Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Informierte Suche

Prof. Dr. Michael Neitzke

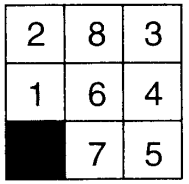
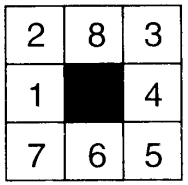
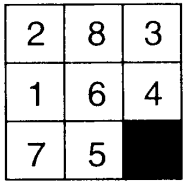
Wie Information über Zustandsqualität nutzen?

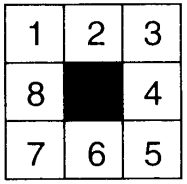
- Wie nah am Zielzustand?
- Also Information über Restkosten
- Denn bisherige Kosten sind ja bekannt
- Restkosten können nur geschätzt werden
  - Genaue Berechnung bedeutet vollständige Suche, also keine Ersparnis
  - Schätzung durch Heuristiken
- Schwierigkeit: Beste Heuristik finden

Was ist eine Heuristik?

- Daumenregel
- Vereinfachte Betrachtung

Verschiedene Heuristiken für das 8-Puzzle

	**5**	**6**	**0**
	**3**	**4**	**0**
	**5**	**6**	**0**
	Anzahl Spiel-steine auf falschen Positionen	Summe der Entfernungen, in denen sich Spielsteine auf falschen Posi-tionen zueinander befinden	2 x Anzahl direkter Spiel-steintausch



Ziel

A-Algorithmus

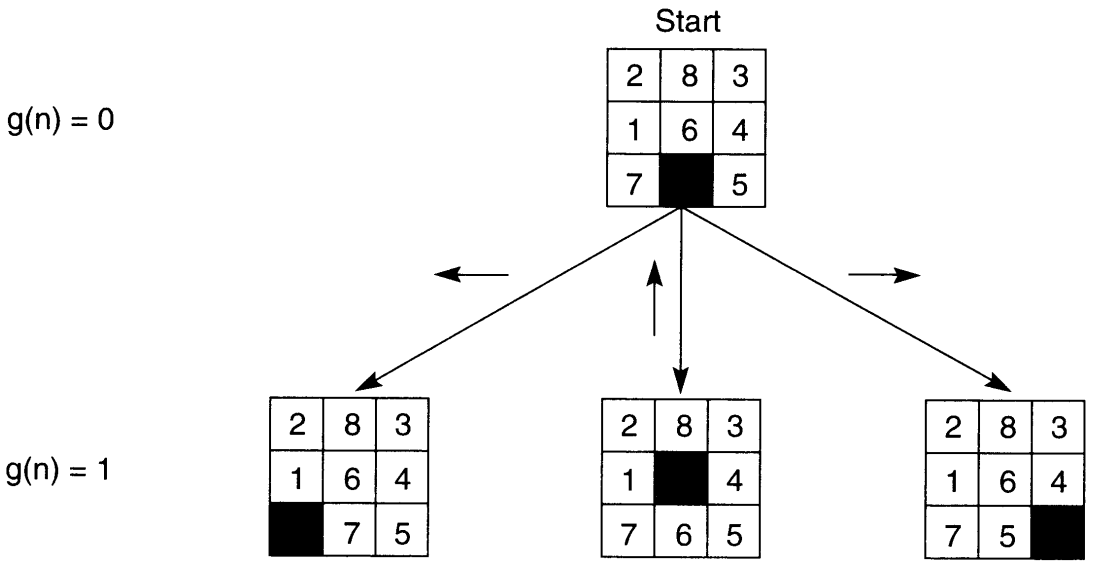
Gesamtkosten f(n) eines Knotens n =

bisherige Kosten g(n)

\+ geschätzte Restkosten h(n)

$$f(n) = g(n) + h(n)$$

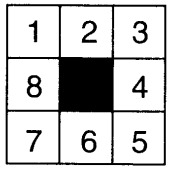
Kosten bei Heuristik „Anzahl falsche Positionen“



Werte von f(n) für jeden Zustand, **6** **4** **6**

wobei:

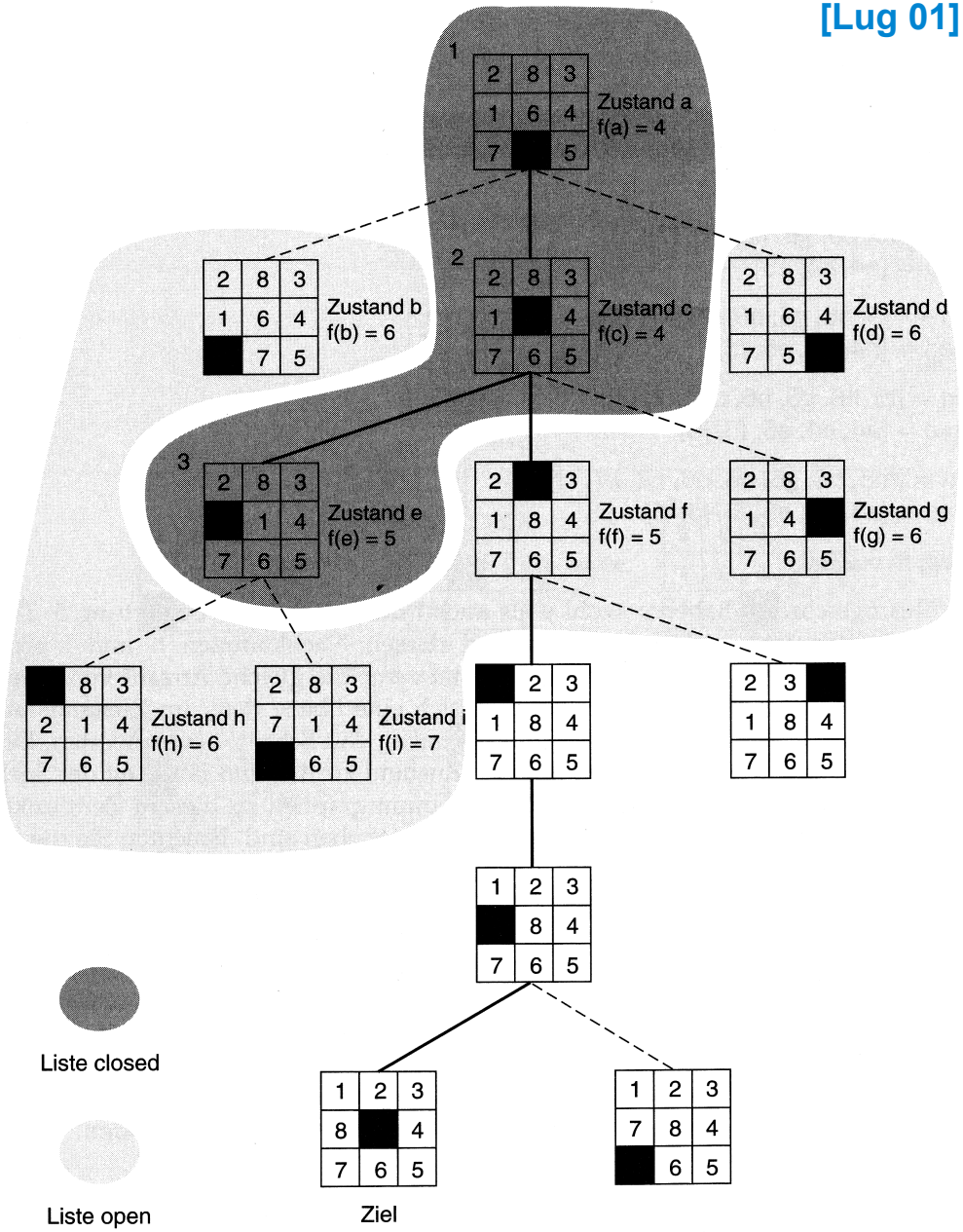
f(n) = g(n) + h(n),
g(n) = tatsächliche Entfernung von n zum Startzustand, und
h(n) = Anzahl Spielsteine auf falschen Positionen.



