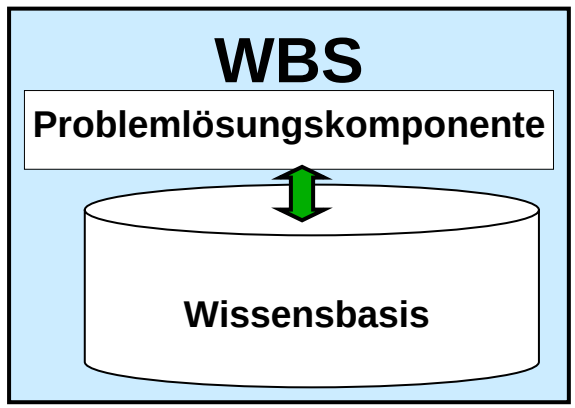
Wissensbasierte Systeme

Sebastian Iwanowski
FH Wedel

Kap. 3:
Algorithmische Grundlagen der KI

Suchstrategien

Warum sind Suchstrategien so wichtig in Wissensbasierten Systemen ?



Die Problemlösungskomponente muss fast immer ein Belegungsproblem für Constraints aus der Wissensbasis lösen !

All problem solvers search

Constraint Satisfaction Problem (CSP)

Spezifikation eines CSP:

- **Variablenmenge**
- **Definitionsbereiche (Domains)**
- **Constraints: Beziehungen zwischen den Variablen**
  (in der Regel Gleichungen oder Ungleichungen)

häufig auch noch dabei:

- **weiche Constraints**
  (Constraints dürfen verletzt werden)
- **Optimierungskriterium**
  (in der Regel Funktion der Variablen, die minimiert oder maximiert werden soll)

gültige Lösung:

Belegung aller Variablen mit Werten, sodass alle harten Constraints erfüllt sind

optimale Lösung:

gültige Lösung, die das Optimierungskriterium optimiert

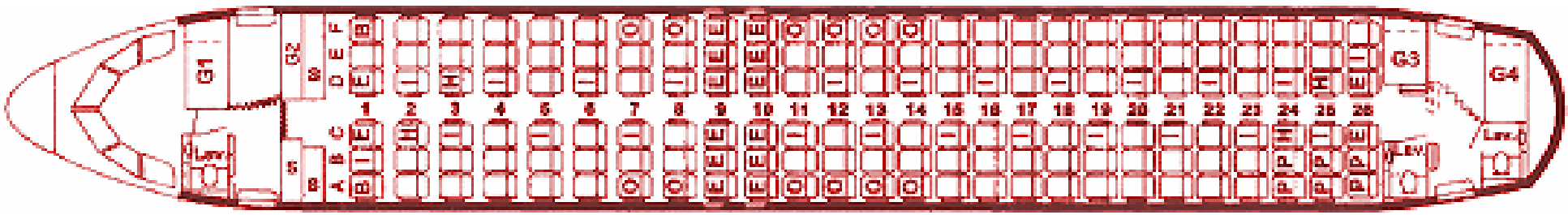
Constraint Satisfaction Problem (CSP)

Anwendungsbeispiele von CSP:

- **Technische Diagnose**
- **Technische Konfiguration**
- **Problem des Handelsreisenden (TSP)**
- **Problem des kürzesten Weges (Routing)**
- **Gewinnspiele**

Bsp. für Konfiguration

Kabinenlayout für Passagierflugzeuge



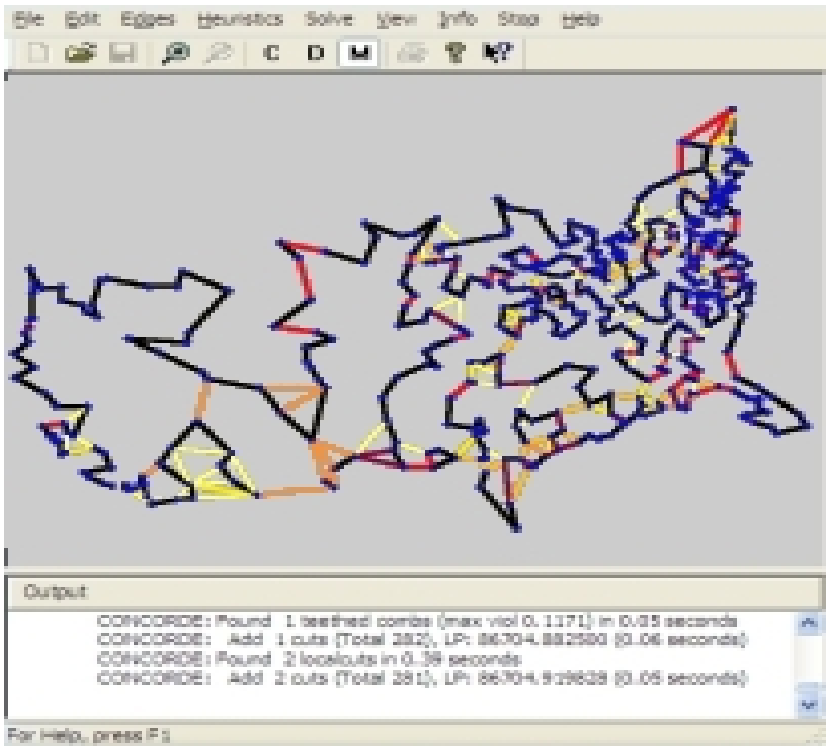
Platzierung der Kabineneinrichtung (Sitze, Küchen, Toiletten, etc.) unter Berücksichtigung von:

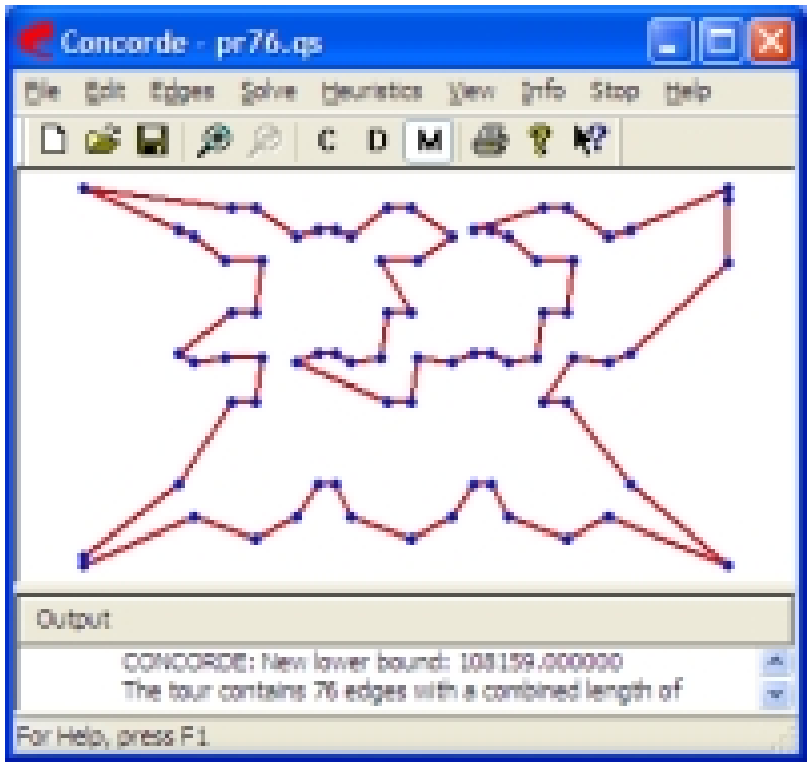
- Kundenwünschen
- Technischen Möglichkeiten
- Legalen Beschränkungen
- Optimalitätskriterien

Das Traveling Salesman Problem (TSP)

Problem:

Finde zu einer gegebenen Menge von Städten mit gegebenen Entfernungen die kürzeste Rundreise, die jede Stadt genau einmal durchquert.



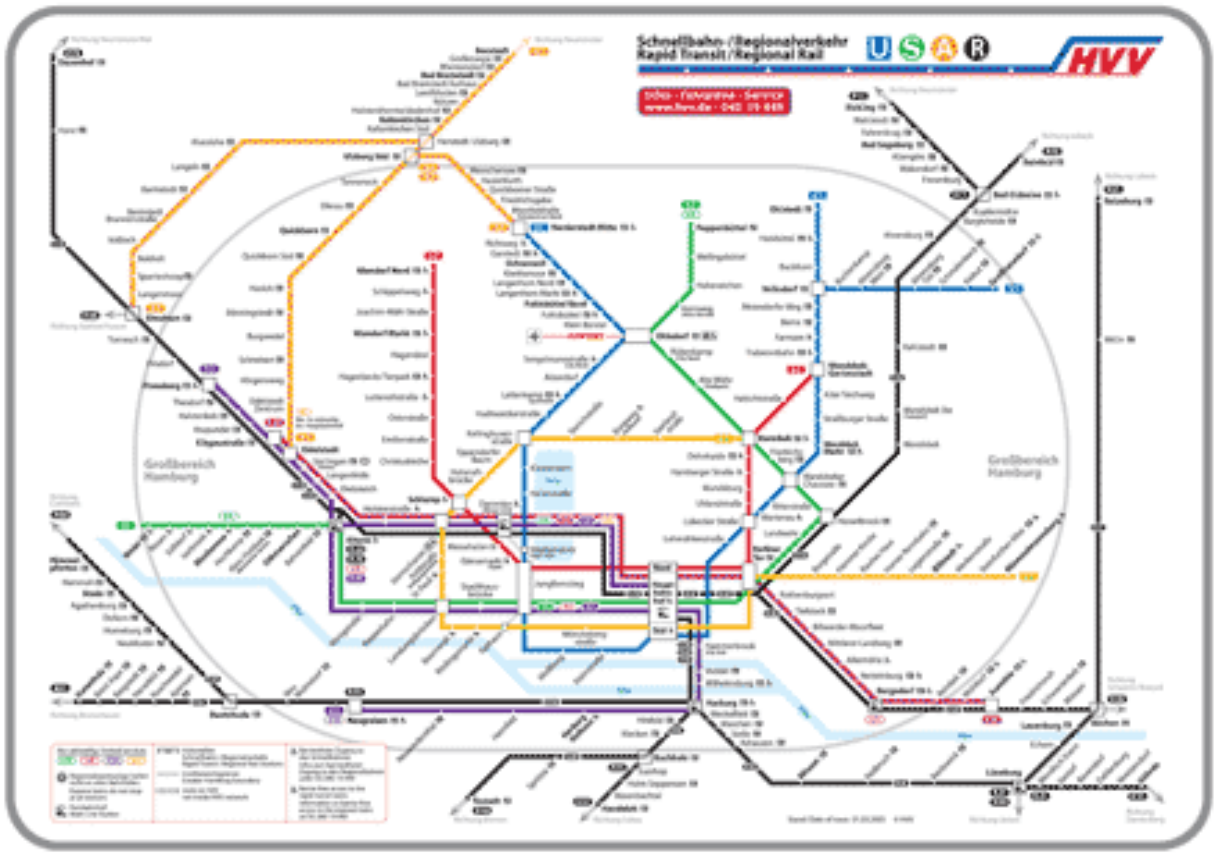


Quelle: http://www.tsp.gatech.edu//index.html

Das Problem des kürzesten Weges

Problem:

