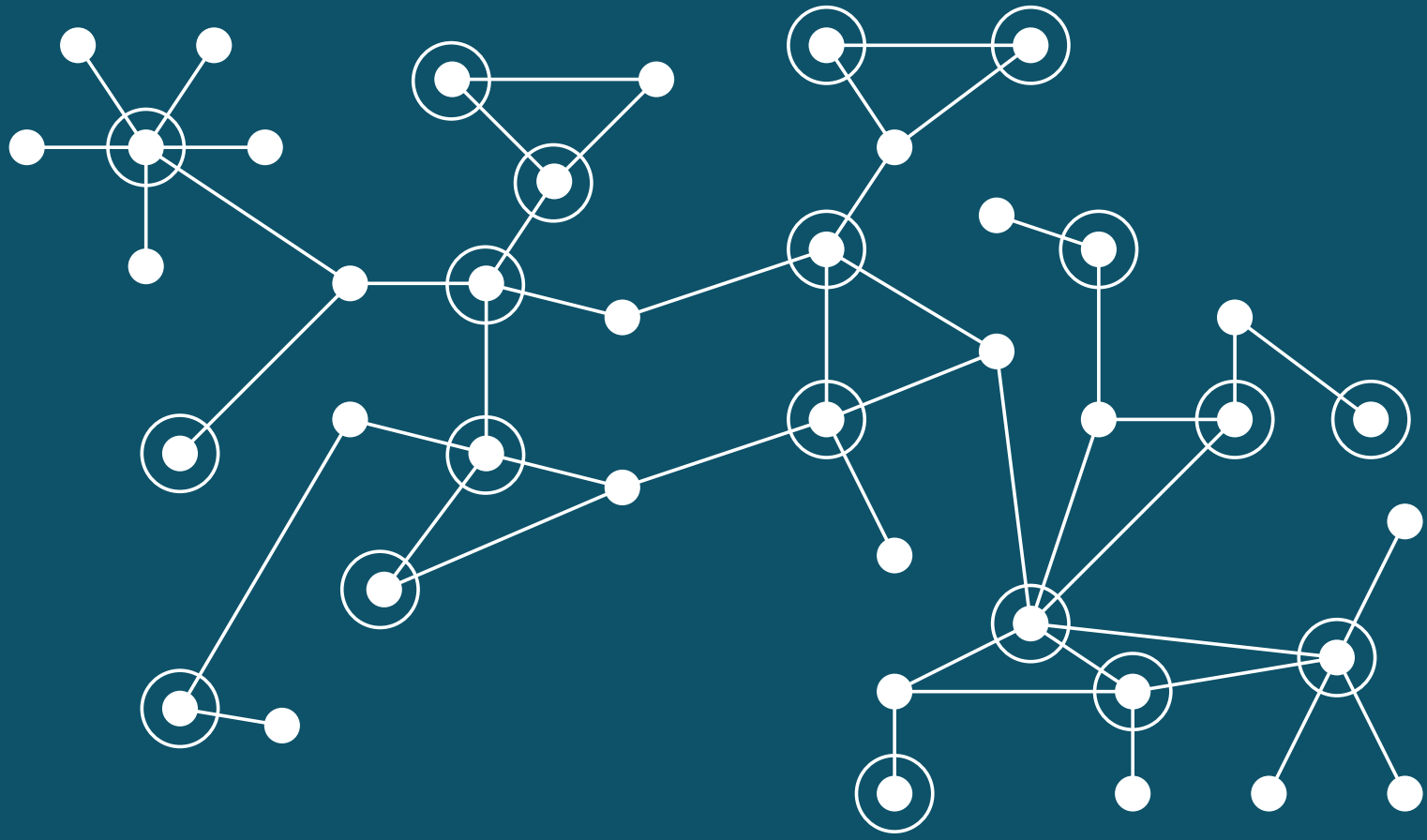
Das Data Cleaning Problem

Beispiel: Ein ehrgeiziger junger Wissenschaftler, Dr. O, hat eine Menge von Daten gesammelt, die seine neue Theorie untermauern würden, und ist voller Freude weil er knapp davor steht, seine Resultate publizieren zu können. Allerdings sind einige wenige Paare der Beobachtungen nicht miteinander konsistent. Es stellt sich folgendes natürliche Problem: Gibt es einen Weg, eine minimale Anzahl solcher Beobachtungen auszuwählen, sodass deren Elimination alle Inkonsistenzen beseitigen würde?

Bemerkung: Anzahl der Datenpunkte: ungefähr 1000.

Frage: Wie kann Dr. O sein Problem lösen?

