A 星算法
及其求解理论的推广和前沿探讨

顾天意

University of New Hampshire

Department of Computer Science

2018 年 1 月 2 日

个人情况

- 2006-2010 SMU，物流工程学院，物流工程专业，本科
- 2010-2012 SMU，科学研究院，物流工程专业，硕士
- 2012-2015 SIPG，上海海勃物流软件，软件工程师
- 2015-2021 UNH, Computer Science, PhD（AI 方向）

微信

个人主页

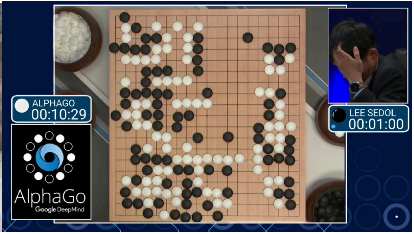
Problems in AI

AI Today

ASIMO
HONDA

Rmax
YAMAHA

03
MDV


ALPHAGO
00:10:29
AlphaGo
Google DeepMind
LEE SEDOL
00:01:00

Problems in AI

Agent ⇔ Environment

Perception: vision, state estimation

Planning: low/high-level, on/off-line, incremental/repair

Acting: dispatching, monitoring, diagnosis

Reflex: sensors → effectors

Reflex with state: sensors + state → effectors + new state

Goal-based: reason from goals to means

Utility-based: use quantitative measure of happiness

What kind of agent?

1. Thermostat
2. autonomous armed drone
3. Mail delivery robot
4. Medical diagnosis system

Environments

Observability: complete, partial, hidden

Predictability: deterministic, strategic, stochastic

Interaction: one-shot, sequential

Time: static, dynamic

State: discrete, continuous (also time, percepts, and actions)

Agents: single, multiagent (competitive, cooperative)

Environments

Observability: complete, partial, hidden

Predictability: deterministic, strategic, stochastic

Interaction: one-shot, sequential

Time: static, dynamic

State: discrete, continuous (also time, percepts, and actions)

Agents: single, multiagent (competitive, cooperative)

State-Space Search

Formalizing Problem Solving

State: hypothetical world state

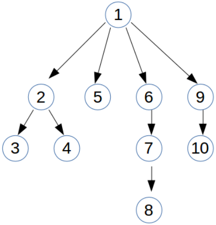
Operators: actions that modify world

Goal: desired state or test

(Herbert Simon and Allen Newell, “Computer simulation of human thinking and problem solving”, 1961)

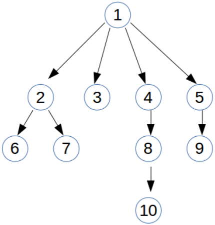
Depth-First Search

```
open ← an ordered list containing just the initial state.
Loop
   If open is empty,
      then return failure.
   Node ← Pop(open).
   If Node is a goal,
      then return Node (or path to it).
      else
         Children ← Expand (Node).
         Add Children to front of open.
```



Breadth-First Search

```
open ← an ordered list containing just the initial state.
Loop
  If open is empty,
    then return failure.
  Node ← Pop(open).
  If Node is a goal,
    then return Node (or path to it).
    else
      Children ← Expand (Node).
      Add Children to end of open.
```



Uniform-Cost Search

```
open ← an ordered list containing just the initial state.
Loop
  If open is empty,
    then return failure.
  Node ← Pop(open).
  If Node is a goal,
    then return Node (or path to it).
    else
      Children ← Expand (Node).
      Merge Children into open, keeping sorted by path cost.
```

Dealing with Graphs

- Check for cycles with ancestors
- Maintain closed list (hash table) to detect duplicates

