SIEMENS

CORPORATE TECHNOLOGY

Vorlesung

Multiagentensysteme

Dr. Jörg Müller

Siemens AG, CT IC 6

joerg.p.mueller@siemens.com

SIEMENS

CORPORATE TECHNOLOGY

Vorlesung Multiagentensysteme

3. HEURISTISCHE VERFAHREN ZUR ENTSCHEIDUNGSFINDUNG

Übersicht

- Problemlösen als Suche
- Einführung in Heuristische Verfahren
- Suchverfahren
  - Allgemeines Modell
  - Bergsteiger-Strategie (Hill-climbing)
  - Monte-Carlo-Methode (Generate & Test)
  - A*-Algorithmus
  - Breitensuche
  - Tiefensuche
  - Backtracking
  - Statistische Methoden: Simulated Annealing

SIEMENS

CORPORATE TECHNOLOGY

Problemlösen als Suche

- Schon mehrfach haben wir in dieser Vorlesung die Modellierung des Problemlösungsprozess als Suche in einer Graphenstruktur aufgefasst.
- Beispiel STRIPS-Planung:
  - Ausgangssituation gegeben durch Menge von Prädikaten
  - Jede Situation (Zustand) kann als Knoten in einem Graphen modelliert werden
  - Übergänge zwischen Zuständen = Kanten = Ausführung von STRIPS-Operatoren
  - Suche = Finden eines Zielzustandes
- Diese Suchprobleme sind oft sehr schwierig (komplexe Zustände, viele Aktionen, viele Nachfolgezustände).
  - Beispiel: Mögliche Aktionen i einer bestimmten Situation eines Schachspiels

SIEMENS

CORPORATE TECHNOLOGY

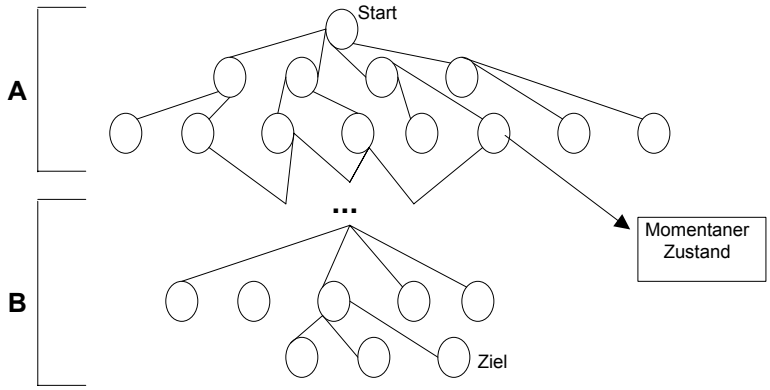
Entscheidungsfindung als Suche

- Die Aufgabe eines Agenten, in einer Situation seine nächste Aktion zu bestimmen (Aktionsauswahl), kann also als Teil eines Suchproblems modelliert werden.
- Hierbei handelt es sich in der Regel um schrittweise vorgehende Problemlösemechanismen. Hierbei hat man mehrere mögliche Schritte zur Auswahl. Dies birgt die Gefahr einer kombinatorischen Explosion.
- Von Interesse sind Methoden, die solche Auswahlen einschränken und in möglichst effizienter Weise nach der richtigen Lösung suchen.

SIEMENS

CORPORATE TECHNOLOGY

Bewertungen von Suchstrategien



A: Wie gut war der bisherige Teil der Lösung?

B: Wie kann man die Qualität des noch fehlenden Teils abschätzen?

SIEMENS

CORPORATE TECHNOLOGY

Beschreibungsebenen für Problemlöseprozesse

Zwei Beschreibungsebenen

- **Modellierung der betrachteten Domäne**
- **Modellierung des Vorgangs des Problemlösens**

- **Für die Beschreibung des Problemlösens bedienen wir uns unterschiedlicher Grundvorstellungen, die einander ergänzen.**

SIEMENS

CORPORATE TECHNOLOGY

Intuitive Modelle

- ***Bergsteigermodell*: Wir bewegen uns in einem Gebirge mit dem Ziel, den höchsten Gipfel zu erklimmen. Die erreichte Höhe gibt den Gütegrad der bisherigen Lösung / Teillösung an.**
- ***Graphensuchmodell* (s. STRIPS): noch zu lösende Teilprobleme sind an die Knoten eines Graphen angeheftet; ein Schritt entlang einer Kante bringt uns (hoffentlich) der Lösung näher.**
- **Die beiden Modelle ergänzen sich:**
  - **Das Bergsteigermodell modelliert die Art der Bewertung**
  - **Das Graphensuchmodell erlaubt eine Buchführung über das Fortschreiten überhaupt**

SIEMENS

CORPORATE TECHNOLOGY

Formales Modell

- Sei G ein gerichteter, beschrifteter Graph.
- Knoten von G beschriftet mit Teilproblemen (bzw. deren Kodierung).
- Kanten von G beschriftet mit Operationen, die zur Problemlösung zur Verfügung stehen. Dabei geht eine Kante *op* von A nach B gdw. Operation *op* reduziert Problem A auf Problem B.

