ARTIFICIAL INTELLIGENCE 4 GAMES

DAVID R. WINER

FEBRUARY 8 2018

UPDATES

- Assignment 1 is still out (due night of Feb 13th)

UPDATES

- Assignment 1 is still out (due night of Feb 13th)
- Getting Started: Pathfinding (Dijkstra and A*)

UPDATES

- Assignment 1 is still out (due night of Feb 13th)
- Getting Started: Pathfinding (Dijkstra and A*)

Agenda

UPDATES

- Assignment 1 is still out (due night of Feb 13th)
- Getting Started: Pathfinding (Dijkstra and A*)

Agenda

- IDA* (variant of A*)
- Division Schemes
- Hierarchical Pathfinding
- *Getting Started: Behavior Trees*

COURSE CONTENT

AI for games:

- AI Methods
  - *Ad-Hoc Behavior Authoring (FSMs, Behavior Trees, Utility, Decision Tree)*
  - *Steering Behavior (seek, arrive, align, wander, pursue, pathfollow)*
  - *Tree* Search (*Dijkstra, A*,* **Division Schemes**, Minimax, MCTS)
  - Planning (STRIPS, HSP, POP, HTNs)
  - Learning (Evolutionary, Supervised, Unsupervised, Reinforcement)
- Ways of using AI in games
  - Playing games
  - Generating content
  - Modeling players

ERRATA FROM LAST LECTURE

How many edges in a fully connected graph?

ERRATA FROM LAST LECTURE

How many edges in a fully connected graph?

If **h(n)** is exactly equal to the cost of moving from n to the goal, then A* will only follow the best path and never expand anything else, making it very fast. Although you can't make this happen in all cases, you can make it exact in some special cases. It's nice to know that given perfect information, A* will behave perfectly.

ERRATA FROM LAST LECTURE

How many edges in a fully connected graph?

If **h(n)** is exactly equal to the cost of moving from n to the goal, then A* will only follow the best path and never expand anything else, making it very fast. Although you can't make this happen in all cases, you can make it exact in some special cases. It's nice to know that given perfect information, A* will behave perfectly.

The lower **h(n)** is, the more node A* expands, making it slower.

ERRATA FROM LAST LECTURE

How many edges in a fully connected graph?

If **h(n)** is exactly equal to the cost of moving from n to the goal, then A* will only follow the best path and never expand anything else, making it very fast. Although you can't make this happen in all cases, you can make it exact in some special cases. It's nice to know that given perfect information, A* will behave perfectly.

The lower **h(n)** is, the more node A* expands, making it slower.

If **h(n)** is sometimes greater than the cost of moving from n to the goal, then A* is not guaranteed to find a shortest path, but it can run faster.

ERRATA FROM LAST LECTURE

How many edges in a fully connected graph?

If **h(n)** is exactly equal to the cost of moving from n to the goal, then A* will only follow the best path and never expand anything else, making it very fast. Although you can't make this happen in all cases, you can make it exact in some special cases. It's nice to know that given perfect information, A* will behave perfectly.

The lower **h(n)** is, the more node A* expands, making it slower.

If **h(n)** is sometimes greater than the cost of moving from n to the goal, then A* is not guaranteed to find a shortest path, but it can run faster.

Admissible heuristics are ones that always underestimate (and therefore are optimal)

https://www.youtube.com/watch?v=g024lzsknDo
https://www.youtube.com/watch?v=X3x7BlLgS-4

A* HEURISTIC ON HEXAGONAL GRID

https://www.redblobgames.com/grids/hexagons/#distances

https://github.com/pgeerkens/HexGridUtilitiesForGames

ID DEPTH-FIRST SEARCH (IDDFS)

ID DEPTH-FIRST SEARCH (IDDFS)

Iterative Deepening

ID DEPTH-FIRST SEARCH (IDDFS)

Iterative Deepening

Depth-first search, but limit depth to some cutoff, increment at each iteration

ID DEPTH-FIRST SEARCH (IDDFS)

