

# ***Künstliche Intelligenz***

Sebastian Iwanowski  
FH Wedel

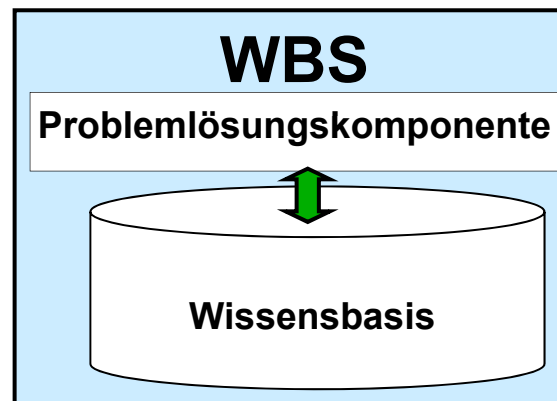
**Kap. 3:**  
KI-Algorithmik

# Suchstrategien

**Bedeutung von Suchstrategien für logisch formulierte Probleme:**

**Suche nach Lösung fürs Erfüllbarkeitsproblem**

**Bedeutung von Suchstrategien für Wissensbasierte Systeme:**



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

**➔ *All problem solvers search***

# Bsp. für wissensbasierte Suchmaschine: PROLOG

## PROLOG ist wissensbasiert:

- **Wissensbasis**

Fakten und Regeln, dynamisch erweiterbar

- **Inferenzmaschine**

Automatische Herleitung neuer Fakten und Regeln mit Resolution und Unifikation

- **Dialogkomponente**

**Eingabe:** Fragen

**Ausgabe:** yes / no, Angabe der Unifikation im Erfolgsfall, Write als „Seiteneffekt“

Yes: Das Prädikat der Frage folgt aus der Wissensbasis.

No: Das Prädikat der Frage folgt nicht aus der Wissensbasis.

*No impliziert nicht, dass das Prädikat als falsch abgeleitet werden kann.*

# **Anwendung: Das Stundenplanproblem (Scheduling)**

**Gegeben endliche Mengen Fächer, Räume, T(Zeiten)**

**Aufgabe: Generierung einer injektiven Funktion  $F \rightarrow R \times T$**

**Nebenbedingungen:**

- **Bestimmte Fächer dürfen nicht zur selben Zeit stattfinden**
- **Nicht jedes Fach darf zu jeder Zeit stattfinden**
- **Nicht jedes Fach darf in jedem Raum stattfinden**

**Weiche Kriterien (dürfen verletzt werden):**

- **Bestimmte Fächer sollen zu bestimmten Zeiten möglichst nicht stattfinden**
- **Bestimmte Fächer sollten möglichst hintereinander stattfinden**
- **Bestimmte Fächer sollten möglichst nicht am selben Tag stattfinden**

**Optimierungsfunktion:**

- **Möglichst wenige Verletzungen von weichen Kriterien**
- **Möglichst wenige Freistunden für Studiengänge**
- **Möglichst gleichmäßige Verteilung auf verschiedene Tage für ...**

# **Anwendung: Das Traveling Salesman Problem (TSP)**

**Gegeben:** Graph mit Knotenmenge  $V$  und bewerteten Kanten zwischen Knoten

**Aufgabe:** Finde Rundreise durch den Graphen,  
die alle Knoten mindestens einmal erreicht.

Nebenbedingungen:

- **Es dürfen nur Kanten des Graphen benutzt werden**

Optimierungsfunktion:

- **Möglichst geringe Gesamtbewertung**

## **Verallgemeinerung in Anwendungen der Logistik:**

Nebenbedingungen:

- **Aufnahme und Auslieferung von Gütern mit Beachtung der Ladekapazitäten**
- **Zeitfenster, wann welche Knoten erreicht werden dürfen**

Weiche Kriterien (dürfen verletzt werden):

- **Bestimmte Kanten sind zu vermeiden**
- **Bestimmte Zeitfenster sind ungünstig**

# **Anwendung: Das Problem des kürzesten Weges**

**Gegeben:** Graph mit Knotenmenge  $V$  und bewerteten Kanten zwischen Knoten

**Aufgabe:** Zu zwei ausgewählten Knoten  $S$  und  $T$ , finde einen Weg durch den Graphen.

Nebenbedingungen:

- **Es dürfen nur Kanten des Graphen benutzt werden**

Optimierungsfunktion:

- **Möglichst geringe Gesamtbewertung**

**Verallgemeinerung in Verkehrsanwendungen (ÖPNV oder Individualverkehr):**

Nebenbedingungen:

- **Kantenbewertungen sind zeitabhängig**
- **Reisender unterliegt Beschränkungen, die bestimmte Kanten individuell anders bewerten bzw. unbenutzbar machen.**

Weiche Kriterien (dürfen verletzt werden):

- **Bestimmte Kanten sind zu vermeiden**
- **Bestimmte Zeitfenster sind ungünstig**

# 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**

# Suchen in Suchgraphen

## Suchmethode: Finden einer Gesamtlösung über Teillösungen

- **Knoten: beschreibt Zustand in der Suchdomäne**
  - Zustand: Belegung von Variablen mit Werten  
**Jeder Zustand hat eine Bewertung.**
- **Kante: Übergang von einem Zustand in einen Folgezustand**  
(in der Regel mit Richtung)
  - Folgezustand: Belegung einer weiteren Variable mit einem Wert unter Beibehaltung der Werte für die bisher belegten Variablen
- **Startknoten: Anfangszustand**  
(ist immer eindeutig)
  - Startknoten: keine Variable hat einen Wert.
- **Zielknoten: gewünschter Endzustand (Lösung des Problems)**  
(es darf mehrere geben)
  - Zielknoten: Alle gewünschten Variablen haben zulässige Werte



