Künstliche Intelligenz

Ilm13

WS13/14

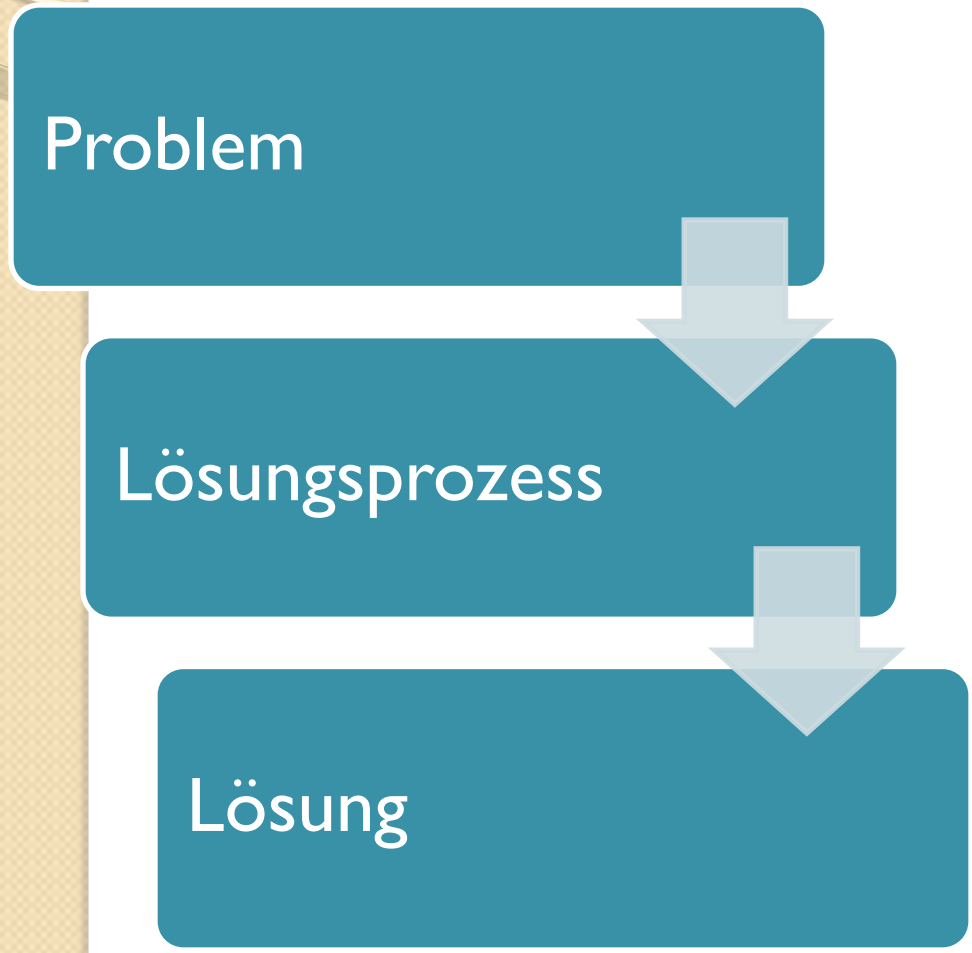
Georg Ringwelski

Wiederholung

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Künstliche Neuronale Netze
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche
4. Wissensverarbeitung
   1. Aussagenlogik
   2. Prädikatenlogik
   3. Logisches Schließen

3. Problemlösen



Spezifikation eines Problemlösers

1. Repräsentation des Problems und der Lösung
2. Identifikation elementarer Lösungsschritte
3. Methode zur systematischen Durchführung der Lösungsschritte

3. Problemlösen

Das Repräsentationsproblem

- Zur maschinellen Verarbeitung notwendig: formale Spezifikation des Problems:

a) Ausgangssituation
b) Änderungsmöglichkeiten
c) Zielsituation

3. Problemlösen

z.B. Planung mit STRIPS

- Ausgangszustand: Eine Menge (Konjugation) von Ausdrücken (zB PL/1)
- Operation: Name + Ein Quadrupel von Mengen von Zuständen
  1. die wahr sein müssen
  2. die falsch sein müssen

  } Vorbedingungen

  3. die wahr werden
  4. die falsch werden

  } Effekte der Operation
- Zielzustand: Eine Menge von Ausdrücken

3. Problemlösen

STRIPS

- Eine Planung in STRIPS ist gegeben durch eine Folge von Operatoren mit
  - Vorbedingungen jedes Operators sind durch vorherige Operatoren bzw Startzustand erfüllt
- Daraus ergibt sich eine Folge von Situationen
  - Beginnend mit Startzustand
  - Endend mit Zielzustand

Beispiel: Blocksworld: aispace.org/planning

3 Problemlösen

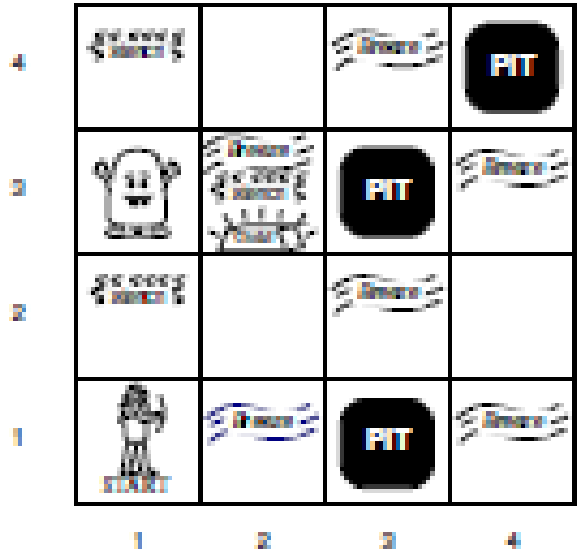
STRIPS, Beispiel

Wumpus-Welt

- Was sind die Zustände?

3 Problemlösen

Example Domain: Wumpus World



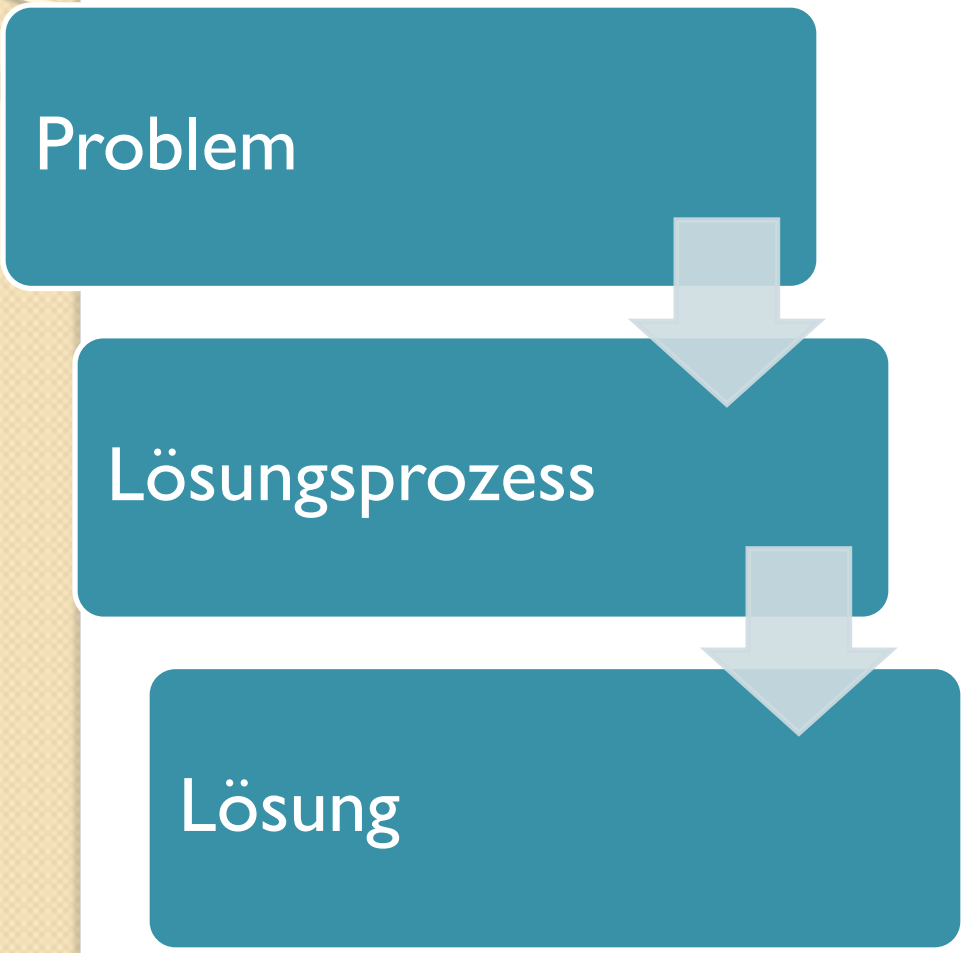
- Want to get to the gold and grab it.
- Want to avoid pits and the "wumpus".
- Clues: breeze near pits and stench near the wumpus.
- Other sensors: wall (bump), gold (glitter), kill (scream)
- Actions: move, grab, or shoot.

