Fast alle bisher betrachteten Optimierungsprobleme lassen sich als lineare, kombinatorische Optimierungsprobleme auffassen.

Die Berechnung approximativer lokaler Minima (2/2)

Ein erster Ansatz

(1) Wenn wir gerade die Lösung T erreicht haben:

- Skaliere die Gewichte: Ersetze w_u durch $\lceil \frac{w_u}{q} \rceil$ mit

$$q = \frac{\varepsilon}{1+\varepsilon} \cdot \frac{\text{gewicht}(T)}{2|U|}.$$

- $\frac{w_u}{q} \leqslant \frac{1+\varepsilon}{\varepsilon} \cdot \frac{2|U| \cdot w_u}{\text{gewicht}(T)} \leqslant \frac{1+\varepsilon}{\varepsilon} \cdot 2|U|.$
- Es werden nur „wenige“ Funktionswerte angenommen.

(2) Bestimme ein lokales Optimum T' für die neuen Gewichte.

Wenn T'' ein beliebiger Nachbar von T' ist:

$$\begin{aligned} \text{gewicht}(T') &= \sum_{u \in T'} w_u \leqslant \sum_{u \in T'} \lceil \frac{w_u}{q} \rceil \cdot q \leqslant \sum_{u \in T''} \lceil \frac{w_u}{q} \rceil \cdot q \\ &\leqslant \sum_{u \in T''} (\frac{w_u}{q} + 1) \cdot q \leqslant \text{gewicht}(T'') + |U| \cdot q. \end{aligned}$$

Klappt das?

- Wenn das Gewicht der Mengen während der lokalen Suche zu stark sinkt, dann adjustiere den Skalierungsfaktor!
- Also, wenn

$$\text{gewicht}(T') < \text{gewicht}(T)/2$$

während der lokalen Suche, dann
setze $T = T'$, gehe zu Schritt (1) und adjustiere q.

- Es ist $\text{gewicht}(T') \leqslant \text{gewicht}(T'') + |U| \cdot q$
- und am Ende der Berechnung gilt $\text{gewicht}(T') \geqslant \text{gewicht}(T)/2$.

$$\begin{aligned}\frac{\text{gewicht}(T') - \text{gewicht}(T'')}{\text{gewicht}(T'')} &\leqslant \frac{|U| \cdot q}{\text{gewicht}(T'')} \leqslant \frac{|U| \cdot q}{\text{gewicht}(T') - |U| \cdot q} \\ &\leqslant \frac{|U| \cdot q}{\text{gewicht}(T)/2 - |U| \cdot q} = \varepsilon\end{aligned}$$

und alles in Butter.

Fazit

Für jedes lineare kombinatorische Optimierungsproblem kann ein ε-approximatives lokales Optimum in $\frac{|U|}{\varepsilon} \cdot \log_2(\text{gewicht}(T_0))$ Schritten bestimmt werden, wenn T_0 die Anfangslösung ist.

- Warum ist die Schranke für die Schrittzahl richtig?
  - Höchstens $\log_2(\text{gewicht}(T_0))$-mal kann das Lösungsgewicht halbiert werden.
  - Zwischen zwei Halbierungen liegen höchstens $O(n \cdot \frac{|U|}{\varepsilon})$ Schritte der lokalen Suche.
- Die Berechnung lokaler Optima ist möglicherweise schwierig, aber

  wir haben ein volles polynomielles Apprximationsschema erhalten, wenn wir ε-approximative lokale Optima berechnen wollen.

FACILITY LOCATION

FACILITY LOCATION

- Eine Menge K von **Kunden**, eine Menge S möglicher **Service Stationen** sowie eine Metrik d auf $K \cup S$ ist gegeben:
  - $\mathbf{d(k, s)}$ ist die Distanz des Kunden $k \in K$ von Service Station $s \in S$.
- Die **Betriebskosten** der Service Station $s \in S$ betragen f_s.
- Definiere die **Anschlusskosten** von Kunde k für eine Menge $X \subseteq S$ von Service Stationen durch $\text{distanz}_X(k) = \min_{s \in X} d(k, s)$.
- Wir suchen eine Teilmenge $X \subseteq S$, so dass

$$C(X) = \sum_{k \in K} \text{distanz}_X(k) + \sum_{s \in X} f_s,$$

die Summe aller Betriebs- und Anschlusskosten, minimal ist.

Lokale Suche für FACILITY LOCATION

(1) Wir beginnen mit einer beliebigen Menge $X \subseteq S$ von Service Stationen.

(2) Wenn das **Entfernen**, **Hinzufügen** oder die **Ersetzung** von Service Stationen zu einer Verbesserung führt, dann wird eine beliebige solche Operation ausgeführt.

(3) Wiederhole Schritt (2) solange möglich.

Wann sollten wir terminieren? Das Erreichen eines lokalen Minimums dauert vielleicht zu lange.
Halte, wenn der erzielte Fortschritt zu gering ist.

Der Approximationsfaktor ist 3 oder schlechter

- Betrachte die Mengen $K = \{k_1, \ldots, k_n\}$ von n Kunden und $S = \{s_0, \ldots, s_n\}$ von $n+1$ Service Stationen.
  s_0 hat Betriebskosten $2n-2$, s_{i+1} hat keine Betriebskosten.
- Die Metrik d ist die Länge kürzester Wege in dem folgenden ungerichteten Graphen G mit Knotenmenge $K \cup S$.
  - G besitzt nur Kanten der Länge 1, nämlich zuerst die Kanten $\{k_i, s_i\}$ und schließlich die Kanten $\{k_i, s_0\}$ für $1 \leqslant i \leqslant n$.
  - Kunde k_i hat Distanz 1 von den Service Stationen s_0 und s_i und damit Distanz 3 zu jeder anderen Service Station.