# Suchen in Suchgraphen

## Suchmethode: Systematisches Verbessern v. Gesamtlösungen

- **Knoten: beschreibt Zustand in der Suchdomäne**
  - **Zustand: Belegung von allen Variablen mit Werten**  
(nicht notwendigerweise alle zulässig)  
**Jeder Zustand hat eine Bewertung.**
- **Kante: Übergang von einem Zustand in einen benachbarten Zustand**  
(in der Regel in zwei Richtungen möglich)
  - **Nachbarzustand: Neubelegung von bestimmten Variablen unter Beibehaltung der Werte für alle anderen Variablen**
- **Startknoten: Anfangszustand**  
(ist immer eindeutig)
  - **Startknoten: Mit irgendeiner Belegung für alle Variablen fängt man an.**
- **Zielknoten: gewünschter Endzustand (Lösung des Problems)**  
(es darf mehrere geben)
  - **Zielknoten: Übergang zu Nachbarzuständen**  
ergibt keine Verbesserung der bisher gefundenen optimalen Bewertung

# Suchen in Suchgraphen

## Verschiedene Suchziele:

- 1) Irgendeine Lösung eines Problems finden bzw. herausfinden, dass es keine gibt.
  - 2) Weitere Lösungen finden bzw. herausfinden, dass es keine weiteren mehr gibt.
  - 3) Alle Lösungen finden
  - 4) Optimale Lösung finden bzw. möglichst gute Lösung finden.
- Expansion eines Knotens: Berechnung aller Folge- bzw. Nachbarknoten

Verschiedene Suchstrategien unterscheiden sich in:

**Welcher Knoten wird als nächstes expandiert ?**

Spezialfall:

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

# Bsp. für Suchbäume in CSP

## Constraint-System:

- 1)  $(2 < x < 4)$
- 2)  $(0 < y < 6)$
- 3)  $(x + y > 7)$
- 4)  $(x \cdot y < 10,5)$

## Definitionsbereich für zulässige Lösungen:

$x, y \in \mathbf{Q}$ ,  
maximal k Stellen nach dem Komma

## Optimierungskriterium:

Minimiere  $|y - x|$

## Suchbaum:

- Jeder Knoten hat festen x- und y-Wert, Knoten können zulässig oder unzulässig sein, für jeden Knoten gibt es eindeutigen Wert für Optimierungsfunktion
- In Ebene i hat jeder x-Wert nur i Stellen nach dem Komma, der y-Wert ist nach Constraint 3) minimal dazu.

## Expansionsstrategien:

- Nur zulässige Knoten werden expandiert
- Es wird immer der rechteste zulässige Knoten expandiert
- ...

# Bsp. für Suchbäume in CSP

## Constraint-System:

- 1)  $(2 < x < 4)$
- 2)  $(0 < y < 6)$
- 3)  $(x + y > 7)$
- 4)  $(x \cdot y < 10,5)$

## Definitionsbereich für zulässige Lösungen:

$x, y \in \mathbf{Q}$ ,  
maximal  $k$  Stellen nach dem Komma

## Optimierungskriterium:

Minimiere  $|y - x|$

## Für festes $k$ :

- Suchraum endlich
- Mehrere zulässige Lösungen
- Immer genau 1 optimale Lösung

## Für $k$ unbeschränkt:

- Suchraum unendlich
- Unendlich viele zulässige Lösungen
- keine optimale Lösung

# Uninformierte Suchstrategien

Im allgemeinen ist nur *blinde (uninformierte) Suche* möglich:

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

Die wichtigsten Suchstrategien:

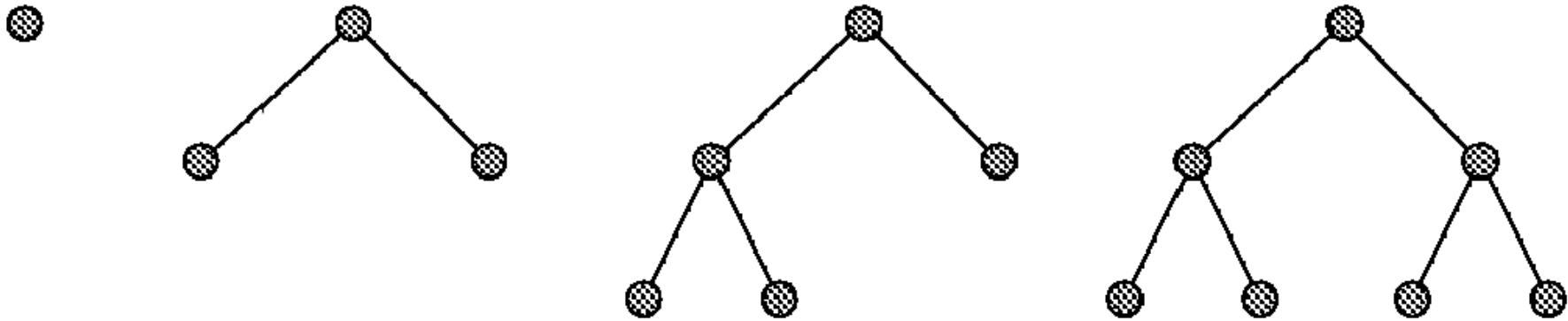
1. Breitensuche (breadth-first-search)
2. Tiefensuche (depth-first-search)
3. Bestensuche (best-first-search)

Weitere Infos zum Thema Suchen: Seminarvortrag und Ausarbeitung von Sven Schmidt, SS 2005, Nr. 4