3. Problemlösen

Sprachen zur Beschreibung von Zuständen

- Modelle aller Art von PL/1 bis Java-Klassen
- Eigenschaften
  - Gültigen (feasible) Lösungen eindeutig identifizierbar
  - Was nicht gültig ist, ist ungültig (CWA)
  - Abgrenzung (hard/crisp) Constraints und soft Constraints bzw. Ziele

3. Problemlösen



Spezifikation eines Problemlösers

1. Repräsentation des Problems und der Lösung
2. Identifikation elementarer Lösungsschritte
3. Methode zur systematischen Durchführung der Lösungsschritte

3 Problemlösen

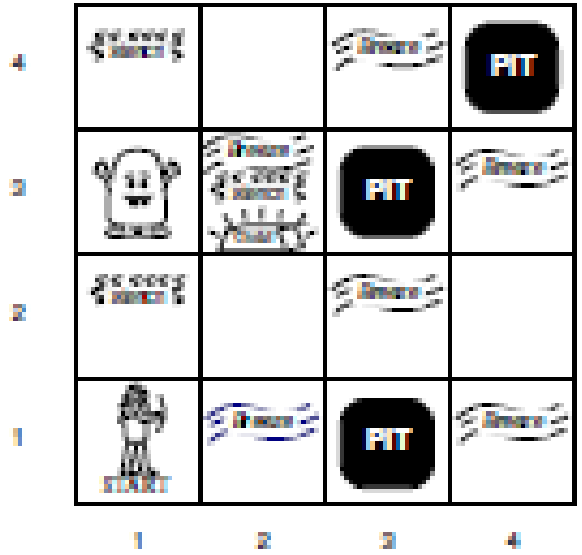
STRIPS, Beispiel

Und schon wieder: Wumpus-Welt

- Was sind die Operationen?
- Wie verändern sie den Zustand?

3 Problemlösen

Example Domain: Wumpus World



- Want to get to the gold and grab it.
- Want to avoid pits and the "wumpus".
- Clues: breeze near pits and stench near the wumpus.
- Other sensors: wall (bump), gold (glitter), kill (scream)
- Actions: move, grab, or shoot.

3. Problemlösen

Sprachen zur Beschreibung von Zustandsübergängen

- Transformationen der Modelle
- zB Zustand von Java-Objekten verädern, Menge geltender Formeln erweitern,…
- Eigenschaften
  - Überführt Zustände in Zustände
  - Meist (bei non-greedy-Problemen) sind mehrere Aktionen in jedem Zustand möglich
  - Manchmal: nur gültige Zustände
  - Manchmal: konfluent

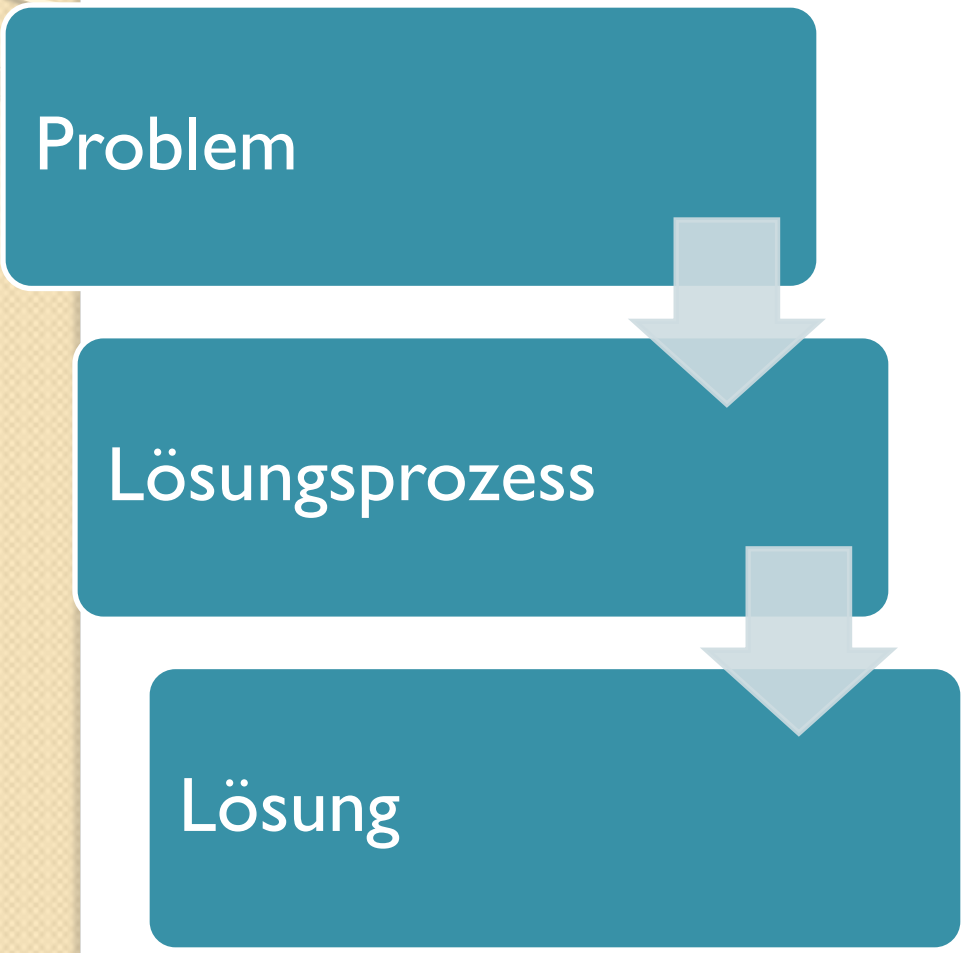
3. Problemlösen

Übung

Modellieren Sie das n-damen-Problem in Java

- Definieren Sie den Startzustand
- Implementieren Sie eine Methode zur Identifikation von Zielzuständen
- Definieren Sie eine Operation zur Transformation von Zuständen

3. Problemlösen



Spezifikation eines Problemlösers

1. Repräsentation des Problems und der Lösung
2. Identifikation elementarer Lösungsschritte
3. Methode zur systematischen Durchführung der Lösungsschritte

3. Problemlösen

Anwendung von Zustandsübergängen

- Erweiterung des Wissens oft durch Modifikation der Problemrepräsentation

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Künstliche Neuronale Netze
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche
4. Wissensverarbeitung
   1. Aussagenlogik
   2. Prädikatenlogik
   3. Logisches Schließen

3.1 Vollständige Suche

Zustandsübergänge bei vollständiger Suche

- iterative Erzeugung des Suchraums
- i.d.R. exponentiell
- Zur Visualisierung des Suchraums benutzt man oft Suchbäume (nicht zur Implementierung)

3. I Vollständige Suche

Beispiele:

- Wie sieht der Suchraum bei n-queens aus?
- Wie sieht der Suchraum bei der Suche nach Lösungen vom „Haus des Nikolaus“ aus
- Wie sieht der Suchraum bei der Suche nach magischen Quadraten aus?

