Diferenciální evoluce s adaptací velikosti populace v závislosti na diverzitě

Radka Poláková
Slezská univerzita v Opavě
746 01 Opava, Na Rybníčku 626/1
Email: radka.polakova@fpf.slu.cz

Petr Bujok
Ostravská univerzita
701 03 Ostrava, Dvořákova 7
Email: petr.bujok@osu.cz

Abstrakt

V článku popisujeme nový mechanismus adaptace velikosti populace v algoritmu diferenciální evoluce. Navržený mechanismus je založen na lineárním snižování míry diverzity populace a dovoluje jak snížení velikosti populace tak i její zvýšení. Efektivitu několika variant algoritmu diferenciální evoluce s a bez adaptivního mechanismu jsme experimentálně porovnali na sadě testovacích funkcí vytvořených pro CEC 2014. Navíc jsme mechanismus porovnali s lineárním snižováním velikosti populace. Výsledky porovnání ukazují, že použití navrženého mechanismu je z hlediska efektivity hledání optima výhodné ve více než polovině testovaných úloh, naopak výsledky se implementací mechanismu zhorší jen zřídka.

1 Optimalizace diferenciální evolucí

Optimalizovat funkci, tj. nalézt bod globálního optima, lze matematickými prostředky. Tento proces je ale u některých funkcí, které např. nejsou diferencovatelné, obtížný nebo přímo nemožný. Avšak optimalizovat funkci lze také prostředky stochastickými, tedy s využitím stochastických algoritmů. Mezi tyto patří také algoritmus diferenciální evoluce (DE). Algoritmus byl poprvé předveden v Storn a Price (1997), práce má aktuálně více než 14000 citací.

Podle původního návrhu algoritmus diferenciální evoluce během výpočtu pro konkrétní funkci pracuje s konstantně nastavenými parametry. Od vzniku algoritmu byly vyvinuty mnohé jeho adaptivní verze. Jednou z nich je algoritmus LSHADE (Tanabe a Fukunaga, 2014), pro který autoři navrhli mechanismus lineárního snižování velikosti populace. Algoritmus LSHADE uspěl v optimalizační soutěži na kongresu CEC (Liang a spol., 2013, 2014). Mechanismus lineárního snižování velikosti populace se stal populárním a využívá jej mnoho verzí DE (Guo a spol., 2015; Awad a spol., 2016; Brest a spol., 2016, 2017), které vznikly po algoritmu LSHADE. Tyto verze se ukázaly efektivními, viz výsledky soutěží CEC - Liang a spol. (2015); Suganthan a spol. (2016); Awad a spol. (2017a). Z uvedeného plyne, že adaptace velikosti populace v algoritmu DE je důležitá. Vhodný výběr hodnoty parametru velikosti populace může podstatně zvýšit efektivitu algoritmu.

Náš návrh adaptace velikosti populace předložený v Poláková a spol. (2019) dovoluje jak zmenšování populace tak i její zvětšování a spočívá v udržování diverzity populace v míře adekvátní fázi výpočtu algoritmu, tj. na začátku výpočtu velkou, na konci výpočtu minimální a během výpočtu algoritmu lineárně se zmenšující. V tomto článku jsme se rozhodli se k popisu mechanismu vrátit, popsat jej v češtině a podat tak informace o něm širšímu publiku.

2 Diferenciální evoluce

Diferenciální evoluce (Storn a Price, 1997) je populační algoritmus pro globální optimalizaci. Pracuje s populací P množiny NP bodů. NP je velikost populace. Populace se během procesu hledání globálního optima vyvíjí. Prvky populace uvažujeme jako kandidáty na řešení. Populace je inicializována náhodně v celém prohledávaném prostoru $S = \Pi_{j=1}^{D}[a_j, b_j], a_j < b_j, j = 1, 2, \ldots, D$. D je dimenze problému. Po inicializaci populace bodů následuje opakování cyklu až do splnění podmínky k ukončení výpočtu. Ukončovací podmínka je často dána jako maximální možný počet výpočtů optimalizované funkce. V těle cyklu se k aktuální populaci P vytváří nová populace Q. Ke každému bodu x_i populace P je vytvořen y_i, tzv. pokusný bod. Jestliže platí, že $f(y_i) \leq f(x_i)$, kde f je optimalizovaná funkce, stává se prvkem nové populace Q pokusný bod y_i, pokud podmínka neplatí, je do populace Q vložen bod x_i.[1] Populace Q je na začátku každého běhu cyklu inicializována jako prázdná množina. Po vytvoření nové populace Q se tato populace Q stává populací P a není-li splněna podmínka ukončení algoritmu, cyklus se opakuje. Každý pokusný bod y_i je vytvořen s využitím operací mutace a křížení.

Mutací vzniká tzv. mutant vektor (bod v prostoru) v_i a to nejčastěji přidáním F-násobku rozdílu nebo rozdílů dvojice či několika dvojic nějakých bodů (prvků) populace k nějakému dalšímu bodu, tzv. základnímu bodu mutace. Vstupní parametr F ovlivňuje vzdálenost mutanta a základního bodu mutace. V DE

[1] Protože maximalizovat funkci g znamená minimalizovat funkci $-g$, můžeme se o optimalizaci bavit jako o minimalizaci.

existuje několik druhů mutace. Pokusný bod y_i je vytvořen ze dvou bodů, původního bodu populace x_i a mutanta v_i, křížením. V diferenciální evoluci je podle původního návrhu možné využít jeden ze dvou druhů křížení, binomické nebo exponenciální. Obě varianty křížení využívají vstupní parametr CR, který ovlivňuje podíl složek mutanta, které přecházejí do pokusného bodu. Kombinace mutace a křížení (tzv. DE-strategie), je často zkracována jako $DE/m/n/c$, kde m je použitá mutace, n je počet rozdílů (diferencí) využitých při vytváření mutanta a c je využitý typ křížení. DE-strategie společně s hodnotami parametrů F a CR se nazývá DE-nastavení.

