CELTRA JACKPOT PLAYER

Izdelal: Tim Jerman (tim.jerman@gmail.com)

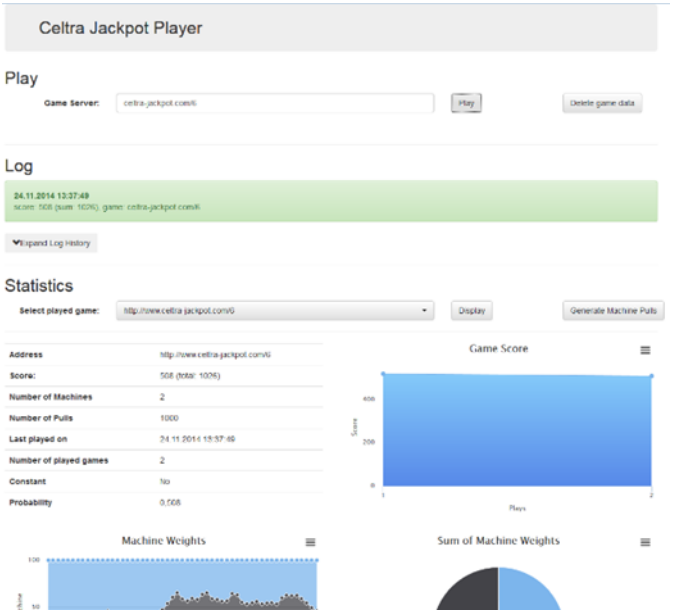
Git repository: https://github.com/timjerman/CeltraJackpotPlayer.git

Cilji: Aplikacija je nastala kot odgovor na Celtrin izziv *Jackpot - koliko denarja priigra tvoj algoritem*? Cilj je bil napisati algoritem, ki bo najhitreje in najnatančneje ocenil verjetnost nagrajevanja igralnih avtomatov ter igral na tiste, s katerimi bo prišel do največje končne nagrade. Glede na področje na katerem Celtra deluje, je bil dodaten, lastni, cilj umestiti algoritem v spletno aplikacijo.

Opis aplikacije in zagon: Spletna aplikacija je sprogramirana v okolju ASP.NET MVC. Postavljena aplikacija je dostopna na spletnem naslovu:

http://jackpotplayer-001-site1.smarterasp.net

Ker je povezava do strežnika počasna, bi prosil, če je le možno, da si aplikacijo z git repositorija postavite lokalno in s tem bistveno pohitrite prenos podatkov med Celtrinim jackpot api-jem in aplikacijo. Če bodo kakšne težave, me prosim kontaktirajte. Spletna stran vsebuje: 1) obrazec za vnos naslova igre in njen zagon ter izbris te igre iz baze podatkov; 2) loge, kjer so prikazane uspešne igre ter morebitne napake; 3) statistiko posamezne igre: trenuten dosežek, skupen dosežek, število igranj,… Hkrati so prikazani tudi grafi dosežkov skozi čas, uteži avtomatov za naslednjo igro in vsota teh uteži.



Slika 1: Spletna aplikacija Celtra Jackpot Player

Zagon aplikacije je mogoč na dva načina:

1) Preko uporabniškega vmesnika. V address input polje $('#address') se vstavi naslov strežnika brez http://www. (npr. celtra-jackpot.com/1) in zažene aplikacijo z gumbom Play $('#post-play'). Če je potreben rezultat igre, je ta po njenem zaključku shranjen v hidden iput-u $('.returned-value'). Če je potrebno ponovno zagnati igro od začetka, jo je potrebno pobrisati tako, da se ponovno vstavi naslov strežnika v polje address in klikne gumb Delete Game Data $('#post-delete).
2) Preko API-ja. Klic je podoben tistemu, ki se ga zahteva za poteg ročice pri Celtrinem API-ju. Address naj **ne** vsebuje **http://www.** Odziv zahtevka je pri zagonu igre rezultat igre ali pa ERR, če pride do napake. Pri zahtevku Delete pa je ta True/False, kar predstavlja uspešen/neuspešen izbris iz baze podatkov.