Iterative Deepening

Depth-first search, but limit depth to some cutoff, increment at each iteration

Start with depth of 1, and it behaves like breadth-first search on first iteration

Increment depth to 2, and repeat

ID DEPTH-FIRST SEARCH (IDDFS)

Iterative Deepening

Depth-first search, but limit depth to some cutoff, increment at each iteration

Start with depth of 1, and it behaves like breadth-first search on first iteration
Increment depth to 2, and repeat

Good for game trees

ID DEPTH-FIRST SEARCH (IDDFS)

Iterative Deepening

Depth-first search, but limit depth to some cutoff, increment at each iteration

Start with depth of 1, and it behaves like breadth-first search on first iteration

Increment depth to 2, and repeat

Good for game trees

Good for when the search space is large and the proximity to the solution is not known

ID DEPTH-FIRST SEARCH (IDDFS)

Iterative Deepening

Depth-first search, but limit depth to some cutoff, increment at each iteration

Start with depth of 1, and it behaves like breadth-first search on first iteration

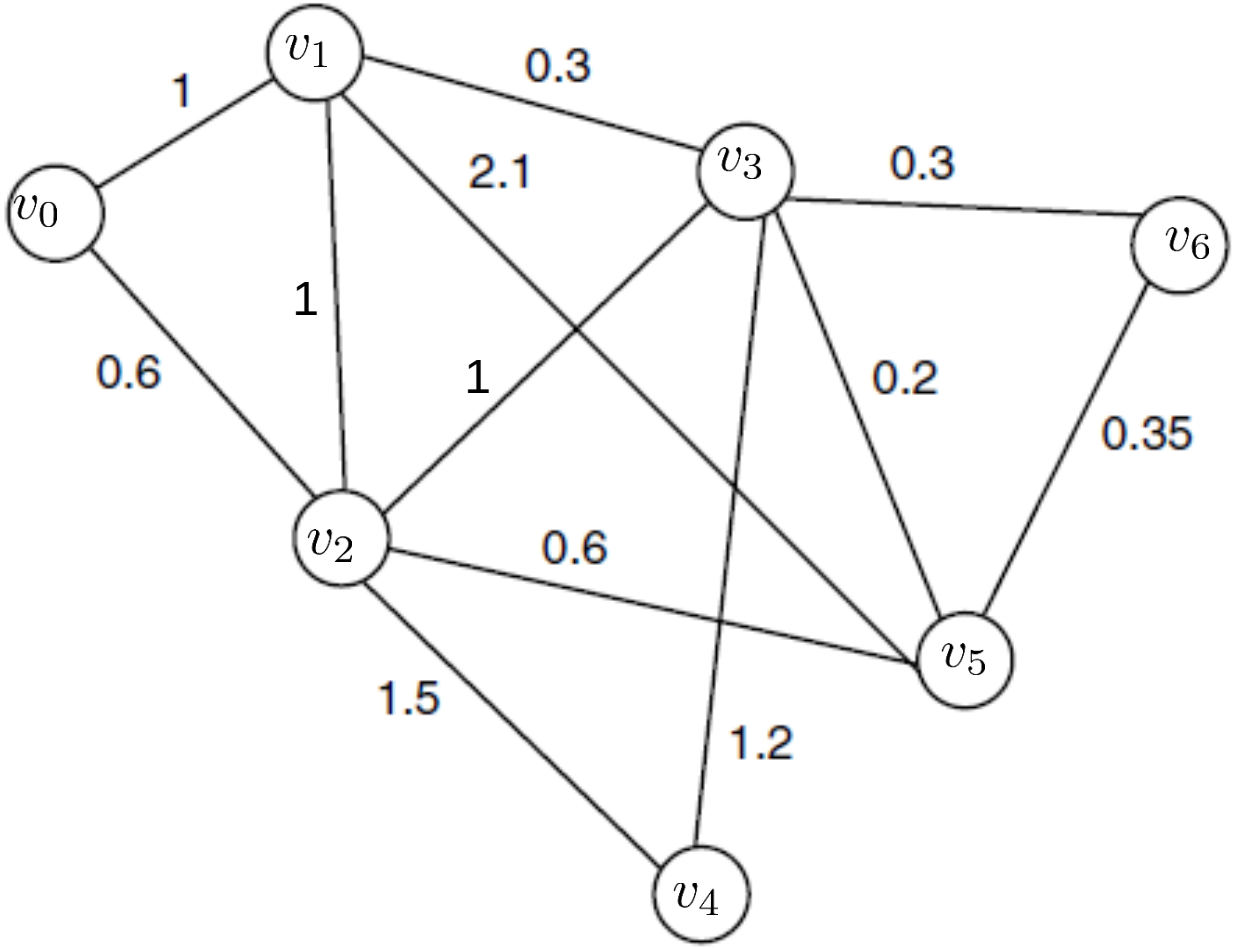
Increment depth to 2, and repeat

Good for game trees

Good for when the search space is large and the proximity to the solution is not known

Good when memory is limited (only needs a stack of nodes which represents the branch of the tree current being expanded, just like DFS)

ID-DFS



IDA*

IDA*

Iterative Deepening A*

IDA*

Iterative Deepening A*

Perform Depth-first search with dynamic threshold

IDA*

Iterative Deepening A*

Perform Depth-first search with dynamic threshold

Threshold starts at $h(n_0)$

and increases for each iteration of the algorithm

IDA*

Iterative Deepening A*

Perform Depth-first search with dynamic threshold

Threshold starts at $h(n_0)$

and increases for each iteration of the algorithm

At each iteration, threshold is the minimum **cost** of all values that exceeded the current threshold.

$$\overbrace{f(n) = g(n) + h(n)}$$

IDA*

Iterative Deepening A*

Perform Depth-first search with dynamic threshold

Threshold starts at $h(n_0)$

and increases for each iteration of the algorithm

At each iteration, threshold is the minimum **cost** of all values that exceeded the current threshold.

$$\overbrace{f(n) = g(n) + h(n)}$$

In two sentences :

Perform A* and suppose the nodes below the depth limit have no child.

If succeed, great, otherwise, perform again with new depth limit that is calculated from the last iteration.