Beispiel: Teil einer Data Cleaning Instanz mit Lösung:



Dr. $\mathcal{O}$ zeigt das Problem Dr. $\mathcal{T}$, einem theoretischen Informatiker. Dieser beweist:

- Data Cleaning ist **NP**-vollständig (Beweis :-)
- Data Cleaning ist auch **APX**-vollständig (falls die Geschichte nach 1990 stattfindet :-) und daher nur relativ schlecht approximierbar. Dann verschwindet Dr. $\mathcal{T}$:-)
- Dr. $\mathcal{O}$ ist verzweifelt, denn er weiss nun: Sein Problem lässt sich im Wesentlichen nur durch systematisches Testen aller möglichen Untermengen für jede angenommene minimale Anzahl k lösen, was mehr als n^k Zeiteinheiten benötigen würde. Dies ist bedauerlicherweise bereits für $k \geq 5$ mit mehr als 10^{15} Schritten an der Grenze des Machbaren. Dr. $\mathcal{O}$ vermutet aber, dass $20 \leq k \leq 50$.

Asymptotische polynomialzeit Komplexität

- Laufzeit begrenzt durch Cn^{α} für Konstanten C und α für hinreichend große $n > n_0$.
- Berechtigte Zwischengfrage: sind Algorithmen mit $O(n^{100})$ wirklich relevant?
- Polynomialzeit Komplexität ist ein erfolgreiches, sich wohlverhaltendes Modell für machbare Berechenbarkeit, weil für fast jedes natürliche Problem gilt:
  - Es kann entweder in $O(n^3)$ gelöst werden (oft kann man das dann sogar auf $O(n \log n)$ verbessern).
  - Oder es ist hoffnungslos schwierig zu lösen, dh **NP**-vollständig oder schwieriger.

▷

- Mit anderen Worten, wenn man das Problem gut lösen kann, dann üblicherweise sogar sehr gut (ein faszinierendes empirisches Resultat über die Natur der Komplexitätstheorie).

- Die Mehrheit der theoretischen Informatiker geht davon aus, dass Probleme des zweiten Typs inhärent schwierig sind und es daher keine effizienten Algorithmen für sie gibt. Die letzten Jahrzehnte haben aber überraschend wenig Fortschritte in Richtung eines Beweises der **P** $\neq$ **NP** Vermutung gebracht, die einer Formalisierung dieser Intuition entspräche und als das wichtigste offene Problem der Informatik und der Grundlagen der Mathematik gilt.

Das Data Cleaning Problem (Fortsetzung)

Glücklicherweise für Dr. O gibt es die Maschinerie der parameterisierten Komplexitätstheorie:

- 1986 konnten Fellows und Langston unter Verwendung der mächtigen Robertson-Seymour graph minor Theorie zeigen, dass für ein festgelegtes k sich das $k-$Data Cleaning Problem in Zeit $O(f(k)n^3)$ lösen lässt, wobei n die Anzahl der Datenpunkte ist und $f(k)$ allerdings eine sehr schnell wachsende Funktion, die nur von k, der Menge der zu eliminierenden Datenpunkte abhängt.

- 1987 zeigte Johnson, dass ein $O(f(k)n^2)$ Algorithmus basierend auf Baumzerlegungen und finite-state dynamischer Programmierung dafür existiert.
- 1988 präsentierte Fellows einen $O(2^k n)$ Algorithmus, der auf einer einfachen Baumsuche basiert.
- 1989 beschrieb Buss einen $O(kn+2^k k^{2k+2})$ Algorithmus.
- 1992 verbesserten Balasubramanian et al. dies zu $O(kn+2^k k^2)$.
- 1998 konnten Balasubramanian et al. dies nochmals zu $O(kn+(53/40)^k k^2)$ verbessern.
- weitere Verbesserungen sind möglich ...

Definition: Vertex Cover

Gegeben: (ungerichteter) Graph $G = (V, E)$, $k \in \mathbb{N}$

Parameter: k. **Gefragt:** Besitzt G einen Vertex Cover der Größe k? (Ein Vertex Cover der Größe k ist eine Teilmenge $V' \subseteq V$ mit $|V'| = k$, sodaß $\forall \{u, v\} \in E$ gilt: $u \in V'$ oder $v \in V'$, $\rightarrow$ jede Kante aus E wird von mindestens einem Knoten aus V' abgedeckt, d.h. berührt).

Wir wissen bereits, dass Vertex Cover **NP**-vollständig ist.

Es gilt aber offenbar auch:

Theorem: $k-$Vertex Cover ist in linearer Zeit lösbar.

Beweis: Gegeben $G = (V, E)$ und fixes k. Man baut einen binären Baum der Tiefe k wie folgt auf. Der Wurzelknoten des Baumes wird mit einer beliebigen Kante $e_1 = \{a, b\} \in E$ beschriftet, die von ihm ausgehende linke Kante mit

▷

Knoten a und die rechte mit Knoten b. Es gilt: entweder a oder b muss in jedem Vertex Cover sein. Jeder darunter liegende Knoten des Baumes wird mit einer weiteren Kante aus E beschriftet, die nicht inzident ist mit einem der Knoten aus V, mit denen die Kanten des Baumes, die zum Wurzelknoten führen, beschriftet sind. Die linken und rechten Kanten werden wieder mit den Knoten aus V beschriftet, und es gilt wieder, dass einer der beiden in jedem Vertex Cover sein. An den 2^k Blättern des Baumes angelangt, kann für jeden Pfad zum Wurzelknoten in linearer Zeit festgestellt werden, ob die auf den Kanten des Pfades angeschriebenen Knoten aus V einen $k-$Vertex Cover beschreiben. Wenn dies an keinen Blatt der Fall ist, dann gibt es keinen Vertex Cover der Größe k (ansonsten hat man einen gefunden). $\surd$

Der Pakt mit dem Teufel

Erfreuliche Konsequenz: Dank der beschriebenen Entwicklung mit Hilfe der parameterisierten Komplexitätstheorie kann Dr. $\mathcal{O}$ sein Data Cleaning Problem für praktisch beliebige Datenmengen mit effizienten Algorithmen lösen, solange die Anzahl der „schlechten“ Punkte unter 70 liegt, was ja bei ihm der Fall war.

