Funktsionaalprogrammeerimise meetod. Projekt "Minesweeper". Mõned märkused.

Ilja Kuzovkin

May 29, 2010

1 Lahendaja

1.1 Admete esitamise abstraktsioon

Lahendamise idee baasiks on võetud lineaarvõrrandsüsteemide koostamise mehhanism. Iga avatud lahtri jaoks koostatake lineaarvõrrand, kus muutujad $x_1 ... x_n$ märgistavad uuritava lahtri naabreid ja vabaliikmeks on avatud lahtri väärtus A
x_1 x_4 x_6
x_2 A x_7
x_3 x_5 x_8
Muutujate kordajaks võivad olla ainult $\{0, 1\}$ kus 1 märgistab sulted lahtri ja 0 kõik ülejäänud lahti olekud. Muutujate arv saab olla ka rohkem kui 8 selle pärast, et iga lahter võrrandsüsteemis peab olema estatud oma muutujaga, võib olla ka väiksem kui 8, kui lahter on näiteks nurgas.
Näide (_ - suletud lahter)
_ 2 _
_ 3 _
_ _ 1
võrrand näeb välja niimodi $1x_1 + 1x_2 + 1_x3 + 0x_4 + 1x_5 + 1x_6 + 1x_7 + 0x_8 = 3$
Kui lahtri ümbruses on olemas ohtlikuks märgistatud lahtrid, nad saavad võrrandis ka kordajat 0 aga vabaliige vähendatakse 1 võrra igaühe jaoks. Näiteks (x - ohtliluks märgitud lahter, _ - suletud lahter)
x 2 _
x 3 _
_ _ 1
võrrandiks tuleb $0x_1 + 0x_2 + 1_x3 + 0x_4 + 1x_5 + 1x_6 + 1x_7 + 0x_8 = 1$
Võrranditest koostakse võrrandsüsteem, mille uuritakse järgmise reeglite abil.

1.2 Võrrandsüsteemi analüseerimisreeglid

1. Kordajate summa on võrdne vabaliikmega
   (a) $1x_1 + 1x_2 + 0x_3 + 0x_4 + 0x_5 + 0x_6 = 2$
   Selline võrrand näitab, et x_1 ja x_2 on mineeritud.
   (b) Sama reegel kehtib kui kordajad ja vabaliige on negatiivsed
   $(-1)x_1 + 0x_2 + 0x_3 + (-1)x_4 + 0x_5 + 0x_6 = -2$
   x_1 ja x_4 on mineeritud
   (c) Kui võrrandis on nii positiivsed, kui ka negatiivsed kordajad ja vabaliige on võrdne 0, siis kõik need muutujad inditseerivad mineeritud väljad
   $(-1)x_1 + 1x_2 + 1x_3 + (-1)x_4 + 0x_5 + 0x_6 = 0$
   x_1, x_2, x_3 ja x_4 on mineeritud

2. Kordajate summa $\neq 0$ ja vabaliige $= 0$

   (a) Nii positiivsed kui ka negatiivsed kordajad näitavad lahtid mida võib avada
   $0x_1 + (-1)x_2 + (-1)x_3 + 0x_4 + 0x_5 + 0x_6 = 0$ on sama mis
   $0x_1 + 1x_2 + 1x_3 + 0x_4 + 0x_5 + 0x_6 = 0$ ja tähenab et
   x_2 ja x_3 on ohutu ja neid võib avada

3. Sama märgiga kordajate summa = vabaliige ja kõige muutujate summa $\neq$ vabaliige

   (a) $0x_1 + (-1)x_2 + (-1)x_3 + 0x_4 + 1x_5 + 0x_6 = -2$
   näitab et x_2 ja x_3 on mineeritud aga x_5 on vaba

   (b) ja vastupidi, juhul kui
   $0x_1 + 1x_2 + 1x_3 + 0x_4 + (-1)x_5 + 0x_6 = 2$
   ka x_2 ja x_3 on mineeritud aga x_5 on vaba