IDA*

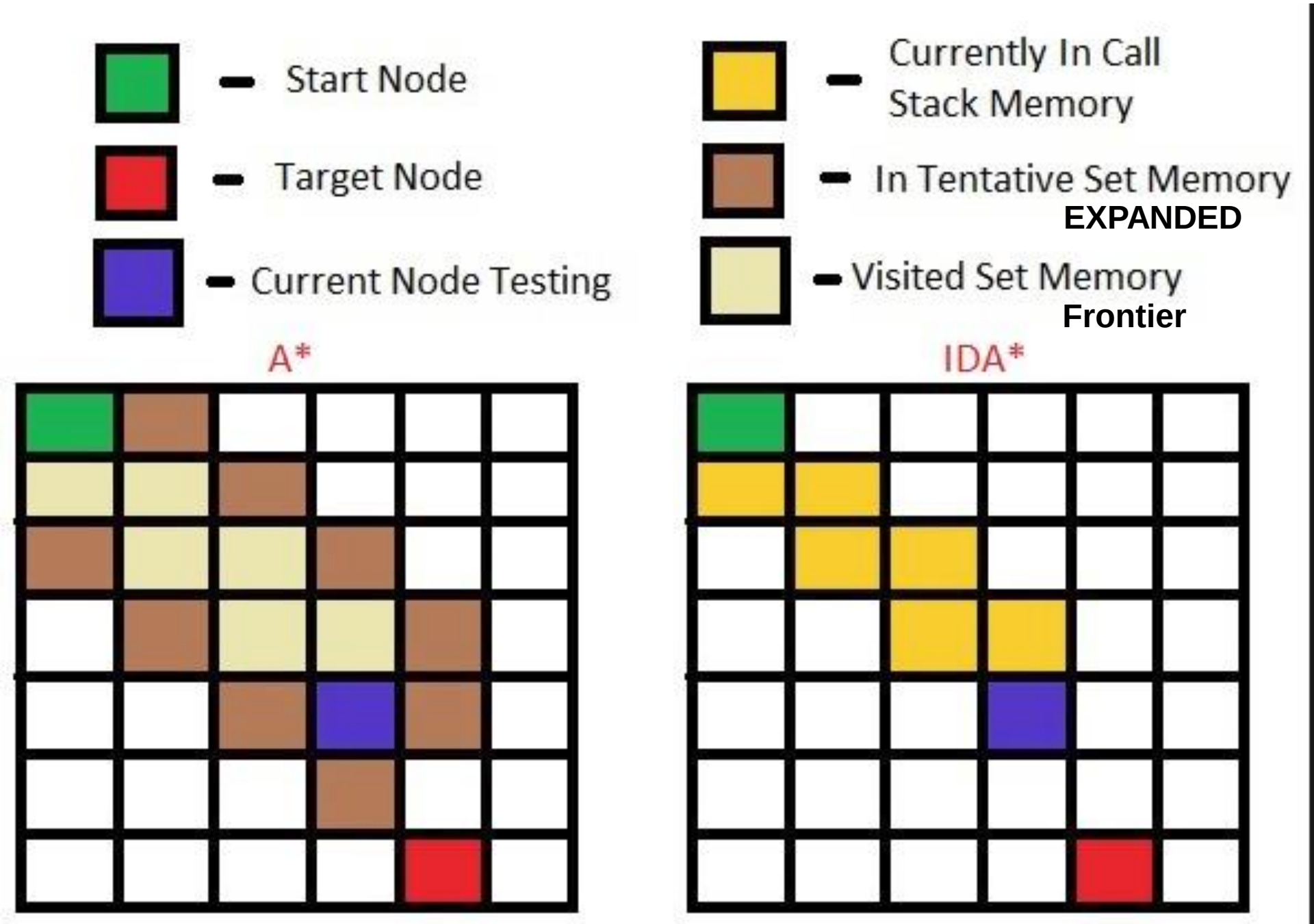
In two sentences :

Perform A* and suppose the nodes below the depth limit have no child.

If succeed, great, otherwise, perform again with new depth limit that is calculated from the last iteration.

Benefits:

- Don’t need a sorted search frontier (priority queue)
- Light on memory


Start Node
Target Node
Current Node Testing
Currently In Call Stack Memory
In Tentative Set Memory
EXPANDED
Visited Set Memory
Frontier
A*
IDA*

IDA*

In two sentences :

Perform A* and suppose the nodes below the depth limit have no child.

If succeed, great, otherwise, perform again with new depth limit that is calculated from the last iteration.

Benefits:

- Don’t need a sorted search frontier (priority queue)
- Light on memory

Disadvantages:

IDA*

In two sentences :

Perform A* and suppose the nodes below the depth limit have no child.

If succeed, great, otherwise, perform again with new depth limit that is calculated from the last iteration.

Benefits:

- Don’t need a sorted search frontier (priority queue)
- Light on memory

Disadvantages:

- Memory not usually an issue

IDA*

In two sentences :

Perform A* and suppose the nodes below the depth limit have no child.

If succeed, great, otherwise, perform again with new depth limit that is calculated from the last iteration.

Benefits:

- Don’t need a sorted search frontier (priority queue)
- Light on memory

Disadvantages:

- Memory not usually an issue
- Does not exploit the heuristic as heavily

VARIANTS OF PATHFINDING

Pathfinding on Nav Meshes

- Daniel Brewer. Tactical pathfinding on a navmesh. Game AI Pro: Collected Wisdom of Game AI Professionals, page 361, 2013
- Paul Tozour and I. S. Austin. Building a near-optimal navigation mesh. AI Game Programming Wisdom, 1:298–304, 2002

Jump Point search on grids

- Daniel Damir Harabor and Alban Grastien. Online Graph Pruning for Pathfinding on Grid Maps. In AAAI, 2011.

