ELEKTROTEHNIŠKI VESTNIK 78(3): 153–158, 2011
EXISTING SEPARATE ENGLISH EDITION

Šahovski program Umko

Borko Bošković, Janez Brest

Univerza v Mariboru, Fakulteta za elektrotehniko računalništvo in informatiko,
Smetanova ulica 17, 2000 Maribor, Slovenija
E-pošta: borko.boskovic@uni-mb.si

Povzetek. Umko je močan odprtokodni šahovski program. Pri njegovem razvoju smo uporabili dobre koncepte iz literature in drugih odprtokodnih projektov. S pomočjo teh konceptov smo želeli implementirati kar se da močnega šahovskega igralca. Da bi to naredili, smo implementirali naslednje koncepte: bitno predstavitev deske, generator potez, paralelni iskalni algoritem, preiskovanje več glavnih variant, transpozicijsko tabelo, standardni šahovski vmesnik za komunikacijo z grafičnim vmesnikom, ocenitveno funkcijo, uporabo baz končnic in uporabo otvoritvene knjižnice. Vse omenjene koncepte bomo podrobneje predstavili v članku.

Umko je program, ki se izvaja na različnih platformah in znotraj različnih grafičnih uporabniških vmesnikov. Zna uporabljati najsodobnejše tehnologije procesorjev. Program vsebuje paralelni iskalni algoritem, ki mu omogoča uporabo več procesorjev oz. jeder hkrati. Na drugi strani program uporablja nov SSE4.2 nabor inštrukcij. Tako paralelni iskalni algoritem kot novi nabor inštrukcij omogočata, da je program hitrejši in močnejši igralec. Program je bil preizkušen in uvrščen na različne neodvisne lestvice. Na teh lestvicah se je uvrstil med 10 najboljših odprtokodnih programov.

Ključne besede: šahovski program, bitna predstavitev deske, ocenitvena funkcija, iskalni algoritem, generator potez, transpozicijska tabela, baze končnic, otvoritvena knjižnica

Chess Program Umko

Umko is a strong open-source chess program developed to collect good concepts from literature and other open-source projects. Using these concepts, we want to implement as strong as possible chess program. To do this, Umko has implemented a bitboard representation, move generator, parallel search algorithm, multiple principal variation search, transposition table, universal chess interface, evaluation function, usage of endgame tablebases and usage of the opening book. The paper provides details of these concepts.

Umko is a program running on several platforms inside different graphical user interfaces and using the modern processor technology. It has a parallel search algorithm allowing its program to simultaneously use more processors or cores and the new SSE4.2 CPU instruction set. Both the parallel search algorithm and the new instruction set enable the program to be a faster and stronger player. Having been tested on different independent rating lists, the program is rated among the top ten open-source chess programs.

1 UVOD

Računalniški šah ima dolgo zgodovino raziskav na področju umetne inteligence. S čedalje večjo računalniško močjo, ki je zrasla do izjemne stopnje, smo priča vse več igram med računalniki in ljudmi, kjer po navadi zmagajo računalniki. Razloga za to sta v glavnem izboljšava strojne opreme in optimizacija šahovskih algoritmov. Prvi računalnik, ki je premagal svetovnega šahovskega prvaka, je bil Deep Blue. Premagal je Garija Kasparova, to je bilo leta 1997 [12]. Leta 2006 je računalniški program Deep Fritz 10 na osebnem računalniku premagal svetovnega prvaka Vladimirja Kramnika [18]. Leta 2005 je s svojo močjo presenetil šahovski program Rybka. Zasedel je prvo mesto na vseh pomembnih lestvicah šahovskih programov in je prvi program, ki je dosegel moč igranja več kot 3000 točk [9]. Še več, odprtokodni in brezplačni programi so postajali močnejši, nekateri od njih so postali tudi močnejši od komercialnih programov. Trenutno je najboljši odprtokodni program Stockfish. Njegova igralna moč je bila na večini lestvic ocenjena na več kot 3200 točk. V letu 2010 je mesto najboljšega programa prevzel brezplačni program Houdini. Njegova igralna moč je za približno 50 točk presegla programa Rybka in za približno 70 točk program Stockfish.

Zakaj razvijalci poskušajo izboljšati že tako zelo močne šahovske programe? Veliko poklicnih šahistov za izboljšanje igralne moči, uporablja šahovske programe. Šahovski programi so zelo uporabni v dopisnem šahu in prostem slogu (ang. *Freestyle Chess*). Čedalje bolj priljubljena postajajo tudi tekmovanja med samimi programi. Navsezadnje se je računalniški šah izkazal tudi kot zelo koristno okolje za preizkušanje različnih metod umetne inteligence.

