Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dejan Tesovnik

Samodejno reševanje slovenskih križank s pomočjo spleta

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Igor Kononenko

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi naj se izdela program, ki samodejno rešuje križanke v slovenskem jeziku s pomočjo spleta in ugankarskega slovarja. Za zanesljivejše izpolnjevanje polj naj se izdela algoritem za ocenjevanje pravilnosti gesel v ugankarskem slovarju. Izdela naj se tudi slovar za hitro iskanje delno izpolnjenih namigov. Program naj bo preverjen na obsežni bazi slovenskih križank. Rezultate analizirajte in primerjate z ostalimi raziskavami s tega področja.

Iskreno se zahvaljujem mentorju profesorju dr. Igorju Kononenku za nasvete, strokovno pomoč ter odzivnost pri izdelavi diplomske naloge.

Ines, Ajda, Tanja in Anže, hvala vam za vso pomoč, podporo in spodbudo.

Kazalo

Povzetek

Naslov: Samodejno reševanje slovenskih križank s pomočjo spleta

Avtor: Dejan Tesovnik

Cilj diplomske naloge je bil izdelati program, ki samostojno rešuje slovenske križanke s pomočjo ugankarskega slovarja in spletnega iskalnika. Za izdelavo programa je bila uporabljena baza 367 križank in ugankarski slovar s 104.132 gesli, slovar spletnih odgovorov pa je bil izdelan s pomočjo spletnega iskalnika. Zaradi ogromnega števila spletnih odgovorov bi bilo preiskovanje celotnega prostora prepotratno. Reševanje križanke se zato izvede v dveh ločenih fazah. Program v prvi fazi delno reši križanko le s pomočjo ugankarskega slovarja. V tej fazi program najde najprimerjnejše odgovore za vsak namig v križanki in z njimi izpolni križanko. V drugi fazi program vključi v reševanje tudi spletne odgovore ter tako pride do končne rešitve. Program je bil razvit s pomočjo učne množice 37 križank, uspešnost programa pa je bila preizkušena na testni množici 330 križank.

Ključne besede: križanke, preiskovalni algoritmi, avtomatsko reševanje, slovar spletnih odgovorov.

Abstract

Title: Automated Slovene crosswords solving with the help of the internet

Author: Dejan Tesovnik

The goal of the thesis was to create a program, that automatically solves Slovene crosswords with the help of puzzle dictionary and web search. For the purpose of creating the program, we used a database of 367 crosswords and puzzle dictionary with 104.123 clue answers. We also created a list of web answers for all the crosswords in the database. Because of the enormous amount of the web answers searching the whole solution space would not be feasible. That is why the program solves each crossword in two separate phases. In the first phase, a crossword gets partially solved with the help of the puzzle dictionary, which creates a list of most probable solution candidates for each clue. In the second phase, a crossword is solved with puzzle dictionary and web answers combined. For the purpose of building the program, we used a set of 37 training crosswords. The final results were obtained from a set of 330 testing crosswords.

Keywords: crosswords, search algorithms, automatic solving, web search dictionary.

Poglavje 1

Uvod

Reševanje križank je eden izmed najbolj razširjenih načinov preganjanja dolgčasa, ki poleg tega tudi nadgrajuje naše znanje in nas navdaja z veseljem, ko dokončamo katero izmed še posebno težkih križank. Kljub popularnosti miselne igre je ta dokaj mlada. Začetki segajo v konec osemnajstega stoletja, na popularnosti pa so križanke začele pridobivati v začetku dvajsetega stoletja, ko so jih začeli tedensko objavljati v ameriškem časopisu New York World [6]. Križanke so bile takrat seveda sestavljene še ročno. Dandanes je to izjemna redkost, saj je prihod računalnikov močno olajšal gradnjo križank.

Izjemno zanimivo je, da računalnikom še ni uspelo premagati človeka pri reševanju le teh [3]. Trenutni računalniki namreč še niso dovolj dobri, da bi se lahko primerjali z najboljšimi reševalci križank na svetu. Računalnikom za uspešno reševanje manjkajo predvsem humor, sposobnost poigravanja z besedami in izkušnje pri reševanju, ki jih pridobimo ljudje. Z napredkom tehnologije pa bodo verjetno računalniki tudi na tem področju prevzeli vodstvo, kot so ga že pri marsikateri miselni igri.

1.1 Motiv za izdelavo diplome

Pri pregledu področja algoritmov za reševanje križank smo opazili, da so nekatere raziskave že bile opravljene [1, 2, 4]. Večina se jih ni lotila reševanja

s pomočjo spleta, nobena pa se ni omejila na slovenske križanke, zato tukaj vidimo možnost, da se s to diplomo doprinese nove ugotovitve in rezultate. Prav tako je problem izredno zanimiv za implementacijo, kar nas je še dodatno motiviralo za izdelavo diplome.

1.2 Cilji

Cilj te diplomske naloge je izdelati učinkovit in natančen program, ki s pomočjo spleta in slovarja poskuša poiskati najboljšo možno rešitev podanih križank. Da bi to dosegli, smo nalogo razdelili na tri ločene enote.

Namen prvega dela je bil za vsak namig v križanki poiskati najverjetnejša gesla v ugankarskem slovarju.

V drugem delu je bil cilj za vsak namig v križanki pridobiti čim bolj točne dogovore s pomočjo spletnih iskalnikov.

Cilj tretjega dela je bil izdelati algoritem, ki s pomočjo prej pridobljenih rešitev namigov najde kar se da pravilno rešitev križanke. Poleg učinkovitosti smo želeli doseči tudi, da bi program rešitev našel v doglednem času.

Poglavje 2

Pregled področja in dosedanjih raziskav

2.1 Križanke

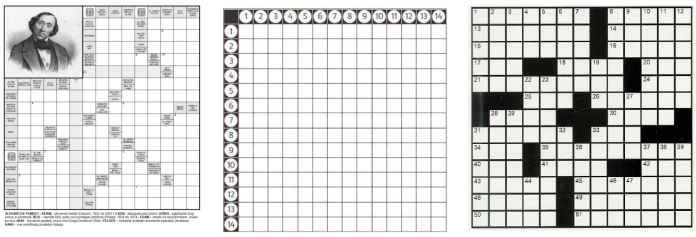
Križanka je vrsta uganke, ki jo predstavlja mreža polj. V prazna polja je potrebno vpisovati črke, ki vodoravno in navpično tvorijo odgovore na namige. V križankah se lahko nahajajo tudi polna polja, v katere se ne sme vpisovati črk. Ta polja so v nekaterih vrstah križank lahko poleg omejevanja mreže namenjena tudi prostoru, kjer so napisani namigi.

2.1.1 Vrste križank

Obstaja veliko število različnih vrst križank. Te se lahko razlikujejo po oblikah, načinih navajanja namigov, načinih izpolnjevanja in še bi lahko naštevali. V nadaljevanju sledi predstavitev nekaj najbolj razširjenih vrst križank.

Skandinavska križanka

Najpogostejša oblika križanke v Sloveniji, kot tudi v nekaterih drugih evropskih državah, je vsekakor skandinavska križanka. Najdemo jo lahko v skoraj



Slika 2.1: Primeri najpogostejših vrst križank. Od leve proti desni: skandinavska križanka, italijanska križanka in ameriška križanka.

da vsaki reviji ali časopisu. Za skandinavsko križanko je značilno, da se namigi nahajajo neposredno v križanki. Ti se nahajajo v poljih, v katera ne pišemo črk. V nekaterih križankah namige nadomestijo tudi slike.

Italijanska križanka

Italijanske križanke se od ostalih razlikujejo po tem, da mora reševalec sam izpolniti tako prazna polja, kot tudi potemniti polja, kjer ni črk. Namigi so včasih podani v pravilnem vrstnem redu, lahko pa so podani tudi povsem naključno.

Ameriška križanka

Za ta tip križanke je značilno, da je mreža sestavljena iz praznih in polnih polj. Črke lahko vpisujemo samo v prazna polja. V nekaterih praznih poljih pa so napisane številke, ki predstavljajo številko namiga, ki pripada polju. Namigi so napisani ob križanki in so ločeni na vodoravne in navpične.

2.2 Dosedanje raziskave

V literaturi je mogoče najti kar nekaj različnih načinov, na katere so se raziskovalci lotili reševanja križank [1, 2, 4, 3]. Vsi so dosegli dokaj dobre

rezultate, ki se lahko kosajo s človeškimi, vendar računalnik še vedno ne zmore premagati najboljših ugankarjev.

Avtorji članka Solving Crossword Puzzles as Probabilistic Constraint Satisfaction (Proverb) [5] so se lotili problema z ločevanjem iskanja odgovorov na različne module, ki so bili specializirani za točno določene odgovore. Sistem WebCrow [1] je bil zgrajen na podobni ideji, le da so med module dodali še modul za iskanje spletnih odgovorov. Avtorji članka A Genetic-Algorithm Approach to Solving Crossword Puzzles [4] so se problema lotili na bolj inovativen način in sicer z uporabo genetskih algoritmov. V nadaljevanju smo na kratko opisali vse omenjene pristope.