Hierarchical A*

- Adi Botea, Martin Muller, and Jonathan Schaeffer. Near optimal hierarchical path-finding. ¨ Journal of Game Development, 1(1):7–28, 2004.
- Nathan Sturtevant. Memory-Efficient Pathfinding Abstractions. In AI Programming Wisdom 4. Charles River Media, 2008.

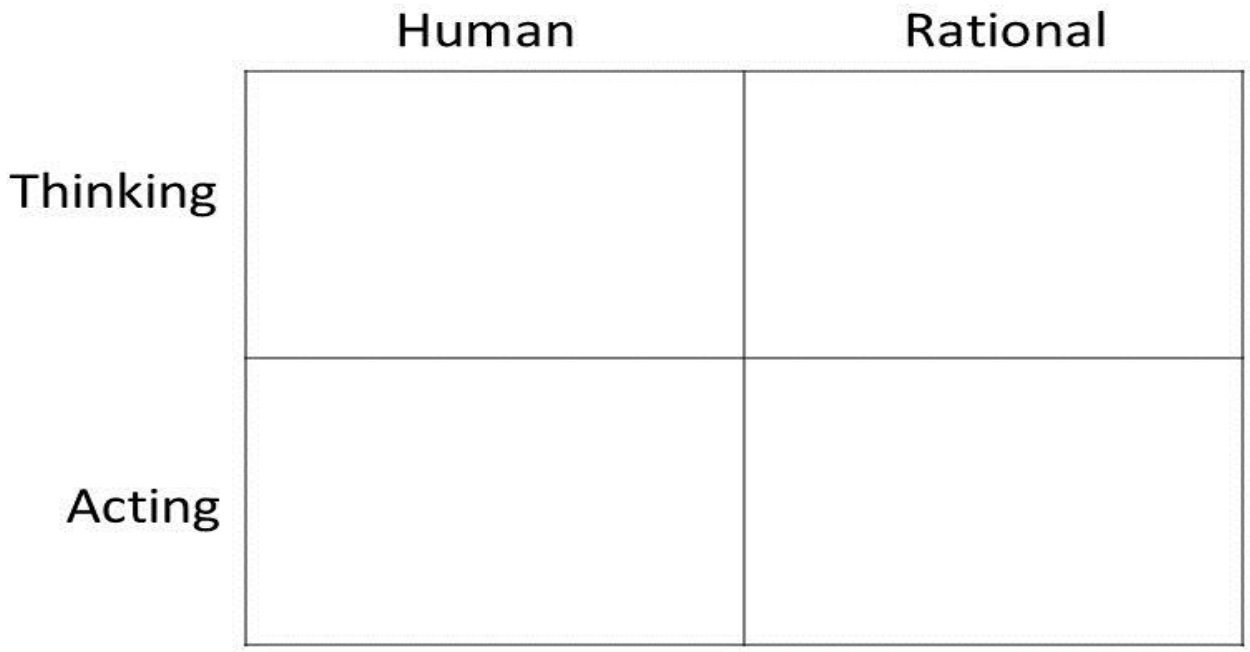
https://www.youtube.com/watch?v=u_GmHXJ3Ti0

NEXT TIME

Feb 15th

Minimax

Approaches to A. I.



This model from Russell and Norvig.

Approaches to A. I.

	Human	Rational
Thinking	Behavior is explained by human-like cognition	Behavior explained by rational thought
Acting	Behavior is like a human's in this situation	Behavior is rational in this situation