Wir schreiben: $A \rightarrow_{\text{op}} B$

SIEMENS

CORPORATE TECHNOLOGY

Definitionen

- Ein Knoten heißt *erzeugt*, wenn seine Beschriftung von derjenigen seines Elternknotens berechnet ist.
- Ein Knoten heißt *expandiert*, wenn er selbst und alle seine Nachfolgeknoten erzeugt sind.
- Die OPEN-Liste ist die Liste aller Knoten, die erzeugt, aber noch nicht expandiert sind.
- Die CLOSED-Liste ist die Liste der expandierten Knoten.
- Der *Startknoten* ist ein ausgezeichneter Knoten, der mit dem Anfangsproblem beschriftet ist..
- *Terminalknoten* (es kann mehrere geben) sind mit dem leeren Problem beschriftet; sie haben keine Nachfolger.

SIEMENS

CORPORATE TECHNOLOGY

- Ein Pfad von Startknoten zu einem Terminalknoten beschreibt einen kompletten Lösungsweg.
- Unterschiedliche Terminalknoten können in bezug auf zusätzliche Bewertungen unterschiedlich gut sein.
- Erwünscht: Finden des *optimalen* Terminalknoten.
- Daraus ergeben sich alternative Wege zur Problemlösung. Dies führt zu einer Erweiterung unseres Graphenmodells.

SIEMENS

CORPORATE TECHNOLOGY

Und-Oder-Graphen

- Wir unterscheiden zwischen zwei Arten von Knoten:
  - ODER-Knoten: Verzweigung, die alternative Vorgehensweisen beschreibt
  - UND-Knoten: Verzweigung, die die Aufspaltung eines Problems in k Teilprobleme beschreibt, die alle gelöst werden müssen
- Einzelne Knoten entsprechen hier also Teillösungen. Terminalknoten sind primitive, nicht weiter unterteilbare Probleme.

SIEMENS

CORPORATE TECHNOLOGY

Lösungsgraphen

Die komplette Lösung wird als UND-ODER-Subgraph P von G repräsentiert, für den folgendes gilt:

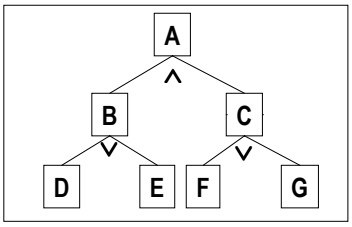
- P enthält den Startknoten.
- Alle Terminalknoten in P sind primitive Probleme, die als gelöst markiert sind.
- Für jeden UND-Knoten K in P, der einen direkten Nachfolger in P hat, gilt, dass P auch alle anderen direkten Nachfolger von K enthält.

SIEMENS

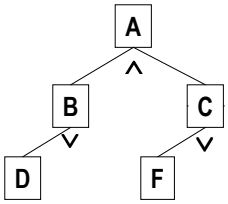
CORPORATE TECHNOLOGY

Beispiel für Lösungsgraphen

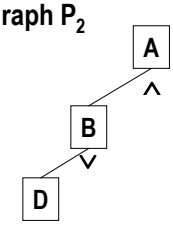
Graph G



Graph P_1



Graph P_2



P_2 ist kein Lösungsgraph!

SIEMENS

CORPORATE TECHNOLOGY

Die Rolle von Heuristiken bei der Suche

- Man kann Suchverfahren danach einteilen, welche Information über die zu behandelnde Problemklasse sie benutzen:
  - (a) Keine spezielle Information
  - (b) Information über den Einzelfall
  - (c) Statistische Informationen über die Problemklasse
- Das Frage, wie Information über den Einzelfall oder über die Problemklasse (Anwendungsdomäne) zur Verbesserung der Suche verwendet werden kann, führt uns zum Begriff der *Heuristik.*

SIEMENS

CORPORATE TECHNOLOGY

Einführung in heuristische Verfahren

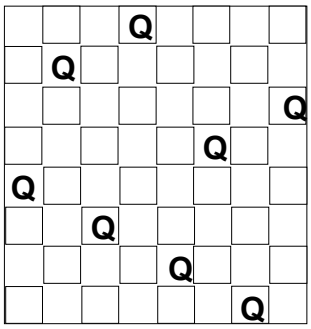
- *Heuristiken*: Kriterien, Methoden oder Prinzipien, die uns helfen, zu entscheiden, welche aus einer Menge möglicher Alternativen wahrscheinlich die vielversprechendste im Hinblick auf die Erreichung eines gegebenen Ziels ist.
- Heuristik = Faustregel.
- Heuristiken garantieren keinen Erfolg, sind aber in den meisten Fällen effektiv.
- Heuristiken werden besonders effektiv in großen Suchproblemen eingesetzt, bei denen nicht alle möglichen Alternativen betrachtet werden können. Hier helfen Heuristiken, den zu betrachtenden Teil des Suchraums zu reduzieren und in kurzer Zeit eine annehmbare Lösung zu finden.

SIEMENS

CORPORATE TECHNOLOGY

Beispiel: N-Damen-Problem

- Gegeben: Schachbrett mit Seitenlänge N (z.B. N=8).
- Ziel: N Damen auf dem Schachbrett so plazieren, dass keine Dame eine andere bedroht.



