Analiza algoritma jDE100 za enokriterijsko globalno optimizacijo z realnimi vrednostmi

Janez Brest[1], Jan Popič[1], Mirjam Sepesy Maučec[2], Borko Bošković[1]

[1] *Laboratorij za računalniške arhitekture in jezike, Inštitut za računalništvo,*
[2] *Laboratorij za digitalno procesiranje signalov, Inštitut za elektroniko in telekomunikacije,*

Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru,
Koroška cesta 46, 2000 Maribor, Slovenija,
janez.brest@um.si, jan.popic1@um.si, mirjam.sepesy@um.si, borko.boskovic@um.si

An Analysis of the jD100 Algorithm for Real-Parameter Single Objective Global Optimization

Abstract.

Solving real-parameter single objective problems is a challenging research topic in evolutionary computation community. At the IEEE Congress on Evolutionary Computation (CEC) 2019, Special Session on 100-Digit Challenge on single objective numerical optimization was organized, where ten problems were defined and participants of this challenge were required to solve each problem and try to reach the accuracy of 10 digits for each of them. The optimum values were known for all problems. There were no time limit and no limit for maximum number of function evaluations. In this paper, we present a version of Differential Evolution algorithm, which is based on the jDE100 algorithm, for tackling 100-Digit Challenge. The main focus of this paper is to find out why many algorithms that participated in 100-Digit Challenge competition were not able to solve all problems, except jDE100. We provide an analysis to show which mechanisms helped the jDE100 algorithm to successfully solve all problems, contrary to other algorithms.

1 Uvod

V vsakdanjem življenju se ljudje pogosto srečujemo z optimizacijo na vsakem koraku, a se morda tega niti ne zavedamo. Nenehno težimo k zmanjševanju stroškov, na poti v šolo ali službo želimo po najkrajši poti ali pa po poti, ki naj najhitreje pripelje do cilje, itd.

V članku obravnavamo numerično optimizacijo. Predstavili bomo novo verzijo algoritma diferencialne evolucije (DE). DE se v zadnjem času uporablja v številnih praktičnih aplikacijah. Reševanje enokriterijskih optimizacijskih problemov z realnimi vrednostmi je aktualna raziskovalna tematika na področju evolucijskega računanja. Na mednarodnem IEEE kongresu o evolucijskem računanju (CEC 2019) se je odvijala posebna sekcija o enokriterijski globalni optimizaciji z realnimi vrednostmi. Znotraj sekcije je bilo tudi tekmovanje '100-Digit Challenge', ki je letos prekoračilo meje konference CEC in je vključevalo tudi konferenci GECCO 2019 in SEMCCO 2019.

V članku bomo govorili o optimizacijskih problemih, pri katerih nas zanimajo minimumi. Problem globalne optimizacije lahko definiramo kot iskanje vektorja $\vec{x}$, ki minimizira funkcijsko vrednost $f(\vec{x})$. Vektor $\vec{x} = \{x_1, x_2, ..., x_D\}$ obsega D spremenljivk. Za vsako spremenljivko x_j ($j = 1, 2, ..., D$) je definirana spodnja $x_{j,low}$ in zgornja $x_{j,upp}$ meja. Rečemo tudi, da D označuje dimenzijo optimizacijskega problema. Funkcija f ni nujno, da je zvezna ali odvedljiva, mora pa biti omejena.

Algoritem DE [7, 8] sta pred več kot dvajsetimi leti predstavila R. Storn in K. Price. Dandanes sta še oba aktivna na raziskovalnem področju evolucijskih algoritmov, predvsem diferencialne evolucije (omenimo, da nekateri strokovnjaki v Sloveniji uporabljajo tudi izraz diferenčna evolucija). Lahko rečemo, da DE predstavlja uspešen algoritem, ki se danes uporablja v mnogih raziskavah in praktičnih aplikacijah [2, 5, 9].

Članek ima sledečo strukturo. V drugem poglavju podamo ozadje treh algoritmov DE. V tretjem poglavju predlagamo novo verzijo (izpeljanko) algoritma in predstavimo njegove podobnosti in razlike z algoritmom jDE100. Eksperimenti, rezultati in analiza so podani v četrtem poglavju. Peto poglavje pa zaključi članek.

2 Ozadje

2.1 Algoritem *jDE*

Algoritem jDE [4], ki je bil prvič predstavljen leta 2006, uporablja samo-prilagodljiv mehanizem krmilnih parametrov, kjer ima vsak posameznik v populaciji svoji vrednosti za krmilna parametra F_i in CR_i. Novi vrednosti krmilnih parametrov $F_{i,g+1}$ in $CR_{i,g+1}$ se izračunata pred mutacijo na naslednji način:

$$F_{i,g+1} = \begin{cases} F_l + rand_1 * F_u, & \text{if } rand_2 < \tau_1, \\ F_{i,g}, & \text{otherwise,} \end{cases}$$

$$CR_{i,g+1} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2, \\ CR_{i,g}, & \text{otherwise,} \end{cases}$$

Tabela 1: Parameteri in njihove vrednosti pri algoritmu jDE100-ver2.0. F_l in CR_l sta parametra, ki smo ju lahko nastavili posebej za vsako funkcijo.