3 Adaptace velikosti populace

Adaptace velikosti populace byla využita jako nástroj k řízení diverzity populace také v Arabas a spol. (1994). V práci je uvedena i myšlenka, že je-li populace příliš malá, může algoritmus předčasně konvergovat a naopak, když je populace příliš velká, může docházet k mrhání výpočetními zdroji. Autoři také zmiňují, že v různých fázích evolučního procesu může být výhodná jiná velikost populace.

Jeden z prvních mechanismů adaptace velikosti populace pro diferenciální evoluci byl uveden v Teo (2006). Autor navrhl dvě modifikace adaptivního mechanismu DESAP. V tomto mechanismu se s každým bodem populace ukládal ještě parametr velikosti populace, který se během výpočtu algoritmu adaptoval a po vytvoření celé nové generace populace se hodnoty tohoto parametru využily pro výpočet velikosti populace pro další průběh algoritmu. Jedna verze mechanismu DESAP pracuje s absolutní velikostí populace a druhá s relativní.

Dalších Z mechanismů adaptace velikosti populace v DE je mechanismus navržený v Brest a Maučec (2008). Algoritmus DE zde začíná pracovat s populací velkého rozsahu. Po určité části výpočtu se velikost populace zmenší na polovinu a to se opakuje až do konce běhu algoritmu. V Wang a Zhao (2013) a Zhu a spol. (2013) se velikost populace upravuje v závislosti na zlepšení či nezlepšení aktuálního řešení problému. V článku Salehinejad a spol. (2017) se pracuje s velikostí populace tak, že dojde-li v generaci k zlepšení řešení, velikost populace se může zmenšit nebo zůstává stejná. Když ke zlepšení nedojde, jsou do populace přidány nové body, či se velikost populace nemění.

Lineární snižování velikosti populace bylo navrženo k zlepšení efektivity algoritmu SHADE (Tanabe a Fukunaga, 2013), vznikl tak algoritmus LSHADE (Tanabe a Fukunaga, 2014). Tento způsob adaptace se objevil v mnoha DE-verzích, které byly navrženy po LSHADE, jedná se např. o iLSHADE (Brest a spol., 2016), jSO (Brest a spol., 2017).

V práci Awad a spol. (2017b) se pro snižování velikosti populace využívá následující schéma. NP se začíná snižovat až od poloviny výpočtu algoritmu. Pro každý bod je vypočítána jeho Eukleidovská vzdálenost od nejlepšího bodu, následně je populace podle těchto vzdáleností setříděna a pak, takto setříděná, je rozdělena do dvou stejně velkých částí. Populace se redukuje lineárně a odstraňují se body patřící do druhé části populace, té horší.

Další články z oblasti DE pracující s diverzitou populace jsou např. Weber a spol. (2009) nebo Yang a spol. (2013) nebo Yang a spol. (2014). Mechanismus uvedený v Gonuguntla a spol. (2015) se zaměřuje na úsporu výpočtů optimalizované funkce v zájmu vyšší efektivity algoritmu. V mechanismu navrženém pro algoritmu FDSADE (Tirronen a Neri, 2009) se zohledňuje diverzita hodnot optimalizované funkce v populaci. Obsáhlý přehled verzí DE zahrnující i verze adaptující velikost populace je možné nalézt v Neri a Tirronen (2010) a nebo např. v Piotrowski (2017).

4 Mechanismus adaptace velikosti populace založený na diverzitě

Diverzitu populace označíme DI a měříme následujícím vztahem

$$DI = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}\sum_{j=1}^{D}(x_{ij} - \overline{x}_j)^2}, \tag{1}$$

kde $\overline{x}_j$ je aritmetický průměr j-tých souřadnic aktuální generace populace bodů

$$\overline{x}_j = \frac{1}{NP}\sum_{i=1}^{NP} x_{ij}. \tag{2}$$

DI je odmocnina z průměrného čtverce vzdálenosti bodu populace a jejího těžiště $\overline{\mathbf{x}} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_D)$. Je zřejmé, že $DI \geq 0$ a když se DI rovná 0, tak jsou všechny body v populaci totožné. Diverzitu počáteční generace populace bodů označme DI_{init}. Tuto DI_{init} použijme jako referenční hodnotu v definici relativní míry diverzity aktuální generace populace

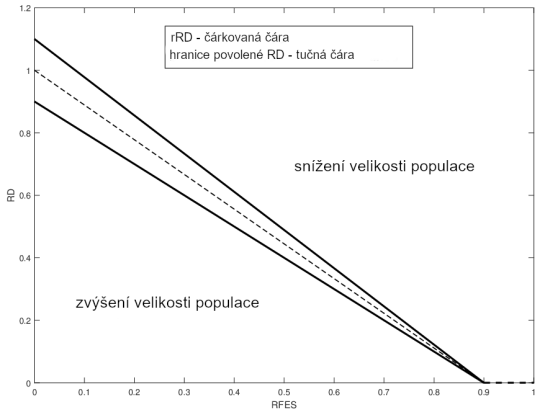
$$RD = \frac{DI}{DI_{init}}. \tag{3}$$

Relativní počet využitých vyhodnocení funkce je definováno následujícím výrazem

$$RFES = \frac{FES}{MaxFES}, \tag{4}$$

kde FES je aktuální počet využitých vyhodnocení optimalizované funkce a $MaxFES$ je celkový povolený počet vyhodnocení funkce během výpočtu. Velikost populace se mění v závislosti na aktuální relativní diverzitě populace. Relativní diverzitu RD jsem navrhli udržovat blízkou rRD. Tato vyžadovaná rRD je

lineárně se snižující od hodnoty 1 na začátku výpočtu až po hodnotu 0 na konci výpočtu algoritmu. Koncept blízkosti RD lineárně se snižující rRD je ilustrován na Obrázku 1. Velikost populace se mění v případě, že



