Grau en Enginyeria Informàtica

Intel·ligència Artificial

Pràctica 1

Lluis Barca Pons
lluis.barca1@estudiant.uib.es

Victor Canelo Galera
victor.canelo1@estudiant.uib.es

20 de novembre de 2022

1. Introducció

Des de l'assignatura d'Intel·ligència Artificial se'ns ha demanat programar amb el llenguatge de programació Python, un seguit d'algorismes que efectuen una cerca en concret. Específicament es tracta de programar els diferents algorismes perquè un agent, en aquest cas una granota, trobi una porció de pizza que es troba a un tauler; el nostre entorn.

Per dur a terme aquesta pràctica, s'han aplicat els diferents conceptes apresos a classe de teoria en relació amb: els algorismes de cerca i les seves diferents característiques, i els conceptes d'agent, percepcions i accions; que són les bases de la intel·ligència artificial.

En referència a l'anàlisi del rendiment dels algorismes, s'avaluaran tant el cost el temps com en memòria per determinar quin dels quatre algorismes és "millor"o "pitjor". Per ambdós variables, és dura a terme un total de 10 execucions i es registraran els valors obtinguts:

- **Temps**: Hem utilitzat la llibreria *time* de Python, on amb el mètode $perf\_time()$ hem pogut calcular el temps que tarda la cerca a fer-se. Aquest valor està representat en mil·lisegons, pel simple fet de facilitar la comparació a simple vista.
- **Memòria**: Hem contat el nombre de fills que genera cada algorisme.

Finalment, hem calculat la respectiva mitjana per treure un valor final. Llavors, comparant aquests valors finals podrem treure conclusions de quin algorisme és més o menys costos en qüestió tant de temps com de memòria utilitzada.

2. Algorismes

2.1. Cerca no informada per amplada

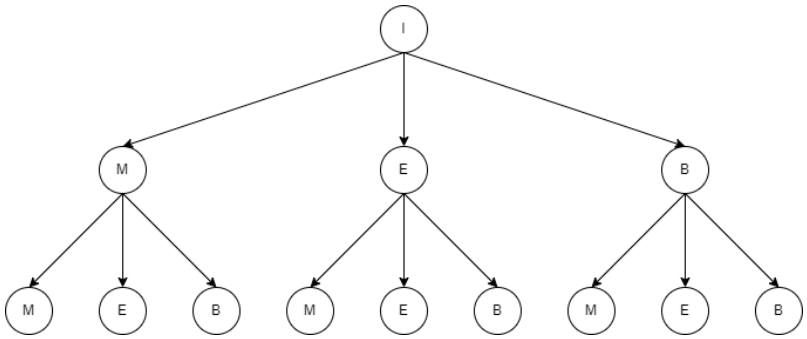
2.1.1. Anàlisis

Aquest algorisme de cerca comença amb dues llistes d'estats. Concretament, una llista d'estats "oberts", que representa els estats per explorar i una llista d'estats "tancats", que representa els estats ja explorats.

A continuació, mentre tenim elements dins la llista d'oberts, el programa anirà avaluant si l'estat actual (el primer de la llista d'oberts en cada iteració), ja s'ha explorat o si no és vàlid; és a dir, fem un clàssic tractament d'errors. Seguidament, es generen els estats fills corresponents i s'avalua si l'estat actual ja és la solució, és a dir, si l'agent ha aparegut justament a la casella objectiu. La següent acció de la iteració seria emmagatzemar els estats fills generats en la llista d'estats oberts (per poder explorar-los) i finalment afegir l'estat actual (que ja hem avaluat) a la llista d'estats tancats.

Finalment, si es troba una solució, s'emmagatzema el conjunt d'accions des de la fulla de l'arbre fins al node arrel. Llavors, aquesta serà la nostra solució.

Cal comentar que en l'execució d'aquest algorisme ens vam adonar que l'agent només botava; la qual cosa ens va semblar estrany. Això es deu al fet que, com es mostra en l'arbre de decisions següent, quan fem la cerca per amplada, l'algorisme avalua la millor opció a cada nivell de l'arbre. Això significa que no té cap perspectiva i sempre que pot, elegeix botar, que és el més ràpid per arribar a la porció de pizza; si no tenim en compte el cost, com és el cas.