Tabela 1: Zahtevek za zagon in zbris igre

Ukaz	Zahtevek	Primer	Odziv
PLAY	/Play/{*address}	http://jackpotplayer-001-site1.smarterasp.net/Play/celtra-jackpot.com/1	# ali ERR
DELETE	/Delete/{*address}	http://jackpotplayer-001-site1.smarterasp.net/Delete/celtra-jackpot.com/1	True/False

Pristop k reševanju: Pri zamisli in razvoju algoritma je imela robustnost izbire končnega avtomata prednost pred hitrostjo njegove izbire. Torej, cilj je pri vsakem zagonu igre od začetka (izbris naučenih parametrov) vedno na koncu izbrati najboljši/e avtomat/e (odvisno od tega ali avtomati vsebujejo uniformne porazdelitve). Razlog za tako odločitev je variabilnost dosežka posameznega avtomata, ki je še posebej pomembna v primerih kjer imajo avtomati podobne porazdelitve. Primer te variabilnosti je prikazan v tabeli 2, kjer je prikazanih 5 ponovitev igranja posameznega konstantnega avtomata v 4. testnem primeru (celtra-jackpot.com/4). Isti avtomat prožimo vseh 10000 ponovitev. Kljub temu, da ima 3. avtomat najvišjo verjetnost, je končni rezultat igre za ta avtomat najvišji le v treh primerih od petih. Pri 4. ponovitvi ima celo prvi avtomat (3. najboljši po verjetnostih) najvišji rezultat igre.

Tabela 2: Rezultat igre za pet neodvisnih ponovitev proženja istega konstantnega avtomata skozi celotno igro.

Avtomat	1. ponovitev	2. ponovitev	3. ponovitev	4. ponovitev	5. ponovitev
1 (p=0.020)	204	216	181	**217**	194
2 (p=0.019)	176	186	195	195	182
3 (p=0.024)	**242**	**239**	229	206	**248**
4 (p=0.023)	226	233	**238**	209	231

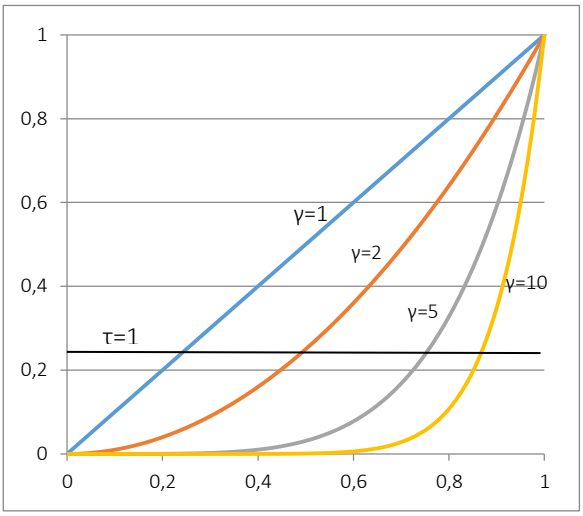
Zaradi variabilnosti, ki je prisotna pri vseh primerih lahko s prehitro odločitvijo izberemo napačen avtomat kar negativno vpliva na končni rezultat pri večjem številu zaporednih ponovitev.

Opis algoritma: Območje potegov (Pulls number) je razdeljeno na več segmentov. Za vsak segment *s* ima vsak avtomat *i* utež *w*, ki določa verjetnost izbire tega avtomata na tem segmentu. $\sum w_i(s) = 1$. Za prvo igro so uteži za vse avtomate enake $w_i=1/N$, kjer je *N* število avtomatov. Po vsakem koncu igre se uteži na novo poračunajo glede na število potegov avtomata in število zadetkov. Le ti z igranjem postajajo natančnejši. S časom se uteži skalirajo tako, da imajo boljši avtomati višjo verjetnost, tistim z nizko verjetnostjo se pa uteži izničijo. V vsaki igri avtomate naključno izbiramo glede na njihovo utež. Avtomat z višjo utežjo ima večjo verjetnost da je izbran.