Obr. 1: Lineárně se snižující rRD během výpočtu a hranice pro akceptovanou RD.

je RD vyšší než $1{,}1 \times rRD$, nebo naopak nižší než $0{,}9 \times rRD$. Toto pravidlo platí pro prvních devět desetin délky výpočtu algoritmu, v poslední desetině výpočtu je požadována nulová hodnota relativní diverzity. RD se vypočítává po každé nově vytvořené generaci populace. Když je splněna podmínka pro změnu velikosti populace, tj. relativní diverzita RD není v blízkosti rRD, NP vzroste o jedničku a do populace je přidán náhodně vygenerovaný bod z prohledávaného prostoru, to v případě, že RD je menší než $0{,}9 \times rRD$. Velikost populace NP se o 1 zmenší, tj. z populace je vyřazen nejhorší bod, v případě, že RD je větší než $1{,}1 \times rRD$.

Velikost populace je třeba udržovat v nějakém „rozumném“ intervalu, není možné, aby např. neomezeně rostla. Prohledávací proces tedy začíná s populací, jejíž velikost je rovna NP_{init} a je udržována v intervalu $\langle NP_{min}, NP_{max} \rangle$. Uvedené parametry jsme v závislosti na výsledcích předchozích experimentů (Poláková, 2017; Poláková a spol., 2017) nastavili na následující hodnoty, $NP_{init} = 50$, $NP_{min} = 8$, $NP_{max} = 5 \times D$.

5 Varianty diferenciální evoluce využité k testování navrženého mechanismu

K otestování efektivity navrženého mechanismu jsme zvolili 8 variant diferenciální evoluce. Jsou to: originální verze algoritmu DE, tři adaptivní verze, které se podle Das a Suganthan (2010), Das a Suganthan (2016) a Al-Dabbagh a spol. (2018) řadí mezi tzv. „state-of-the-art“ algoritmy, jmenovitě CoDE (Wang a spol., 2011b), EPSDE (Mallipeddi a spol., 2011) a jDE (Brest a spol., 2006). Pátým algoritmem zařazeným do našich testů je algoritmus *b6e6rl* (Tvrdík a Poláková, 2013), jedná se o efektivní verzi soutěživé DE navržené v Tvrdík (2006). Dalším algoritmem zahrnutým v testech je SHADE (Tanabe a Fukunaga, 2013), vítěz soutěže CEC 2013 (Loshchilov a spol., 2013). Modifikace tohoto algoritmu (Tanabe a Fukunaga, 2014) s implementovaným mechanismem lineárního snižování velikosti populace je také velmi úspěšným algoritmem (Liang a spol., 2014). Dalším algoritmem využitým v našich experimentech je algoritmus IDE (Tang a spol., 2015) a posledním algoritmem je algoritmus jSO (Brest a spol., 2017), který byl druhým algoritmem v pořadí [2] a současně nejlepší DE variantou na soutěži optimalizačních algoritmů na kongresu CEC 2017 (Awad a spol., 2017a).

Původní verze DE je v tomto článku využita s nejčastěji používanou *DE/rand/1/bin* strategií, nastavení vstupních parametrů je následující, $F = 0{,}8$, $CR = 0{,}5$. Algoritmus CoDE (Wang a spol., 2011b) vytváří pro každý bod vždy tři adepty na pokusný bod a jeden z nich pak vybírá na základě hodnoty optimalizované funkce v nich jako finální. CoDE vytváří adepty na pokusný bod třemi strategiemi, *DE/rand/1/bin*, *DE/rand/2/bin* a *DE/current-to-rand/1/-*, a dvojice parametrů k nim vybírá z následujících dvojic, $(1; 0{,}1)$, $(1; 0{,}9)$ a $(0{,}8; 0{,}2)$.

Algoritmus EPSDE (Mallipeddi a spol., 2011) využívá množinu strategií, množinu hodnot pro parametr F a množinu hodnot pro parametr CR. Každý bod populace má přiřazenu trojici parametrů (strategie, F, CR). Pokud je tato trojice úspěšná, tj. vytvoří pokusný bod lepší než bod původní, zůstává trojicí parametrů pro tento bod. V opačném případě, se buď generuje nová trojice nebo se vybírá náhodně některá trojice parametrů z úspěšných trojic, které se během celého výpočtu ukládají. Množina strategií obsahuje *DE/best/2/bin*, *DE/rand/1/bin* a strategii *DE/current-to-rand/1/-*, množina hodnot pro parametr F je $\{0{,}4; 0{,}5; 0{,}6; 0{,}7; 0{,}8; 0{,}9\}$ a množina hodnot pro parametr CR je $\{0{,}1; 0{,}2; 0{,}3; 0{,}4; 0{,}5; 0{,}6; 0{,}7; 0{,}8; 0{,}9\}$.

Algoritmus jDE (Brest a spol., 2006) je jednou z prvních a současně velmi efektivních adaptivních verzí DE. Pracuje se strategií *DE/rand/1/bin*. Při inicializaci populace se ke každému bodu populace inicializují i jeho vlastní parametry F a CR. Každý z nich se může s malou pravděpodobností před každým výpočtem pokusného bodu změnit - reinicializovat. V případě, že pomocí takto změněné dvojice parametrů se vytvoří úspěšný pokusný bod, tato nová dvojice parametrů F a CR se pak stává dvojicí parametrů příslušející tomuto prvku populace v dalším výpočtu. V případě opačném se příslušnému bodu navrací zpět jeho původní dvojice hodnot parametrů.