2 Lahendaja algoritmi implementatsiooni kirjeldus

Lahendamisprotsess, mida alustab *solveField* funktsioon, näeb välja niimodi

1. Tuvastajale antakse lahendava väljaku positsiooni ja väljakute listi, mida oleme genereetitud serveri jaoks

   (a) Kui Tuvastaja tagastab väljaku - kohe saadame see väljak serverile ja lahendamisprotsess lõpeb

   (b) Kui Tuvastaja ei tagasta midagi, siis teeme lahendamissammud, mis on kirjeldatud allpool (punktid 2. kuni 10.)

2. Saadakse kõikide välja peal olevate avatud lahtide koordinaadid ja väärtused

3. Igaühe jaoks nendest leitakse naaberlahtrid ja nende olekud

4. Iga avatud lahti jaoks koostakse talle vastav võrrand, mis pannakse kirja listi kujul $[a_1, ..., a_n, b]$ kus $a_1..a_n$ on muutujate kordajad ja b on vabaliige

   (a) Kui lahter on kinni, siis vastav $a_i = 1$

   (b) Kui lahter on juba ohtlikuks määratud, siis vastav $a_i = -1$

   (c) Muude lahri olekute puhul $a_i = 0$

5. Iga võrrandi jaoks -1 asendatakse 0-dega ja vabaliikmest lahutatakse -1 summa (vt. peatükk 1.1)
   $[1, 0, 0, -1, -1, 0, 0, 1, -1, 1, 0, 5]$ muutub
   $[1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 2]$ ks

6. Võrranditest koostakse maatriks (int listide list)

7. Koostatakse võrrandite list, kuhu lähevad

   (a) Võrrandsüsteemi võrrandid

   (b) Võrrandid, mis on saadetud maatriksi igast reast iga selle all oleva ridade lahutamisega. Näiteks kui maatriksis on 4 rida (4 võrrandid LVSis) siis listi lähevad:
   $rida4 - rida3, rida4 - rida2, rida4 - rida1, rida3 - rida2, rida3 - rida1, rida2 - rida1$

8. Igal eelmises punktid saadetud real kujul $[a_1, ..., a_n, b]$ proovitakse reeglid (peatükk 1.1) mille tulemusena saame nimekiri lahtritega, mida võib avada, ja nimekiri lahtritega, mida on vaja ohtlikuks märgistada.

9. Kui eelmise sammuga ei saanud leida ühtegi ohtliku või vaba lahtri, siis võetakse naiivselt esimene suletud lahter

10. Saadud tulemus saadetakse serverile, ja kui vastusena saame uus mänguväljaku olek, siis kordame protsessi alustades punktist 1.

3 Tuvastaja

Tuvastaja saab 2 parameetrit

1. Väljakute listi, kus iga väljaku jaoks on märgitud ainult kus on miinid ja kus neid ei ole

2. Praegu mängitava mängu väljaku, kus on näha, millised lahtid on juba avatud ja millised on ohtlikuks määratud

Nende andmete põhjal peab Tuvastaja leidma väljakute listist ühe, kus miinide paigaldus on sama, mis praegusel mängul ja otsustada kas praegu lahendav väljak ongi seesama, millega me võrdleme või ei ole.