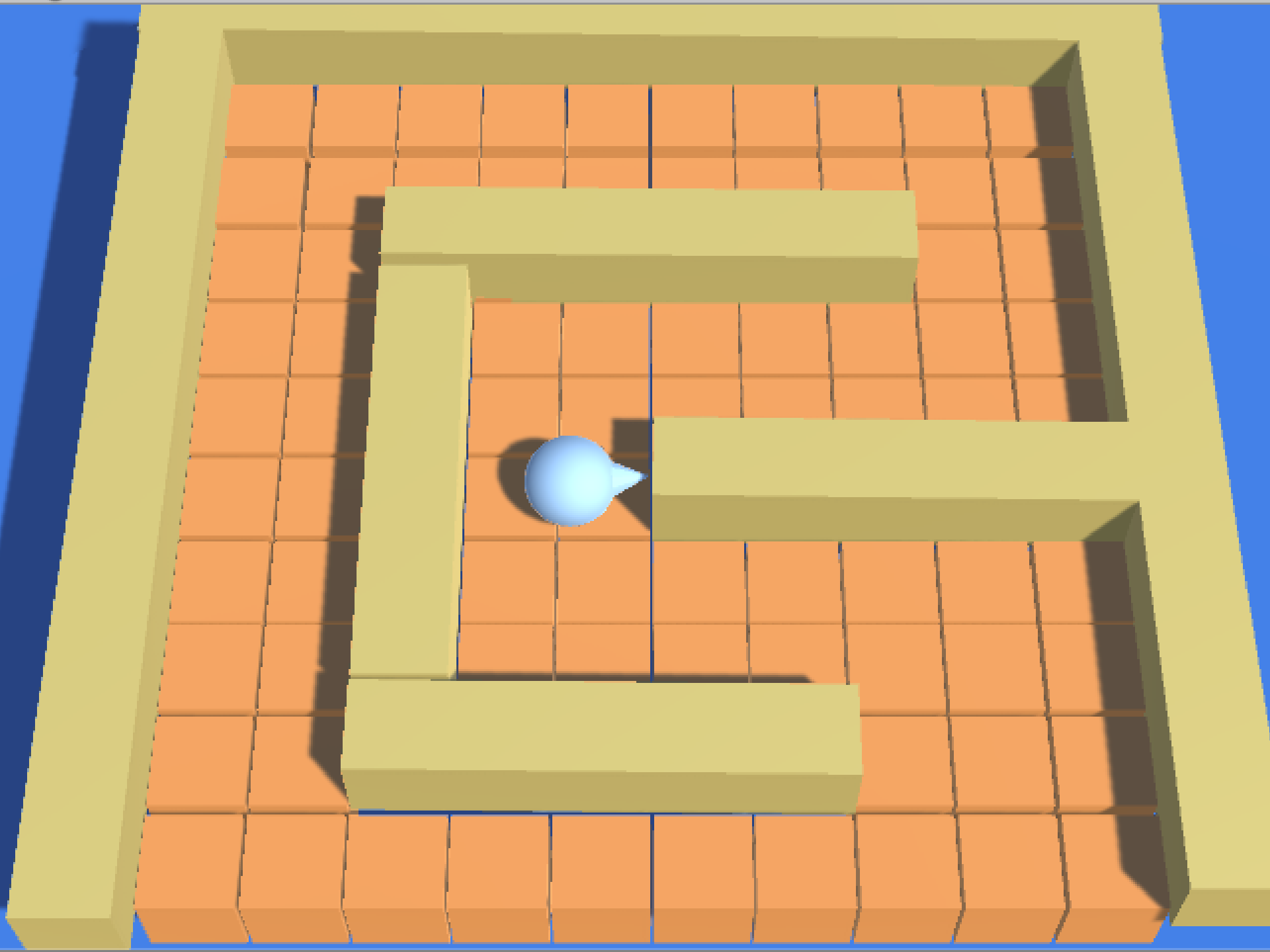
This model from Russell and Norvig.

Approaches to A. I.

	Human	Rational
Thinking	Behavior is explained by human-like cognition	Behavior explained by rational thought
Acting	Behavior is like a human's in this situation	Behavior is rational in this situation

This model from Russell and Norvig.

Optimal behavior is rational, sometimes. Sometimes more rational



EVOLUTIONARY LEARNING WITH STEERING

https://www.youtube.com/watch?v=BhsgLeY_Q-Y