2.2.1 Proverb

Projekt Proverb [2, 5] avtorji opisujejo kot prvi program, ki učinkovito rešuje križanke s pomočjo specializiranih modulov. Moduli vračajo seznam besed, ki naj bi bile najbolj verjetni odgovori na podani namig. Nekateri moduli podajo poleg vsake možne rešitve tudi verjetnost, da je pravilna.
Nekaj primerov uporabljenih modulov:

- Moduli na podlagi seznama besed
  - besede iz celotnega slovarja, ki imajo točno dolžino
- Moduli na podlagi slovarja križank
  - besede, katerih opis se točno ujema z namigom
- Moduli za iskanje informacij v dokumentih
  - besede z delnim ujemanjem namiga
  - besede na podlagi iskanja sopomenk v namigu
  - seznam najbolj pogostih besed, ki so se pojavljale v člankih z besedami namiga
- Moduli na podlagi baz podatkov
  - filmi – iskanje naslovov in igralcev na strani www.imdb.com
  - glasba, literatura in geografija iz enciklopedij in spleta

- Skladenjski moduli
  - odgovori na vprašanja s praznim prostorom za odgovor (Fill-in-the-blanks)
  - odgovori na vprašanja tipa "KindOf"– deluje podobno kot Fill-in-the-blanks, uporablja le druge vire informacije

Pred začetkom reševanja se s pomočjo modulov za vsak namig naredi seznam besed, ki so potem kandidati za možne rešitve. Algoritem za iskanje rešitev križank je bil zasnovan na način, da poišče najbolj verjetna prekrivanja polj in potem preko večih iteracij poskuša najti najverjetnejšo rešitev.

Testne križanke so pridobili iz sedmih različnih virov. Skupaj jih je bilo 370, od tega jih je bilo 70 izbranih za treniranje modulov in reševanja. Rezultati so bili zelo spodbudni, saj algoritem v povprečju pravilno reši 95,3% vseh besed v lahkih križankah in 89,5% vseh besed v težkih križankah. Avtorji trdijo, da je rezultat boljši od povprečnega reševalca, a vseeno slabši od profesionalnega.

2.2.2 WebCrow

Pri projektu WebCrow [1] so se lotili reševanja križank s pomočjo spleta. Osredotočili so se na izključno križanke v italijanskem jeziku. V raziskavi so uporabili 685 križank iz revije La Settimana Enigmistica in časopisa La Repubblica, izmed katerih so jih 60 naključno izbrali za testiranje.

Sistem se je zgledoval po projektu Proverb in je za iskanje odgovorov uporabil specializirane module, vendar za razliko od Proverba ne uporablja nekaterih specifičnih modulov. Uporabili so več modulov, ki so odgovore iskali po različnih ugankarskih slovarjih, naredili pa so tudi poseben modul, ki je odgovore iskal po spletu. Za iskanje odgovorov v spletnem modulu so uporabili iskalnik Google™. Sestavili so tri različne poizvedbe, s čimer so želeli povečati verjetnost, da bo med zadetki pravilen odgovor.

Tipi spletnih poizvedb:

- AND operator za vse besede v vprašanju
- OR operator za vse besede v vprašanju
- OR operator za vse besede v vprašanju z različnimi spoli in števili

Ker je iskanje po spletu dokaj zamudno, so si rezultate poizvedb shranili na trdi disk, kar je pohitrilo in olajšalo delo modulu. Najdeni odgovori so bili razporejeni glede na verjetnost, da je odgovor pravilen. To so dosegli s preverjanjem pogostosti vsake besede v posameznih dokumentih.

Verjetnost, da je spletni modul podal pravilen odgovor v seznamu vseh odgovorov, je bila 68%, v 13,5% pa je bil pravilen odgovor na prvem mestu. Spletni modul je za vsako poizvedbo podal 499 možnih odgovorov. Modul na osnovi ugankarskega slovarja je pravilen odgovor podal v 71,1%, v 0,4% pa je bil pravilen odgovor na prvem mestu. Ta modul je za vsako poizvedbo podal čez 1000 možnih odgovorov. Ko so združili sezname iz vseh modulov, je združen seznam vseboval pravilno besedo v 99,3%.

Reševanje vsake posamezne križanke so omejili na 15 minut. WebCrow je v povprečju uspel pravilno rešiti 68,8% vseh besed in 79,9% vseh črk.

2.2.3 Reševanje križank s pomočjo genetskih algoritmov

Razsikava A Genetic-Algorithm Approach to Solving Crossword Puzzles [4] se je lotila problema na povsem drugačen način in sicer s pomočjo genetskih algoritmov.

Avtorja sta se osredotočila na enostavnejše verzije križank, kjer so vse rešitve namigov enako dolge. Takšne križanke imajo manj polj, kjer se namigi križajo in so s tem bolj omejene, kar močno olajša reševanje.
Glede na to, da je bila raziskava narejena leta 1993, predvidevamo, da so si izbrali takšne križanke iz praktičnih razlogov. Takrat namreč računalniki še niso imeli dovolj hitrih procesorjev in velikega pomnilnika, da bi lahko uspešno reševali večje križanke. Izbira pristopa z genetskim algoritmom je še

dodatno omogočila, da program pride do dobrih rešitev v doglednem času. Cena tega pristopa je, da ne zagotavlja optimalne rešitve.

Za iskanje odgovorov na namige so uporabili poseben leksikon, ki jim je omogočal direkten pristop do besed s pravimi črkami na določenih mestih. Leksikon je bil izračunan in sestavljen pred začetkom reševanja križanke.

Genetski algoritmi omogočajo, da se reševanja težjih problemov lotimo s principi evolucije, kjer se boljše rešitve križajo v naslednjo generacijo rešitev, slabše pa odpadejo. Rešitev lahko v tem primeru poimenujemo kar gen. V tej raziskavi so bili geni sestavljeni iz zaporedja besed, ki so na koncu predstavljale rešitev križanke. Mutacija med iteracijami je bila izvedena tako, da je na naključnih genih izbrala naključno besedo in jo zamenjala z drugo, ki je ustrezala postavitvi presekov ostalih besed.

Avtorja sta bila z rezultatom zadovoljna. Pristop z genetskimi algoritmi se je po uspešnosti primerjal z deterministčnimi pristopi, ampak je bil v povprečju vseeno slabši. Program so več kot stokrat pognali na eni sami križanki. Povprečna rešitev križanke je bila 91,17% pravilnih črk. Algoritem ni uspel rešiti križanke bolje, kot so jo bralci revije, v kateri je bila objavljena.

Poglavje 3

Opis podatkov in uporabljena orodja

V tej raziskavi smo se omejili na ameriške in skandinavske križanke v slovenskem jeziku. Ta dva tipa križank sta si dokaj podobna, kar nam omogoča, da lahko uporabimo večjo zbirko križank za testiranje. Omejitev na slovenski jezik je smiselna zato, ker do sedaj še nobena raziskava ni bila opravljena na izključno slovenskih križankah.

3.1 Uporabljena orodja

Za izdelavo naloge smo izbrali razvojno okolje *Eclipse IDE*. Za programski jezik smo si izbrali Javo. Deloma zato, ker je za ta jezik že spisanih veliko uporabnih knjižnic ter zaradi pozitivnih izkušenj na preteklih projektih.

Algoritem za reševanje križank smo napisali v celoti, brez uporabe zunanjih knjižnic. Pomagali smo si le s knjižnico *JFreeChart*, ki nam je pomagala vizualizirati podatke med testiranjem. Pri izdelavi grafičnega okolja za testiranje reševanja križank, smo uporabili vtičnik *WindowBuilder*, s katerim smo si olajšali postavitev uporabniškega vmesnika. Za pridobivanje spletnih odgovorov nam je delo olajšala knjižnica *jsoup*, s katero smo uspešno izluščili podatke iz spletnih strani. Delo nam je olajšal tudi program *Bless Hex Edi-*

tor, brez katerega bi težko razvozlali podatkovno strukturo spletnih križank. Za končno vizualizacijo rezultatov reševanja smo si pomagali s programskim jezikom *R* in programom *LibreOffice Calc*.

3.2 Baza križank

Da bi bili rezultati raziskave kar se da verodostojni, smo poskušali poiskati čim večjo bazo križank. Želeli smo tudi, da so si križanke med seboj čim bolj različne, tako po velikosti, kot tudi po postavitvi namigov, poleg vprašanj pa smo potrebovali še rešitve, s katerimi smo lahko preverili uspešnost reševanja. Obsežno zbirko slovenskih križank, ki ustreza zgornjim kriterijem, smo našli na strani `http://www.mojnet-si.net` in jo, z dovoljenjem lastnice strani, tudi uporabili v naši raziskavi.

Celotna baza vsebuje 367 križank. Križanke so velikosti od 5x8 pa vse do 27x27 in imajo poleg vprašanj podane tudi pravilne odgovore. Datoteke so tipa .ccj, ki so namenjene programu *Crossword Compiler*, ki pa je na žalost komercialni program in si z njim nismo mogli pomagati. Z veliko mero potrpežljivosti nam je uspelo datoteke dešifrirati in pretvoriti v format, ki je združljiv z našim programom.