Finde zu zwei Punkten A und B in einem gegebenen Verkehrsnetz den kürzesten Weg von A nach B, der ausschließlich Strecken dieses Verkehrsnetzes benutzt.



Weitere Infos: Seminarvortrag und Ausarbeitung von Stefan Görlich, SS 2005, Nr. 5,
http://www.fh-wedel.de/~iw/Lehrveranstaltungen/SS2005/SeminarKI.html

Gewinnspiele

Problem:

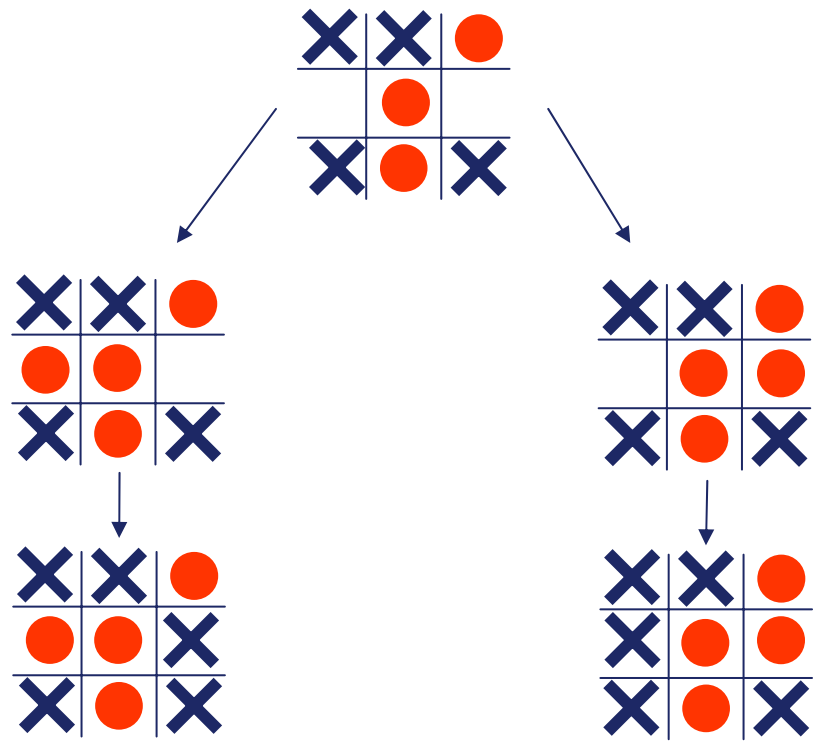
Finde eine Strategie von einem Startzustand zu einem Gewinnzustand

Bsp: Tic-Tac-Toe

Ausgangszustand

Folgezustand ●

Folgezustand ✕



Quelle und weitere Infos: Seminarvortrag und Ausarbeitung von Nils Böckmann, SS 2005, Nr. 13
http://www.fh-wedel.de/~iw/Lehrveranstaltungen/SS2005/SeminarKI.html

Suchen in Suchgraphen

Suchgraph:

- **Knoten: beschreibt Zustand in der Suchdomäne**
- **Kante: Übergang von einem Zustand in den nächsten**
  (in der Regel mit Richtung)
- **Startknoten: Anfangszustand**
  (ist immer eindeutig)
- **Zielknoten: gewünschter Endzustand (Lösung des Problems)**
  (es darf mehrere geben)

wünschenswert:

- **Suchgraph ist Suchbaum**
  (Pfad vom Startknoten zu jedem Zielknoten ist eindeutig)

Verschiedene Suchziele:

1) **Alle Lösungen eines Problems finden**
2) **Die beste Lösung finden**
3) **Ein paar gute Lösungen finden**

Lösen von CSP mit Suchbäumen

- **Zustand:** Belegung von Variablen mit Werten
- **Folgezustand:** Belegung einer weiteren Variable mit einem Wert oder einer Variable mit einem „besseren“ Wert unter Beibehaltung der Werte für die (anderen) bisher belegten Variablen
  Achtung: Manchmal darf nicht jeder Wert von einem alten aus erreicht werden!
- **Startknoten:** keine Variable hat einen Wert
- **Zielknoten:** gültige Lösung
- **Expansion** eines Knotens: Berechnung aller Folgeknoten

Verschiedene Suchstrategien unterscheiden sich in:

Welcher Knoten wird als nächstes expandiert ?