Soutěživá DE (Tvrdík, 2006) pracuje s několika nastaveními algoritmu DE, které mají na začátku

[2] Algoritmus jSO byl dokonce původně deklarován jako celkový vítěz soutěže.

výpočtu stejnou pravděpodobnost využití při tvorbě pokusného bodu. Čím je nastavení úspěšnější, tj. čím častěji vytváří pokusné body lepší než body původní, tím více se zvyšuje pravděpodobnost jejího využití. V případě, že některá z pravděpodobností je příliš malá, pravděpodobnosti se „resetují“ na navzájem rovnající se hodnoty. Algoritmus *b6e6rl* (Tvrdík a Poláková, 2013) pracuje se dvěma strategiemi (*DE/randrl/1/bin* a *DE/randrl/1/exp*), parametr F může nabývat jedné ze dvou hodnot $0{,}5$ a $0{,}8$ a parametr CR má tři možné hodnoty. Různé kombinace těchto hodnot F a CR vedou k 6 různým nastavením se strategií *DE/randrl/1/bin* a 6 různým nastavením se strategií *DE/randrl/1/exp*. V *b6e6rl* tedy soutěží těchto dvanáct nastavení DE.

SHADE (Tanabe a Fukunaga, 2013) je algoritmus postavený na základech algoritmu JADE (Zhang a Sanderson, 2009) s mutací *current-to-pbest/1* a archivem, do kterého se zapisují prvky populace, které byly v populaci přepsány svým pokusným bodem. V JADE se F a CR generují z Cauchyho resp. normálního rozdělení. První parametry těchto rozdělení (k využití v následující generaci) se počítají ze všech v aktuální generaci úspěšně využitých hodnot odpovídajících parametrů. V SHADE se využívá adaptace parametrů F a CR „zděděná“ z JADE, jen jsou zde navíc použity dvě kruhové paměti o velikosti H. V SHADE se z úspěšně využitých hodnot parametrů F a CR počítají opět první parametry - parametry polohy pro obě rozdělení, jen se zde uchovává posledních H takto vypočítaných hodnot. Při generování F a CR pro využití k vytvoření pokusného bodu se nejdříve náhodně zvolí jeden z indexů i, $i \in \{1, 2, 3, \ldots, H\}$, v kruhových pamětech a F je náhodné číslo z Cauchyho rozdělení s prvním parametrem z příslušné kruhové paměti s indexem i a CR je náhodné číslo z normálního rozdělení s prvním parametrem z příslušné kruhové paměti s indexem i. Všechny prvky v kruhových pamětech jsou inicializovány na hodnotu $0{,}5$. Druhý parametr obou využitých rozdělení je stále roven $0{,}1$.

Algoritmus IDE (Tang a spol., 2015) pracuje s populací bodů rozdělenou na dvě části, S (lepší) a I (horší). Velikost těchto dvou částí populace se za běhu algoritmu mění. Na začátku výpočtu je část S málo početná, naopak část I obsahuje celý zbytek populace. Takto zůstávají, co se týká velikosti, obě části populace až téměř do konce výpočtu algoritmu, následně velikost části populace S rychle roste, až je rovna velikosti populace. IDE využívá nově nadefinovanou mutaci. Dále i výpočet algoritmu je rozdělen na dvě části. V první části algoritmus využívá jako základní bod mutace náhodně vybraný bod populace. V druhé části běhu algoritmu se jako základní bod pro mutaci využívá aktuálně nejlepší bod populace. Navíc i využívaný typ mutace pracuje jinak pro část populace S a jinak pro část I. Parametry F a CR jsou nastaveny v závislosti na pořadí bodu odlišně pro každý bod populace, nejmenší hodnoty pro nejlepší bod a největší hodnoty pro nejhorší bod populace. Navíc je v IDE implementován mechanismus, kterým se předchází předčasné konvergenci.

Algoritmus jSO (Brest a spol., 2017) je vylepšené iLSHADE (Brest a spol., 2016), což je vylepšené LSHADE (Tanabe a Fukunaga, 2014). LSHADE je SHADE s lineárním snižováním velikosti populace. Algoritmus jSO má ve srovnání s LSHADE několik odlišných vlastností. jSO také využívá mechanismus lineárního snižování velikosti populace, ale výpočet začíná s $NP = 25 \times \sqrt{D} \times logD$ místo $18 \times D$. Parametr p, který využívá mutace *current-to-pbest/1*, se během výpočtu lineárně snižuje od $0{,}25$ do $0{,}125$, zatímco v LSHADE je p konstantní. Velikost kruhových pamětí je pro jSO nastavena na hodnotu 5. Ostatní vlastnosti jSO algoritmu najdete v Brest a spol. (2017).

6 Experimenty

Pro posouzení efektivity navrženého mechanismu adaptace velikosti populace jsme využili všech osmi variant DE popsaných v kapitole 5. V algoritmech DE, CoDE, EPSDE, jDE, *b6e6rl*, SHADE a IDE se neadaptuje velikost populace. Naproti tomu v algoritmu jSO je implementován mechanismus lineárního snižování velikosti populace. Z tohoto důvodu jsme pro každý z algoritmů DE, CoDE, EPSDE, jDE, *b6e6rl*, SHADE a IDE vytvořili další dvě verze. První s implementací mechanismu adaptace velikosti populace založeného na její diverzitě, tuto verzi jsme vždy označili předponou „d“, druhou s implementací lineárního snižování velikosti populace, tuto verzi jsme vždy označili předponou „L“. Takto jsme získali 21 různých algoritmů, tj. původních sedm algoritmů, sedm algoritmů s mechanismem diverzity (dDE, dCoDE, dEPSDE, atd.) a sedm algoritmů s lineárním snižováním velikosti populace (LDE, LCoDE, LEPSDE, LjDE, atd.). K takto získaným 21 algoritmům jsme ještě přidali jSO a algoritmus, který jsme získali z jSO odebráním mechanismu lineárního snižování velikosti populace a následnou implementací mechanismu řízení velikosti populace na základě diverzity, tento algoritmus jsme označili djSO.