Ziel

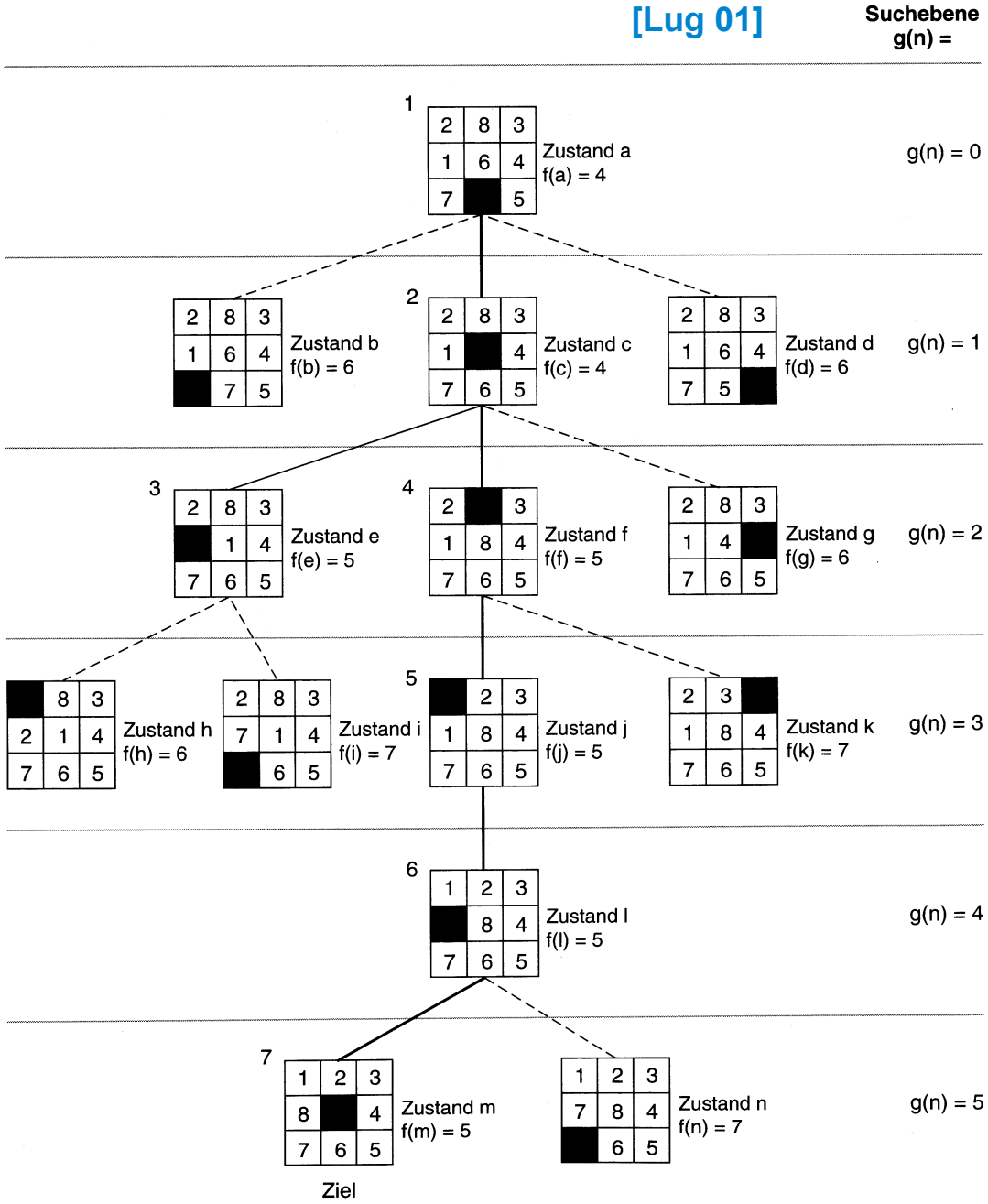
Nach 3 Schritten

- Geschlossene Knoten:
  Alle Kindknoten sind bekannt
- Offene Knoten:
  Noch nicht expandiert,
  also Kindknoten nicht
  bekannt
- Grau unterlegte Mengen
  betreffen die Situation
  nach 3 Schritten



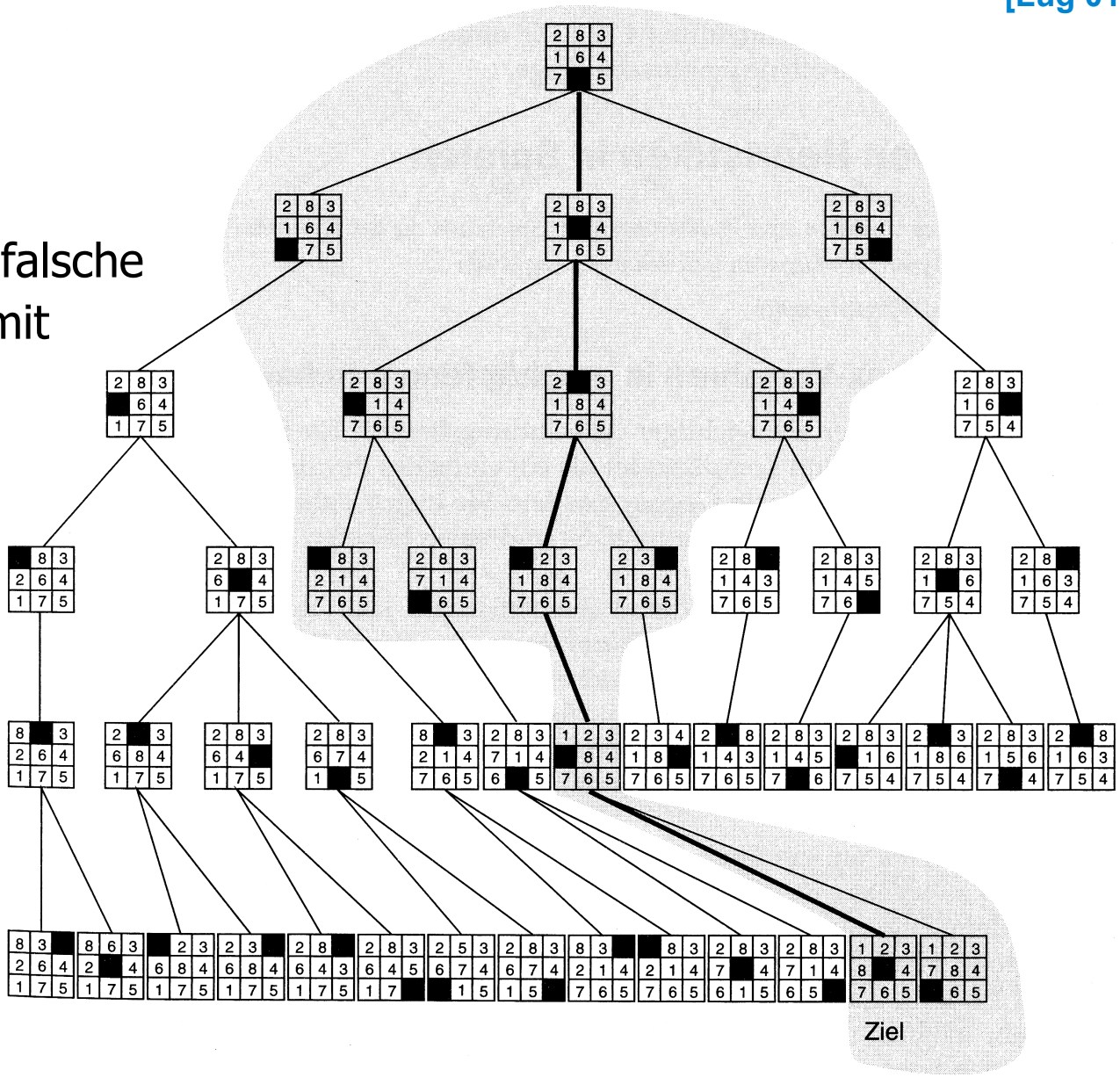
Vollständiger Baum

- Gestrichelte Linien: Pfade, die nicht weiter untersucht werden, weil die Knoten zu ungünstig bewertet sind



Vergleich

- Heuristik „Anzahl falsche Positionen“ wird mit Breitensuche verglichen



Diskutieren Sie!

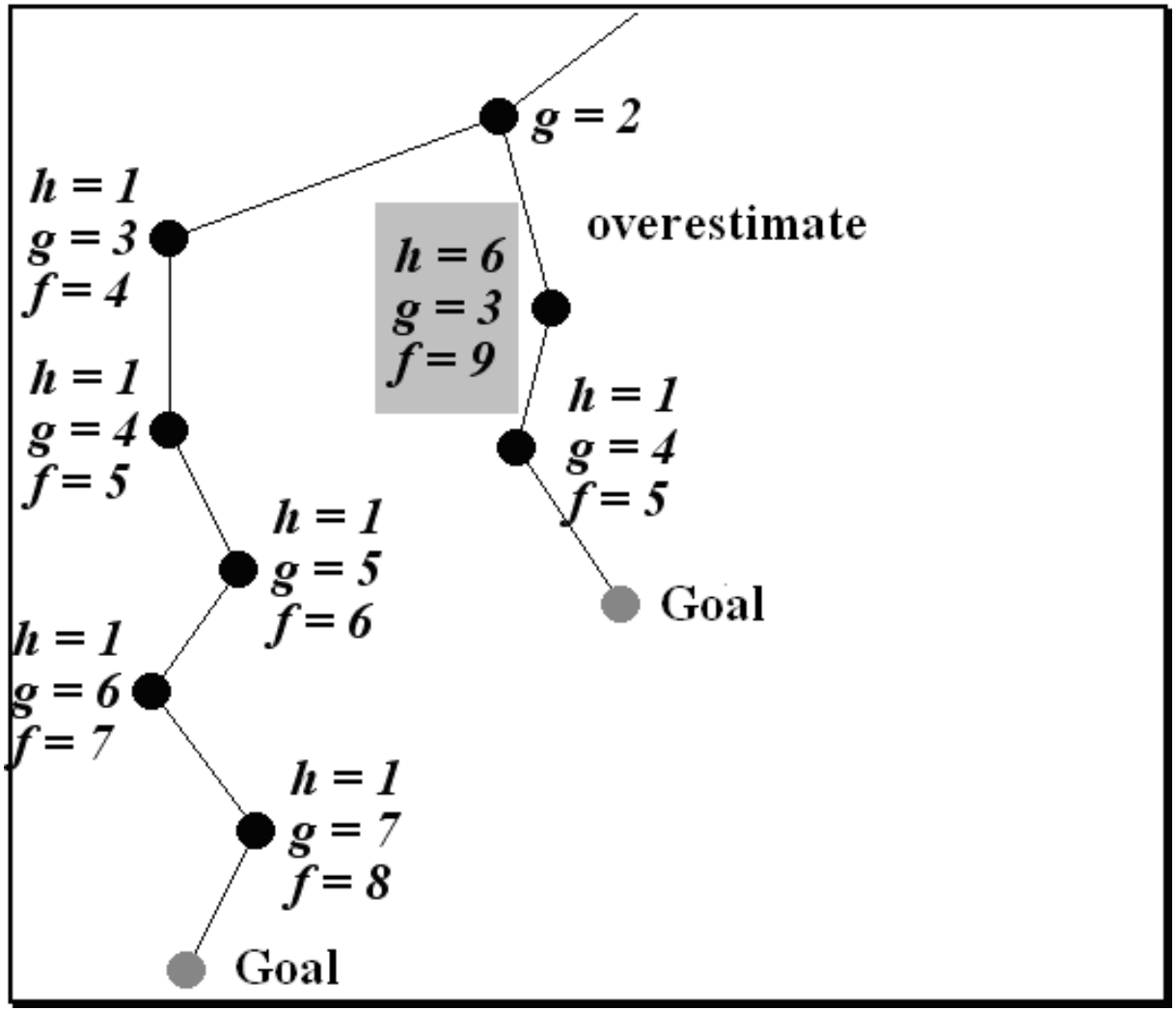
- Wie wirkt es sich (im ungünstigen Fall) aus, was bedeutet es, wenn die Restkosten
  - potenziell überschätzt werden?
  - nie überschätzt werden?
  - immer auf 0 geschätzt werden?
  - ganz exakt berechnet werden?