3.1 Vollständige Suche

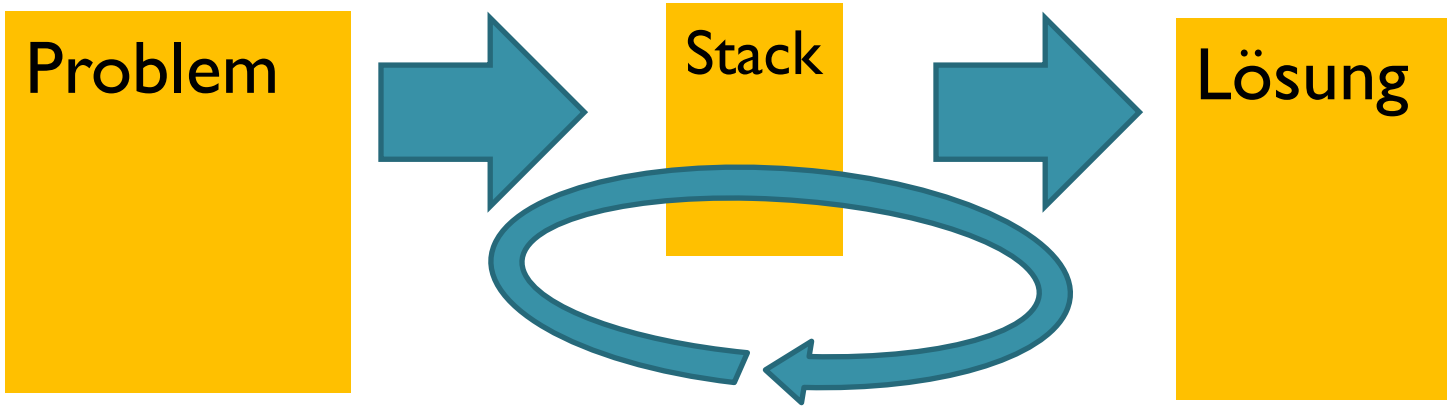
Traversierung des Suchraums

- Systematik bei der Durchführung der Lösungsschritte
  - Tiefe-zuerst und Breite-zuerst
  - Heuristiken und Varianten
- Implementiert durch unterschiedliche Verwaltung des „Wissens“

3.1 Vollständige Suche

Tiefe-zuerst rekursiv oder iterativ

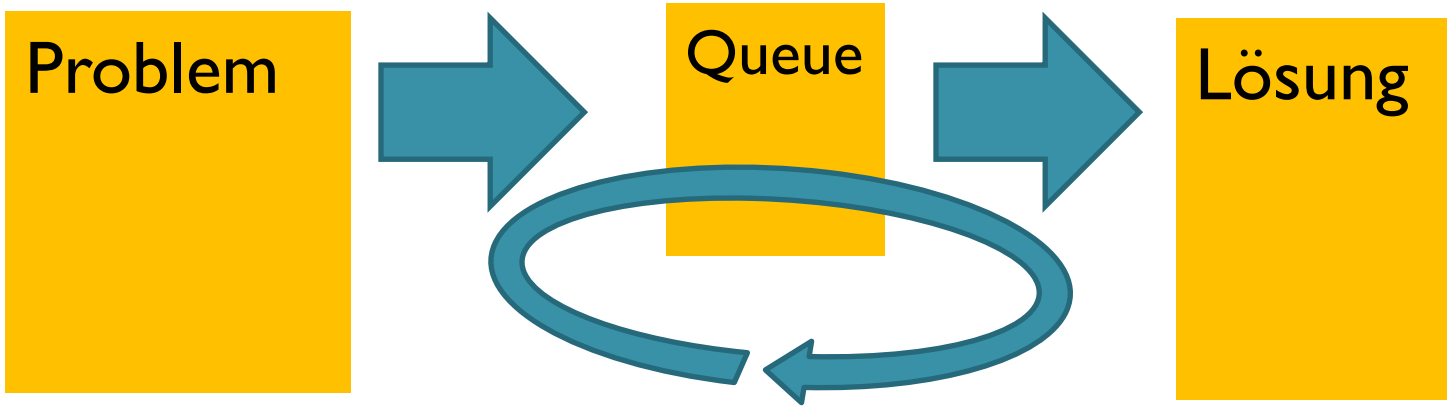
- backtracking nutzt System-stack zur Verwaltung des Wissens
- iterative Variante: Teillösungen in Stack<Teillösung> abspeichern



3.1 Vollständige Suche

Breite-zuerst

- iterativ
- Teillösungen in Queue<Teillösung>



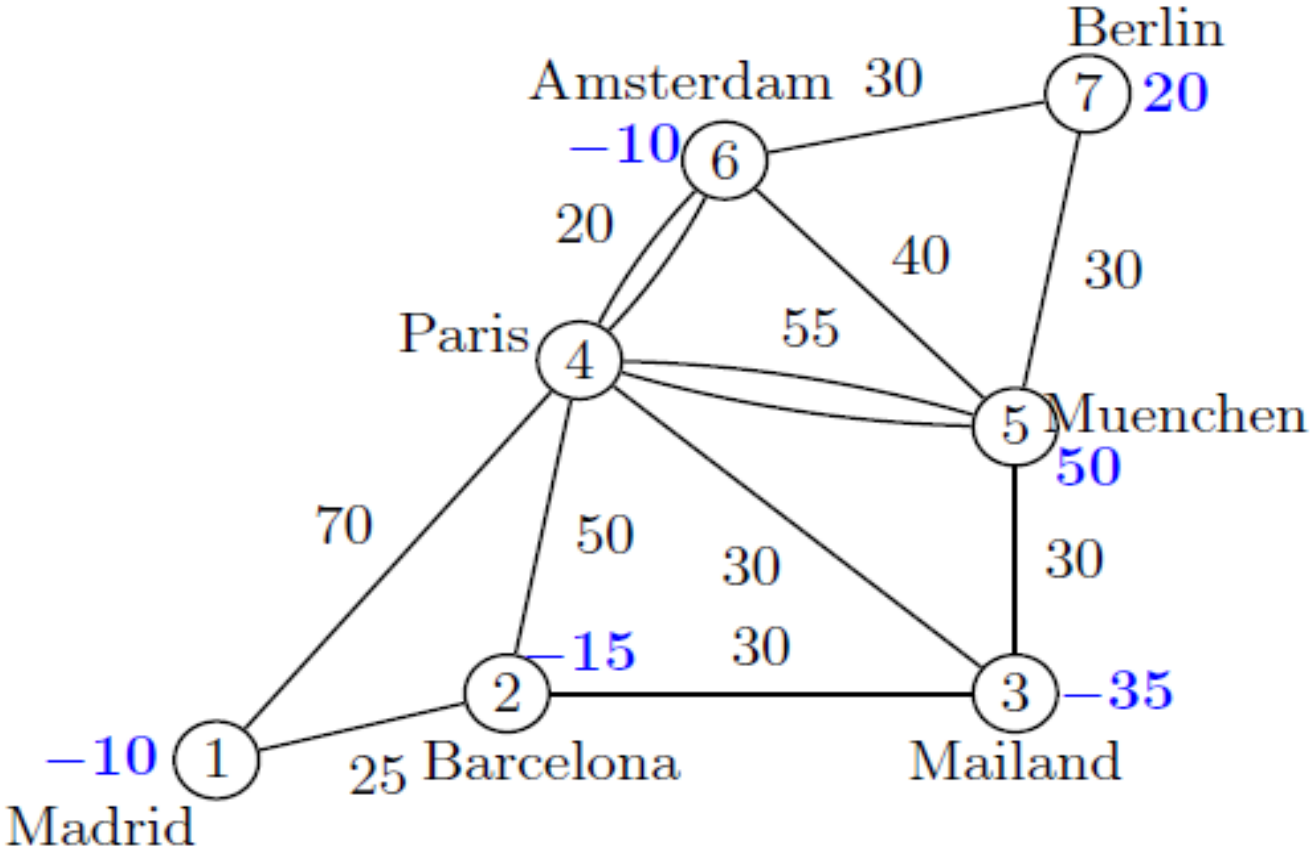
Fallstudie: Kürzeste Rundreise

Was ist die kürzeste Rundreise durch n Orte, zu denen eine Entfernungsmatrix gegeben ist?