Parameter	Vrednost	Kratek opis
F_l	0.2 za F4, F7	spodnja meja skalirnega faktorja
F_l	0.1 za F8	spodnja meja skalirnega faktorja
F_l	0.001 za F9	spodnja meja skalirnega faktorja
F_l	0.15 pri ostalih funkcijah	spodnja meja skalirnega faktorja
F_u	1.1	zgornja meja skalirnega faktorja
CR_l	0.1 za F8	spodnja meja parametra križanja
CR_l	0.9 za F9	spodnja meja parametra križanja
CR_l	0.0 pri ostalih funkcijah	spodnja meja parametra križanja
CR_u	1.1	zgornja meja parametra križanja
F_{init}	0.5	inicialna vrednost skalirnega faktorja
CR_{init}	0.5	inicialna vrednost parametra križanja
τ_1	0.1	verjetnost samo-prilagajanja skalirnega faktorja
τ_2	0.1	verjetnost samo-prilagajanja parametra križanja
bNP	1000	velikost $\mathbf{P_b}$
sNP	25	velikost $\mathbf{P_s}$
$ageLmt$	$1e9$	št. ovrednotenj, ko se naj zgodi reinicializacija populacije
eps	$1e-16$	majhna vrednost za primerjavo, če sta dve vrednosti podobni
$maxFEs$	$1e12$	en izmed ustavitvenih pogojev (Pogoj ni bil izpolnjen!)
$myEqs$	25	reinicializacija, če ima $myEps\%$ posameznikov podobno funkcijsko vrednost

kjer so $rand_j$, $j \in \{1, 2, 3, 4\}$ naključne vrednosti, porazdeljene uniformno na intervalu $[0, 1]$. Vrednosti τ_1 in τ_2 sta verjetnosti, s katerima krmilimo krmilna parametra F in CR. Podroben opis algoritma najdemo v literaturi [4, 3].

2.2 Algoritem jDE100

Algoritem jDE, ki smo ga na kratko opisali v predhodnem podpoglavju, smo nadgradili v algoritem jDE100 [3], ki je sodeloval v tekmovanju 100-Digit Challenge na CEC 2019. jDE100 uporablja dve različno veliki populaciji in njima prilagojeni reinicializaciji.

V evolucijskem procesu algoritma jDE100 imamo tri ponavljajoče se korake:

1. Izvede se ena generacija na veliki populaciji, $\mathbf{P_b}$.
2. Če je potrebno, se najboljši posameznik, $\vec{x}_{best}$, kopira v manjšo populacijo $\mathbf{P_s}$.
3. Izvede se več generacij na majhni populaciji, $\mathbf{P_s}$.

Velikost velike populacije je m-kratnik ($m \in 1, 2, ...$) velikosti majhne populacije, kar pomeni, da algoritem razdeli število evaluacij med obe populaciji: polovica evaluacij za veliko in polovica za majhno populacijo, če pri slednji izvede m generacij.

Pomembna je tudi reinicializacija, ki jo ločeno izvedemo na obeh populacijah, če je izpolnjen pogoj, da ima vnaprej predpisano število najbolje ocenjenih posameznikov v dani populaciji zelo podobno funkcijsko vrednost. V tem primeru smo ocenili, da se je populacija ujela v lokalni optimum. Reinicializacijo velike populacije izvedemo tako, da vse posameznike naključno reinicializiramo. Podobno storimo tudi pri majhni populaciji, z razliko da najboljšega posameznika pustimo nespremenjenega (ga ne reinicializiramo).

Velika populacija skrbi za večjo raznolikost posameznikov, manjša pa omogoča hitrejšo konvergenco iskanja. Najboljšega posameznika kopiramo iz velike populacije, v kateri je bil najden, v manjšo. V obratni smeri pa ne kopiramo.

Algoritem ima parametre, ki so predstavljeni v tabeli 1. Kot zanimivost omenimo, da smo F_u nastavili na vrednost 1.1 (tako je $F \in [F_l, F_l + F_u]$)), kar pomeni, da je zgornja meja parametra večja od 1. Podobno smo nastavili tudi $CR_u = 1.1$.