Lösung

- Überschätzung:
  Der günstigste Weg zum Ziel wird ignoriert, wenn die Restkosten deutlich zu hoch eingeschätzt werden. So kann es passieren, dass ein anderes, ungünstigeres Ziel gefunden wird. Da auch immer die bisherigen Kosten zu Buche schlagen, kann das Verfahren nicht beliebig lange auf einem ungünstigen Weg bleiben.
- Unterschätzung:
  Hohe Unterschätzung bedeutet wenig Information. Es wird wenig zielgerichtet vorgegangen, es werden viele Pfade untersucht. Aber: Die beste Lösung kann nicht verfehlt werden.
- Restkosten = 0:
  Dies ist der Extremfall der uninformierten Breitensuche
- Restkosten exakt berechnet:
  Das setzt eine aufwändige Berechnung voraus, also keine Ersparnis durch Heuristik.

Überschätzung der Restkosten

- Günstigstes Ziel (rechts) wird verfehlt



A*-Algorithmus: Zulässigkeit

- Schätzfunktion / Heuristik ***h*** heißt **zulässig**, wenn sie die Restkosten nicht überschätzt:

$$g(\text{Zielknoten}) - g(n) \geq h(n), \text{ für alle } n, \text{ bzw.}$$

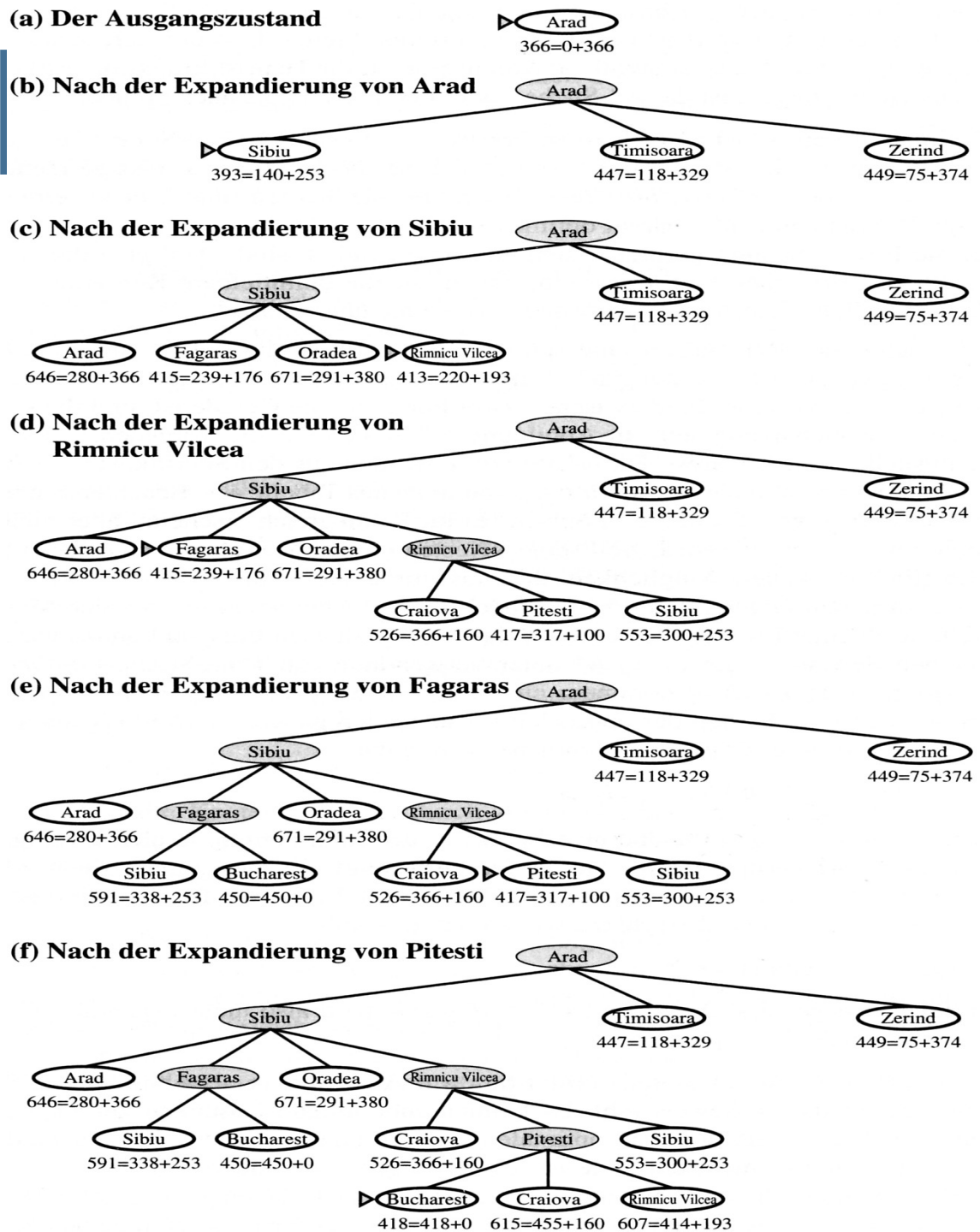
$$h^*(n) \geq h(n), \text{ für alle } n,$$

(*h**: Funktion der tatsächlichen Kosten)

- **A*-Algorithmus:** *h* ist zulässig
- A* ist optimal
- Für zwei zulässige Funktionen h_1, h_2 gilt:

Wenn $h_1(n) \geq h_2(n)$ für alle n, dann ist h_1 informierter

A*-Suche: Beispiel



Monotonie (Konsistenz) der Schätzfunktion

- Heuristik ist monoton (konsistent), wenn die Kosten zwischen **zwei beliebigen Knoten** nie überschätzt werden
  - Das heißt: Die Werte von f entlang eines Pfades dürfen nicht fallen
- Im Detail:
  - Annahme: Knoten liegt irgendwo auf optimalem Pfad
  - Überschätzung kann bedeuten, dass zunächst nicht-optimaler Pfad zu diesem Knoten gefunden wird
  - Wegen Zulässigkeit wird dann aber später auch der optimale Pfad gefunden
  - Ohne Überprüfung der Wegekosten würde der optimale Pfad keine Berücksichtigung finden
- Zulässige Heuristiken sind meistens auch monoton (Gegenbeispiele schwer zu finden)

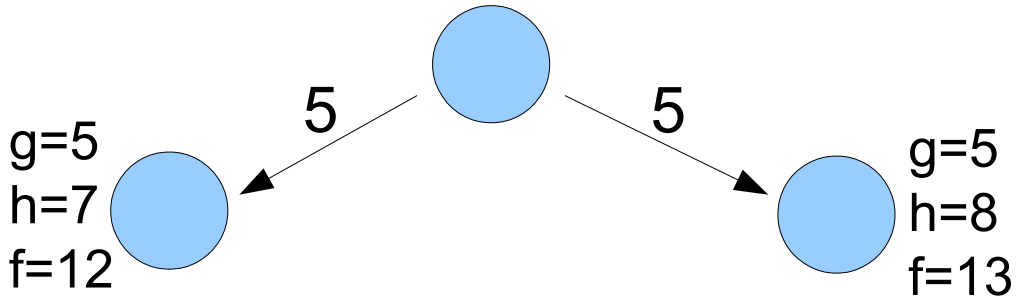
Erläuterungen

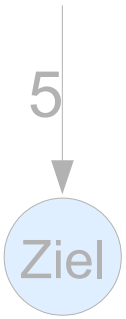
Monotonie bedeutet, dass nicht nur die Kosten bis zum Zielzustand (das wäre Zulässigkeit) nicht überschätzt werden dürfen, sondern außerdem auch die Kosten zu beliebigen anderen Zuständen.