Pro experimenty jsme zvolili testovací sadu 30 funkcí vytvořenou pro soutěž optimalizačních algoritmů uspořádanou v rámci kongresu CEC 2014 (Liang a spol., 2013). Maximální velikost populace při testech algoritmů s implementovaným námi navrženým mechanismem byla nastavena na hodnotu $NP_{max} = 5 \times D$. Nastavení ostatních parametrů každého z algoritmů jsme převzali z jejich původní definice. Všech 23 algoritmů jsme testovali na čtyřech úrovních dimenze, $D = 10, 30, 50, 100$. Pro každý z 23 testovaných algoritmů a každý z 30×4 optimalizačních problémů jsme provedli 51 opakování (51 běhů daného algoritmu). Tedy jsme získali 51 výsledků, tj. 51 minim, nalezených daným algoritmem k danému optimalizačnímu problému. Všech takto získaných 23×120 (2760) sad 51 výsledků jsme

zhodnotili níže uvedeným způsobem.

Všechny testované algoritmy byly implementovány v software Matlab 2010b a všechny výpočty byly provedeny na standardním PC s Windows 7 a konfigurací: Intel(R) Core(TM)i7-4790 CPU 3.6 GHz, 16 GB RAM. Všechny statistické výpočty byly provedeny v R software (R Core Team, 2015).

7 Výsledky experimentů

Nejdříve nás zajímal vliv implementace navrženého adaptivního mechanismu na efektivitu sedmi testovaných algoritmů, které v původní verzi velikost populace neadaptují. Porovnání algoritmů s pevně stanovenou velikostí populace pro celý běh algoritmu (DE, CoDE, EPSDE, jDE, *b6e6rl*, SHADE a IDE) s jejich variantami, které využívají mechanismus úpravy velikosti populace na základě diverzity (dDE, dCoDE, dEPSDE, djDE, *db6e6rl*, dSHADE a dIDE) je uvedeno v Tabulce 1, kde jsou shrnuty výsledky 840 dvouvýběrových Wilcoxonových testů. Z tabulky je zřejmé, že implementace adaptivního mechanismu založeného na diverzitě zvyšuje efektivitu testovaných algoritmů (kromě IDE, kde je počet vítězství 47) ve více než polovině z testovaných optimalizačních problémů. Počet vítězství d-verze je u všech algoritmů vyšší než počet proher. Pokud uvažujeme každou testovanou dimenzi samostatně, tak i zde platí téměř ve všech případech, že výher algoritmu s implementací navrženého adaptivního mechanismu je více než jeho proher, jedinou výjimkou je opět IDE v dimenzi $D = 100$, zde má dIDE o pět proher více než výher. Uvažujeme-li výhry a prohry pro všechny algoritmy dohromady, každou dimenzi zvlášť, ve všech třech vyšších dimenzích je podíl výher zhruba $2/3$. V dimenzi $D = 10$ je podíl výher sice menší, ale na druhou stranu je počet proher v této dimenzi minimální. Uvažujeme-li celkově všechny dimenze dohromady pro všechny testované algoritmy dohromady, je podíl výher verzí s navrženým mechanismem vyšší než 60 %, zatímco původní verze byly úspěšnější pouze zhruba v 11 % z testovaných úloh. Ve zhruba 26 % testovaných úloh nebyl rozdíl v efektivitě původní verze a jeho verze s implementací mechanismu adaptace velikosti populace na základě diverzity statisticky významný.

Dále nás zajímalo, zda je efektivnější lineární snižování velikosti populace nebo, zda k větší efektivitě námi testovaných verzí DE vede implementace mechanismu založeného na diverzitě populace. Porovnání algoritmů z těchto dvou skupin variant testovaných algoritmů je uvedeno v Tabulce 2, kde jsou shrnuty výsledky 960 dvouvýběrových Wilcoxonových testů.

Výsledky tohoto porovnání ukazují, že pro všechny algoritmy, kromě jSO, platí, že ve všech třech vyšších dimenzích je využití nově navrženého mechanismu adaptace velikosti populace výhodnější než využití mechanismu lineárního snižování velikosti populace. Fakt, že jSO si vede v tomto porovnání mnohem úspěšněji s lineárním snižováním velikosti populace, je pravděpodobně způsoben tím, že vhodné hodnoty parametrů tohoto algoritmu byly v průběhu jeho vývoje testovány právě s mechanismem lineárního snižování velikosti populace a změna mechanismu za jiný pravděpodobně velmi narušila jejich optimalitu.

alg.	dimenze	10	30	50	100	$\sum$
DE	# vítěz.	26	27	24	27	104
	# proher	0	0	0	0	0
	# $\approx$	4	3	6	3	16
jDE	# vítěz.	13	18	20	14	65
	# proher	1	2	5	11	19
	# $\approx$	16	10	5	5	36
IDE	# vítěz.	5	14	17	11	47
	# proher	1	4	4	16	25
	# $\approx$	24	12	9	3	48
SHADE	# wins	7	24	23	23	77
	# proher	0	0	3	5	8
	# $\approx$	23	6	4	2	35
b6e6rl	# vítěz.	9	19	20	18	66
	# proher	0	1	4	11	16
	# $\approx$	21	10	6	1	38
CoDE	# vítěz.	25	20	22	21	88
	# proher	0	4	5	4	13
	# $\approx$	5	6	3	5	19
EPSDE	# vítěz.	18	19	23	22	82
	# proher	0	2	4	6	12
	# $\approx$	12	9	3	2	26
$\sum$	# vítěz.	103	141	149	136	529
	# proher	2	13	25	53	93
	# $\approx$	105	56	36	21	218

Tab. 1: Počet vítězství a proher d-mechanismu proti pevně nastavené velikosti populace - výsledky na základě 840 výsledků výpočtu Wilcoxonova dvouvýběrového statického testu.

