Introduction to Robotics

Search and Motion Planning

WS 2015 / 16

Prof. Dr. Daniel Göhring
Intelligent Systems and Robotics
Department of Computer Science
Freie Universität Berlin

State-Space Search

Graph $[Z, Op]$

- States Z (Vertices)
- Operators $Op \subseteq Z \times Z$ (Edges)
- Initial states $z_{initial} \in Z$
- Target states $Z_{final} \subseteq Z$

- Cost functions $c: Z \times Z \rightarrow R^+$

  Costs of a path $w = z_0\, z_1 \ldots z_n \in Z^*$:

  $c(z_0\, z_1 \ldots z_n) := \Sigma_{i=1,\ldots,n}\, c(z_{i-1}, z_i)$
- Estimator function $\sigma: Z \rightarrow R$ (Heuristic) for remaining costs from z to a target state.

Slides: H.-D. Burkhard

State-Space Search

- Tasks:

- Can a goal state $z \in Z_{final}$ be reached from initial state $z_{initial} \in Z$?

- Find a way from initial state $z_{initial} \in Z$ to a target state $z \in Z_{final}$

- Find an optimal path from initial state $z_{initial} \in Z$ to a target state $z \in Z_{final}$

Complexity (Number of States / Vertices)

- 8 Puzzle: 9! states

  9!/2 = 181.440 reachable
- 15 Puzzle: 16! states
- 16!/2 reachable
- Rubik‘s cube: $12 \cdot 4{,}3 \cdot 10^{19}$ states
- 1/12 reachable: $4{,}3 \cdot 10^{19}$
- Towers of Hanoi: 3^n States for n discs

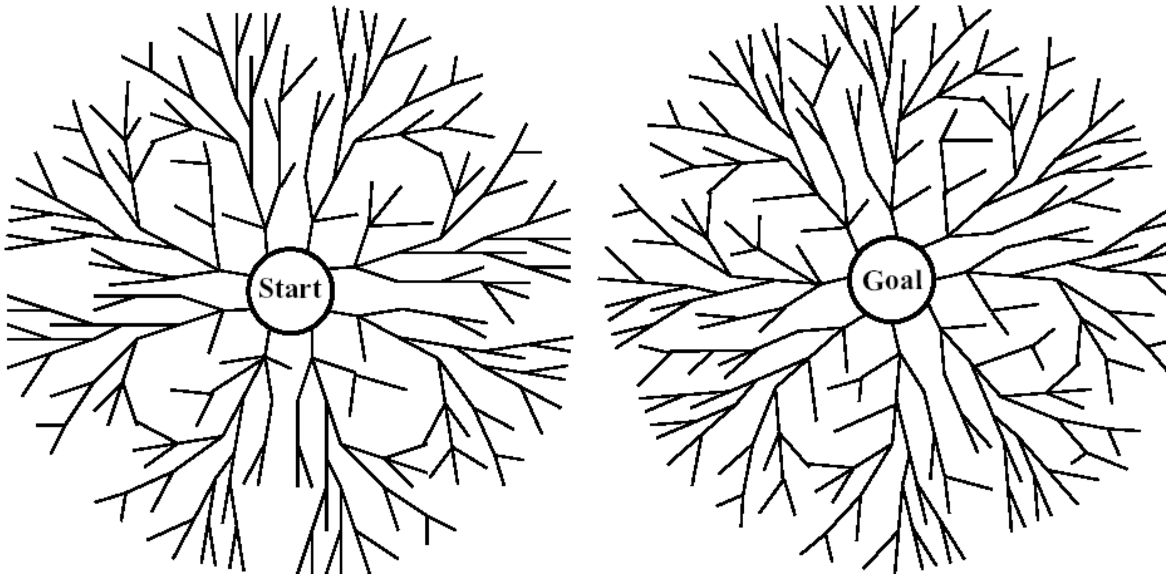
  solvable in $(2^n) - 1$ moves
- Checkers: approx. 10^{40} games of average length
- Chess: approx. 10^{120} games of average length
- Go: 3^{361} states