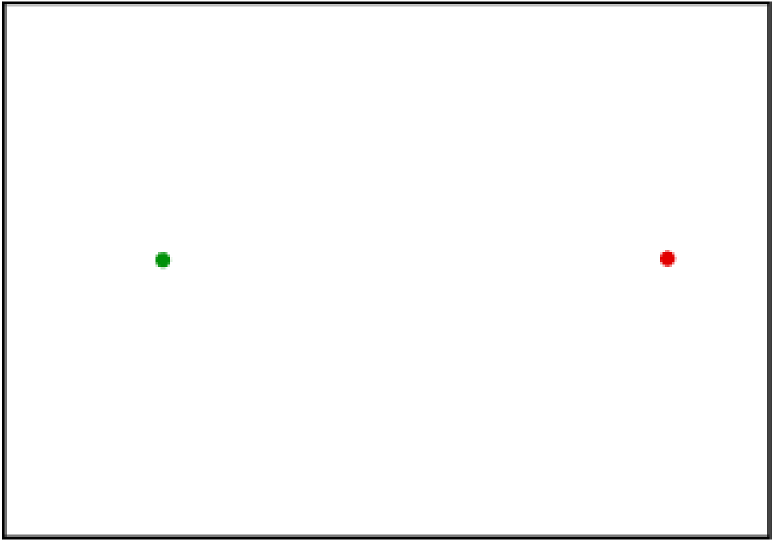
Dealing with Graphs

- Check for cycles with ancestors
- Maintain closed list (hash table) to detect duplicates

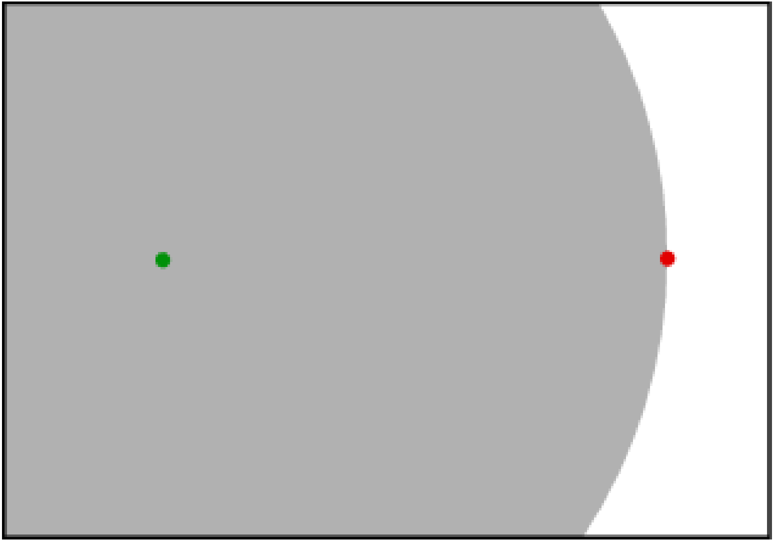
Dijkstra!!!

A^*

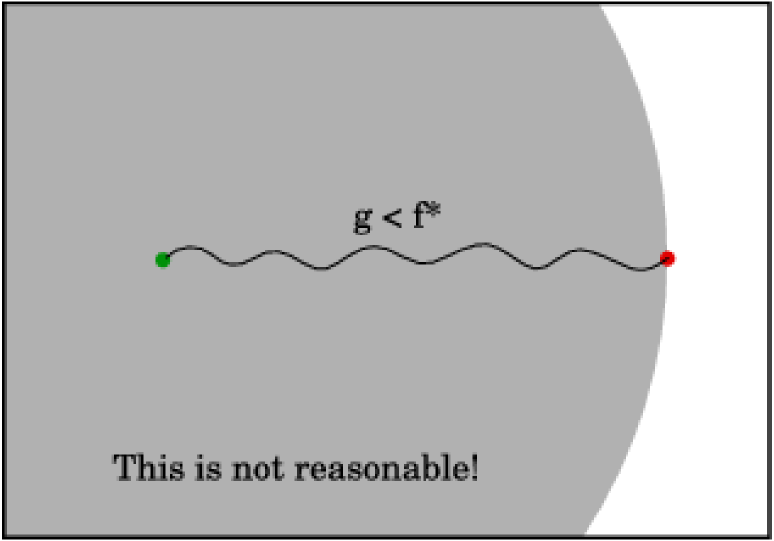
UCS Behavior



UCS Behavior



UCS Behavior



Heuristic Evaluation

> Heureka(Eureka)!
> — Archimedes

Eureka（古希腊词语）

本词条缺少**信息栏**、**名片图**，补充相关内容使词条更完整，还能快速升级，赶紧来编辑吧！

Eureka，古希腊词语，英文读音[juə'ri:kə]，词性为感叹词，意思是“我找到了！我发现了！”