- $X^* = \{s_1, \ldots, s_n\}$ ist ein globales Minimum mit $C(X^*) = n$.
- $X = \{s_0\}$ ist ein lokales Minimum mit $C(X) = 2n-2+n = 3n-2$.
  - Wir dürfen s_0 aus X nicht entfernen.
  - Die Hinzunahme einer weiteren Service Station senkt die Gesamtkosten nicht.
  - Ersetze s_0 durch s_i: Betriebskosten sinken von $2n-2$ auf Null, Anschlusskosten steigen von n auf $1+3(n-1) = 3n-2$.

Das Ergebnis

Sei X^* ein globales Minimum und X ein lokales Minimum.

Dann gilt

$$C(X) \leqslant 3 \cdot C(X^*).$$

- Wir zeigen nur $C(X) \leqslant 5 \cdot C(X^*)$.
  - Das Argument schätzt Anschluß- und Betriebskosten des lokalen Optimums X gegen die Kosten des globalen Minimums X^* ab:
    - $\sum_{k \in K} \text{distanz}_X(k) \leqslant C(X^*)$,
    - und $\sum_{s \in X} f_x \leqslant 2 \cdot \left[C(X^*) + \sum_{k \in K} \text{distanz}_X(k) \right] \leqslant 4 \cdot C(X^*)$.
- Wir beginnen mit der Abschätzung der Anschlusskosten.

  Wenn $\text{distanz}_X(k) = d(k, s)$, dann ist $\text{naechste}_X(k) = s$ die günstigste Service Station für den Kunden k.

Eine Abschätzung der Anschlusskosten

- Sei $s \in X^* \setminus X$.
  - Wenn wir s zu X **hinzufügen**, dann ist $C(X) \leqslant C(X \cup \{s\})$, denn X ist ein lokales Minimum.
  - Also folgt

$$f_s + \sum_{k,\text{naechste}_{X^*}(k)=s} \text{distanz}_{X^*}(k) \geqslant \sum_{k,\text{naechste}_{X^*}(k)=s} \text{distanz}_X(k).$$

- Nach Summation über alle Service Stationen $s \in X^* \setminus X$, wenn gemeinsame Kosten berücksichtigt werden

$$C(X^*) \geqslant \sum_{s \in X^* \setminus X} f_s + \sum_{k \in K} \text{distanz}_{X^*}(k) \geqslant \sum_{k \in K} \text{distanz}_X(k)$$

und das war zu zeigen.

Die Abschätzung der Betriebskosten (1/3)

- **Ersetze** eine Service Station $s \in X$ durch eine **nächstliegende** Service Station $s^* \in X^*$. Da X ein lokales Minimum ist, gilt

$$f_s \leqslant f_{s*} + \sum_{k,\text{naechste}_X(k)=s} [\, d(k,s^*) - d(k,s) \,].$$

- Wie groß ist der Unterschied $d(k,s^*) - d(k,s)$?
  - Mit der Dreiecksungleichung folgt $d(k,s^*) - d(k,s) \leqslant d(s,s^*)$.
  - Service Station s^* ist die zu s nächstliegende Service Station in X^*. Also ist $d(s,s^*) \leqslant d(s,\text{naechste}_{X^*}(k))$.
  - Schiebe k zwischen s und $\text{naechste}_{X^*}(k)$: $d(k,s^*) - d(k,s)$

$$\begin{aligned} &\leqslant d(s,\text{naechste}_{X^*}(k)) \leqslant d(s,k) + d(k,\text{naechste}_{X^*}(k)) \\ &= \text{distanz}_X(k) + \text{distanz}_{X^*}(k). \end{aligned}$$

- Also

$$f_s \leqslant f_{s*} + \sum_{k,\text{naechste}_X(k)=s} [\, \text{distanz}_X(k) + \text{distanz}_{X^*}(k) \,]$$

Die Abschätzung der Betriebskosten (2/3)

Wir wissen: $f_s \leqslant f_{s*} + \sum_{k,\text{naechste}_X(k)=s} [\ \text{distanz}_X(k) + \text{distanz}_{X^*}(k)\ ]$.

- Wenn jede Service Station in X^* höchstens einmal als eine nächstliegende Service Station auftritt:

$$\begin{aligned} \sum_{s\in X} f_s &\leqslant \sum_{k\in K} [\ \text{distanz}_X(k) + \text{distanz}_{X^*}(k)] + \sum_{s^*\in X^*} f_{s^*} \\ &= \sum_{k\in K} \text{distanz}_X(k) + C(X^*) \leqslant 2C(X^*), \end{aligned}$$

  und sogar $C(X) \leqslant 3C(X^*)$ folgt.
- Wenn aber Service Station $s^* \in X^*$ nächstliegend zu mehreren Stationen $s \in X$ ist?

Primäre und sekundäre Service Stationen

Für s in X sei $s^* \in X^*$ die nächstliegende Service Station in X^*.

s ist eine primäre Service Station, wenn s die nächstliegende Service Station von s^* in X ist und sonst ist s sekundär.

Sekundäre Service Stationen

$s \in X$ sei eine sekundäre Service Station und $s^* \in X^*$ sei die zu s nächstliegende Service Station in X^*.

- Eine Station $s' \in X$ ($s \neq s'$) hat einen kleineren Abstand von s^*.
- Entferne s und weise alle, bisher s zugewiesenen Kunden der Service Station s' zu.
  - Wir sparen die Betriebskosten von s.
  - Um wieviel steigen die Anschlusskosten, wenn k vorher s zugewiesen wurde?

$$\begin{aligned} d(k,s') - d(k,s) &\leqslant d(s,s^*) + d(s^*,s') \leqslant 2 \cdot d(s,s^*) \\ &\leqslant 2 \cdot d(s, \text{naechste}_{X^*}(k)) \\ &\leqslant 2 \cdot [d(s,k) + d(k, \text{naechste}_{X^*}(k))] \\ &= 2 \cdot [\text{distanz}_X(k) + \text{distanz}_{X^*}(k)] \,. \end{aligned}$$

Die Abschätzung der Betriebskosten (3/3)

Wenn Kunde k vorher der Station s zugewiesen wurde:
$d(k, s') - d(k, s) \leqslant 2 \cdot [\text{distanz}_X(k) + \text{distanz}_{X^*}(k)]$.

