Algorithmen für schwierige Probleme

Britta Dorn

Wintersemester 2011/12

10. November 2011

Analyse von Suchbäumen

In den bisherigen Beispielen waren die Branchings immer sehr einfach und regulär:
In jedem Schritt wählt man ein Element aus, verringert k um den Wert 1. Dies führt bei l Verzweigungen zu einem Suchbaum der Größe $O(l^k)$.

Betrachtet man kompliziertere Branchings mit mehreren Fallunterscheidungen (vgl. Bsp. Vertex Cover), wird es auch komplizierter, die Größe des Suchbaums gut abzuschätzen.

Analyse von Suchbäumen

Um die Laufzeit für den Suchbaumalgorithmus zu bestimmen, brauchen wir die Größe des Baumes.

In nicht-degenerierten Bäumen mit Knotenmenge V (jeder innere Knoten hat mindestens zwei Kinder) gilt

$$\#\text{ Blätter} > \frac{|V|}{2}$$

Also gibt $O(\#\text{Blätter})$ die Größe des Suchbaums an.

Problem: Normalerweise haben wir nur eine Rekursionsgleichung für die Größe! (also nichts Explizites)

Bsp. VERTEX COVER: Tafel.
T_i – Größe des Suchbaums mit Tiefe i
B_i – Anzahl der Blätter des Suchbaums T_i.

Aufgabe: Rekursionsgleichungen verstehen/lösen!

Rekursionsgleichungen

Allgemein:

Lineare homogene Rekursionsgleichung mit konstanten Koeffizienten der Ordnung t

$B_n = c_1 B_{n-1} + c_2 B_{n-2} + \ldots c_t B_{n-t}$
(c_i sind die konst. Koeffizienten)

Gesucht: Explizite Gleichung einer Folge, die obige Rekursionsgleichung erfüllt.
Die obige Gleichung hat t Freiheitsgrade, nämlich die Anfangswerte $B_0, B_1, \ldots, B_{t-1}$. Wenn diese festgelegt sind, dann ist die Folge durch die Rekursionsgleichung eindeutig bestimmt!

Rekursionsgleichungen

$B_n = c_1 B_{n-1} + c_2 B_{n-2} + \ldots c_t B_{n-t}$
(c_i sind die konst. Koeffizienten)

Es gilt: Die Koeffizienten bestimmen das "charakteristische Polynom" der Rekursionsgleichung

$$p(x) = x^t - c_1 x^{t-1} - c_2 x^{t-2} - \cdots - c_t$$

Das char. Polynom hat t Nullstellen (nicht unbedingt reell). Kennt man diese, kann man die Lösungsfolge bestimmen:

Falls alle d Nullstellen $(r_1, \ldots, r_t)$ verschieden sind, ist die Lösungsfolge

$B_n = k_1 r_1^n + \cdots + k_t r_t^n$
(k_i so, dass es mit den Anfangsbedinungen $B_0, \ldots, B_{t-1}$ passt.)

Beispiel Tafel

Rekursionsgleichungen

Falls alle d Nullstellen $(r_1, \ldots, r_t)$ verschieden sind, ist die Lösungsfolge

$B_n = k_1 r_1^n + \cdots + k_t r_t^n$
(k_i so, dass es mit den Anfangsbedinungen $B_0, \ldots, B_{t-1}$ passt.)

Falls Nullstellen mehrfach auftreten, dann bekommt man noch andere Terme mit rein, z.B. für $p(x) = (x-2)^3$:
2 ist dreifache Nullstelle. Lösung ist
$B_n = k_1 \cdot 2^n + k_2 \cdot n \cdot 2^n + k_3 \cdot n^2 \cdot 2^n$. Kommt bei uns aber nie vor.

Rekursionsgleichungen

Falls alle d Nullstellen $(r_1, \ldots, r_d)$ verschieden sind, ist die Lösungsfolge

$B_n = k_1 r_1^n + \cdots + k_t r_t^n$
(k_i so, dass es mit den Anfangsbedinungen $B_0, \ldots, B_{t-1}$ passt.)

