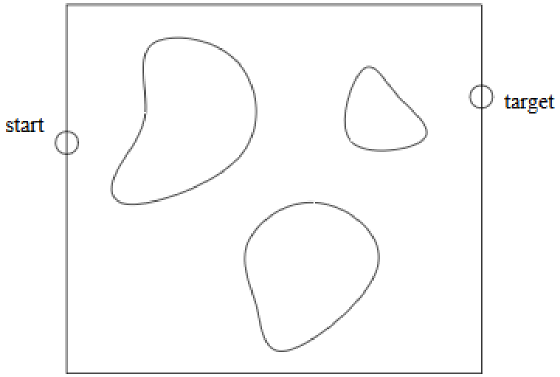
geometric A*



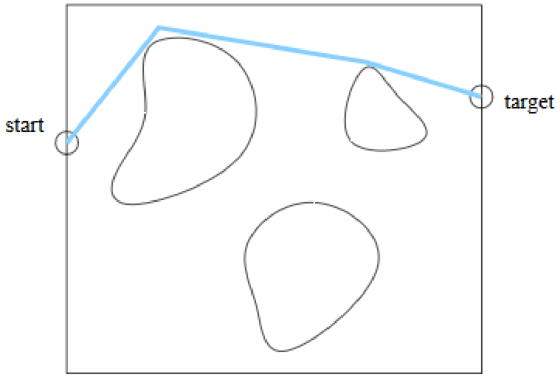
the problem: go from start to end

region of the plane

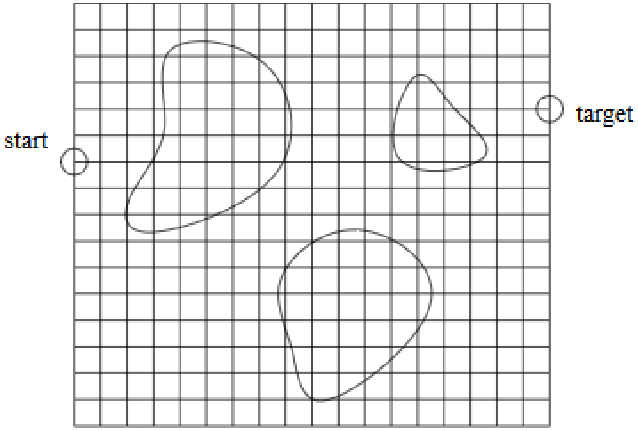
minimal path

applications:

- autonomous vehicles
- non-player characters in video games

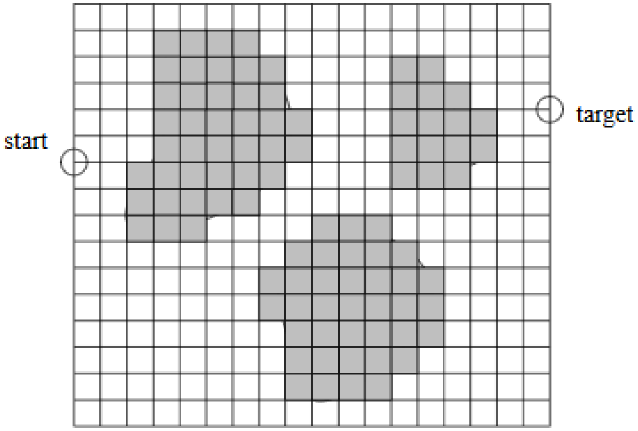


actual minimal path hard to find



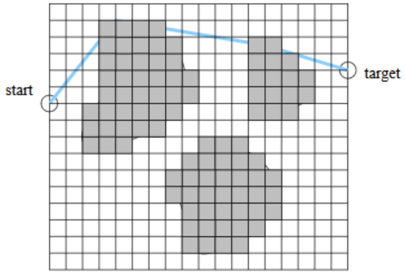
approximation: grid

only moving on the grid + diagonals



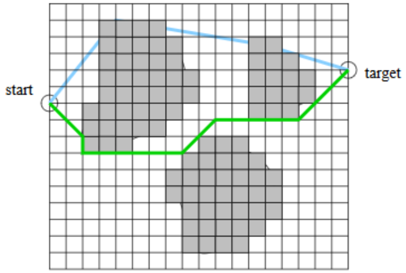
occluded areas on the grid

forbidden: diagonals inside a black box
horizontal or vertical line surronded by two black boxes



