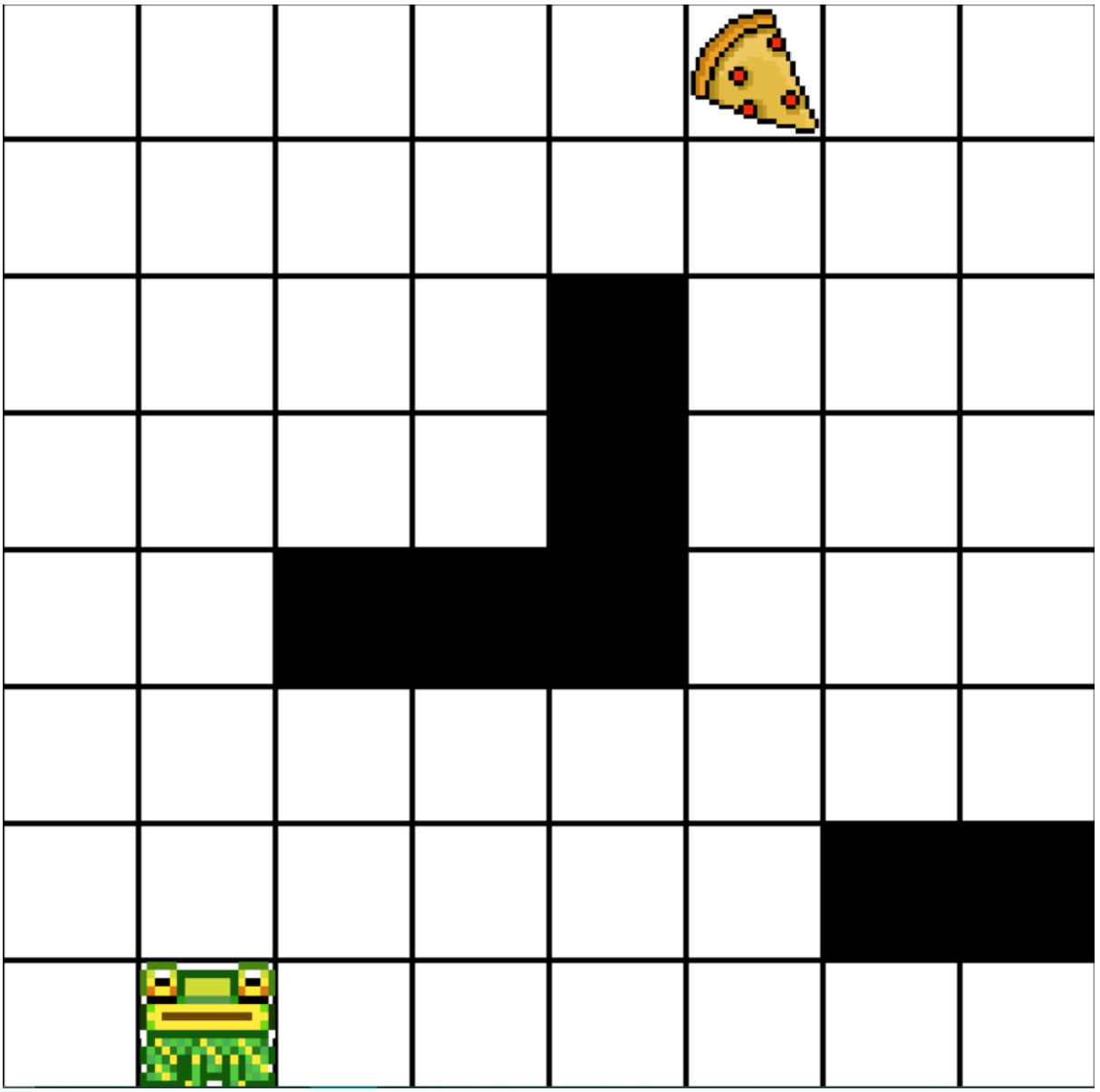
Intel·ligència Artificial

Pràctica 1



Lluís Picornell Company
43477189J
Xavier Vives Marcus
43462573W

Cerca desinformada

Funcionament

La cerca no informada d'aquesta pràctica ha estat implementada com una cerca per amplada. Ens hem basat en l'exercici de les quiques fet a classe.

