Poročilo:

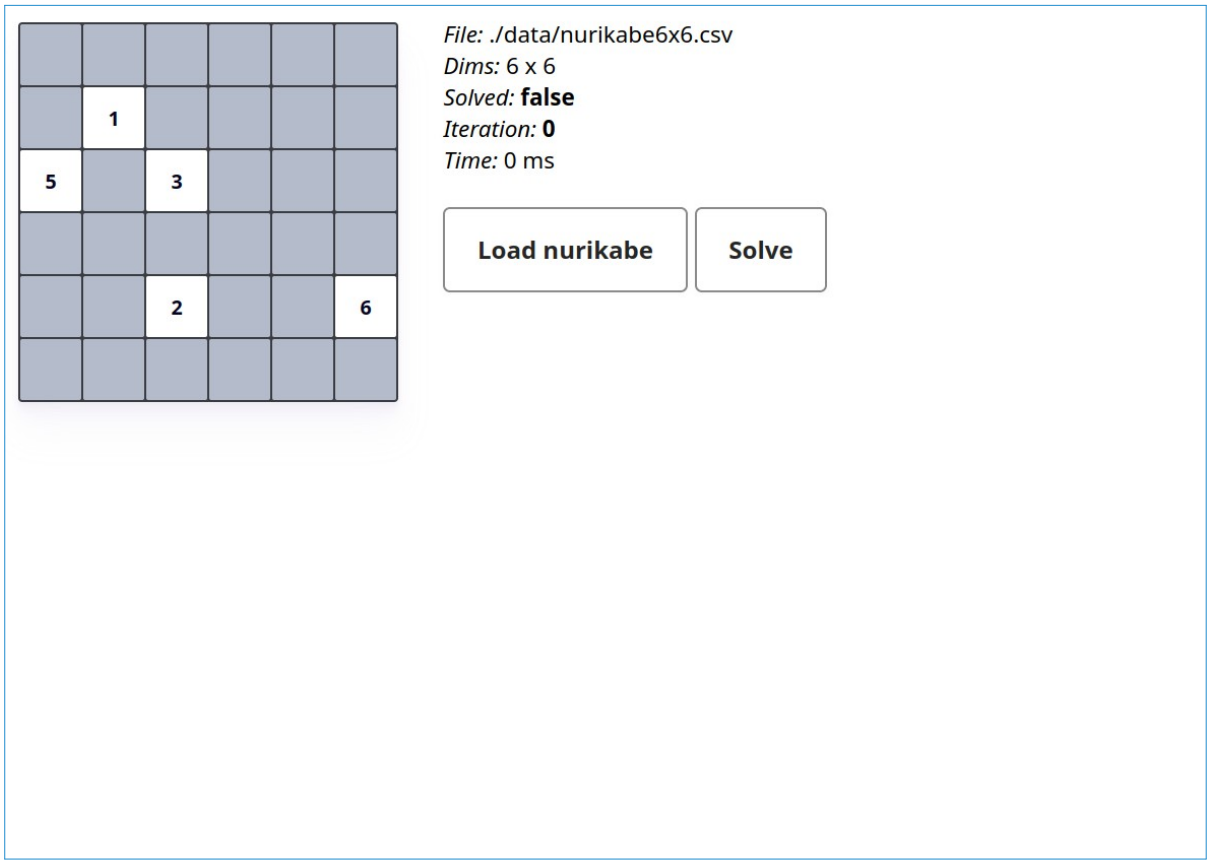
Nurikabe

Simon Vajs

Kazalo

Uporabljene tehnologije

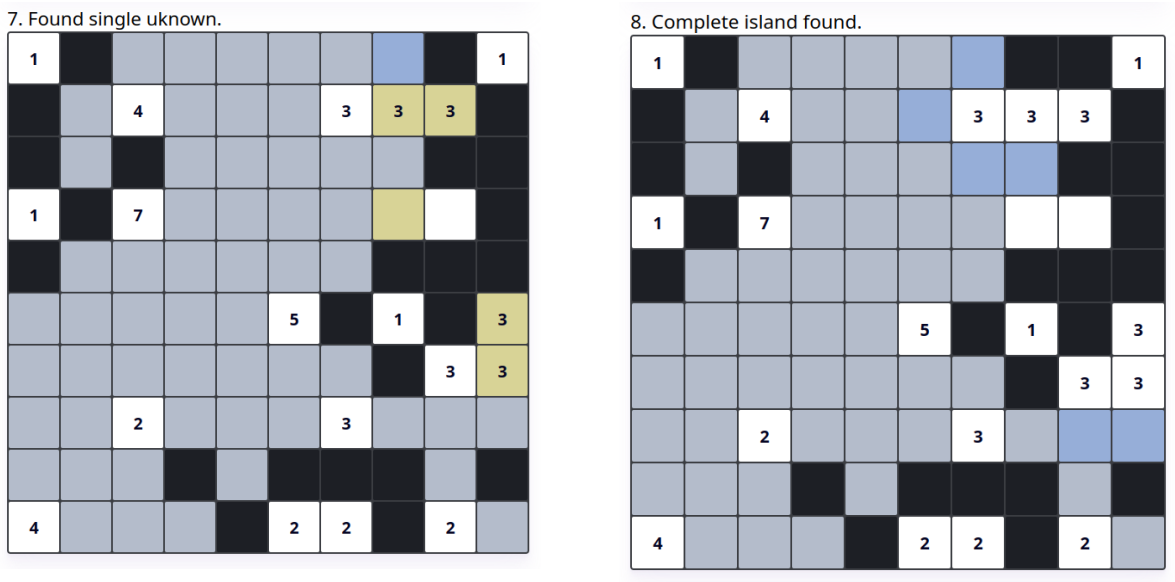
Za izvedbo naloge sem uporabil programski jezik Rust. Odločil sem se tudi, da bom Rust kodo pretvoril v spletno strojno kodo (angl. WebAssembly ali WASM), ker je ta primerna za uporaba z Javascript in HTML. Pri prikazu rezultatov sem si pomagal z preprosto spletno stranjo, s katero sem lahko klical WASM metode z reševanje uganke Nurikabe.



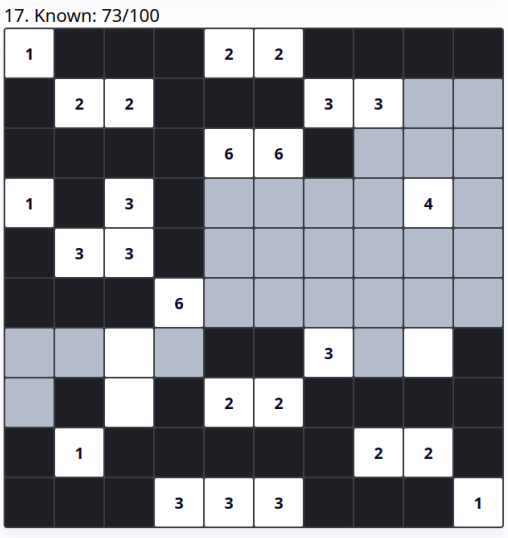
Poizkus 1: Pravila

Prvo sem poizkusil rešit problem z ne hevrističnim postopkom, kjer z vsako iteracijo izvede preverjanje pravil od manj do bolj računalniško zahtevnih. Program oz. reševalec (angl. solver) se ustavi, kadar se nobeno pravilo ne ujema s trenutnim stanjem mreže, prav tako preide v naslednjo iteracijo reševanja, kadar je eno pravilo prepoznano in polja pobarvana. Na sliki 1 je primer ko program najde situacijo s samo eno možno lokacijo širjenja v iteraciji 7 in takoj v naslednji iteraciji prepozna prvi končan otok, ki ima vsa bela polja označena.

.