Když neuvažujeme jSO, které má pro dimenzi $D = 10$ podobné výsledky jako v ostatních dimenzích, pak pro dimenzi $D = 10$ jsou mezi testovanými algoritmy tři, pro které je využití námi navrhovaného mechanismu úpravy velikosti populace méně výhodné než využití lineárního snižování velikosti populace, jsou to jDE, SHADE a *b6e6rl*. Podíváme-li se však detailněji, jsou pro jDE a SHADE počty proher a výher v $D = 10$ srovnatelné a pouze pro algoritmus *b6e6rl* je počet proher v této dimenzi znatelně vyšší než počet výher. Pro ostatní čtyři testované algoritmy (DE, IDE, CoDE a EPSDE) je i v dimenzi $D = 10$ využití d-mechanismu výhodnější. Uvažujeme-li každý z testovaných algoritmů samostatně (dohromady ve všech dimenzích) jsou v našem porovnání pouze dva algoritmy (neuvažujeme-li jSO), pro které platí, že počet výher d-verze algoritmu není větší než polovina ze všech tes-

tovaných úloh. Jsou to jDE a SHADE. V obou těchto případech je však stále více výher d-verze algoritmu než-li jeho proher. Nyní se věnujme každé z dimenzí samostatně. V každé z dimenzí se objevuje zhruba 40 proher (z 240 případů) d-mechanismu. S rostoucím dimenzí se zvětšuje i počet výher d-mechanismu, naopak počet shod s rostoucí dimenzí klesá. Celkově v tomto porovnání d-mechanismus nad L-mechanismem vyhrál ve více než 60 % všech případů, prohrál ve zhruba 17 % případů a nevýznamný rozdíl ve výsledcích nastal ve zhruba 23 % případů.

alg.	dim.	10	30	50	100	$\sum$
DE	# vítěz.	23	30	29	27	109
	# proher	1	0	0	0	1
	# $\approx$	6	0	1	3	10
jDE	#vítěz.	8	12	16	20	56
	# proher	10	6	7	7	30
	# $\approx$	12	12	7	3	34
IDE	# vítěz.	17	27	28	28	100
	# proher	2	0	0	0	2
	# $\approx$	11	3	2	2	18
SHADE	#vítěz.	7	12	13	15	47
	# proher	8	8	7	10	33
	# $\approx$	15	10	10	5	40
b6e6rl	# vítěz.	3	16	23	25	67
	# proher	9	3	2	4	18
	# $\approx$	18	11	5	1	35
CoDE	# vítěz.	22	29	29	29	109
	# proher	0	1	1	1	3
	# $\approx$	8	0	0	0	8
EPSDE	# vítěz.	19	21	21	21	82
	# proher	0	2	2	5	9
	# $\approx$	11	7	7	4	29
jSO	# vítěz.	3	4	6	6	19
	# proher	16	14	17	17	64
	# $\approx$	11	12	7	7	37
$\sum$	# vítěz.	101	151	165	171	589
	# proher	46	34	36	44	160
	# $\approx$	93	55	39	25	211

Tab. 2: Počet vítězství a proher d-mechanismu proti L-mechanismu - výsledky na základě 960 výsledků výpočtu Wilcoxonova dvouvýběrového statického testu.

Výsledky všech 23 testovaných variant algoritmu diferenciální evoluce byly porovnány Friedmanovým testem, do testu vstupoval vždy medián ze všech 51 nalezených minim. Nulová hypotéza o shodě efektivity algoritmů byla pro všechny čtyři dimenze zamítnuta s p-hodnotou menší než $2,2 \times 10^{-16}$. Výsledky tohoto porovnání jsou zachyceny v Tabulce 3, jsou zde uvedena průměrná pořadí algoritmu pro každou dimenzi. Algoritmy jsou seřazeny od nejlepšího k nejhoršímu podle průměrného pořadí přes všechny 4 dimenze. V závorkách jsou uvedena pořadí algoritmů v rámci dané dimenze. Uvažujeme-li celkové pořadí algoritmů, vidíme, že až na jSO pro všechny testované algoritmy platí, že jejich d-verze je úspěšnější než originální algoritmus a také než jeho L-verze. Pro algoritmy SHADE, jDE, *b6e6rl*, EPSDE je pořadí verzí (d-verze, L-verze, fixní velikost populace). Pro algoritmy IDE, CoDE a DE je pak pořadí verzí (d-verze, fixní velikost populace, L-verze).

Z tabulky je také zřejmý celkový vítěz testu, je jím algoritmu jSO, který byl taky úspěšný na soutěži CEC 2017. jSO je v celkovém pořadí následován algoritmem djSO a třetí algoritmus v pořadí je d-verze algoritmu SHADE. Celkové pořadí algoritmů v Tabulce 3 odpovídá očekávané efektivitě optimalizačních algoritmů. jSO je úspěšný algoritmus, který byl vytvořen v roce 2017. SHADE a IDE jsou efektivní algoritmy, které vznikly několik málo let před jSO. Na druhé straně algoritmy EPSDE a CODE vznikly někdy okolo roku 2010.