1. Modellieren Sie die Teillösungen
2. Definieren Sie Ableitungsschritte und skizzieren Sie den sich daraus ergebenden Suchraum.
3. Wie traversieren Tiefensuche und Breitensuche diesen Suchraum?
4. Skizzieren Sie BFS und DFS für dieses Problem in Pseudocode

Fallstudie: Kürzeste Rundreise

TSP Beispiel: Landkarte



3.1 Vollständige Suche

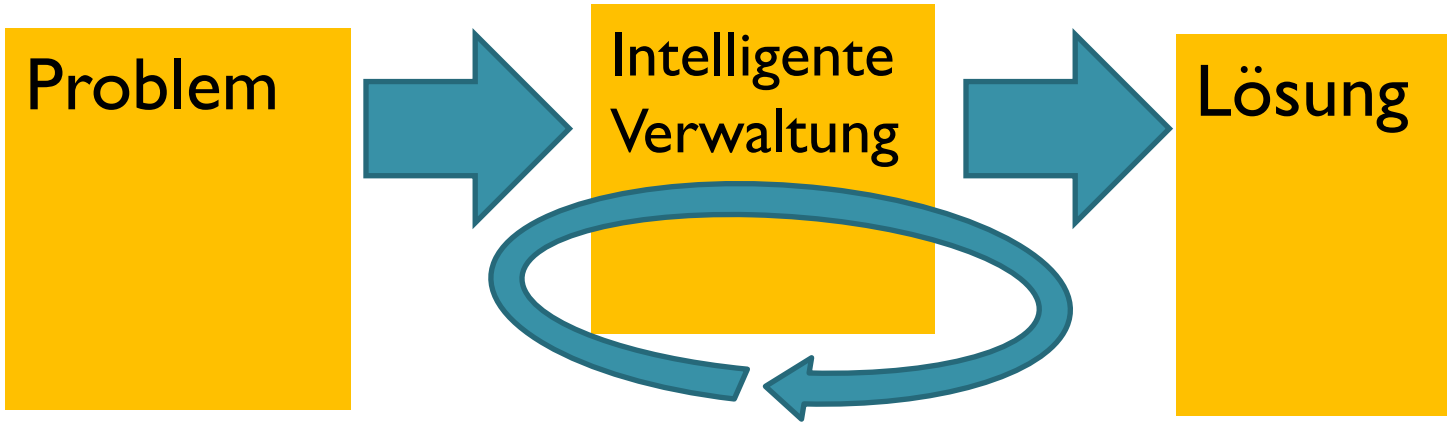
Vollständigkeit

- Alle $O(c^n)$ mit Ableitungsschritten erreichbaren Teillösungen werden berücksichtigt
- n = Anz. der Entscheidungsvariablen
- c = Anz. möglicher Werte für die Variablen
- Nachweis von „keine Lösung“ möglich
- Nachweis der Optimalität möglich
- ABER: Diese Nachweise kosten $O(c^n)$ Schritte

3.1 Vollständige Suche

Heuristiken (bei vollständiger Suche)

- Verbesserung der durchschnittlichen Performanz
- kein Verlust der Vollständigkeit (also gleiche Komplexität)
- Implementiert durch Präferenzen bei der Weiterentwicklung



3.1 Vollständige Suche

Branch and Bound

- Optimierung als Problemlöseaufgabe
- Verzögerte Entwicklung (bisher) schlechter Teillösungen
- Bound: Teillösungen werden (faktisch) nicht weiterentwickelt, wenn beweisbar ist, dass sie keine Verbesserung mehr bringen können

Fallstudie: Kürzeste Rundreise

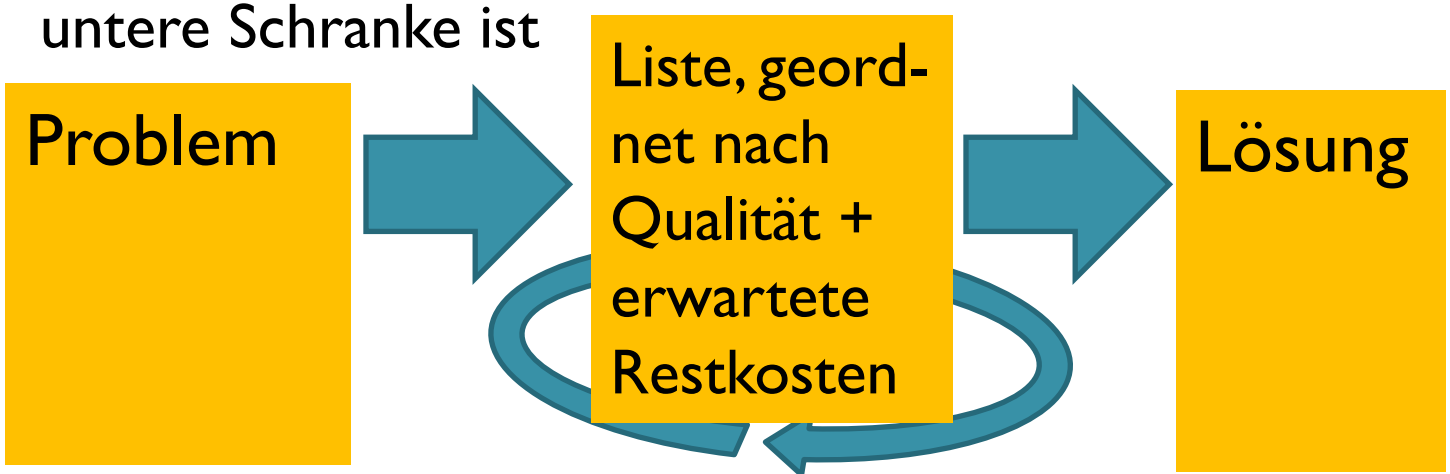
Wie kann BnB den Suchraum beim Finden einer kürzesten Rundreise (TSP) einschränken?

Wie kann BnB die Reihenfolge der weiter zu entwickelnden Teillösungen darüberhinaus verbessern?

3.1 Vollständige Suche

Best-First-Search / A*

- die Entwicklung schlechter Teillösungen wird verzögert/verworfen wie bei BnB
- Zusätzlich: vielversprechende Kandidaten werden durch Abschätzung und Verrechnung der zu erwartenden Restkosten bevorzugt
- Gleiche Komplexität, oft schneller
- Vollständigkeit bleibt erhalten, wenn Abschätzung eine untere Schranke ist



Fallstudie: Kürzeste Rundreise

Welche Untere Schranke lässt sich bei der Lösung von TSP für BestFirstSearch einsetzen?

Welchen Aufwand macht diese Berechnung der unteren Schranke?

Fallstudie: Golomb Lineal

Ein Golomb-Lineal der Länge n besteht aus n Markierungen, so dass die Differenz der Markierungen jeweils paarweise unterschiedlich ist. Gesucht ist für jedes n das kürzeste Lineal (d.h. ein Minimum für die größte Markierung)

Modellieren Sie das Problem, geben Ableitungsschritte an und skizzieren Sie den Suchraum. Welche Teile des Suchraums besuchen DFS, BFS und BnB, welche davon können mit A* noch ausgeschlossen werden, und wie?

Beispiele

1. Welche unteren Schranken gibt es bei der Suche nach kürzesten Wegen in Geo-Informationssystemen? Simulieren Sie die Suche mit A*

2. Welche Teile des Suchraums beim JSSP können durch BnB und welche weitern durch untere Schranken ausgeschlossen werden?