Tudi mi smo želeli razviti močan šahovski program. Da bi to naredili, smo zbrali dobre koncepte iz literature in odprtokodnih projektov. Glede na te vire ima naš program implementirane naslednje koncepte: bitno predstavitev deske, generator potez, paralelni iskalni algoritem, preiskovanje več glavnih variant, transpozicijsko

Prejet 6. junij, 2010
Odobren 15. julij, 2011

tabelo, standardni šahovski vmesnik za komunikacijo z grafičnim vmesnikom, ocenitveno funkcijo, uporabo baz končnic znotraj iskalnega algoritma in uporabo otvoritvene knjižnice. Program ima implementiran paralelni iskalni algoritem, ki omogoča programu, da uporablja več procesorjev ali jeder hkrati. Na drugi strani pa program lahko uporablja nov nabor procesorskih inštrukcij SSE4.2. Paralelni iskalni algoritem in nov nabor inštrukcij omogočata programu, da uporablja sodobno tehnologijo procesorjev in postane hitrejši, s tem pa tudi močnejši igralec.

Ker naš program temelji na odprtokodnih projektih, je objavljen pod licenco GNU GPL v3 na spletni strani sistema SourceForge: http://umko.sourceforge.net/. SourceForge je eden največjih spletnih sistemov za razvoj odprtokodne programske opreme. Zagotavlja brezplačne storitve, ki pomagajo ljudem, da razvijajo programsko opremo in jo delijo z drugimi ljudmi. Umko je implementiran s pomočjo programskega jezika C++, tako da ga je mogoče prevesti s prevajalnikom GNU C++ za operacijske sisteme Linux, Android, Mac OS X in Windows ter za arhitekture i586, x64 in ARM.

Članek je organiziran takole: 2. poglavje podrobneje predstavi program oz. njegove sestavne dele. Eksperimentalni rezultati in doseženi ratingi so predstavljeni v 3. poglavju. Članek končujemo s sklepi in smernicami za nadalnje delo, ki so podani v 4. poglavju.

2 PREDSTAVITEV PROGRAMA

Umko nima lastnega grafičnega uporabniškega vmesnika, je konzolna aplikacija, ki komunicira z grafičnim vmesnikom preko protokola UCI (ang. *Universal Chess Interface*). Naš program teče kot zunanji proces znotraj grafičnega vmesnika. Le–ta krmili naš program s pomočjo standardnih tokov V/I in protokola UCI. Glede na to, da naš program hkrati komunicira z grafičnim vmesnikom in analizira določeno pozicijo, ima vsaj dve niti. Ena se uporablja za komunikacijo, medtem ko se druga ali preostale uporabljajo za analizo določene pozicije in iskanje najboljše poteze. Ker želimo razviti program za operacijske sisteme Windows in Unix, uporablja program niti Windows ali POSIX. Med prevajanjem se ugotovi ciljni operacijski sistem in glede nanj se uporabijo določene niti oz. knjižnice za gradnjo programa.

Osnovni sestavni deli sodobnih šahovskih programov so: predstavitev igre, iskalni algoritem, generator potez, transpozicijska tabela, ocenitvena funkcija, otvoritvena knjižnica in baze končnic. Predstavitev igre omogoča programu, da upravlja pozicije in poteze. Vpliva pa tudi na hitrost šahovskega programa. Generator potez je mehanizem, ki generira legalne poteze, kakor hitro je to mogoče. Iskalni algoritem je odgovoren za iskanje najboljše poteze glede na vrednosti, ki jih vrača ocenitvena funkcija. Ta vsebuje šahovsko znanje, ki ji omogoča, da ocenjuje pozicije. Transpozicijska tabela (TT) omogoča iskalnemu algoritmu, da se izogne večkratnemu iskanju ene in iste pozicije. Otvoritvena knjižnica omogoča šahovskemu programu, da uporabi znanja in izkušnje ljudi oz. računalniških igralcev v otvoritveni fazi igre. Podobno baze končnic omogočajo izbiro močnih potez v končni fazi igre.

Šah je igra, ki je časovno omejena. To pomeni, da je šahovski program časovno omejen pri odločanju, katera poteza je najboljša v določeni poziciji. Zato mora biti šahovski program hiter. To mu omogoča, da preišče več pozicij v določenem času, da je izbrana poteza verjetno boljša in da je program boljši igralec. Na drugi strani šahovski program vsebuje šahovsko znanje, ki prav tako vpliva na hitrost programa. Če program vsebuje več znanja, postaja počasnejši in nasprotno, manj znanja ko vsebuje, hitrejši postaja. Zato razvijalci šahovskih programov morajo najti ravnovesje med količino znanja in hitrostjo, da bi dobili močan šahovski program.