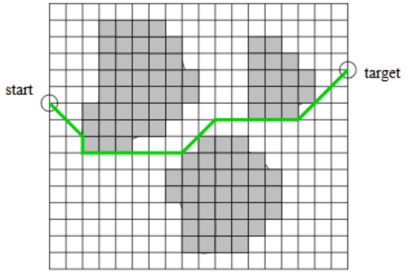
real optimal paths may become impossible as is

meaning: with the grid simplifications some optimal paths become impossible
the simplification actually may make paths longer



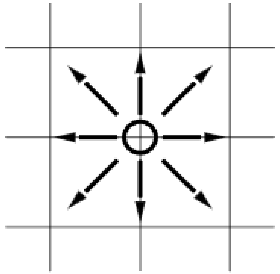
an optimal path in the grid

feasible also in the original problem
but not optimal



finding optimal paths: use A*

a state for each point in the grid
allowed moves



allowed moves

is a restriction

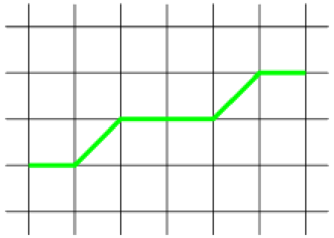
path not really optimal

---

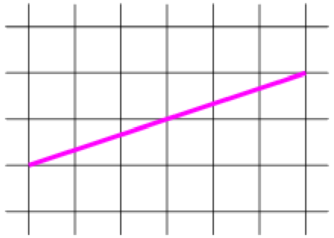
suboptimality:

- the occluded areas may not be as in the grid
- the grid constraints the movements

increasing resolution attenuates the first
not the second

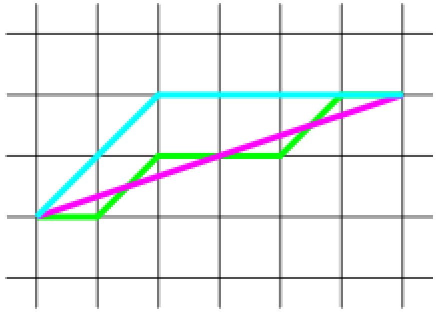


vs.



optimal path vs. optimal in the grid

resolution irrelevant
(why?)



every path in the grid is the same as:

- first all diagonal moves
- then all straight moves

increasing resolution is irrelevant

---

path sub-optimal: really a problem?

vehicles: large number of turns

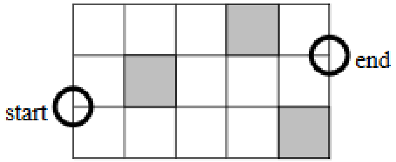
npc: innatural movements (drunken-like)

---

some solutions:

- visibility graphs
- post smoothing
- Field D*
- Theta*

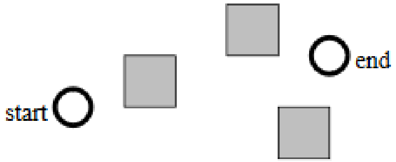
visibility graph



simple problem

---

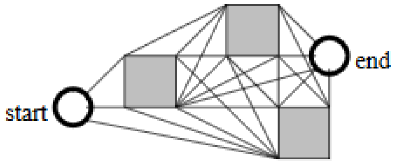
visibility graph: no grid



consider only the occluded areas

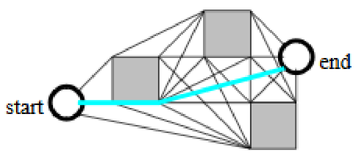
---

visibility graph: allowed moves



link every corner to every other
also: start and end

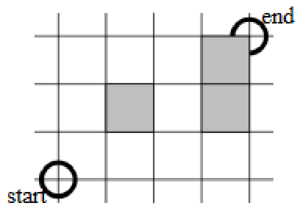
visibility graph: minimal path



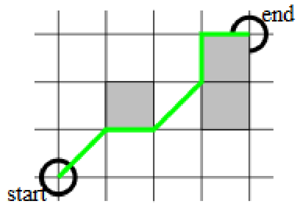
path in this new grid is optimal

but: many links
on the grid: only from each state to neighbors

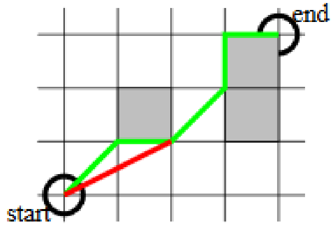
post-smoothing



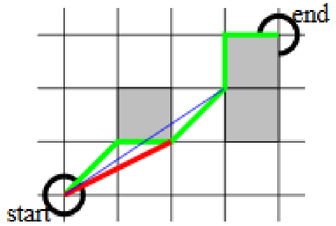
first find optimal path in the grid
then smooth it