<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>

# Uninformierte Suchstrategien

## Breitensuche (breadth-first-search):



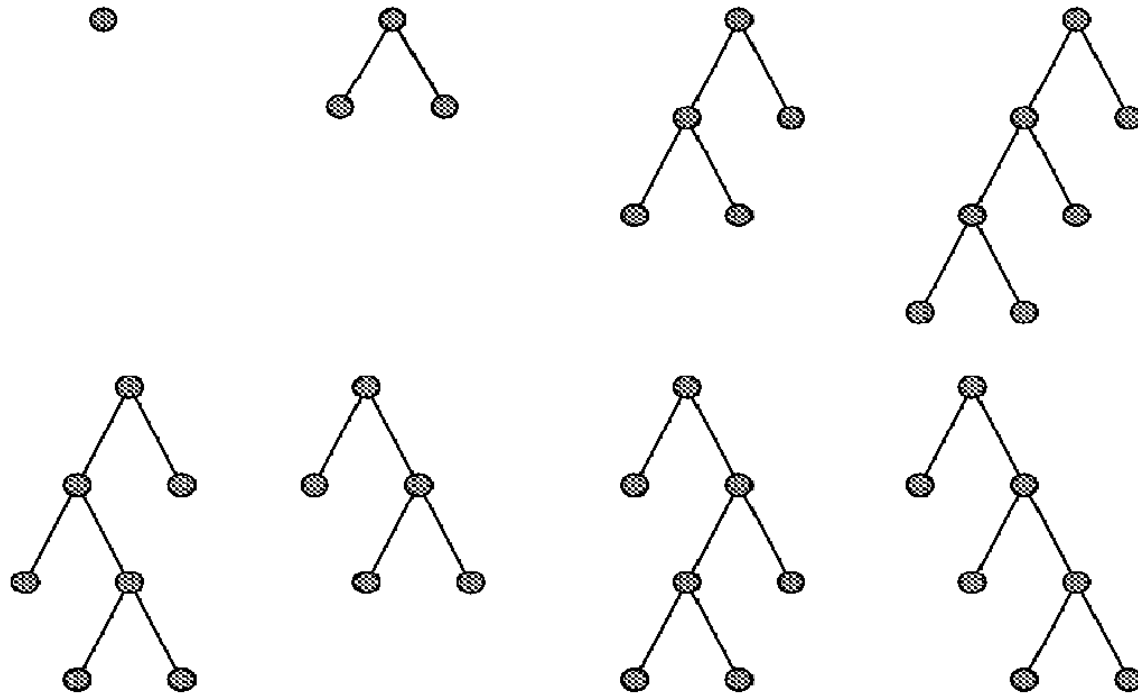
*Problemgröße: Tiefe des Suchbaums*

**Exponentieller** Aufwand für Zeit und Platz

**Für Suchprobleme in den meisten Fällen nicht relevant**

# Uninformierte Suchstrategien

## Tiefensuche (depth-first-search)



**Exponentieller** Aufwand für Zeit

*Problemgröße: Tiefe des Suchbaums*

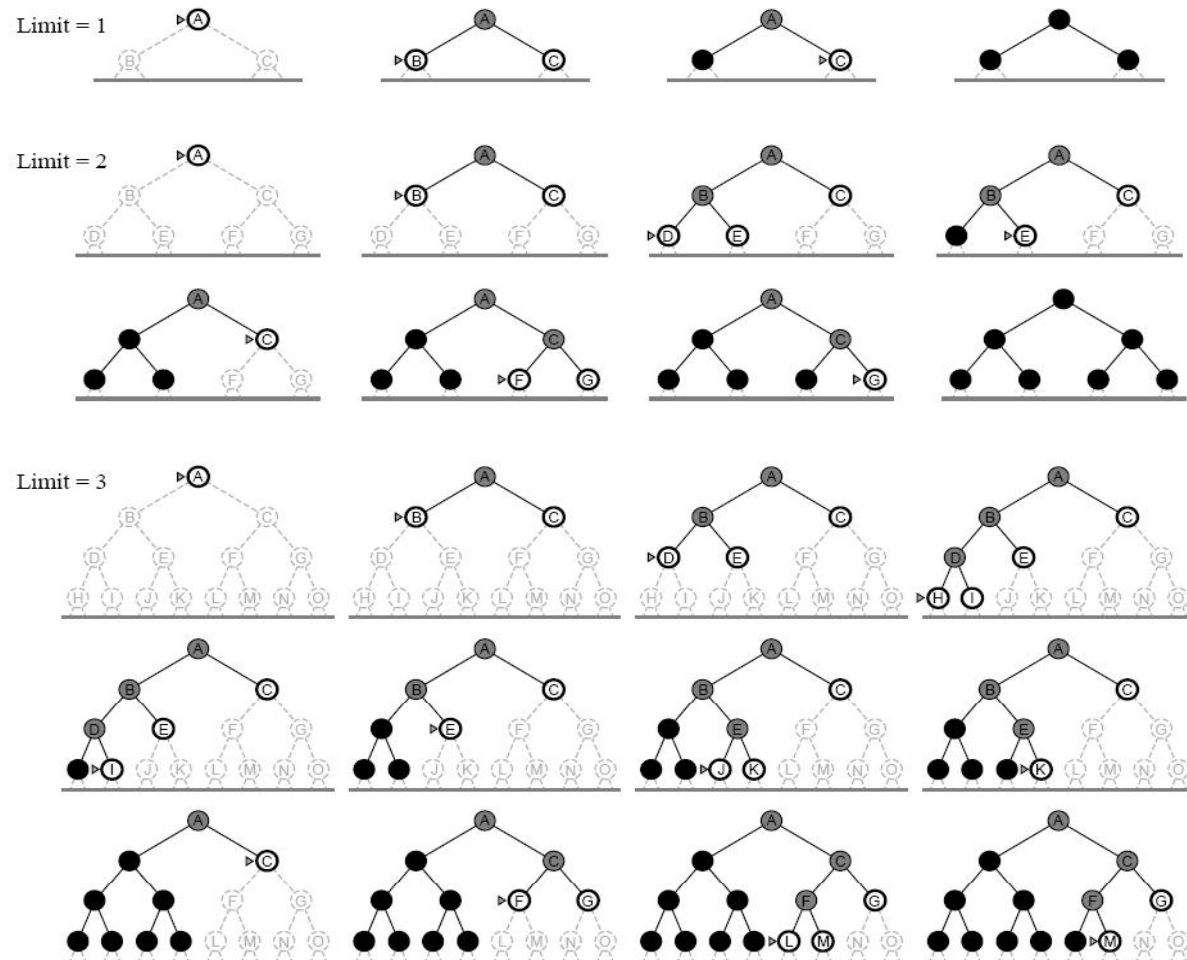
**Linearer** Aufwand für Platz

Der „Normalfall“ für allgemeine Suchprobleme

# Uninformierte Suchstrategien

## 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

## Bestensuche (best-first-search)

- zusätzlich sei gegeben: Bewertungsfunktion für die Zustände
- Suchziel: Finde die beste Lösung (und dann erst andere).
- 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

