Adaptivni algoritem diferencialne evolucije za uglaševanje parametrov ocenitvene funkcije računalniškega šaha

Borko Bošković, Sašo Greiner, Janez Brest, Viljem Žumer

Univerza v Mariboru

Fakulteta za elektrotehniko računalništvo in informatiko

Inštitut za računalništvo

Smetanova ulica 17, 2000 Maribor, Slovenija

E-pošta: borko.boskovic@uni-mb.si

Adaptive differential evolution algorithm for tuning the parameters of a chess evaluation function

Computer chess provides a competitive and dynamic environment. In this environment individuals (chess programs) with competition try to survive into successive generations. In this way the evolution tunes parameters of a chess program. This environment was used for the tuning with differential evolution algorithm. In this algorithm we introduced competition between two corresponding individuals and an adaptive mutation control parameter. Competition in this form reduces time complexity according to competition "all-to-all". According to our experiment and obtained results we discovered that adaptive mutation control parameter forms population with large standard deviation of parameters in the last generation if algorithm didn't find a "good" individual. We can use this property to continue evolution until the mutation control parameter becomes sufficiently small.

1 Uvod

Računalniški šah omogoča oblikovanje tekmovalnega in dinamičnega okolja, ki je zelo podobno naravnemu okolju. V tem okolju posameznike predstavljajo šahovski igralci oz. programi. Znotraj evolucije posamezniki tekmujejo. Glede na njihovo uspešnost, v naslednje generacije preživijo boljši posamezniki. Tako ohranjajo boljši genetski material v populaciji. Uspešnost posameznika definira njegova relativna uspešnost glede na njegove nasprotnike. Tako dobimo ocenitveno vrednost posameznika, ki jo določajo izidi iger, brez posredovanja človeka in njegovega strokovnega znanja [8].

Šah je igra s popolno informacijo med dvema nasprotnikoma. To pomeni, da imata v vsaki fazi igre, oba igralca popoln pregled nad igro (pozicijo in potezami). Danes so šahovski programi (sistemi) zelo močni šahovski igralci. Celo najboljši šahovski igralci (ljudje) jih s težavo premagujejo. Razlog temu so napredni iskalni algoritmi, ki preiskujejo nadaljevanja v igri in nato izberejo najboljšo potezo. Razvijalci šahovskih programov iskalne algoritme še vedno izboljšujejo, čeprav je vedno večji poudarek na algoritmih učenja oz. na uglaševanju parametrov šahovskih programov. Algoritmi, ki jih uporabljajo, so npr. vzpenjanje na hrib, simulirano ohlajanje, "Temporal Differences Learning" [1] in evolucijski algoritmi [7, 6].

Naš algoritem temelji na principu Darwinove evolucije. Uporabili smo algoritem diferencialne evolucije (DE - Differential Evolution), ki se je izkazal kot zelo učinkovit pri reševanju različnih problemov [10, 9]. Tudi pri uglaševanju iger se je izkazal za zelo učinkovitega [5, 4]. Ta algoritem smo še dodatno optimizirali, tako da smo vpeljali tekmovanje pripadajočih posameznikov in adaptacijo kontrolnega parametra F. Iz pridobljenih rezultatov smo ugotovili, da tekmovanje z dvema igrama pripadajočih posameznikov, omogoča konvergenco k "dobrim" parametrom in da vpeljana adaptacija omogoča ugotoviti ali smo našli dobre parametre. To ugotovitev lahko uporabimo v algoritmu tudi kot zaustavitveni pogoj evolucije.

V drugem poglavju predstavljamo naš algoritem za uglaševanje parametrov. Tretje poglavje vsebuje eksperiment in prikaz dobljenih rezultatov. Kratek zaključek opravljenega dela in nadaljnje delo podajamo v četrtem poglavju.

