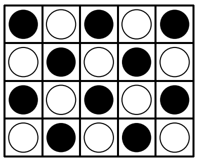
Kuidas indekseerida clobberi seise?

Jan Willemson

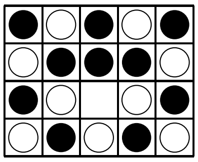
7. jaanuar 2004. a.

1 Mäng clobber

Mängu clobber leiutasid 2002. aasta veebruaris Richard Nowakowski, J. P. Grossman ja Michael Albert [6]. Seda mängitakse mustade ja valgete nuppudega $m \times n$ ruudustikul. Algseisus asuvad mustad ja valged nupud vaheldumisi:



Käiakse kordamööda (alustab nt must) ning käia saab parajasti siis, kui käigul olija mõne nupu kõrval (üleval, all, paremal või vasakul) asub vastavärvi nupp; käiguks on siis oma nupu liigutamine sellele ruudule ning vastase nupu eemaldamine. Pärast esimest käiku võib plats välja näha selline:



Kaotab mängija, kes enam käiku teha ei saa.

Otse reeglite põhjal võib clobberi kohta öelda järgmist.

- Mäng on lõplik, tsükliteta ja lõppeb ühe mängija võiduga.
- Kui kaks rühma nuppe on ühenduse kaotanud, siis see enam ei taastu. Seetõttu laguneb clobberi seis mängu käigus küllalt kiiresti järjest väiksemateks sõltumatuteks alamseisudeks.
- Mingis seisus on mustal käiguvõimalus parajasti siis, kui sama võimalus oleks ka valgel. Veel enamgi, igas seisus on mustal ja valgel sama arv käike.

Kuna mäng käib käigupuuduse peale, tähendab viimane omadus, et mingit mõistlikku seisuhinnangut on küllalt raske välja mõelda – iga käik halvendab sama palju mõlema mängija seisu.

Clobberit võib mängida ka ühe-, kolme-, nelja- jne mõõtmelisel laual, samuti *solitaire*-variandis [2].

2 Clobberi programmeerimine

Clobberi programmi kirjutades tuleb paratamatult arvestada teise ülaltoodud omadusega ning valmistuda hindama suurt hulka väikesi seise. Seejuures esinevad paljud alamseisud mängupuus palju kordi ning neid pole otstarbekas iga kord uuesti analüüsida. Selle asemel tasuks kord läbi vaadatud seis salvestada mingisse otsistruktuuri koos tema väärtusega (st tulemusega mõlemapoolse optimaalse mängu korral). Küsimus on, kuidas seda efektiivselt teha.

Zobristi paisktabel

Tehnika mänguseisude salvestamiseks paisktabelisse pakkus välja A. Zobrist aastal 1970 [7]. Zobristi paiskfunktsiooni väärtuse arvutamiseks tehakse mingi seisu korral järgmised sammud.

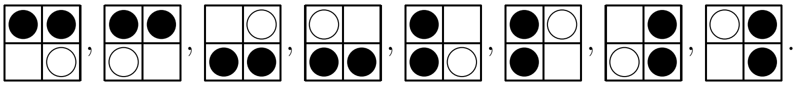
Mängu alul koostatakse $M \times N$ massiiv juhuslikest (nt 64-bitistest) arvudest, kus M on erinevate mängunuppude arv (nt clobberi korral 2, male korral 12) ja N mängulaua ruutude arv (nt clobberi korral $m \cdot n$, male korral 64).

Kui on tarvis leida mingi seisu paiskfunktsiooni väärtus, leitakse, millised nupud antud seisus millistel ruutudel asuvad ja XORitakse massiivi vastavad elemendid. Tulemus on paisktabeli võtmeks, millele vastavale aadressile salvestatakse mänguseisu väärtus.