## Der Algorithmus von Dijkstra auf kantenbewerteten Graphen

(Spezialfall von Best-First-Search)

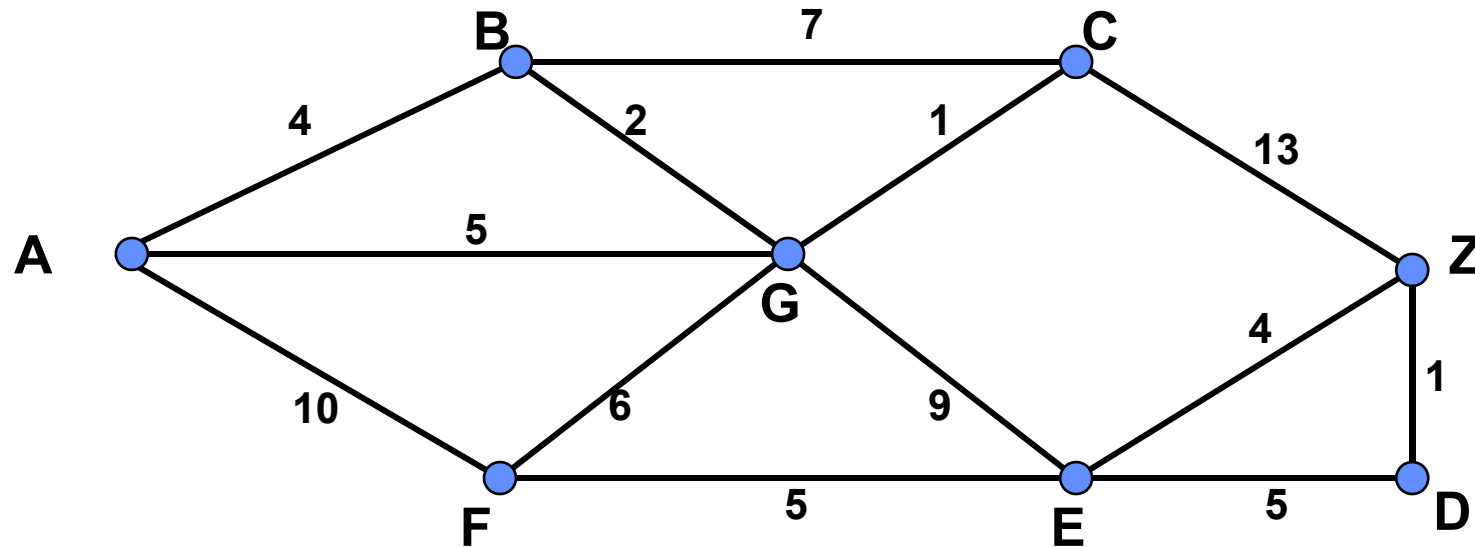
Für alle Kanten  $(u,v)$  gibt es Bewertungsfunktion:  
 $Länge(u,v) :=$  Länge der Kante von Ecke  $u$  nach Ecke  $v$

**Voraussetzung an Kantenbewertung:** Alle Kantenlängen müssen nichtnegativ sein.

### Algorithmus für Suche des Weges von A nach B mit minimaler Kantenlänge:

- 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  $Weglänge(N) := Länge(A,N)$  und alle anderen Ecken V mit  $Weglänge(V) := \infty$ .
- 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**:  
 $Weglänge(N) := \min \{Weglänge(N), Weglänge(V) + Länge(V,N)\}$ .bis  $V = B$