Wir interessieren und nur für eine Abschätzung der Größe des Suchbaums (mit $\mathcal{O}$-Notation), deshalb reicht es für uns, die betragsmäßig größte Nullstelle (oben r_1) zu kennen.
Bei uns haben die Rekursionsgleichungen die folgende Form:

$B_n = B_{n-d_1} + B_{n-d_2} + \ldots B_{n-d_r}$, Anfangswerte
$B_0 = B_1 = \ldots B_{d-1} = 1$

$(d_1, d_2, \ldots, d_r)$ heisst **Branchingvektor**. Die betragsmäßig größte Nullstelle des charakteristischen Polynoms ist die **Branchingzahl** α

Rekursionsgleichungen

Es gilt folgender Satz:

Theorem

Ein tiefenbeschränkter Suchbaum mit Branchingvektor $(d_1, \ldots, d_r)$ (k Stufen) hat Größe $|\alpha|^i$ (mit α betragsmäßig größte Nullstelle des entsprechenden char. Polynoms)

α heisst *Branchingzahl*

Klar: Reihenfolge der Zahlen im Branchingvektor spielt keine Rolle für Branchingzahl.

Ein bisschen Theorie zu Rekursionsgleichungen

Für uns reicht:

Rekursion $\rightarrow$ Branchingvektor $\overset{\text{Mathematik}}{\rightarrow}$ Branchingzahl $\alpha \rightarrow$ Größe des Suchbaums ist $O(\alpha^{\text{Tiefe}})$

Ein bisschen Theorie zu Rekursionsgleichungen

Für uns reicht:

Rekursion $\rightarrow$ Branchingvektor $\overset{\text{TABELLE}}{\rightarrow}$ Branchingzahl $\alpha \rightarrow$ Größe des Suchbaums ist $O(\alpha^{\text{Tiefe}})$

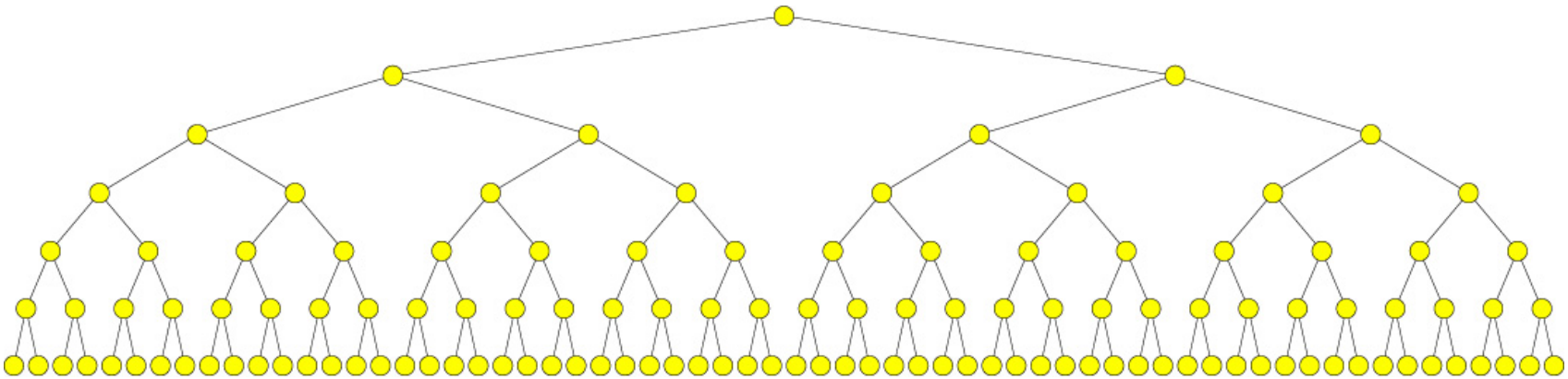
Tabelle für Branchingzahlen

branching vector	branching number
$(1,1)$	2.0
$(1,2)$	1.6180
$(1,3)$	1.4656
$(1,4)$	1.3803
$(2,1)$	1.6180
$(2,2)$	1.4142
$(2,3)$	1.3247
$(2,4)$	1.2720

branching vector	branching number
$(1,1,1)$	3.0
$(1,1,2)$	2.4142
$(1,1,3)$	2.2056
$(1,1,4)$	2.1069
$(1,2,1)$	2.4142
$(1,2,2)$	2.0
$(1,2,3)$	1.8929
$(1,2,4)$	1.7549

Ziel: Unterscheide Unterfälle so, dass möglichst günstige Branchingvektoren auftreten!