Cerca

A partir de l'estat inicial generam un conjunt de fills, i els ficam a la llista de nodes oberts. Miram si algun d'ells és l'estat meta. Si no ho son, generam els seus fills i ficam els pares a la llista de nodes tancats per no haver-los de visitar més d'una vegada.

Continuem així de manera iterativa fins a trobar l'estat meta o no tenir més estats per explorar.

En acabar aquesta exploració tornam una llista d'accions fruit de tots els nodes pare que han duit fins al node solució.

Actúa

Generem l'estat a partir de les percepcions que tenim, cridem a cerca i anam retornant les accions que ens torna cerca per ordre.

Rendiment

Com es tracta d'un algorisme no informat, no cerca la millor solució sino que retorna la primera solució que troba, el cost no sol ser òptim.

Pel que fa al temps d'execució, la següent taula mostra el temps que ha tardat en vàries execucions.

N. execució	Temps (s)
1	11.002
2	13.004
3	3.991
4	6.044
5	14.305
6	12.335

7	9.994
8	10.987
9	11.005
10	17.318
mitjana	10.998500
desviació estàndard	3.843127

És considerablement més lent que A* ja que exploram nodes que no duen a la solució.

Algorisme A*

Funcionament

Cerca

Per tal de fer la cerca amb l'agent A Estrella, ens hem basat en l'algorisme A* utilitzat al programa de "monedes" vist a classe.

És parescut a la cerca no informada però, a diferència d'aquesta, només explora els nodes que considera "bons" això es fa fent que la llista de nodes oberts sigui una cua de prioritat fent que l'indicador de la prioritat sigui la suma de l'heurística amb el cost que ha suposat arribar fins aquell node.

La resta és pràcticament igual que a la cerca no informada. Generam un conjunt de fills de l'estat en millor heurística i l'afegim a la llista de nodes tancats, afegim els fills a la llista de nodes oberts i miram si algun d'ells és l'estat meta. Si no ho és, generam els fills del "millor" estat (fent ús de la prioritat que tengui dins la llista d'oberts). Això es repeteix fins a trobar l'estat meta o no tenir més estats per explorar.

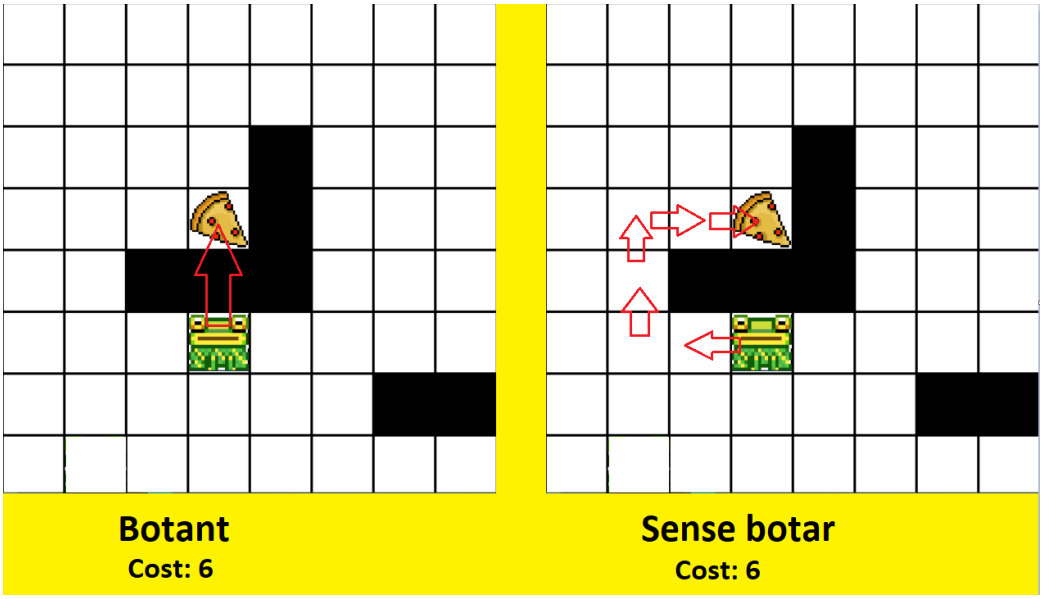
En acabar aquesta exploració tornam una llista d'accions fruit de tots els nodes pare que han duit fins al node solució.