- X ist ein lokales Minimum:

$$f_s \leqslant \sum_{k,\text{naechste}_X(k)=s} 2 \cdot [\text{distanz}_X(k) + \text{distanz}_{X^*}(k)] \,.$$

- Nach Unterscheidung primärer und sekundärer Stationen:

$$\begin{aligned} \sum_{s \in X} f_s &\leqslant \sum_{s^* \in X^*} f_{s^*} + 2 \sum_{k \in K} [\text{distanz}_X(k) + \text{distanz}_{X^*}(k)] \\ &\leqslant 2 \left[C(X^*) + \sum_{k \in K} \text{distanz}_X(k) \right] \\ &\leqslant 4C(X^*) \end{aligned}$$

und das war zu zeigen.

Und die Laufzeit?

- Die **schlechten** Nachrichten: Die Laufzeit kann superpolynomiell werden.
- Die **guten** Nachrichten: Brich ab, wenn nur noch eine kleine Verbesserung erzielt werden kann.

- Akzeptiere nur Verbesserungen, die den Wert der aktuellen Lösung um mindestens $\Delta = 1 - \frac{\varepsilon}{|S|}$ reduzieren.
- Wir erhalten eine $(3 + \varepsilon)$-approximative Lösung und es werden höchstens $\log_{1/\Delta} Q$ Verbesserungsschritte durchgeführt.
  - ⋆ Q ist der Quotient zwischen dem Wert einer schlechtesten und besten Lösung.
  - ⋆ Benutze unsere obige Analyse.

k-Median

k-MEDIAN

- Eine Menge K von n Punkten ist gegeben sowie eine Metrik d.
- Wir suchen eine Menge M von höchstens k Punkten aus K, so dass

$$\sum_{x \in K} \min_{y \in M} d(x, y)$$

minimal ist.

- k-MEDIAN ist fast ein Spezialfall von FACILITY LOCATION:

  Erlaube höchstens k Service Stationen und nimm an, dass keine Betriebskosten entstehen.
- Auch diesmal funktioniert die lokale Suche.

Lokale Suche für k-MEDIAN

(1) Beginne mit einer beliebigen Auswahl von k Punkten.

(2) Wiederhole solange möglich:

> Ersetze einen Punkt der gegenwärtigen Auswahl durch einen unbenutzten Punkt, wenn dies zu einer Verbesserung führt.

Ohne Beweis:

Der Wert eines lokalen Optimums übertrifft den Wert eines globalen Optimums um höchstens den Faktor fünf.

k-Center

k-CENTER

- Eine Menge V von Punkten ist gegeben.
- Auf der Punktmenge V ist eine Metrik $d : V \times V \to \mathbb{R}$ gegeben.
- Gesucht ist eine Teilmenge $C \subseteq V$ von k „Zentren“, so dass

$$D = \max_{v \in V} \min_{c \in C} d(c, v)$$

kleinstmöglich ist.

- **Übungsaufgabe:**
  k-CENTER hat effiziente 2-approximative Algorithmen.
  (1) Zu Anfang sind alle Punkte in V unmarkiert.
  (2) Wiederhole, solange es unmarkierte Punkte in C gibt:

  Füge einen beliebigen unmarkierten Punkt $v \in V$ zu C hinzu und markiere alle Punkte $w \in V$ mit Abstand $d(v, w) \leqslant 2D$.

  Kommentar: Wenn $D \geqslant \mathrm{opt}$, dann ist $|C| \leqslant k$.

Der Approximationsfaktor 2 ist optimal

Wenn P $\neq$ NP,
dann gibt es keinen δ-Approximationsalgorithmus für $\delta < 2$.

- DOMINATING SET: Für einen ungerichteten Graphen $G = (V, E)$ und Knotengewichte $w_u \geqslant u$ bestimme eine leichteste („dominierende“) Teilmenge $V' \subseteq V$, so dass
  jeder Knoten $v \notin V'$ mindestens einen Nachbarn in V' hat.
- DOMINATING SET ist NP-vollständig.
- Reduziere DOMINATING SET auf k-CENTER:
  - Sei $G = (V, E)$ und k eine Instanz von DOMINATING SET.
  - Für k-CENTER wähle V als Punktemenge mit der Distanz
    $$d(u, v) = \begin{cases} 1 & \{u, v\} \in E, \\ 2 & \text{sonst.} \end{cases}$$
  - Ist V' eine dominierende Menge der Größe k in G, dann
    - ⋆ wähle V' als Menge der Zentren und die Maximaldistanz ist eins.
  - Besitzt G hingegen keine dominierende Mengen der Größe k:
    - ⋆ Die Maximaldistanz ist zwei.

MAX LEAF SPANNING TREE und MIN DEGREE SPANNING TREE

Spannbaumprobleme und ihre Nachbarschaft

Für MAX LEAF SPANNING TREE

- ist ein ungerichteter, zusammenhängender Graph G gegeben.
- Gesucht ist ein Spannbaum mit der größtmöglichen Blattzahl.

Für MIN DEGREE SPANNING TREE

- ist ein ungerichteter, zusammenhängender Graph G gegeben.
- Gesucht ist ein Spannbaum, so dass der maximale Knotengrad möglichst klein ist.

Die Nachbarschaft

- Fügen wir eine neue Kante e zu einem Spannbaum B hinzu, dann wird genau ein Kreis geschlossen. Entfernen wir eine Kante e' des Kreises, dann ist auch $B' = B \setminus \{e'\} \cup \{e\}$ ein Spannbaum.
- Wir sagen, dass B' ein Nachbar von B ist.

MAX LEAF SPANNING TREE: Der Algorithmus

(1) Sei B ein beliebiger Spannbaum für den ungerichteten, zusammenhängenden Graphen $G = (V, E)$.

(2) Solange B einen Nachbarn B' mit mehr Blättern besitzt, ersetze B durch B'.

(3) Gib B aus.

- Da die Zahl der Blätter in jeder Iteration ansteigt, gibt es höchstens $|V|$ Iterationen.
- Wir zeigen, dass unser Algorithmus 10-approximativ ist.