据传，阿基米德在洗澡时发现浮力原理，高兴得来不及穿上裤子，跑到街上大喊：“Eureka(我找到了)！”[1-13]

Heuristic knowledge is useful, but not necessarily correct.

Heuristic algorithms use heuristic knowledge to solve a problem.

A heuristic function takes a state and returns a lower bound on the cost-to-go to reach a goal.

(Newell and Ernst, 1965; Lin, 1965)

A* Search

Consider estimated final path cost! $f(n) = g(n) + h(n)$

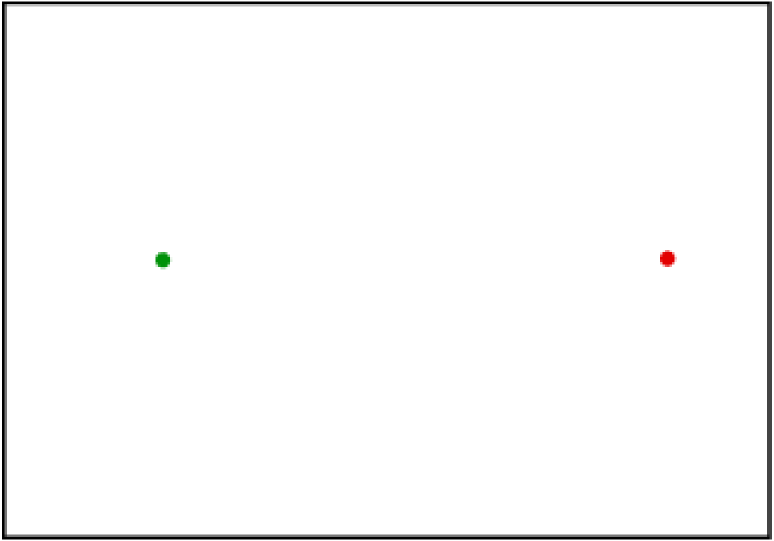
```
$Q$ ← an ordered list containing just the initial state.
Loop
   If $Q$ is empty,
      then return failure.
   *Node* ← Pop($Q$).
   If *Node* is a goal,
      then return *Node* (or path to it)
      else
         *Children* ← Expand (*Node*).
         Merge *Children* into $Q$, keeping **sorted by** $f(n)$.
```

An Example: the 8-puzzle

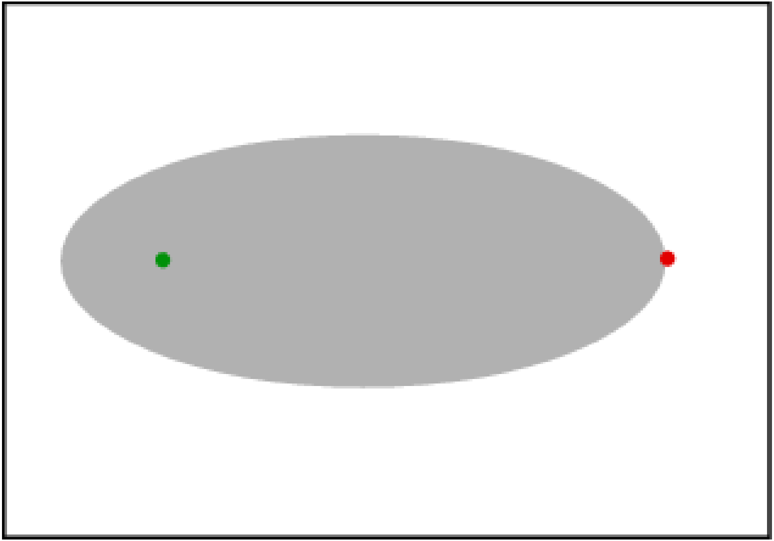
h(n) = number of tiles out of place. (The blank is not a tile.)

