lec #9

Conditions for Optimality:

① Heuristic is admissible and optimistic

② Consistency (monotonicity)

properties of A*:

- optimal → لو بيحقق الشروط
- Complete

→ Time complexity: exponential

[4] Iterative Deeping A*

بيشوف الإجمالى عشان يمنع من ال space

- to reduce memory requirements for A*
- the Cutoff is the f. Cost (g+h) rather than the depth

بدل ما بيحصل Cut off لما يوصل ل depth معينة، بيعمل Cutoff بناءً على إجمالى ال cost

- at each iteration: the cuttoff value is the smallest f-Cost of any node

⑤ Recursive BFS

- applying BFS using only linear space

Subject موضوع الدرس Date التاريخ

- uses the F-limit variable keeps track of the Best alternative value path.
- if the current node exceeds this limit, the recursion returns to the alternative path
- Optimal if h(n) is admissible
- linear space complexity.
- time complexity depends on ① Accuracy of h(n)
  ② how often the best path changes

Example: Admissible function : 8-puzzle

عايزين نعرف Heuristics تقيس مدى قربنا أو بعدنا عن الحل.

3	1	0
⑦	②	④
5		6
8	3	1

Start state

↓

	1	2
3	4	5
6	7	8

Goal state

من فكرتين

- $h(n)$ = misplaced tiles (عدد الحجات اللي مش في مكانها)
- $h_2(n)$ = Manhattan distance (عشان توصل لمكانها عايزة كام خطوة)

$h_1(S) = 8$ → الـ8 قطع مش في مكانهم

$h_2(S) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$

الـ step عشان ⑦

steps for ②

Subject موضوع الدرس

بين الأهم من التاريخ الثابتة

(Dominance Heuristics)

- لو عندي ال © ← Admissible

$h_2(n) \geq h_1(n)$

then. $h_2(n)$ dominates $h_1(n)$ (for all (n))

← اشتغل على اللي ليها قيم أكبر

- Domination translates to efficiency

Relaxed problems

عشان نعرف توصل لـ h(n) صح لازم نعمل Relaxation للمشكلة يعني نبسطها شوية

من حالة $h_1(n)$	من حالة $h_2(n)$
A tile can move anywhere (يعني تتحرك في اي مكان على طول من خطوة واحدة)	A tile can move to any adjacent square
then $h_1(n)$ gives the shortest path.	then $h_2(n)$ gives the shortest path.

- Relaxed problem = A problem with fewer restrictions on the actions

Subject موضوع الدرس Date التاريخ

Chapter: 4

Beyond classical Search

local Search algorithms →

نوع من انواع الـ optimization

Notes

لو . لو عندي goal معروف ساعتها الـ path اللي هيهمني
. ساعات بيكون عندي, حتى الـ goal نفسه مش عندي الـ
goal نفسه ساعتها الـ path مش هيهمني

- the goal state itself is the solution
  target → to find a configuration that satisfy constraints.

- keep a single current state and try to improve it
  ↓
  aim to find the best state according to an objective function.

- state-space landscape
  1. location defined by state
  2. elevation defined by h(n) cost

Subject موضوع الدرس Date التاريخ

- elevation = cost (global minimum) ↘
اقل تكلفة ممكنة

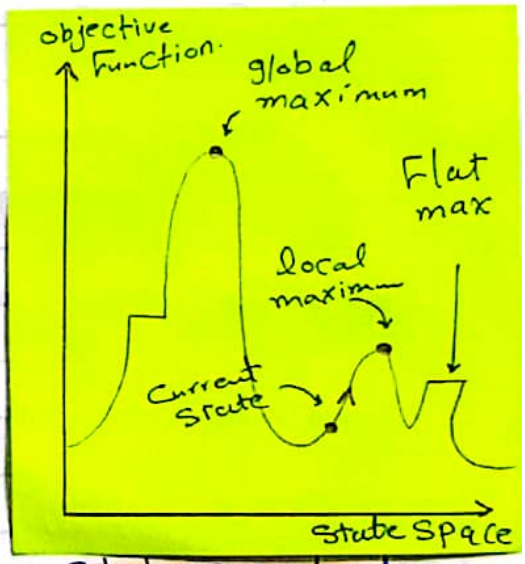
- elevation = objective (global maximum) ↘
اكبر انتفاع ممكن