2.1.2. Rendiment

A continuació les dades recollides:

	Fills generats	**Temps (ms)**
1	97	2,9981
2	246	0,8099
3	227	0,7167
4	110	0,8621
5	49	2,4388
6	98	3,0289
7	67	1,3413
8	83	1,3708
9	99	0,3724
10	96	0,4764
Mitjana	**117**	**1,4415**

Llavors, podem dir que aquest algorisme de cerca genera una mitjana de 117 fills per execució i ho fa en 1,4415 mil·lisegons.

2.2. Cerca informada per A*

2.1.1. Anàlisis

Aquest algorisme de cerca és pràcticament igual que el de cerca per amplada, però amb lleugers canvis que es comentaran a continuació.

Realment observem dos canvis. Per una banda, la llista d'estats oberts ara és una cua de prioritat, i això ens permetrà (juntament amb el segon canvi) establir prioritats i fer una cerca realment informada. El segon canvi és el fet d'incloure el càlcul del cost de cada estat. Aquest càlcul es du a terme sumant el mateix pes del moviment més l'heurística; que en el nostre cas és calcular la distància entre l'agent i la posició de meta.

Llavors, amb aquests dos canvis aconseguim un algorisme òptim, comparat amb la cerca no informada; ho comprovarem en el següent apartat.

2.1.2. Rendiment

A continuació les dades recollides:

	Fills generats	**Temps (ms)**
1	48	1,9899
2	77	0,6519
3	114	1,7425
4	130	0,2064
5	74	0,5979
6	45	0,7140
7	91	0,8330
8	103	1,2543
9	75	0,9988
10	31	0,1988
Mitjana	**79**	**0,9188**

Llavors, podem dir que aquest algorisme de cerca genera una mitjana de 79 fills per execució i ho fa en 0,9188 mil·lisegons.

2.3. Cerca per Min i Max

2.3.1. Anàlisis

Aquest algorisme de cerca comença calculant les respectives puntuacions dels agents. Això ens permetrà anar comparant-los en cada estat i poder saber quin dels dos és el guanyador.

Seguidament, avaluem si aquest estat és el final o si la profunditat de l'arbre és massa gran; en cas que es complís qualsevol de les dues condicions, es retornaria la puntuació i l'estat actual (i significaria que hem acabat). Seguidament, en cas de no complir-se les condicions anteriors, es generen els estats fills corresponents per a la seva posterior avaluació.

Finalment, depenent de si el torn actual és de MAX o de MIN, realitzarem la maximització o minimització de la puntuació de l'estat fill. Amb això obtindrem quin dels dos agents ha guanyat.

2.3.2. Rendiment

A continuació les dades recollides:

	Fills generats	**Temps (ms)**
1	41	4,3486
2	96	3,7194
3	70	4,9102
4	143	4,7258
5	72	3,9856
6	26	2,7442
7	28	4,1675
8	67	4,3904
9	69	3,1194
10	143	3,2266
Mitjana	**76**	**3,9338**

Llavors, podem dir que aquest algorisme de cerca genera una mitjana de 76 fills per execució i ho fa en 3,9338 mil·lisegons.

3. Conclusió

Aquesta pràctica ens ha servit per posar en pràctica els coneixements que hem anat adquirint durant les classes de teoria, de forma més pràctica. Ens ha servit per posar-nos en una situació més propera a la realitat a l'hora de pensar un problema en termes d'agents.

Durant la pràctica hem tingut alguna sèrie de problemes a l'hora d'implementar l'algorisme de Min i Max; sobretot a l'hora de calcular les puntuacions dels respectius agents segons l'estat anterior. Això era crucial, ja que aquestes puntuacions ens permetien comparar quin dels dos agents anava guanyant i quin perdent. D'altra banda, no hem aconseguit acabar d'implementar l'algorisme genètic, però hem deixat tota la implementació que hem realitzat.

Finalment, la comparació del rendiment dels diferents algorismes implementats:

$$Ratio\_Amplada = \frac{1}{Memoria \cdot Temps} = \frac{1}{117 \cdot 1,4415} = 5,929 \cdot 10^{-3}$$

$$Ratio\_AEstrella = \frac{1}{Memoria \cdot Temps} = \frac{1}{79 \cdot 0,9188} = 13,777 \cdot 10^{-3}$$

$$Ratio\_MiniMax = \frac{1}{Memoria \cdot Temps} = \frac{1}{76 \cdot 3,9338} = 3,344 \cdot 10^{-3}$$

Aquesta equació ens permet establir una relació entre la memòria utilitzada i el temps requerit per a cada algorisme. Per tant, a major ràtio, major rendiment; la qual cosa ens indica que l'algorisme més eficient és l'implementat per A*.