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Künstliche Neuronale Netze
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche
4. Wissensverarbeitung
   1. Aussagenlogik
   2. Prädikatenlogik
   3. Logisches Schließen

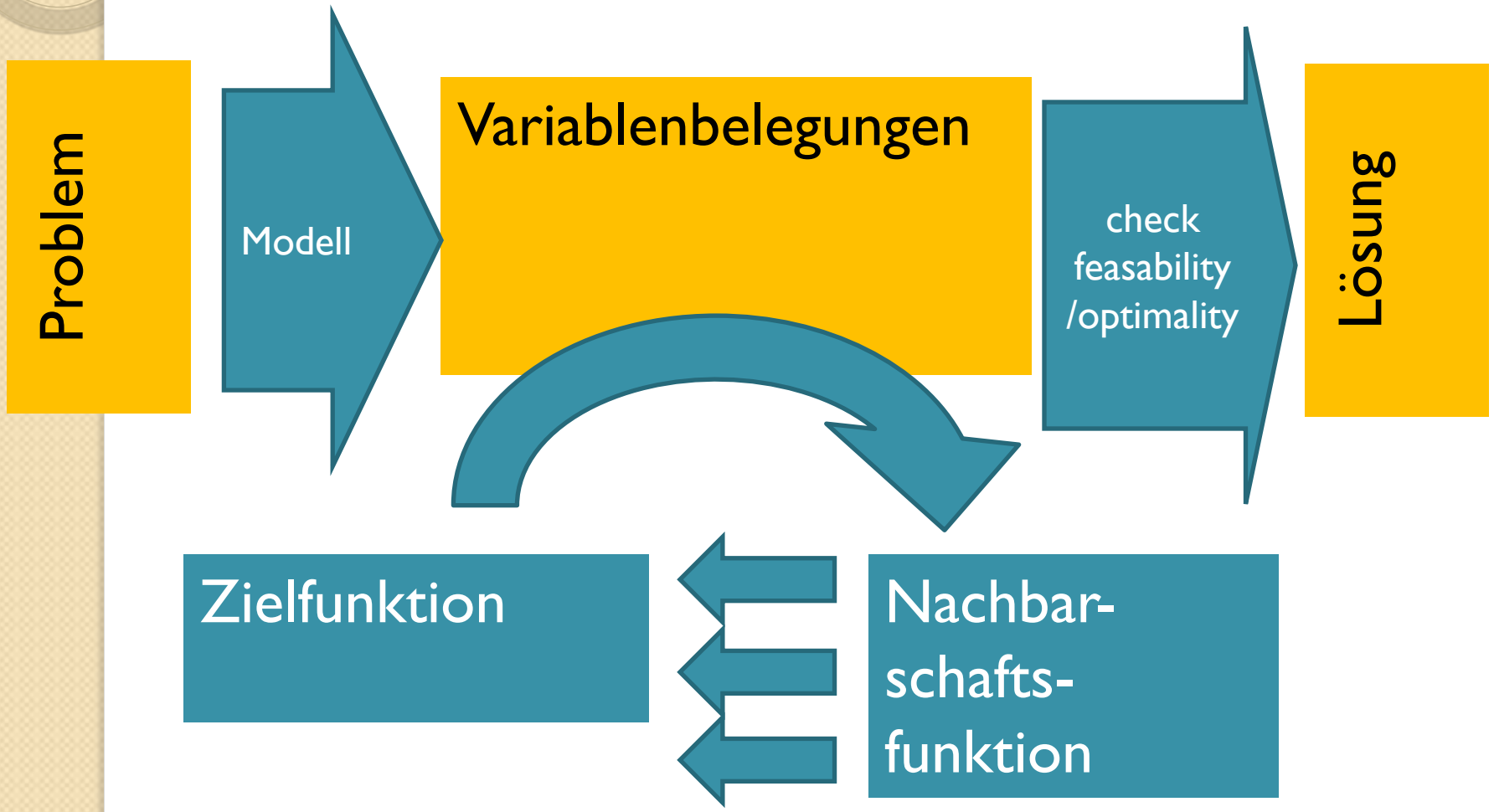
3.2 Unvollständige Suche

Local Search

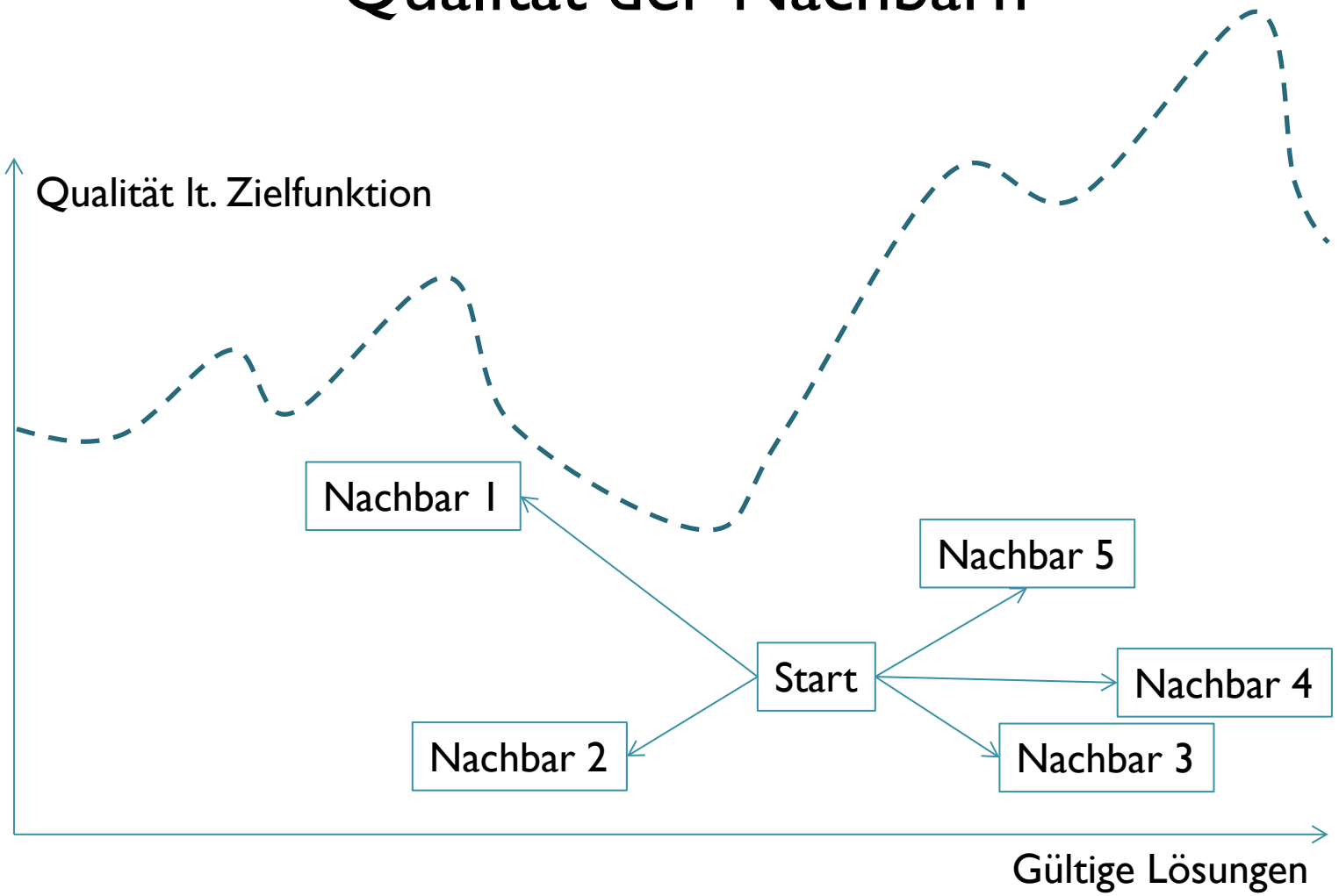
- A family of problem solving algorithms
- main algorithm:
  1. create an instantiation of all variables
  2. while(time left and not knowingly optimal) try to improve the instantiation by changing values
- can solve some large problems very quickly (e.g. 1 Mio queens in less than 1 minute)

3.2 Unvollständige Suche

Problemlösen mit Lokaler Suche



3.2 Unvollständige Suche



Beispiele

Definieren Sie jeweils Modell, eine Nachbarschaftsfunktion und eine Zielfunktion für folgende Suchprobleme:

- n-Queens
- Golomb Lineal
- Travelling Salesperson

3.2 Unvollständige Suche

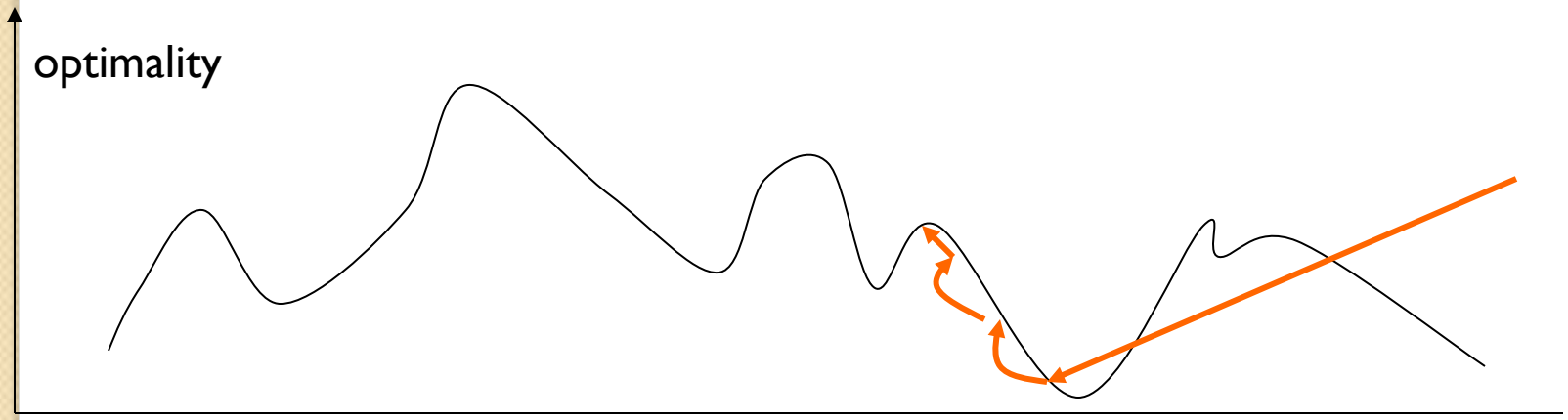
Der LS Grundalgorithmus: hill climbing

```
for i =1 to MaxTries // optional
  a = (random(a1),…,random(an));
  atTop=false;
  while(! atTop)
          N = neighbors(a)
          x = selectBest(N);
          if(is_worse(x,a)) atTop=true;
          else a=x;
   deliverSolution(a); // the local opt
```

3.2 Unvollständige Suche

Das Problem bei LS: lokale Optima

- Hill Climbing erreicht nur lokale Optima

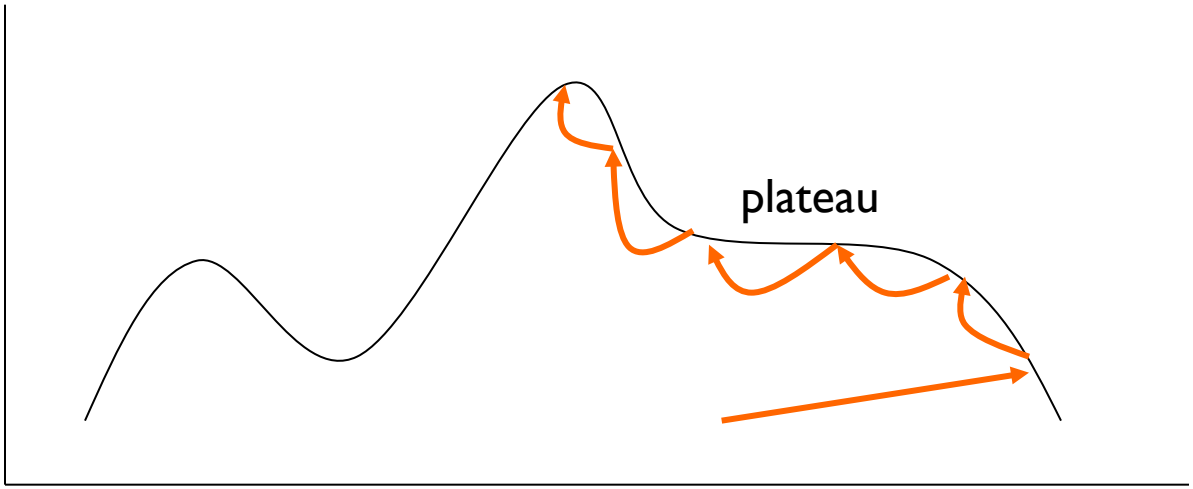


3.2 Unvollständige Suche

Heuristiken zum Verlassen lokaler Optima

1. Plateau Search: In Lokalen Optima dürfen nicht-verbessernde Nachbarn verfolgt werden

→Oszillation verhindern, Terminierung sichern



3.2 Unvollständige Suche

Heuristiken zum Verlassen lokaler Optima

2.) Tabu Search

- Speichern der n letzten Schritte/Veränderungen/Zustände/Belegungen
- Diese werden für die m nächsten Schritte nicht durchgeführt/erreicht
- Besonders geeignet zum Verhindern von Oszillation

3.2 Unvollständige Suche

Heuristiken zum Verlassen lokaler Optima

3.) Suchverfahren dynamisch anpassen:

- Ändern der Zielfunktion in Lokalen Optima z.B. Constraint Weighting
- Erweiterung der Nachbarschaftsfunktion in Lokalen Optima
- … wobei alle Anforderungen erfüllt bleiben

3.2 Unvollständige Suche

Heuristiken zum Verlassen lokaler Optima

4.) Simulated Annealing / Sintflut-Alg.

- Rauschen in Bewertungsfunktion einbeziehen
- So können verschlechternde Nachbarn ausgewählt werden
- Rauschen im Verlauf des Algorithmus reduzieren
- Oft ist weiterer Schutz vor Oszillation nötig

3.2 Unvollständige Suche

Fallstudie

Implementieren Sie Simulated Annealing zur Suche nach kürzesten Golomb-Linealen

3.2 Unvollständige Suche

Genetic Algorithm (GA)

- general purpose algorithm for problem solving
- adapting principles of natural selection
- decision variables are encoded in finite sequences
- instantiations are called **chromosomes** or **indviduals**
- a finite set of chromosomes is called **population**
- the quality of a chromosome is called ist **fitness**

3.2 Unvollständige Suche

Genetic Algorithm

```
INITIALIZATION of population;
repeat
   EVALUATION of fitness of all
    chromosomes;
   SELECTION of best chromosomes;
   RECOMBINATION of selected chromosomes;
   MUTATION randomly modify chromosomes;
   REPLACEMENT of parts of former
    population;
until terminating condition is met
```

3.2 Unvollständige Suche

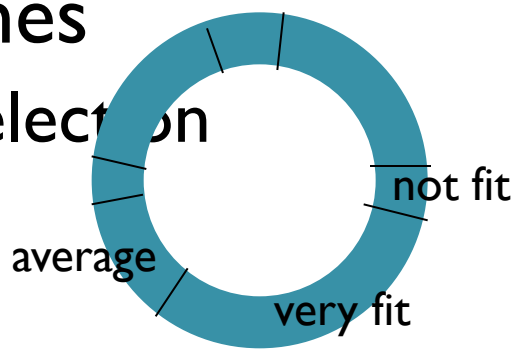
GA parameters

- population size: how many chromosomes are held (e.g. 50)
- selection: which may recombine (more to come on this) (e.g. 60%)
- recombination: parent number and offspring size (more to come) (e.g. 2→2)
- extent of mutation: likelyhood of random change of a value in a chromosome (e.g. 0.05)
- replacement quota: keep how many chromosomes of old polpulation (e.g. 90%)

3.2 Unvollständige Suche

GA selection operators

- fitness proportionate: random with bias to fitter chromosomes
  - e.g. roulette-wheel selection



- ordinal selection: select the best in disjunctive random subsets of population
  - e.g. tournament selection

3.2 Unvollständige Suche

GA recombination operators

- Many operators in literature, most are problem specific
- in „real problems“ problem specific are usually better
- generic operators exist:
  - k-point crossover: cut chromosomes at k points and mix genes in between
  - many more in [BurkeKendall05]

Beispiel

Implementieren Sie einen GA zum finden kürzester Rundreisen eines Handlungsreisenden (TSP)

3.2 Unvollständige Suche

GA extensions

- parallelization: easy to implement multi-process-version because of inherant parallelism
- hybrids
  - apply hill-climbing for each new member of population
  - enforce consistency in chromosomes
  - use domain specific heuristics in operators
- evaluation relaxation: use easy to compute approximation of fitness function

3.2 Unvollständige Suche

complexity

- worst case complexity: undefined (non-deterministic)
- average complexity: $O(|V|^2)$ evaluations of fitness function, according to [BurkeKendall05]
- performance is subject to statistical testing

3.2 Unvollständige Suche

things to know about GA

- Appliers use off-the-shelf implementations as OO-frameworks
- programming with GA is mainly
  - modeling the individuals
  - implementing the fitness function
  - Implementing crossover-function
  - adjusting parameters
- consider represenation in chromosomes carefully (bits, chars, objects?)