Wenn man sich zu einem Nachfolge-Zustand bewegt, erfährt man ja die tatsächlichen Kosten. In der Gesamtkostenbetrachtung werden dann also die bisher geschätzten Kosten zu diesem Nachfolge-Zustand durch die tatsächlichen ersetzt. Wenn für jede Teilstrecke keine Überschätzung vorliegt, also typischerweise eine leichte Unterschätzung, dann führt das Ersetzen der geschätzten Kosten für eine Teilstrecke durch die tatsächlichen Kosten zu einer Erhöhung der Gesamtkosten. Auf dem Weg vom Startzustand zum Zielzustand erhöhen sich also die Gesamtkosten von Schritt zu Schritt immer weiter. Daher wird eine Heuristik, die auch Teilstrecken nicht überschätzt, als monoton bezeichnet.

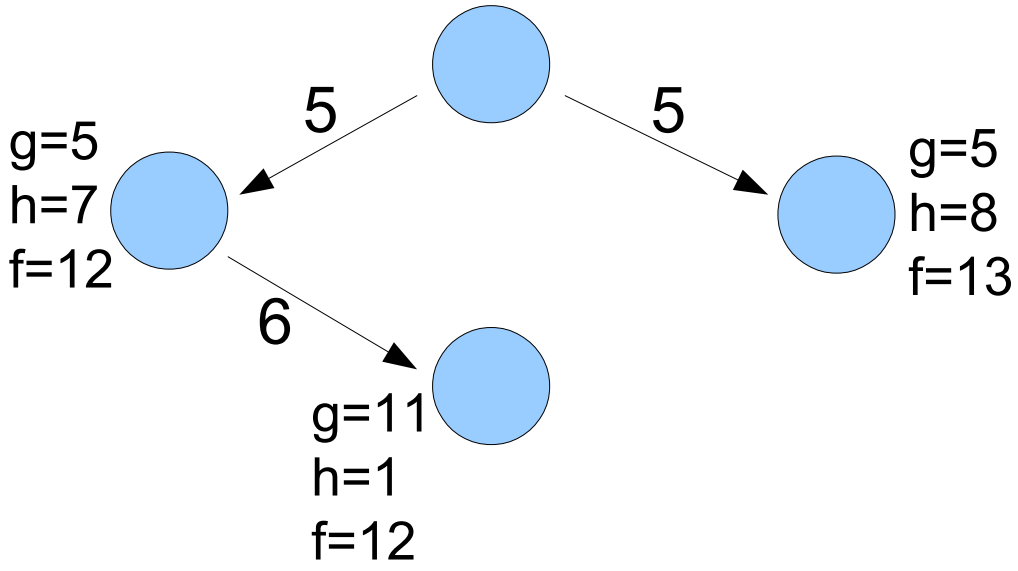
Wenn eine Heuristik zwar zulässig, aber nicht monoton ist, können also Teilstrecken überschätzt werden. Dies kann dazu führen, dass zu einem innen liegenden Zustand nicht der optimale (also nicht der kostengünstigste) Pfad gefunden wird. Aufgrund der Zulässigkeit der Heuristik, würde zu diesem Zustand, wenn er denn auf dem optimalen Pfad zum Zielzustand liegt, später auch der optimale Pfad für diese Teilstrecke gefunden werden. Zu diesem innen liegenden Zustand würde also ein zweiter, kostengünstigerer Weg gefunden werden. Dies erfordert ein Update der Kosten zu dem Zustand und nachfolgender Berechnungen. Wenn eine Heuristik nicht monoton, wohl aber zulässig ist, muss man sich also um eine Aktualisierung der Kosten zu den Zuständen aus der Closed-List und ihrer Nachfolger kümmern. Dies ist bei einer monotonen Heuristik nicht nötig.

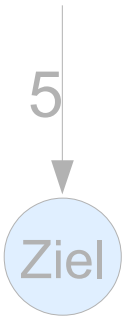
Beispiel: Zulässige, nicht-monotone Schätzfunktion



