Intro to AI 2021B Ex.1

Alon Tifferet , Muaz Abdeen
31497638 300575297

Uninformed Search

(1) a. True.

Both of them uses a priority queue as a fringe.

In case of A^*, the evaluation function of the cost of node n is: $f(n) = g(n) + h(n)$, where g is the actual cost to n, and h is an admissible heuristic.

However, in UCS we use the same evaluation function with null heuristic, $\forall n \; h(n) = 0$, which is admissible.

b. True, if iterative deepening starts always from 0

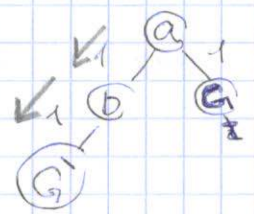
case (1) if iterative deepening starts alway from 0

If by contradiction the solution is not optimal, then there must be another goal state at shallower depth, but according to the iterative deepening it starts with depth 0 to m (the ~~maximum~~ maximum depth of the problem, so it must finds the goal at the shallower depth, contradiction.

It may perform worse (time and space) but finds an optimal solution always if step cost = 1.

case False, if we can assign any initial value to d_{max}.

counter example



if init $d_{max} = 2$ iterative deepening DFS will return G` which is not optimal

(2) Missionaries and Cannibals:

a. Formal search problem $= \langle S, S_0, G, A, F, C \rangle$

$S = \{(c,m,b)\}$ the state space given in the question

$S_0 = \{(3,3,0)\}$

$G = \{(0,0,0), (0,0,1)\}$

A = { moving the boat to the opposite side ~~from one bank to the other~~ }

{ move the boat from side 0 to 1,

move from 1 to 0 }

$C = C((S, action)) = 1$ uniform cost for all actions

F = the transition function

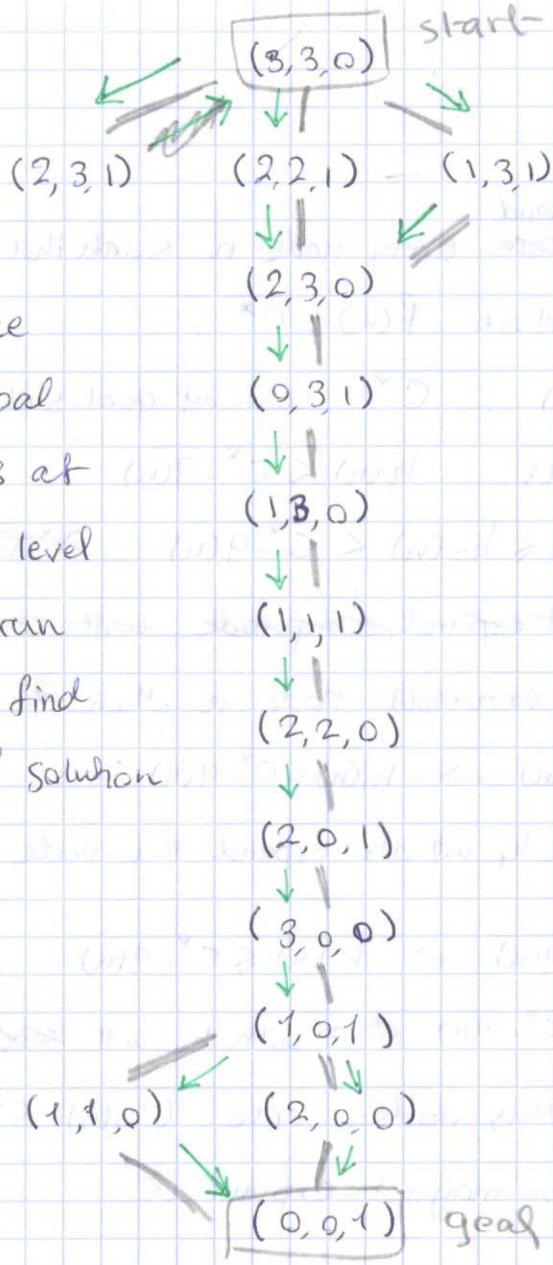
$$
\begin{aligned}
(c,m,0) \rightarrow\ & (c-2, m, 1) : && c \geq 2,\ m = c-2 \\
& (c-1, m, 1) : && c \geq 1,\ m = c-1 \\
& (c-1, m-1, 1) : && c, m \geq 1 \quad c = m \\
& (c, m-1, 1) : && m \geq 2,\ c = m-1 \\
& (c, m-2, 1) : && m \geq 2,\ c = m-2
\end{aligned}
$$