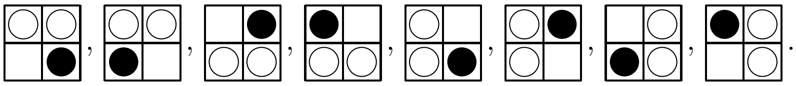
Kirjeldatud paiskfunktsiooni eeliseks on tema kiirus, väljundi ühtlane jaotus ja lihtne uuendatavus (selleks, et saada ühe seisu paiskfunktsiooni väärtusest ühekäigulise muudatusega tekkinud seisu paiskfunktsiooni väärtust, tuleb teha kaks XOR-tehet).

Zobristi paisktabel clobberi jaoks

Zobristi paisktabelit on lihtne kasutada kogu mänguseisu kordumise kindlakstegemiseks. Male ja kabe korral oluliselt paremat tulemust saavutada ei annagi, kuid eelpool nägime, et clobberi seisudel on omadus küllalt kiiresti väikesteks sõltumatuteks osadeks laguneda. See viib mõttele, et tegelikult tuleks paisktabelisse indekseerida just alamseise. Sõltumatute alamseisude identifitseerimine pole iseenesest raske ülesanne, kuid õnnetuseks võivad seisud olla pööratud. Halvemal juhul võib ühel seisul olla kuni kaheksa erinevat reinkarnatsiooni:



Lisaks paneme tähele, et toodud seisud on isomorfsed vahetatud värvidega seisudega teise mängija käigul olles ja ka neid seise pole enam tegelikult vaja teistkordselt läbi vaadata:



Hakates üht seisu salvestama kuueteistkümne eri seisuna, ähvardab Zobristi paisktabeli töökiirus kuusteist korda halvemaks minna. Olukord läheb veelgi hullemaks, kui analüüsida clobberit kõrgemates dimensioonides. Näiteks kolmemõõtmelist kujundit võib (koos "pahupidipööramisega") pöörata 48 viisil, seega saame koos värvide vahetusega 96 isomorfset seisu.

Kas seda probleemi saab paiskfunktsiooni modifitseerides kuidagi leevendada, ongi käesoleva seminari põhiküsimus.

3 Veel materjale

Artikkel [6] sisaldab lõpus kõik tähtsamad viited olemasolevatele materjalidele clobberi kohta, sh Toby Gottfried'i Java appletile [3]. 2002. aasta kevadel Tartu Ülikoolis toimunud mänguteooria seminaris valmis samuti kolm Clobberi programmi, Intenetis on neist kättesaadav Anton Litvinenko oma [5]. Zobristi paiskfunktsioni kasutamise kohta leiab palju viiteid otsimootorite abil, näiteks tasub pilk peale heita sellele, kuidas on paisktabel ehitatud GnuGO jaoks, vt [1] peatükk 14. Eespool püstitatud probleemiga on tegelenud Antti Huima [4], kes pakkus välja go-laua pöörete ning peegelduste suhtes hästi käituva paiskfunktsiooni. Võibolla tasuks just sellest allikast hakata otsima lahendust ka meie ülesandele.

Viited

[1] GnuGO dokumentatsioon. `http://www.delorie.com/gnu/docs/gnugo/gnugo_toc.html`.

[2] Erik D. Demaine, Martin L. Demaine, and Rudolf Fleischer. Solitaire Clobber. `http://citeseer.nj.nec.com/demaine02solitaire.html`.

[3] Toby Gottfried. Clobber Java applet. http://www.gottfriedville.net/games/clobber/.

[4] Antti Huima. A Group-Theoretic Zobrist Hash Function. http://persoweb.francenet.fr/~fgrieu/zobrist.pdf, 2000.

[5] Anton Litvinenko. Clobber Java applet. http://www.cs.ut.ee/~jan/game/ClobberApplet.html.

[6] Ivars Peterson. Getting clobbered. *Science News Online*, 161(17), aprill 2002. http://www.sciencenews.org/20020427/mathtrek.asp.

[7] Albert L. Zobrist. A new hashing method with application for game playing. Technical report, The University of Wisconsin, 1970.