  Es kann sogar gezeigt werden, dass der Approximationsfaktor durch 5 beschränkt ist.

- $k_d(B)$ ist die Anzahl der Knoten vom Grad d im Baum B.
- $k_1(B)$ ist die Anzahl der Blätter von B.
- **Behauptung**: $\sum_{d \geqslant 3} k_d(B) < k_1(B)$.

- Ein Baum mit n Knoten beistzt genau $n-1$ Kanten. Also ist

$$\begin{aligned} 2 \cdot (n-1) &= \sum_d d \cdot k_d(B) = k_1(B) + 2 \cdot k_2(B) + \sum_{d \geqslant 3} d \cdot k_d(B) \\ &\geqslant k_1(B) + 2 \cdot k_2(B) + 3 \cdot \sum_{d \geqslant 3} k_d(B). \end{aligned}$$

- Es ist $n = k_1(B) + k_2(B) + \sum_{d \geqslant 3} k_d(B)$ und deshalb

$$\begin{aligned} k_1(B) + 3 \cdot \sum_{d \geqslant 3} k_d(B) &\leqslant 2 \cdot (n - k_2(B) - 1) \\ = \quad 2 \cdot (k_1(B) + \sum_{d \geqslant 3} k_d(B) - 1). \end{aligned}$$

Analyse (2/3)

Die Anzahl der Blätter ist größer als die Anzahl der Knoten mit mindestens drei Nachbarn.

Jeder Spannbaum ohne Knoten vom Grad zwei ist 2-approximativ.

- Sei B_{opt} ein Baum mit größtmöglicher Blattzahl und sei B der von Algorithmus bestimmte Baum.
- Ein maximaler 2-Weg durchläuft nur Knoten vom Grad zwei als innere Knoten und endet in Knoten vom Grad ungleich zwei.
  - Maximale 2-Wege zerlegen die Grad-2 Knoten von B in disjunkte Klassen. Wieviele Klassen gibt es?
  - Maximale 2-Wege enden in Knoten vom Grad ungleich zwei: Es gibt $\leqslant k_1(B) + \sum_{d \geqslant 3} k_d(B) \leqslant 2 \cdot k_1(B)$ viele maximale 2-Wege.

Behauptung: Jeder maximale 2-Weg in B durchläuft höchstens vier Blätter von B_{opt} als innere Knoten.

Dann hat $B_{\text{opt}} \leqslant k_1(B) + 8 \cdot k_1(B) + k_1(B) = 10 \cdot k_1(B)$ Blätter.

Analyse (3/3)

Zeige, dass jeder maximale 2-Weg in B höchstens vier Blätter von B_{opt} als innere Knoten durchläuft.

- Der maximale 2-Weg W durchlaufe die fünf Blätter v_1, v_2, v, v_2', v_1' von B_{opt} in dieser Reihenfolge.
- Sei v' ein Knoten, der nicht von W durchlaufen wird.
  - In B_{opt} gibt es genau einen Weg W' von v nach v'.
  - W' beginnt in v, durchläuft Knoten von W und springt von einem Knoten w zum ersten nicht von W durchlaufenen Knoten u.
  - W' durchläuft weder den Knoten v_2' noch den Knoten v_2:

    Beide Knoten sind Blätter in B_{opt}.
- Füge die Kante $\{u, w\}$ zu B hinzu.
  - Die Hinzunahme erzeugt einen Kreis in B, der entweder die Knoten v_1', v_2' oder die Knoten v_2, v_1 durchläuft.
  - Wir verlieren höchstens das Blatt u, gewinnen aber durch das Entfernen einer mit v_2', bzw. v_2 inzidenten Kante zwei neue Blätter:

    B ist kein lokales Optimum.

MIN-DEGREE SPANNING TREE

MIN-DEGREE SPANNING TREE

Für einen ungerichteten Graphen G

bestimme einen Spannbaum mit kleinstmöglichem Grad.

- Das Sprachenproblem

  Hat G einen Spannbaum vom Grad höchstens d

  ist NP-vollständig:
  - G hat genau dann einen Spannbaum mit maximalem Grad zwei, wenn G einen Hamiltonpfad besitzt, also einen Weg, der alle Knoten genau einmal besucht.
  - Die Frage, ob ein ungericheter Graph einen Hamiltonpfad besitzt, ist NP-vollständig.
- Um wieviel unterscheiden sich lokale und globale Minima?
  - Es gibt Graphen mit einem Spannbaum vom Grad drei, aber lokalen Minima mit unbeschränkt großem Grad!
  - Können wir diese lokalen Minima zerstören?

Mehr Verbesserungsmöglichkeiten

Sei B' ein Nachbar des Spannbaums B, es gelte also

$$B' = B \setminus \{u_1, u_2\} \cup \{u_3, u_4\}.$$

Wir sagen, dass B' **etwas-besser** als B ist, wenn die B-Grade von u_1, u_2 größer als die B'-Grade von u_3, u_4 sind:

$$\max\{\mathrm{grad}_B(u_1), \mathrm{grad}_B(u_2)\} > \max\{\mathrm{grad}_B(u_3), \mathrm{grad}_B(u_4)\} + 1.$$

- Einige der alten lokalen Minima werden zerstört, weil Nachbarn „plötzlich“ verbessernd sind.
- Aber es gibt jetzt etwas-bessere Nachbarn, deren maximaler Grad nicht kleiner ist.
  - Läuft dann die lokale Suche möglicherweise im Kreis?
  - Nein, die Gradfolge von B, wenn absteigend geordnet, ist lexikographisch größer als die Gradfolge von B'.

Lokale Suche für MIN DEGREE SPANNING TREE

(1) Sei B ein beliebiger Spannbaum für den ungerichteten, zusammenhängenden Graphen $G = (V, E)$.

(2) Solange B einen etwas-besseren Nachbarn $B' = B \setminus \{u_1, u_2\} \cup \{u_3, u_4\}$ besitzt und

$$\max\{\operatorname{grad}_B(u_i) \mid 1 \leqslant i \leqslant 4\} \geqslant \max_v \operatorname{grad}_B(v) - \log_2 |V|$$

gilt, ersetze B durch B'.