Algoritmus	$D = 10$		$D = 30$		$D = 50$		$D = 100$		avg
jSO	6,9	(1)	4,4	(1)	4,3	(1)	4,5	(1)	5,0
djSO	9,9	(8)	5,4	(2)	5,2	(2)	5,8	(2)	6,6
dSHADE	8,5	(4)	6,2	(3)	6,1	(3)	6,0	(3)	6,7
LSHADE	7,5	(3)	6,6	(4)	6,9	(4)	7,1	(4)	7,0
dIDE	7,3	(2)	7,2	(5)	8,0	(5)	9,4	(8)	8,0
djDE	9,9	(9)	9,1	(8)	8,2	(6)	8,5	(5)	8,9
IDE	8,8	(6)	8,4	(6)	10,0	(9)	9,1	(6)	9,1
db6e6rl	10,7	(11,5)	8,7	(7)	8,5	(7)	10,1	(11)	9,5
dEPSDE	10,4	(10)	9,6	(9)	9,3	(8)	9,7	(10)	9,7
LjDE	8,6	(5)	10,4	(10)	10,3	(10)	10,3	(12)	9,9
SHADE	10,7	(11,5)	11,7	(13)	10,8	(12)	9,5	(9)	10,7
jDE	12,5	(15)	12,0	(14)	10,7	(11)	9,2	(7)	11,1
Lb6e6rl	9,2	(7)	10,7	(11)	11,6	(13)	13,5	(16)	11,3
b6e6rl	12,8	(16)	11,3	(12)	12,5	(14)	11,0	(13)	11,9
LEPSDE	13,8	(17)	13,9	(16)	13,1	(15)	11,9	(14)	13,2
EPSDE	14,7	(18)	13,0	(15)	13,7	(16)	12,2	(15)	13,4
LIDE	11,7	(13)	14,1	(17)	13,9	(17)	15,1	(17)	13,7
dCoDE	12,3	(14)	15,7	(18)	14,9	(18)	15,9	(18)	14,7
dDE	16,1	(20)	17,4	(20)	17,0	(20)	15,9	(19)	16,6
CoDE	18,0	(21)	17,2	(19)	16,3	(19)	17,8	(20)	17,3
LCoDE	16,0	(19)	20,1	(21)	20,6	(21)	20,6	(21)	19,3
DE	20,5	(23)	21,0	(22)	21,7	(22)	21,3	(22)	21,1
LDE	19,2	(22)	22,2	(23)	22,7	(23)	21,8	(23)	21,5

Tab. 3: Pořadí algoritmů v každé z dimenzí (podle výsledků Friedmanova testu) a průměrné pořadí algoritmů.

8 Závěr

V článku jsme navrhli nový adaptivní mechanismus pro úpravu velikosti populace v algoritmu diferenciální evoluce. Mechanismus je založen na řízení míry diverzity populace v diferenciální evoluci. V porovnání s oblíbeným a v poslední době často využívaným adaptivním mechanismem lineárního snižování velikosti populace v DE náš mechanismus nedovoluje pouze snižování velikosti populace, ale také její zvyšování. Námi navržený mechanismus jsme experimentálně porovnali se zmiňovaným lineárním snižováním velikosti populace.

Provedené testy nově navrženého mechanismu přinesly slibné výsledky. Takováto adaptace velikosti populace v DE, v porovnání s využitím mechanismu lineárního snižování velikosti populace, vede

v převažující části testovaných úloh k vyšší efektivitě procesu hledání optima.

V budoucnu bychom se chtěli zabývat implementací navrženého mechanismu do dalších evolučních algoritmů a také hledáním takové modifikace této adaptace, která zlepší i výsledky jednoho z aktuálně nejefektivnějších verzí algoritmu diferenciální evoluce, tedy algoritmu jSO.

Poděkování

Tento příspěvek je financován ze Strukturálních a investičních fondů Evropské unie OP VVV, z projektu „Zvýšení kvality vzdělávání na Slezské univerzitě v Opavě ve vazbě na potřeby Moravskoslezského kraje“, CZ.02.2.69/0.0/0.0/18_058/0010238.

Tento příspěvek je věnován našemu společnému školiteli doc. Josefu Tvrdíkovi, CSc., který nás před necelými dvěma lety opustil. Nebyl to jen náš školitel, ale i dlouholetý kamarád a především úžasný člověk. Je nám ctí, že jsme se mohli učit právě od něj.

Literatura

Al-Dabbagh, R. D., Neri, F., Idris, N. a Baba, M. S. (2018). Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm and Evolutionary Computation*, 43:284–311.

Arabas, J., Michalewicz, Z. a Maluwka, J. (1994). GAVaPS - a genetic algorithm with varying population size. V *Proceedings of IEEE Congress on Evolutionary Computation, 1994*, str. 73–78. IEEE.

Awad, N. H., Ali, M. Z., Liang, J. J., Qu, B. a Suganthan, P. N. (2017a). CEC17 special session on single objective numerical optimization single bound constrained real-parameter numerical optimization.

Awad, N. H., Ali, M. Z. a Suganthan, P. N. (2017b). Ensemble of parameters in a sinusoidal differential evolution with niching-based population reduction. *Swarm and Evolutionary Computation*, 39.

Awad, N. H., Ali, M. Z., Suganthan, P. N. a Reynolds, R. G. (2016). An ensemble sinusoidal parameter adaptation incorporated with l-shade for solving cec2014 benchmark problems. V *IEEE Congress on Evolutionary Computation 2016*, str. 2958–2965.

Brest, J., Greiner, S., Boškovič, B., Mernik, M. a Žumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10:646–657.