- localsearch algorithms explore this landscape

Complete
لو لقى الـ goal
Complete يبقى

optimal.
لو لقى الـ goal، طلع الـ
Global min/max
optimal, Complete يبقى

problem: depending on initial state,
can get stuck in local max ↘



State space landscape

Algorithms

1) Hill-climbing search (HCS)

Function Hill-Climbing (problem)
returns (state with local max)
{
initial Current ← make_node (problem.initial state)
loop { → هيحاول يجيب اقرب نقطة حواليه اعلى منه
neighbor ← highest value successor of (Current)
← if neighbor.value ≤ Current.value
(return current)
Current ← neighbor }
}

Check

Subject موضوع الدرس Date التاريخ

- loop terminates when it reaches a "peak"
↓
اعلى حاجة ممكنة حواليه ، ممكن تطلع local او global

Example: 8-queens problem

Heuristic Function h=17 ← Current
↓ بعد الـ queens اللي ممكن تـ attack بعضها

h=12 ← Best
↓ احسن حركة ممكن اعملها

← بعد التحسينات الـ local حركة ورا التانية ، اخسن (h) طلع بيوا (h=1)

local min ↓

* Hill-Climbing search = greedy local search
↓ لأنه بيشوف احسن حل قريب منه بيروح له مش البعيد

- HCS frequently gets stuck in local (maxima/min)
=
↓ بتقف عند حتة مش قادرين نخسن بعدها ، مش بتكون goal

- Ridges (flat)
جزء مستوى فيه كل الـ max جنب بعض
sequence of local max that is very difficult to navigate

- plateaux
progress is no longer possible.
كل اللي حواليا اسوء منه

بس مختلف، صعب اوي

مش دايما بيمشي local، بيعمل حركات فجائية في الاتجاه العكسي

Subject موضوع الدرس Date التاريخ

2 Simulated Annealing Search

key idea: skip local maxima by allowing some "bad" moves but gradually decrease their size and frequency.

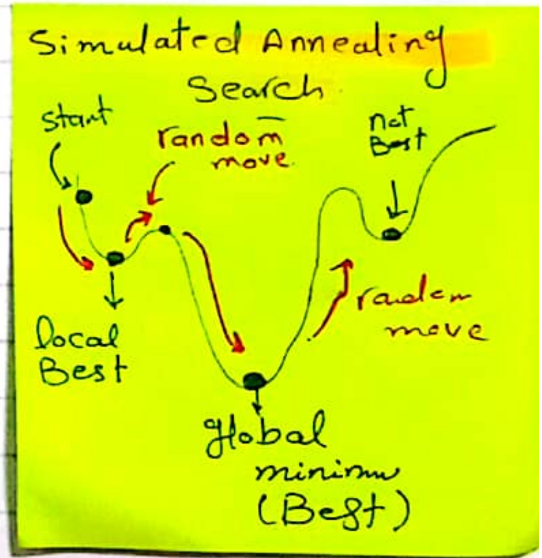
1 - instead of the Best move, make a random move
2 - take a step to escape local maxima
3 - if it improves the situation then it's accepted.

لو مثلا بتدور على الـ min وتنزل لـ local min، اعمل حركة عشوائية واطلع عشان تشتغل من local min

- physical analogy with annealing process to harden metals

يشبه عملية تقوية الحديد، الصلب (تسخين شديد + تبريد فجأة)

Algorithm → just for reading



3 local Beam Search (LBS)

بيمشي مع كذا state في نفس الوقت ويدور على حل.

key idea: keep track of k states rather than only one

- at each iteration:
all the successors of all k states are generated

لو كلهم مجاش لو يجيبو solution، في منهم هيقعدوا في local وعلى الأقل واحد هيوصل للـ goal

- useful info can be passed among parallel threads

بيتواصلوا مع بعض عشان اول ما يوصلوا للحل الـ Search بيقف