Podroben opis algoritma bralec najde v prispevku [3], saj je naš glavni namen v tem prispevku analiza mehanizmov. Zanima nas, ali lahko nastavitve parametrov algoritma jDE100 odločilno vplivajo na uspešnost reševanja problemov na tekmovanju. Preden se lotimo analize, v nas-

Tabela 2: Opis funkcij za '100-Digit Challenge' [6].

	funkcija	$F_i^* = F_i(x^*)$	D	meje iskalnega prostora
1	Storn's Chebyshev Polynomial Fitting Problem	1	9	[-8192, 8192]
2	Inverse Hilbert Matrix Problem	1	16	[-16384, 16384]
3	Lennard-Jones Minimum Energy Cluster	1	18	[-4, 4]
4	Rastrigin's Function	1	10	[-100, 100]
5	Griewank's Function	1	10	[-100, 100]
6	Weierstrass Function	1	10	[-100, 100]
7	Modified Schwefel's Function	1	10	[-100, 100]
8	Expanded Schaffer's F6 Function	1	10	[-100, 100]
9	Happy Cat Function	1	10	[-100, 100]
10	Ackley's Function	1	10	[-100, 100]

Tabela 3: 50 zagonov algoritma a200-25 za vsako funkcijo, ki so v tabeli prikazani glede na število najdenih pravilnih decimalk.

funkcija	število pravilnih decimalk											št. točk
	0	1	2	3	4	5	6	7	8	9	10	
F1	0	0	0	0	0	0	0	0	0	0	50	10
F2	0	0	0	0	0	0	0	0	0	0	50	10
F3	0	0	0	0	0	0	0	0	0	1	49	10
F4	0	0	0	0	0	0	0	0	0	0	50	10
F5	0	0	0	0	0	0	0	0	0	0	50	10
F6	0	0	0	0	0	0	0	0	0	0	50	10
F7	0	0	0	0	0	0	0	0	0	0	50	10
F8	0	0	0	0	0	0	0	0	0	0	50	10
F9	0	0	1	26	17	5	0	0	0	0	1	4.36
F10	0	0	0	0	0	0	0	0	0	0	50	10
											Skupno število točk:	**94.36**

lednjem podpoglavju na kratko opisujemo še en algoritem, ki bo vključen v analizo.

2.3 Algoritem rjDE

Algoritem rjDE je nadgradnja algoritma *jDE* z mehanizmom reinicializacije populacije. Omenjeni mehanizem je podoben tistemu, ki smo ga opisali v algoritmu jDE100. Opis algoritma rjDE najdemo v [1], kjer je bila narejena primerjava zmogljivosti algoritmov na 100-Digit Challenge pri omejitvi največjega števila vrednotenj funkcije ($maxFES = 10^7$).

3 Hibrid algoritmov jDE100 in rjDE

Z namenom, da bi ugotovili, kateri mehanizmi omogočajo dobro delovanje algoritma jDE100 na 100-Digit Challenge, smo zasnovali novo verzijo algoritma, ki je hibrid med algoritmom jDE100 in algoritmom rjDE. Čemu izbrati tak način, če pa imamo na drugi strani morda bolj trivialno rešitev: v algoritmu jDE100 bi lahko po vrsti izklopili en mehanizem in pognali algoritem. Implementacijsko ideja ni zahtevna, a izvedba enega poskusa (50 zagonov 10 problemov) traja skoraj 4 dni. Izvedba teh eksperimentov bi trajala predolgo, saj ob izvedbi eksperimenta (npr. za veliko populacijo, malo populacijo, reinicializacijo velike in majhne populacije, samoprilagajanje krmilnih parametrov F in CR, kopiranje najboljšega posameznika iz velike v majhno populacijo, itd.) in študiji vpliva parametrov dobimo preveč kombinacij. Zato smo načrtovali novo verzijo algoritma (poimenujmo ga a200-25), ki ima naslednje mehanizme in nastavitve:

- ohranimo inicializacijo, kot jo ima jDE100,
- ohranimo dve populaciji, a spremenimo velikost ene izmed njiju. Velikost velike populacije je 200 velikost majhne populacije pa 25. (Spomnimo, da ima jDE100 velikosti populacij 1000 in 25.),
- malenkost spremenimo tudi vrednosti nekaterih parametrov, ki so predvsem posledica zmanjšanja veli-

kosti velike populacije. Vrednost $F_l = 1.0/\sqrt{bNP}$ v primeru velike populacije, in $F_l = 1.0/\sqrt{sNP}$ v primeru majhne populacije. Vrednosti bNP in sNP označujeta velikost velike in majhne populacije. Parameter F_l je enak tudi v algoritmu rjDE, kjer imamo le eno populacijo. $CR_l = 0.0$ za vse funkcije, razen za F9, kjer je $CR_l = 1.0$.