2 Algoritem uglaševanja

Naš algoritem za uglaševanje parametrov ocenitvene funkcije šahovskega programa temelji na algoritmu diferencialne evolucije. To je algoritem, ki v vsaki generaciji (G) vsebuje eno populacijo (P_G). Ta populacija vsebuje NP D–dimenzionalnih vektorjev oz. posameznikov.

$$\boldsymbol{X}_{G,i} = \{X_{G,i,1}, X_{G,i,2}, ..., X_{G,i,D}\};$$

$$i = 1, 2, ..., NP. \quad (1)$$

Posameznike v našem primeru predstavljajo vektorji, katerih komponente so parametri ocenitvene funkcije. Ti parametri določajo obnašanje programa. Tako z iskanjem dobrih vektorjev iščemo dobre nastavitve ocenitvene funkcije. Skozi evolucijo algoritem DE nad populacijo izvaja operacije križanja, mutacije in selekcije. Naš algoritem v DE vpeljuje tekmovanje med posamezniki, na osnovi katerega se odloča, kateri posameznik je boljši in adaptacijo kontrolnega parametra F, ki ga uporabljamo pri operaciji mutacije. Način delovanja našega algoritma ponazarja spodnji algoritem.

```
Inicializacija(P_0);
while(nadaljuj uglaševanje) {
    F = določiF(P_G);
    P_V = Mutacija(P_G, F);
    P_U = Križanje(P_G, P_V, CR);
    Tekmovanje(P_G, P_U);
    P_{G+1} = Selekcija(P_G, P_U);
}
```

V prikazanem algoritmu P_0 predstavlja začetno populacijo, P_V mutirano populacijo, P_U poskusno populacijo, P_G populacijo trenutne generacije in P_{G+1} populacijo naslednje generacije. F in CR sta kontrolna parametra algoritma DE.

2.1 Inicializacija

Na začetku populacijo P_0 inicializiramo z vrednostmi parametrov, ki so izbrani z uniformnim naključnim generatorjem, med mejami parametrov $(X_{j,low}, X_{j,high}; j = 1, 2, ..., D)$. Meje parametrov določa uporabnik glede na problem, ki ga uglašuje.

2.2 Adaptacija kontrolnega parametra skaliranja F

Kontrolni parameter skaliranja F uporabljamo pri mutaciji za ustvarjanje mutiranih vektorjev. Ta parameter določa velikost razlike parametra mutiranega vektorja glede na vektor v trenutni populaciji. Tako smo se odločili, da bomo vrednost parametra F določali glede na razpršenost parametrov v populaciji, kot prikazujejo spodnje enačbe.

$$F = 2\frac{\sum_{i=1}^{D} KV_i}{D}, \qquad KV_i = \frac{\sigma_i}{\overline{X_i}};$$

$$\sigma_i = \sqrt{\frac{\sum_{j=1}^{NP}(X_{i,j} - \overline{X_i})^2}{NP - 1}}. \quad (2)$$

Faktor F dobi vrednost povprečja koeficientov variacij KV_i. Le–te računamo na osnovi standardnih odklonov σ_i in povprečnih vrednosti $\overline{X_i}$. V literaturi [10, 9] so priporočila, da naj bo faktor F na intervalu $[0, 2]$, zato povprečje parametrov variacij množimo še z 2. Taka adaptacija parametra F omogoča algoritmu DE, da ima v začetnih generacijah bolj razpršene parametre. V kasnejših generacijah, ko algoritem najde boljše posameznike, se razpršenost zmanjša, s tem pa tudi parameter F, ki pripomore k hitrejši konvergenci algoritma.

2.3 Mutacija

Mutacija ustvari mutirano populacijo P_V glede na trenutno populacijo P_G z uporabo določene strategije mutacije. Za vsak vektor iz trenutne populacije mutacija ustvari mutiran vektor $\boldsymbol{V}_{G,i}$, ki predstavlja posameznika mutirane populacije.

$$\boldsymbol{V}_{G,i} = \{V_{G,i,1}, V_{G,i,2}, ..., V_{G,i,D}\};$$
$$i = 1, 2, ..., NP. \quad (3)$$

Pri mutaciji lahko uporabimo različne strategije mutiranja. V našem pristopu smo uporabili eno izmed najbolj uporabljenih strategij, ki jo avtorji označujejo z $rand/1$ [10, 9]. Ta strategija mutira vektorje na naslednji način:

$$\boldsymbol{V}_{G,i} = \boldsymbol{X}_{G,r1} + F \cdot (\boldsymbol{X}_{G,r2} - \boldsymbol{X}_{G,r3}). \quad (4)$$

kjer so $r1, r2, r3$ naključno izbrani med seboj različni indeksi, ki so različni od i in so na intervalu $[1, NP]$. Ti indeksi se znova določajo pri vsakem ustvarjanju mutiranega vektorja. Faktor F, katerega vrednost se določa glede na razpršenost populacije, je skalirni parameter mutacije na intervalu $[0, 2]$.