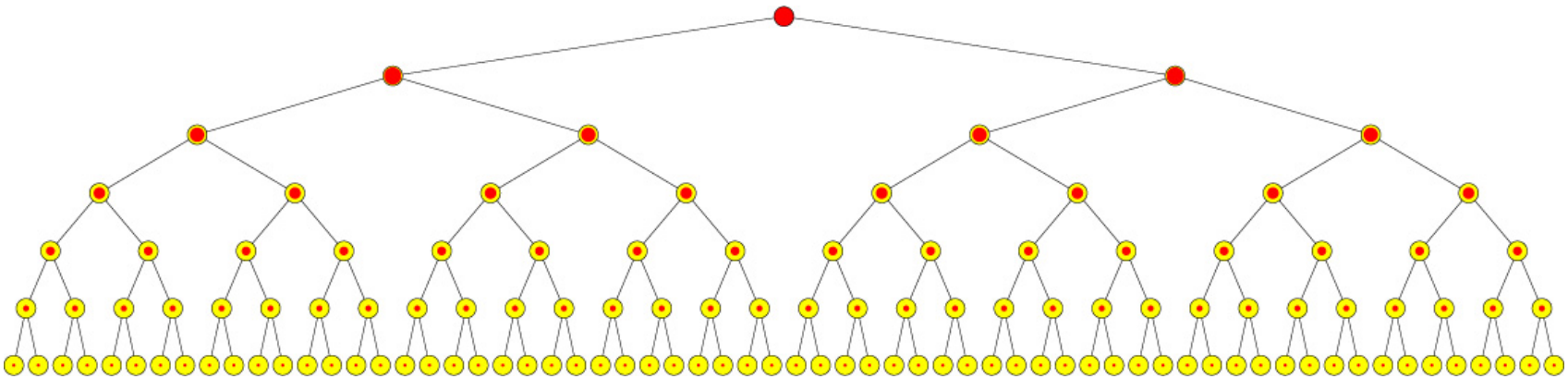
Verflechtung von Problemkernreduktion und Suchbäumen



Der Parameterwert wird in jeder Stufe verringert, aber es kann sein, dass die Größe der Instanz sich kaum ändert.

Verflechtung von Problemkernreduktion und Suchbäumen

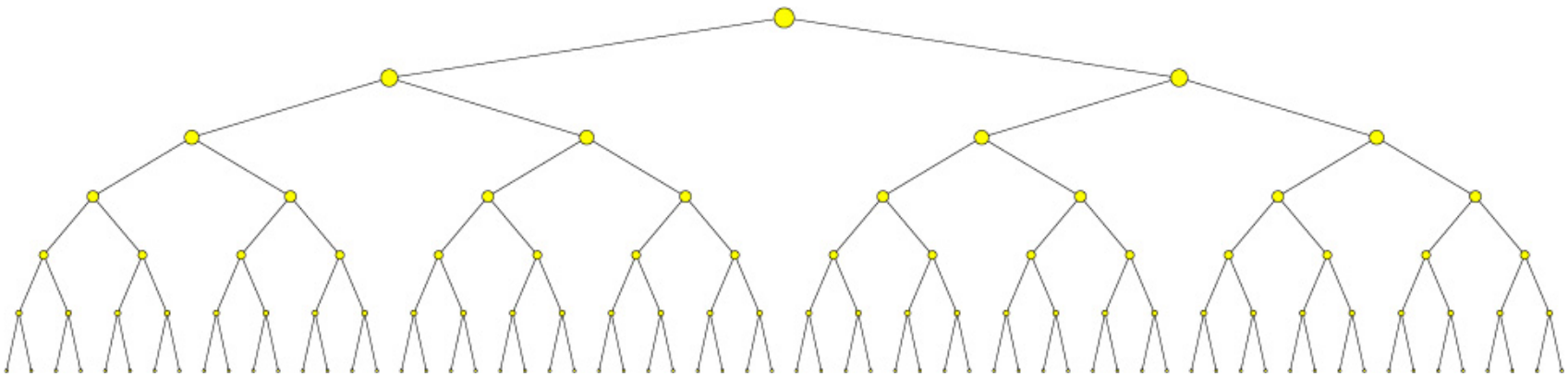
ohne Verflechtung:



Roter Teil jedes Knoten deutet Wert des Parameters an. Rekursion stoppt, falls Parameterwert klein genug ist. Fast alle Knoten sind Blätter oder in der Nähe von Blättern und haben kleinen Parameterwert.

