Intelligente Systeme

Einführung

Prof. Dr. Rudolf Kruse Georg Ruß
Christian Moewes

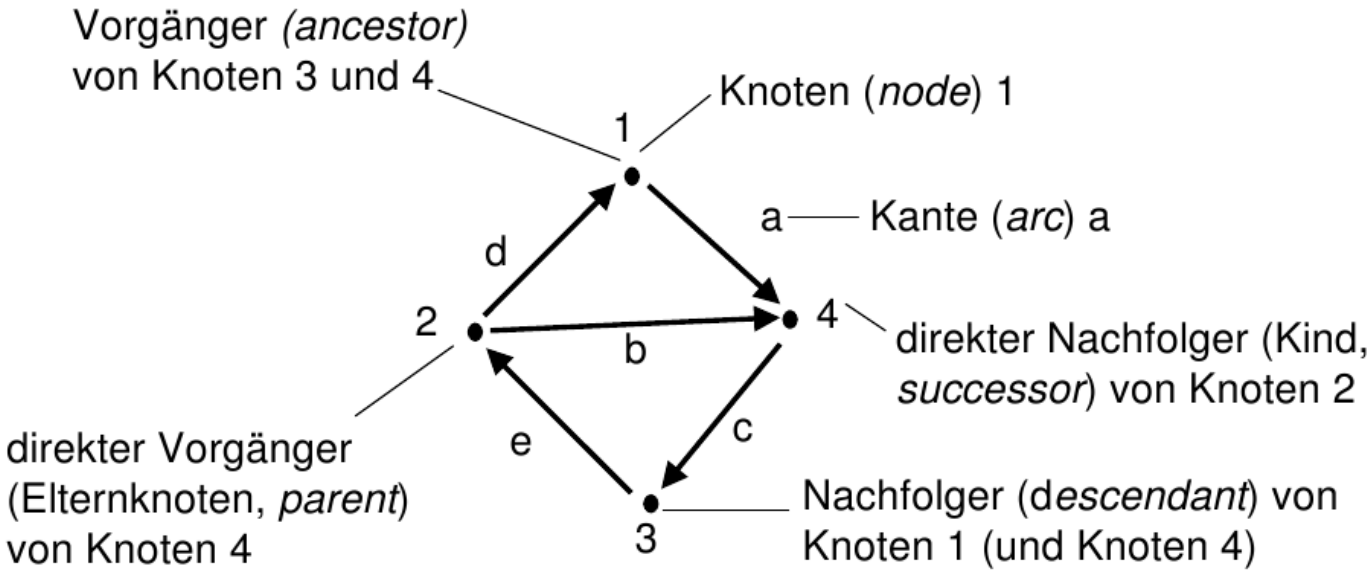
{kruse,russ,cmoewes}@iws.cs.uni-magdeburg.de

Arbeitsgruppe Computational Intelligence
Institut für Wissens- und Sprachverarbeitung
Fakultät für Informatik
Otto-von-Guericke Universität Magdeburg