	2	8	3		1	2	3
Start state:	1	6	4	Goal state:	8	⊔	4
	7	⊔	5		7	6	5

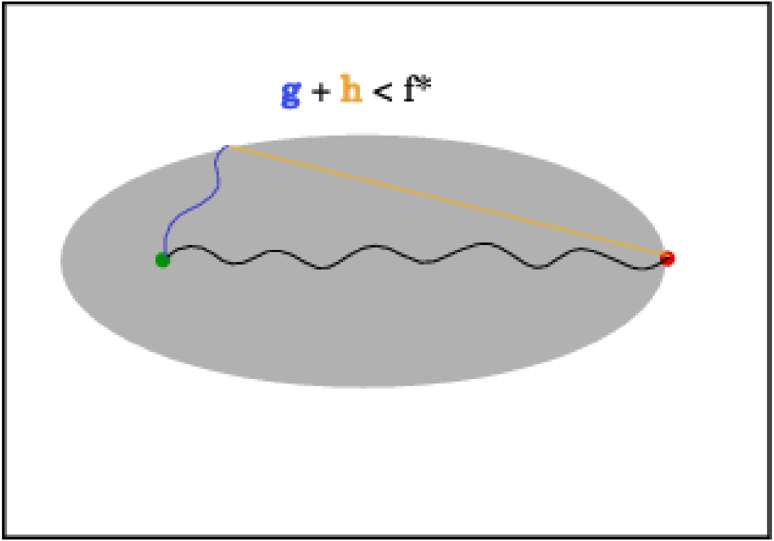
Why Fewer Nodes?



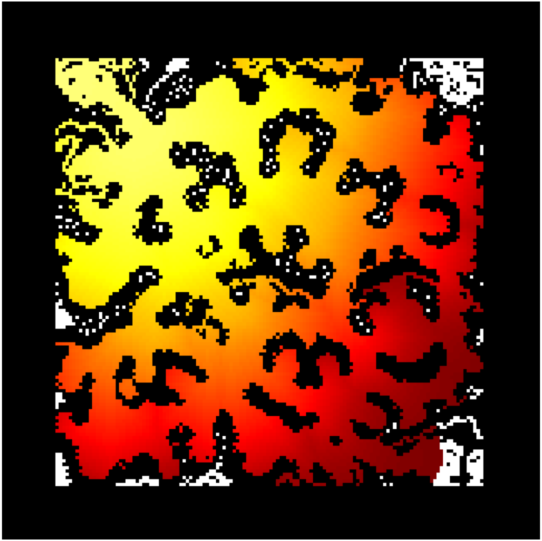
Why Fewer Nodes?



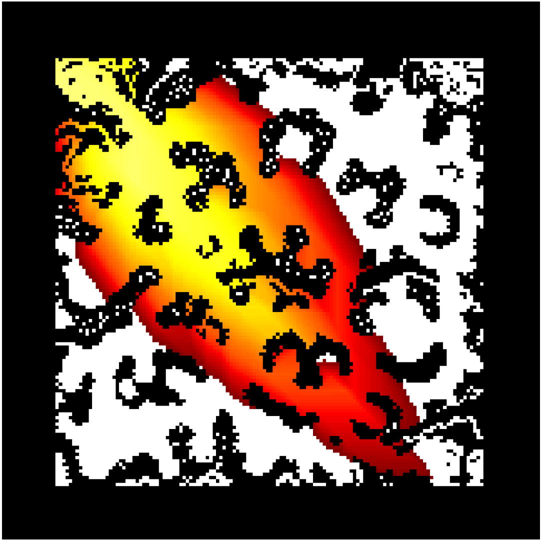
Why Fewer Nodes?