Uninformierte Suchstrategien für CSP

Im allgemeinen für CSP nur *blinde (uninformierte) Suche* möglich:

Es gibt keine Information über günstige Suchrichtungen (das Ziel wird erst bei Erreichen erkannt)

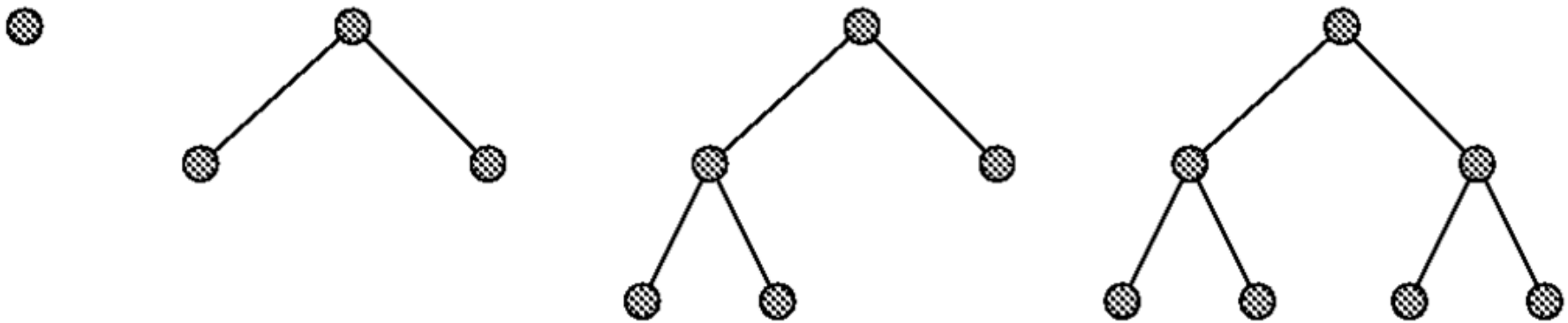
Die wichtigsten Suchstrategien:

1. Breitensuche (breadth-first-search)
2. Tiefensuche (depth-first-search)
3. Bestensuche (best-first-search)

Weitere Infos zum Thema Suchen: Seminarvortrag und Ausarbeitung von Sven Schmidt, SS 2005, Nr. 4
http://www.fh-wedel.de/~iw/Lehrveranstaltungen/SS2005/SeminarKI.html

Uninformierte Suchstrategien für CSP

Breitensuche (breadth-first-search):



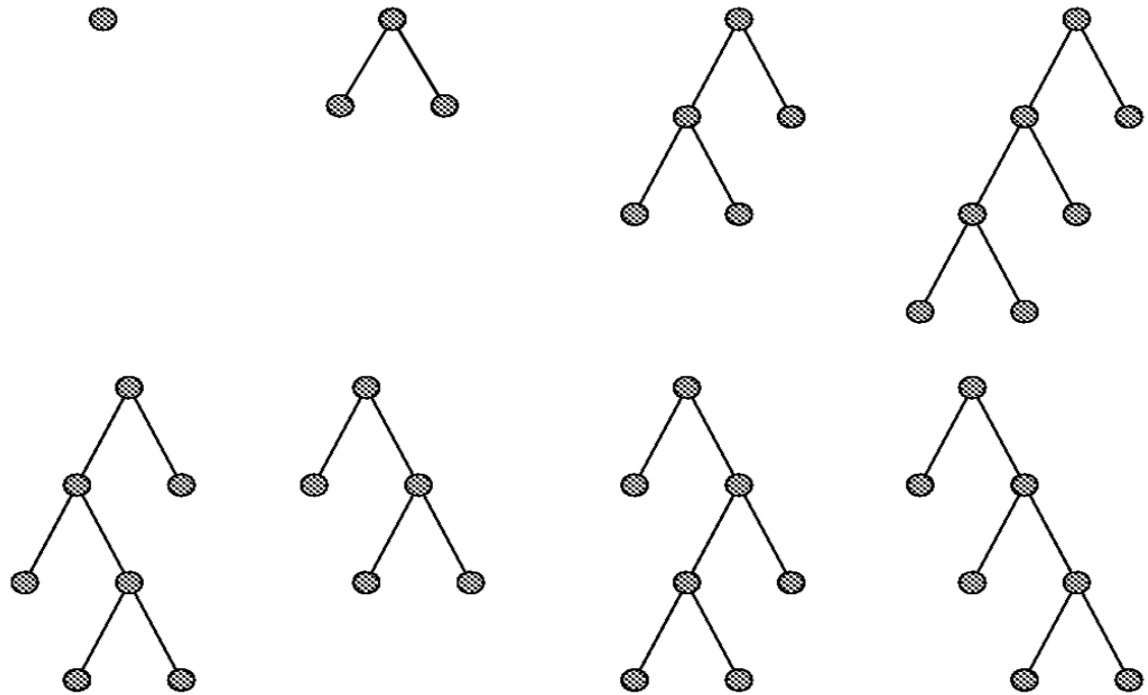
Exponentieller Aufwand für Zeit und Platz

Problemgröße: Tiefe des Suchbaums

Für CSP in den meisten Fällen nicht relevant

Uninformierte Suchstrategien für CSP

Tiefensuche (depth-first-search)