Uninformierte Suche

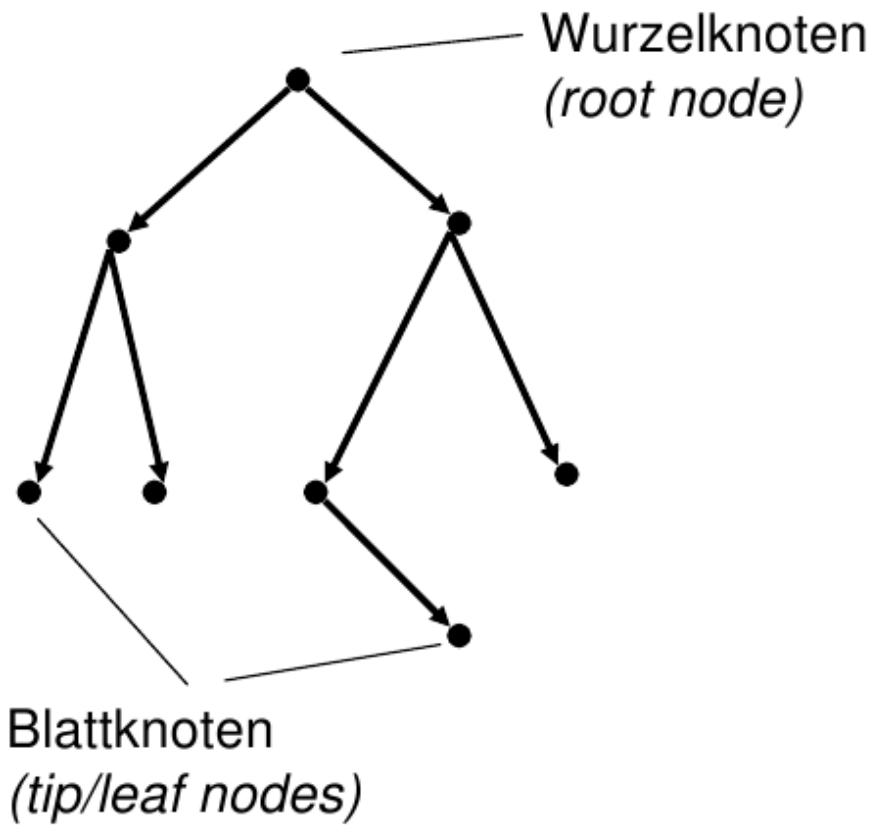
Notationen (1)

Graphnotation



Notationen (2)

Baumnotation

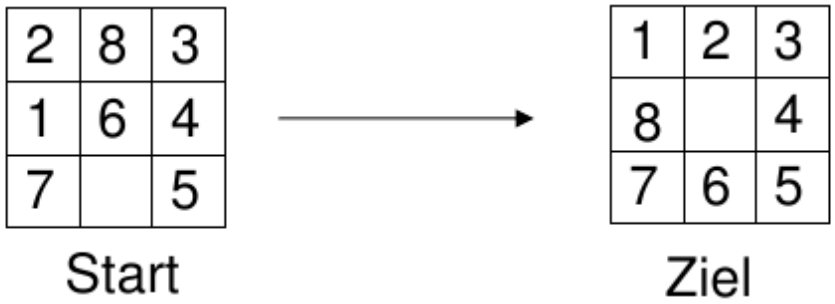


Notationen (3)

- Die Zustandsgraphen für reale Probleme sind in der Regel zu groß, als dass man sie explizit als Graph angeben könnte.
- Deshalb verwendet man meist andere Repräsentationen, d.h. man sucht
  - eine geeignete Formulierung des Problems als Suchproblem,
  - wählt eine implizite Repräsentationsform der Suchgraphen
  - und effiziente Suchalgorithmen.

Notationen (4)

Beispiel: Schiebepuzzle (8-Puzzle)



- Zustandsgraph:
  - Knoten: 3x3-Array
  - Kanten: Verschiedene Zug-Kodierungen möglich
    - 8x4 bei Verwendung der Teile (z.B. *bewege 1 links*)
    - besser (weil einfacher): 4 bei Bewegungsrichtung der Leerstelle (rechts, links, oben, unten)

Notationen (5)

Beispiel: Schiebepuzzle (8-Puzzle)

- Größe des Zustandsgraphen:
  - Die Knoten müssen alle möglichen Zustände (definiert durch Positionen der Teile) beschreiben, d.h. man benötigt $9! = 362880$ Knoten
  - Den Graphen des 15-Puzzles kann man nicht mehr direkt im Rechner speichern
- Weitere Beispiele:
  - Spielbäume: Tic-Tac-Toe, Schach etc.