4 Eksperiment in rezultati

Rezultati, ki smo jih dobili, ko smo 50-krat pognali algoritem a200-25 na funkcijah 100-Digit Challenge, so prikazani v tabeli 3. Opazimo lahko, da je algoritem uspešno reševal 9 funkcij in da je bila najtežja F9. Dodatno lahko opazimo, da je v enem primeru uspel najti rešitev le na 9 decimalk natančno, in sicer pri funkciji F3.

Analizirajmo funkcijo F9. Če je možnih 10 točk pri vsaki funkciji, potem je vrednost 4.36 nekoliko nižja od polovice. Poudarimo, da je algoritem uspel najti 5 pravilnih decimalk v 5 primerih, v enem primeru je uspel najti vseh 10 decimalk. Primerjava z rezultati ostalih algoritmov, ki so sodelovali v tekmovanju, kaže, da je algoritem a200-25 dosegel kar solidne rezultate tudi na funkciji F9, ki je povzročala največ težav tekmovalnim algoritmom.

Omenimo, da je algoritem rjDE dosegel 78,6 točk na funkcijah s tekmovanja 100-Digit Challenge pri $maxFES = 10^7$ [1], algoritem DE 42,68 točk in *jDE* 66,48 točk. Ugotovimo lahko, da je vrstni red algoritmov (DE, *jDE*, rjDE) določen s številom mehanizmov – bolj sofisticiran algoritem je dosegel večje število točk. Ko smo algoritem rjDE pozneje pognali z $maxFES = 10^9$, je dosegel 85 točk.

Primerjava a200-25 z algoritmi DE, *jDE* in rjDE pa pove, da sta dve populaciji pomembni, saj sta algoritmu a200-25 prinesli več točk. Prav tako je pomemben tudi parameter CR. In zadnja, kar pa ni nujno najmanj pomembna, ugotovitev iz eksperimentalnega dela v tem prispevku, je parameter velikosti velike populacije. Funkcija F9 oziroma rezultati te funkcije kažejo, da je včasih potrebna tudi nekoliko večja populacija.

Zahvala

J. Brest in B. Bošković priznavata financiranje prispevka s strani Javne agencije za raziskovalno dejavnost Republike Slovenije, raziskovalni program P2-0041 – Računalniški sistemi, metodologije in inteligentne storitve. M. S. Maučec priznava financiranje prispevka s strani Javne agencije za raziskovalno dejavnost Republike Slovenije, raziskovalni program P2-0069 – Napredne metode interakcij v telekomunikacijah.

5 Zaključek

V prispevku smo načrtovali novo verzijo algoritma za reševanje problema 100-Digit Challenge z namenom, da smo lahko analizirali in izpostavili mehanizme in nastavitve pomembnih vrednosti parametrov, s pomočjo katerih je algoritem jDE100 dokaj uspešno reševal probleme na omenjenem tekmovanju. Radi bi izpostavili, da je velikost populacije imela tudi vpliv na uspešnost pridobljenih točk pri najtežji funkciji na tekmovanju, F9, z imenom "Happy Cat".

Nadaljnje smernice razvoja algoritmov za izziv 100-Digit Challenge vključujejo razvoj novih mehanizmov, s katerimi bi zmanjšali število potrebnih evaluacij funkcij, predvsem na funkcijah F8 in F9.

Literatura

[1] Amina Alić, Klemen Berkovič, Borko Bošković, and Janez Brest. Population size in differential evolution. In Aleš Zamuda, editor, *7-th Joint International Conferences on Swarm, Evolutionary and MemeticComputing Conference (SEMCCO 2019) & Fuzzy And Neural Computing Conference(FANCCO 2019)*, 2019.

[2] Borko Bošković and Janez Brest. Protein folding optimization using differential evolution extended with local search and component reinitialization. *Information Sciences*, 454:178–199, 2018.

[3] J. Brest, M. Sepesy Maučec, and B. Bošković. The 100-digit challenge : algorithm jDE100. In *IEEE Congress on Evolutionary Computation (CEC) 2019*, pages 19–26. IEEE, 2019.

[4] Janez Brest, Sašo Greiner, Borko Bošković, Marjan Mernik, and Viljem Žumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[5] Mirjam Sepesy Maučec, Janez Brest, Borko Bošković, and Zdravko Kačič. Improved Differential Evolution for Large-Scale Black-Box Optimization. *IEEE Access*, 2018.

[6] K. V. Price, N. H. Awad, M. Z. Ali, and P. N. Suganthan. Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization. Technical report, Nanyang Technological University, Singapore, November 2018.

[7] 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.

[8] Rainer Storn and Kenneth Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[9] Aleš Zamuda and Janez Brest. Self-adaptive control parameters' randomization frequency and propagations in differential evolution. *Swarm and Evolutionary Computation*, 25:72–99, 2015.