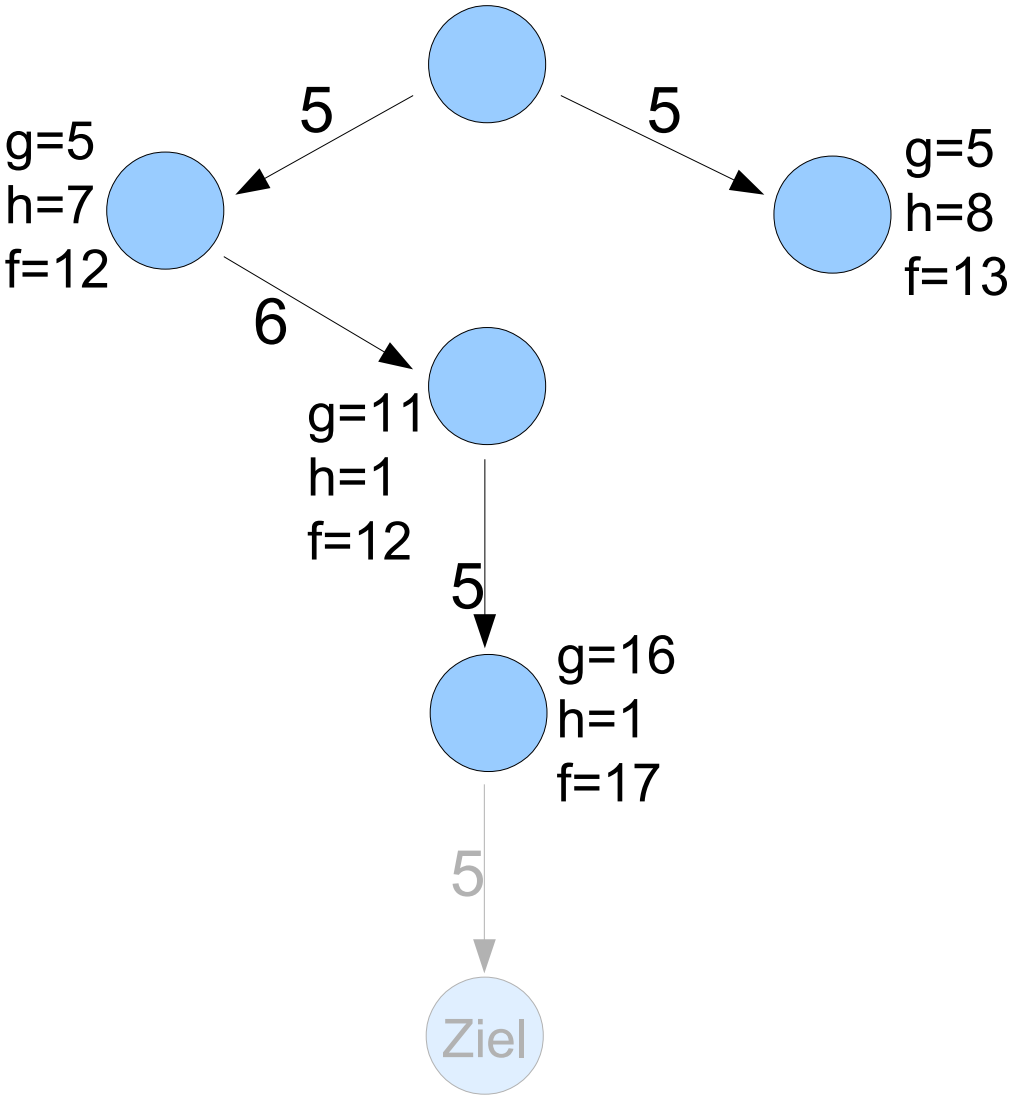


Beispiel: Zulässige, nicht-monotone Schätzfunktion

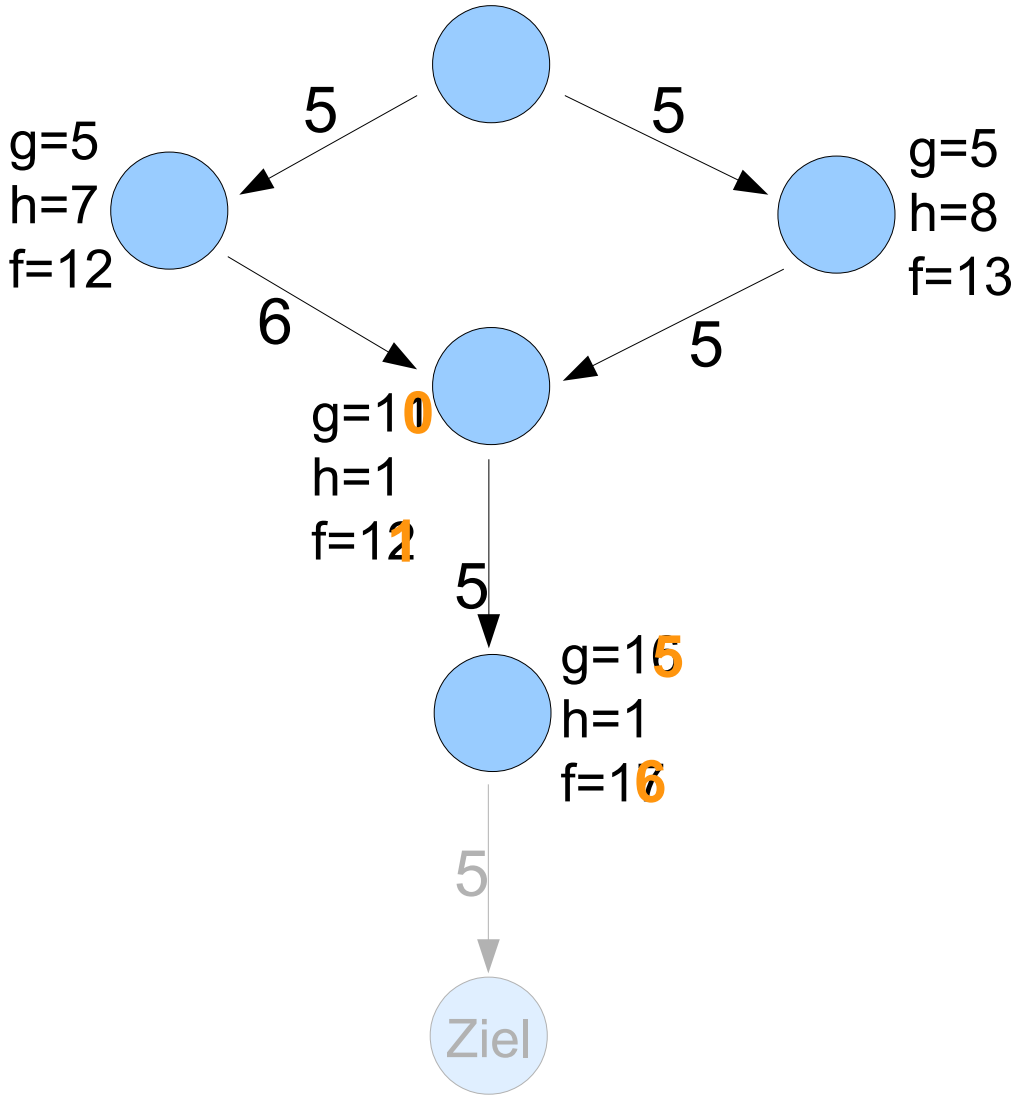




Beispiel: Zulässige, nicht-monotone Schätzfunktion



Beispiel: Zulässige, nicht-monotone Schätzfunktion

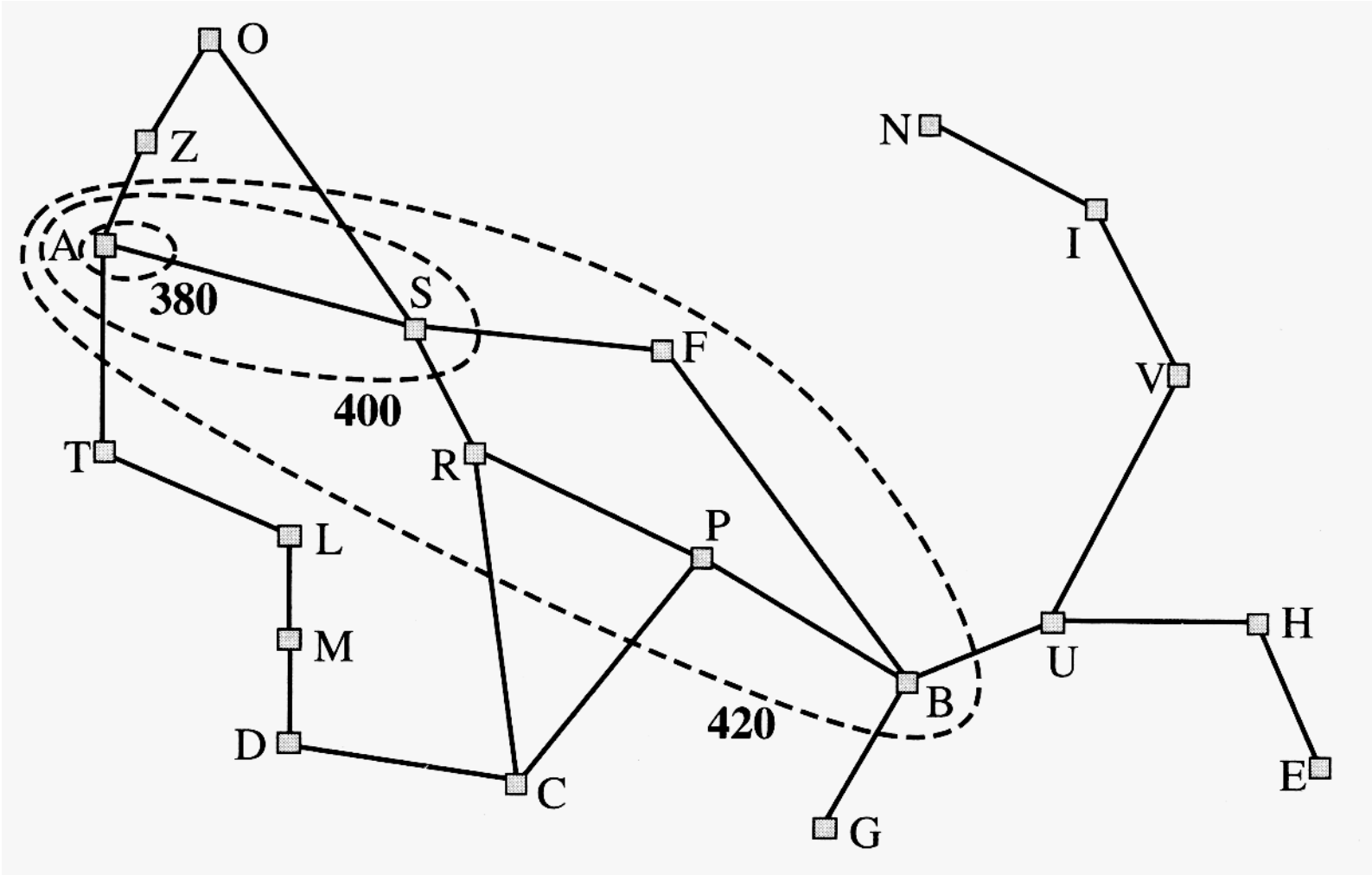


A*: Bewertung von Heuristiken

- Bestmögliche Heuristik ist unsinnig
  - Tatsächliche Kosten müssen berechnet werden (die Aufgabe, die eigentlich vereinfacht werden sollte)
- h(n) = 0 entspricht Breitensuche
- Kompromiss erforderlich
- Gute Heuristiken haben effektiven Verzweigungsfaktor nahe 1
  - Sei N Anzahl der erzeugten Knoten bis Lösung in Tiefe t gefunden wurde:

$$N+1 = 1 + v_{eff} + v_{eff}^2 + v_{eff}^3 + \ldots + v_{eff}^t$$

Gute A*-Heuristik: Geringer Verzweigungsfaktor



Vergleich verschiedener Heuristiken

- h1: Anzahl Spielsteine auf falschen Positionen
- h2: Summe der Entfernungen

	Suchkosten			Effektiver Verzweigungsfaktor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2,45	1,79	1,79
4	112	13	12	2,87	1,48	1,45
6	680	20	18	2,73	1,34	1,30
8	6384	39	25	2,80	1,33	1,24
10	47127	93	39	2,79	1,38	1,22
12	3644035	227	73	2,78	1,42	1,24
14	–	539	113	–	1,44	1,23
16	–	1301	211	–	1,45	1,25
18	–	3056	363	–	1,46	1,26
20	–	7276	676	–	1,47	1,27
22	–	18094	1219	–	1,48	1,28
24	–	39135	1641	–	1,48	1,26

Abbildung 4.8: Vergleich der Suchkosten und der effektiven Verzweigungsfaktoren für die ITERATIVE-DEEPENING-SEARCH- und A*-Algorithmen mit h_1 und h_2. Die Daten wurden über 100 Instanzen des 8-Puzzles für verschiedene Lösungslängen gemittelt.

Automatische Erzeugung zulässiger Heuristiken!