Exponentieller Aufwand für Zeit

Linearer Aufwand für Platz

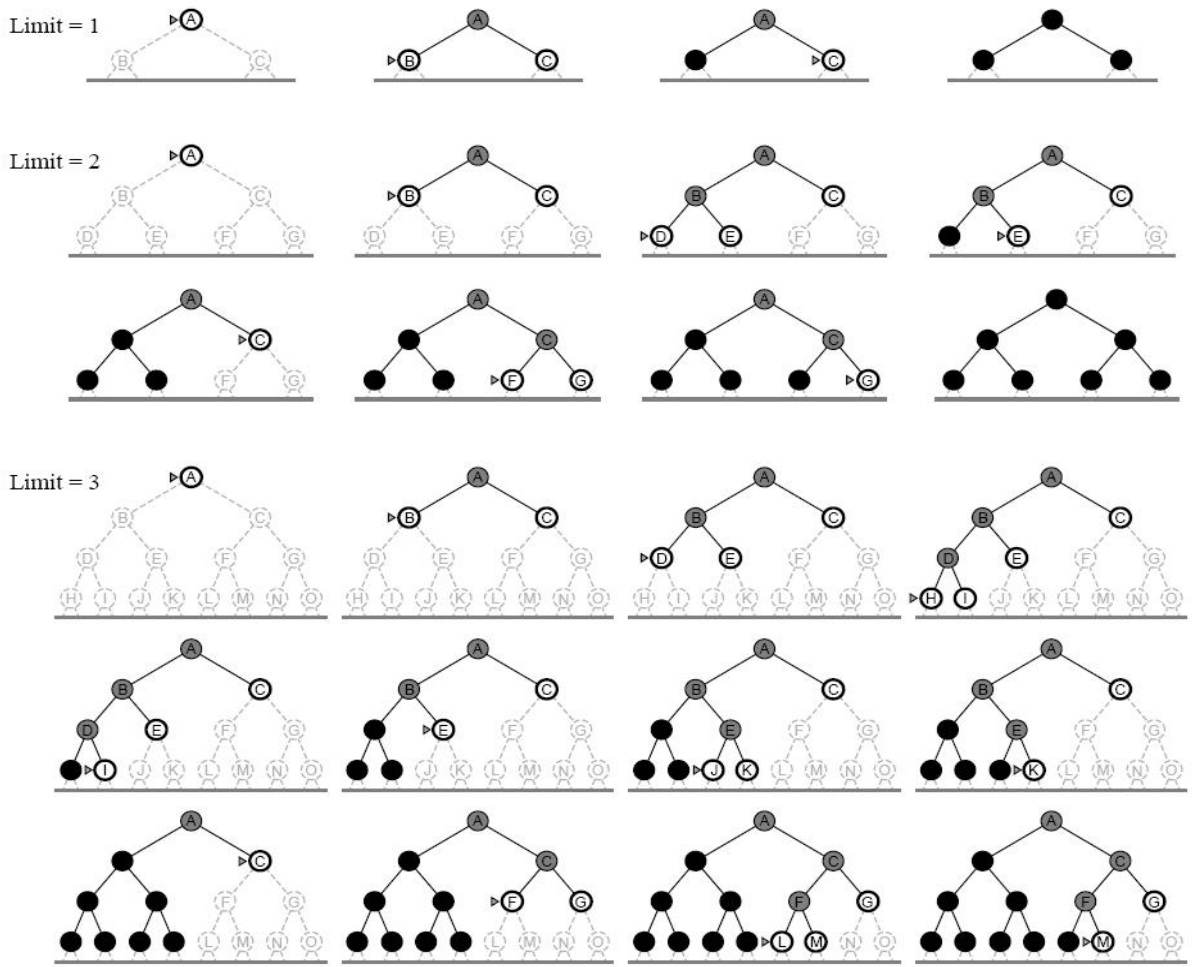
Der „Normalfall“ für allgemeine CSP

Problemgröße: Tiefe des Suchbaums

Uninformierte Suchstrategien für CSP

Beschränkte Tiefensuche

- Tiefensuche wird nur bis zu vorgegebener Suchtiefe durchgeführt.
- Bei Misserfolg kann die Suchtiefe nachträglich erhöht werden und die Tiefensuche neu starten.



Uninformierte Suchstrategien für CSP

Bestensuche (best-first-search)

- zusätzlich sei gegeben: Bewertungsfunktion für die Zustände
- Expandiere jeweils den Zustand mit bester Kostenbewertung

→ ***Mischung zwischen Tiefen- und Breitensuche***

Im *schlechtesten Fall* ist das nicht besser als Breitensuche:

Exponentieller Aufwand für Zeit und Platz

Problemgröße:
Tiefe des Suchbaums

Bei guten Bewertungsfunktionen ist das *Durchschnittsverhalten* viel besser!

Bei speziellen Problemen ist sogar der schlechteste Fall viel besser:

Bsp.: Spezialfall „Kürzeste-Wege-Problem“:

Algorithmus von Dijkstra (quadratischer Aufwand für Zeit, linearer für Platz)

Problemgröße: Anzahl der Knoten

Uninformierte Suchstrategien für CSP

Der Algorithmus von Dijkstra auf bewerteten Graphen

(Spezialfall von Best-First-Search)

Voraussetzung: Alle Kantenlängen müssen nichtnegativ sein.