Sarnaseid väljakud leitakse järmiste reeglite abil (olgu lahendav väljak - G ja see millega võrdleme - A)

1. Ühtegi lahter, mis on mineeritud väljakul A, ei saa olla avatud väljakul G (funktsioon $similarityRule1$)

2. Ühtegi lahter, mis ei ole mineeritud väljakul A, ei saa olla ohtikuks märgitud väljakul G (funktsioon $similarityRule1$)

3. Iga lahter, mis on avatud väljakul G, kui ta asuks väljakul A, peab näitama õige arv miine oma ümbrusest (funktsioon $similarityRule2$)

Kui kõik need reeglid kehtivad mingi väljaku A jaoks, siis väljak A peetakse sarnaseks. Niimodi koostatkse sarnaste väljakute nimekirja. Kui nimekirjas ei ole ühtegi väljakut, siis Tuvastaja ei oska pakkuda midagi Lahendajale. Kui nimekirjas on rohkem kui üks sarnane väljak, siis ka Tuvastaja ei oska Lahendagale midagi pakkuda. Juhul aga kui nimekirjas on üks ainus väljak (nimetame seda M), siis on lootust, et see sobib Lahendajale.
Seda sobivust kontrollitakse viimase reegli abil: vaadetakse, kui palju lahreid väljakust G on avastatud (avastute hulka kuuluvad avatud ja ohtlikuks määratud lahtrid). Juhul, kui

$$\frac{(avastatud)}{(lahtreidkooku)} * 100 > P$$

kus P on konstant (meie programmis 30) siis väljakut M antakse Lahendajale.

4 Kiirus

4.1 Lahendaja

Lahendaja algoritm muutub aeglaseks väljaku lahtite arvu suurenemisega. 50 mängu 2 mängijaga lahendab ta 67 sekundiga, kui väljaku mõõde on 7x12. Aga lahendamisaeg kasvab kiiresti: 16x30 väljaku puhul nende 50 mängu 2 mängijaga prognoseeritav lahendamisaeg on 5 tundi sama arvuti peal. Kiirus sõltub avatute lahtrite kogusest väjaku peal. Iga avatud lahtri jaoks koostatakse võrrand ja pärast iga võrrand sarnastatakse üks kord kõigi teiste võrranditega. 4 võrrandite puhul tuleb

$$4 * 3 + 3 * 2 + 2 * 1 + 4 = 24$$

sammu. Võrrand, mis näitab sammude arvu, mida peab tegema algoritm n avatud lahtrite korral on

$$\frac{(n-1)^3}{3} + (n-1)^2 + \frac{(n-1)*2}{3} + n$$

Saame siis $O(n^3)$ keerukust. Üks optimiseerimisidee seisneb selles, et mitte vaadelda need avatud lahtrid, mille übruses kõik väljad on juba avastatud (avatud või märgitud), sest nende analüüs annab meile tulemust, mida juba teame. Tõepoolest, pärast selliste võrrandite süsteemist eemaldamist kiirus suurenes: väljakul 7x12 sai keskmine lahendamis aeg 39.8 sekundit (50 mängu 2 mängijat), optimiseerimata versioon aga näitas sama konfiguratsiooni peal 66.7 sekundit (keskmine).

4.2 Tuvastaja

Tuvastaja algoritmi keerukus peaks olema $O(mn)$, kus m on väljakute arv ja n on väljaku lahtrite arv. Lahendava ülesanne raamis see töötab piisavalt kiiresti.

5 Tulemused

Keskmine arv väljakuid, mida Lahendaja koos Tuvastajaga oskasid korrektselt lahendada oli 7 kuni 14 (50 st). Viidud testide põhjal on näha, et õnnestumiste arv sõltub vähe väljaku suurusest.

6 Märkused

Panin serveri koodis, et mäng toimuks väljakul 7x12 20 miinidega. 16x30-ga oli minu algoritm kahjuks liiga aeglane, et vabalt testida.