2.4 Križanje

Po operaciji križanja, mutacija oblikuje poskusno populacijo P_U. Glede na i-ti vektor iz trenutne populacije $\boldsymbol{X}_{G,i}$ in njegov pripadajoči mutiran vektor $\boldsymbol{U}_{G,i}$, križanje ustvari poskusni vektor na naslednji način:

$$\boldsymbol{U}_{G,i} = \{U_{G,i,1}, U_{G,i,2}, ..., U_{G,i,D}\};$$

$$U_{G,i,j} = \begin{cases} V_{G,i,j}, & rand_j \leq CR \quad \vee \quad j = j_{rand} \\ X_{G,i,j}, & drugače; \end{cases}$$

$$i = 1, 2, ..., NP, \quad j = 1, 2, ..., D. \quad (5)$$

CR je faktor križanja na intervalu $[0, 1)$ in predstavlja verjetnost, da bo parameter poskusnega vektorja dobljen iz mutiranega vektorja. Indeks j_{rand} je naključno izbrano celo število na intervalu $[1, NP]$ in je odgovorno za to, da poskusni vektor vsebuje vsaj en parameter mutiranega vektorja.

Po končanem križanju dobimo poskusne posameznike katerih parametri so lahko izven iskanega območja $(X_{j,low}, X_{j,high})$. V tem primeru posameznike popravimo tako, da vrednosti prezrcalimo v iskano območje na naslednji način:

$$U_{G,i,j} = \begin{cases} X_{j,low} + (X_{j,low} - U_{G,i,j}), & U_{G,i,j} < X_{j,low} \\ X_{j,high} - (U_{G,i,j} - X_{j,high}), & U_{G,i,j} > X_{j,high} \\ U_{G,i,j}, & drugače. \end{cases} \quad (6)$$

2.5 Tekmovanje

Ko dobimo poskusno populacijo, moramo oceniti njene posameznike, glede na posameznike v trenutni populaciji. Ocenjevanje opravimo tako, da pripadajoči posamezniki ($\boldsymbol{X}_{G,i}, \boldsymbol{U}_{G,i}$) med seboj tekmujejo. Posameznika tekmujeta tako, da odigrata dve igri. Igri odigrata tako, da ima vsak od igralcev v eni igri bele figure. Glede na izida iger tekmovalca dobita določeno število točk, ki predstavlja njihovo relativno oceno. Za zmago igralec dobi 2 točki, za remi 1 in poraz, 0 točk.

2.6 Selekcija

Operacija selekcije, glede na zbrane točke v tekmovanju, izbere kateri posameznik bo preživel v naslednjo generacijo. Pri izbiri smo uporabili naslednje pravilo:

$$\boldsymbol{X}_{G+1,i} = \begin{cases} \boldsymbol{U}_{G,i}, & št.\ točk(\boldsymbol{U}_{G,i}) > št.\ točk(\boldsymbol{X}_{G,i}) \\ \boldsymbol{X}_{G,i}, & drugače. \end{cases} \quad (7)$$

V naslednjo generacijo preživijo posamezniki z večjim številom točk. V primeru enakega števila točk, se naš algoritem razlikuje od navadnjega algoritma DE, v naslednjo generacijo preživi posameznik trenutne populacije. Posameznik trenutne populacije je preživel vsaj eno generacijeo in verjetnost, da vsebuje boljše parametre, je večja.

3 Eksperiment

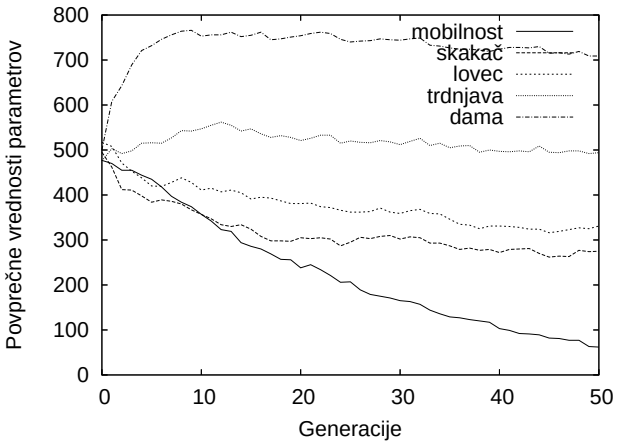
Delovanje našega algoritma smo preizkusili z uglaševanjem parametrov šahovskega programa BBChess [3, 2]. Program BBChess je vseboval ocenitveno funkcijo, ki jo prikazuje enačba:

$$ocenitev = X_m(M_b - M_č) + \sum_{i=0}^{5} X_i(N_{i,b} - N_{i,č}). \quad (8)$$

V enačbi (8) X_i predstavlja parametre materialnih vrednosti figur, N_i pa število figur tipa i določene barve ($č$ – črna, b – bela). M predstavlja število možnih polj, na katere lahko postavimo figure določene barve oz. mobilnost. Parameter mobilnosti predstavlja X_m.

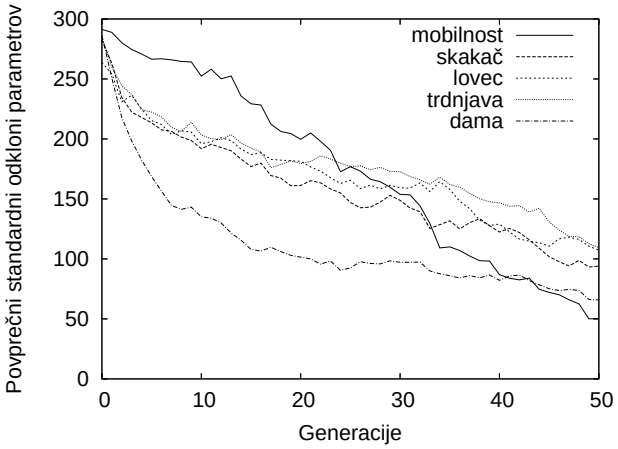
V eksperimentu smo parameter materialne vrednosti kmeta fiksirali na 100 in globino preiskovanja šahovskega algoritma nastavili na 5 polpotez. Uglaševanje smo izvajali skozi 50 generacij. Velikost populacije je bila NP=10. Za to velikost smo se odločili zaradi časovne zahtevnosti algoritma. Kontrolni parameter CR smo nastavili na 0.9, glede na priporočila literature [10, 9]. Meje parametrov ocenitvene funkcije so bile $X_{j,low} = 0, X_{j,high} = 1000$.

Proces uglaševanja parametrov s predstavljenimi parametri smo zagnali 50 krat. Eksperiment je trajal približno 5 dni. Povprečne vrednosti parametrov v vsaki generaciji za vseh 50 zagonov smo povprečili (Slika 1) in ugotovili, da je bil proces uglaševanja uspešen. Razmerje uglašenih parametrov ustrezajo parametrom, ki jih vsebuje teorija šaha.



Slika 1. Povprečne vrednosti parametrov ocentivene funkcije za 50 zagonov.
Figure 1. Average parameter values of evaluation function for 50 runs.

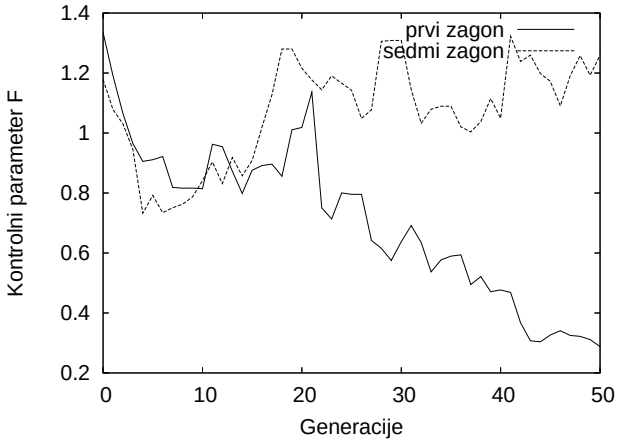
Za vse zagone smo povprečili še standardne odklone parametrov (Slika 2). V nasprotju s pričakovanji je bil povprečni standardni odklon za končno populacijo relativno visok.