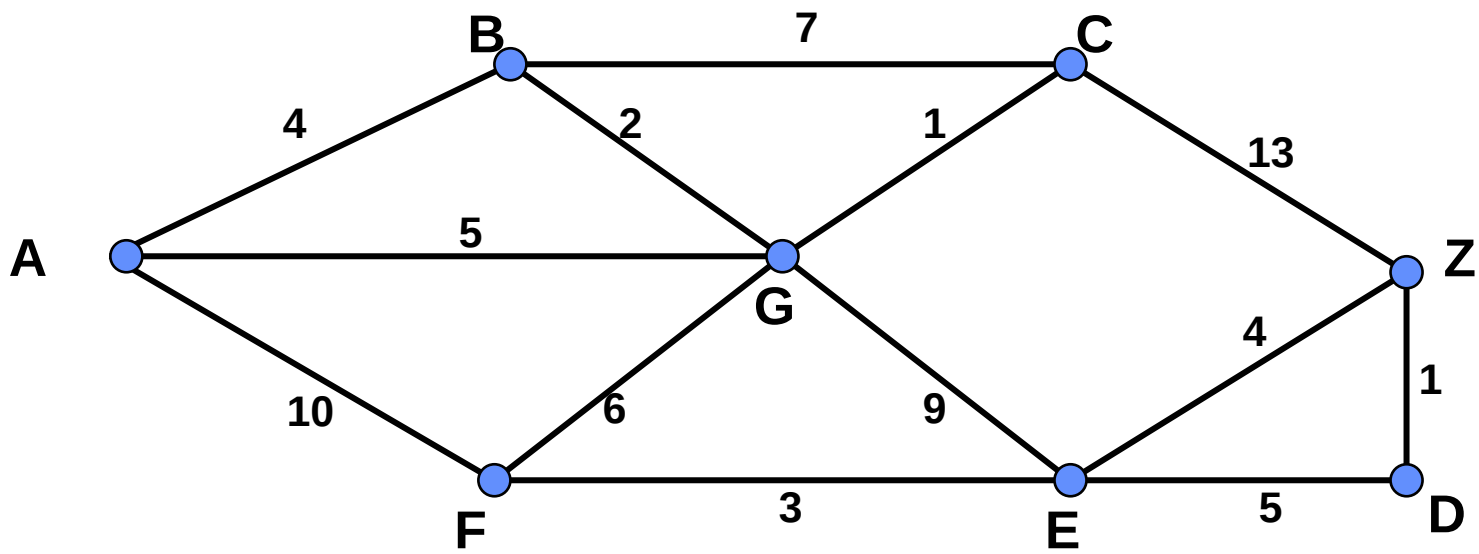
Ziel: Es soll der Weg mit der minimalen Kantenlänge von A nach B gefunden werden.

Algorithmus:

- In der Menge **`Berechnet`** sei nur die Ecke A. Markiere A mit *Weglänge* (A) = 0.
  In der Menge **`Unberechnet`** sind alle anderen Ecken des Graphen.
  Markiere die Nachbarn N von A mit *Weglänge* (N) := Länge der Kante von A nach N
  und alle anderen Ecken V mit *Weglänge* (V) = ∞.
- Wiederhole:
  Wähle die Ecke V aus **`Unberechnet`** mit der kleinsten *Weglänge* (V)
  und verschiebe sie in die Menge **`Berechnet`**.
  Betrachte alle Nachbarn N von V aus **`Unberechnet`**:
  Ersetze *Weglänge* (N) durch das Minimum der bisherigen *Weglänge* (N) und
  (*Weglänge* (V) + (Länge der Kante von V zu N)).

  bis V = B

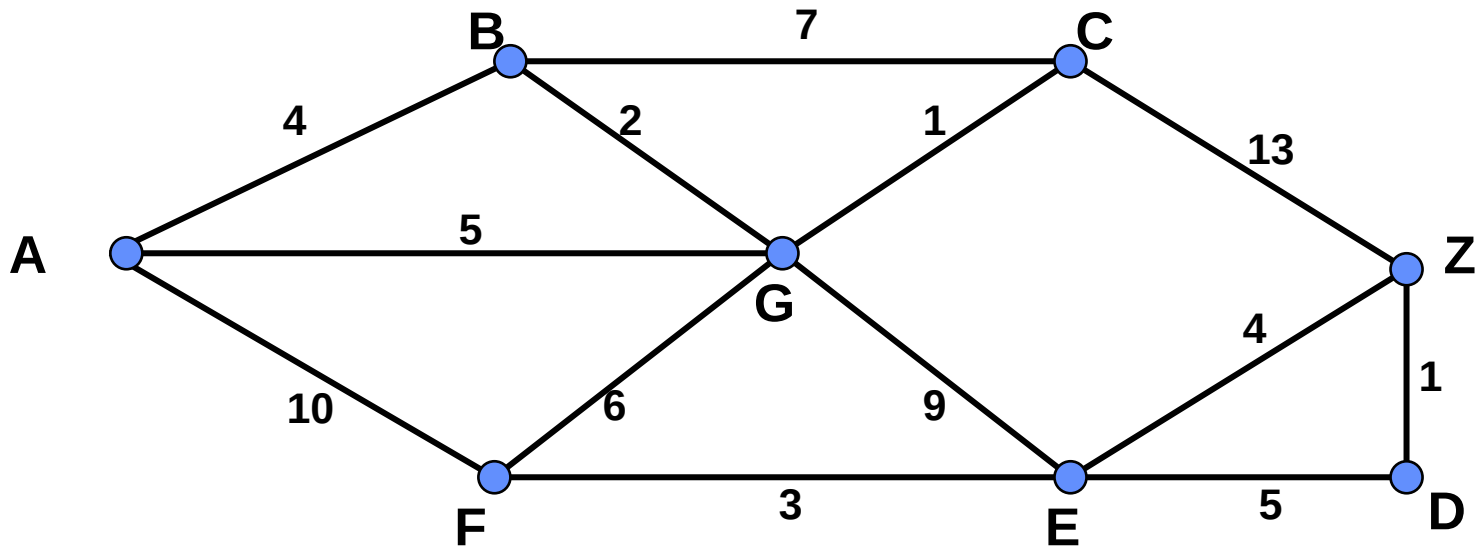
Beispiel für Algorithmus von Dijkstra



Kürzester Weg von A nach Z: A → F → E → Z (17 Einheiten)