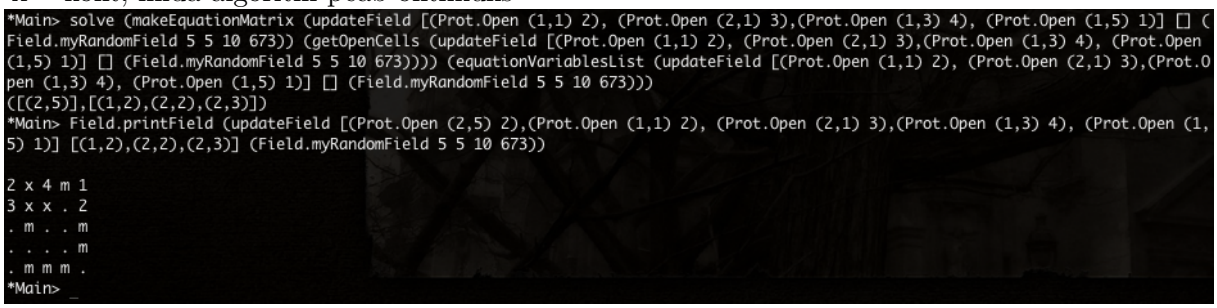
7 Näited

7.1 Juhusilku väljaku genereerimine ja selle uuendamine

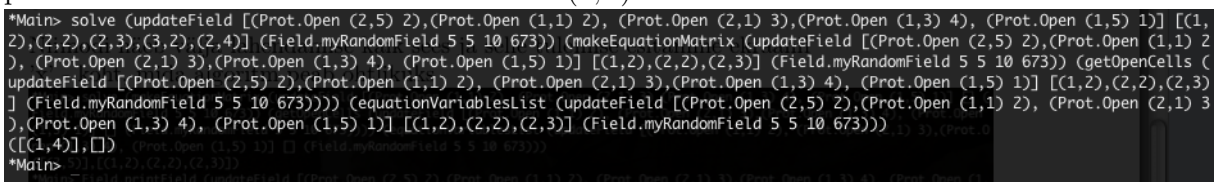
Mineeritud lahtrid on märgitud tähega 'm' - algoritmi jaoks see on sama mis suletud lahter. Tegin neid nähtavaks enda mugavuse jaoks.

7.2 Lahendamissamm

Niimodi näeb välja lahendamise käik sees ja selle tulemise esitamine ekraanil
'x' - koht, mida algoritm peab ohtlikuks



Järgmine samm - midagi pakkuda ei oska (siin ka inemine ei oskaks midagi täpselt öelda) - pakub naiivselt esimene lahter mis on kinni $(1, 4)$



7.3 Lahendus leitud Tuvastaja abil

```
0 0 0 2 x . . . . . .
1 2 1 4 x . . . . . .
x 2 x 3 x . . . . . .
1 2 1 2 2 . . . . . .
0 0 0 0 2 . . . . . .
2 3 2 1 1 . . . . . .
. . . . . .
Lets try to open [(1,6),(1,8),(1,9),(1,10),(1,11),(1,12),(2,6),(2,10),(2,11),
(2,12),(3,6),(3,7),(3,8),(3,9),(3,10),(3,11),(3,12),(4,7),(4,8),(4,9),(4,10),
(4,11),(4,12),(5,6),(5,7),(5,8),(5,10),(5,11),(6,8),(6,9),(6,10),(6,12),(7,4),
(7,5),(7,6),(7,8),(7,10),(7,11),(7,12)] and flag [(1,7),(2,7),(2,8),(2,9),
(4,6),(5,9),(5,12),(6,6),(6,7),(6,11),(7,1),(7,2),(7,3),(7,7),(7,9)]
Solved 100.0% of field.
```