Expansion Strategies

- **Directions**
  - Forward, start with $z_{initial}$
    (forward chaining, data driven, bottom up)
  - Reverse Z_{final}
    (backward chaining, goal driven, top down)
  - Bi-directional
- **Expansion**
  - Depth first
  - Breadth first
- **Additional information**
  - blind search („uninformed“)
  - heuristisc search with σ („informed“)

Bi-Direktional Breadth-Search

- Parallel search from start and goal until meeting



- Search depth from both sides only half

Freie Universität Berlin

Expansion

Data structures :

•OPEN List:

A vertex is "open" , if it was constructed but not expanded (neighboring vertices not calculated)

•CLOSED List:

A vertex is „closed" , if it was fully expanded (all neighboring vertices are known)

Further information:

Predecessor / successor of vertices for reconstruction of found paths

Heuristic Search for best Way: A*

Costs to reach z' from z :

– If z' is reachable from z :

$$g(z,z') := \mathrm{Min}\{\, c(s) \,/\, s \text{ path from } z \text{ to } z' \,\},$$

– Else: $g(z,z') := \infty$

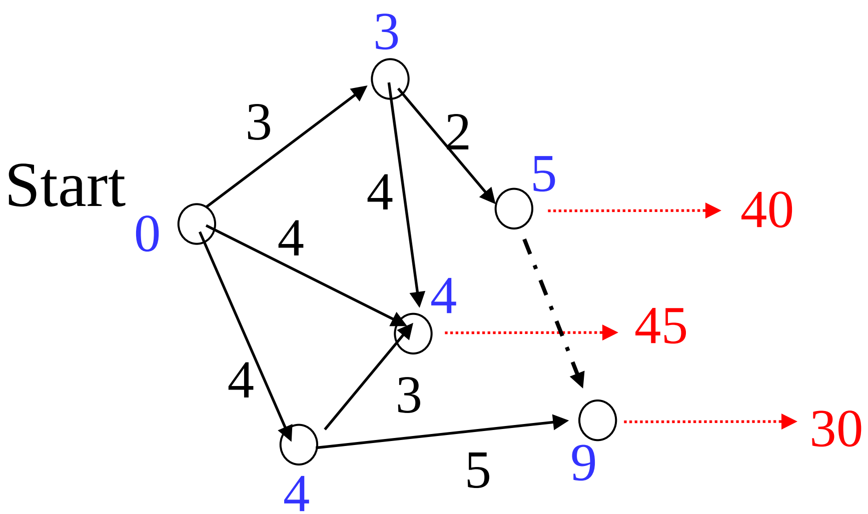
Tentative cost calculation during expansion:

$G' = [\, V', E' \,]$ is a (known) partial graph of G

$$g'(z, z', G') := \mathrm{Min}\ \{\, c(s) \,/\, s \text{ path in } G' \text{ from } z \text{ to } z' \,\}$$

$$\boxed{g'(z,z',G') \geq g(z,z')}$$

Heuristic Search for Best Path

- 

$g'(z_0, z', G')$: so far known costs to reach z´ from Start

$\sigma(z')$: estimated costs to reach target state, starting from z´

Algorithm A* („soft form“) For Trees

- A*0: (Start) OPEN := [z_0] , CLOSED := [ ] .
- A*1: (negative exit)
  If OPEN = [ ] : EXIT(„no“).
- A*2: (positive exit)
  If z first vertice in OPEN:
  If z is target: EXIT(„yes:“ z).
- A*3: (expand)
  OPEN := OPEN - {z} . CLOSED := CLOSED $\cup${z} .
  Succ(z):= set of successors of z.
  If Succ(z)= {} : Goto A*1.
- A*4: (Organization of OPEN)
  – OPEN := OPEN $\cup$ Succ(z) with increasing order of $g'(z_0, z', G') + \sigma(z')$
- Goto A*1.

Search space as a tree: CLOSED not used.