Animation der Funktionsweise: siehe beiliegende ppt-Präsentation
Quelle und weitere Infos zum Algorithmus von Dijkstra:
Seminarvortrag und Ausarbeitung von Alex Prentki, WS 2004, Nr. 14
http://www.fh-wedel.de/~iw/Lehrveranstaltungen/WS2004/SeminarMC.html

Beispiel für Algorithmus von Dijkstra



Kürzester Weg von G nach Z: G → E → Z (13 Einheiten)

Knoten (Wegstrecke von G, direkter Vorgänger):

A(5,G)		A(5,G)		A(5,G)					
B(2,G)		B(2,G)							
C(1,G)									
D(∞)	→	D(∞)	→	D(∞)	→	D(∞)	→	D(∞)	→ D(14,E)
E(9,G)		E(9,G)		E(9,G)		E(9,G)		E(9,G)	
F(6,G)		F(6,G)		F(6,G)		F(6,G)			
Z(∞)		Z(14,C)		Z(14,C)		Z(14,C)		Z(14,C)	Z(13,E)

Informierte (Heuristische) Suchstrategien für CSP

Gegebene Zusatzinformation:

Schätzfunktion h(Zustand) als Maß für die Entfernung zu einem Zielknoten

- nicht zu aufwändig
- aber genau genug, um Suchfunktion nicht in die Irre zu führen

h() liefert einen positiven Wert: Je kleiner der Wert, desto näher der Zielknoten

Anwendung: „Bergsteigen“

- Spezialform der Tiefensuche
- Es wird genau der Knoten expandiert, der den besten Schätzfunktionswert aufweist
- Beim Aufsteigen im Suchbaum wird der jeweils nächstbeste Knoten expandiert.

Informierte (Heuristische) Suchstrategien für CSP

Gegebene Zusatzinformation:

Schätzfunktion h(Zustand) als Maß für die Entfernung zu einem Zielknoten

- nicht zu aufwändig
- aber genau genug, um Suchfunktion nicht in die Irre zu führen

h() liefert einen positiven Wert: Je kleiner der Wert, desto näher der Zielknoten

Anwendung: Optimistisches Bergsteigen

- Spezialform der Tiefensuche
- Es wird nur der Knoten berechnet, der den besten Schätzfunktionswert aufweist
- Zurücksetzen ist nicht möglich: Wenn Schätzfunktion Fehler macht, wird kein Ergebnis gefunden.

Informierte (Heuristische) Suchstrategien für CSP

Gegebene Zusatzinformation:

Schätzfunktion h(Zustand) als Maß für die Entfernung zu einem Zielknoten

- nicht zu aufwändig
- aber genau genug, um Suchfunktion nicht in die Irre zu führen

h() liefert einen positiven Wert: Je kleiner der Wert, desto näher der Zielknoten

Anwendung: A*-Verfahren

- Spezialform der Bestensuche
- Es wird genau der Knoten expandiert, bei dem die Summe von Kostenbewertung und Schätzfunktionswert optimal ist.

Weitere Infos für die Anwendung von A* in öffentlichen Verkehrsnetzen:
Seminarvortrag und Ausarbeitung von Stefan Görlich, SS 2005, Nr. 5
http://www.fh-wedel.de/~iw/Lehrveranstaltungen/SS2005/SeminarKI.html

Informierte (Heuristische) Suchstrategien für CSP

Der A*-Algorithmus auf bewerteten Graphen

(Verallgemeinerung des Algorithmus von Dijkstra)