3.2 Unvollständige Suche

Erfolgsrezepte bei „Local Search“

- Erzeuge eine (gute) Initialisierung
- Verwende möglichst ausschließlich gültige Variablenbelegungen
- Nachbarschaftsfunktion muss potentiell alle Variablenbelegungen erreichen können
- Implementiere sehr schnelle Nachbarschafts- und Zielfunktion, LS lebt von vielen Versuchen

Implementierung von Local Search

Kochrezept für LS-Algorithmen

While(noch Entwicklungszeit)

Verbesserung der Nachbarschaftsfunktion für die typischen, zu erwartenden Probleme

forall(Parameter des Algorithmus)

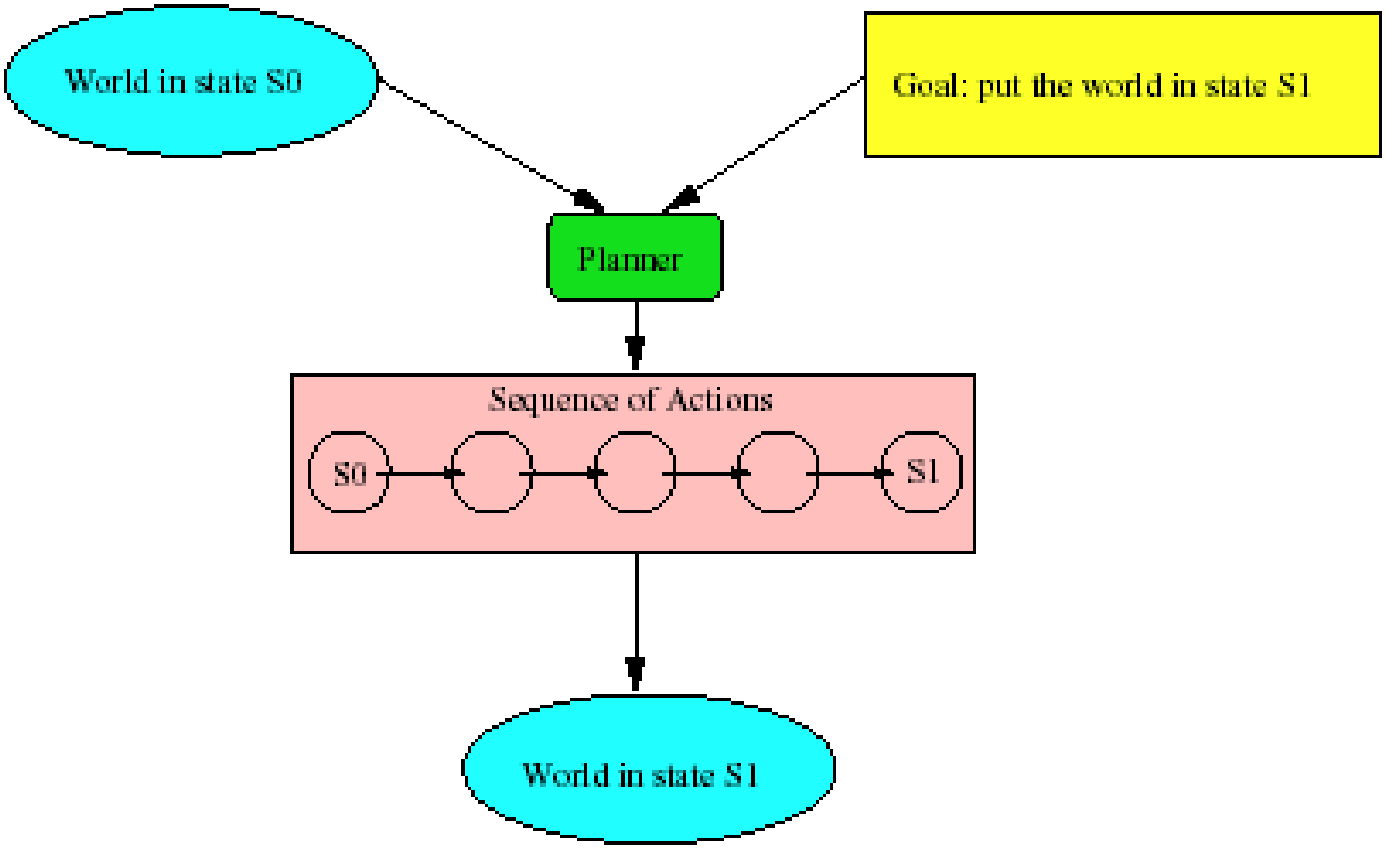
statistische Evaluierung der Nachbarschaftsfunktion mit den Parametern

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Künstliche Neuronale Netze
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche
   3. Planung
4. Wissensverarbeitung
   1. Aussagenlogik
   2. Prädikatenlogik
   3. Logisches Schließen

3.3 Planung

Was ist Planung (Planning)?



3.3 Planung

Planung vs. Suche

- Im Prinzip kein Unterschied:
  1. Probiere in jedem Zustand alle möglichen Erweiterungen aus
  2. Ist ein erreichter Zustand ein Zielzustand?
  3. Nein → goto 1.
- **Problem**: „Richtung unklar“ → Irrwege werden incht abgebrochen
- **Lösungsidee**: „gute“ Aktionen durch Heuristiken bevorzugt auswählen

3.3 Planung

Beispiel

Water-Bucket-Problem, ein klassisches Planungsproblem:

- Man hat drei Eimer, einem mit 8 l, einen mit 5l einen mit 3l Inhalt.
- Startzustand: der 8l Eimer ist voll.
- Zielzustand: im 8l-Eimer und im 5l-Eimer sind jeweils genau 4 Liter
- Wie muss man das Wasser hin und her schütten?
- (Wie verhält sich das, wenn es regnet?)

3.3 Planung

Situationskalkül

- Aktionen a überführen Zustände S_i in Zustände S_{i+1}, formal: $S_{i+1} = do(a,S_i)$
- Aktionen a dürfen nur durchgeführt werden, wenn bestimmte Bedingungen erfüllt sind formal: Poss(a,S) muss aus Wissensbasis ableitbar sein
- Aufbau einer Wissensbasis bei der Suche
- Logische Äquivalenzen und (Inferenz-)regeln zur Erweiterung und Anwendung des Wissens

3.3 Planung

Übung

Modellieren Sie das Water-Bucket-Problem im Situationskalkül

- Man hat drei Eimer, einem mit 8 l, einen mit 5l einen mit 3l Inhalt.
- Startzustand: der 8l Eimer ist voll.
- Zielzustand: im 8l-Eimer und im 5l-Eimer sind jeweils genau 4 Liter
- Wie muss man das Wasser hin und her schütten?

Wie kann man das Problem lösen?