Kommentar:

- Wir verlangen, dass der maximale Grad aller beteiligten Knoten groß ist.
- Übungsaufgabe: Die Laufzeit ist polynomiell in $|V|$.

(3) Gib B aus.

MIN DEGREE SPANNING TREE: Die Analyse (1/4)

(X, Π) ist ein Zeuge, wenn

- $X \subseteq V$ eine Knotenmenge und Π eine Zerlegung von V ist
- und alle Kanten, die verschiedene Klassen von Π verbinden, mit mindestens einem Knoten aus X inzident sind.

Die Anzahl der Klassen der Zerlegung Π bezeichnen wir mit $\mathrm{rank}(\Pi)$.

- Sei B ein beliebiger Spannbaum. Wieviele Kanten sind in B mit einem Knoten aus X inzident?
  - Entferne alle mit Knoten aus X inzidenten Kanten aus B.
  - Wir erhalten mindestens $\mathrm{rank}(\Pi)$ Zusammenhangskomponenten.
- B hat mindestens $\mathrm{rank}(\Pi) - 1$ mit Knoten aus X inzidente Kanten.

Für jeden Zeugen (X, Π) gilt

$$\text{kleinstmöglicher Grad} \geqslant \frac{\mathrm{rank}(\Pi) - 1}{|X|}.$$

MIN DEGREE SPANNING TREE: Die Analyse (2/4)

Wir suchen einem Zeugen (X, Π) für den $\frac{\text{rank}(\Pi)-1}{|X|}$ groß ist.

- X sollte möglichst klein sein und
- der Grad der Knoten in X sollte möglichst groß sein, so dass die Herausnahme viele Zusammenhangskomponenten erzeugt.

- Sei B der vom Algorithmus berechnete Spannbaum.
- Betrachte die Mengen $X_d = \{v \in V \mid \text{grad}_B(v) \geqslant d\}$.
  - Es ist $X_d \subseteq X_{d-1} \subseteq \cdots \subseteq X_{d-\log_2 n}$ für den maximalen Grad d von B.
  - Wenn stets $|X_{d-i-1}| > 2 \cdot |X_{d-i}|$: $|X_{d-\log_2 n}| > 2^{\log_2 n} \cdot |X_d| \geqslant n$.
  - Dies ist unmöglich: Es gibt $d^* > d - \log_2 n$ mit $|X_{d^*-1}| \leqslant 2|X_{d^*}|$.
- Π ist die Zerlegung bestehend aus
  - den Einermengen für alle Knoten in X_{d^*} und den
  - Zusammenhangskomponenten von B nach Herausnahme von X_{d^*}.

Behauptung: (X_{d^*-1}, Π) ist ein Zeuge.

MIN DEGREE SPANNING TREE: Die Analyse (3/4)

Zeige: Alle Kanten $\{u_3, u_4\}$, die verschiedene Klassen in Π verbinden, sind mit einem Knoten in X_{d^*-1} inzident.

- Wenn die Kante $\{u_3, u_4\}$ mit einer Einermenge $\{v\}$ für $v \in X_{d^*}$ inzident ist: Kein Problem, denn $X_{d^*} \subseteq X_{d^*-1}$.
- Ansonsten verbindet $\{u_3, u_4\}$ Zusammenhangskomponenten, die durch die Herausnahme von X_{d^*} entstanden sind.
  Und wenn $u_3 \notin X_{d^*-1}$ wie auch $u_4 \notin X_{d^*-1}$?
  - Dann ist $\max\{\text{grad}_B(u_3), \text{grad}_B(u_4)\} < d^* - 1$.
  - Betrachte alle Klassen-internen Kanten.
    - ★ Fügen wir die Klassen-verbindende Kante $\{u_3, u_4\}$ hinzu, wird kein Kreis geschlossen.
    - ★ Fügen wir jetzt die Knoten aus X_{d^*} und ihre inzidenten Kanten hinzu, dann erhalten wir B und zuzüglich die Kante $\{u_3, u_4\}$.
    - ★ Ein Kreis mit einer Kante $\{u_1, u_2\}$ (für $u_1 \in X_{d^*}$) wird geschlossen.
  - $B \setminus \{u_1, u_2\} \cup \{u_3, u_4\}$ ist ein etwas-besserer Nachbar und unser Algorithmus stoppt nicht mit B: Widerspruch.

MIN DEGREE SPANNING TREE: Die Analyse (4/4)

- (X_{d^*-1}, Π) ist ein Zeuge und deshalb
- kleinstmöglicher Grad $\geqslant \frac{\mathrm{rank}(\Pi)-1}{|X_{d^*-1}|}$.

- Wie groß ist $\mathrm{rank}(\Pi)$?
  - Sei A_{d^*} die Anzahl der Kanten, die in B mit einem Knoten aus X_{d^*} inzident sind.
  - B ist ein Baum, und deshalb ist $\mathrm{rank}(\Pi) = A_{d^*} + 1$.
- Weiterhin ist $A_{d^*} \geqslant d^* \cdot |X_{d^*}| - (|X_{d^*}| - 1)$, denn
  - alle Knoten in X_{d^*} haben mindestens d^* inzidente Kanten und
  - $\leqslant |X_{d^*}| - 1$ Kanten des Baums B verbinden zwei Knoten aus X_{d^*}.

$$\begin{aligned}\text{kleinstmöglicher Grad} &\geqslant \frac{\mathrm{rank}(\Pi)-1}{|X_{d^*-1}|} \geqslant \frac{d^* \cdot |X_{d^*}| - |X_{d^*}| + 1}{|X_{d^*-1}|} \\ &\geqslant \frac{d^* \cdot |X_{d^*}| - |X_{d^*}| + 1}{2|X_{d^*}|} > \frac{d^*-1}{2}.\end{aligned}$$

MIN DEGREE SPANNING TREE: Zusammenfassung

- Wir wissen: $d^* \geqslant d - \log_2 n$.
- Als Konsequenz:
  $d - \log_2 n \leqslant d^* < 2 \cdot (\text{kleinstmöglicher Grad}) + 1$.