$$
\begin{aligned}
(c,m,1) \rightarrow\ & (c+2, m, 0) : && c \leq 1,\ m = 3 \\
& (c+1, m, 0) : && c \leq 2,\ m = 3 \\
& (c+1, m+1, 0) : && c = m = 1 \\
& (c, m+1, 0) : && m \leq 2,\ c \leq m+1 \\
& (c, m+2, 0) : && m \leq 1,\ c \leq m+2
\end{aligned}
$$

another way to define F, is explicitly define adjacency list:

(3,3,0) → [(2,3,1), (2,2,1), (1,3,1)]
(2,3,1) → [(3,3,0)]
(2,2,1) → [(3,3,0), (2,3,0)]
(1,3,1) → [(3,3,0), (2,3,0)]
(2,3,0) → [(2,2,1), (1,3,1), (0,3,1)]
(0,3,1) → [(2,3,0), (1,1,0)]
(1,1,0) → [(0,3,1), (2,2,0)]
(2,2,0) → [(1,1,0), (2,0,1)]
(2,0,1) → [(2,2,0), (3,0,0)]
(3,0,0) → [(2,0,1), (1,0,1)]
(1,0,1) → [(2,0,0), (1,1,0)]
(2,0,0) → [(1,0,1), (0,0,1)]
(1,1,0) → [(1,0,1), (0,0,1)]

b.



we can notice that the goal state exists at the deepest level so we can run DFS and find an optimal solution

In this graph we run BFS (not DFS) so ~~the~~ BFS expands every node before reaches to the goal state, BFS always returns an optimal solution.

c. No, because the adjacency list of each node does not contain any illegal ~~m~~ state (c overnumbered m) and the search algorithm depends on these lists because it explores only the node adjacents ~~o~~, ~~not every po~~, and our transition function takes this into consideration in every move.

Informed Search

① True

A^* search will ~~explore~~ expand every node n ~~such~~ that $f(n) < C^*$ and some nodes where $f(n) = C^*$.

$f(n) = g(n) + h(n)$, C^*: cost of goal state

this can be written as: $h(n) < C^* - g(n)$

So : $\forall n : h_1(n) \leqslant h_2(n) < C^* - g(n)$ ~~A^* uses~~

~~A^* uses h_1 will expand every node will expand~~

if A^* with h_2 expands node n then:

① $h_2(n) < C^* - g(n) \Rightarrow h_1(n) < C^* - g(n)$

and A^* with h_1 will also expand this node

② $h_2(n) = C^* - g(n) \Rightarrow h_1(n) \leqslant C^* - g(n)$

if $h_1(n) < C^* - g(n)$ A^* with h_1 will ~~surely~~ definitely expand this node, else, $(h_1(n) = C^* - g(n))$ it may or may not expand it

In general the statement is true except possibly for some nodes with $f(n) = C^*$

② h_1 is admissible

Every misplaced tile needs at least one move to fit in its correct place, so number of actual needed moves will never over estimate the number of misplaced tiles.

h_2 is admissible

every misplaced tile needs to move Manhattan Distance steps at least in order to get to its correct place, so the actual cost of moving every tile to its right place will never ~~overestimate~~ be less that the sum of Manhattan distances.

h_3 is admissible

every tile that is just out of row or out of column needs to be moved at least once, and every tile that is both out of row and out of column must be moved at least twice, so h_3 will never overestimate the actual number of moves needed to solve the puzzle.