SIEMENS

CORPORATE TECHNOLOGY

N-Damenproblem

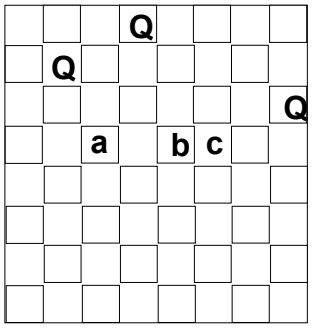
- Andere Formulierung: Keine Zeile, Spalte oder Diagonale darf mehr als eine Königin enthalten.
- Für alle N > 3 gilt: Das N-Damenproblem ist lösbar.
- Inkrementelle Lösung: N Damen werden nacheinander (z.B. Zeile um Zeile) auf dem Schachbrett plaziert.
- Dies erlaubt uns, bei jeder neuen Dame die Zahl der zu betrachtenden Positionen einzuschränken.
- Dies ist möglich, weil das N-Damenproblem „gutmütig“ ist: es ist nie möglich, einmal aufgetretene Verletzungen der Rand-bedingungen durch zukünftige Entscheidungen wieder gutzumachen. Dies erlaubt uns, unnütze Konfigurationen früh zu verwerfen.

Heuristiken im N-Damenproblem

- **Wenn wir, wie beschrieben, nacheinander pro Reihe eine Dame setzen, scheint es intuitiv, bei der Auswahl der Position für die nächste Dame möglichst viele unbedrohte Felder zu lassen.**
- ***Heuristik 1*: Bevorzuge eine Position, die die höchstmögliche Anzahl nicht bedrohter Felder im verbleibenden Teil des Schachbretts lässt.**
- ***Heuristik 2*: Bei der Auswahl der als nächstes zu besetzenden Zeile gilt, dass Zeilen, die nur noch wenige unbedrohte Felder enthalten, wahrscheinlich schneller blockiert werden. D.h., wir sollten versuchen, die Zeilen zuerst zu besetzen, die die kleinste Anzahl unbedrohter Felder hat.**

Realisierung Heuristik 1

- **Wir beschreiben die Attraktivität einer Alternative x durch eine Funktion f1(x), die für x die Zahl der unbedrohten Felder liefert.**
  - **f1(a) = 8**
  - **f1(b) = 9**
  - **f1(c) = 10**
- **D.h., Heuristik 1 würde für Option c sprechen**
- **In der Tat führen Varianten b und c zu Lösungen, a hingegen nicht.**

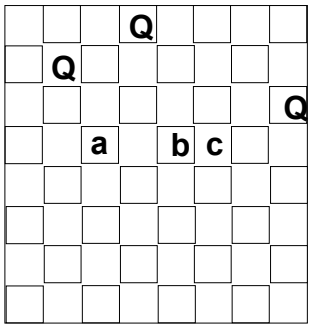


SIEMENS

CORPORATE TECHNOLOGY

Realisierung Heuristik 2

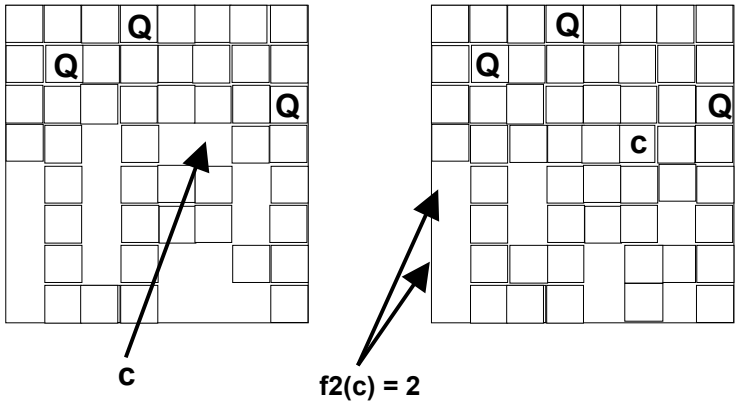
- Wir beschreiben die Attraktivität einer Reihe r durch eine Funktion f2(r), die für r das Minimum der unbedrohten Felder pro Reihe über alle noch zu besetzenden Reihen liefert.
  - f2(a) = 1
  - f2(b) = 1
  - f2(c) = 2
- Auch hier schneidet Option c am besten ab.



SIEMENS

CORPORATE TECHNOLOGY

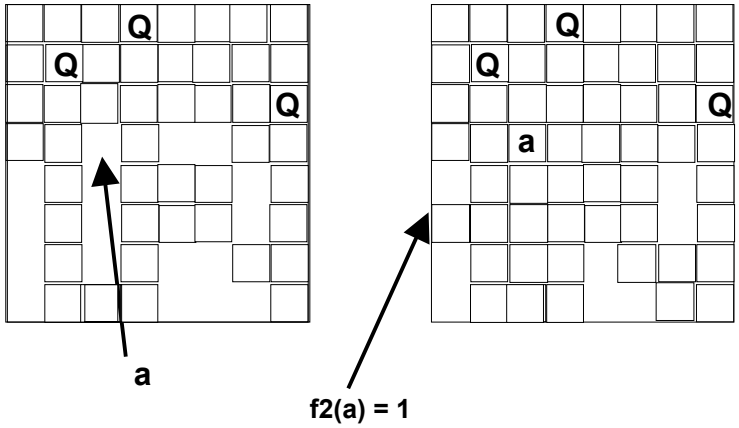
Illustration der Funktion f2 für Option c



SIEMENS

CORPORATE TECHNOLOGY

Illustration der Funktion f2 für Option a



SIEMENS

CORPORATE TECHNOLOGY

Analyse der Heuristiken für das N-Damenproblem

- Wenn für eine Alternative die Funktionen f1 oder f2 den Wert 0 ergeben, braucht diese Alternative nicht weiterverfolgt zu werden (Abbruchkriterium).
- f2 bietet f1 gegenüber den Vorteil, dass sie Abbruchkriterien früher erkennt und somit eine effektivere Suche erlaubt.
- Eine wesentliche Frage ist, welche Charakteristika von Heuristiken es gibt, die sie für eine Klasse von Problemen geeignet erscheinen lassen. Wir werden diese Frage gleich untersuchen.
- Zunächst untersuchen wir jedoch einige Suchstrategien anhand des N-Damenproblems.