Unser Algorithmus bestimmt einen Spannbaum mit Grad d, wobei

$$d \leqslant 2 \cdot (\text{kleinstmöglicher Grad}) + 1 + \log_2 n.$$

Unsere Lösung ist fast 2-approximativ.

Lokale Suche in variabler Tiefe

Lokale Suche in variabler Tiefe

Die Annahme: Der Lösungsraum ist eine Teilmenge von $\{0,1\}^n$.

(1) Setze EINGEFROREN $= \emptyset$, $\mathcal{L} = \{y\}$ und $z = y$.
// EINGEFROREN ist eine Menge von Positionen und
// $\mathcal{L}$ ist eine Menge von Lösungen.

(2) Wiederhole, solange EINGEFROREN $\neq \{1, \ldots, n\}$

(2a) Bestimme eine beste Lösung $z' \neq z$ in der k-Flip Nachbarschaft von z.

(2b) Friere alle im Wechsel von z nach z' geänderten Positionen ein, füge z' zu $\mathcal{L}$ hinzu und setze $z = z'$.
// Wir erlauben Aufwärtsbewegungen, da z' eine schlechtere
// Lösung als z sein darf. Durch das Einfrieren geänderter
// Positionen wird die Schleife höchstens n Mal durchlaufen.

(3) Gib die beste Lösung in $\mathcal{L}$ als neue Lösung y' aus und wiederhole das Verfahren gegebenenfalls für y'.

MINIMUM BALANCED CUT

- Ein ungerichteter Graph $G = (V, E)$ gegeben.
- **Ziel:** Bestimme eine Zerlegung $V = V_1 \cup V_2$ mit $|V_1| = \lfloor \frac{|V|}{2} \rfloor, |V_2| = \lceil \frac{|V|}{2} \rceil$ so dass die Anzahl **kreuzender** Kanten, also die Anzahl aller Kanten mit einem Endpunkt in V_1 und einem Endpunkt in V_2,

  minimal ist.

- Eine Anwendung der lokalen Suche in variabler Tiefe wählt die 2-Flip Nachbarschaft.
  - Ein Nachbar einer Zerlegung $V = V_1 \cup V_2$ ist von der Form $V = U_1 \cup U_2$, wobei U_1 durch das Entfernen und das nachfolgende Einfügen eines Elementes aus V_1 entsteht.
  - Wir werden später eine Anwendung von Simulated Annealing kennenlernen.
- Für **MINIMUM BALANCED CUT** werden gute experimentelle Ergebnisse berichtet. Gleiches trifft auf **TSP** zu.

Der Metropolis Algorithmus

Der Metropolis Algorithmus

(1) Sei y eine Anfangslösung und sei T die Temperatur.

(2) Wiederhole hinreichend oft:

- Wähle zufällig einen Nachbarn $y' \in \mathcal{N}(y)$.
- Wenn $f(y') \leqslant f(y)$, dann akzeptiere y' und setze $y = y'$.
  Ansonsten setze $y = y'$ mit Wahrscheinlichkeit $e^{-\frac{f(y')-f(y)}{T}}$.

  // Je schlechter der Wert des neuen Nachbarn y', umso geringer
  // die Wahrscheinlichkeit, dass y' akzeptiert wird.
  // Schlechte Nachbarn haben nur eine Chance bei
  // entsprechend hoher Temperatur.

Grenzverteilung und Konvergenzgeschwindigkeit

Mit welcher Wahrscheinlichkeit wird die Lösung y besucht, wenn der Metroplis-Algorithmus genügend häufig wiederholt wird?

$f_y(t)$ sei die relative Häufigkeit, mit der der Metropolis-Algorithmus die Lösung y in den ersten t Schritten besucht. Dann gilt

$$\lim_{t\to\infty} f_y(t) = \frac{e^{-f(y)/T}}{Z}, \text{ wobei } \quad Z = \sum_{y, L(y)} e^{-f(y)/T}.$$

- Wenn man also lange genug sucht, dann besucht man gute Lösungen mit sehr viel höherer Wahrscheinlichkeit als schlechte.
- Wie lange ist lange? Leider häufig „**sehr lange**“.
  Die Konvergenzgeschwindigkeit ist leider i. A. sehr langsam.

Beispiel: Das Vertex Cover Problem (1/2)

Der leere Graph G $= (\mathbf{V}, \emptyset)$:

- Anfänglich wird die Knotenmenge nur reduziert und Metropolis verhält sich wie die lokale Suche.
- Umso kleiner die gegenwärtige Lösung ist, umso größer ist die Anzahl der schlechteren Nachbarn.
  - Die schlechteren Nachbarn sind also wahrscheinlicher als die guten.
  - Die Akzeptanzwahrscheinlichkeit einer schlechteren Lösung ist aber nur unwesentlich kleiner:

    Schlechtere Lösungen werden kurz über lang gewählt:

Der Metropolis-Algorithmus bekommt Angst vor der eigenen Courage, wenn gute, aber längst nicht optimale Lösungen erreicht werden.

Beispiel: Das Vertex Cover Problem (2/2)

Der Sterngraph:

- Der Metropolis-Algorithmus zeigt seine Stärke: Selbst wenn das Zentrum irgendwann entfernt wird, so ist die Wahrscheinlichkeit hoch, dass es nach nicht zu langer Zeit wieder betrachtet wird.
  - Mit beträchtlicher Wahrscheinlichkeit wird das Zentrum, wenn nicht im ersten, dann in folgenden Versuchen wieder aufgenommen.
  - Danach, wenn mindestens ein Satellit **nicht** in der jeweiligen Lösung liegt, bleibt das Zentrum erhalten, da sonst keine Überdeckung vorliegt.
- Allerdings hat Metropolis auch für den Sterngraphen Angst vor der eigenen Courage, wenn die Überdeckung genügend klein ist:

  Unbenötigte Satelliten werden aufgenommen!
- Wir brauchen ein Verfahren, das Aufwärtsbewegungen „mit der Zeit“ erschwert.