2.1 Predstavitev igre

Predstavitev igre omogoča šahovskemu programu upravljanje potez in pozicij. Vpliva tudi na hitrost generatorja potez, ocenitveno funkcijo in na splošno na hitrost vseh sestavnih delov programa. Obstaja veliko različnih načinov za predstavitev pozicij in potez. Šahovska pozicija vsebuje naslednje informacije: položaj figur, kateri igralec je na potezi, možnosti rokade, polje en passant, število polpotez od zadnje poteze kmeta ali jemanja figure in število vseh potez, ki so bile odigrane, da je nastala ta pozicija. Pri predstavitvi igre je najpomembnejše, kako je predstavljen položaj figur oz. predstavitev deske. Za predstavitev deske smo uporabili bitno predstavitev deske (ang. *Bitboard*). Bitboard je 64-bitno celo število brez predznaka, kjer vsak bit pomeni kvadrat na šahovnici. Ker šahovska igra vsebuje več različnih figur, potrebujemo več bitnih števil oz. eno bitno število za vsak tip figure in barvo. Glavna prednost te predstavitve je, da omogoča hiter izračun potez in izrazov v ocenitveni funkciji [11], [19], [3], [4].

Predstavitev potez je tudi zelo pomembna. Poteze spreminjajo pozicije in kot take se uporabljajo znotraj iskalnega algoritma, kjer se shranjujejo v transpozicijsko tabelo. Zato mora biti predstavitev potez kompaktna. Mi smo uporabili 16-bitna cela števila brez predznaka. Znotraj teh števil so zakodirane naslednje informacije: tip poteze, začetno polje in končno polje poteze. Glede na te informacije se v iskalnem algoritmu spreminjajo pozicije z igranjem in vračanjem poteze. V šahovskem programu je treba tudi primerjati pozicije med seboj. V ta namen smo uporabili ključe Zobrist [24], ki so skoraj unikatna števila za pozicije. Ti ključi se uporabljajo tudi znotraj transpozicijske tabele in otvoritvene knjižnice.

2.2 Iskalni algoritem

Ker je narava šahovske igre zapletena, ni načina, kako izdelati program, ki bi igral popoln šah. Vendar je mogoče razviti iskalni algoritem in ocenitveno funkcijo, ki

skupaj omogočata programu, da postane močan igralec [23]. Ocenitvena funkcija je odgovorna za statično oceno pozicij, iskalni algoritem pa za dinamično oceno pozicij in izbiro potez, ki bodo odigrane.

Na splošno, iskalni algoritmi šahovskih programov temeljijo na algoritmu alfa–beta oz. minimax . V našem programu smo uporabili Principal Variation Search (PVS) [17], [2], [3], ki je izboljšan alfa–beta algoritem. Glavna ideja tega algoritma je, da razlikuje vozlišča (pozicije) glavne variante (PV-vozlišča) in vozlišča, ki ne pripadajo glavni varianti (ne PV-vozlišča). PV-vozlišča se preiskujejo z določenim oknom, na drugi strani ne PV-vozlišča z minimalnim oknom. Okno določata argumenta alfa in beta. Za minimalno okno je razlika med tema argumentoma enaka ena. Algoritem PVS smo uporabili, ker omogoča različne načine klestenja med PV-vozlišči in ne PV-vozlišči. Glavna varianta bo verjetno odigrana in je bolje, da se pri teh vozliščih uporabi manj agresivno klestenje in nasprotno, bolj agresivno klestenje pri ne PV-vozliščih.

Za ovrednotenje listov iskalnega drevesa smo uporabili iskanje mirovanja (ang. *Quiescence Search*). Namen tega iskanja je, da se algoritem izogne učinku obzorja [22]. To pomeni, če bi se iskanje ustavilo v listih drevesa in bi se ocenile pozicije, bi lahko dobljene ocene vsebovale veliko napako. To bi se lahko zgodilo, če bi zadnja odigrana poteza v veji iskalnega drevesa bila jemanje in bi bila naslednja mogoča poteza povratno jemanje. Zato iskanje mirovanja v listih drevesa nadaljuje preiskovanje in se poskuša izogniti učinku obzorja ter pravilno oceniti pozicije.

V iskalnem algoritmu smo implementirali naslednje razširitve iskanja in tehnike klestenja: klestenje vozlišč blizu listov (ang. *Futility Pruning*) [10], klestenje z ničelno potezo (ang. *Null Move Pruning*) [6], redukcije pri poznejših potezah (ang. *Late Move Reductions*) [16], transpozicijska tabela (ang. *Transposition Table*) [5], razširitve izstopajočih potez (ang. *Singular Extensions*) [1], notranje iterativno poglabljanje (ang. *Internal Iterative Deepening*), razširitve edine mogoče poteze (ang. *Single Move Extensions*), razširitve šaha (ang. *Check Extensions*), razširitve potez prostih kmetov (ang. *Passed Pawns Extensions*) in razširitve povratnih jemanj (ang. *Recapture Extensions*). Klestenje vozlišč blizu listov je tehnika, ki klesti vozlišča blizu listov glede na dobljene ocene iz ocenitvene funkcije. Klestenje z ničelno potezo je tehnika, ki s pomočjo ničelne poteze (zamenja igralca na potezi) in redukcije globine iskanja razpozna vozlišča, ki jih je mogoče klestiti. Redukcije pri poznejših potezah je tehnika, ki zmanjša globino preiskovanja pri tihih potezah (poteze, ki ne jemljejo figur ali dajo šah nasprotniku), ki so bliže repu generiranih potez. Transpozicijska tabela shranjuje informacije, ki omogočajo iskalnemu algoritmu, da se izogne večkratnemu iskanju ene in iste pozicije. Razširitev izstopajočih potez je tehnika, kjer se razširi iskanje poteze, za katero se zdi, da je bistveno boljša od preostalih potez. Notranje iterativno poglabljanje je tehnika, podobna iterativnemu poglabljanju s to razliko, da se tukaj poglabljanje izvaja znotraj vozlišč iskalnega drevesa. Iz imen preostalih razširitev: razširitev edine mogoče poteze, razširitev šaha, razširitev potez prostih kmetov in razširitev povratnih jemanj, je mogoče razbrati, za kaj gre, zato teh razširitev ne bomo posebej opisovali.