- Ableitungsschritte
- Suchalgorithmus

3.3 Planung

Eigenschaften des Situationskalküls

- Hohe Komplexität
- Keine Nebenläufigkeit
- Rahmenproblem (Frameproblem): Es muss alles explizit dargestellt werden,
  - → auch alle negativen Fakten
  - → auch Dinge, die (noch) gar keine Rolle spielen
  - → wir brauchen eine closed world

3.3 Planung

STRIPS

- STanford Research Institute Problem Solver
- Anno 1971
- Name eines Systems (der Planer), später Name der formalen Eingabesprache für das System
- Idee: Alles bleibt gleich, außer es wird explizit anders angegeben (CWA), Zustände sind Konjunktionen von propositionalen Formeln→ Kein Rahmenproblem mehr
- Effizienter lösbar aber weniger mächtig als Situationskalkül

3.3 Planung

STRIPS

- Zustand: Eine Menge (Konjugation) von propositionalen Ausdrücken (zB PL/1)
- Operation: Name + Ein Quadrupel von Mengen von Zuständen
  1. die wahr sein müssen
  2. die falsch sein müssen

  } Vorbedingungen

  3. die wahr werden
  4. die falsch werden

  } Effekte der Operation
- Häufig werden 1+ 2 zu Vorbedingungen und 3+4 zu Nachbedingungen zusammengefasst

3.3 Planung

STRIPS

- Eine Planung in STRIPS ist gegeben durch eine Folge von Operatoren mit
  - Vorbedingungen jedes Operators sind durch vorherige Operatoren bzw Startzustand erfüllt
- Daraus ergibt sich eine Folge von Situationen
  - Beginnend mit Startzustand
  - Endend mit Zielzustand

Beispiel: Blocksworld: aispace.org/planning

3.3 Planung

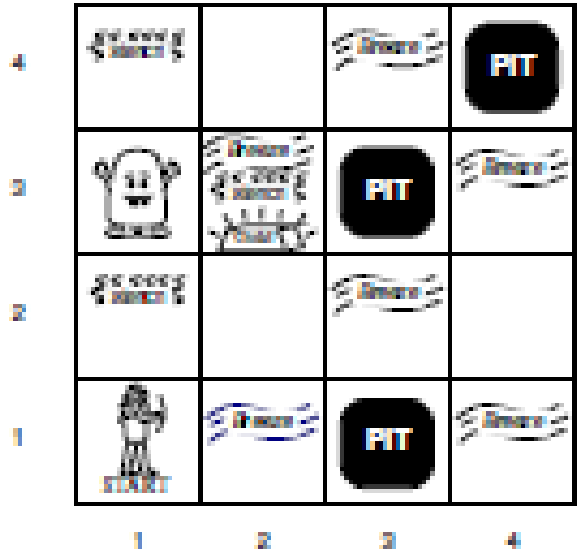
STRIPS, Beispiel

Wumpus-Welt

- Was sind die Operatoren
- Wie findet man eine Lösung

3.3 Planung

Example Domain: Wumpus World



- Want to get to the gold and grab it.
- Want to avoid pits and the "wumpus".
- Clues: breeze near pits and stench near the wumpus.
- Other sensors: wall (bump), gold (glitter), kill (scream)
- Actions: move, grab, or shoot.

3.3 Planung

Planung mit STRIPS

- Das Planungsproblem wird als Suchproblem gelöst.
- Lösung vorwärtsgerichtet: vom Anfangszustand zum Zielzustand: progression planning
  - ineffizient
- Lösung rückwärtsgerichtet vom Zielzustand zum Anfangszustand: regression planning
  - Effizienter
  - Betrachtet werden nur „zielgerichtete Aktionen“
- Mit STRIPS ist beides möglich

3.3 Planung

Regression Planning mit STRIPS

S:= alle positiven Literale des Zielzustands

Solange (S != Anfangszustand)

Wähle s aus S

Wähle Operation a, die s in add-list hat

Füge Vorbedingungen von a in S ein

- Beispiel: Blocksworld: Planning Applet : aispace.org/planning

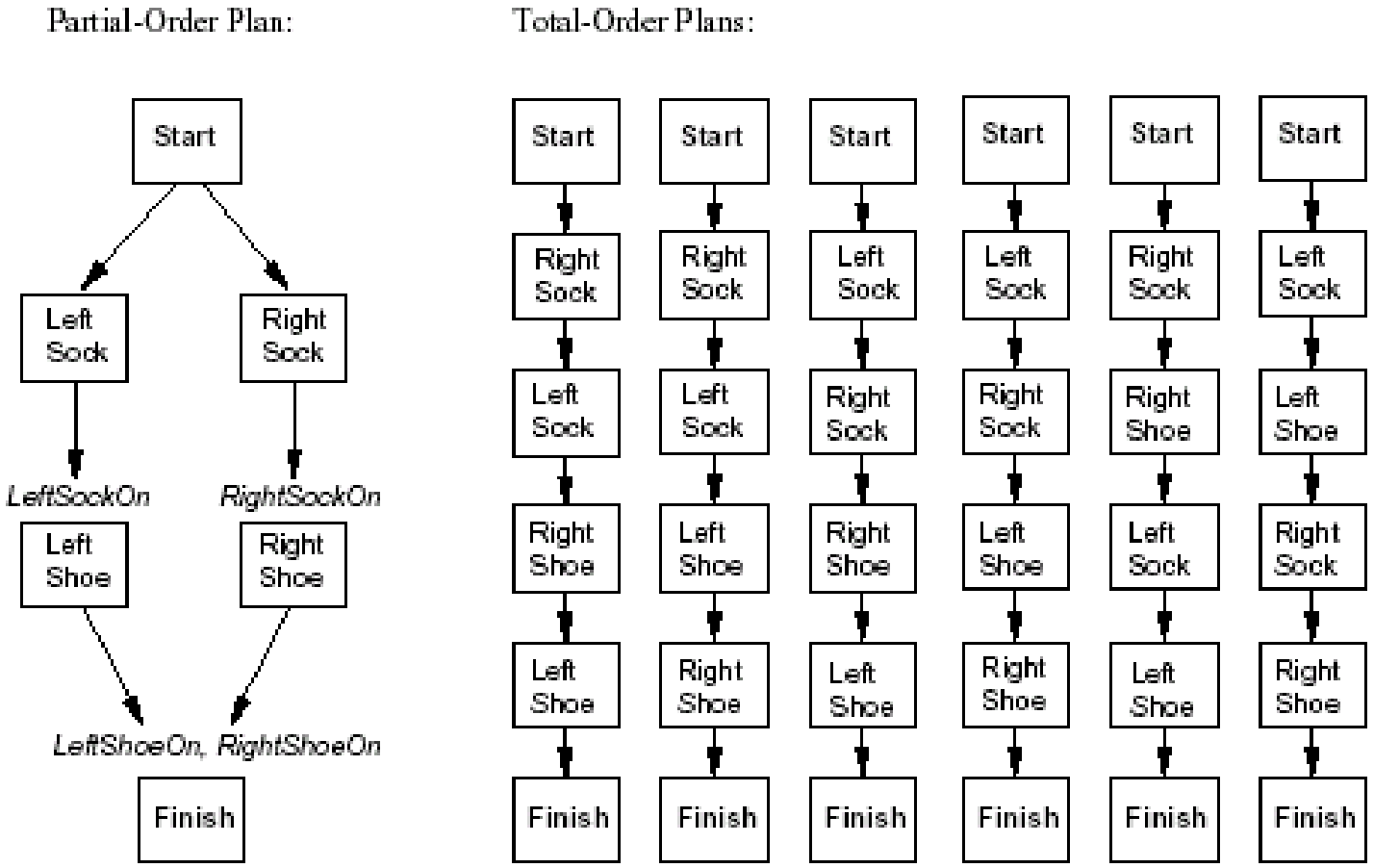
3.3 Planung

Partiell geordnetes Planung

- Teilzustände können sich unabhängig voneinander verändern
- Nutzung von Nebenläufigkeit möglich
- Verzicht auf Definition einer Ordnung über allen Aktionen

3.3 Planung

Beispiel Partiell geordnetes Planung [A. Frisch]



Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Künstliche Neuronale Netze
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche
   3. Planung
4. Wissensverarbeitung
   1. Aussagenlogik
   2. Prädikatenlogik
   3. Logisches Schließen