Algorithm A* („soft form“) For Trees and for Cyclic Graphs

- A*0: (Start) OPEN := [z_0] , CLOSED := [ ] .
- A*1: (negative exit)
  If OPEN = [ ] : EXIT(„no“).
- A*2: (positive exit)
  If z first vertice in OPEN:
  If z is target: EXIT(„yes:“ z).
- A*3: (expand)
  OPEN := OPEN - {z} . CLOSED := CLOSED ∪{z} .
  Succ(z):= set of successors of z.
  If Succ(z)= {} : Goto A*1.
  NEW = Succ(z) – {z´ | z´ ϵ Succ(z) and z´ ϵ CLOSED and $g´(z_0, z´,G´) >= g´(z_0, z´,G´_{old})$ }
- A*4: (Organization of OPEN)
  - OPEN := OPEN ∪ NEW with increasing order of $g´(z_0, z´,G´) + \sigma(z´)$
  - If elements z´ occur twice in OPEN, delete entry with bigger costs.
- Goto A*1.

If search space has cycles: use CLOSED.

Freie Universität Berlin

Algorithm A* („soft form“)

Definition: $f(z) := \text{Min}\ \{\ g(z,z_{final})\ |\ z_{final} \in Z_{final}\ \}$

= real costs from z to target state (vertex)

($f(z_0)$ = cost of the optimal path)

Heuristic function σ is called optimistic (a.k.a. admissible) or underestimating,

if $\sigma(z) \leq f\,(z)$ for all $z \in Z$.

Proposition :

Given.: Ex. $\delta > 0$ with $c(z,z') > \delta$ for all z,z' .

σ is optimistic heuristic.

every node has a finite number of successors.

One can prove: If solution exists, A* (soft form) finds an optimal path.

Special Cases

$c \equiv 0$:

Search for best path with heuristic σ and no cost function (Hill climbing)

$\sigma \equiv 0$:

Search for best path without heuristic

($\sigma \equiv 0$ ist also optimistic heuristic) - Dijkstra

$c \equiv 1$ ($g' \equiv$ Search depth) , $\sigma \equiv 0$:

Breadth first search

Influence of Heuristic σ

- Trade-off between quality of result and calculation efford

- Criteria for order of extension
- Same order as $g' + \sigma$ is also provided by

  $a \cdot (g' + \sigma) + b$ for arbitrary positive constants a, b .

- optimal order for $\sigma = f$
- σ_2 *more effective for* σ_1 if $\sigma_1 \leq \sigma_2 \leq f$

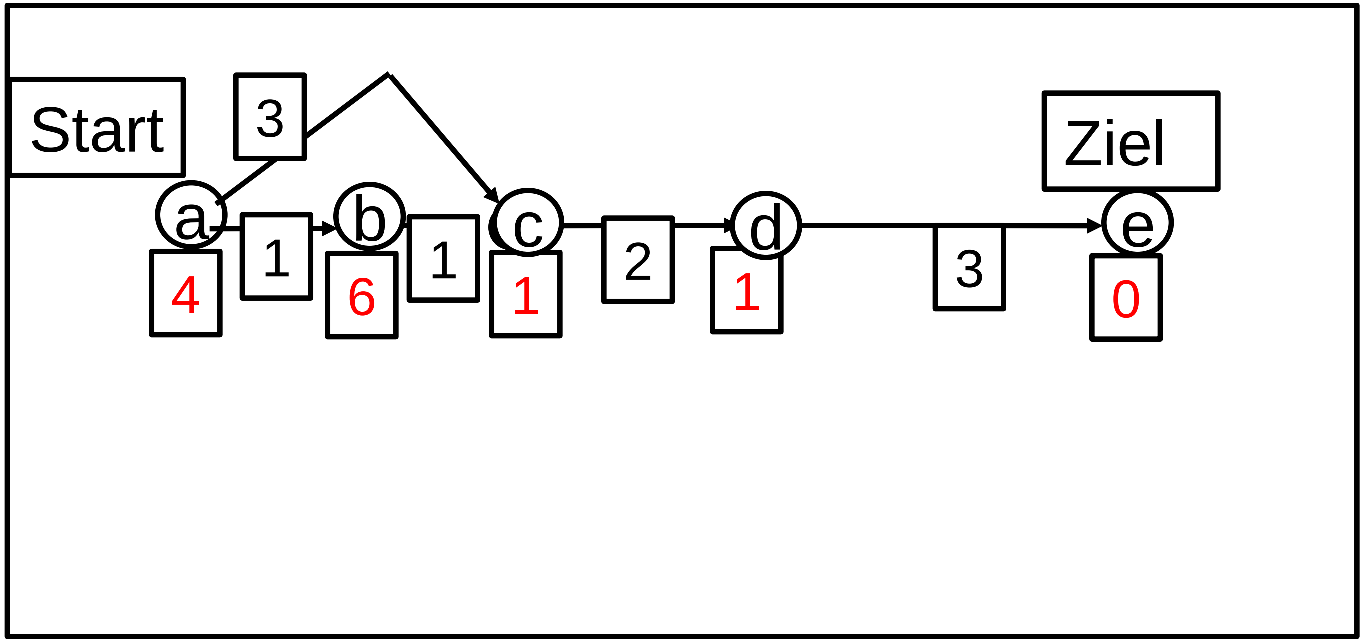
  (Hierarchy for heuristic functions)