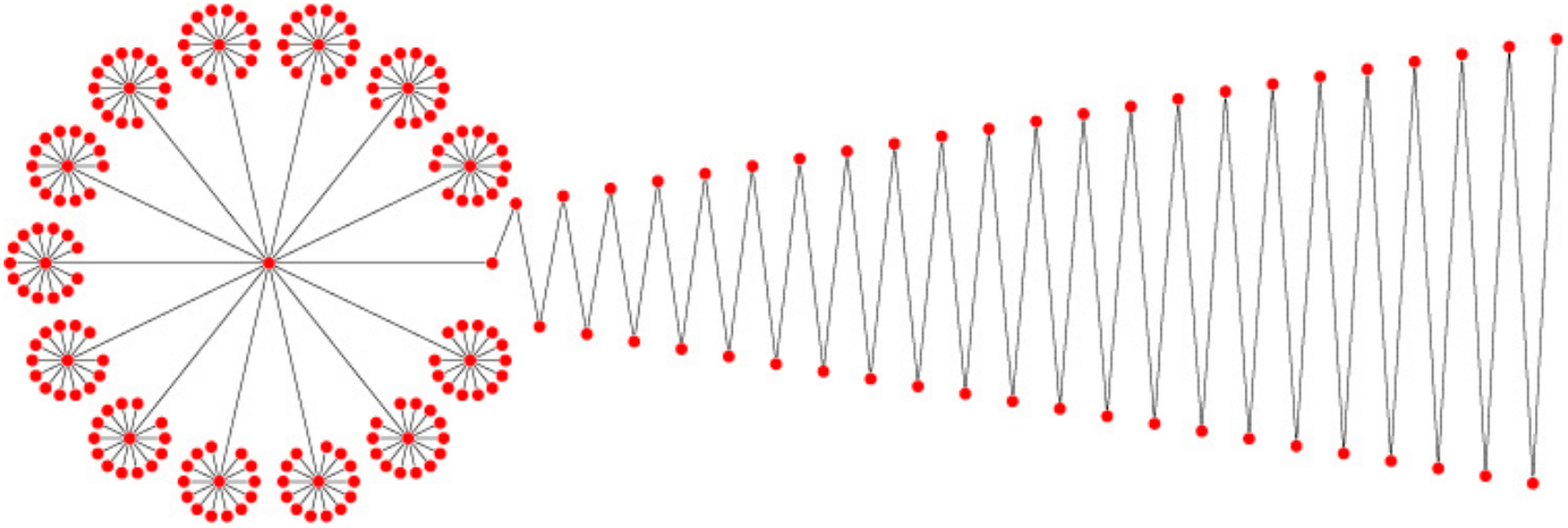
Verflechtung von Problemkernreduktion und Suchbäumen

mit Verflechtung:



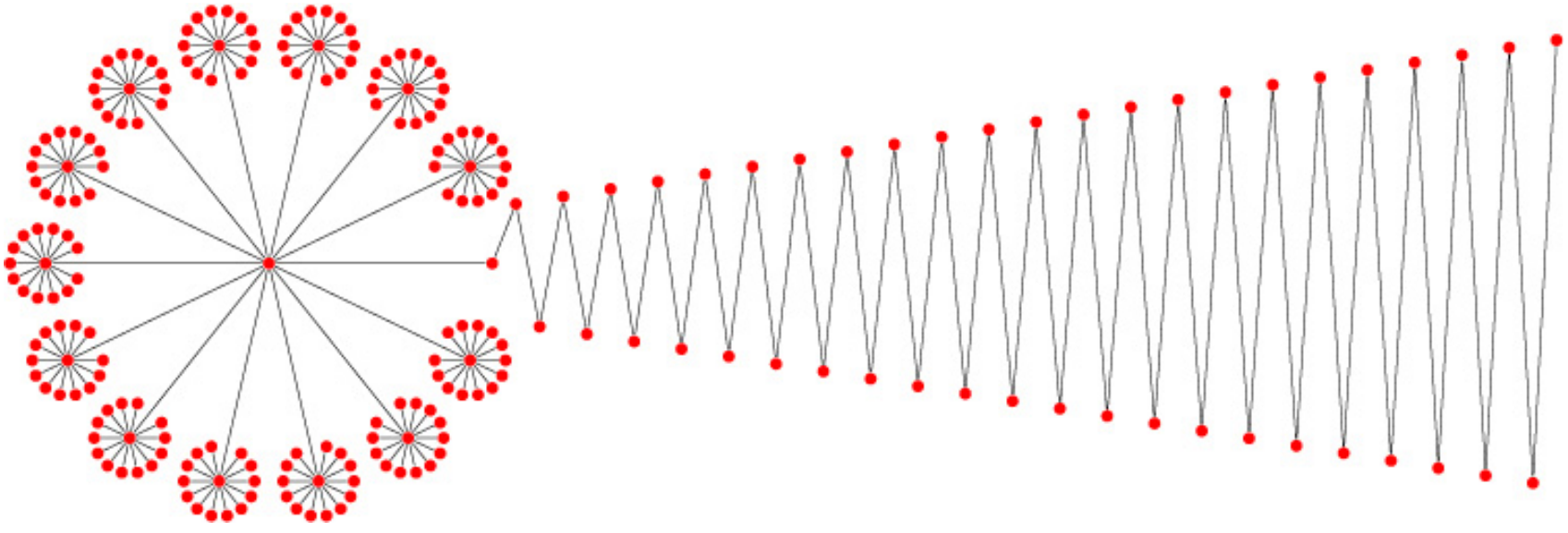
Nicht nur die Parameterwerte werden kleiner, sondern auch die Größen der Teilinstanzen (angedeutet durch die Größe der Knoten...)