Simulated Annealing

Simulated Annealing

Wir müssen Aufwärtsbewegungen nach entsprechend langer Suchzeit signifikant erschweren. Wie? Wir senken die Temperatur!

Eine Analogie aus der Physik:

- Erhitzt man einen festen Stoff so stark, dass er flüssig wird und läßt man ihn dann wieder langsam abkühlen, so ist der Stoff bestrebt, möglichst wenig der zugeführten Energie zu behalten.
- Der Stoff bildet eine Kristallgitterstruktur; Eiskristalle sind ein Beispiel.
- Je behutsamer nun das Ausglühen (*engl.: Annealing*) durchgeführt wird, umso reiner ist die Gitterstruktur.

  Unregelmäßigkeiten in der Gitterstruktur, die durch zu rasches Abkühlen entstehen, stellen lokale Energieminima dar.

Simulated Annealing: Der Algorithmus

(1) Sei y die Anfangslösung und sei **T** die Anfangstemperatur.

(2) Wiederhole hinreichend oft:

- Wähle zufällig einen Nachbarn $y' \in \mathcal{N}(y)$.
- Wenn $f(y') \leqslant f(y)$, dann akzeptiere y' und setze $y = y'$.
  Ansonsten setze $y = y'$ mit Wahrscheinlichkeit $\mathbf{e}^{-\frac{\mathbf{f(y)-f(x)}}{\mathbf{T}}}$.

(3) Wenn die Temperatur noch nicht genügend tief ist, dann wähle eine neue, niedrigere Temperatur **T** und führe Schritt (2) aus. Ansonsten gib die beste erhaltene Lösung aus.

MINIMUM BALANCED CUT (1/2)

Nach der Anwendung der **lokalen Suche in variabler Tiefe** wenden wir jetzt **Simulated Annealing** an.

- Wir lassen beliebige Zerlegungen $(W, V \setminus W)$ zu und wählen

$$f(W) := |\{e \in E \mid |\{e\} \cap W| = 1\}| + \underbrace{\alpha \cdot (|W| - |V \setminus W|)^2}_{\text{Strafterm}}$$

  als zu minimierende **Zielfunktion**. Mit dem Strafterm versuchen wir eine perfekte Aufteilung zu erzwingen.
- Wir wählen die **1-Flip Nachbarschaft**. Die Startlösung ist eine zufällig gewählte Teilmenge und ihr Komplement.
- Die **Anfangstemperatur** wird so gesetzt, dass 40% aller Nachbarn akzeptiert werden.
- Halte die **Temperatur** für $16 \cdot |V|$ Schritte konstant und senke dann um den Faktor $0,95$ („geometrische Abkühlung“).

- **Für zufällige Graphen:**
  - Simulated Annealing schneidet sehr gut ab und schlägt die lokale Suche in variabler Tiefe, selbst wenn beide Verfahren gleich lange laufen.
- **Strukturierte Graphen:** 500 (bzw. 1000) Punkte werden zufällig aus dem Quadrat $[0, 1]^2$ gewählt und zwei Punkte werden verbunden, wenn sie nahe beieinander liegen.
  - Simulated-Annealing bricht ein und die lokale Suche in variabler Tiefe ist deutlich überlegen.
- Die beiden Graphklassen unterscheiden sich in der Struktur ihrer lokalen Minima:
  - Die strukturierten Graphen haben Minima mit weitaus größeren Anziehungsbereichen als die zufällig generierten Graphen.
  - Die „Sogwirkung“ lokaler Minima ist für strukturierte Graphen schwieriger zu überwinden als für Zufallsgraphen.

Evolutionäre Algorithmen

Evolutionäre Algorithmen

- Die Zielfunktion f sei über einem Lösungsraum zu maximieren.
- Die Evolution als Vorbild: Wie können wir das Erfolgsprinzip der Evolution, „**Survival of the Fittest**“, algorithmisch umsetzen?
  - Die Fitness eines Individuums (oder einer Lösung) y stimmt mit dem Funktionswert $f(y)$ überein.
  - Eine gegebene Population von Lösungen ist zu verbessern, indem Lösungen „**mutieren**“ oder zwei oder mehrere Lösungen miteinander „**gekreuzt**“ werden.
    - ⋆ Dazu werden **Mutations-** und **Crossover-Operatoren** angewandt.

Die Grobstruktur evolutionärer Algorithmen

(1) **Initialisierung:** Eine Anfangspopulation P von Lösungen ist zu bestimmen.

(2) Wiederhole, bis eine genügend gute Lösung gefunden wurde:

  (2a) **Selektion zur Reproduktion**: Eine Teilmenge $P' \subseteq P$ der gegenwärtigen Population P von Lösungen wird ausgewählt.

  (2b) **Variation:** Neue Lösungen werden aus P' mit Hilfe der Mutation- und Crossover-Operatoren gewonnen.

  (2c) **Selektion zur Ersetzung:** Die neue Population ist festzulegen.

Die Implementierung evolutionärer Algorithmen (1/2)

- **Initialisierung:** Die Anfangspopulation wird
  - durch eine zufällige Auswahl definiert, bzw. mit Hilfe problem-spezifischer Heuristiken.
  - **Diversität** ist zu erzwingen, damit der gesamte Lösungsraum „repräsentiert“ wird.
- **Selektion zur Reproduktion:** Eine Teilmenge $P' \subseteq P$ von Elternlösungen ist auszuwählen. Zu den häufig benutzten Auswahlverfahren gehören:
  - die **zufällige Auswahl nach der Gleichverteilung**: Jede Lösung in P wird mit gleicher Wahrscheinlichkeit gewählt,
  - die **zufällige Auswahl nach der Fitness:** Falls f positiv ist, wird $y \in P$ mit Wahrscheinlichkeit $\frac{f(y)}{N}$ für $N = \sum_{z \in P} f(z)$ oder mit Wahrscheinlichkeit $\frac{e^{f(x)/T}}{M}$ für $M = \sum_{z \in P} e^{f(z)/T}$ gewählt
  - oder die **Turnier-Selektion**: Wähle k Lösungen aus P nach der Gleichverteilung und lasse die k' fittesten der k Lösungen zu.