Optimization\Local search

a. we can represent first point by $g(V_1,V_2) = ||V_1|-|V_2||$

and ~~th~~ the second part by $h(V_1,V_2) = |\{(v_1,v_2)\in E \mid v_1\in V_1, v_2\in V_2\}|$

so a function can be $f(V_1,V_2) = g(V_1,V_2) + h(V_1,V_2)$

b. We can start in ~~a~~ a state where all vertices in V are in one grupe i.e. $V_1 = V$ $V_2 = \emptyset$, and the ~~neigh~~ neighbors of each state will be the addition of $v\in V_1$ into V_2 (and the removal of said v from V_1) and all the avilable swap

i.e. $v_1\in V_1$ $v_2\in V_2$ $\Rightarrow$ $V_1 = (V_1\setminus\{v_1\})\cup\{v_2\}$, $V_2 = (V_2\setminus\{v_2\})\cup\{v_1\}$.

c. I expect gradiant ascent to run faster, but for simulated anhealing to get a more optimal solution. since s.a can overcome local minima of $f(V_1,V_2)$ with the use of intredusing a mesgure of randomness.

d. we can mark $|V| = n$, $V = \{v_1, \ldots, v_n\}$

then each individual will be vector in $\{0,1\}^n$

where $e_i = 0$ if $v_i \in V_2$ and $e_i = 1$ if $v_i \in V_1$

we'll use $f(V_1, V_2)$ from a as fitness function.

e. we can difine mutation as the random

chane that sends $e_i = 0 \to 1$ $e_i = 1 \to 0$

and crossover is if a, b are indevviduals

then $a = (a_1, \ldots, a_n)$ $b = (b_1, \ldots, b_n)$ then c

the cross over $c = (a_1, \ldots, a_j, b_{j+1}, \ldots, b_n)$ for $1 \le j \le n$

f. while iterative improvment will probbly

run faster if we want more close to

optimal solution will want to use genetic

algorithem since it is less likly to get stuck

at local min then iterative improvment is.

CSP

let E be all ~~a~~ the lines and V all stations

and let $X = \{x_1, \dots, x_n\}$ all the trains, will mark

$X_i = \{x^1_{i,1}, x^1_{i,2}, x^2_{i,1}, x^2_{i,2}, \dots, x^k_{i,1}, x^k_{i,2}\}$ where k is the number of stations in the path of train i. $x^j_{i,1}$ is the arivol to station j and $x^j_{i,1}$ is the diparture.

note that for every $j \neq 1$

$x^j_{i,1}$ is just equal $x^{j-1}_{i,2} + x^{j-1}_i$ where x^{j-1}_i is the time from station $j-1$ for train i to next target j in her route. so x is the variables. ($x^1_{i,1}$ is just the starting time of the train)

the domaine for $x^j_{i,2}$ is any time ~~number that~~ that follows $x^j_{i,2} \geq 1_{min} + x^j_{i,1}$, that is the domain

(CSP)

~~e_1~~ constrains are

let e_i_trains $= \{(x_1, y_1), \dots (x_k, y)\}$

where x_j is the number of train that pass through e_i and y_j is the index of the stop before the pass then $A_{e_i} = \{(y_j, y_h) \mid j \neq h\}$

and ther meaning is that if t is the time requred for e_i then $(X^{y_j}_{x_j,2} - t, X^{y_j}_{x_j,2} + t) \not\ni X^{y_h}_{x_h,2}$

then $A = \bigcup_{e \in E} A_e$

B_{v_i} v_i_max = m number of trains allowd in the station B_{v_i} v_i_trains $= \{(a_1, b_1), \dots\}$

whe a_j number of the train and b_j index of the station in train a_j then

$B_{v_i} = \{(b_{j_0}, b_{j_1}, \dots, b_{j_m}) \mid j_0 < j_1 < \dots < j_m\}$

and the meaning is tat the is no j_k $0 \le k \le m$ that for him $\forall_{h \ne k}$ $(X^{b_{j_k}}_{a_{j_k},1}, X^{b_{j_k}}_{a_{j_k},2}) \cap (X^{b_{j_h}}_{a_{j_h},1}, X^{b_{j_h}}_{a_{j_h},2}) \neq \emptyset$

so $B = \bigcup_{v \in V} B_v$

so the constraints are $A \cup B$.

a good heuristics will be least containing value since there are a lot of constrains there is a need to reduese back trackig since a lot of steps will requere that.