Celoten algoritem je bil implementiran skupaj z iterativnim poglabljanjem (ang. *Iterative Deepening*), aspiracijskim iskanjem (ang. *Aspiration Search*) in iskanjem v korenskem vozlišču (ang. *Root Search*). Iterativno poglabljanje je tehnika, ki programu omogoča, da iterativno povečuje globino iskanja [15]. Ta tehnika skupaj s preostalimi tehnikami, kot sta npr. transpozicijska tabela in zgodovinska hevristika (ang. *History Heuristic*), omogoča iterativno izboljšanje zaporedja potez. Čeprav se na tak način določena vozlišča preiskujejo večkrat, se učinkovitost celotnega algoritma poveča. Tako je zato, ker je algoritem alfa–beta občutljiv na zaporedje potez. Aspiracijsko iskanje je tehnika, ki zmanjša iskalni prostor [14]. Namesto da bi preiskali celoten iskalni prostor, algoritem ugiba oceno pozicije in preiskuje okoli te vrednosti z določenim oknom. Če je dobljena ocena zunaj okna, je treba iskanje ponavljati, dokler se ocena ne znajde znotraj iskalnega okna. Iskanje v korenskem vozlišču vrne oceno iskalnega algoritma in hkrati potezo, ki bo odigrana. V našem programu smo v to iskanje vključili še iskanje več glavnih variant. To pomeni, da program lahko preiskuje več glavnih variant hkrati. Poteze glavnih variant se preiskujejo kot PV-vozlišča z manj agresivnim klestenjem. S pomočjo protokola UCI se preiskovane variante posredujejo grafičnemu vmesniku. Tako lahko uporabnik sočasno analizira več variant hkrati in dobi njihove ocene.

Da bi povečali hitrost preiskovanja oz. globino preiskovanja z uporabo več procesorjev ali jeder, ima naš program implementiran algoritem “Young Brothers Wait” [7], [8]. To je paralelni algoritem, kjer se v določenem vozlišču prva poteza preiskuje brez paralelizacije, preostale poteze pa se nato preiskujejo vzporedno. Ker pri vzporednem iskanju vsaka iskalna nit piše in bere iz transpozicijske tabele, se lahko zgodi, da so uporabljeni podatki nepravilni. Zato je treba transpozicijsko tabelo prilagoditi paralelnemu algoritmu [13].

Vse predstavljene tehnike iskalnega algoritma smo v našem programu izboljšali glede na dobljene ideje iz odprtokodnih šahovskih programov Toga II in Stockfish.

2.3 Generator potez

Kot smo že omenili, je iskalni algoritem zelo občutljiv na zaporedja potez. Dobro zaporedje potez izboljša učinkovitost iskalnega algoritma in moč šahovskega programa. V našem programu smo implementirali “Magic Bitboard Move Generator”, ki temelji na prej opisani bitni predstavitvi pozicij. Dodatno pa omogoča hitro generiranje potez drsečih figur (ang. *Sliding Pieces*) [19].

S samo nekaj inštrukcijami se izračuna indeks baze bitnih števil, ki vsebujejo bitno predstavitev napadov obeh linij za figuri lovca in trdnjavo. Za izboljšanje zaporedja generiranih potez smo uporabili: transpozicijsko tabelo, oceno statične izmenjave (ang. *Static Exchange Evaluation*), ubijalske poteze (ang. *Killer Moves*) in zgodovinsko hevristiko (ang. *History Heuristic*).

Implementirani generator potez ima štiri sheme. Povzete so po šahovskem programu Toga II. Dve shemi se uporabljata za PVS-vozlišča in dve za QS-vozlišča. Pri PVS-vozliščih se tedaj, ko igralec ni v šahu, uporablja naslednja shema:

- poteza transpozicijske tabele,
- dobra jemanja (ocena statične izmenjave),
- ubijalske poteze,
- tihe poteze (zgodovinska hevristika) in
- slaba jemanja (ocena statične izmenjave).

