Subject: lec #6 Date: / /

Real World problems

① **Touring problems**
related to route-finding problem
but with differences

مشكلة الطرق
والحركة مرمدنية
للتانية كان اسمها
route finding
problem

② **Robot navigate problems**
generalization of the
route finding problem
(rather than discrete set of routes, robots can move in
a continuous space with infinite set of actions)

③ **VLSI layout problem** (very large scale integration)
: عايزين نحط مجموعة components في circuit عشان تعمل task
معينة وعايزين المساحة المحدودة تبات الـ circuit تكون صغيرة
requires positioning millions of components and connections
on one slip.

objectives: minimize Area
minimize circuit delay

④ **Automatic assembly sequencing**
هجمع حاجات بترتيب معين عشان يطلع الـ object مضبوط

⑤ **protein design**
: تصميم نظام متكامل من يناسب
حالات مرضية معينة

Searching for Solutions

- search process = Building up a Search tree

- root = initial state
- branches = actions.
- nodes = States in space

فى ال Search بيشتغل ازاى؟

Important:

ال state space دايماً هنرسمه عامل graph مش شرط tree. بس ال Search هنرسمه دايماً tree. وال Solution هيبقى leave ف ال tree بتاعت ال Search

① At each test. testing whether the current state is a goal state

بيعمل test كل ما يقف عند state معينة

② if not, Expanding the current state. by applying each action to generate a new set of states.

بيشوف كل ال Actions اللي يقدر يعملها ف ال current state اللي واقف عندها

③ Selecting one state, And putting others aside for later (In case the selected state didn't lead to a Solution)

بيختار بقا واحد state من اللي طلعهم . ويخلي الباقي على جنب عشان يرجعلهم لو اللي معاه مشتغلتش.

طريقة اختيار ال state اللي هيمشي فيها بتختلف من Algorithm للتاني واسمها

→ Search strategy

ال list ثابتة ال states اللي بتتحفظ بيها
في حالة ان ال state اللي اخترتها مش شغالة
(backtracking)

Note: → **Frontier:** the set of all leaf nodes available for expansion at any given point.

Initial tree search Algorithm

Formulation
← سؤال الامتحانات Func

function **treeSearch** (problem) returns **Solution** (لو لقى الحل) or **Failure** (لو ملقاش الحل)

initialize the Frontier using initial state
loop do:
  if frontier is empty return Failure.
  ← ده معناه ان معنديش حاجة بدور فيها أصلا.
  else:
    choose a leaf node and remove it from the frontier
    ← اختار حل من الحلول وشيله من ال list
    if the node contains goal state
    then: return the corresponding solution ← لو فيها حل بيرجع هو الحل
    else: expand the chosen node, and add the resulting node to Frontier ← لو مفيهاش ارجع بحث تاني من الأول

Important ← الإضافة لل Frontier، السحب منه هيختلف على حسب ال technique بتاع ال Search

بسرعة بالك مش هنفضل نعدي على نفس الـnodes مرتين !!

Avoiding Repeated States

- Expanding repeated states by loopy pathes
  → المفروض الـ State اللي عملناها select مقدرش أعمله تاني

- loopy path = Redundant path

- Explored set [closed list]
  ↳ which remembers every expanded nodes

حلها:
انه زي ما عاملين list للـ nodes اللي عملنالها expand, لسه معملناش check, نعمل list للحجات اللي خلصت عشان منرجعلهاش تاني

problems with repeated states = Graph Search

الذكي كان اسمه tree Search

Initial Graph-Search Algorithm

Function Graph-Search (problem) returns Solution Failure

initialize the Frontier using initial state.
initialize the explored set to empty
loop do
  if frontier is empty return failure
  else: choose a leaf node and remove it from Frontier
  add this node to explored Set.
  if node contains goal state return Solution
  expand the chosen node only if it's not in the Frontier or the explored set.

State space vs. Search tree

مش ضروري ابني ال statespace كله قبل ما اشتغل بس ضروري ابني ال tree

node	state
• datastructure used to represent the search tree هي تمثل ال state بس فيها معلومات كتير اكتر عن ال state	• configuration of the real world
• has parent or children	• has no parents or children
• is on particular path في طريق محدد بمشي فيه عشان اوصلها	• —

Note → two different nodes can contain the same state. if it's generated by different search path

ممكن يكون في state واحدة موجودة في اكتر من node لأنها موجودة في اكتر من path

Needed data structures

① **state**
state to which the node corresponds
ال state اللي ال node بتمثلها.

② **parent**
the node that generates this node

③ **Action**
the action that was applied to generate it
ال Action اللي , ماشي عشانه

④ **path cost**
the cost from the initial state to this node

Frontier : priority queue

Remember,
1. test
2. Expand
3. Select

Search strategies

- search strategy is defined by the order of node expansion

هنشوف هنقيم الـ Search strategy على اى أساس؟

1. Completeness — تخلص ويلاقي حل ولا
2. Optimality — بيلاقي الحل بأقل تكلفة ولا.
3. time Complexity
4. Space Complexity

(note) فى فرق ما بين الـ search time والـ execution time
كدة اللي يهمنا

(الـ Factors اللي بتأثر على الـ time والـ space)

b → maximum branching Factor of the search tree
يعني لو هي Binary يبقى b = 2 → أكبر عدد ممكن أطلعه من الـ node الواحدة.

d → depth of least-Cost Solution
الـ Solution الأقرب هلاقيه فى الـ tree لأنه ده هيوريني هخش كم عمق قدامي

m → maximum length of any Path in the state space
من أقرب نقطة لآخر نقطة الـ max بتاعي قدامي

Uninformed (Blind) Search

- no additional information about states beyond problem definition
- they can: - generate successors
  - distinguish goal state from non goal state

الفرق بين ال Uniformed tech. في ال order اللي بعمل بيه
Expand لل nodes في tree

[1] Breadth-first Search (BFS)

بيمشي في ال tree بالعرض level by level

- Expand and check nodes at level d
  then: Expand and check nodes at level d+1
  and so on...

ودة هيضمن انه يجيلك أقرب solution

- nodes visited first will be expanded first
- Frontier: FIFO queue