3.3 Ugankarski slovar

Pri reševanju križank smo si pomagali z ugankarskim slovarjem, ki smo ga z dovoljenjem podjetja EZI Software pridobili na strani `http://www.4ezi.com/ezisoftware/projects/krizanke/slovar.php`. V slovarju je 104.132 gesel z opisi. Pri analizi namigov iz križank in odgovorov slovarja smo ugotovili, da se odgovori točno ujemajo z namigi v križankah le v 5% vseh gesel. To je dober znak, da je bil slovar pridobljen iz drugih virov kot baza križank, kar dodatno poveča nepristranskost raziskave.

3.4 Slovar spletnih odgovorov

Za hitrejše in neovirano delovanje programa smo se odločili, da rezultate spletnega iskanja na namige shranimo v lokalno bazo. To nam omogoča, da program dostopa do podatkov brez spletne povezave in zmanjša število zahtev na spletne iskalnike – število teh je namreč časovno omejeno.

Po kratkem testiranju različnih iskalnikov smo imeli namen uporabiti iskalnik Google™. Zaradi velikega števila indeksiranih slovenskih strani so bile poizvedbe na tem iskalniku najbolj relevantne. Na žalost smo kmalu ugotovili, da ne podpira javnega brezplačnega dostopa do rezultatov iskanja. Poskusili smo z izluščevanjem podatkov direktno iz spletne strani, a nismo bili uspešni, saj je število zahtev na minuto prenizko, da bi lahko v doglednem času pridobili odgovore na vse namige.

Končna baza odgovorov je nastala s pomočjo iskalnika *DuckDuckGo*. Ta sicer podpira javni vmesnik do rezultatov iskanja, vendar vrača preveč okrnjene rezultate, da bi bili uporabni. Podatke smo zato pridobili z direktnim izluščevanjem s spletnih strani. V povprečju je vsak namig s spletno poizvedbo dobil 3719 možnih odgovorov, kar je veliko več kot pri iskalniku Google™ – tam jih je bilo okoli 200. Po pregledu odgovorov smo ugotovili, da je med njimi večje število tujih besed, kar je rezultat slabše podpore slovenskih strani v iskalniku. Večje število odgovorov v tem primeru ne pomeni večje verjetnosti, da je med njimi pravilen odgovor, ampak le večje število neustreznih odgovorov.

Poglavje 4

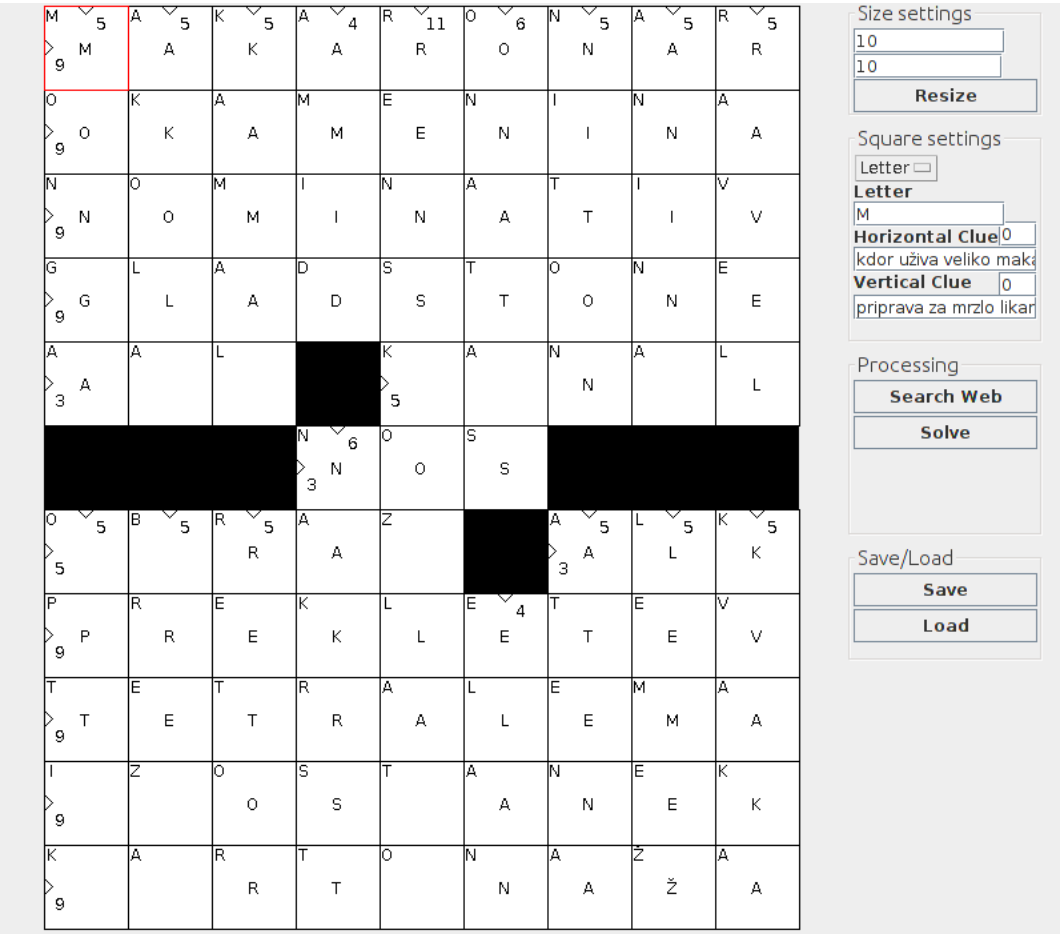
Reševanje križank

4.1 Izdelava grafičnega vmesnika

Za lažje testiranje in prikaz reševanja smo se odločili, da vložimo čas tudi v izdelavo grafičnega uporabniškega vmesnika. Izdelave smo se lotili z *Eclipse IDE* vtičnikom *WindowBuilder*, s katerim smo postavili vse potrebne elemente.

Izgled grafičnega vmesnika je prikazan na sliki 4.1. Na desni strani je postavljen meni, kjer je mogoče spreminjati dimenzije križanke ter njenih polj kot tudi shranjevanje in nalaganje križanke iz datotek.

Za prikaz križanke je narejen poseben element, ki na enostaven način prikaže izgled križanke. Z njim je prav tako možno urejati križanko. Polna polja predstavljajo blokade, prazna pa vnosna polja. V praznih poljih je v zgornjem levem kotu izpisana pravilna rešitev polja, medtem ko puščice na robovih predstavljajo, če ima polje dodeljen horizontalni ali vertikalni namig. S klikom na polja križanke se v stranskem meniju prikažejo lastnosti izbranega polja. S tem se lahko urejajo vertikalni in horizontalni namigi, rešitev polja in tip polja.



Slika 4.1: Izgled grafičnega vmesnika med reševanjem križanke.

4.2 Priprava strukture slovarjev za hitrejše iskanje odgovorov

Narava problema reševanja križank od nas zahteva, da po slovarju možnih odgovorov ne iščemo le besede, ki se začnejo ali končajo na določeno črko, temveč tudi besede, ki vsebujejo točno določene črke na točno določenih mestih. S sprotnim izpolnjevanjem križanke lahko za delno izpolnjene odgovore na namige hitro izključimo besede, ki jih ni možno vstaviti. To zahteva, da je operacija iskanja po slovarju odgovorov izredno pogosta, kar pomeni, da

mora biti le ta kar se da učinkovita. Pri velikem številu možnih odgovorov za vsak namig dokaj hitro odpade opcija slepega preiskovanja celotnega slovarja. Za potrebe naše naloge smo se zato lotili izdelave posebnega slovarja, ki omogoča hitro iskanje besed s črkami na točno določenih mestih.

Izdelave slovarja smo se lotili s pomočjo posebnega drevesa. Primer drevesne strukture slovarja lahko najdemo na sliki 4.2. Vsakemu vozlišču, razen korenskemu, pripada svoja črka. Drevo sestavimo tako, da se pri vstavljanju besed generirajo vozlišča s pripadajočimi črkami. Na primer, če želimo vstaviti besedo NOGA, se korenskemu vozlišču doda naslednik s črko N. Temu se potem doda naslednik O in tako naprej do konca besede. Če sledimo vozliščem po hierarhiji navzdol, s tem sestavljamo besede v slovarju. V primeru, da vozlišče z določeno črko že obstaja, ne naredimo novega, ampak uporabimo obstoječega. Ko so v drevo dodane vse besede, dodamo vsakemu vozlišču povezavo do vseh njegovih potomcev, razporejenih po nivojih. S tem lažje ugotavljamo, na katerih mestih so na voljo določene črke.