Slika 1: Primer1 (10x10) z iteracijo 7 in 8



Slika 2: Primer4 (10x10) rešen s pravili 73%

Pravila, ki se v programu izvedejo v istem vrstnem redu, so sledeča:

1. Končan otok – preveri za že končane otoke in pobarva mejne celice črno.
2. Eno ne znano - če se lahko črna ali bela regija širiti samo v eno smer (torej imata samo enega neznanega soseda, na sliki so to sive barve), se ta mora pobarvat s isto barvo regije, saj bi ta postala ločena v nasprotnem primeru.
3. Dve ne znani celici – kadar ima otok še samo eno potrebno belo celico in se dotika dveh sivih polj, se lahko v določenih situacijah diagonalna celica pobarva črno (ne implementirano).
4. Na meji – kadar sta dva otoka na meji samo enega ne znanega polja med njima, se ta lahko pobarva samo črno, prav tako če se dotikata z robi na diagonalah se ta sosednji polji pobarvata črno.
5. Mogoči bazen – gre skozi celotno mrežo in prepozna robove, 2x2 polja s tremi črnimi celicami. Če je situacija najdena, se zadnja celica pobarva belo, in nadaljuje.
6. Preverjanje napak (angl. Contradiction) - preveri samo da ni postal bazen in da je še možno zapolniti vsa bela polja.
7. Ne dosegljivo – tu se uporabi iskanje v širino oz. BFS (angl. Breath first search). Pregleda za vsako neznano polje ali še lahko doseže kateri otok. Če to ni več mogoče, se ta mora pobarvat črno.

Če pri nobenem pravilu ni bila prepoznana vsaj ena primerna situacija, se reševanje prekine.

Sicer gre s takšnimi pravili rešiti igro Nurikabe, je to izvedljivo samo za manjše mreže ali pa velike z večino otokov ne večjih od 3, kar pa tudi ni realistično. Ta program ni rešil nobenega podanega primera 10x10. Rešil je samo primere 5x5 ali pa 6x6, ki sem jih poiskal na spletu.

V nekaterih poznanih rešitvah [1][2] uporabijo še druge tehnike, ena od teh je ugibanje. Ugibanje se izvede na zadnje (ostale situacije ne prepoznana na mreži) in naključno izbere eno iz med neznanih mejnih celicah z že znanimi in jo označi belo ali črno v upanju, da v ponovni iteraciji na mreži z ugibom prepozna napako. Takrat pomeni, da je celica nasprotne barve.

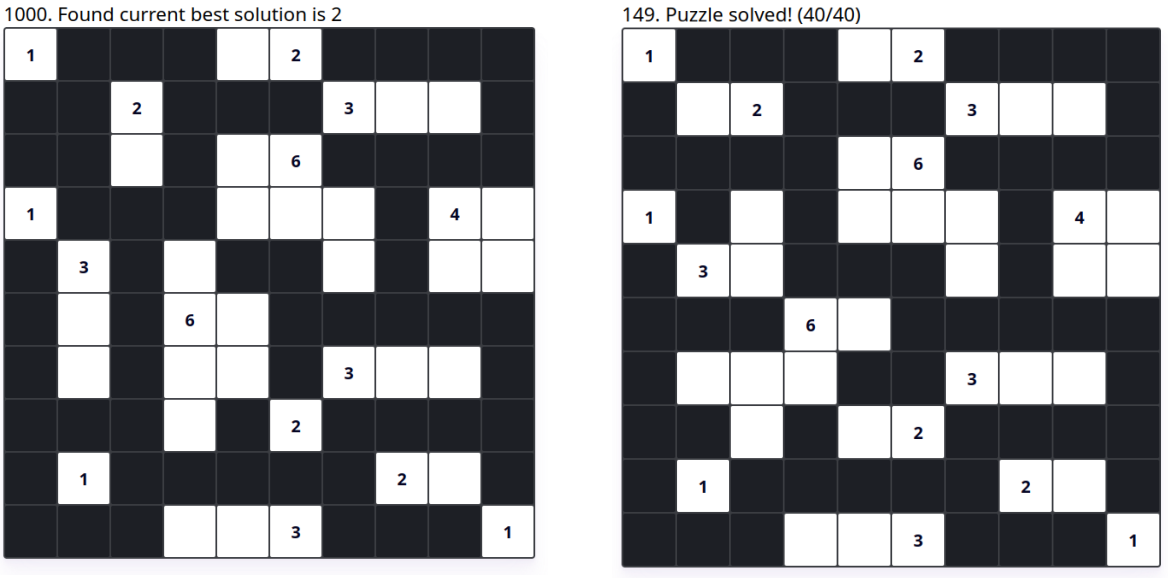
Ugibanje nisem implementiral, saj nisem bil prepričan, da bi rešila ta večje podane primere. Zato sem moral poiskati bolj primerno rešitev in tako naletel na reševanje s kolonijo mravelj.