SIEMENS

CORPORATE TECHNOLOGY

Generate & Test-Strategie

- Annahme: Wir können vom Startknoten direkt alle anderen Knoten erzeugen.
- Algorithmus:

```
Lösung_gefunden = false;
Solange nicht Lösung_gefunden
  Erzeuge nächsten Knoten K // Generate
  Wenn
      K ist Terminalknoten    // Test
  Dann
      Setze Lösung_gefunden = true
      Gib K zurück
```

- Wird zum Erzeugen eine zufällige Funktion verwendet, nennt man diesen Algorithmus auch Monte-Carlo-Methode.

SIEMENS

CORPORATE TECHNOLOGY

Eigenschaften des Generate&Test-Verfahren

- Worst-Case: Blinde Suche (Monte-Carlo), kann sehr ineffizient sein.
- Die Generate-Funktion muss unter Umständen zu viele Knoten erzeugen.
- Die Test-Funktion kann unter Umständen sehr aufwendig sein.
- Meistens gibt es effektivere Verfahren als Generate&Test.

Backtracking-Verfahren

- Generate&Test generiert stets komplette Lösungen.
- Idee: Beim iterativen Entwickeln einer Lösung können wir Information (z.B. Fehlschläge) verwenden, um den Suchraum zu verkleinern.
- Backtracking bricht im Fall einer ungültigen k-Teillösung ab, der entsprechende Teilbaum des Problemgraphen wird nicht weiter durchsucht, es wird stattdessen zur letzten gültigen $(k-1)$-Teillösung zurückgesprungen.
- N-Damenproblem: Eine gültige Teillösung der Länge $k<=N$ entspricht einem Tupel $(q_1,...,q_k)$, wobei q_j jeweils die Koordinate der Dame in Reihe j ist.

Forward-Checking-Verfahren

- Das Forward-Checking-Verfahren versucht, aktiv bei der Erstellung einer Teillösung Konfigurationen zu beschneiden, die nicht konsistent sind (d.h., keine Lösung mehr sein können).
- Unser intuitives Vorgehen beim Erstellen von Teillösungen entspricht diesem Verfahren: Wann immer eine Dame auf dem Feld postiert wird, werden alle Felder, die in der gleichen Reihe, der gleichen Spalte oder auf der Diagonale liegen, aus den Mengen der möglichen Werte für die noch zu postierenden Damen gelöscht.
- Die hierbei angenommene Sicht ist die des Finite Domain Constraint Solving:
  - Variablen mit endlichen Wertebereichen
  - Constraints zwischen Variablen als Randbedingungen

SIEMENS

CORPORATE TECHNOLOGY

Bewertung heuristischer Suchverfahren

- In der Folge wollen wir auf der Basis des zuvor beschriebenen Suchmodells ein Modell für heuristische Suchverfahren entwickeln und einige Eigenschaften solcher Verfahren herausarbeiten.
- Hierzu führen wir eine Reihe von Bezeichnungen ein:
- K = Knotenmenge
- T = Menge der Terminalknoten
- s = Startknoten
- p(n) = Pfad von Knoten n zu einem Knoten aus T
- succ(n) = Menge der Nachfolgeknoten von n

SIEMENS

CORPORATE TECHNOLOGY

Bewertungsfunktionen

- g*(n): Kosten des billigsten Pfades von s zu Knoten n
- $g_p(n)$: Kosten des Pfades p von s zu n
- g(n): Kosten des billigsten bisher bekannten Pfades von s zu n
- k(n,m): Kosten des billigsten Pfades von n nach m (undefiniert, wenn kein Pfad von n nach m existiert)
- h*(n): Kosten des billigsten Pfades von n zu einem Knoten in T
- K* (= h*(s)): Kosten des billigsten Pfades überhaupt
- h(n) = Schätzfunktion für h*(n)

SIEMENS

CORPORATE TECHNOLOGY

Schätzfunktionen

- Wir können nun die Gesamtkosten eines Pfades von s durch einen Knoten n zu einem Knoten in T wie folgt abschätzen:
- f(n) = g(n) + h(n): Schätzfunktion für billigsten Pfad von s durch n zu einem Knoten in T.
- f*(n) = g*(n) + h*(n): Kosten des tatsächlich billigsten (optimalen) Pfad von s durch n zu einem Knoten in T.

Es gilt:

- $g(n) \geq g^*(n)$
- $f^*(s) = K^*$

SIEMENS

CORPORATE TECHNOLOGY

Eigenschaften heuristischer Schätzfunktionen

- Funktion g ist kein Schätzwert, sondern ein bekannter Erfahrungswert.
- Funktion h beinhaltet eine Schätzung; in der Güte von h liegt der Wert der Problemlösung.
- Ziel: minimiere $|h^*(n) - h(n)|$ über alle n.

