Uporaba umetne inteligence za zmago igre 2048

Seminarska naloga pri predmetu Osnove umetne inteligence

Sebastian Mežnar

11. december 2018

2048 je preprosta računalniška igra v kateri združuješ skupaj ploščice z istimi števili in iz njih narediš ploščo, na kateri je njun seštevek. Cilj te igre je narediti ploščico z številom 2048. V nalogi sem to igro poiskušal zmagati s pomočjo hevrističnih preiskovalnih algoritmov.

1 Uvod

Umetna inteligenca se dandanes uporablja na raznih področjih. Področja kot so medicina, logistika, statistika, ekonomija,... so zaradi nje zelo napredovala. Uporablja se jo pa lahko tudi za lažje, bolj preproste in zabavne stvari kot na primer igre.

S pomočjo umetne inteligence lahko igre igramo na različne načine, kot na primer s preiskovanjem, z ocenjevanjem kako dobra bo naslednja poteza (s pomočjo nevronske mreže), s statistično izbiro naslednje poteze, ... V nalogi sem se odločil, da bom uporabil hevristično preiskovanje za iskati naslednjo potezo, pri tem pa sem dal programu minimalno znanje o igri in le nekaj preprostih navodil.

2 O igri

2048 je odprtokodna igra, dostopna na internetnem naslovu https://github.com/gabrielecirulli/2048. Cilj te igre je sestaviti ploščico s številom 2048 iz manjših ploščic. Igralec lahko v vsaki potezi ploščice potisne v eno izmed štirih smeri (gor, dol, levo, desno), ob tem pa se zaletijo ob sosednje in se združijo, če je na dveh sosednjih isto število. Po vsaki potezi se na igralnem polju pojavi nova ploščica z številom 2 oziroma 4. Igra se konča, ko igralec ne more opraviti nobene poteze več.

Rezultat se v igri računa tako, da se vsakič, ko igralec dve ploščici združi, rezultatu prišteje vrednost nove ploščice. Igra se najpogosteje igra na polju velikosti 4 x 4. Za polje take velikosti lahko naredimo največ ploščico z vrednostjo 131972, največji možen rezultat pa je 3,932,156.

3 Ideja

V igri se ploščice na prazna polja postavljajo naključno, zato moramo uporabiti hevristični algoritem za iskanje naslednje poteze. Po raziskovanju po spletu sem ugotovil, da večina ta

problem reši tako, da poda programu omejitve kot naprimer, da se morajo ploščice z visokimi števili držati skupaj in ostati čim bolj v kotu, saj tako ostane več polj za druge ploščice. Ker sem želel, da program ne vsebuje znanja o igri in taktik, sem se namesto tega odločil za preprostejši pristop.

Glavna ideja algoritma, ki sem ga uporabil je, da če v vsako smer odigraš veliko število naključnih iger (prva poteza v željeno stran, vse naslednje pa naključno dokler se igra ne konča), se napake izenačijo in lahko ugotovimo, katera smer je za nas najbolj ugodna. Tako ob vsaki potezi algoritem odigra določeno število naključnih iger, izmed katerih vsaka vrne končni rezultat. Nato se iz teh rezultatov izracuna povprečje za vsako smer ter izbere poteza, pri kateri je povprečni rezultat najvišji.

4 Rezultati.

Testiral sem več konfiguracij, ki sem jih ločil glede na število iger med potezami. Poleg tega sem preizkusil tudi igre z naključnimi potezami. Za vsako konfiguracijo sem odigral več iger ter dobil najnižji, najvišji ter povprečen rezultat čez vse igre, poleg tega pa procent iger, kjer je bila najvišja ploščica nad 2048 oziroma 4096. Rezultate sem prikazal v tabeli 1.

Tabela 1: Število iger in uspeh

število iger	najnižji/povprečni/najvišji rezultat	procent iger nad 2048/4096	odigranih iger
naključna poteza	232/1085.31/2760	0/0	100
1	488/4354.4/12048	0/0	150
10	1316/14746.96/35640	19/0	100
100	11632/38468.33/77700	91/24	100
1000	71620	100/100	1

Podajanje rezultati naj bo primerno strukturirano. Če ima naloga več podnalog, uporabi podpoglavja. Če bi želel poročati o rezultatih izčrpno in pri tem uporabiti vrsto tabel ali grafov, razmisli o varianti, kjer v tem poglavju prikažeš in komentiraš samo glavne rezultate, kakšne manj zanimive detajle pa vključite v prilogo (glej prilogi A in B).

5 Viri in literatura

2048 Wikipedia (10.12.2018)

6 Izjava o izdelavi domače naloge.

Domačo nalogo in pripadajoče programe sem izdelal sam.

Priloge

A Podrobni rezultati poskusov.

Če je rezultatov v smislu tabel ali pa grafov v nalogi mnogo, predstavi v osnovnem besedilu samo glavne, podroben prikaz rezultatov pa lahko predstaviš v prilogi. V glavnem besedilu ne pozabi navesti, da so podrobni rezultati podani v prilogi.

B Programska koda.

Za domače naloge bo tipično potrebno kaj sprogramirati. Če ne bo od vas zahtevano, da kodo oddate posebej, to vključite v prilogo. Čisto za okus sem tu postavil nekaj kode, ki uporablja Orange (`http://www.biolab.si/orange`) in razvrščanje v skupine.

```
import random
import Orange

data_names = ["iris", "housing", "vehicle"]
data_sets = [Orange.data.Table(name) for name in data_names]

print "%10s␣%3s␣%3s␣%3s" % ("", "Rnd", "Div", "HC")
for data, name in zip(data_sets, data_names):
    random.seed(42)
    km_random = Orange.clustering.kmeans.Clustering(data, centroids = 3)
    km_diversity = Orange.clustering.kmeans.Clustering(data, centroids = 3,
        initialization=Orange.clustering.kmeans.init_diversity)
    km_hc = Orange.clustering.kmeans.Clustering(data, centroids = 3,
        initialization=Orange.clustering.kmeans.init_hclustering(n=100))
    print "%10s␣%3d␣%3d␣%3d" % (name, km_random.iteration, \
    km_diversity.iteration, km_hc.iteration)
```