Izbira segmentov: Začetni segmenti se pri prvi igri dinamično nastavijo glede na verjetnost avtomatov. Višja kot je verjetnost in več kot je zadetkov manjši so segmenti. Nasprotno so segmenti večji za nižje porazdelitve s čimer zagotavljamo višjo natančnost. Dinamična izbira segmentov npr. omogoča to, da se 9. testni primer obravnava kot konstantni, kljub temu da ni naveden kot tak, ker je na nekem območju verjetnost nič za vse avtomate. Za nekonstantne avtomate se po določenem številu zadetkov število segmentov poveča, zaradi česar natančneje izberemo meje verjetnosti.

Test konstantnosti: Ali so avtomati v igri konstantni napovedujemo s kumulativnim testom Kolmogorov-Smirnov. Za konstantne avtomate na izbiro uteži dodatno vpliva globalna verjetnost posameznega avtomata.

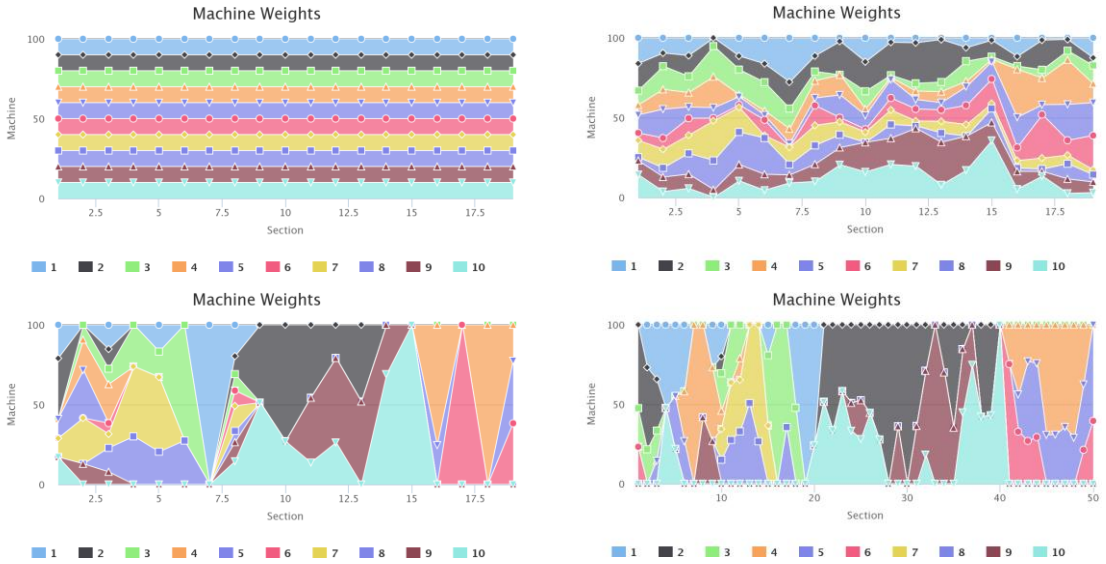
Izračun uteži: Po kočani igri so začetne uteži $w_i(s)$ izračunane kot razmerja med zadetki in potegi na segmentu *s*. Na posameznem segmentu uteži normaliziramo med 0 in 1 glede na najvišjo utež. Uteži potenciramo s potenco $\gamma \geq 1$ in najnižje porežemo z upragovljanjem s pragom τ (glej sliko 2). Potenca γ se povečuje sorazmerno s številom igranj in verjetnostjo avtomata. Izbira končnega avtomata je hitrejša pri višjih verjetnostih avtomatov. Z višjimi potencami γ se slabše uteži bolj odmikajo od najvišje. Hkrati lahko padejo pod



Slika 2: Skaliranje in upragovljanje uteži glede na število igranj in verjetnost avtomatov.

