CELTRA: Jackpot – koliko denarja priigra tvoj algoritem?

Izdelal: Klemen Gantar

Če to berete, se vam zahvaljujem za razumevanje in se še enkrat opravičujem za rahlo zamudo in nepopolno oddajo pdf-ja.

link do repozitorija:

https://onedrive.live.com/redir?resid=AEC8FCFC1D9A907C!4969&authkey=!AGLLuKD1h7taaQ&ithint=folder%2c

navodila za zagon kode,

//update

Za lažji zagon sem na repozitorij naložil še datoteko dist, ki vsebuje izvršno datoteko celtra.exe in vse potrebne module in datoteke. Tako ni več potrebe za namestitev pythona in dodatnih modulov, kot je opisano spodaj. Prenesite datoteko dist in v cmd-ju iz nje zaženite celtra.exe z ukazom: "celtra.exe vhodni_url"

//

Za zagon je potrebna distribucija Pythona (sam uporabljam verzijo 2.7 vendar bi moralo delovati tudi na novejših). Poleg osnovne verzije Pythona algoritem uporablja programski modul SciPy. Če vaša distribucija ne vsebuje modula, ga lahko namestite skupaj z distribucijo Anaconda na spletnem mestu http://continuum.io/downloads.
Program lahko zaženete kot klasični python program. Tu naj še enkrat omenim, da mora vaša različica pythona vsebovati modul ScyPy, sicer program nebo deloval.

Primer zagona programa iz ukazne vrstice: "python celtra.py vhodni_url"

opis ciljev, pristopa k reševanju in rezultatov

Glavni cilj naloge je bila preprosta implementacija čim bolj učinkovitega algoritma, ki zagotovi hitro in zanesljivo iskanje najboljšega avtomata. Za dosego tega cilja sem implementiral algoritem, ki ne upošteva zgolj izračunane napovedi verjetnosti avtomata, temveč avtomate izbira glede na verjetnost, da le-ti presežejo trenutno najbolšega. Pri tem upošteva napoved verjetnosti najboljšega avtomata in število zmag ter potegov posameznih avtomatov.Glavni element algoritma je torej

funkcija ki z enostranskim binominalnim testom določi kakšna je verjetnost, da posamezen avtomat preseže verjetnost trenutno najboljšega avtomata.

Zanesljivost algoritma sem izboljšal s preprostim trikom, ki avtomatom z manjšim številom zmag (0 ali 1 zmaga) števila zmag ne zmanjša za 1, kot je sicer storjeno v enostranskem binominalnem testu. Tako na začetku avtomatom z manjšim številom poskusov zagotovimo rahlo prednost, ki nam omogoča zanesljivejšo oceno verjetnosti vseh avtomatov in zmanjša upliv nesrečnih naključij, kateri lahko, če se pojavijo na začetku, močno zmanjšajo učinkovitost algoritma.

Drugi del naloge je bilo zaznavanje sprememb v verjetnosti. Ker sem zaradi drugih obveznosti in odlašanja z reševanjem izziva začel prepozno, sem imel za ta del malo časa in je zato veliko prostora za izboljšave. Za zaznavanje sprememb sem uporabil preprosto metodo, ki s pomočjo drsečega okna (uporabil sem fifo vrsto) spremlja spreminjanje trendov pri vsakem od avtomatov. Če je okno polno, primerjam izračunano verjetnost znotraj okna in skupno verjetnost avtomata.
Za ta namen se spet poslužim binominalnega testa, vendar je ta tokrat dvostranski in nam pove kakšna je verjetnost, da pri verjetnosti avtomata pride do razmerja zmag in porazov hranjenih v drsečem oknu. Če je ta verjetnost manjša od neke konstantno določene meje (0.001) pomeni, da je prišlo do spremembe.

Ko sem govoril o izboljšavah sem imel v mislih predvsem optimizacijo ali mogoče online prilagajanje dveh parametrov (velikost drsečega okna in meja za zaznavanje sprememb), ki sta uporabljena in odločilno vplivata na dosežen rezultat. Poleg tega bi lahko zaznavanje opravljal na več nivojih, kjer bi bila zaznana sprememba na višjem nivoju povod za dodatno analizo in potrditev spremembe na nižjih nivojih. Možnosti je ogromno a žal čas ni bil na moji strani.

Vseeno sem z rezultatom zadovoljen. Tudi naloga sama mi je bila zelo všeč zato upam, da bo tudi v prihodnje na voljo kaj podobnega.