Če je igralec v šahu, se uporablja naslednja shema:

- poteza transpozicijske tabele,
- dobra izogibanja šahu (ocena statične izmenjave) in
- slaba izogibanja šahu (ocena statične izmenjave).

Podobne sheme se uporabijo pri QS-vozliščih. Če igralec ni v šahu, se uporabi naslednja shema:

- poteza transpozicijske tabele,
- dobra jemanja (ocena statične izmenjave) in
- poteze, ki šahirajo nasprotnika (le tedaj, ko je globina preiskovanja enaka 0).

Če je igralec v šahu, se uporabi naslednja shema:

- poteza transpozicijske tabele in
- vse poteze izogibanja šahu.

Prva uporabljena tehnika, ki izboljšuje zaporedje potez, je transpozicijska tabela. To je razpršena tabela (ang. *Hash Table*), ki vsebuje določene informacije o pozicijah. Kot smo že omenili, se te informacije uporabljajo v iskalnem algoritmu, da ne preiskujemo ene in iste pozicije večkrat. Dodatno se v transpozicijsko tabelo shranjujejo tudi poteze. Če poteza za določeno pozicijo obstaja v transpozicijski tabeli, se preiskuje kot prva in tako se izboljša zaporedje potez. Druga uporabljena tehnika za izboljšavo zaporedja potez je ocena statične izmenjave. Ta tehnika omogoča ločevanje med dobrimi in slabimi potezami jemanj. Za določeno potezo se oceni izmenjava potez na enem polju [20]. Če je ocena večja ali enaka 0, potem gre za dobro jemanje, v nasprotnem primeru za slabo jemanje. Tretja uporabljena tehnika za izboljšavo zaporedja potez vključuje ubijalske poteze. To so dobre tihe poteze, ki so povzročile rez (ang. *Cutoff*) v vozliščih prejšnjih vej iskalnega drevesa in so enako oddaljene od korenskega vozlišča. Zadnja tehnika, ki smo jo uporabili, je zgodovinska hevristika [21]. Uporablja se za dinamično urejanje tihih potez. Skozi preiskovanje se te poteze ovrednotijo glede na število rezov in pripadajočih globin iskanja. Na podlagi teh hevristik se nato tihe poteze urejajo. Omenjeni izraz rez je dogodek, ko se določeno vozlišče preiskuje in je po preiskovanju določene poteze ocena pozicije nad zgornjo mejo iskalnega okna (beta). To pomeni, da preostalih potez ni treba preiskovati.

2.4 Ocenitvena funkcija

Ocenitvena funkcija je zelo pomemben del šahovskih programov. Vsebuje šahovsko znanje in statično ocenjuje pozicije. Te ocene so cela števila, ki se uporabljajo v iskalnem algoritmu in na podlagi njih se oceni korensko vozlišče oz. pozicija. Naša ocenitvena funkcija ima implementirane ideje, ki so povzete iz šahovskega programa Toga II in nadgrajene z določenimi idejami iz šahovskega programa Stockfish. Tako naša ocenitvena funkcija vsebuje naslednja šahovska znanja:

- preproste pozicije končnic (KK, KNK, KBK, KRK, KQK, KNKN, KBKB, KRKR, KQKQ, KNNK, KBBK, KBNK, KBKN, KRRK, KQQK, KQRK),
- vzorci (ujete in blokirane figure),
- figure (material, mobilnost, pozicijske vrednosti),
- kralj (nevihta kmetov, zaščitna polja, napad figur),
- struktura kmetov (veriga kmetov, podvojeni kmeti, izolirani kmeti, zaostali kmeti, kandidati za promocijo, prosti kmeti, nezaustavljivi prosti kmeti) in
- grožnje (napad figur).

Našteta znanja se pri ocenjevanju računajo za otvoritveno fazo in končno fazo igre. Na koncu se s pomočjo interpolacije glede na fazo pozicije v igri izračuna končna ocena.

2.5 Otvoritvena knjižnica in baze končnic

Program, ki uporablja otvoritveno knjižnico in baze končnic, je močnejši igralec. Otvoritvena knjižnica vsebuje izkušnje in znanje ljudi in programov o otvoritveni fazi igre. To omogoča programu, da v otvoritveni fazi igre igra močne poteze zelo hitro. Tako tudi naš program lahko uporablja otvoritvene knjižnice programa Polyglot. Te knjižnice za določeno pozicijo ponujajo poteze in njihove pripadajoče verjetnosti. To pomeni, da program izbira med ponujenimi potezami z določeno verjetnostjo. Tako program igra različne otvoritvene variante. V nasprotnem primeru, brez uporabe otvoritvene knjižnice, bi program igral le nekaj otvoritvenih variant in bi bil predvidljiv.