Zwei Schätzstrategien:

- Pessimistisch: $h^*(n) \leq h(n)$ für jeden Knoten n.
- Optimistisch: $h^*(n) \geq h(n)$ für jeden Knoten n.
- Die pessimistische Strategie führt zu einer Kostenüberschätzung, die optimistische Strategie zu einer Kostenunterschätzung.

4.4 A*-Algorithmen für heuristische Suche

- Eine grundlegende Klasse von Algorithmen wird unter dem Namen A*-Algorithmen zusammengefasst.
- Oft redet man jedoch vereinfachend vom A*-Algorithmus.
- Der A*-Algorithmus hat zwei Parameter:
  - Die (anwendungsspezifische) Kostenfunktion g
  - Die Schätzfunktion h
- Der A*-Algorithmus hat zwei wesentliche Funktionen:
  - In jeder Situation innerhalb der Suche dient er der Auswahl des nächsten zu untersuchenden Knotens
  - Für jeden untersuchten Knoten n führt er Buch über den Weg, der zu diesem Knoten geführt hat (Liste WEG(n)).

Der A*-Algorithmus

(1) Setze g(s)=0, CLOSED=[], OPEN=[s], WEG(s)=[s];
berechne h(s) sowie f(s) = g(s) + h(s).

(2) Wenn OPEN=[], dann STOP: keine Lösung gefunden.

(3) Wähle n aus OPEN mit f(n) minimal;
setze OPEN=OPEN – [n], CLOSED=[n] + CLOSED

(4) Wenn n ∈ T, dann STOP: Lösung n gefunden mit Kosten g(n)

(5) Wenn n ∉ T, dann betrachte alle n‘∈ Succ(n):

a. Wenn n‘ ∉ OPEN und n‘ ∉ CLOSED, dann setze WEG(n‘) = [n‘] + Weg(n), OPEN = [n‘]+OPEN, g(n‘) = k(n, n‘)+g(n) und berechne f(n‘)=g(n‘)+h(n‘)

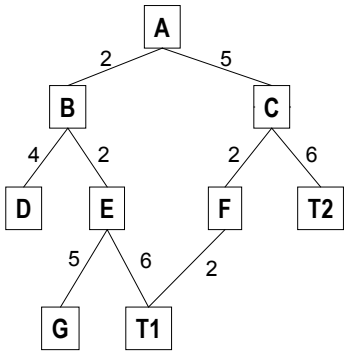
b. Wenn n‘∈ OPEN oder n‘∈ CLOSED, und g(n)+k(n,n‘)<g(n‘), dann setze WEG(n‘)=[n‘]+WEG(n) und g(n‘)=g(n)+k(n,n‘). Falls speziell n‘ in CLOSED, dann setze OPEN=[n‘]+OPEN und CLOSED=CLOSED – [n‘]

(6) Gehe zu (2)

SIEMENS

CORPORATE TECHNOLOGY

A*-Algorithmus: Beispiel



Seien h, h´ Schätzfunktion mit:

h(A)=8; h(B)=7; h(C)=3; h(D)=∞;
h(E)=5; h(F)=1; h(G)=∞;
h(T1)=h(T2)=0;

h‘(F) = 4; ansonsten h‘ wie h

Wenden wir h an, so findet der Algorithmus für das Beispiel die optimale Lösung; wenden wir h´ an, nicht.

Warum?

SIEMENS

CORPORATE TECHNOLOGY

Anmerkungen zum Algorithmus

- **Ein Knoten kann auf unterschiedliche Weise erreicht werden. Deshalb kann ein in Schritt (5) erreichter Nachfolgeknoten bereits auf der OPEN oder CLOSED-Liste stehen (Fall (5b)). In diesem Fall ist zu prüfen, ob der neu gefundene Weg kürzer ist als der beste bisher bekannte.**
- **Wurde in Schritt (5b) ein neuer kürzester Weg für einen Knoten gefunden, der bereits expandiert wurde (d.h., Nachfolgeknoten hat), so muss die Funktion f für den Knoten und alle Nachfolgeknoten neu berechnet werden. Dazu wird er aus der CLOSED-Liste gelöscht und in die OPEN-Liste abgelegt.**
- **Schritt (5a) deckt den Fall ab, dass ein neuer Knoten gefunden wurde. Dieser wird auf die OPEN-Liste abgelegt.**

Bekannte Ausprägungen des A*-Algorithmus

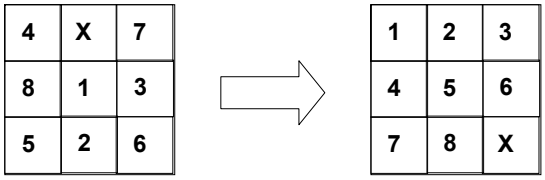
- Es stellt sich heraus, dass viele bekannte Suchverfahren als Variation des A*-Algorithmus darstellbar sind.
- Breitensuche in einem Baum:
  - Wähle k(n, n‘) = 1 für alle n und n‘ mit n‘$\in$ Succ(n)
  - Wähle h(n) = 0 für alle n
  - Es gilt: Kosten g(n) = Tiefe des Knoten im Baum
  - Weiterhin gilt: Die Knoten gleicher Tiefe werden untersucht, bevor ihre Nachfolger expandiert werden
- Tiefensuche in einem Baum
  - Erfordert leichte Modifikation des A*-Algorithmus
  - Für n‘ in Succ(n) berechne f(n‘) = f(n) – 1. Setze f(s)=0.