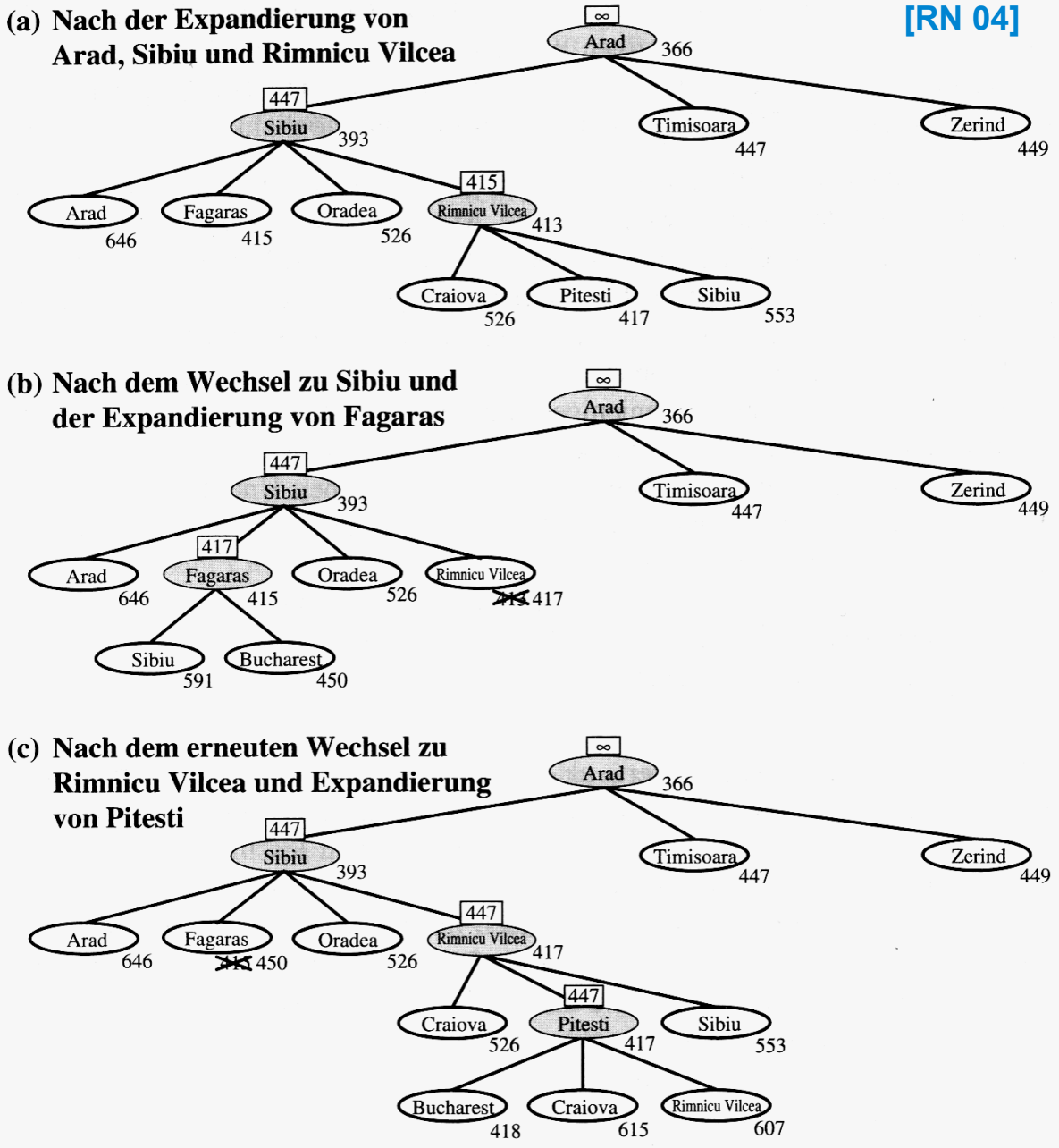
- Möglich, wenn Suchproblem in formaler Sprache ausgedrückt wird
- Methode (u. a.): Erzeugung "gelockerter" Probleme, dann Formulierung entsprechender Heuristik
  - Heuristik entspricht der **genauen** Pfadlänge eines gelockerten Problems
  - Beispiel 8-Puzzle: Anzahl Spielsteine auf falschen Positionen ist Pfadlänge, wenn beliebige Sprünge des leeren Feldes erlaubt sind
  - Wenn der Heuristik ein gelockertes Problem zugeordnet werden kann, ist sie zulässig
- Programm ABSOLVER hat beste bekannte Heuristik für 8-Puzzle und erste sinnvolle Heuristik für Zauberwürfel gefunden

A*: Bewertung des Aufwands

- Für die meisten Probleme exponentiell
  - Anzahl der expandierten Knoten wächst exponentiell in Abhängigkeit von der Tiefe bis zum Zielknoten
- Primäres Problem: Speicherbedarf
- Verbesserung durch Modifikation des Algorithmus
  - Weniger Speicherbedarf zu Lasten der Rechenzeit
  - Nach wie vor optimal und vollständig
  - RBFS (Recursive Best First Search)
  - IDA* (Iterative Deepening A*)
    - Tiefensuche !

RBFS

[RN 04]


(a) Nach der Expandierung von Arad, Sibiu und Rimnicu Vilcea
∞ Arad 366
447 Sibiu 393
Timisoara 447
Zerind 449
Arad 646
Fagaras 415
Oradea 526
415 Rimnicu Vilcea 413
Craiova 526
Pitesti 417
Sibiu 553
(b) Nach dem Wechsel zu Sibiu und der Expandierung von Fagaras
∞ Arad 366
447 Sibiu 393
Timisoara 447
Zerind 449
Arad 646
417 Fagaras 415
Oradea 526
Rimnicu Vilcea 417
Sibiu 591
Bucharest 450
(c) Nach dem erneuten Wechsel zu Rimnicu Vilcea und Expandierung von Pitesti
∞ Arad 366
447 Sibiu 393
Timisoara 447
Zerind 449
Arad 646
Fagaras 450
Oradea 526
447 Rimnicu Vilcea 417
Craiova 526
447 Pitesti 417
Sibiu 553
Bucharest 418
Craiova 615
Rimnicu Vilcea 607

Erläuterung IDA*

- Zunächst wird nur der Startknoten expandiert
- Der günstigste Wert **f(n)** aller Kindknoten wird global gespeichert
- Neustart mit gespeichertem Wert als Schranke:
  - Alle Knoten mit **f(n) ≤ Schranke** werden expandiert, und zwar in der Reihenfolge einer Tiefensuche
  - Der günstigste Wert **f(n)** eines Blattknotens wird als neue Schranke gespeichert
  - Erneuter Neustart mit neuer Schranke
- Wert der Schranke wächst mit jeder Iteration, da Kosten ja unterschätzt werden
- Achtung: IDA* ist eine Tiefensuche.
  - Pfade und Knoten, die im Rahmen des Backtracking nicht weiter verfolgt werden, werden gelöscht

Bewertung RBFS, IDA*

- Jetzt ist Zeitaufwand limitierend
  - Unnötig wenig Speicherbedarf
- Verbesserung:
  - Verfügbaren Speicher nutzen, erst dann mehr rechnen
  - MA* (Memory-bounded A*)
  - SMA* (Simplified MA*)

SMA*

- Arbeitsweise:
  - Wie A* bis Speicher voll ist
  - Dann Löschen des schlechtesten Unterbaums mit Übertragung des besten Wertes auf den Vorgänger (wie RBFS)
- Nachteil:
  - Bei bestimmten Problemen abwechselndes Untersuchen bzw. Wegschalten konkurrierender Pfade
    - Potenziell hoher Rechenaufwand sobald Speichergrenze erreicht
- Einziger Ausweg:
  - Verzicht auf Optimalität oder Vollständigkeit

Erläuterung

- Zum Nachteil von SMA*:
  Das Löschen von Teilbäumen und Wechseln zu einem anderen Ast ist ja nur dann hilfreich, wenn der neue Ast wenigstens für einige Schritte weiter expandiert wird. Wenn jedoch bereits nach ein, zwei Schritten klar wird, dass der erste Ast doch der bessere war und nach dem Löschen des zweiten Asts und ein, zwei Schritten festgestellt wird, dass nun der zweite Ast wieder günstiger erscheint, so ergibt sich ein ständiges Löschen und Wiederherstellen von Ästen ohne dass nennenswerter Fortschritt in der Suche erreicht wird.

Lernen, besser zu suchen?

- Erlernen der besten Suchstrategie ist möglich
- Idee: Auswertung der schrittweisen Expansion des Zustandsbaums aus übergeordneter Sicht
  - Zustandsraum auf Metaebene
  - Was sind die Merkmale der Teilbäume, die vergeblich untersucht wurden?

Diskutieren Sie!

- Unter welchen Bedingungen könnte es nützlich sein, die bisherigen Kosten nicht zu berücksichtigen?