# Beispiel für Algorithmus von Dijkstra

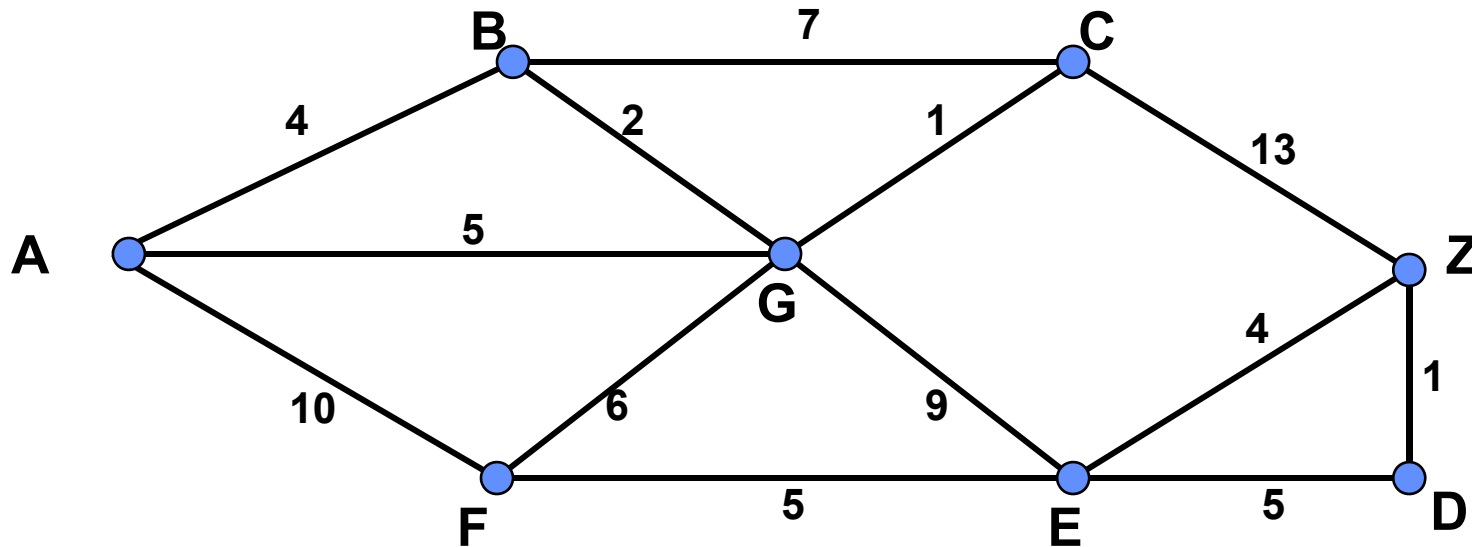


**Kürzester Weg von A nach Z:  $A \rightarrow F \rightarrow E \rightarrow Z$  (17 Einheiten)**

Animation dieser Aufgabe und weitere Infos zum Algorithmus von Dijkstra:  
Seminarvortrag und Ausarbeitung von Alex Prentki, WS 2004, Nr. 14

<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/WS2004/SeminarMC.html>

# Beispiel für Algorithmus von Dijkstra



**Kürzester Weg von G nach Z:  $G \rightarrow E \rightarrow Z$  (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( $\infty$ )	$\rightarrow$	D( $\infty$ )	$\rightarrow$	D( $\infty$ )	$\rightarrow$	D( $\infty$ )	$\rightarrow$ D(14,E)
E(9,G)		E(9,G)		E(9,G)		E(9,G)	
F(6,G)		F(6,G)		F(6,G)			
Z( $\infty$ )		Z(14,C)		Z(14,C)		Z(14,C)	Z(13,E)

# Informierte (Heuristische) Suchstrategien

## Gegebene Zusatzinformation:

**Schätzfunktion  $h(\text{Zustand})$**  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 Schätzfunktionwert aufweist
- Beim Aufsteigen im Suchbaum wird der jeweils nächstbeste Knoten expandiert.

**Hauptproblem: Warteschleifen in lokalen Maxima**

# Informierte (Heuristische) Suchstrategien

## Gegebene Zusatzinformation:

**Schätzfunktion  $h(\text{Zustand})$**  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 Schätzfunktionenwert aufweist
- Zurücksetzen ist nicht möglich: Wenn Schätzfunktion Fehler macht, wird nicht das optimale oder sogar gar kein Ergebnis gefunden.

**Hauptproblem: Feststecken in lokalen Maxima**

# Informierte (Heuristische) Suchstrategien

## Gegebene Zusatzinformation:

**Schätzfunktion  $h(\text{Zustand})$**  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/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>

# Informierte (Heuristische) Suchstrategien

## Der A\*-Algorithmus auf kantenbewerteten Graphen

(Verallgemeinerung des Algorithmus von Dijkstra) (*Zustandsbewertung = Knotenbewertung*)

**Voraussetzung an Kantenbewertung:** Alle Kantenlängen müssen nichtnegativ sein

**Voraussetzung an Heuristik  $h_B(u)$**  für die Abschätzung bzgl. Weglänge  $w_B(u)$  zum Zielknoten B:

**Zulässigkeitsbedingung:**  $h_B(u) \leq w_B(u)$

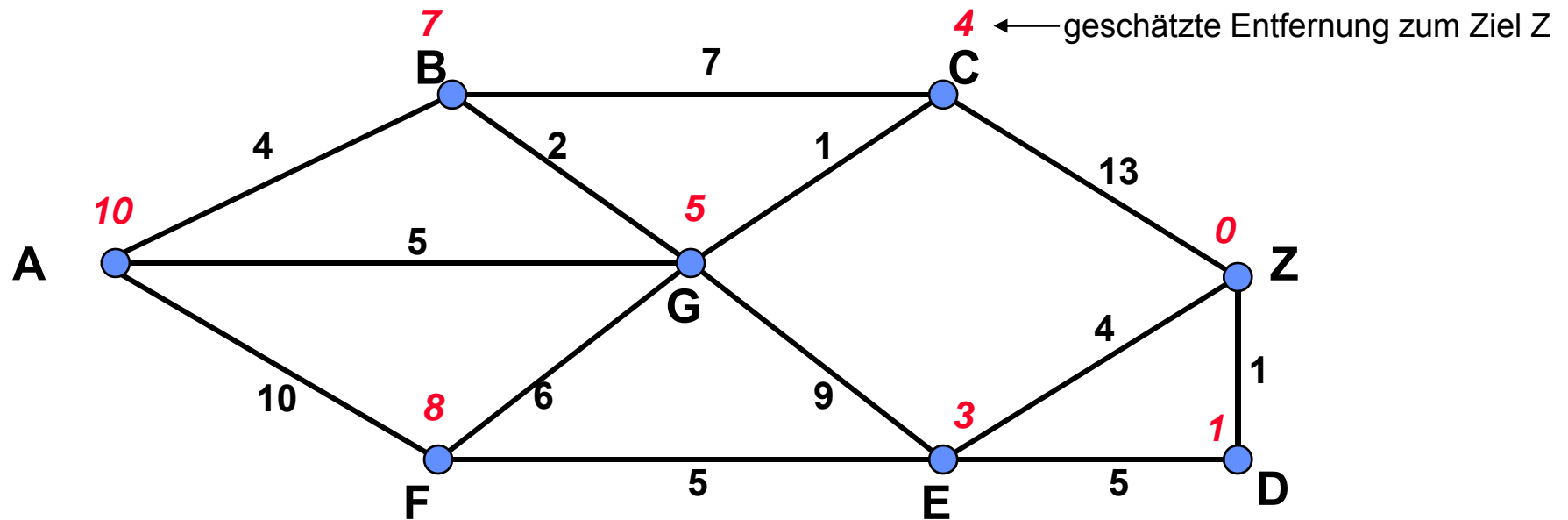
**Monotoniebedingung:**  $h_B(u) \leq h_B(v) + \text{Länge}(u,v)$

