PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **18** (1990/1991)
Številka 2
Strani 90–93

Janez Žerovnik:

PROBLEM TRDNJAV NA ŠAHOVNICI IN POŽREŠNA METODA

Ključne besede: računalništvo, matematika, problem trdnjav, požrešna metoda.

Elektronska verzija: http://www.presek.si/18/1032-Zerovnik.pdf

PROBLEM TRDNJAV NA ŠAHOVNICI IN POŽREŠNA METODA

Nedavno sem naletel na zanimiv problem. Kljub temu, da je na prvi pogled videti enostaven, ga je v splošnem težko rešiti.

Začnimo v dveh dimenzijah. Imejmo šahovnico velikosti 3×3. Poskusimo postaviti nanjo kar najmanj trdnjav, tako da bo vsako polje napadeno vsaj po enkrat.

Prepričajmo se, da potrebujemo vsaj 3 trdnjave.

1. Če postavimo v vsaki vrstici vsaj eno trdnjavo, potem so v rešitvi gotovo vsaj 3 trdnjave.
2. Pa recimo, da v rešitvi obstaja vrstica brez trdnjave. Ker so v vrstici 3 polja, morajo biti v ostalih dveh vrsticah vsaj 3 trdnjave, če hočemo, da bodo napadena vsa polja v vrstici brez trdnjave.

Slika 1. Skica ene od optimalnih rešitev za $n = 2$

Če ne bi šlo drugače, bi lahko poskusili vse mogoče postavitve trdnjav, vendar tega ne priporočam. Vseh postavitev je namreč kar

$$2^{3^2} = 2^9 = 512$$

saj je polj na šahovnici $3^2 = 9$, na vsako polje pa lahko trdnjavo postavimo, ali pa ne.

Posplošimo zdaj primer na več dimenzij! V n dimenzijah imejmo n–dimenzionalno "šahovnico" s $3 \times 3 \times \cdots \times 3 = 3^n$ polji. Vsako polje šahovnice lahko opremimo s koordinatami. Ker smo se omejili na dolžino šahovnice 3, je koordinata eno od števil 0, 1, 2. Poljubno polje 3–dimenzionalne šahovnice tedaj enolično opišemo s trojico (i, j, k), na primer $(0, 0, 0)$, $(0, 2, 1)$, $(2, 2, 0)$ itd. Položaj trdnjave opišemo s poljem, na katero je postavljena. Trdnjava na dvodimenzionalni šahovnici napada vsa polja, ki se od njenega položaja razlikujejo v natanko eni koordinati. In posplošimo: trdnjava na n–dimenzionalni šahovnici napada vsa polja, ki se od njenega položaja razlikujejo v natanko eni koordinati. Povzemimo:

Problem trdnjav: *Za dani n poišči minimalno število trdnjav, ki jih lahko postavimo na n–dimenzionalno šahovnico (s stranicami dolžine 3) tako, da bodo vsa polja šahovnice napadena.*

Za kasnejšo rabo se dogovorimo, da bomo vsako postavitev trdnjav na šahovnico, pri kateri so vsa polja napadena, imenovali *(približna) rešitev* problema. Rešitev, pri kateri je uporabljeno minimalno število trdnjav pa imenujmo *optimalna rešitev.*

V angleščini problem imenujejo "*the Football–Pool problem*", kar bi lahko za silo prevedli kot "*problem športne napovedi*". Velja namreč naslednje: vsako rešitev problema (množica n-teric ničel, enic in dvojk) lahko uporabimo za športno napoved z n-pari. Rešitev ima lepo lastnost: vnaprej vemo, da bomo gotovo (vsaj enkrat) zadeli drugo nagrado (zgrešili bomo največ en izid). Ker plačamo pri športni napovedi vsak stolpec (torej vsako n-terico), je optimalna rešitev najcenejša rešitev z lastnostjo, da vedno zadane vsaj drugo nagrado.