Brest, J. a Maučec, M. S. (2008). Population size reduction for the differential evolution algorithm. *Appl Intell*, 29:228–247.

Brest, J., Maučec, M. S. a Boškovič, B. (2016). iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. V *IEEE Congress on Evolutionary Computation 2016*, str. 1188–1195.

Brest, J., Maučec, M. S. a Boškovič, B. (2017). Single objective real-parameter optimization: Algorithm jSO. V *IEEE Congress on Evolutionary Computation 2017*, str. 1311–1318.

Das, S. a Suganthan, P. N. (2010). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15:4–31.

Das, S. a Suganthan, P. N. (2016). Recent advances in differential evolution: An updated survey. *Swarm and Evolutionary Computation*, 27:1–30.

Gonuguntla, V., Mallipeddi, R. a Veluvolu, K. C. (2015). Differential evolution with population and strategy parameter adaptation. *Mathematical Problems in Engineering*, 2015.

Guo, S.-M., Yang, C.-C., Hsu, P.-H. a Tsai, J. S.-H. (2015). A seff-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC2015 benchmark set. V *IEEE Congress on Evolutionary Computation (CEC) 2015 Proceedings*, str. 1003–1010.

Holland, J. (1992). Genetic algorithms - computer programs that evolve in ways that resemble natural selection can solve complex problems even their creators do not fully understand. *Scientific American*, str. 66–72.

Liang, J. J., Qu, B. a Suganthan, P. N. (2013). Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. [online] http://www.ntu.edu.sg/home/epnsugan/.

Liang, J. J., Qu, B. a Suganthan, P. N. (2014). Ranking results of CEC14 special session and competition on real-parameter single objective optimization. [online] http://www3.ntu.edu.sg/home/epnsugan/.

Liang, J. J., Qu, B., Suganthan, P. N. a Chen, Q. (2015). CEC15 competition on learning-based real-parameter single objective optimization.

Loshchilov, I., Stuetzle, T. a Liao, T. (2013). Ranking results of CEC13 special session and competition on real-parameter single objective optimization. [online] http://www3.ntu.edu.sg/home/epnsugan/.

Mallipeddi, R., Suganthan, P. N., Pan, Q. K. a Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11:1679–1696.

Neri, F. a Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review*, 33:61–106.

Piotrowski, A. P. (2017). Review of differential evolution population size. *Swarm and Evolutionary Computation*, 32:1–24.

Poláková, R. (2017). Controlling population size in differential evolution by diversity mechanism. V *International Conference on Artificial Intelligence and Soft Computing: LNAI 10245 Artificial Intelligence and Soft Computing - Part 1*, str. 408–417, Heidelberg. Springer.

Poláková, R., Tvrdík, J. a Bujok, P. (2017). Adaptation of population size according to current population diversity in differential evolution. V *2017 IEEE Symposium Series on Computational Intelligence (SSCI) Proceedings*, str. 2627–2634. IEEE.

Poláková, R., Tvrdík, J. a Bujok, P. (2019). Differential evolution with adaptive mechanism of population size according to current population diversity. *Swarm and Evolutionary Computation*, 50(100519).

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Salehinejad, H., Rahnamayan, S. a Tizhoosh, H. R. (2017). Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Applied Soft Computing*, 52:812–833.

Storn, R. a Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization*, 11:341–359.

Suganthan, P. N., Ali, M. Z. a Awad, N. H. (2016). CEC16 special session on single objective numerical optimization single parameter - operator set based case.

Tanabe, R. a Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. V *IEEE Congress on Evolutionary Computation 2013*, str. 71–78.

Tanabe, R. a Fukunaga, A. (2014). Improving the search performance of SHADE using linear population size reduction. V *IEEE Congress on Evolutionary Computation 2014*, str. 1658–1665.

Tang, L., Dong, Y. a Liu, J. (2015). Differential evolution with an individual-dependent mechanism. *IEEE Transactions on Evolutionary Computation*, 19:560–574.

Teo, J. (2006). Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing*, 10:673–686.

Tirronen, V. a Neri, F. (2009). Differential evolution with fitness diversity self-adaptation. *Nature-Inspired Algorithms for Optimization*, str. 199–234.

Tvrdík, J. (2006). Competitive differential evolution. Matoušek, R. a Ošmera, P. (zost.), V *MENDEL 2006, 12th International Conference on Soft Computing*, str. 7–12, Brno.

Tvrdík, J. a Poláková, R. (2013). Competitive differential evolution applied to CEC 2013 problems. V *IEEE Congress on Evolutionary Computation 2013*, str. 1651–1657.

Wang, H., Rahnamayan, S. a Wu, Z. (2011a). Adaptive differential evolution with variable population size for solving high-dimensional problems. V *IEEE Congress on Evolutionary Computation*, str. 2626–2632.

Wang, X. a Zhao, S. (2013). Differential evolution algorithm with self-adaptive population resizing mechanism. *Mathematical Problems in Engineering*.

Wang, Y., Cai, Z. a Zhang, Q. (2011b). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15:55–66.

Weber, M., Neri, F. a Tirronen, V. (2009). Distributed differential evolution with explorative–exploitative population families. *Genetic Programming and Evolvable Machines*, 10:343–371.

Yang, M., Cai, Z., Li, C. a Guan, J. (2013). An improved adaptive differential evolution algorithm with population adaptation. V *GECCO '13 Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, str. 145–152.

Yang, M., Li, C., Cai, Z. a Guan, J. (2014). Differential evolution with auto-enhanced population diversity. *IEEE Transactions on Cybernetics*, 45:302–315.

Zhang, J. a Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13:945–958.

Zhu, W., Tang, Y., Fang, J. a Zhang, W. (2013). Adaptive population tuning scheme for differential evolution. *Information Sciences*, 223:164–191.