Algorithm A* („hard form“)

- A*0: (Start) OPEN := $[z_0]$, CLOSED := [ ] .
- A*1: (negative exit)
  Falls OPEN = [ ] : EXIT(„no“).
- A*2: (positive exit)
  If z first state in OPEN.
    If z target state: EXIT(„yes:“ z).
- A*3: (expand)
  OPEN := OPEN - {z} . CLOSED := CLOSED $\cup$ {z} .
  Succ(z):= set of successors of z.
  If Succ(z)= {} : Goto A*1.
- A*4: (Organize of OPEN)
  OPEN := OPEN $\cup$ (Succ(z) - CLOSED) sorted by increasing $g'(z_0, z', G') + \sigma(z')$
- Goto A*1.

Algorithmus A* („hard form“)

- Problem:
- For optimistic σ the hard form is not always correct



Freie Universität Berlin

Algorithm A* („hard form“)

Definition:

Heuristic σ is called *consistent*,

if for all states z', z'' holds:

$$\sigma(z') \leq g(z', z'') + \sigma(z'')$$

Lemma: If σ is consistent, σ is optimistic.

(The reverse does not necessarily apply)

Algorithm A* („hard form“)

Proposition :

Given: Ex. $\delta > 0$ with $c(z,z') > \delta$ for all z,z' .

σ is a consitent heuristic

One can prove: if a solution exists, A* (hard form) finds an optimal path

Algorithm A* („hard form“)

- „consistent“ functions harder to find then „optimistic“
- It is also possible to use an optimistic heuristic with weaker pruning of search space
- Erase states in OPEN (or Succ(z)) only if new evaluation is worse than earlier evaluation.

Algorithm A*

- Search costs ~ Number of states to expand,
- ~ Calculational effort of σ and g'

- Space requirements exponential

- Search costs vs. costs of solution (optimal / suboptimal solutions)
- Measure: Path length/expanded vertices

Memory Saving Variants of Algorithm A*

- Iterative Deepening A* (IDA*) in analogy to IDA:
  - Depth first till boundary $g'(z_0, z', G') + \sigma(z')$ is exceeded from earlier iteration

- SMA* (simplified memory-bounded A*)