Backtracking und A*-Algorithmen

- Beschreibung einer Backtracking-Strategie als Tiefensuche erfordert weitere Modifikation der Tiefensuche. Wir skizzieren hier das Vorgehen nur informell.
  - Wir beschränken die maximale Suchtiefe durch eine Schranke S
  - Die Funktion f misst wieder die Tiefe des Baumes.
  - Wähle den ersten Knoten n in OPEN; wenn seine Tiefe den Wert S überschreitet oder alle von ihm ausgehenden Zweige bearbeitet sind, dann setze OPEN=OPEN-[n]; sonst wähle neuen Knoten n‘ in Succ(n) und setze WEG(n‘)=[n‘]+WEG(n).
  - Wenn n‘ in T: STOP mit Lösung n‘. Sonst:
    - wenn Succ(n‘)= $\emptyset$, entferne n‘ von OPEN, rufe Prozedur rekursiv auf
    - Wenn Succ(n‘) $\neq \emptyset$, wähle einen Nachfolger aus, füge ihn zu OPEN hinzu und rufe die Prozedur rekursiv auf.

Beispiel A*-Algorithmus: Das N-Puzzle

- nxn-Schachbrett, n^2-1 Steine durchnumeriert von 1 bis n^2-1, welche auf die Felder verteilt werden. Das freigebliebene Feld wird durch „x“ gekennzeichnet.
- Vorgegeben ist eine beliebige Verteilung der Steine; die Steine sind zu sortieren wie im Bild gezeigt für n=3.
- Erlaubt ist ein Verschieben eines Steines nach rechts, links, oben, unten auf ein freies Feld



- Funktion g(n) zählt die Anzahl der Züge zum Knoten n aus der Ausgangskonfiguration.
- Für h*(n) wollen wir zwei alternative Schätzungen untersuchen:
  - $h_1(n)$ = Anzahl der Steine auf falschen Feldern
  - $h_2(n)$ = Summe der horizontalen und vertikalen Abstände der einzelnen Steine von ihrer Zielposition
- Es gilt:
  - Beide Funktionen sind optimistische Schätzungen von h*
  - Für jeden Knoten n gilt $h_1(n) \leq h_2(n) \leq h^*(n)$
- Wir bemerken: die im N-Puzzle verwendeten Schätzfunktionen verwenden im Gegensatz z.B. zur Breiten- oder Tiefensuche Wissen über die Problemstellung. Wir könnten sagen, dass sie *besser informiert* sind.

Eigenschaften des A*-Algorithmus

- Wenn überhaupt eine Lösung existiert (d.h., $T \neq \emptyset$), dann terminiert der A*-Algorithmus mit einem $t \in T$.
- Wird eine optimistische Schätzfunktion h verwendet, liefert der A*-Algorithmus in diesem Fall eine optimale Lösung.
- Für optimistische Schätzfunktionen gilt weiterhin:
  - Wenn ein Knoten n expandiert wird, dann gilt $f(n) \leq K^*$
  - Jeder Knoten n in OPEN mit $f(n) < K^*$ wird auch tatsächlich expandiert
  - A* schaltet Knoten n mit $f(n) > K^*$ endgültig aus
- Sicherer Weg, eine optimistische Schätzfunktion zu wählen: $h(n)=0$ für alle n.
- Dies bedeutet allerdings eine geringe Sucheffizienz, denn h beinhaltet keinerlei Information über das Problem.

Informiertheit von Schätzfunktionen

- Für zwei optimistische Schätzfunktionen h_1 und h_2 heisst h_2 besser informiert als h_1, falls $h_1(n) < h_2(n)$ für jeden Knoten aus $n \in T$.
- h_2 heißt nicht schlechter informiert als h_1, wenn $h_1(n) \leq h_2(n)$ für alle $n \in T$.
- Im obigen Beispiel für das N-Puzzle war h_2 nicht schlechter informiert als h_1.

SIEMENS

CORPORATE TECHNOLOGY

Informiertheit von A*-Algorithmen

- Auf der Basis der Eigenschaften von Schätzfunktionen halten wir folgendes Resultat fest:
- Seien $A_1^* = A_1^*(g, h_1)$ und $A_2^* = A_2^*(g, h_2)$ zwei A*-Algorithmen mit optimistischen Schätzfunktionen h_1 und h_2, wobei h_2 besser informiert ist als h_1.
  Dann gilt: Jeder Knoten, der von A_2^* expandiert wird, wird auch von A_1^* expandiert. A_2^* arbeitet in diesem Sinne also nicht schlechter als A_1^*.
- Nachteil dieser Beobachtung: Wir wollen nicht die Zahl der expandierten Knoten minimieren, sondern die Gesamtanzahl der Expansionen. Deshalb führen wir den Begriff der Monotonie ein.

SIEMENS

CORPORATE TECHNOLOGY

Monotone Schätzfunktionen

- Eine Schätzfunktion h heißt monoton, falls für alle n und n‘∈ Succ(n) gilt:

$$h(n) \leq k(n, n') + h(n')$$

- Wir fordern also eine lokale Konsistenz zwischen den Werten von h und den Kantenkosten.
- Analogie; Repräsentiert man die Werte von h und k durch Vektoren, so entspricht dies der Dreiecksungleichung aus der linearen Algebra.