Podobno kot otvoritvena knjižnica, ki izboljšuje moč programa v otvoritveni fazi igre, baze končnic izboljšujejo njegovo moč v končni fazi igre. Baze končnic omogočajo programu, da oceni določene pozicije končnic brez napak. Naš program uporablja baze končnic programa Gaviota. Te baze vsebujejo pozicije končnic s petimi ali manj figurami. To pomeni, da program igra brez napak pri pozicijah, ki vsebujejo pet ali manj figur. Baze končnic se uporabljajo tudi tedaj, ko korensko vozlišče iskalnega drevesa vsebuje več kot pet figur in pozicije znotraj iskalnega drevesa vsebujejo pet ali manj figur. Tako baze končnic pomagajo iskalnemu algoritmu, da izbira močnejše poteze v končni fazi igre, celo tedaj, ko pozicije vsebujejo več kot pet figur.

Tabela 1: Testne knjižnice (Umko 1.1b)

Testna knjižnica	Čas	Trans. tab.	Hitrost	Globina	Uspešnost	Rat. čaš	Rating
Win at Chess	5	64 (70,7%)	1,74e+06	26,1 (36,2)	294/300	54,8	-
1001 Winning Chess Sacrifices	5	64 (80,1%)	1,73e+06	21,7 (37,9)	872/1001	738,3	-
1001 Brilliant Ways to Checkmate	5	64 (7,8%)	1,34e+06	55,1 (16,1)	988/1001	89,4	-
Strategic Test Suite	10	128 (94,1%)	1,63e+06	18,0 (40,1)	996/1300	3752,6	-
Encyclopedia of Chess Middlegame	20	128 (94,0%)	1,68e+06	20,8 (47,0)	677/770	2429,0	-
Bratko-Kopec	60	256 (95,6%)	1,56e+06	23,8 (46,8)	18/24	404,0	-
LCT II	600	512 (98,0%)	1,75e+06	27,4 (65,6)	29/35	3777,3	2740
BT 2630	900	512 (96,7%)	1,67e+06	29,8 (69,7)	27/30	2889,4	2534
BS 2830	900	512 (96,1%)	1,54e+06	26,1 (71,6)	19/27	7847,6	2771
Nolot	3600	512 (99,8%)	1,54e+06	27,8 (83,4)	6/11	18935,7	-

Tabela 2: Analiza pozicije (Nolot št. 4)

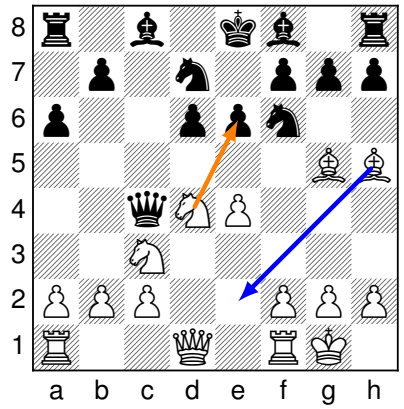
Globina	Ocena	Čas	Hitrost	TT	Poteza
23 (43)	0,83	315	1,48e+6	99,8	h5e2
23 (50)	1,04	370	1,51e+6	99,8	d4e6
23 (52)	1,78	488	1,50e+6	99,8	d4e6
29 (65)	2,11	3003	1,47e+6	99,8	d4e6

3 REZULTATI

3.1 Eksperimentalni rezultati

Umko je bil preizkušen z reševanjem testnih knjižnic. Ta test smo opravili na računalniku s procesorjem Intel(R) Core(TM)2 CPU 6600 2.40GHz. Operacijski sitem je bil Linux in uporabljeni sta bili dve niti za iskalni algoritem. Doseženi rezultati so prikazani v tabeli 1. Prvi stolpec prikazuje ime knjižnice. Drugi stolpec prikazuje čas v sekundah, ki je bil dovoljen za reševanje pozicije. Tretji stolpec prikazuje velikost in povprečno zasedenost transpozicijske tabele. Povprečno hitrost preiskovanja oz. število vozlišč na sekundo prikazuje četrti stolpec. Peti stolpec vsebuje prikaz povprečne globine in povprečne selektivne globine (maksimalno število polpotez od korena drevesa do listov). Šesti stolpec prikazuje uspešnost programa oz. število rešenih pozicij in število vseh pozicij v določeni knjižnici. Sedmi in osmi stolpec prikazujeta rating čas in opcijsko dosežen rating. Določene knjižnice na podlagi rating časa in števila rešenih problemov omogočajo izračun doseženih ratingov.

Način delovanja programa je ponazorjen v tabeli 2. Analizirana je 4. pozicija (slika 1) iz testne knjižnice Nolot. V tabeli prvi stolpec prikazuje globino iskanja skupaj s selektivno globino. Preostali stolpci prikazujejo oceno pozicije, čas v sekundah, hitrost preiskovanja (milijon vozlišč na sekundo), zasedenost transpozicijske tabele v odstotkih in izbrano potezo. Iz rezultatov vidimo, da je program v globini preiskovanja 23 našel pravo rešitev. Pri tem je ocena znašala 1,04 oz. prednost enega kmeta. Da je program našel to rešitev, je potreboval 370 sekund. Hitrost preiskovanja je znašala 1,5 milijona vozlišč na sekundo. Zasedenost transpozicijske tabele je bila 99,8 odstotna. Iskanje se je končalo, ko je program preiskoval 29. globino, in ocena pozicije se je zvišala na 2,11 (prednost dveh kmetov).