Verflechtung von Problemkernreduktion und Suchbäumen: Beispiel Vertex Cover



Kopf: $(k-1)(k-2)+1$ Knoten (größter Teil), Schwanz: $3k+1$ Knoten, d.h. insgesamt k^2+4 Knoten. Im Bild für $k=15$.

Verflechtung von Problemkernreduktion und Suchbäumen: Beispiel Vertex Cover



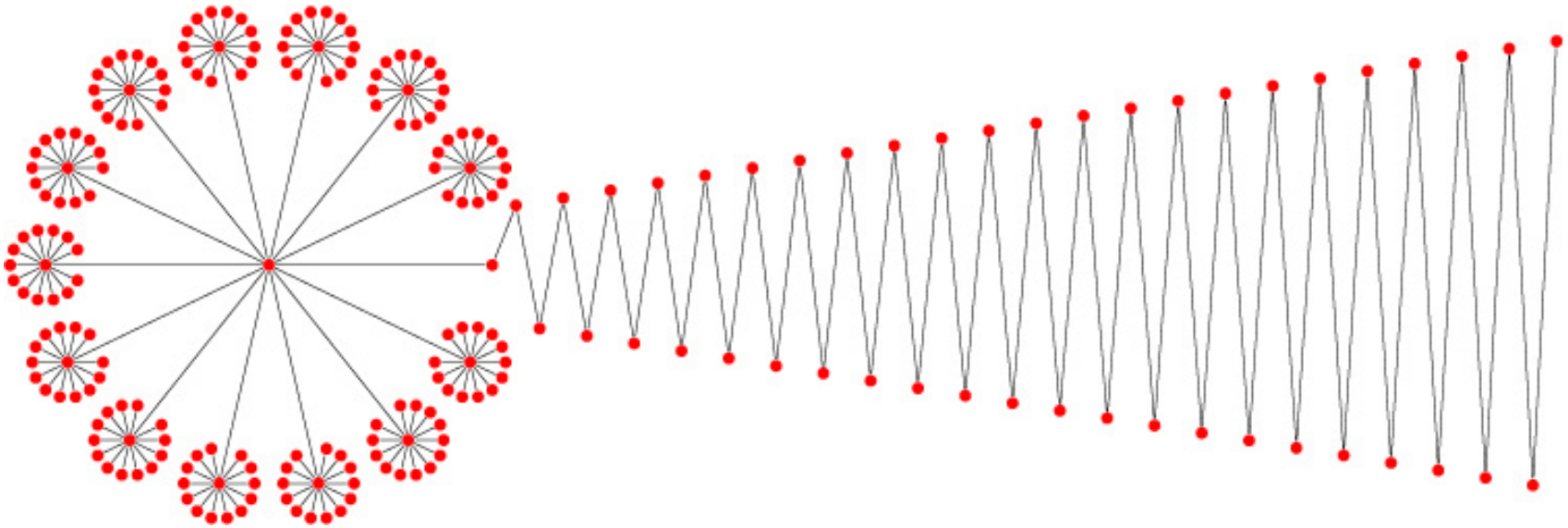
Reduktionsregel von Buss

Falls es einen Knoten mit Grad $> k$ gibt, nimm ihn ins VC auf, lösche ausgehende Kanten, verringere Wert von k um eins.