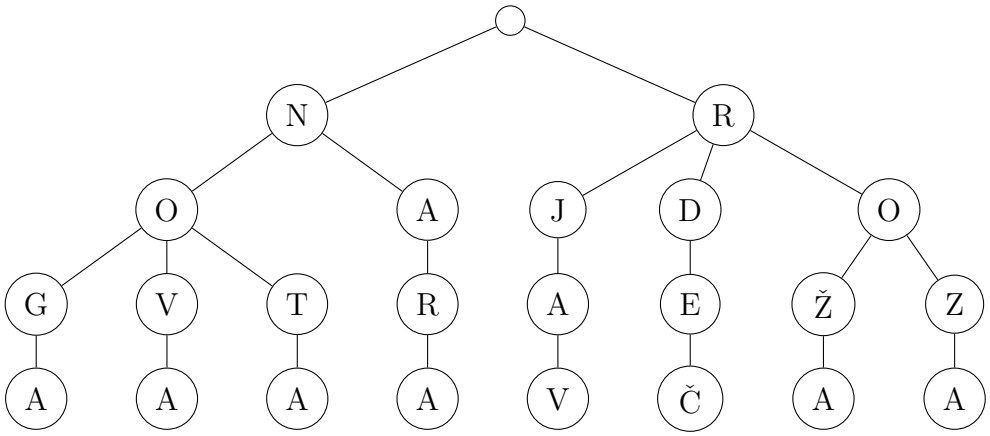
Prednost tega pristopa je ta, da nam ni potrebno preiskovati celotnega prostora, da ugotovimo, če imamo na voljo besedo, ki jo je možno vstaviti v delno izpolnjena polja. To močno poveča hitrost iskanja, še posebej, če je seznam besed velik. Slabost tega pristopa je, da porabimo za drevo veliko več prostora, kot s samim seznamom besed.

4.3 Vrednotenje odgovorov v ugankarskem slovarju

Dobra lastnost ugankarskega slovarja je, da poleg gesel vsebuje tudi njihove opise. Opisi gesel se ponavadi popolnoma ali delno ujemajo z namigi v križankah, s čimer si lahko pomagamo pri ocenjevanju ustreznosti odgovorov. Da bi to preverili, smo se odločili izdelati algoritem, s katerim bi lahko primerjali podobnosti stavkov gesel in namigov.

Problema smo se na začetku lotili z uporabo obstoječih algoritmov, a smo

Slika 4.2: Primer drevesne strukture izdelanega slovarja za namig, ki išče odgovor dolžine štirih črk. Slovar vsebuje besede: NOGA, NOVA, NOTA, NARA, RJAV, RDEČ, ROŽA, ROZA. Vozlišča lahko direktno dostopajo do svojih potomcev, kar pospeši iskanje delno izpolnjenih polj namiga.



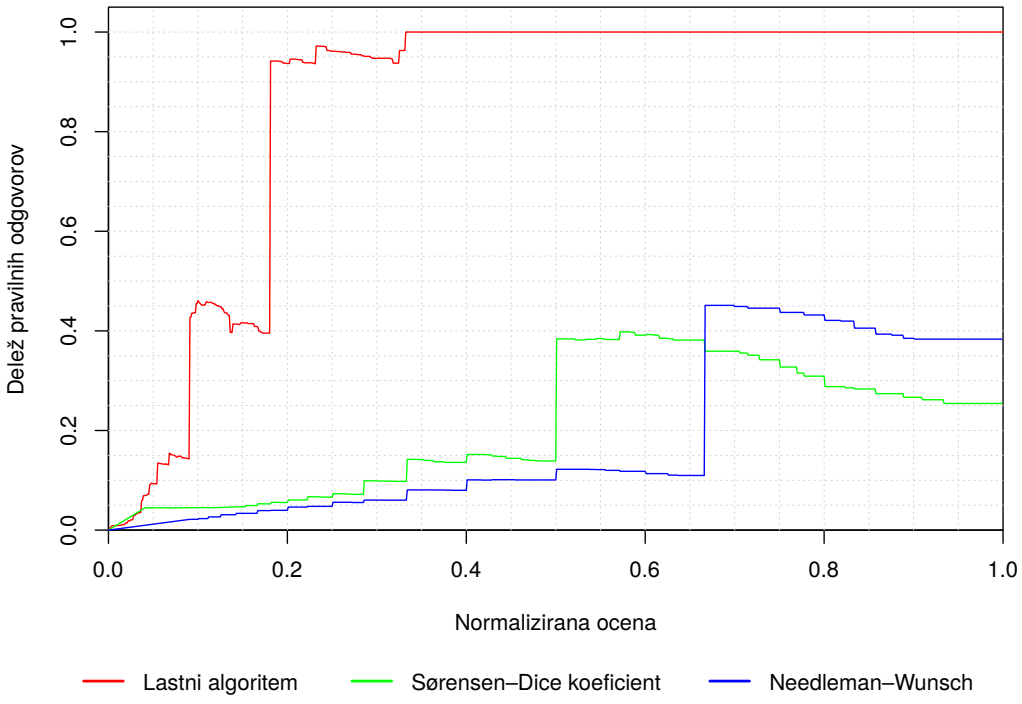
na koncu izdelali lasten algoritem, s katerim smo dosegli boljše rezultate.

Needleman–Wunsch [7] se ponavadi uporablja v bioinformatiki za primerjanje podobnosti zaporedij v genih. Algoritem smo prilagodili, da namesto zaporedij v genih primerja zaporedja besed v stavkih.

Sørensen–Dice koeficient [8] se ponavadi uporablja za oceno podobnosti dveh množic. Algoritem smo prilagodili do te mere, da elemente v množici predstavljajo besede stavkov.

Lastni algoritem se je izkazal za najbolj uspešnega. Algoritem primerja ujemanje besed med dvema množicama, poleg tega pa preveri tudi njihovo zaporedje. S tem so bolje ocenjeni stavki, ki imajo poleg enakih besed le-te tudi v enakem zaporedju.

Namigi, na katerih so narejeni testi ocenjevanja, so pridobljeni iz učne množice križank, ki jo sestavlja 37 križank. S tem smo hoteli zagotoviti, da ocen ne bi prekomerno prilagodili na celotno množico križank. Primerjava uspešnosti algoritmov je prikazana na sliki 4.3.



Slika 4.3: Primerjava uspešnosti algoritmov za vrednotenje odgovorov v ugankarskem slovarju.

4.4 Potek reševanja

Reševanje križanke smo se odločili razdeliti na dva dela. Razlog je v tem, da lahko iz ugankarskega slovarja dobimo veliko več informacij o pravilnosti odgovorov, kot iz spletnega slovarja. Križanko zato v prvem delu rešimo izključno s pomočjo ugankarskega slovarja. Končni rezultat prvega dela smo poimenovali *nastavek rešitve*. V drugem delu za reševanje uporabimo oba slovarja in pridemo do končne rešitve.

Pri izdelavi programa, smo si pomagali izključno s križankami iz učne množice. S tem smo si želeli zagotovili, da programa ne bi prekomerno prilagodili. Da bi bile križanke v tej množici čim bolj raznolike, smo izbrali 37 križank, s katerimi smo zajeli različne velikosti, oblike in zahtevnosti.

4.4.1 Priprava slovarjev

Pred pričetkom reševanja baze križank smo na učni množici križank pognali algoritem, ki je našel mejo ocene, s katero smo iz ugankarskega slovarja zajeli vsaj 80% vseh pravilnih odgovorov.

To dosežemo tako, da za vsak namig v križanki poiščemo njegove možne odgovore z ocenami iz ugankarskega slovarja. Za vsak namig potem vzamemo oceno, s katero je bil ocenjen pravilen odgovor. Ko imamo ocene vseh pravilnih odgovorov, enostavno postavimo mejo, kjer bi jih 80% ostalo med možnimi odgovori. Postopek ponovimo za vse križanke v množici in na koncu vzamemo povprečje vseh mej. Primer postopka priprave meje je prikazan na sliki 4.4.