Actúa

Generem l'estat a partir de les percepcions que tenim, cridem a cerca i anam retornant les accions que ens torna cerca per ordre.

Rendiment

És important notar que en aquest algorisme pràcticament mai es realitza l'acció de botar ja que no surt rentable tenint en compte el seu cost. Pràcticament en totes les ocasions que es pot botar surt més o igual de rentable fer el camí caminant.



Pel que fa al temps d'execució, la següent taula mostra el temps que ha tardat en vàries execucions.

N. execució	Temps (s)
1	4.997
2	9.9985
3	8.992
4	10.001
5	2.994
6	10.999
7	5.993
8	4.994
9	10.997
10	4.024
mitjana	7.398950
desviació estàndard	3.209004

Com és d’esperar, obtenim temps d’execució menors que a la cerca no informada ja que només exploram molts menys nodes.

Algorisme minimax

Funcionament

L'algorisme de minimax implementat segueix l'estructura que ens va ser proporcionada en pseudocodi en el document minimax.txt.

L'algorisme consisteix en donat un estat que conté dos jugadors, intentar prediure el que fera el contrari i elegir el millor moviment tenint en compte aquest fet.

Minimax

Per això mitjançant una heurística, en el nostre cas la diferencia de distància de Manhattan entre els jugadors i la meta, elegim quin és el millor moviment, però tenint en compte que el contrari també farà el seu millor moviment, per el que nosaltres haurem de fer el millor moviment tenint en compte la seva elecció. Aquest procés és recursiu i pot tenir tanta profunditat com un desitgi, tenint en compte que el cost és exponencial.

Per tal de dur a terme es va haver de modificar lleugerament la classe Estat.
En aquesta nova versió un estat es defineix per una percepció, una posició, el seu nom i el nom de min.

Estat

La majoria de mètodes han seguit igual, els que han canviat són:

es_legal

Ara també té en compte si es troba damunt de l'altre "rana".

calcular_distanciaManhattan

Aquest mètode no existeix a l'altra versió d'Estat.
Calcula la diferència de distàncies de Manhattan dels dos jugadors respecte a la meta.

Com més gran és el resultat millor és per max, un resultat negatiu vol dir que Min va guanyant.

La resta continuen igual a excepció de petits canvis de nomenclatura i sintaxi.

Rendiment

No hem aconseguit que els agents trobin un 100% de les vegades la solució. Hi ha casos en els que els agents actuen de forma extranya amb les parets i queden atrapats.

Tenen un percentatge d'acert d'un 90%.

Pel que fa al temps d'execució, la següent taula mostra el temps que ha tardat en vàries execucions.

N. execució	Temps (s)
1	2.204
2	3.095
3	2.130
4	2.092
5	5.139
6	2.148
7	4.152
8	2.156
9	1.562
10	1.521
mitjana	2.619900
desviació estàndard	1.195304

És l'algorisme més ràpid i que té les dades menys dispersades. Això és completament lògic tenint en compte que com hi ha dues "ranes", la probabilitat de que alguna d'elles apareixi just davora la pizza és bastant superior fet que fa que amb pocs moviments ja acabi el joc.

Algorisme genètic

Funcionament

Cerca

Generam una població de ‘n’ individus i els posam dins d’una cua de prioritats per així tenir-los ordenats en funció de com d’aprop estan de la solució (funció *fitness*).

Llavors, apartam els ‘m’ millors individus i comprovam si algun d’ells ha trobat la solució. En cas de que l’haguin trobat, retornam la solució que ha trobat aquest individu.

En cas contrari, feim que es reprodueixin entre ells i generam una nova població de ‘n’ individus formada per la descendència dels ‘m’ millors individus de la generació anterior. Repetim fins trobar una solució.

Individu

Definició

Un individu compté un llistat d’accions, la posició inicial i la posició final.