### Algorithmus für Suche des Weges von A nach B mit minimaler Kantenlänge:

- 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* (A,N)  
*Schätzung* (N) := *Weglänge* (N) +  $h_B(N)$   
und alle anderen Ecken V mit *Weglänge* (V) :=  $\infty$  und *Schätzung* (V) :=  $\infty$ .
  - 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**:  
*Weglänge* (N) :=  $\min \{ \text{Weglänge} (N), \text{Weglänge} (V) + \text{Länge} (V,N) \}$ .  
*Schätzung* (N) := *Weglänge* (N) +  $h_B(N)$  (falls Aktualisierung notwendig).
- bis  $V = B$



# Beispiel für A\*-Algorithmus



**Kürzester Weg von G nach Z:  $G \rightarrow E \rightarrow Z$  (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)		<span style="border: 1px solid red; padding: 2px;">B(2,G,9)</span>				
<span style="border: 1px solid red; padding: 2px;">C(1,G,5)</span>						
D( $\infty$ )	$\rightarrow$	D( $\infty$ )	$\rightarrow$	D( $\infty$ )	$\rightarrow$	D(14,E,15)
E(9,G,12)		E(9,G,12)		<span style="border: 1px solid red; padding: 2px;">E(9,G,12)</span>		
F(6,G,13)		F(6,G,14)		F(6,G,14)		F(6,G,14)
Z( $\infty$ )		Z(14,C,14)		Z(14,C,14)		<span style="border: 1px solid red; padding: 2px;">Z(13,E,13)</span>

# Informierte (Heuristische) Suchstrategien

## Der A\*-Algorithmus auf kantenbewerteten Graphen

(Verallgemeinerung des Algorithmus von Dijkstra) (Zustandsbewertung = Knotenbewertung)

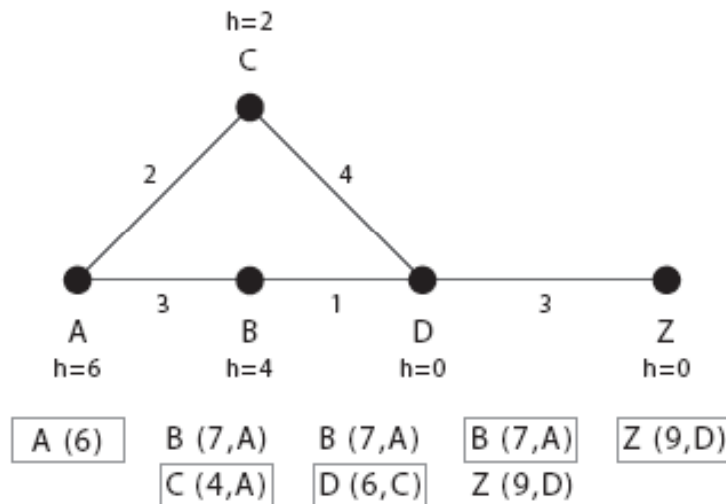
Voraussetzung an Kantenbewertung: Alle Kantenlängen müssen nichtnegativ sein

Voraussetzung an Heuristik  $h_B(u)$  für die Abschätzung bzgl. Weglänge  $w_B(u)$  zum Zielknoten B:

Zulässigkeitsbedingung:  $h_B(u) \leq w_B(u)$

**Was passiert bei Wegfall der Monotoniebedingung:**  $h_B(u) \leq h_B(v) + \text{Länge}(u,v)$

**Beispiel:**



Aus: Diplomarbeit Andre Keller (SS 2008)

**Fehler:** D wird nicht mehr aktualisiert, weil es schon in **Berechnet** ist.

# Informierte (Heuristische) Suchstrategien

## Der A\*-Algorithmus auf kantenbewerteten Graphen

(Verallgemeinerung des Algorithmus von Dijkstra) (Zustandsbewertung = Knotenbewertung)

**Voraussetzung an Kantenbewertung:** Alle Kantenlängen müssen nichtnegativ sein

**Voraussetzung an Heuristik**  $h_B(u)$  für die Abschätzung bzgl. Weglänge  $w_B(u)$  zum Zielknoten B:

**Nur Zulässigkeitsbedingung:**  $h_B(u) \leq w_B(u)$

### Algorithmus für Suche des Weges von A nach B mit minimaler Kantenlänge:

- 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  $Weglänge(N) := Länge(A, N)$   
 $Schätzung(N) := Weglänge(N) + h_B(N)$  und alle anderen Ecken V mit  $Weglänge(V) := \infty$  und  $Schätzung(V) := \infty$ .
- 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 **Berechnet** und **Unberechnet**:  
 $Weglänge(N) := \min \{Weglänge(N), Weglänge(V) + Länge(V, N)\}$ .  
 $Schätzung(N) := Weglänge(N) + h_B(N)$  (falls Aktualisierung notwendig).
  - Falls Aktualisierung bei Nachbarn N aus **Berechnet** erfolgte:  
Verschiebe N wieder nach **Unberechnet**.

bis  $V = B$

# Allgemeine Optimierungsverfahren für CSP

## Für Suche einer Gesamtlösung über Teillösungen:

### 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

## Fürs systematische Verbessern von Gesamtlösungen:

### 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, ...

Weitere Details zum Thema Constraintsysteme:

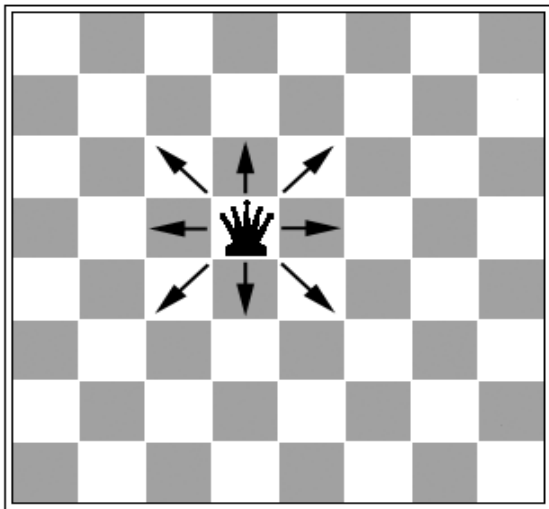
Seminarvortrag und Ausarbeitung von Stefan Schmidt, SS 2005, Nr. 6,

<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>

# Allgemeine Optimierungsverfahren für CSP

## Verfahren der minimalen Konflikte (Min-Conflicts)

### Anwendungsbeispiel: 8-Damen-Problem



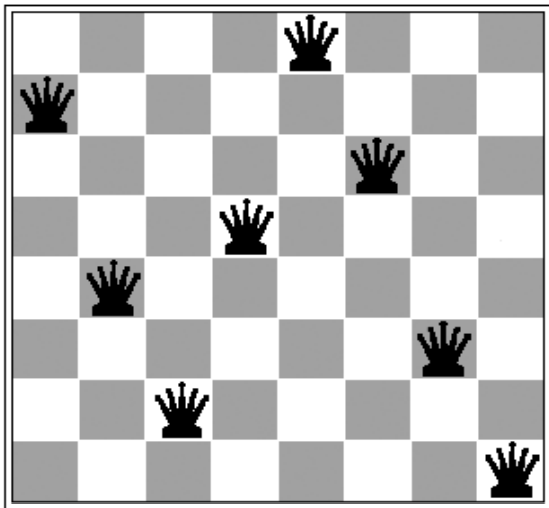
Quelle: Seminarvortrag von Stefan Schmidt, SS 2005, Nr. 6,

<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>

# Allgemeine Optimierungsverfahren für CSP

## Verfahren der minimalen Konflikte (Min-Conflicts)

### Anwendungsbeispiel: 8-Damen-Problem

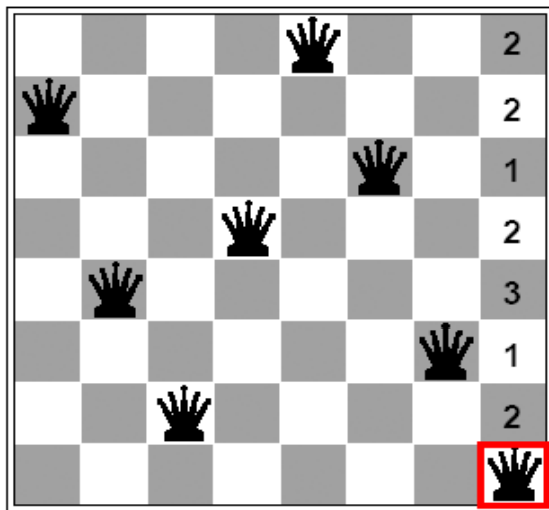


Quelle: Seminarvortrag von Stefan Schmidt, SS 2005, Nr. 6,  
<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>

# Allgemeine Optimierungsverfahren für CSP

## Verfahren der minimalen Konflikte (Min-Conflicts)

### Anwendungsbeispiel: 8-Damen-Problem



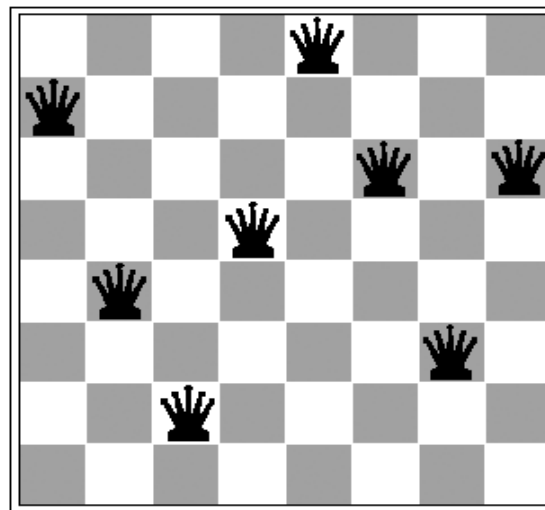
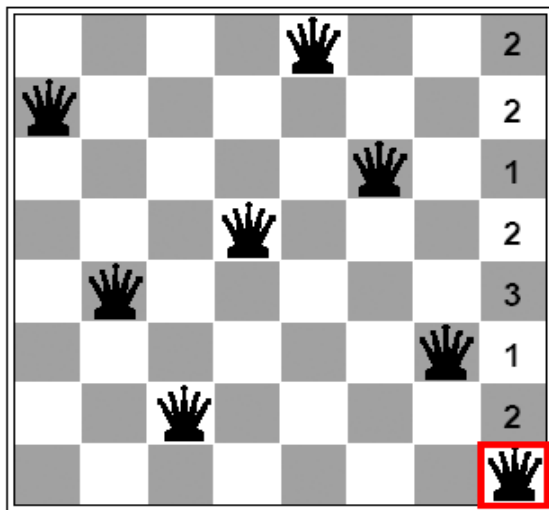
Quelle: Seminarvortrag von Stefan Schmidt, SS 2005, Nr. 6,  
<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>