Die Implementierung evolutionärer Algorithmen (2/2)

- **Variation:** λ Nachkommen werden aus P' mit Hilfe der Mutations- und Crossover-Operatoren erzeugt.
- **Selektion zur Ersetzung:** Die neue Generation ist festzulegen. Die Populationsgröße sei stets μ.
  - In der Plus-Auswahl werden die μ Fittesten aus den μ Lösungen der alten Generation und ihren λ Nachkommen bestimmt. Man spricht von der $(\mu + \lambda)$-Selektion.
  - In der Komma-Auswahl wird $\lambda \geqslant \mu$ angenommen, und die μ fittesten Nachkommen bilden die neue Generation. Man spricht von der (μ, λ)-Selektion.

Mutationsoperatoren (1/2)

- **Bitmutationen**: Man nimmt an, dass der Lösungsraum der n-dimensionalen Würfel $\mathbb{B}^n = \{0,1\}^n$ ist.
  - Entweder flippt man jedes Bit einer Elternlösung $y \in \mathbb{B}^n$ mit einer kleinen Wahrscheinlichkeit p (mit $p \cdot n = \Theta(1)$)
  - oder man ersetzt y durch einen Nachbarn y' in der k-Flip Nachbarschaft von y für kleine Werte von k.
- **Mutationen für reellwertige Vektoren:** Im $\mathbb{R}^n$ ersetzt man eine Elternlösung y durch $\mathbf{y}' = \mathbf{y} + \mathbf{m}$, wobei die Komponenten von m zufällig und unabhängig voneinander mit Erwartungswert 0 gewählt werden.
  - Entweder man wählt die Komponenten m_i von m zufällig aus einem fixierten Intervall $[-a, a]$
  - oder man erlaubt unbeschränkte Komponenten, die zum Beispiel gemäß der Normalverteilung gezogen werden.

    Für schlechte Lösungen erlaubt man eine hohe Standardabweichung und reduziert die Standardabweichung je besser die Lösung ist.

Mutationsoperatoren (2/2)

- Für den **Permutationsraum**

$$\mathbb{S}^n = \{\pi \mid \pi \text{ ist eine Permutation von } \{1, \ldots, n\}\}$$

  betrachtet man **Austausch-** und **Sprungoperatoren**.
  - In einem Austauschoperator wird ein Paar (i, j) mit $1 \leqslant i \neq j \leqslant n$ zufällig und gleichverteilt aus allen möglichen Paaren gezogen. Die $i$te und $j$te Komponente der Elternlösung werden vertauscht.
  - Auch für einen Sprungoperator werden Paare (i, j) zufällig gezogen. Die $i$te Komponente y_i einer Elternlösung y wird an die Position j gesetzt und die restlichen Komponenten werden passend verschoben.
- Stärker problemabhängige Mutationsoperatoren sind sinnvoll, solange die Diversität nicht zu stark eingeschränkt wird.

  Wenn zum Beispiel eine Elternlösung y durch ein besseres lokales Maximum ersetzt wird, dann ist ein zukünftiges Verlassen des lokalen Maximums schwierig.

Crossover Operatoren

$y_1, \ldots, y_k$ seien k Eltern, wobei $k \geqslant 2$ gelte.

- Für den Würfel $\mathbb{B}^n$ betrachtet man nur den Fall $k = 2$. Im **r-Punkt Crossover** wählt man $r \leqslant 2$ Positionen $1 \leqslant i_1 < \cdots < i_r \leqslant n$. Der Sprößling z von y_1 und y_2 erbt die ersten i_1 Bits von y_1, die $i_2 - i_1$ Bits in den Positionen $[i_1 + 1, i_2]$ von y_2, die $i_3 - i_2$ Bits in den Positionen $[i_2 + 1, i_3]$ von y_1 und so weiter.
- Im $\mathbb{R}^n$
  - verwendet man den r-Punkt Crossover ebenfalls.
  - Sehr verbreitet ist das **arithmetische Crossover**: $\sum_{i=1}^{k} \alpha_i y_i$ wird zum Nachkommen von $y_1, \ldots, y_k$. Der wichtige Spezialfall $\alpha_1 = \cdots = \alpha_k = \frac{1}{k}$ wird **intermediäre Rekombination** genannt.
- Im Permutationsraum $\mathbb{S}^n$ wählt man meistens $k = 2$ Eltern y_1 und y_2. Zum Beispiel werden Positionen $1 \leqslant i_1 < i_2 \leqslant n$ zufällig ausgewürfelt. Die in den Positionen $i_1, \ldots, i_2$ liegenden Komponenten von y_1 werden dann gemäß y_2 umgeordnet.

Zusammenfassung (1/2)

- Die **strikte lokale Suche** führt zu einer verbesserten Lösung, die aber nur einem lokalen Optimum entspricht.

  Zu den erfolgreichen Anwendungen gehören
  - der Simplex-Algorithmus für die lineare Programmierung,
  - die Methode des Gradientenabstiegs in der Minimierung von Funktionen,
  - und die Bestimmung approximativer Lösungen für
    - ★ FACILITY LOCATION
    - ★ sowie die Spannbaumprobleme MAX LEAF SPANNING TREE und MIN DEGREE SPANNING TREE.

Zusammenfassung (2/2)

- Die **lokale Suche in variabler Tiefe** erlaubt Aufwärtsbewegungen und führt für das TSP und das Minimum Balnced Cut Problem zu guten Lösungen.
- Der **Metropolis Algorithmus** akzeptiert Aufwärtsbewegungen mit einer durch einen Temperatur-Parameter T gesteuerten Wahrscheinlichkeit. Wenn die Temperatur T langsam gesenkt wird, erhält man Simulated Annealing.
- **Evolutionäre Algorithmen** verbessern eine Lösungspopulation mit Hilfe von Mutations- und Crossover-Operatoren. Die Auswahl der neuen Population wird durch die Fitness, also durch den Funktionswert der Lösungen gesteuert.