- self.__accions
    - Llistat de les accions que ha realitzat un individu
- self.__pos_inicial
    - Indica on ha aparegut la “rana” abans de fer cap moviment.
    - Útil per comprovar si un individu és legal
- self.__pos_final
    - Casella resultant d’aplicar totes les accions partint de la posició apuntada per self.__pos_inical.
    - Atribut útil per poder calcular la seva funció *fitness* i comprovar si un moviment és legal

Funció fitness

Per calcular la funció fitness de cada individu, sumam la distància a la pizza (té un pes de 80%) i la quantitat de moviments que ha tardat en arribar-hi (té un pes del 20%). D’aquesta manera penalitzam els individus que arriben a la solució però fent moltíssims moviments i afavorim els individus que arriben a la solució amb pocs moviments.

fitness = 80*(distància_manhattan) + 20*(n_moviments)

Generació de moviments

Genera fins a ‘c’ accions per un individu.

El funcionament és el següent:
1.- Genera aleatòriament una acció, comprova si el resultat és legal, si ho és l’afegeix. En cas contrari no l’afegeix.

2.- Comprova si ha arribat a la solució. Si ha arribat → deixa de generar accions. Si no ha arribat → torna al pas 1.

El bucle es repeteix fins a ‘c’ vegades

Funció offspring

Rep dos individus i genera ‘d’ fills.

Per cada fill, decideix aleatòriament quin és el pare i quin és la mare (dels individus que ha rebut la funció) els moviments del fill estan formats per els ‘a’ primers moviments del pare i els ‘b’ moviments de la mare. Tant ‘a’ com ‘b’ son generats aleatòriament.

Finalment, cada fill té una probabilitat de l’1% de mutar. Els fills mutants que donen lloc a solucions no legals son descartats.

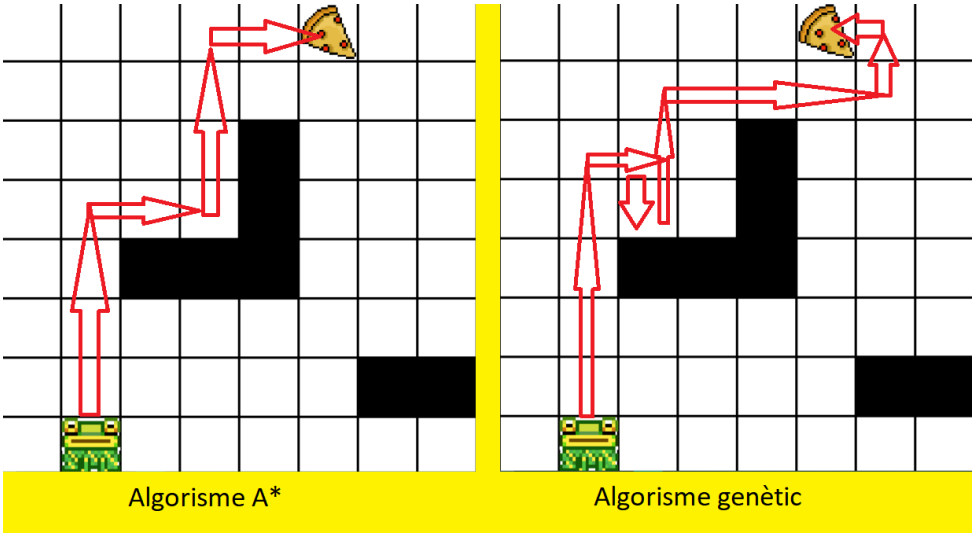
Funció muta

Sel·lecciona aleatòriament una de les accions de l’individu i la canvia per una altre acció també generada aleatòriament.

Rendiment

A diferència dels altres algorismes, l'algorisme genètic no sempre troba el millor camí fins a la solució i de vegades les solucions que troba fan varis moviments innecessaris.

A continuació es compara la solució trobada per l'algorisme A* i sa solució generada per l'algorisme genètic.



Pel que fa al temps d'execució, la següent taula mostra el temps que ha tardat en vàries execucions.

N. execució	Temps (s)
1	12.005
2	11.997
3	3.997
4	23.004
5	10.997
6	4.998
7	12.006
8	5.996
9	18.003
10	8.993
mitjana	11.1996
desviació estàndard	5.901836

El que més penalitza el temps d'execució son les solucions que realitzen varis moviments innecessaris com és el cas de les execucions 4 i 9 i per això se'ns dispara el valor de la desviació estàndard sent l'algorisme amb les dades més dispersades.