Notationen (6)

Implizite Darstellung des Zustandsgraphen

- Beschreibung des Startknotens
  - Datenstruktur, die Anfangsbedingung beschreibt
- Operatoren
  - Funktionen, die den Übergang von einem Zustandes in einen anderen Zustand mittels einer Aktion beschreiben
- Zielbedingung
  - Beschreibung des Endzustandes oder Boole'sche Funktion der Zustandsbeschreibung

Breitensuche (breadth-first)

- Die Nachfolgerfunktion erzeugt für einen Zustand alle möglichen Zustände, die durch Anwendung der Operatoren erzeugt werden können.
- Hier: Operatoren in der Reihenfolge Leerstein links, oben, rechts, unten.
- Es werden keine Pfeile zurück eingezeichnet!
- *Vorteil:* Breitensuche findet Lösung mit minimaler Weglänge.
- *Nachteil:* Der erzeugte Baum wächst exponentiell in der Tiefe.

Breitensuche

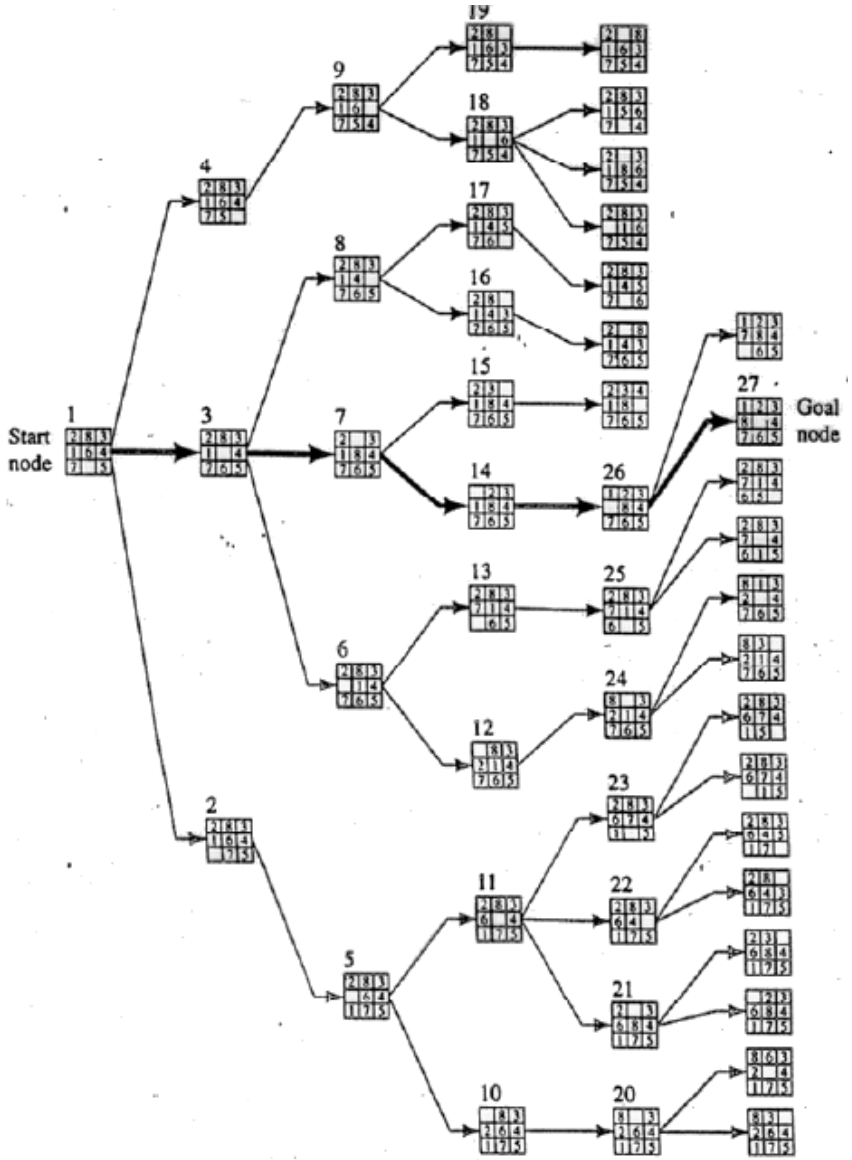
Schiebepuzzle (8-Puzzle)