Anytime (or Weighted) A*

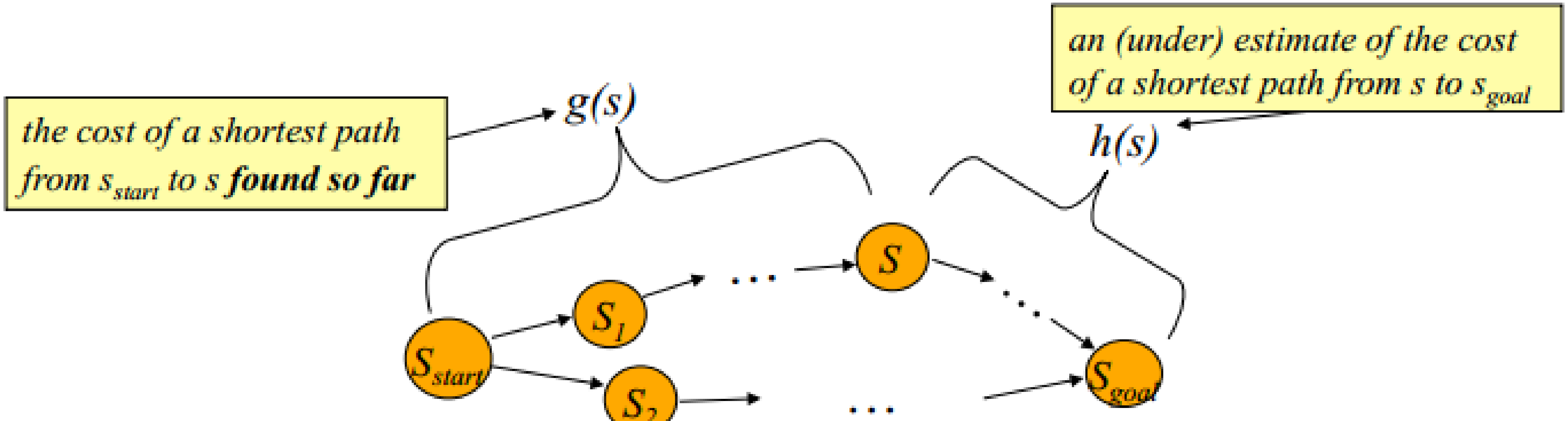
- If the heuristic σ (sometimes also denoted h) is closer to the real costs, less vertices have to be expanded

- But „inflating“ the heuristics can lead to a heuristic which is not optimistic (admissible),

(the following slides with courtesy to Likhachev)

Heuristics in Heuristic Search

- Dijkstra's: expands states in the order of $f = g$ values
- A* Search: expands states in the order of $f = g+h$ values
- **Weighted A*:** expands states in the order of $f = g+\varepsilon h$ values, $\varepsilon > 1$ = bias towards states that are closer to goal



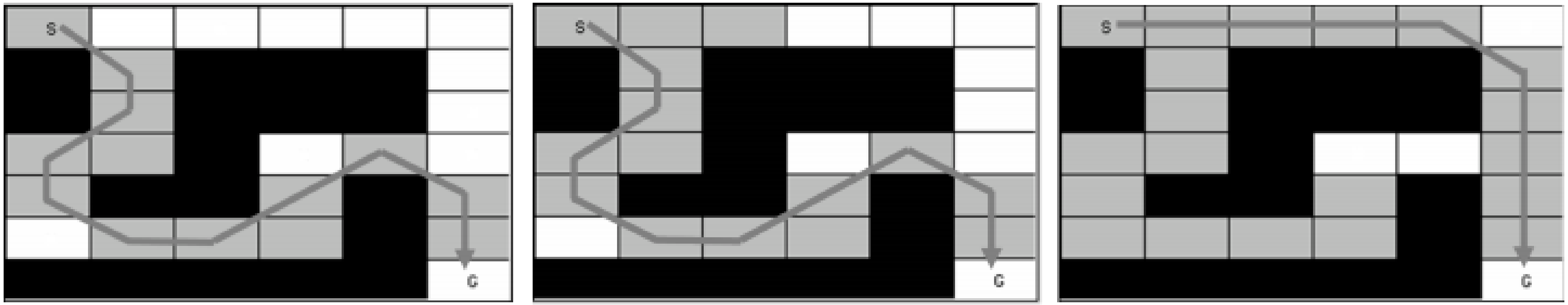
Suboptimality

- Heuristic includes factor ε
- Suboptimality is bounded by factor ε
  - The length of the found solution is not longer than ϵ times the optimal solution
- Exampe:
  - costs from cell to cell are 1
  - Heuristic is the larger of coordinate difference from current cell to goal cell
  - Start is upper left cell
  - Goal is lower right
  - Obstacles black
  - Free space white
  - Expanded cells gray

Anytime Search based on weighted A*

- Constructing anytime search based on weighted A*:
  - find the best path possible given some amount of time for planning
  - do it by running a series of weighted A* searches with decreasing ε:

$\varepsilon = 2.5$ | $\varepsilon = 1.5$ | $\varepsilon = 1.0$



13 expansions
solution=11 moves

15 expansions
solution=11 moves

20 expansions
solution=10 moves

Anytime A*

- Problem:
  - Running A* with increasing ϵ each time from scratch can be very expensive
  - Many states remain the same through various iterations
  - A solution for reusing the search results is described in ARA* (Likhachev)

- ARA* : an efficient version of the above that reuses state values within any search iteration

ARA*

- Efficient series of weighted A* searches with decreasing ε:

ComputePathwithReuse function

while($f(s_{goal})$ > minimum f-value in OPEN)

remove s with the smallest $[g(s)+ \varepsilon h(s)]$ from OPEN;

insert s into CLOSED;

for every successor s' of s

if $g(s') > g(s) + c(s,s')$

$g(s') = g(s) + c(s,s')$;

if s' not in CLOSED then insert s' into OPEN;

otherwise insert s' into INCONS

ARA*

- Efficient series of weighted A* searches with decreasing ε:

ComputePathwithReuse function

while($f(s_{goal})$ > minimum f-value in OPEN)
remove s with the smallest $[g(s)+ \varepsilon h(s)]$ from OPEN;
insert s into CLOSED;
for every successor s' of s
if $g(s') > g(s) + c(s,s')$
$g(s') = g(s) + c(s,s')$;
if s' not in CLOSED then insert s' into OPEN;
otherwise insert s' into INCONS

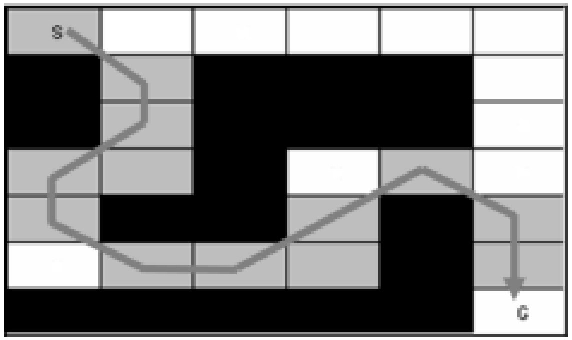
---

set ε to large value;
$g(s_{start}) = 0$; OPEN = $\{s_{start}\}$;
while $\varepsilon \geq 1$
CLOSED = {}; INCONS = {};
ComputePathwithReuse();
publish current ε suboptimal solution;
decrease ε;
initialize OPEN = OPEN U INCONS;