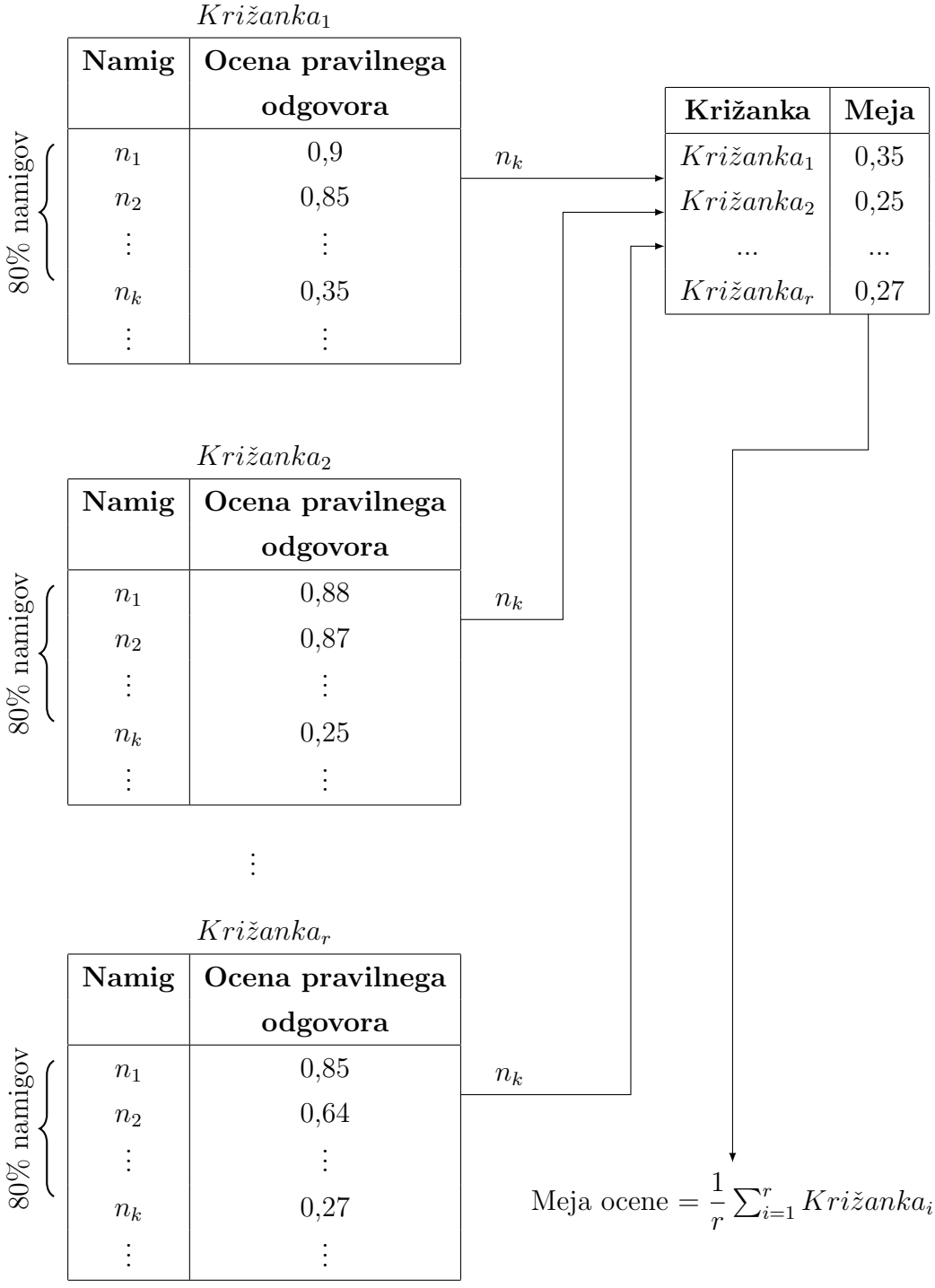
4.4.2 Sestavljanje nastavka rešitve

Reševanje prazne križanke, z velikim številom možnih odgovorov za vsak namig, se je izkazalo za izredno dolgotrajno. Zaradi tega razloga smo se lotili sestave nastavka križanke. S tem smo želeli čim hitreje poiskati delno rešitev križanke s čim višjo točnostjo odgovorov.

Pred začetkom sestavljanja nastavka za vsak namig v križanki poiščemo 3 najverjetnejše odgovore v ugankarskem slovarju. V tem koraku še ne odstranjujemo odgovorov, katerih ocena je pod mejo, saj so lahko vseeno pravilni in vodijo do pravilne rešitve. Iz dobljenih odgovorov potem za vsak namig pripravimo slovar za hitrejše iskanje.

V Algoritmu 1 je podan poenostavljen postopek sestave nastavka. V vrstici 13 algoritem poišče začetno rešitev, iz katere potem izhaja v naslednjih korakih. Na ta način dobimo dobro začetno izhodišče rešitve. Postopek iskanja rešitve je podan v vrstici 36. Metoda s pomočjo rekurzije preizkuša besede namigov in sproti shranjuje najboljšo rešitev križanke. Metoda je časovno omejena, da zaključi iskanje v primeru, ko nekaj časa ne najde boljše rešitve.

Iz izhodišča rešitve nadaljuje zanka v vrstici 22, ki najde najbolje ocenjeno



Slika 4.4: Prikaz priprave meje ugankarskega slovarja. V tem primeru n_k predstavlja namig, ki ima oceno slabšo kot 80% ostalih namigov v križanki.

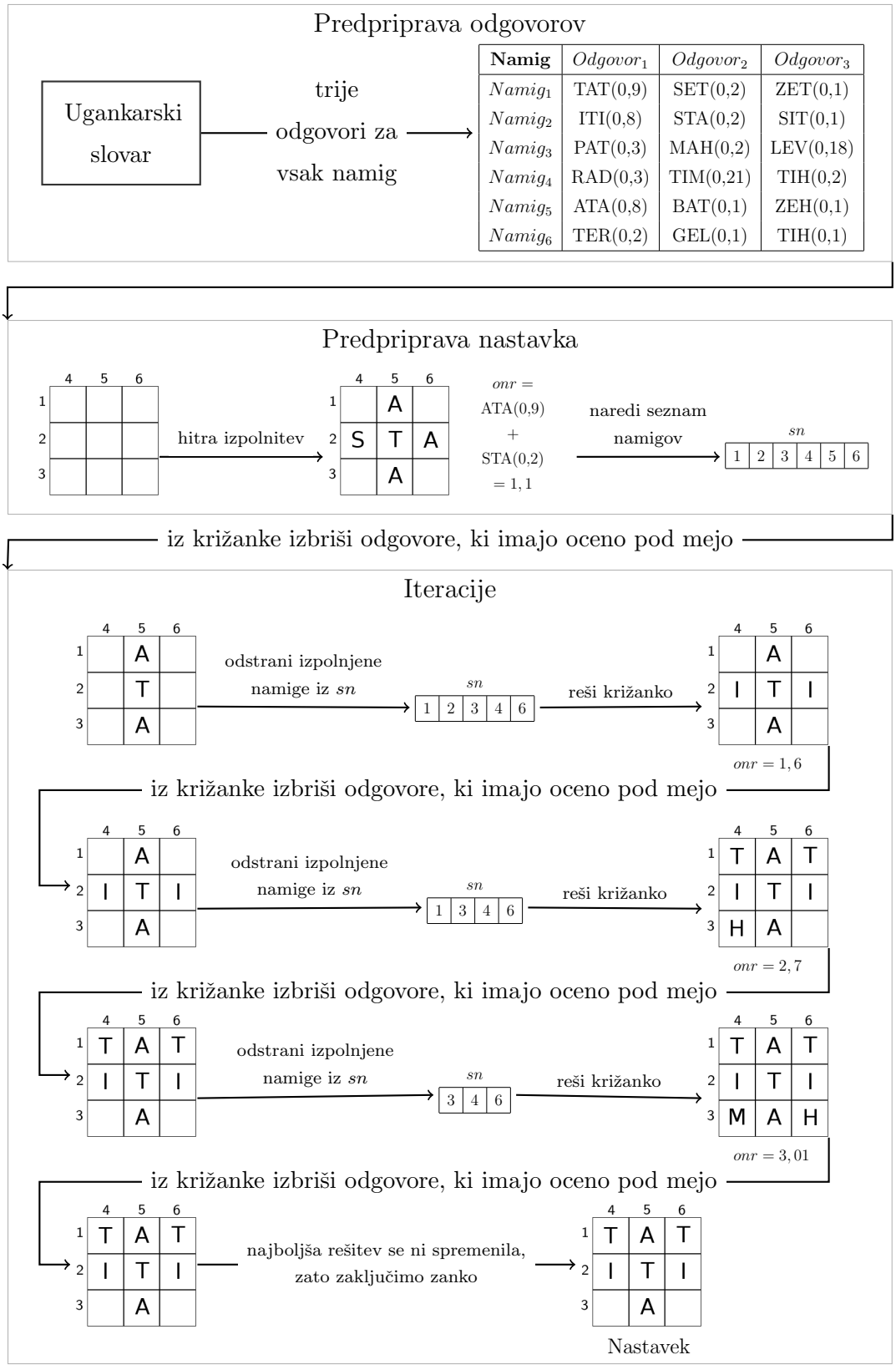
rešitev križanke. Pred vsako iteracijo, se iz najboljše rešitve križanke počisti odgovore, ki so pod predizračunano mejo. Na ta način dobimo nastavek samo z odgovori, ki so zelo verjetni, da so pravilni. Zanka se zaključi, ko noben namig nima več veljavnih odgovorov, ali pa se število namigov, ki so na voljo, ni spremenilo. Da se število namigov, ki so na voljo, ne spremeni, se zgodi le v primeru, da so ocene veljavnih odgovorov pod mejo. Rezultat algoritma je najboljša najdena rešitev križanke, ki se v nadaljevanju uporabi kot nastavek. Primer priprave nastavka na enostavni križanki je podan s sliko 4.5.

4.4.3 Iskanje končne rešitve

V algoritmu za iskanje končne rešitve smo namigom poleg odgovorov iz ugankarskega slovarja dodali še odgovore, najdene preko spleta. Zaradi velikega števila možnih odgovorov, ki jih doda spletni slovar, smo se iskanja rešitve lotili na drugačen način kot pri nastavku. Možnih odgovorov za vsak namig je lahko namreč tudi po več tisoč, prav tako pa nimamo ocen odgovorov, s katerimi bi si lahko pomagali pri iskanju rešitve.

V Algoritmu 2 je podan poenostavljen postopek iskanja končne rešitve. V začetno funkcijo dobimo nastavek iz Algoritma 1, ki ga uporabimo kot izhodišče za končno rešitev. Zanka v vrstici 13 ponavlja postopek dodajanja odgovorov v rešitev križanke, dokler noben izmed namigov nima več odgovorov, ki ustrezajo poljem v križanki. Z vsako iteracijo se izbere namige, ki imajo najmanj možnih kombinacij s pravokotnimi namigi. Za najbolje ocenjen namig se v seznam izbranih namigov doda še njegove pravokotne namige. Z izbranimi namigi algoritem poišče najboljšo rešitev križanke. Rešitev namiga, ki je imel najmanj kombinacij pred začetkom reševanja, nato zapiše v trenutno rešitev križanke.