SIEMENS

CORPORATE TECHNOLOGY

Eigenschaften monotoner Schätzfunktionen

- Monotone Schätzfunktionen sind stets optimistisch.
- Für A* = A*(g, h) mit monotonem h gilt:
  - g(n) = g*(n) für alle n ∈ CLOSED
  - Wenn n nach m expandiert wurde, dann gilt f(n) ≥ f(m) (d.h., die Folge der f-Werte expandierter Knoten ist nicht abnehmend)
  - Wenn n expandiert wurde, so gilt g*(n) + h(n) ≤ K*
  - Jeder Knoten mit g*(n) + h(n) < K* wird tatsächlich expandiert.
- Folgerungen daraus:
  - Wenn A* einen Knoten zur Expansion auswählt, dann hat er bereits einen optimalen Weg zu dem Knoten gefunden.
  - Der Test in A*, ob ein expandierter Knoten bereits in CLOSED ist, kann entfallen, d.h., man wird keine Abkürzungen finden.

SIEMENS

CORPORATE TECHNOLOGY

Bergsteigermodell der Problemlösung

- Zum Abschluss dieses Kapitels untersuchen wir das zu Beginn angedeutete zweite Modell der Problemlösung: Das Bergsteigermodell
- Eine bekannte Problemlösungsstrategie ist die Methode des steilsten Anstieges (auch: Gradientenmethode). Hierbei wird immer der Nachfolgeknoten n‘ eines Knotens n gewählt, der gegenüber n den größten Wert der Bewertungsfunktion liefert.
- D.h., man geht bei der Problemlösung immer in die Richtung des steilsten Anstiegs innerhalb des Lösungsraums.
- Äquivalente Betrachtung: Suche nach dem tiefsten Tal (Minimum).

Probleme mit der Gradientenmethode

- Die Terminierung der Methode ist nicht garantiert, z.B.:
  - auf einem Plateau
  - am Rande eines Kraters
  - bei fester Schrittlänge ist es möglich, dass man stets über den „Gipfel“ hinwegspringt
- Es besteht die Gefahr, sich auf einem lokalen Optimum zu verirren.
- Die Terminierung der Methode im Falle eines Plateau oder Kraters kann durch eine Version der Gradientenmethode mit Buchführung erreicht werden, die sich bereits expandierte Knoten merkt und verhindert, dass ein Knoten mehrfach expandiert wird. In diesem Fall entspricht die Methode einer Tiefensuche ohne Backtracking-Möglichkeit.

SIEMENS

CORPORATE TECHNOLOGY

Optimierungen der Gradientenmethode

- Verkleinere die Schrittweiten bei größeren Werten der Bewertungsfunktion f (beziehungsweise des Gradienten).
- Vergrößere gelegentlich (statistisch) die Schrittweite oder erlaube Schritte, die bergab führen.
- „Iterierte Monte-Carlo-Methode“:
  - Bestimme den Ausgangspunkt der Suche zufällig
  - Verwende die Gradientenmethode, um ein lokales Optimum zu finden
  - Wiederhole dies N mal und speichere das jeweils beste Ergebnis
  - Gib das beste Ergebnis nach N Durchläufen zurück
- Eine weitere statistische Methode ist „Simulated Annealing“.

SIEMENS

CORPORATE TECHNOLOGY

Simulated Annealing

- Die Simulated Annealing-Methode wurde entwickelt, um auch bei Nachbarschaftstopologien mit lokalen Optima (Minima) die Wahrscheinlichkeit zu reduzieren, dass das globale Optimum verfehlt wird.
- Analogie: Abkühlen flüssiger Stoffe zu kristallinen Strukturen aus der Thermodynamik
- Ziel: Minimiere Kostenfunktion E: $X \rightarrow \Re$, wobei X die Menge der möglichen Lösungspunkte ist.
- Zu jedem x in X seien die Nachbarn $N_x \subseteq X$ von x die Menge der Punkte von X, die im ursprünglichen Problem in einem Schritt von x aus erreichbar sind.

SIEMENS

CORPORATE TECHNOLOGY

Unterschiede zwischen Simulated Annealing und Bergsteigerverfahren

- Die Nachbarn der momentan gefundenen Lösung werden nicht systematisch auf ihren Zielfunktionswert untersucht. Vielmehr wird aus allen Nachbarn mittels einer gleichverteilten Wahrscheinlichkeit ein Nachbar ausgesucht. Dabei ist p(x,y) die Wahrscheinlichkeit, dass y als Nachfolgepunkt von x vorgeschlagen wird. Wir setzen voraus:
  - $p(x, y) > 0 \Leftrightarrow Y \in N_x$
  - p symmetrisch
- Eine Lösung x wird auch zugunsten einer schlechteren Nachbarlösung y (d.h. E(y) > E(x)) verworfen, allerdings nur mit einer Akzeptanzwahrscheinlichkeit $q_c(x,y)$

SIEMENS

CORPORATE TECHNOLOGY

Akzeptanzwahrscheinlichkeit

- **Die Funktion q hängt neben dem Ausmaß der Zielfunktionsverschlechterung von einer exogenen Variable c ab, die gemäß der physikalischen Analogie als Temperatur bezeichnet wird.**
- **Ist c sehr groß, gleicht die Suche einem „Random Walk“ durch den Lösungsraum. Wird c auf einen Wert sehr nahe 0 gesetzt, verhält sich der Algorithmus wie die Gradientenmethode (d.h., Verschlechterungen der Lösung werden praktisch nicht mehr akzeptiert).**

$$q_c(x,y) = \begin{cases} 1, falls\ E(x) \geq E(y) \\ e^{\frac{E(x)-E(y)}{c}}, sonst \end{cases}$$