Start

2	8	3
1	6	4
7		5

Ziel

1	2	3
8		4
7	6	5



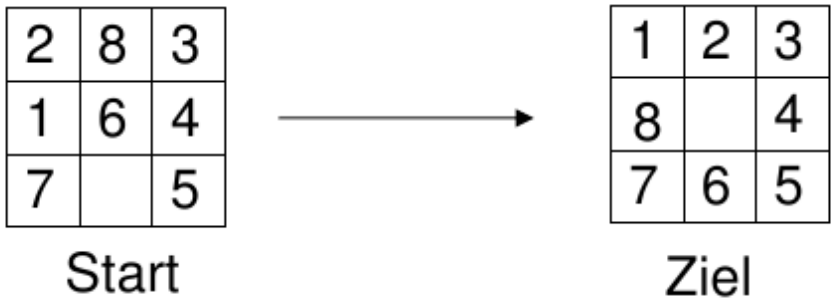
Tiefensuche

- Es wird immer nur ein Nachfolgerknoten (nicht alle) in einer festgelegten Reihenfolge der Operationen erzeugt. Die noch möglichen Operationen werden vermerkt.
- Ist ein Nachfolgerknoten erzeugt, wird als nächstes dessen Nachfolger erzeugt.
- Es wird eine Tiefenbegrenzung eingeführt.
- Vorteil: Speicherplatzbedarf ist linear zur Tiefenbegrenzung.
- Nachteil: Findet den kürzesten Weg nur zufällig; manchmal sogar "ziemlich blinde" Suche.

Tiefensuche

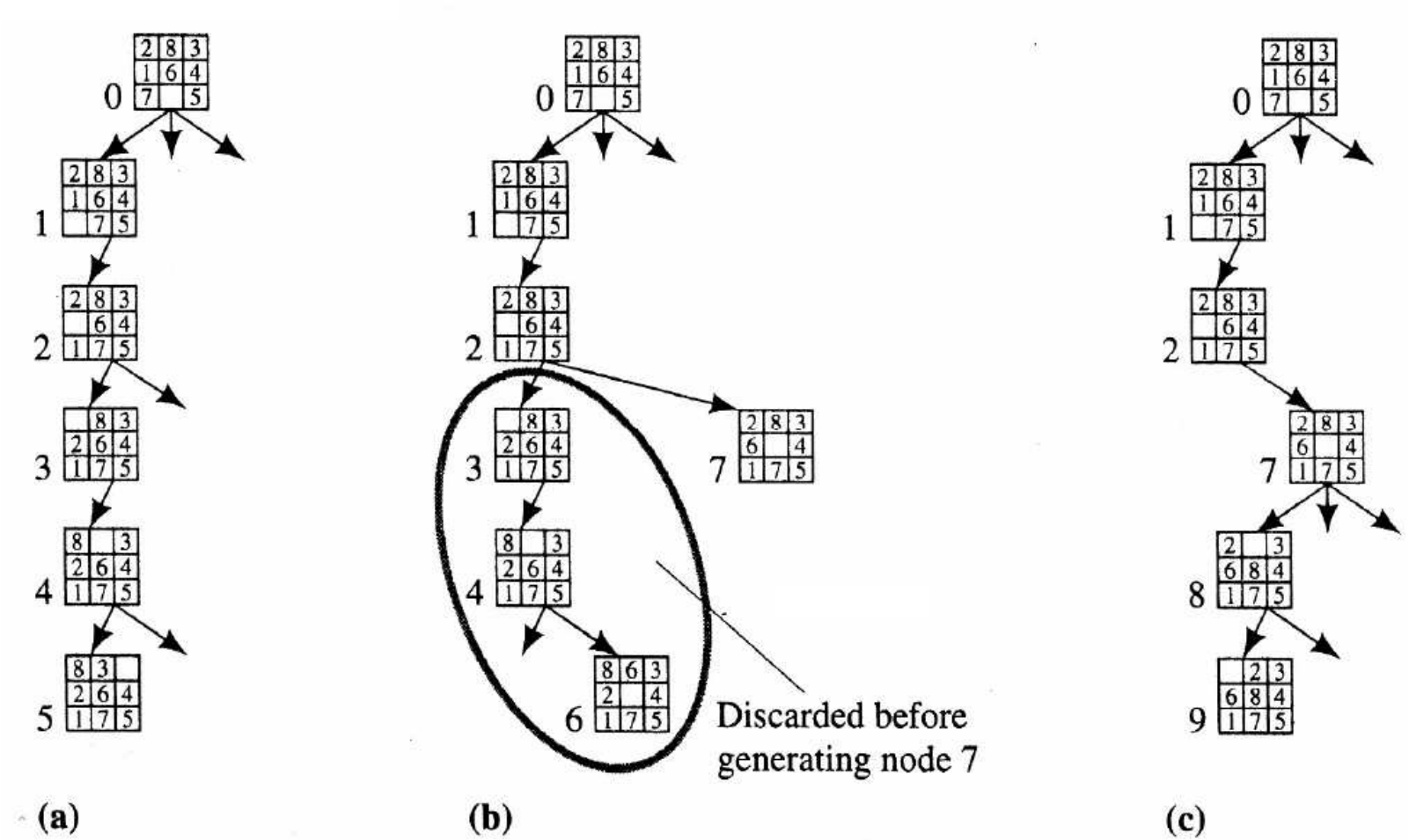
Beispiel Schiebepuzzle (8-Puzzle)