Za iskanje rešitve algoritem uporabi funkcijo v vrstici 28. Funkcija se s pomočjo rekurzije sprehodi čez seznam izbranih namigov in vstavlja v križanko njihove odgovore. Najboljše ocenjena rešitev križanke se sproti shranjuje. Ocena rešitve križanke se pridobi s skupnim seštevkom izpolnjenih polj



Slika 4.5: Poenostavljen primer priprave nastavka križanke.

Algoritem 1 Priprava nastavka križanke

```
1: Spremenljivke
2: križanka "podana križanka"
3: nr "najboljša rešitev križanke"
4: tr "trenutna rešitev križanke"
5: onr "ocena najboljše rešitve"
6: sn "seznam namigov"
7: mo "meja ocene, pridobljena iz učne množice križank"
8:
9: function NAREDI NASTAVEK KRIŽANKE(križanka, mo)
10:     tr = križanka
11:     nr = križanka
12:     onr = ocena nr
13:     for vsak namig v križanki do
14:         REŠI KRIŽANKO(namig)
15:         if ocena tr je večja od onr then
16:             nr = tr
17:             onr = ocena nr
18:         end if
19:     end for
20:     sn = naredi seznam vseh namigov iz tr
21:     odstrani rešitve namigov iz nr, katerih ocena je pod mo
22:     repeat
23:         tr = nr
24:         iz sn odstrani vse namige, ki so brez odgovorov na tr
25:         for vsak namig v sn do
26:             REŠI KRIŽANKO(namig)
27:             if ocena tr je večja od onr then
28:                 nr = tr
29:                 onr = ocena nr
30:             end if
31:         end for
32:         odstrani rešitve namigov iz nr, katerih ocena je pod mo
33:     until nr se ni spremenila v zadnji iteraciji
34:     return nr
35: end function
```

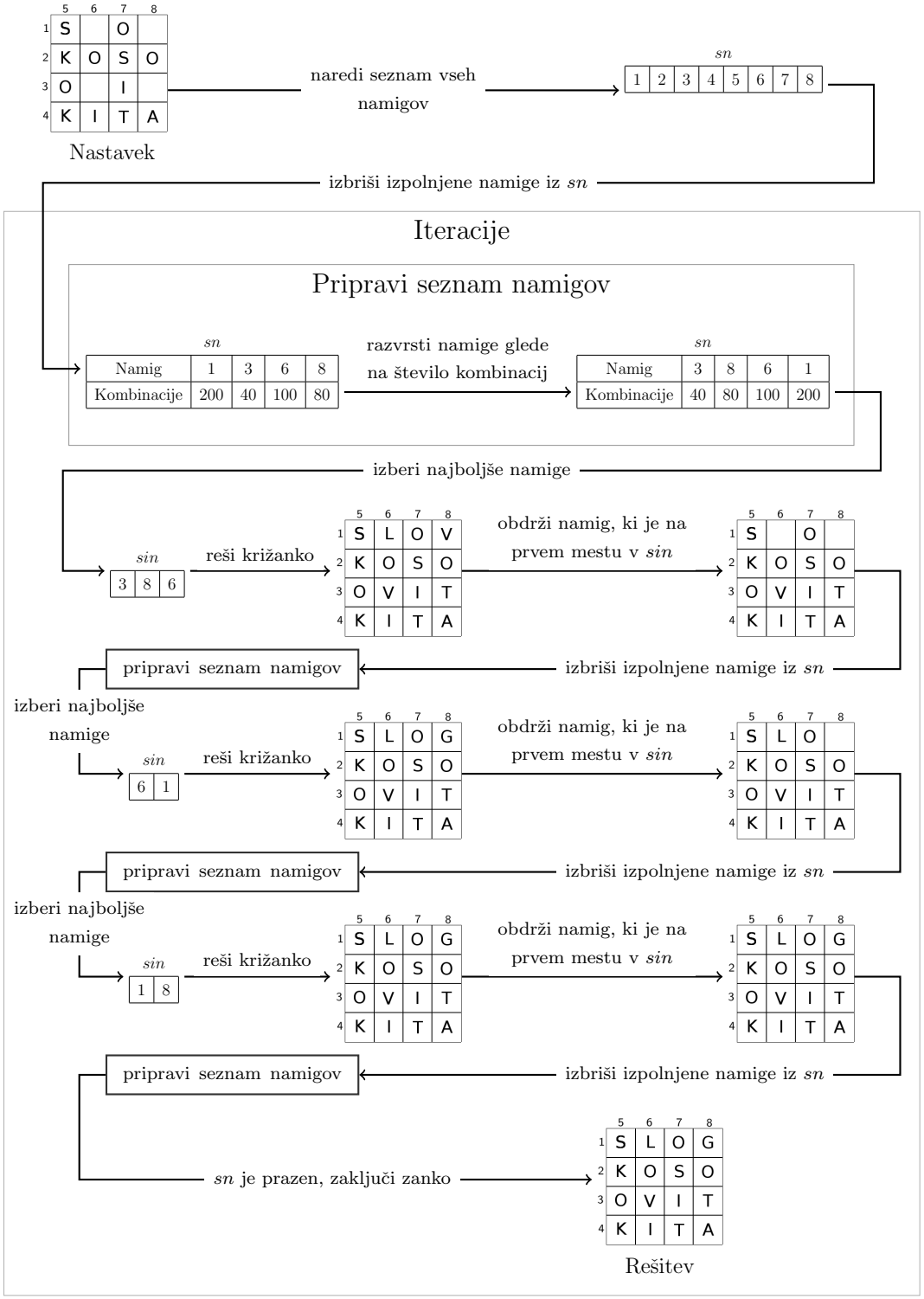
```
36: function REŠI KRIŽANKO(namig)
37:     if namig vsebuje odgovor, ki se prilega v polja tr then
38:         for vsak odgovor iz namiga, ki se prilega v polja tr do
39:             vstavi odgovor v polja namiga
40:             if ocena tr je boljša od nr then
41:                 nr = tr
42:             end if
43:             for vsak pravokotninamig na namig do
44:                 REŠI KRIŽANKO(pravokotninamig)
45:             end for
46:             odstrani odgovor iz polj namiga
47:         end for
48:     end if
49: end function
```

– več kot je izpolnjenih polj, boljša je rešitev. Primer iskanja končne rešitve na enostavni križanki je razviden na sliki 4.6.

4.4.4 Časovna zahtevnost reševanja

Časovna zahtevnost brez vseh omejitev je pri problemu reševanja križank izjemno zahtevna. Zahtevnost reševanja križanke s surovo močjo v najslabšem primeru ocenjujemo na $O(c^n)$, kjer je c povprečno število odgovorov na namig, n pa je število vseh namigov. Ker v naši raziskavi namigi vsebujejo tudi po več tisoč možnih odgovorov, se takšen način reševanja hitro izkaže za izredno neučinkovitega.

Časovno zahtevnost za pripravo nastavka v najslabšem primeru smo ocenili na $O(3^n)$, kjer je n število namigov v križanki, 3 pa je konstantno število odgovorov za vsak namig. Algoritem je v praksi porabil precej manj časa, saj smo funkcijo za iskanje rešitve časovno omejili. Časovna omejitev deluje tako, da zaključi funkcijo za iskanje rešitve za trenutno iteracijo, če ta določen čas ne vrne boljše rešitve. Ta pristop iskanja nam je omogočil doseči empirično časovno zahtevnost $O(n)$ (razvidno na sliki 4.7) v križankah, kjer



Slika 4.6: Poenostavljen primer iskanja končne rešitve križanke. Za preglednejši prikaz postopka smo proceduro "Pripravi seznam namigov"podrobneje prikazali samo enkrat.

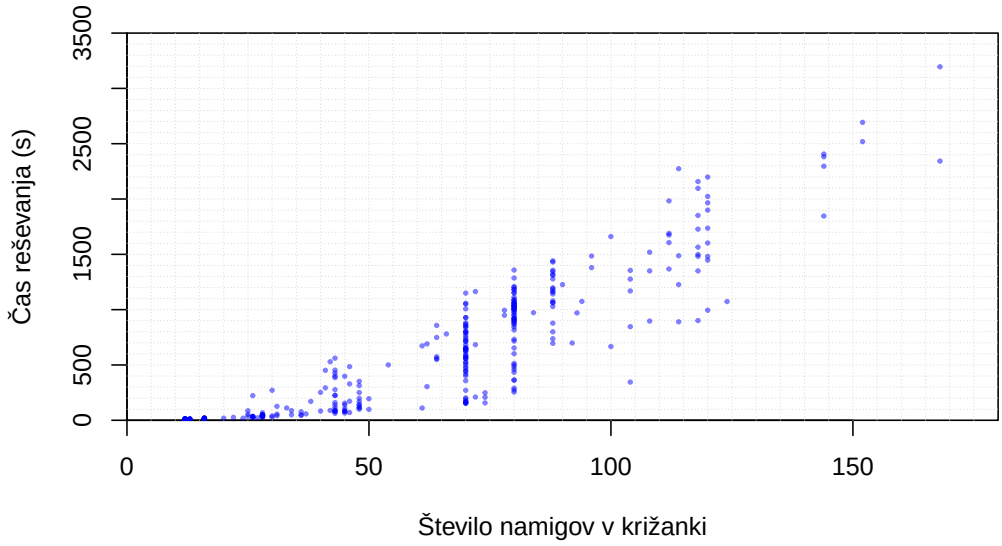
Algoritem 2 Iskanje končne rešitve križanke

```
1: Spremenljivke
2: nastavek "nastavek križanke pridobljen z Algoritmom 1"
3: nr "najboljša rešitev križanke"
4: tr "trenutna rešitev križanke"
5: sn "seznam namigov"
6: sin "seznam izbranih namigov za trenutno iteracijo"
7: rn "rešen namig"
8:
9: function NAJDI KONČNO REŠITEV KRIŽANKE(nastavek)
10:     tr = nastavek
11:     nr = tr
12:     sn = naredi seznam vseh namigov iz tr
13:     repeat
14:         iz sn izbriši rešene namige in namige brez odgovorov
15:         razvrsti namige v sn glede na število kombinacij z ostalimi namigi
16:         iz sn izberi namige z najmanjšim možnim število kombinacij in jih zapiši v sin
17:         nr = tr
18:         REŠI KRIŽANKO(0)
19:         rn = prvi namig iz sin, ki ima v nr rešena vsa polja
20:         zapiši rešitev rn v tr
21:         izbriši rn iz sn
22:         for vsak namig v sn do
23:             izbriši odgovore iz namiga, ki ne ustrezajo poljem v tr
24:         end for
25:     until sn je prazen
26:     return tr
27: end function
```