UCS Behavior



A* Behavior



Optimality of A*

1. For admissible h, f can be made non-decreasing.
2. A* expands nodes in order of non-decreasing f.
3. Must examine all nodes with f < f*.

Heuristics

Simplified problem must give lower bound on original!

1. Relaxation: fewer and/or weaker constraints
   - Sometime efficient closed form
2. Abstraction: simplify token identity
   - Smaller search space

Want highest value

- If $h_1(n) \leq h_2(n)$ for all n, h_2 dominates h_1

Need fast computation

Suboptimal Search

Problem Settings

optimal: minimize solution cost

suffer all with $f(n) = g(n) + h(n) < f^*$

---

greedy: minimize solving time

bounded suboptimal: minimize time subject to relative cost bound (factor of optimal)

bounded cost: minimize time subject to absolute cost bound

contract: minimize cost subject to absolute time bound

anytime: iteratively converge to optimal

utility: maximize given function of cost and time

Weighted A*

$$\mathrm{f}'(\mathrm{n}) = \mathrm{g}(\mathrm{n}) + \mathrm{w} \cdot \mathrm{h}(\mathrm{n})$$

- nodes with high h(n) look even worse
- suboptimality bounded: within a factor of w of optimal!

wA* Behavior

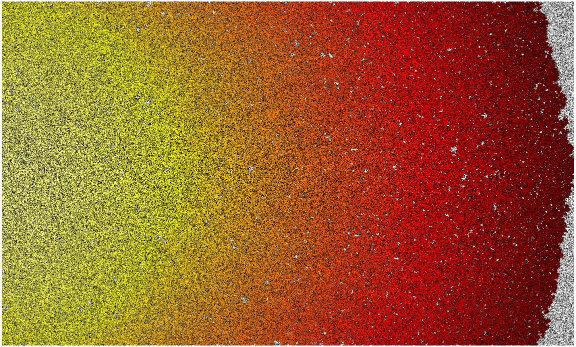


图: optimal: uniform-cost search

wA* Behavior

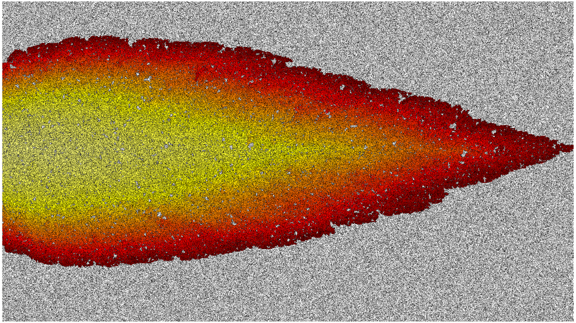


图: optimal: A*

wA* Behavior

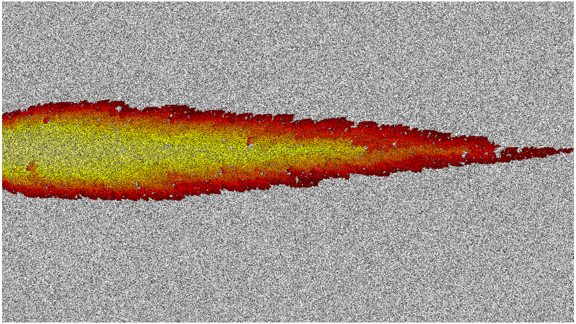


图: bounded suboptimal: Weighted A*

Anytime Search

Anytime A*

1. run weighted A*
2. keep going after finding a goal
3. keep best goal found (can test at generation)
4. prune anything with f(n) > incumbent

Anytime Restarting A* (ARA*): lower weight after finding each solution

Real-time Search

LSS-LRTA*

1. single A* lookahead (LSS)
2. update all h values in LSS
3. move to frontier

Summary

Uninformed: DFS, UCS
Optimal: A*
Bounded suboptimal: wA*
Anytime: Anytime A*, ARA*
Real-time: LSS-LRTA*