Sonderfall: *f(n) = h(n)*

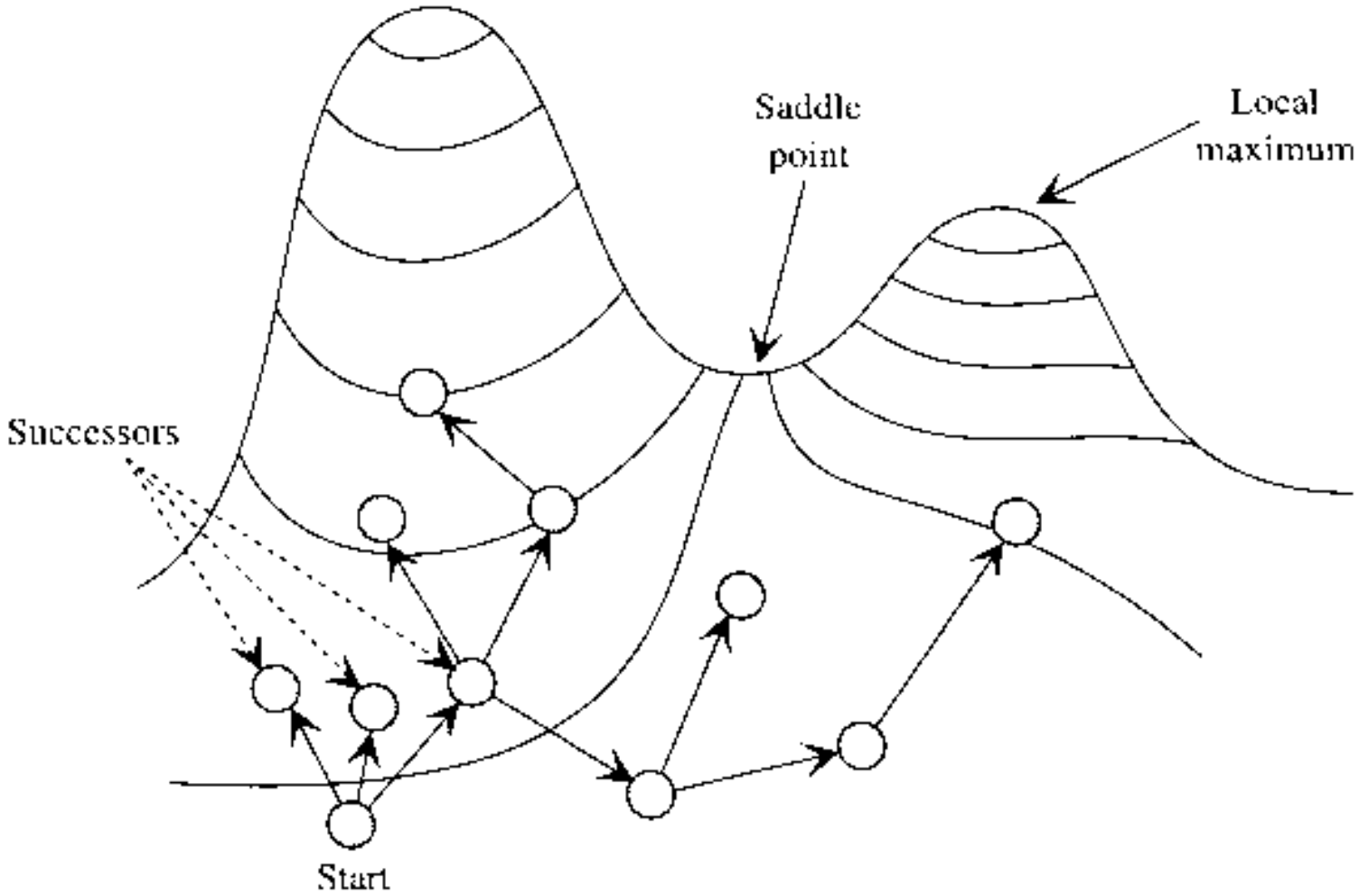
- Nützlich
  - *für bestimmte Suchprobleme zur Bestimmung von Lösungspfaden???*
  - wenn nicht der Pfad, sondern ein Zielzustand gesucht wird
  - für Optimierungsprobleme (h(n) liefert Wert des Zustands)

- Optimistisches Bergsteigen
- Bergsteigen mit Backtracking
- Gierige Bestensuche

Optimistisches Bergsteigen (Hill Climbing)

- Sonderform der Tiefensuche
- Nur direkte Nachfolger des aktuellen Knotens werden betrachtet
- Nachfolger mit geringstem Wert von ***h*** wird gewählt (steilster Anstieg)
- Nachfolger ohne Verbesserung gegenüber aktuellem Zustand scheiden aus
- Häufig sehr effizient
- Extrem geringer Speicherbedarf (maximale Verzweigungsrate)
- Nicht vollständig
- Problem: lokale Maxima

Problem der lokalen Maxima



Umgang mit lokalen Maxima: Sprünge (1)

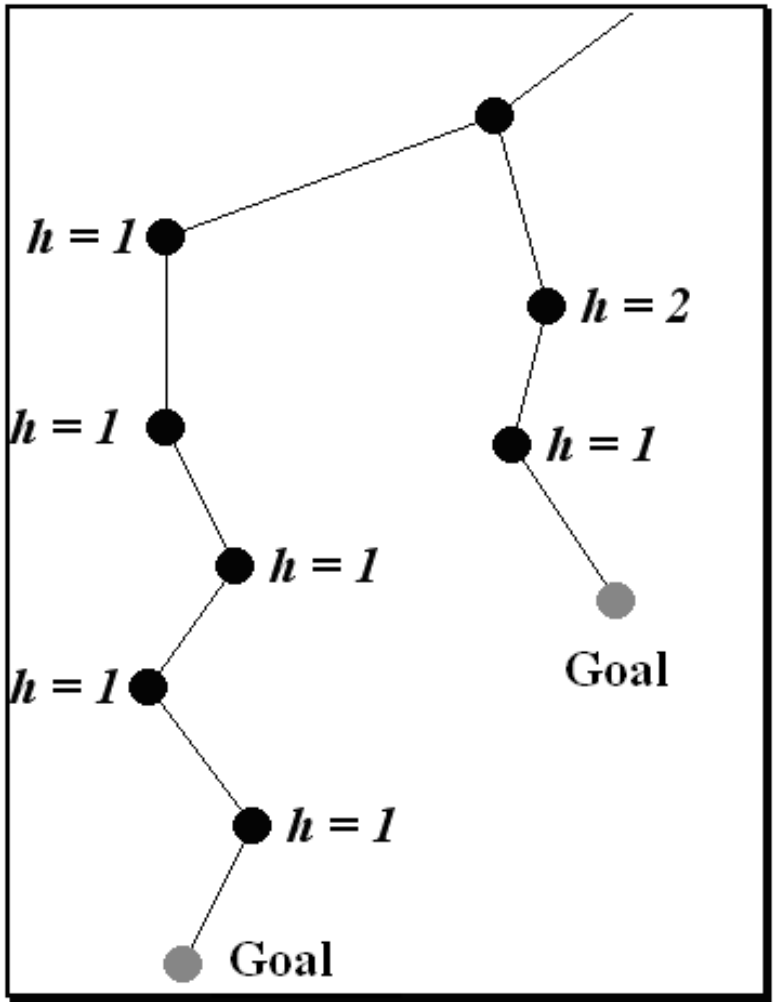
- Stochastisches Bergsteigen
  - Für bestimmten Prozentsatz der Schritte:
    - Zufällige Auswahl des Nachfolge-Zustands
    - Zufälliger Sprung
    - Zufälliger Neustart
- Simulated Annealing
  - Zufällige Sprünge mit abnehmender Sprungweite

Umgang mit lokalen Maxima: Sprünge (2)

- Stochastische Strahlsuche
  - Mehrere (k) Startpunkte gleichzeitig
  - Nächste Runde jeweils mit den k besten Nachfolgern
  - Stochastische Strahlsuche: Je schlechter der Wert des Zustands, desto höhere Wahrscheinlichkeit für Sprung
- Variante der stochastischen Strahlsuche: Genetische Algorithmen
  - Kombination von zwei günstigen Zustände zu einem neuen Nachfolger plus zufälliger Mutation
  - Anfangs große Schritte im Zustandsraum (weil Eltern sehr unterschiedlich), später kleinere Schritte (Gemeinsamkeit mit Simulated Annealing)