7.4 Landus leitud puhtalt Lahendajaga

```
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
Lets try to open [(1,1)] and flag []
Answer: open [Open (1,1) 1] flagged []

1 . . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
Lets try to open [(1,2)] and flag []
Answer: open [Open (1,2) 2] flagged []

1 2 . . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
```

```
Lets try to open [(1,3)] and flag []
Answer: open [Open (1,3) 3] flagged []

1 2 3 . . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
Lets try to open [(1,4)] and flag [(2,3)]
Answer: open [Open (1,4) 4] flagged [(2,3)]

1 2 3 4 . . . . .
. . x . . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
Lets try to open [(2,1)] and flag [(2,2),(2,4),(1,5),(2,5)]
Answer: open [Open (2,1) 1] flagged [(2,2),(2,4),(1,5),(2,5)]

1 2 3 4 x . . . . .
1 x x x x . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
Lets try to open [(3,1),(3,2)] and flag []
Answer: open [Open (3,1) 1,Open (3,2) 2] flagged []

1 2 3 4 x . . . . .
1 x x x x . . . . .
1 2 . . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
Lets try to open [(4,1),(4,2),(3,3),(4,3)] and flag []
Answer: open [Open (3,3) 3,Open (4,1) 1,Open (4,2) 1,Open (4,3) 0,Open (3,4) 4,
Open (4,4) 1,Open (5,2) 1,Open (5,3) 1,Open (5,4) 1] flagged []

1 2 3 4 x . . . . .
1 x x x x . . . . .
1 2 3 4 . . . . .
1 1 0 1 . . . . .
. 1 1 1 . . . . .
. . . . . .
. . . . . .
Lets try to open [(5,5),(6,1),(6,2),(6,3)] and flag [(5,1)]
```

```
Answer: open [Open (5,5) 2,Open (6,1) 1,Open (6,2) 2,Open (6,3) 2] flagged [(5,1)]

1 2 3 4 x . . . . . .
1 x x x x . . . . . .
1 2 3 4 . . . . . .
1 1 0 1 . . . . . .
x 1 1 1 2 . . . . . .
1 2 2 . . . . . .
. . . . . .
Lets try to open [(4,5),(6,5),(7,1),(7,2)] and flag [(6,4),(7,3)]
Answer: open [Open (4,5) 1,Open (6,5) 3,Open (7,1) 0,Open (7,2) 1]
flagged [(6,4),(7,3)]

1 2 3 4 x . . . . . .
1 x x x x . . . . . .
1 2 3 4 . . . . . .
1 1 0 1 1 . . . . . .
x 1 1 1 2 . . . . . .
1 2 2 x 3 . . . . . .
0 1 x . . . . . .
Lets try to open [(3,6),(4,6),(5,6),(7,4)] and flag [(3,5)]
Answer: open [Open (3,6) 2,Open (4,6) 2,Open (5,6) 2,Open (7,4) 3] flagged [(3,5)]

1 2 3 4 x . . . . . .
1 x x x x . . . . . .
1 2 3 4 x 2 . . . . .
1 1 0 1 1 2 . . . . .
x 1 1 1 2 2 . . . . .
1 2 2 x 3 . . . . . .
0 1 x 3 . . . . . .
Lets try to open [(2,6),(2,7),(3,7),(4,7)] and flag [(5,7),(6,6),(7,5)]
Answer: open [Open (2,6) 4,Open (2,7) 1,Open (3,7) 0,Open (2,8) 1,Open (3,8) 0,