Die Idee bei der parameterisierten Komplexitätstheorie kann man wie folgt zusammenfassen: Man versucht die ganze Schwierigkeit des Problems zum Parameter zu verschieben. Dabei sollten die realen Instanzen, an denen wir interessiert sind, nur hinreichend kleine Parameterwerte benötigen, sodass sich dieser Pakt mit dem Teufel auch tatsächlich auszahlt.

Parameterisierte Komplexitätstheorie

Definition: Eine parameterisierte Sprache L ist eine Untermenge $L \subseteq \Sigma^* \times \Sigma^*$. Falls $(x, y) \in L$, dann bezeichnen wir mit x den Hauptteil und mit y den Parameter, wobei y oft eine natürliche Zahl sein wird (da als konstant angenommen, spielt der daraus resultierende exponentielle Sprung keine Rolle).

Definition: Eine parameterisierte Sprache L wird als fixed parameter tractable (FPT) bezeichnet, wenn in Zeit $f(k) \times n^{\alpha}$ festgestellt werden kann, ob $(x, k) \in L$, wobei $|x| = n$, α ist eine Konstante unabhängig von n und k, und f ist eine beliebige Funktion (unabhängig von n).

Theorem: Vertex Cover ist fixed parameter tractable.

Beweis: siehe Theorem Seite 8.

Es ist allerdings nicht überraschend, dass nicht alle parameterisierten Probleme zu FPT gehören. Wir können die Plausibilität von fixed parameter intractability gewisser Probleme mit Hilfe entsprechender Reduktionen erhärten, indem wir (ähnlich wie bei der **NP**-Vollständigkeit) von vielen Problemen zeigen, dass sie alle gleich schwierig sind, ohne dass von einem davon gezeigt werden konnte, dass es in FPT ist.

Definition: Eine parameterisierte Transformation (Reduktion) einer parameterisierten Sprache L in eine andere parameterisierten Sprache L' ist ein Algorithmus, der aus einer Eingabe (x,k) ein Paar (x',k') berechnet, wobei:

- $(x,k) \in L \Leftrightarrow (x',k') \in L'$,
- $k' = g(k)$ ist eine Funktion nur in k, und
- die Berechnung erfolgt in Zeit $O(f(k) \times n^{\alpha})$, wobei $|x| = n$, α ist eine Konstante unabhängig von n und k, und f ist eine beliebige Funktion.

Beispiel: Es kann hilfreich sein, zu beobachten wie parameterisierte Transformationen sich von den bisher betrachteten polynomialzeit Transformationen unterscheiden. Erinnern wir uns, dass in einem Graphen $G = (V, E)$ mit n Knoten eine Menge $V' \subseteq V$ eine $k-$Clique in G ist, gdw $V - V'$ ein Vertex Cover des komplementären Graphen G' (in dem Kanten existieren gdw sie in G nicht existieren) ist. Das ergibt eine natürliche polynomialzeit Reduktion von dem parameterisierten Clique Problem zum parameterisierten Vertex Cover Problem. Aber es ist keine parameterisierte Transformation, da $k' = n - k$ keine Funktion ist, die nur von k abhängt. Die besten bisher gefundenen Algorithmen für k-Clique haben Komplexität $n^{\Omega(k/2)}$, und man nennt die Komplexitätsklasse der Probleme, auf die sich Clique reduzieren lässt, $W[1]$ (Name!).

Ein weiteres Beispiel für ein W[1]-vollständiges Problem ist das k-Schritt Halte-Problem für NTM (das allgemeine Problem ist ja bekanntermaßen sogar unentscheidbar!). Andererseits gibt es Probleme, auf die Clique mittels parameterisierter Transformation reduziert werden kann, aber keine Transformation in die andere Richtung bekannt ist, sodass sich daraus eine ganze Hierarchie von parameterisierten Komplexitätsklassen ergibt:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[SAT] \subseteq W[P] \subseteq AW[P] \subseteq XP$$

Von vielen parameterisierten Problemen ist inzwischen die Komplexität ermittelt worden, es gibt aber auch noch sehr viel zu tun, und im Falle von FPT kann man die Theorie auch dazu verwenden, effiziente Algorithmen zu konstruieren! $\rightarrow$ Übersicht über weitere Resultate:

http://www.cs.mun.ca/~harold/W_hier/