Probleme durch Wegfall der bisherigen Kosten

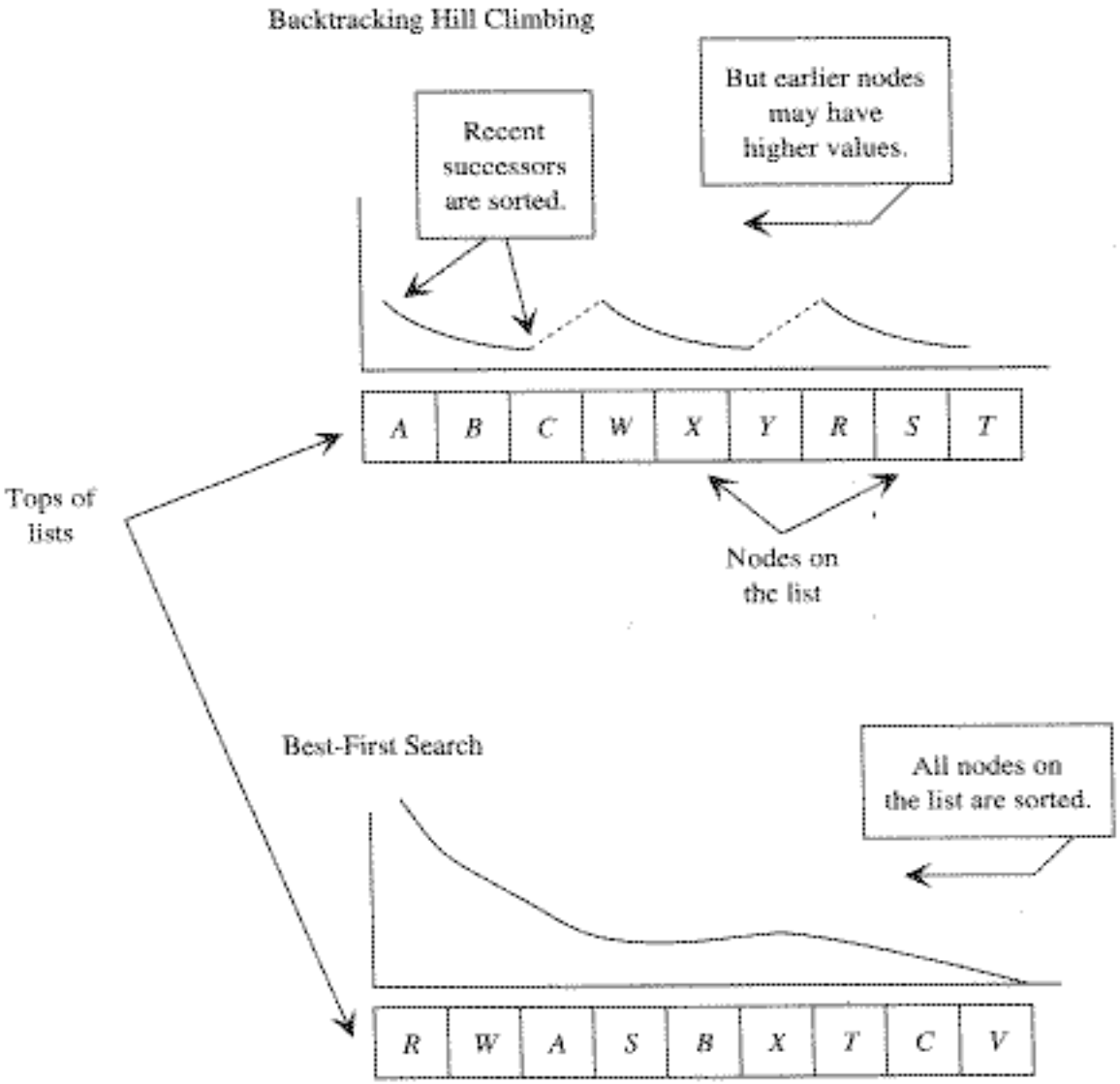
- Optimale Lösung kann verfehlt werden
- Lösung insgesamt kann verfehlt werden (Verlust der Vollständigkeit)
  - unendlich lange Pfade mit guter Bewertung ***h***



Bergsteigen mit Backtracking

- Informierte Tiefensuche ohne Berücksichtigung der bisherigen Kosten (***g***)
- Geschwister-Knoten jeweils untereinander sortiert
- LIFO-Prinzip für die Gruppen von Geschwister-Knoten
- Backtracking
- Nicht vollständig

Bergsteigen m. Backtr. vs. Gierige Bestensuche



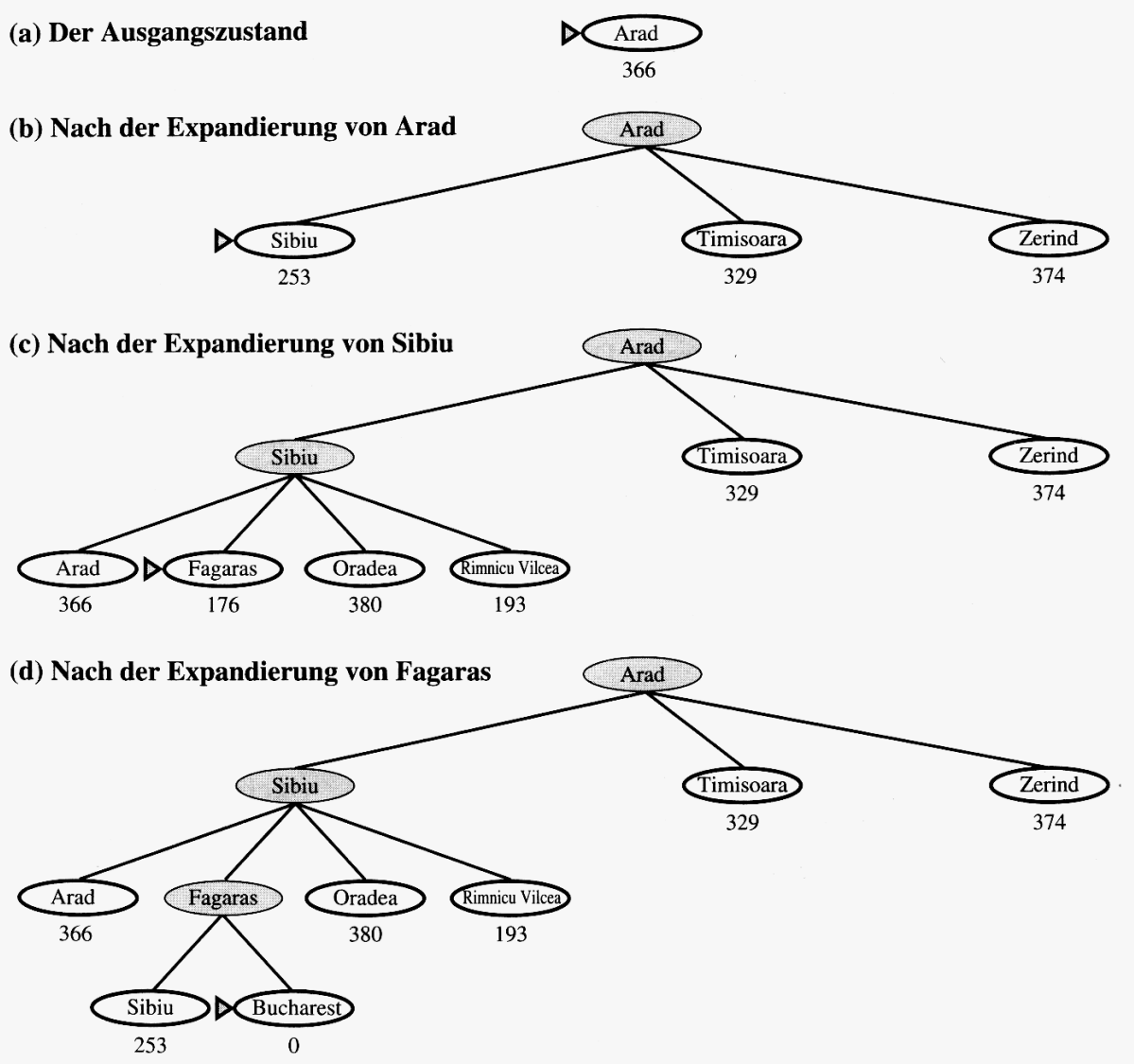
Erläuterung

- Bei der gierigen Bestensuche werden also alle Knoten der Open List sortiert, genau wie bei A*. Der Unterschied zu A* besteht lediglich darin, dass die bisherigen Kosten g(n) ignoriert werden.
- Bergsteigen mit Backtracking ist ein Tiefensuchverfahren. Die bei der Expansion eines Knotens erzeugten Kindknoten werden also immer vorn in die Open List eingefügt, allerdings im Gegensatz zur uninformierten Suche jeweils untereinander sortiert. So besteht die Open List also aus sortierten Teil-Listen von Geschwisterknoten, zuerst kommen die im Baum am weitesten unten liegenden Geschwisterknoten, dann die auf der Ebene darüber, usw.

Gierige Bestensuche

- Keine Berücksichtigung der bisherigen Kosten (***g***)
- Alle Knoten sortiert
  - Also wie A* ohne bisherige Kosten
- Nicht vollständig
  - Eine anhaltend zu optimistische Restschätzung eines schlechten Pfades führt dazu, dass dieser Pfad beliebig lange verfolgt wird.

Gierige Bestensuche: Beispiel



Wie kann der Stoff nachbereitet werden?

- Russell, Norvig: "Künstliche Intelligenz – Ein moderner Ansatz", Kapitel 4
  - Sehr ausführlich
  - Gut zu lesen

S2: Lernziele

- V1: Eigenschaften, Vor- und Nachteile, Arbeitsweise der folgenden Suchverfahren erläutern können, sie untereinander vergleichen können und Alternativen nennen können, um mit Nachteilen umzugehen::
  - A*, IDA*, RBFS, SMA*
  - Optimistisches Bergsteigen, Bergsteigen mit Backtracking, Gierige Bestensuche

S3: Allgemeiner Suchalgorithmus

Diskutieren Sie!

- Welche Lösungsbestandteile muss ein allgemeines Suchverfahren aufweisen?

Allgemeiner Suchalgorithums: Pseudocode

```
procedure search;
  paths:= [[start1],[start2],..,[startn]];
  success:= false;
  repeat
    p:= first(paths);
    success:= goal(first(p));
    if not success then begin
      children:= expand(first(p)));
      newpaths:= generate_new_paths(children,p);
      paths:= insert(newpaths,rest(paths));
    end;
  until success;
  return reverse(p);
end.
```

Praktikum / Prüfung

- Stufe 1:
  - A* in allen Aspekten gut verstehen
- Stufe 2:
  - Verbesserungen des A* ebenfalls berücksichtigen, also IDA*, SMA*,...
  - Der Bereich der informierten Suche, wie hier vorgestellt
- Stufe 3:
  - ABSOLVER
  - Heuristiken lernen
  - Etwas konzeptionell Komplexeres oder Umfassenderes als die Stufe-2-Themen
- Nicht geeignet, weil zu einfach:
  - Dijkstra-Alg.
  - Uninformierte Suche