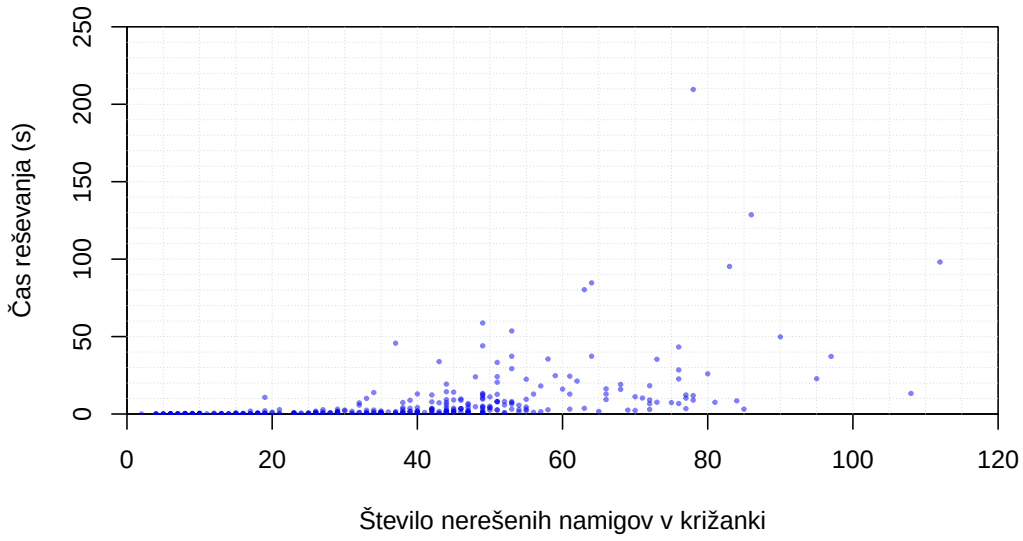
```
28: function REŠI KRIŽANKO(i)
29:     if i < dolžina sin then
30:         namig = sin[i]
31:         if namig vsebuje odgovor, ki se prilega v polja tr then
32:             for vsak odgovor iz namiga, ki se prilega v polja tr do
33:                 vstavi odgovor v polja namiga
34:                 if ocena tr je boljša od nr then
35:                     nr = tr
36:                 end if
37:                 REŠI KRIŽANKO(i+1)
38:                 odstrani odgovor iz polj namiga
39:             end for
40:         end if
41:     end if
42: end function
```

pride do časovne omejitve v vseh iteracijah. V praksi algoritem tako potrebuje časovno zahtevnost $O(3^n)$ le pri manjših križankah, pri večjih pa je ta $O(n)$.

Časovno zahtevnost algoritma pri iskanju končne rešitve smo za najslabši primer ocenili na $O(n^2)$, kjer je n število nerešenih namigov v križanki. Večje število nerešenih namigov prinese tudi večje število iteracij pri iskanju končne rešitve. Časovna zahtevnost vsake iteracije je prav tako odvisna od števila nerešenih namigov v križanki, kar pa naredi časovno zahtevnost algoritma kvadratno. Čas reševanja smo izboljšali s tem, da pri vsaki iteraciji algoritem izbere določeno število namigov z manjšim številom možnih odgovorov, kar zmanjša število kombinacij odgovorov, ki jih je potrebno preveriti. Zaradi tega razloga je kvadratna časovna zahtevnost v meritvah slabše opazna, a vseeno prisotna, slednje je tudi razvidno na sliki 4.8.



Slika 4.7: Časovne meritve izdelave nastavkov križank. Časovno zahtevnost v najslabšem primeru ocenjujemo na $O(3^n)$, vendar se je s pomočjo časovne omejitve funkcije za iskanje v praksi izkazala za $O(n)$.



Slika 4.8: Časovne meritve iskanja končne rešitve križank. Časovno zahtevnost ocenjujemo na $O(n^2)$.

Poglavje 5

Rezultati

5.1 Rezultati reševanja

Kot smo že omenili, smo bazo križank razdelili na dva ločena dela. Prvi del smo uporabili kot učno množico, s pomočjo katere smo izdelali program in na kateri smo tudi izračunali mejo ocene za ugankarski slovar. Učno množico sestavlja 37 križank različnih velikosti, oblik in težavnosti.

Drugo množico smo namenili testiranju programa in jo sestavlja preostalih 330 križank. Spodaj opisani rezultati reševanja so pridobljeni iz testne množice. Za izračun rezultatov smo uporabili računalnik z 8 GB pomnilnika in procesorjem Intel® Core™ i7-2600K. Za izračun končnih rezultatov vseh 330 križank je program porabil 59 ur in 50 minut.

Glavno oceno rešitve smo določili z odstotki pravilno rešenih namigov v križanki. Za to oceno smo se odločili zato, ker program izpolnjuje križanke na podlagi reševanja namigov s celimi besedami in ne s posameznimi črkami. Poleg te ocene smo beležili tudi odstotke pravilno rešenih polj v križankah. Razlog je v tem, da že eno samo napačno polje v križanki ustvari dva nepravilno rešena namiga, medtem ko dve napačni polji predstavljata že tri ali štiri napačno rešene namige. Prikaz obeh ocen nam omogoča nazornejšo predstavitev rezultatov.

Rezultati raziskave so vidni v tabeli 5.1. Da je lažje razvidno, do kakšne

mere so bile križanke rešene v prvem in drugem koraku, smo podali rezultate nastavkov in končnih rešitev. Kot zanimivost pa smo navedli še polji "Povprečje vseh rešenih namigov"in "Povprečje vseh rešenih polj", v katerih smo podali povprečno oceno namigov in polj neodvisno od križank, katerim pripadajo.

Iz rezultatov lahko ugotovimo, da je odstotek pravilno rešenih namigov precej nižji kot odstotek rešenih polj. Kot smo že omenili, lahko eno napačno izpolnjeno polje ustvari več napačno izpolnjenih namigov, kar je tudi razlog za dokaj veliko razliko med ocenama.

Rezultati posameznih križank so razvidni na sliki 5.1. Na tej sliki smo rezultate vseh križank razporedili glede na odstotek pravilno rešenih namigov. Poleg ocene namigov, smo podali tudi oceno rešenih polj. Na sliki 5.2 so razvidni rezultati križank glede na njihovo velikost. Podali smo tako oceno namigov, kot posameznih polj v križankah.

Iz grafov lahko razberemo, da je program rešil 7 križank povsem pravilno, ena križanka pa je bila rešena brez pravilno rešenih namigov. Pri najslabše ocenjeni križanki je program vseeno rešil 22% polj pravilno. Vredno je poudariti, da so vse najslabše in najbolje rešene križanke med najmanjšimi križankami v bazi. Povprečna ocena rešitev se je izkazala za približno enako, ne glede na velikost križanke. Se pa z večjimi križankami zmanjša varianca ocen.

5.2 Primerjava rezultatov

Če rezultate naše raziskave primerjamo z rezultati drugih raziskav, so se izkazali za manj uspešne. Primerjali smo jih z rezultati raziskav Proverb [2, 5] in WebCrow [1].

Sistem Proverb je v križankah pravilno rešil povprečno 95,3% namigov, WebCrow je v povprečju pravilno rešil 68,8% namigov, naš program pa je v povprečju rešil 52,9% namigov. Potrebno je poudariti, da sta oba sistema vsebovala veliko število specializiranih modulov, ki so bili narejeni za točno

Tabela 5.1: Povprečni rezultati rešenih križank. Podana je tako povprečna ocena nastavkov, kot povprečna ocena končnih rešitev. Polji "Povprečje vseh rešenih namigov"in "Povprečje vseh rešenih polj"predstavljata namige in polja neodvisno od križank, katerim pripadajo.