Slika 1: Bronstein – Kotov, Budimpešta 1950

Tabela 3: Doseženi ratingi na različnih neodvisnih lestvicah

Rating lestvica	Program	Rating
CCRL 40/4	Umko 1.1 64-bit 2CPU	2912±17
CCRL 40/40	Umko 1.0 64-bit	2908±29
CEGT 40/20	Umko 1.0 x64 4CPU	2848±62
CEGT 40/4	Umko 1.0 x64 1CPU	2811±13
IPON	Umko 1.1 SSE42	2635±11

3.2 Dosežen rating

Umko je bil preizkušen in je na različnih lestvicah šahovskih programov dosegel različne ratinge. Ti so skupaj s 95 odstotnimi intervali zaupanja prikazani v tabeli 3. CCRL (ang. *Computer Chess Rating Lists*) je klub ljudi, ki radi preizkušajo različne programe. Programe tudi primerjajo med seboj tako, da izračunajo njihove ratinge in jih rangirajo. CCRL ima dve lestvici: 40/40 in 40/4. Na prvi lestvici se igrajo igre, kjer programi odigrajo 40 potez v 40 minutah (ponavljajoče). Na drugi lestvici imajo programi na voljo 4 minute za 40 potez (ponavljajoče). Časovni omejitvi na obeh lestvicah sta prilagojeni tako, kot če bi se igralo na računalniku s procesorjem AMD64 X2 4600+ (2.4GHz). Na obeh lestvicah je Umko zasedel 25. mesto (junij, 2011). Med odprtokodnimi programi se je uvrstil na 6. (40/4) oz. 7. (40/40) mesto.

Podobno kot CCRL, CEGT (ang. *Chess Engines Grand Tournament*) je skupina ljudi, ki radi preizkušajo šahovske programe in hkrati delijo dobljene rezultate

z vsemi šahovskimi entuziasti. Prav tako imajo dve lestvici 40/4 in 40/20. IPON je še ena lestvica. V nasprotju s prejšnjima dvehma tukaj programi igrajo po načinu "pondering" (razmišljajo tudi, ko niso na potezi). Časovno so programi omejeni s 5 minutami na igro in imajo 3 sekunde dodatka na potezo. Na tej lestvici je bil preizkušen Umko, ki vsebuje inštrukcije SSE4.2, in se je uvrstil na 20. mesto.

Umko igra tudi na strežniku freechess.org nasproti ljudem in računalnikom iz vsega sveta. Na tem strežniku je naš program dosegel "blitz" rating 2416 iz 858 iger, "standard" rating 2500 iz 911 iger in "lightning" rating 2619 iz 155 iger. Ti ratingi so doseženi s pomočjo enakega računalnika, kot je bil uporabljen za reševanje testnih knjižnic.

V tem poglavju so prikazani ratingi treh različic našega programa, ki pa se nekoliko razlikujejo. Rating namreč pomeni relativno moč programa. Vse različice programa so približno enake po igralni moči, razlika med njimi je le v manjših popravkih ali izboljšavah.

4 SKLEP

V prispevku smo predstavili najboljši slovenski šahovski program, Umko. Program ima implementirane koncepte in ideje, povzete iz literature in iz odprtokodnih projektov. Uporablja tudi najnovejše tehnologije procesorjev, kot sta uporaba več procesorjev oz. jeder sočasno in uporaba novega nabora procesorskih inštrukcij SSE4.2. Program je razvit kot odprtokodni projekt na spletnem sistemu SourceForge. Tako je prosto dostopen in hkrati teče na različnih platformah. Preizkusili smo ga na operacijskih sistemih Linux, Android, Mac OS X in Windows. Glede na dosežene rezultate ugotavljamo, da je Umko močan šahovski igralec. Na neodvisnih lestvicah se uvršča med 25 najboljših programov. Med odprtokodnimi programi se uvršča med 10 najboljših programov. Zanimivo je tudi, da se je Umko izkazal za boljšega kot nekateri komercialni programi, med njimi je tudi znani šahovski program Chessmaster 11.

Pri nadaljnjem delu bomo program poskušali izboljšati. Dodali mu bomo več šahovskega znanja in mu izboljšali paralelni iskalni algoritem. Na drugi strani bomo programu dodati nov koncept, ki mu bo omogočal, da bo igral z različno močjo. Tako bo postal zanimivejši za različne igralce, od začetnikov do velemojstrov.