- Operationsreihenfolge: links, oben, rechts, unten
- Tiefenbegrenzung: 5



Tiefensuche

Beispiel Schiebepuzzle (8-Puzzle)



Tiefensuche

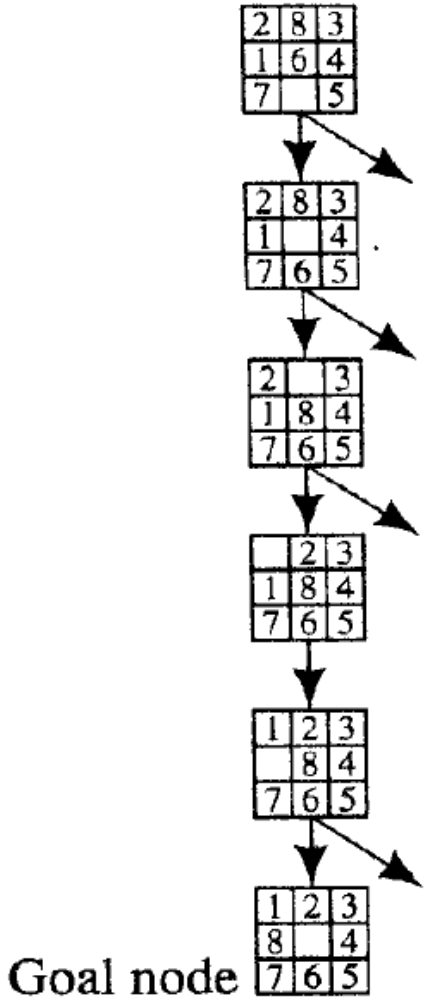
Beispiel Schiebepuzzle (8-Puzzle)