Open (2,9) 1,Open (3,9) 0,Open (2,10) 2,Open (3,10) 1,Open (4,7) 1,Open (4,8) 1,
Open (4,9) 0,Open (4,10) 1,Open (5,8) 1,Open (5,9) 1,Open (5,10) 1]
flagged [(5,7),(6,6),(7,5)]

1 2 3 4 x . . . . . .
1 x x x x 4 1 1 1 2 . .
1 2 3 4 x 2 0 0 0 1 . .
1 1 0 1 1 2 1 1 0 1 . .
x 1 1 1 2 2 x 1 1 1 . .
1 2 2 x 3 x . . . . . .
0 1 x 3 x . . . . . .
Lets try to open [(1,8),(6,7),(6,8),(6,9),(7,6)] and flag [(6,10)]
Answer: open [Open (1,8) 1,Open (6,7) 4,Open (6,8) 3,Open (6,9) 2,Open (7,6) 3]
flagged [(6,10)]

1 2 3 4 x . . 1 . . . .
1 x x x x 4 1 1 1 2 . .
1 2 3 4 x 2 0 0 0 1 . .
1 1 0 1 1 2 1 1 0 1 . .
x 1 1 1 2 2 x 1 1 1 . .
```

```
1 2 2 x 3 x 4 3 2 x . .
0 1 x 3 x 3 . . . . . .
Lets try to open [(4,11),(5,11),(6,11),(7,9),(7,10)] and flag [(3,11),(7,8),(7,7)]
Answer: open [Open (4,11) 2,Open (5,11) 2,Open (6,11) 2,Open (7,9) 2,Open (7,10) 1]
flagged [(3,11),(7,8),(7,7)]

1 2 3 4 x . . 1 . . . .
1 x x x x 4 1 1 1 2 . .
1 2 3 4 x 2 0 0 0 1 x .
1 1 0 1 1 2 1 1 0 1 2 .
x 1 1 1 2 2 x 1 1 1 2 .
1 2 2 x 3 x 4 3 2 x 2 .
0 1 x 3 x 3 x x 2 1 . .
Lets try to open [(1,11),(2,11),(7,11)] and flag []
Answer: open [Open (1,11) 2,Open (2,11) 3,Open (7,11) 1] flagged []

1 2 3 4 x . . 1 . . 2 .
1 x x x x 4 1 1 1 2 3 .
1 2 3 4 x 2 0 0 0 1 x .
1 1 0 1 1 2 1 1 0 1 2 .
x 1 1 1 2 2 x 1 1 1 2 .
1 2 2 x 3 x 4 3 2 x 2 .
0 1 x 3 x 3 x x 2 1 1 .
Lets try to open [(3,12),(6,12),(7,12)] and flag [(5,12)]
Answer: open [Open (3,12) 2,Open (6,12) 1,Open (7,12) 0] flagged [(5,12)]

1 2 3 4 x . . 1 . . 2 .
1 x x x x 4 1 1 1 2 3 .
1 2 3 4 x 2 0 0 0 1 x 2
1 1 0 1 1 2 1 1 0 1 2 .
x 1 1 1 2 2 x 1 1 1 2 x
1 2 2 x 3 x 4 3 2 x 2 1
0 1 x 3 x 3 x x 2 1 1 0
Lets try to open [(4,12)] and flag [(2,12)]
Answer: open [Open (4,12) 2] flagged [(2,12)]

1 2 3 4 x . . 1 . . 2 .
1 x x x x 4 1 1 1 2 3 x
1 2 3 4 x 2 0 0 0 1 x 2
1 1 0 1 1 2 1 1 0 1 2 2
x 1 1 1 2 2 x 1 1 1 2 x
1 2 2 x 3 x 4 3 2 x 2 1
0 1 x 3 x 3 x x 2 1 1 0
Lets try to open [(1,6)] and flag []
Answer: open [Open (1,6) 3] flagged []

1 2 3 4 x 3 . 1 . . 2 .
1 x x x x 4 1 1 1 2 3 x
1 2 3 4 x 2 0 0 0 1 x 2
1 1 0 1 1 2 1 1 0 1 2 2
x 1 1 1 2 2 x 1 1 1 2 x
1 2 2 x 3 x 4 3 2 x 2 1
```

```
0 1 x 3 x 3 x x 2 1 1 0
Lets try to open [(1,9)] and flag [(1,7)]
Answer: open [Open (1,9) 1] flagged [(1,7)]

1 2 3 4 x 3 x 1 1 . 2 .
1 x x x x 4 1 1 1 2 3 x
1 2 3 4 x 2 0 0 0 1 x 2
1 1 0 1 1 2 1 1 0 1 2 2
x 1 1 1 2 2 x 1 1 1 2 x
1 2 2 x 3 x 4 3 2 x 2 1
0 1 x 3 x 3 x x 2 1 1 0
Lets try to open [(1,12)] and flag [(1,10)]
Solved 100.0% of field.
```