ARA*

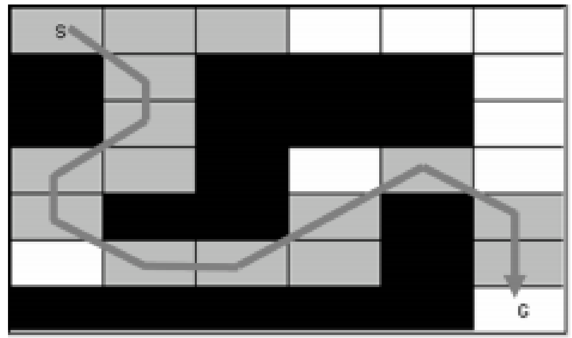
- **A series of weighted A* searches**

ε =2.5



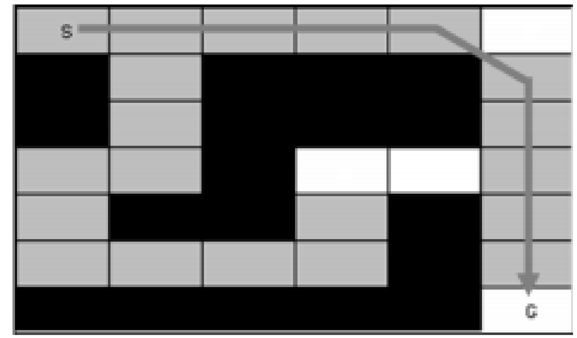
13 expansions
solution=11 moves

ε =1.5



15 expansions
solution=11 moves

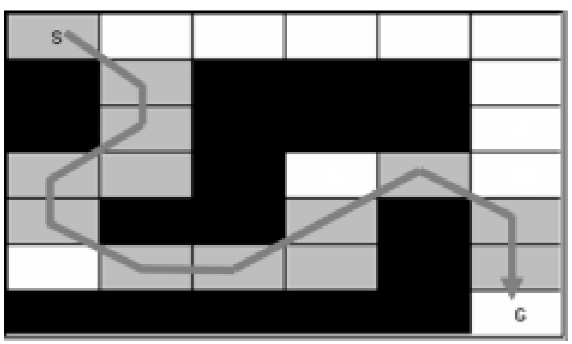
ε =1.0



20 expansions
solution=10 moves

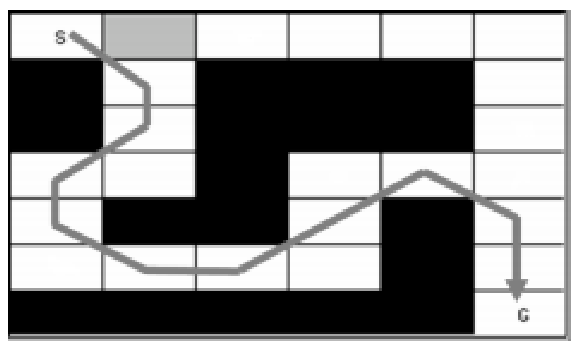
- **ARA***

ε =2.5



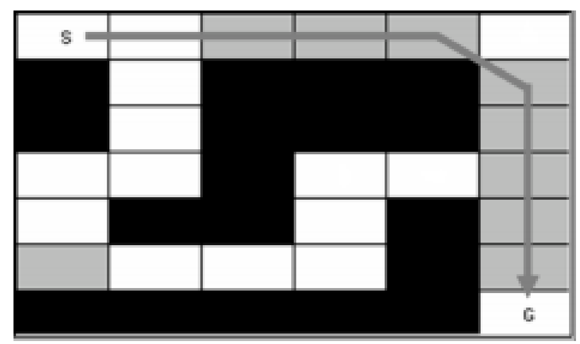
13 expansions
solution=11 moves

ε =1.5



1 expansion
solution=11 moves

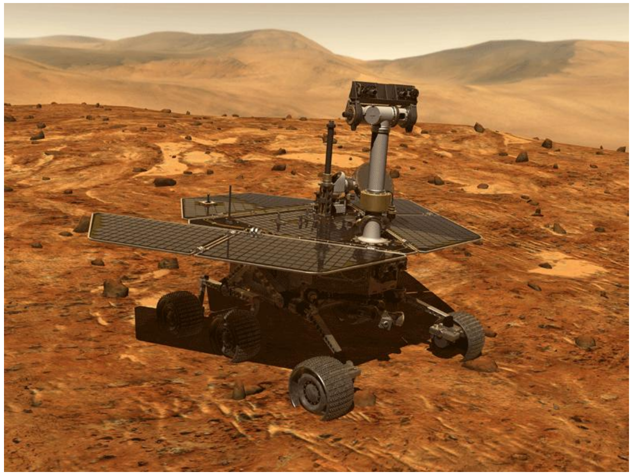
ε =1.0



9 expansions
solution=10 moves

D* and Variations

- D* and its variations have been used on Mars rovers Opportunity and Spirit and by CMU at the DARPA Grand Challenge



- Roughly:
  - D* works as A* but from goal to target
  - Every expanded node „knows“ its predecessor
  - When start node (vertex) is the next node, the search is done
  - modes are marked: NEW (was never in OPEN), OPEN, CLOSE (no longer in OPEN), LOWER, RAISE (cost is higher than last time in OPEN)

Freie Universität Berlin

Literature

- J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.
- Hans-Dieter Burkhard: Einführung in die Künstliche Intelligenz (2008)
- Likhachev, ARA*, CMU
- Choset, Motion Planning, CMU,
- Toussaint, Lecture Notes Robotics, 2011
- Dr. John (Jizhong) Xiao, City College New York