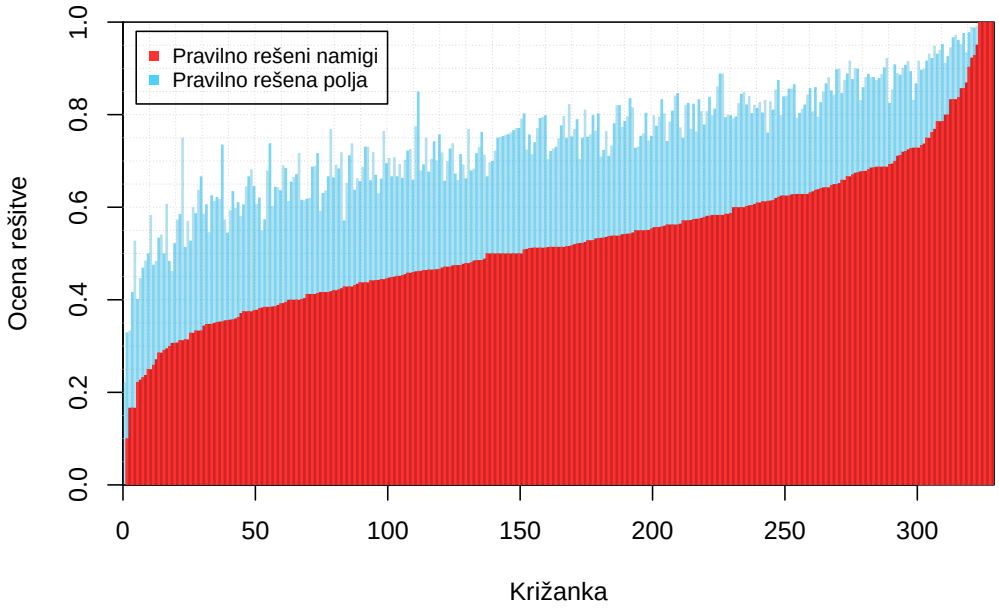
Odstotki rešenih namigov

Povprečje nastavkov	42,2%
Povprečje končnih rešitev	52,9%
Povprečje vseh rešenih namigov	51,6%

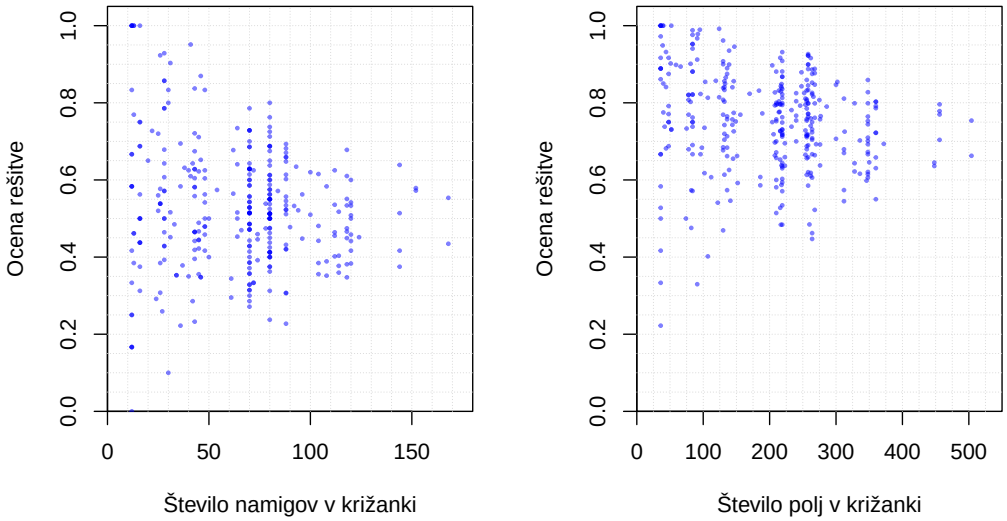
Odstotki rešenih polj

Povprečje nastavkov	66,4%
Povprečje končnih rešitev	75,1%
Povprečje vseh rešenih polj	74,0%

določene tipe vprašanj, kar je po vsej verjetnosti močno pripomoglo k boljšim rezultatom. Predvidevamo tudi, da k uspešnosti rezultatov močno pripomore jezik, v katerem program rešuje križanke. Proverb je reševal križanke v angleškem jeziku, medtem ko je WebCrow reševal križanke v italijanskem jeziku. Uspešnost Proverba pripisujemo dejstvu, da na spletu obstaja veliko število baz in slovarjev v angleškem jeziku, s katerimi so si pomagali pri izdelavi Proverba. Poleg tega je angleški jezik bolj enostaven za primerjavo stavkov, saj ne vsebuje toliko spolov, sklonov in slovničnih števil kot slovenščina in italijanščina.



Slika 5.1: Rezultati križank, razporejeni glede na odstotek pravilno rešenih namigov.



Slika 5.2: Prikaz rezultatov reševanja glede na velikost križanke.

Poglavje 6

Zaključek

V diplomskem delu nam je uspelo uspešno izdelati program za avtomatsko reševanje križank. Program za izpolnjevanje križank uporablja ugankarski slovar in odgovore pridobljene s spletnim iskalnikom. Spletno iskanje za vsako vprašanje v križanki najde ogromno število možnih odgovorov. To je tudi razlog, da program križanko reši v dveh korakih – najprej sestavi delno rešitev s pomočjo ugankarskega slovarja, nato pa v drugem koraku zaključi reševanje s pomočjo vseh odgovorov. Za testiranje programa smo pridobili obsežno bazo slovenskih križank, na kateri nam je uspelo doseči povprečno oceno 52,9% rešenih namigov na križanko (75,1% rešenih polj). Čeprav smo pričakovali boljše rezultate, smo z rezultati raziskave zadovoljni, saj se je reševanje slovenskih križank izkazalo za dokaj kompleksen problem.

Verjamemo, da je možno rezultate raziskave z nekaj truda še dodatno izboljšati. Ocenjujemo, da bi s sledečimi izboljšavami lahko dosegli rezultat, ki ga je dosegel sistem WebCrow [1].

Že pri samem spletnem iskanju bi lahko uporabili drug, bolj primeren iskalnik za iskanje slovenskih odgovorov. Za boljšega se je izkazal iskalnik Google™, vendar ga zaradi problemov pri pridobivanju odgovorov nismo uporabili. Če bi našli način, s katerim bi lahko preko tega iskalnika pridobivali odgovore, bi se lahko v veliki meri izognili velikemu številu neprimernih odgovorov.

Lahko bi izboljšali tudi samo pridobivanje odgovorov iz spletnih strani s tem, da bi si beležili, s katere spletne strani smo dobili določen odgovor. S pomočjo učne množice bi lahko na ta način pridobili seznam domen, kjer so bili največkrat najdeni pravilni odgovori. Odgovore s teh naslovov bi lahko bolje ocenili in si s tem pomagali pri izpolnjevanju križank.

Možne so tudi izboljšave pri iskanju odgovorov v ugankarskem slovarju. Pri postopku primerjav namigov in gesel, opisanem v razdelku 4.3, bi lahko uporabili tezaver in z njim primerjali besede v osnovnih sklonih in številih. Na ta način bi lahko bolje ocenili gesla in s tem izboljšali pravilnost nastavkov. Posledično bi lahko z izločanjem nepravilnih odgovorov pospešili iskanje rešitve.

Dodatne izboljšave so možne tudi z implementacijo specializiranih modulov, ki bi bili zadolženi za iskanje točno določenih odgovorov, kot so naslovi filmov in glasbe, glavna mesta držav, grška božanstva in tako dalje. Tega načina sta se poslužila tako Proverb [2, 5], kot tudi WebCrow [1] in z njim dosegla dobre rezultate. Implementacija takšnih modulov bi lahko nadomestila pripravo nastavka in s tem močno pohitrila ter izboljšala uspešnost reševanja.

Literatura

[1] Marco Ernandes, Giovanni Angelini, and Marco Gori. Webcrow: A web-based system for crossword solving. In *AAAI*, pages 1412–1417, 2005.

[2] Michael L Littman, Greg A Keim, and Noam Shazeer. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134(1):23–55, 2002.

[3] Steve Lohr. The computer's next conquest: Crosswords. Dosegljivo: `http://www.nytimes.com/2012/03/17/technology/computer-matching-wits-with-humans-in-crossword-tournament.html`, 2012. [Dostopano: 1. 6. 2016].

[4] Titus DM Purdin and Geoff Harris. A genetic-algorithm approach to solving crossword puzzles. In *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice*, pages 263–270. ACM, 1993.

[5] Noam M Shazeer, Michael L. Littman, and Greg A. Keim. Solving crossword puzzles as probabilistic constraint satisfaction. In *AAAI/IAAI*, pages 156–162, 1999.

[6] Wikipedia. Crossword. Dosegljivo: `https://en.wikipedia.org/wiki/Crossword`, 2016. [Dostopano: 1. 6. 2016].

[7] Wikipedia. Needleman–wunsch algorithm. Dosegljivo: `https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm`, 2017. [Dostopano: 26. 11. 2017].

[8] Wikipedia. Sørensen–dice coefficient. Dosegljivo: https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient, 2017. [Dostopano: 27. 11. 2017].