Poizkus 2: Kolonija mravelj

Iskal sem druge rešitve za rešitev težave in našel nekaj, ki jih lahko implementiram [6]. Dva od teh sta brute force in optimizacija kolonije mravelj . Brute force zapolni mrežo z 2x2 polji. Vsako polje ima lahko od 2-16 možnih 2x2 polj, odvisno od od označenih otokov. Išče pa seveda vse možne načine kako zapolnit mrežo in sproti preverja, če je trenutno stanje možna rešitev. Je izvedljivo, samo ni primerno za velika polja, sploh pa ne 10x10.

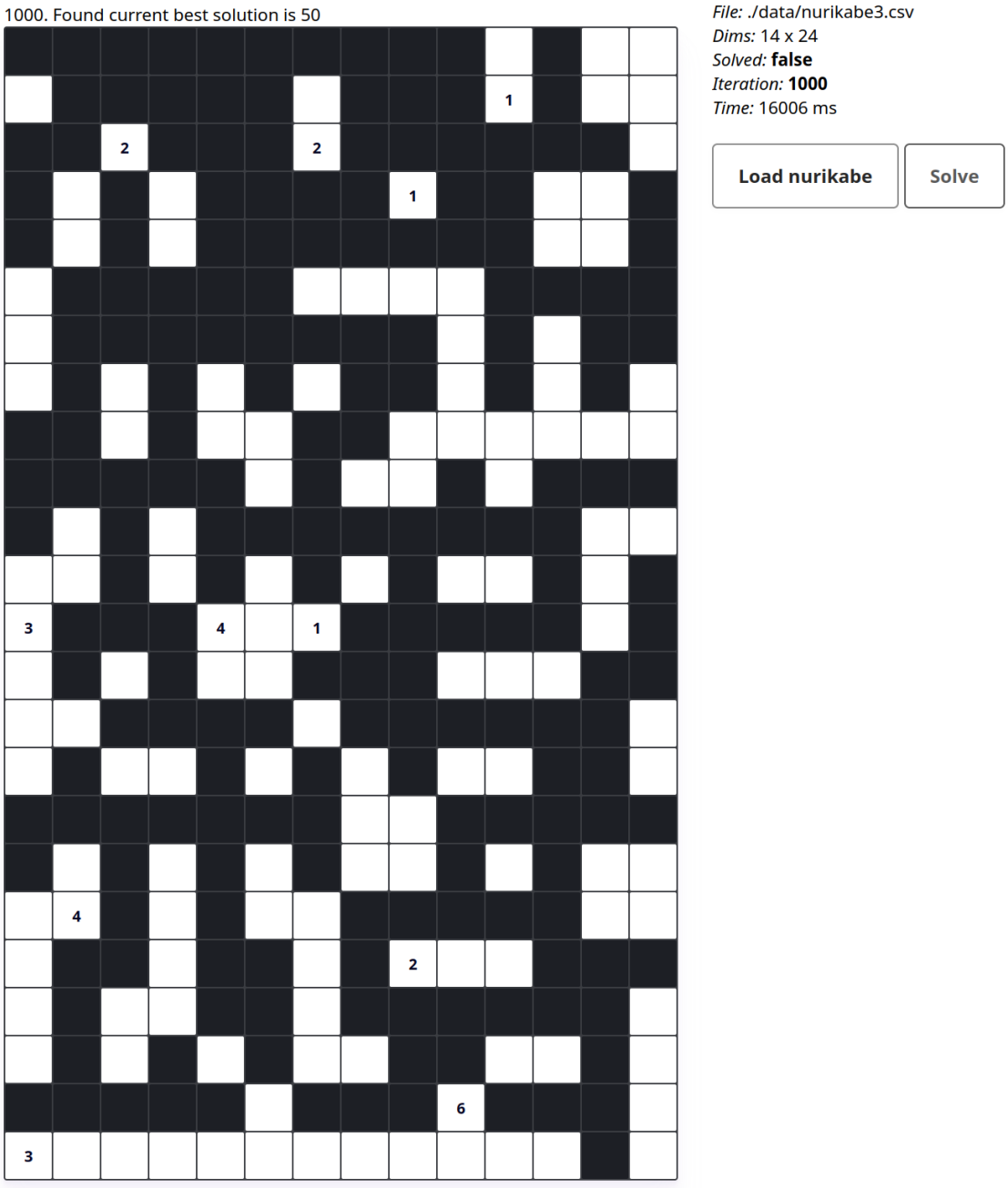
S optimizacijo kolonij mravelj ali ACO (angl. Ant Colony Optimisation) je mogoče najti rešitev s ustvarjanjem mravelj (ena rešitev na generacijo), ki zapolnijo popolnoma obarvano črno mrežo s belimi celicami z izvori začetnih lokcaij otokov, tako da ne krši dobenega postavljenega pravila. V vsaki generacijo se npr. najde 10 rešitve, ki prestavljajo mravlje, nato pa izbere najboljša, ki nadomesti trenutno rešitve in fermone. Glede na fermone se mravlja odloča v katero smer se bo ta širila, tako se glede na trenutno morebitno rešitev več krat izberejo celice, ki so bližje rešitvi. Rešitev se oceni z enačbo: (možne bele celice – bele celice rešitve + število bazenov). Vrednost 0 prestavlja pravilno rešitev, kadar se pa najde boljša rešitev od trenutne, te ta nadomesti trenutno mrežo in fermone.