Slika 2. Povprečne vrednosti standardnega odklona parametrov ocenitvene funkcije za 50 zagonov.
Figure 2. Average standard deviation of evaluation function parameters for 50 runs.

Da bi ugotovili, kaj je vzrok temu, smo analizirali še re-

zultate vsakega zagona posebej. V šestih zagonih smo ugotovili, da algoritem ni našel dobrih parametrov in da je končna populacija razpršena podobno kot začetna populacija, kar je bilo razvidno tudi iz vrednosti adaptivnega kontrolnega parametra F.



Slika 3. Povprečne vrednosti standardnega odklona parametrov ocenitvene funkcije za 50 zagonov.
Figure 3. Average standard deviation of evaluation function parameters for 50 runs.

Primerjavo kontrolnega parametra F med prvim in sedmim zagonom prikazuje slika (3). V prvem zagonu je algoritem našel dobre parametre in razpršenost se je v primerjavi z začetno populacijo znatno zmanjšala. V sedmem zagonu algoritem ni uspel najti dobrih parametrov in razpršenost je ostala približno enaka kot v začetni populaciji. Ta lastnost razlikuje naš algoritem od osnovnega algoritma DE s konstantno vrednostjo parametra F (priporočena vrednost 0.5). S priporočeno vrednostjo lahko algoritem DE konvergira tudi k slabim parametrom. To lastnost našega algoritma lahko uporabimo tudi kot zaustavitveni pogoj evolucije. Kadar se kontrolni parameter F zmanjša na določeno vrednost, končamo algoritem. Male vrednosti kontrolnega parametra F ne omogočajo več bistvenih izboljšav posameznikov.

4 Zaključek

V članku smo predstavili algoritem za uglaševanje parametrov ocenitvene funkcije šahovskega programa. Algoritem temelji na algoritmu diferencialne evolucije in vsebuje tekmovanje med pripadajočimi posamezniki ter adaptacijo kontrolnega parametra F. Vpeljano tekmovanje je časovno manj zahtevno kot v primeru tekmovanja vsak z vsakim. Adaptacija kontrolnega parametra omogoča algoritmu, da dokler ne najde dobrih parametrov, oblikuje populacije z razpršenimi parametri. V nasprotnem primeru konvergira k dobrim parametrom. To lastnost lahko uporabimo tudi kot zaustavitveni pogoj evolucije.

V nadaljevanju našega dela bomo poskušali še z adaptacijo velikosti populacije NP, glede na vrednost adaptivnega kontrolnega parametra F. Tako bi lahko zmanjšali število potrebnih iger in zmanjšali časovno zahtevnost algoritma.

Literatura

[1] Jonathan Baxter, Andrew Tridgell, and Lex Weaver. Learning to play chess using temporal differences. *Machine Learning*, 40(3):243–263, 2000.

[2] B. Bošković, J. Brest, and V. Žumer. Objektno orientirano načrtovanje in implementacija računalniškega šaha. *Elektrotehniški vestnik*, 73(1):31–37, 2006.

[3] B. Bošković, S. Greiner, J. Brest, and V. Žumer. The representation of chess game. In *Proceedings of the 27th International Conference on Information Technology Interfaces*, pages 381–386, 2005.

[4] B. Bošković, S. Greiner, J. Brest, and V. Žumer. Uglaševanje računalniškega šaha z uporabo diferencialne evolucije. In *Zbornik štirinajste mednarodne Elektrotehniške in računalniške konference ERK 2005*, pages 79–82, 2005.

[5] Ken Chisholm. *Co-evolving Draughts Strategies with Differential Evolution*, chapter 9, pages 147–158. McGraw-Hill, London, 1999.

[6] David B. Fogel, Timothy J. Hays, Sarah L. Hahn, and James Quon. A self-learning evolutionary chess program. *Proceedings of the IEEE*, 92(12):1947–1954, 2004.

[7] Graham Kendall and Glenn Whitwell. An evolutionary approach for the tuning of a chess evaluation function using population dynamics. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 995–1002, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 2001. IEEE Press.

[8] Simon M. Lucas and Graham Kendall. Evolutionary computation and games. *IEEE Computational Intelligence Magazine*, 2006.

[9] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution, A Practical Approach to Global Optimization*. Springer, 2005.

[10] R. Storn and K. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.