Start

2	8	3
1	6	4
7		5

Ziel

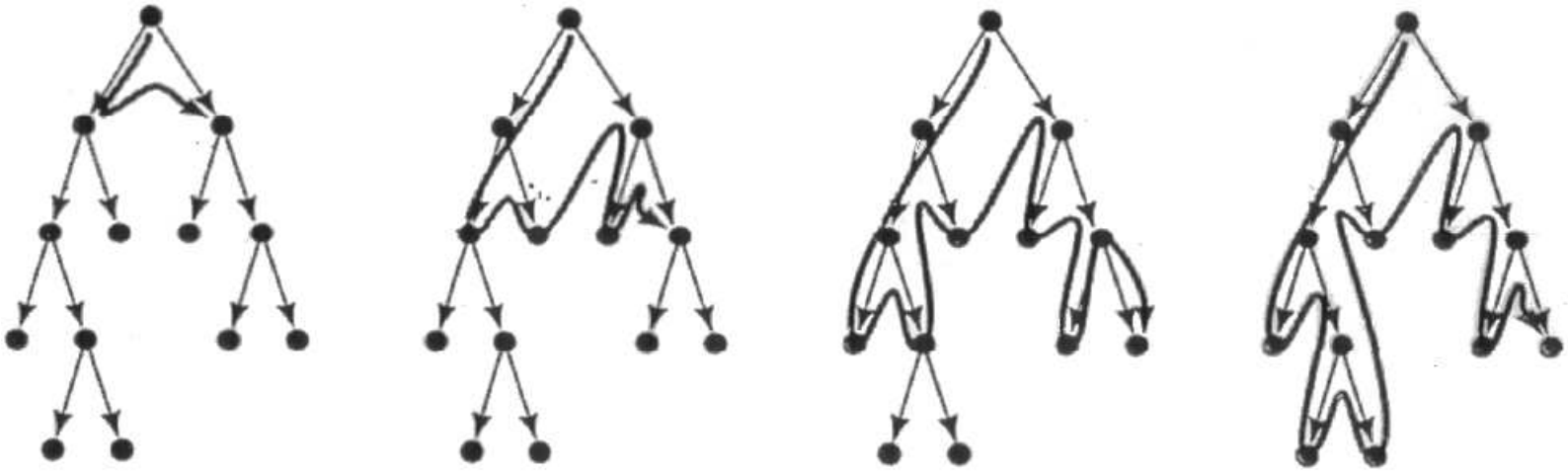
1	2	3
8		4
7	6	5



Iterierte Tiefensuche

iterative deepening

- Idee: Die Tiefenbegrenzung wächst kontinuierlich



- Findet kürzeste Wege!

Iterierte Tiefensuche

Aufwand (1)

- Beispiel: Suchbaum mit Verzweigungsfaktor b (branching factor) und Zielknoten in Tiefe d (depth)
- Bei der Breitensuche müssen

$$N_b = 1 + b + b^2 + \ldots + b^d = \frac{b^{d+1} - 1}{b - 1}$$

Knoten erzeugt werden.

Iterierte Tiefensuche

Aufwand (2)

- Bei der iterierten Tiefensuche beträgt die Anzahl der Knoten:

$$
\begin{aligned}
N_{id} &= \sum_{j=0}^{d}(1 + b + \ldots + b^j) = \sum_{j=0}^{d} \frac{b^{j+1} - 1}{b - 1} \\
&= \frac{1}{b-1} \sum_{j=0}^{d}(b^{j+1} - 1) \\
&= \frac{1}{b-1} \left\{ b(\sum_{j=0}^{d} b^j) - \sum_{j=0}^{d} 1 \right\} \\
&= \frac{1}{b-1} \left[b\frac{b^{d+1} - 1}{b-1} - (d+1) \right] \\
&= \frac{b^{d+2} - 2b - bd + d + 1}{(b-1)^2}
\end{aligned}
$$

Iterierte Tiefensuche

Aufwand (3)

- Vergleicht man N_b und N_{id}, so ergibt sich für große d:

$$
\begin{aligned}
\frac{N_b}{N_{id}} &= \frac{(b^{d+1}-1)(b-1)^2}{(b-1)(b^{d+2}-2b-bd+d+1)} \\
&= \frac{b(b^d-1)(b-1)}{b^2\left(b^d-\frac{2}{b}-\frac{d}{b}+\frac{d+1}{b^2}\right)} \\
&= \frac{(b-1)}{b}\cdot\frac{b^d-1}{b^d-\frac{2}{b}-\frac{d}{b}+\frac{d+1}{b^2}} \\
=_{d\to\infty} \quad & \frac{b-1}{b}\cdot 1 = \frac{b-1}{b}
\end{aligned}
$$

- d.h. iterierte Tiefensuche benötigt nur wenig mehr Knoten als die Breitensuche!

Iterierte Tiefensuche

Aufwand (4)

- Anschaulich:
  - Die Blattebene dominiert stets, mit wachsender Tiefe immer stärker, z.B. enthält die Blattebene für b=2 stets genau einen Knoten mehr als der Rest des Baumes.
  - Bei größerem Verzweigungsfaktor ist das Verhältnis noch günstiger für die Blattebene.