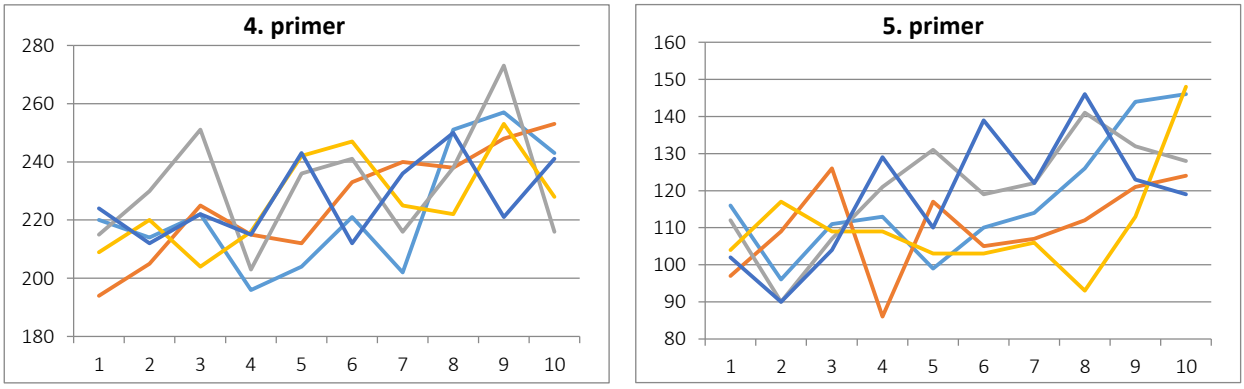
prag τ, zaradi česar se postavijo na nič. Pri konstantnih avtomatih uteži dodatno popravljamo glede na verjetnost skozi vse segmente. Če se verjetnost avtomata dovolj razlikuje od preostalih ga izberemo. S številom iger se znižuje prag za izbiro najboljšega avtomata.

Rezultati: Primer spreminjanje uteži prikazuje Slika3, kjer so prikazane uteži za 1., 3., 5. in 7. igro testnega primera 10 z desetimi avtomati. Na začetku imamo 19 segmentov in enake uteži za vse avtomate. Skozi igre se zvišujejo uteži tistim avtomotom, ki več zadanejo. Po sedmih igrah se je povečalo tudi število segmentov (4. graf). Na koncu lahko opazimo 3 različna območja. Na vsakem območju ima več različnih avtomatov neničelne uteži kar po tolikšnem času pomeni, da imajo isto verjetnost. Zaradi tega je vseeno katerega izmed njih se uporablja.



Slika 3: Spreminjanje uteži skozi število igranj 10. testnega primera.

Slika 5 prikazuje še grafa dveh konstantnih primerov z nizkimi verjetnostmi avtomatov. Za vsak primer je igranih 5 novih iger z desetimi zaporednimi igranji. Zaradi variabilnosti različno hitro dosežemo končno izbrani



Slika 4: Grafa prikazujeta dva konstantna primera z nizkimi verjetnostmi. Za vsakega od obeh primerov je igranih 5 na novo začetih iger.

avtomat.Več primerov je na voljo na spletni aplikaciji, kjer si lahko ogledate vse testne primere in njihove uteži.

Zaključek: Spletna Aplikacija Celtra Jackpot Player je nastala v želji, da bi bil algoritem narejen za Celtrin izziv postavljen v aplikacijo. Razvoj algoritma je šla v smeri iskanja najboljše robustnosti. Temu je tudi podrejena izbira parametrov. Če bi ostalo še nekaj časa bi se razvoj preselil v testiranje (unit testing). Odsotnost le tega je edina očitna pomankljivost te aplikacije.