SIEMENS

CORPORATE TECHNOLOGY

Übergangswahrscheinlichkeit

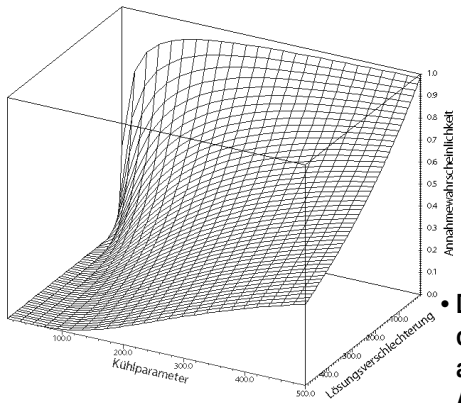
- **Die Überlegung beim Simulated Annealing besteht nun darin, die Suche mit einem hohen Wert von c zu beginnen und danach c im Laufe der Suchschritte, die insgesamt ausgeführt werden, langsam abzusenken („Abkühlung“)**
- **Simulated Annealing-Algorithmen können grundsätzlich als Markov-Ketten modelliert werden, wobei für die *Übergangswahrscheinlichkeit* von Zustand x nach Zustand y gilt:**

$$t(x,y) = \begin{cases} q_c(x,y) \cdot p(x,y)\ für\ x \neq y \\ 1 - \sum_{z \neq x} q_c(x,z) \cdot p(x,z)\ für\ x = y \end{cases}$$

SIEMENS

CORPORATE TECHNOLOGY

Illustration des Abkühlprozesses



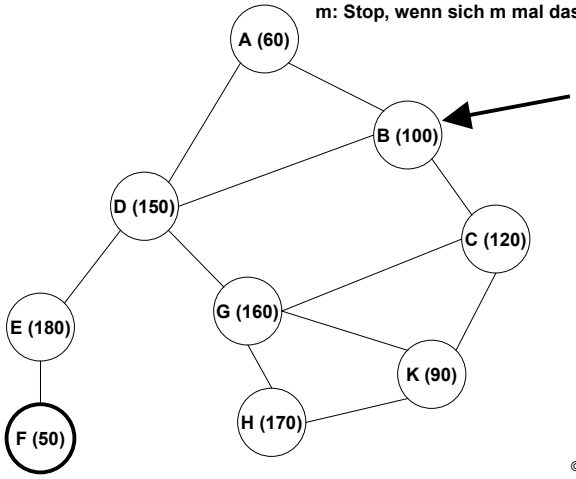
- **Der Kühlparameter (Temperatur) c wird im Verlauf des Verfahrens abgekühlt. Damit sinkt die Akzeptanzwahrscheinlichkeit für eine gegebene Lösungsverschlechterung**

SIEMENS

CORPORATE TECHNOLOGY

Beispiel

Verlauf für c, z.B.: (100,50,25,10,5,2,1,0.5,0.25,...)
Zwei weitere Kontrollparameter:
k: (alle k Schritte wird c verringert)
m: Stop, wenn sich m mal das gleiche Element behauptet



$q_{100}(B,D)=0.61$

$q_{100}(B,C)=0.82$

$q_{10}(B,D)=0,006$

$q_{10}(B,C)=0.14$

Annealing-Algorithmus

(1) Wähle $x \in X$ beliebig.

(2) Setze $i=1$; $x_i=x$; $j=1$.

(3) Setze $r=1$; $s=1$; $c=c_j$; $q=q_c$; // s ist Indexvar. für m,
// r ist Indexvar. Für k

(4) Setze $u=x_i$; wähle $y \in N_u$ mit Wkt. $p(u,y)$;
Akzeptiere z=y mit Wkt. $q_c(u,y)$ oder
z=u mit Wkt. $1 - q_c(u,y)$;
Setze s=1 falls z=y, oder s=s+1 falls z=u;
Setze i=i+1 und x_i=z;
Setze r=r+1;

(5) Wenn s=m, dann STOP mit Ausgabe x_i;
Wenn s<m: wenn r<k gehe zu (4);
wenn r=k, setze j=j+1 und gehe zu (3).

Praktische Erwägungen

- Die Konvergenz einer Suche mit Simulated Annealing gegen die globalen Optima in X kann sichergestellt werden, sofern die Abkühlung der Temperatur c hinreichend langsam vonstatten geht.
- In empirischen Tests zeigt sich aber, dass eine Abkühlung, für die Konvergenz garantiert werden kann, meist zu unvertretbar langen Rechenzeiten führt.
- Allerdings lassen sich auch mit zu schneller Abkühlung fast immer bessere Ergebnisse erzielen als mit der Verwendung der klassischen Bergsteigerverfahren.

Lerninhalte dieses Kapitels

- Verstehen des Konzeptes der heuristischen Suche als wichtige Basismethode zur Entscheidungsfindung autonomer Agenten
- Vorstellung unterschiedlicher graphenbasierte Suchalgorithmen, insbesondere A*-Algorithmus und seine Eigenschaften
- Auf dem „Bergsteigermodell“ der Suche beruhende Algorithmen: Methode des steilsten Anstiegs und Simulated Annealing