ZAHVALA

Avtorja se zahvaljujeta avtorjema odprtokodnega programa Toga II Thomasu Gakschu in Fabienu Letouzeyu, avtorjem odprtokodnega programa Stockfish Tordu Romstadu, Marcu Costalbi in Jooni Kiiski, avtorju baz končnic Miguelu A. Ballicori, avtorju programa Polyglot Fabienu Letouzeyu, skupinam CCRL, CEGT in IPON in spletnemu sistemu SourceForge.

LITERATURA

[1] Thomas Anantharaman, Murray S. Campbell in Feng-hsiung Hsu. Singular extensions: adding selectivity to brute-force searching. *Artificial Intelligence*, 43:99–109, 1990.

[2] Yngvi Björnsson. *Selective Depth-First Game-Tree Search*. doktorska disertacija, University of Alberta, 2002.

[3] B. Bošković. Implementacija računalniškega šaha, 2004.

[4] B. Bošković, S. Greiner, J. Brest in V. Žumer. The Representation of Chess Game. V *Proceedings of the 27th International Conference on Information Technology Interfaces*, str. 381–386, 2005.

[5] D.M. Breuker, J. W. H. M. Uiterwijk in H. J. Van Den Herik. Replacement Schemes for Transposition Tables. *ICCA Journal*, 17:183–193, 1994.

[6] Omid David-Tabibi in Nathan Netanyahu. Extended Null-Move Reductions. V H. van den Herik, Xinhe Xu, Zongmin Ma in Mark Winands, uredniki, *Computers and Games*, volume 5131 of *Lecture Notes in Computer Science*, str. 205–216. Springer Berlin / Heidelberg, 2008.

[7] R. Feldmann, P. Mysliwietz in B. Monien. A Fully Distributed Chess Program. *Advances in Computer Chess 6*, 1991.

[8] Rainer Feldmann. *Game Tree Search on Massively Parallel Systems*. doktorska disertacija, 1993.

[9] Harald Fietz. Beyond the 3000 Elo barrier A glance behind the scenes of the Rybka chess engine. *Chess*, str. 18–21, 2007.

[10] E.A. Heinz. Extended Futility Pruning. 21(2):75–83, 1998.

[11] Ernst A. Heinz. How DarkThought Plays Chess. *ICCA Journal*, (3):166–176, 1997.

[12] Feng-hsiung Hsu. IBM's Deep Blue Chess Grandmaster Chips. *IEEE Micro*, 19:70–81, 1999.

[13] Robert M. Hyatt in Timothy Mann. A lockless Transposition-Table Implementation for Parallel Search. *ICGA Journal*, 25(1):36–39, 2002.

[14] Hermann Kaindl, Reza Shams in Helmut Horacek. Minimax Search Algorithms With and Without Aspiration Windows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:1225–1235, December 1991.

[15] Richard E. Korf. Depth-first Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27:97–109, 1985.

[16] D. Levy, D. Broughton in M Taylor. The SEX algorithm in Computer Chess. *ICCA Journal*, 12(1):10–21, 1989.

[17] T. A. Marsland in M. Campbell. Parallel Search of Strongly Ordered Game Trees. *ACM Computing Surveys*, 14:533–551, 1982.

[18] Dylan Loeb Mcclain. Once Again, Machine Beats Human Champion at Chess. *New York Times*, 2006.

[19] Fritz Reul. *New Architectures in Computer Chess*. doktorska disertacija, Tilburg University, 2009.

[20] Fritz Reul. Static Exchange Evaluation with $\alpha\beta$-Approach. *ICGA Journal*, 33(1):3–17, 2010.

[21] Jonathan Schaeffer. The History Heuristic. *ICCA Journal*, 6(3):16–19, 1983.

[22] Günther Schrüfer. A Strategic Quiescence Search. *ICCA Journal*, 12:3–9, 1989.

[23] C. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(4):256, 1950.

[24] Albert Zobrist. A New Hashing Method with Application for Game Playing. *ICCA Journal*, 13(2):69–73, 1970.

Borko Bošković je diplomiral leta 2004 in doktoriral leta 2010 na Fakulteti za elektrotehniko računalništvo in informatiko Univerze v Mariboru. Trenutno na omenjeni fakulteti zaseda mesto asistenta z doktoratom. Njegova področja raziskovanja so šahovski algoritmi in evolucijsko računanje. Na strokovnem področju se ukvarja tudi s programskimi jeziki, integracijskim programiranjem in procesiranjem naravnih jezikov.

Janez Brest je diplomiral leta 1995, magistriral leta 1998 in doktoriral leta 2000 na Fakulteti za elektrotehniko računalništvo in informatiko Univerze v Mariboru. Trenutno je redni profesor na omenjeni fakulteti. Njegova področja raziskovanja vključujejo evolucijsko računanje, umetno inteligenco in optimiziranje. Njegova strokovna področja zajemajo tudi programske jezike, spletno programiranje, paralelno in porazdeljeno procesiranje.