Slika 3 prikazuje isti primer, na levi strani ne rešen po 1000 iteracij (1 iteracija je 1 mravlja), na desni pa rešen po 149 iteracij. V obeh primerih je uporabljen 10 mravelj na generacijo. Število iteracij, ki jih je potrebno izvesti do rešitve, je lahko kar različno. Čas potreben za 5000 iteracij je povprečno 5-6s za 10x10 mreže. Seveda je za ene primere potrebno veliko več iteracij, za nekatere primere 10x10 pa tudi ne najde rešitve še po eni minuti.



Slika 3: Primer4(10x10) rešen s kolonijo mravelj

Na sliki 4 je prikazan najtežji podani primer velikosti 14x24 po 1000 iteracij s kolonijo mravelj. Vidimo lahko, da po 1000 ponovitvah in 16 sekundah še vedno nismo blizu rešitve. Iskanje prave rešitve bi verjetno trajala zelo dolgo, če bi ta sploh bila najdena s trenutno rešitvijo.



Slika 4: Primer3(14x24) iteracija 1000 s kolonijo mravelj

V praksi bi lahko tudi združil oba postopka za hitrejše iskanje. Prvo bi zagnal reševanje s pravili, nato pa nadaljeval s kolonijo mravelj. Verjetno bi lahko vsako generacijo nastavili vrednost fermonov za celice, ki jih že poznamo iz prvega postopka, na 1 (za belo) ali 0 (za črno), tako se bodo vedno izbrala prvo celice s vrednostjo fermona 1. Prav tako takrat ni potrebno testirati pravilnost celice, saj so že pravilne.

Viri

1. https://www.puzzle-nurikabe.com/ (16.8.2024)
2. https://www.conceptispuzzles.com/index.aspx?uri=puzzle/nurikabe/rules (16.8.2024)
3. https://en.wikipedia.org/wiki/Nurikabe_(puzzle) (16.8.2024)
4. https://github.com/Microsoft/nurikabe (16.8.2024)
5. Orodje za ročno ali avtomatsko reševanje ugank nurikabe, Kristjan Žagar, 2022
6. Reševanje igre Nurikabe s kolonijami mravelj, Tadej Bešenič, 2022
7. Solving Nurikabe with Ant Colony Optimization, Martyn Amos, Huw LLoyd, 2019