Reduktionsregel von Buss greift hier nicht
(alle Knoten haben Grad $< k$)

Verflechtung von Problemkernreduktion und Suchbäumen: Beispiel Vertex Cover

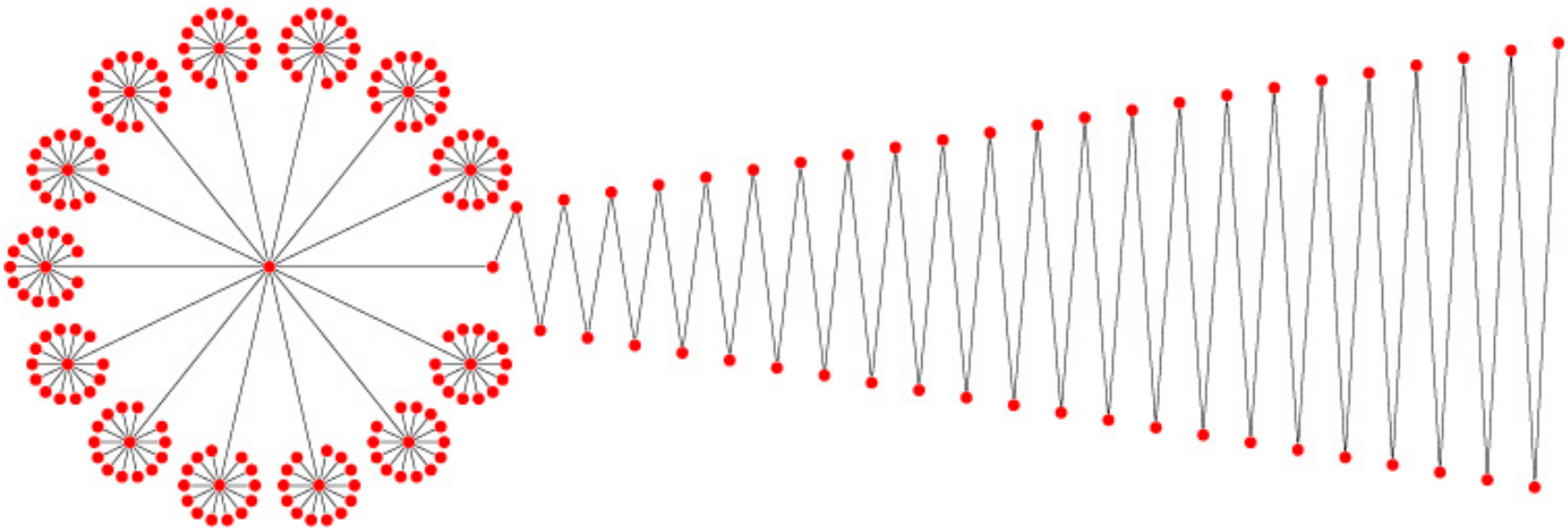
Normaler Suchbaumalgorithmus



Falls Suchbaumalgorithmus ("wähle zufällig Kante aus") mit Kanten von rechts beginnt, werden nach und nach Knoten entfernt, aber am Kopf des Graphen ändert sich nichts, die Teilinstanzen haben alle Größe $\Theta(k^2)$ während des Branchingprozesses.

Verflechtung von Problemkernreduktion und Suchbäumen: Beispiel Vertex Cover

Jetzt: immer wieder Datenreduktion zwischendurch anwenden!



Sobald die zweite Kante entfernt und der Wert des Parameters um 2 verringert wurde, greift die Regel von Buss und der ganze Kopf kann entfernt werden!

Ein paar Bemerkungen zu Suchbäumen

Suchbaum-Algorithmen....

- sind eine der wichtigsten Techniken zur Bearbeitung des harten Kerns eines Problems.
- können einfach parallelisiert werden.
- können oftmals noch beschleunigt/verbessert werden, indem man sie mit Heuristiken kombiniert (z.B. Branch & Bound).
- können mit Approximationsalgorithmen kombiniert werden.

Ende Kapitel 3: Tiefenbeschränkte Suchbäume

- Einfache Suchbäume:
  - Vertex Cover $O(2^k)$
  - Independent Set auf planaren Graphen $O(6^k)$
  - Closest String $O(d^d)$
  - 3-Hitting Set $O(3^k)$
  - Übungsblätter...
- Analyse von Suchbäumen: Branchingvektor, Branchingzahl.
  - Beispiel Vertex Cover $O(1.342^k)$
- Verflechtung von Datenreduktion und Suchbäumen, Beispiel Vertex Cover/Komet