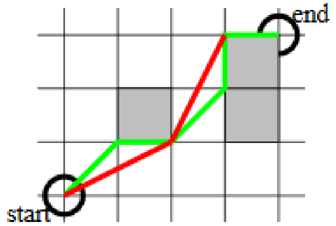
optimal path in the grid



third is visible from one: skip second

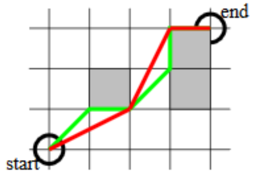


fourth is not visible from one: do not skip third
repeat from third node



fifth node visible from third
skip fourth

etc.

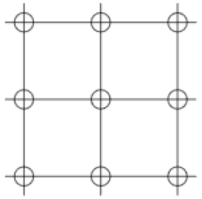


post-smoothing algorithm:

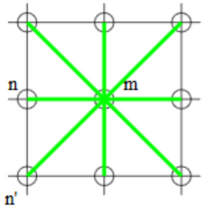
fix path by skipping nodes if successors are in line of sight of predecessor

---

field D*



the grid



`A*` on grid: allow only moves on the grid

not only!

also: `d(m) = min d(n) + n→m` only for nodes `n,n',`... on the grid

[note] A* allows only moves on the grid, but the grid matters not only to this. When determining the distance of a node from the start, only the precedecessors on the grid are considered; this is the second aspect of the algorithm affected by the grid.

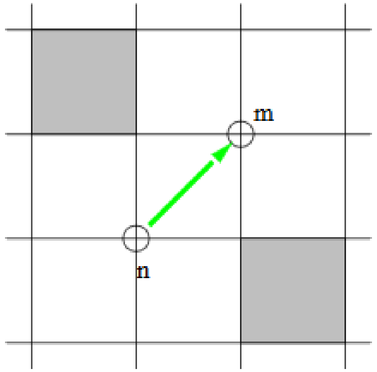
In the figure, `m` is a node for which field A* is about to calculate the distance `d(m)`. Only nodes on the grid like `n,n',`... are considered as *predecessors* of `m`. Therefore, `d(m)` is calulated using them only.

theta*

incorporates smoothing into searching

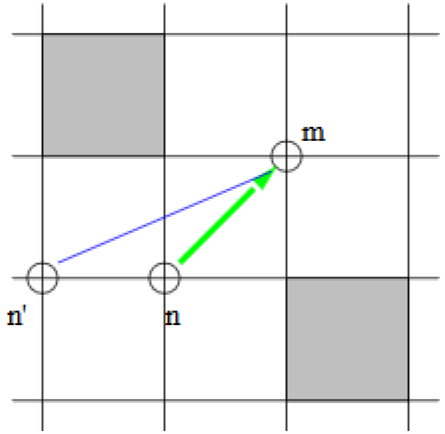
---

allowed moves



A* considers only moves like n→m
n and m neighbors in the grid

shortcuts

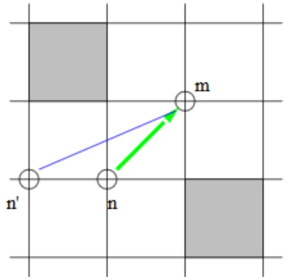


theta* checks whether n has a predecessor n'
with a line-of-sight to m

n' becomes a new predecessor of m

matters to?…

shortcut = short distance



theta* checks whether `n` has a predecessor `n'`
with a line-of-sight to `m`

`n'` becomes a new predecessor of `m`

matters on:

- value of `d(m)`
  `n'→m` shorter than `n'→n→m`
- optimal path:
  shorter
  smoothed (no turn on `n`)

[note] Precisely, theta* stores the best precedecessor of each node `n`, and only checks the line-of-sight from it to `m`. The new allowed move `n'→m` is not really added as an allowed move, only `n'` stored as the new best predecessor of `m` if `start⇒n'→m` is currently the shortest path to `m`.