# Allgemeine Optimierungsverfahren für CSP

## Verfahren der minimalen Konflikte (Min-Conflicts)

### Anwendungsbeispiel: 8-Damen-Problem

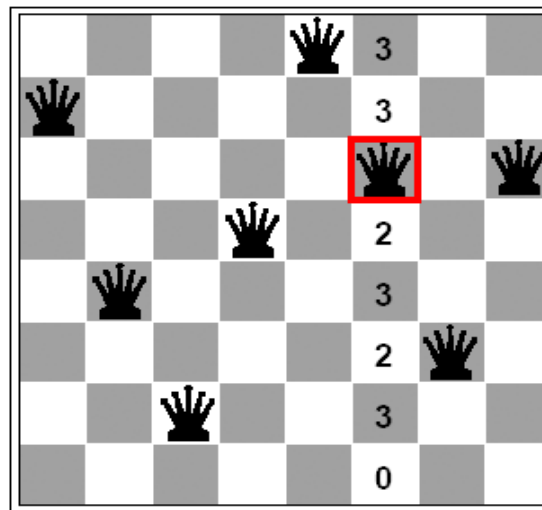
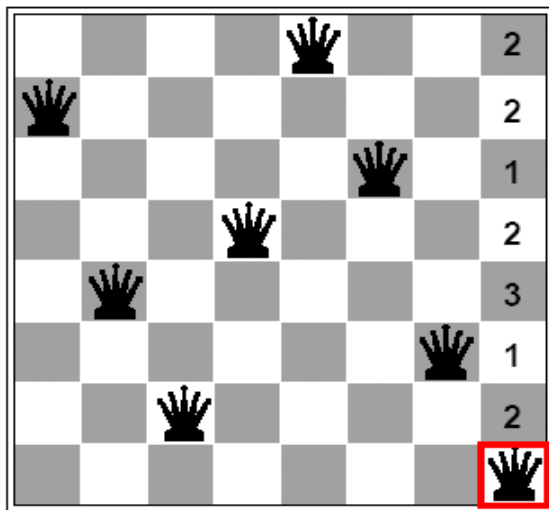


Quelle: Seminarvortrag von Stefan Schmidt, SS 2005, Nr. 6,  
<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>

# Allgemeine Optimierungsverfahren für CSP

## Verfahren der minimalen Konflikte (Min-Conflicts)

### Anwendungsbeispiel: 8-Damen-Problem

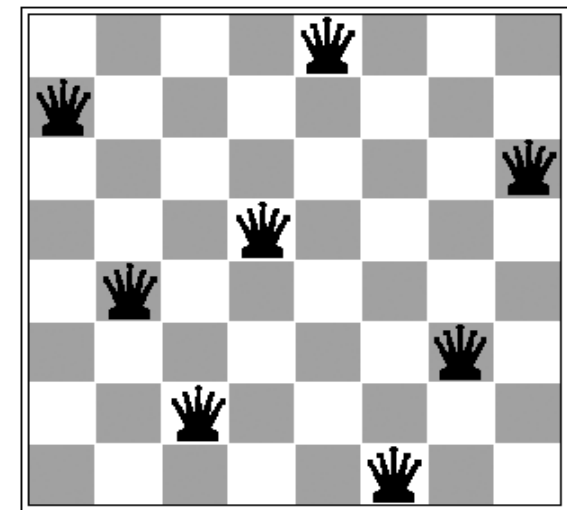
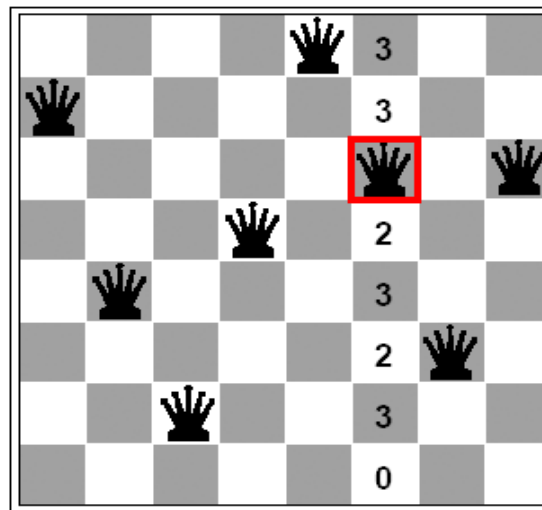
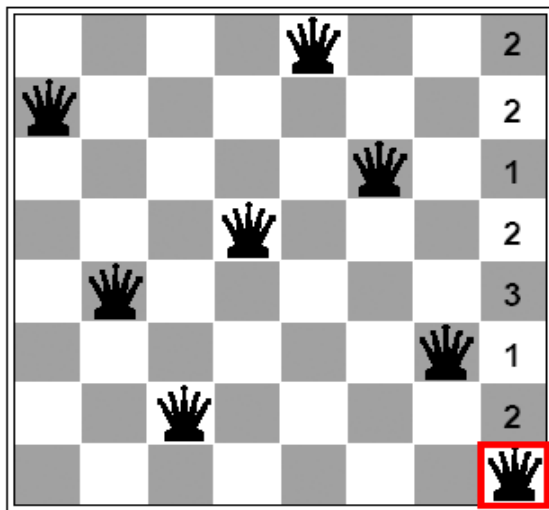


Quelle: Seminarvortrag von Stefan Schmidt, SS 2005, Nr. 6,  
<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>

# Allgemeine Optimierungsverfahren für CSP

## Verfahren der minimalen Konflikte (Min-Conflicts)

### Anwendungsbeispiel: 8-Damen-Problem



Quelle: Seminarvortrag von Stefan Schmidt, SS 2005, Nr. 6,  
<http://www.fh-wedel.de/archiv/iw/Lehrveranstaltungen/SS2005/SeminarKI.html>

# Allgemeine Optimierungsverfahren für CSP

## Arbeiten mit Tabu-Listen in Suchgraphen:

- Lege ein bestimmtes Gültigkeitsintervall für die Arbeit des Algorithmus fest, z.B. durch eine vorgegebene Anzahl von Operationen
- Protokolliere alle benutzten Kanten beim Übergang zwischen Zuständen
- Alle im vorangeegangenen Gültigkeitsintervall benutzten Kanten dürfen nicht noch einmal benutzt werden, ebenso nicht deren Umkehrungen.

### **Wird hauptsächlich benutzt beim systematischen Verbessern von Gesamtlösungen**

- Gute Ergebnisse in der Logistik (TSP-Verallgemeinerungen)