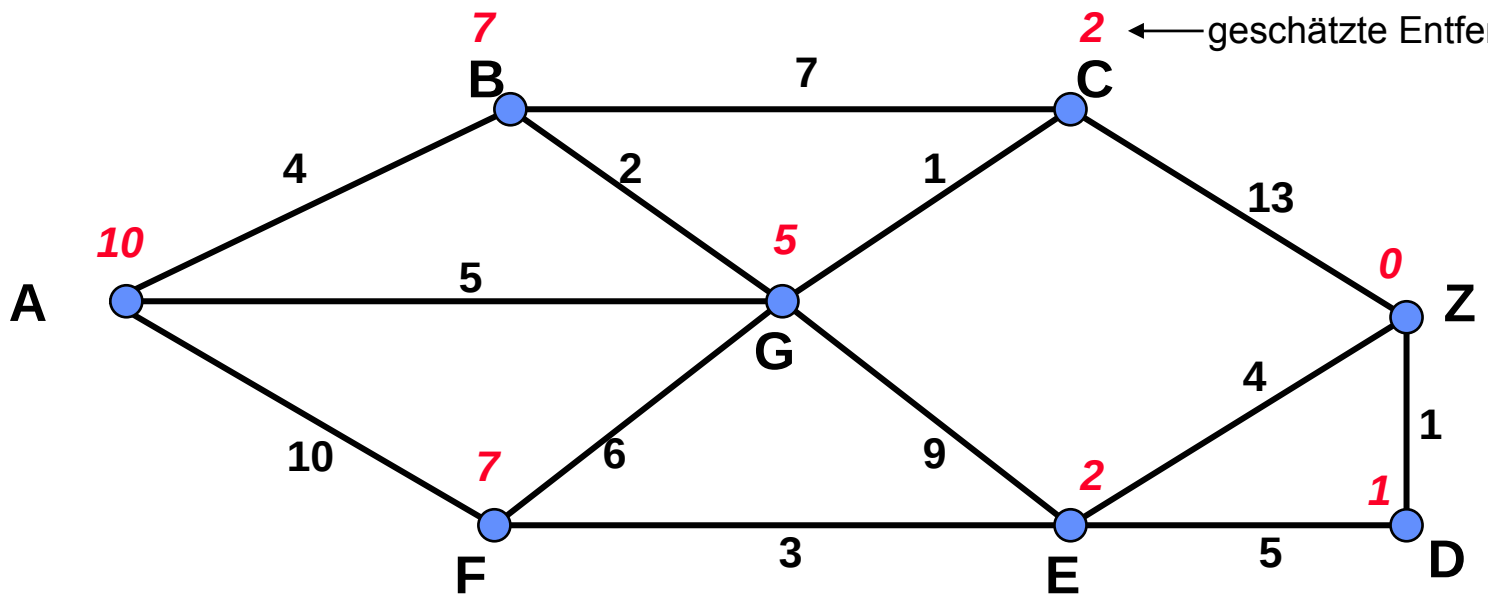
Voraussetzung: Alle Kantenlängen müssen nichtnegativ sein.
Schätzfunktion $h_z(x)$ für Weg $w_z(x)$ von x zum Ziel z mit $h_z(x) \leq w_z(x)$

Ziel: Es soll der Weg mit der minimalen Kantenlänge von A nach B gefunden werden.

Algorithmus:

- In der Menge **`Berechnet`** sei nur die Ecke A. Markiere A mit Weglänge (A) = 0.
  In der Menge **`Unberechnet`** sind alle anderen Ecken des Graphen.
  Markiere die Nachbarn N von A mit *Weglänge* (N) := Länge der Kante von A nach N
  Schätzung (N) := *Weglänge* (N) + h_B(N)
  und alle anderen Ecken V mit *Weglänge* (V) = ∞ und *Schätzung* (V) = ∞.
- Wiederhole:
  Wähle die Ecke V aus **`Unberechnet`** mit der kleinsten *Schätzung* (V)
  und verschiebe sie in die Menge **`Berechnet`**.
  Betrachte alle Nachbarn N von V aus **`Unberechnet`**:
  Ersetze *Weglänge* (N) durch das Minimum der bisherigen *Weglänge* (N) und (*Weglänge* (V) + (Länge der Kante von V zu N)).
  Aktualisiere Schätzung (N) := *Weglänge* (N) + h_B(N) (falls notwendig).
  bis V = B

Beispiel für A*-Algorithmus



Kürzester Weg von G nach Z: G → E → Z (13 Einheiten)

Knoten (Wegstrecke von G, direkter Vorgänger, Schätzung zum Ziel):

A(5,G,15)		A(5,G,15)		A(5,G,15)		A(5,G,15)
B(2,G,9)		**B(2,G,9)**				
C(1,G,3)						
D(∞)	→	D(∞)	→	D(∞)	→	D(14,E,15)
E(9,G,11)		E(9,G,11)		**E(9,G,11)**		
F(6,G,13)		F(6,G,13)		F(6,G,13)		F(6,G,14)
Z(∞)		Z(14,C,14)		Z(14,C,14)		**Z(13,E,13)**

Allgemeine Optimierungsverfahren für CSP

Zurücksetzen (Backtracking)

- Teste alle Constraints auch bei unvollständigen Variablenbelegungen
- Zustände, die irgendwelche Constraints bereits verletzen, werden nicht weiter expandiert

Vorwärtstest (Forward Checking)

- Reduziere alle Domains für alle noch nicht belegten Variablen, sodass keine Konflikte zwischen Constraints mehr entstehen.
- Setze zurück, wenn die Domains dadurch leer werden.

Allgemeine Optimierungsverfahren für CSP

Verfahren der minimalen Konflikte (Min-Conflicts)

Idee:

- Start mit einer beliebigen Wertebelegung
- Auswählen von Variablen und Zuweisung neuer Werte, die weniger Konflikte verursachen solange, bis System gelöst

Vorteile:

- bei vielen Praxis-Problemen gutes Laufzeitverhalten
- „Reparieren“ bei kleinen Veränderungen des Systems

Nachteile:

- „Hängen bleiben“ in lokalen Minima
  - Gegenmaßnahmen: Random-Walk, Tabu-Liste, ...

Quelle und weitere Details zum Thema Constraintsysteme:
Seminarvortrag und Ausarbeitung von Stefan Schmidt, SS 2005, Nr. 6,
http://www.fh-wedel.de/~iw/Lehrveranstaltungen/SS2005/SeminarKI.html