Kljub temu, da na prvi pogled problem ni videti prezahteven, je povzetek znanega kaj skromen. S kombinatorično analizo so uspeli poiskati optimalne rešitve v dimenzijah 2,3,4 in 5. Dokaz, da je dobljena rešitev za dimenzijo 5 res optimalna, je dolg kar 10 strani! Za večje dimenzije je znanih nekaj zgornjih mej za optimalne rešitve, toda nič več. Dokazov, da so rešitve zares najboljše, torej optimalne, ni. Za računalnikarje bo zanimivo, da so doslej najboljše rešitve (in s tem najboljše zgornje meje za optimalno rešitev) v dimenzijah 6,7 in 8 našli z računalnikom. Algoritem, ki so ga uporabili, je popularno "*ohlajanje*" (angleško "*Simulated Annealing*") in je sam po sebi zanimiv, vendar njegov opis presega namene tega članka. Morda kdaj drugič, tukaj pa povejmo le to, da je algoritem za rezultate na Sliki 2 porabil veliko (ure in ure) časa na velikih računalnikih.

n	2	3	4	5	6	7	8
število trdnjav	3	5	9	27	≤ 73	≤ 186	≤ 468

Slika 2. Optimalne rešitve ali najboljše ocene zanje

Poglejmo si enostavni poskus računanja rešitev po "*požrešni metodi*". Opišimo na kratko idejo požrešne metode. Rešitev gradimo postopoma. Na vsakem koraku dodamo tisti element, ki največ prispeva h "*kriterijski funkciji*". Sprejmemo ga le, če je s tem elementom razširjena množica še "*dopustna*". Kako deluje požrešna metoda na različnih problemih, si

lahko preberete na primer v knjigi Jerneja Kozaka Podatkovne strukture in algoritmi.

Naš program postopa preprosto: izbere si polje, ki še ni napadeno, in ga pokrije s trdnjavo, ki poleg izbranega pokrije še kar največ novih polj. To ponavlja, dokler niso vsa polja napadena.

V našem konkretnem primeru je rešitev poljubna množica trdnjav. Kriterijska funkcija je vsota števila trdnjav v rešitvi in števila še nenapadenih polj; optimalna je torej rešitev, ki napada vsa polja z najmanj trdnjavami. Dopustna je vsaka delna rešitev, ki jo je mogoče dopolniti do prave rešitve. V primeru problema trdnjav so vse delne rešitve dopustne, v splošnem pa utegnemo potrebovati nekaj več pazljivosti.

Požrešna metoda je enostavna, zato ne moremo pričakovati, da bo optimalna za vse probleme. Res se izkaže, da je za nekatere probleme mogoče dokazati, da je požrešna metoda "prava", za mnoge probleme pa velja, da s požrešno metodo ne bomo dobili optimalnih rešitev. Videli bomo, da to zadnje velja tudi za problem trdnjav na šahovnici.

Če algoritem poženemo, za $n = 2, 3, 4, 5$ dobimo optimalne rešitve! Ali to pomeni, da je problem tako enostaven? Na žalost ne, pri $n = 6$ je rezultat algoritma že 90 trdnjav, kar je precej slabše od enostavne rešitve, ki jo dobimo, če sestavimo skupaj tri rešitve za $n = 5$ (Premisli, da taka konstrukcija pri vsakem n vedno da rešitev za $n + 1$ s trikratnim številom trdnjav!). Če pogledamo na Sliko 2, vidimo, da je med $n = 4$ in $n = 5$ konstrukcija, ki sestavi tri manjše rešitve, celo optimalna, nasploh pa to seveda ni res.

n	2	3	4	5	6	7	8
število trdnjav	3	5	9	27	90	248	679

Slika 3. Rešitve, dobljene s požrešnim algoritmom

Kaj pa če bi način izbire trdnjav, ki naj pokrijejo izbrano polje nekoliko spremenili? Izdam vam, da sem nekaj različnih strategij preizkusil, pa se niso izkazale za dobre. Na primer: postavil sem trdnjavo kar na izbrano polje ali pa (slučajno) nekam na okolico, itd.

Morda bi se splačalo poskusiti s programom sestavljati (delne) rešitve za manjše dimenzije? Če je vašemu računalniku kdaj dolgčas, pa poskusite še vi! Še enkrat opozorimo, da je zelo malo možnosti, da bi hitro lahko našli kakšno 'doslej najboljšo rešitev'. Zelo zanimivo bi že bilo imeti algoritem (program), ki bi dajal boljše rešitve od požrešne metode.