VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Návrh nového algoritmu SOMA pro optimalizaci s omezeními

New SOMA for constrained optimization

2020 Dominik Sobek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Dominik Sobek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Návrh nového algoritmu SOMA pro optimalizaci s omezeními
New SOMA for Constrained Optimization

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je navrhnout vylepšení pro hejnový algoritmus SOMA. A tento algoritmus pak aplikovat na jeden z benchmarků CEC a následující úlohy: návrh parametrů tlakového válce, návrh konstrukce svařovaných nosníků, návrh parametrů pružiny.

1. Student se důkladně seznámí s algoritmem SOMA a s úspěšnými evolučními a hejnovými algoritmy, které byly představeny v průběhu minulých let.
2. Student navrhne netriviální vylepšení algoritmu SOMA - může se jednat o hybridizaci, aplikování tzv. surrogate metod apod.
3. Student provede statistické srovnání výkonu nově navrženého algoritmu s původní verzí SOMA a s vybranými state-of-the-art algoritmy na zvoleném benchmarku CEC a na třech výše zmíněných úlohách.
4. Pro úlohy jako návrh parametrů tlakového válce, návrh konstrukce svařovaných nosníků a návrh pružiny vytvoří student jednoduchou vizualizaci, která bude demonstrovat vývoj nejlepšího řešení.
5. Student důkladně zdokumentuje svůj postup a výsledky v textu závěrečné práce.

Seznam doporučené odborné literatury:

[1] ZELINKA, Ivan. SOMA—self-organizing migrating algorithm. In: New optimization techniques in engineering. Springer, Berlin, Heidelberg, 2004. p. 167-217.
[2] DOS SANTOS COELHO, Leandro; MARIANI, Viviana Cocco. An efficient cultural self-organizing migrating strategy for economic dispatch optimization with valve-point effect. Energy Conversion and Management, 2010, 51.12: 2580-2587.
[3] DEEP, Kusum, et al. A self-organizing migrating genetic algorithm for constrained optimization. Applied Mathematics and Computation, 2008, 198.1: 237-250.
[4] PARPINELLI, Rafael S.; LOPES, Heitor S. New inspirations in swarm intelligence: a survey. International Journal of Bio-Inspired Computation, 2011, 3.1: 1-16.
[5] MARLER, R. Timothy; ARORA, Jasbir S. Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization, 2004, 26.6: 369-395.
[6] COELLO, Carlos A. Coello; MONTES, Efrén Mezura. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics, 2002, 16.3: 193-203.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Lenka Skanderová, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.	prof. Ing. Pavel Brandštetter, CSc.
vedoucí katedry	*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2020 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2020 .....

Tímto bych chtěl poděkovat Ing. Lence Skaderové Ph.D. za vstřícnou pomoc při vytváření této práce a také za volbu velice zajímavého tématu.

Abstrakt

Práce je věnována moderním hejnovým algoritmům v oblasti optimalizace s omezeními. Cílem je nalézt takové mechanizmy a techniky, které by přinesly zlepšení výkonu Samo-organizující se algoritmu (SOMA).
Je zde popsán postupný vývoj vedoucí k nalezení vhodné úpravy. Jednotlivá dílčí řešení jsou okomentována a srovnána s původními strategiemi. Konečná úprava SOMA představuje inovativní přístup k migraci jedinců, kdy dochází ke skokům ke dvěma adaptivně se posunujícím cílovým bodům. Změněn je také způsob perturbace, který nově využívá chaoticky generovaných čísel. Přestavený algoritmus byl srovnán se state-of-art algoritmy na funkci návrhu parametrů tlakového válce, návrhu konstrukce svařovaných nosníků, návrhu parametrů pružiny a na funkcích z benchmarku CEC 2017. Na základě srovnání výkonu algoritmů dle pravidel CEC 2017 a statistických testů lze konstatovat, že nově představená strategie výrazně přispívá ke zlepšení výkonu SOMA.

Klíčová slova: Evoluční algoritmy, Hejnové algoritmy, Inteligence hejna, Optimalizace s omezeními, Samo-organizující se algoritmus, Vraní vyhledávací algoritmus, Algoritmus vlků obecných, Sinus-Kosinus algoritmus, Chaos, Kvadratická interpolace, CEC 2017, návrh parametrů tlakového válce, návrh konstrukce svařovaných nosníků, návrh parametrů pružiny

Abstract

The thesis is focused on the modern swarm algorithms and constrained optimization. The goal of this thesis is to improve the original version of the Self-organizing migrating algorithm (SOMA). The gradual development leading to finding a suitable solution is described in this thesis. Every partial solution is commented and compared with the original SOMA strategies. The novelty of the developed algorithm lies in the principle of migration of individuals - the individuals move in the direction of two adaptively shifting target locations. The method of perturbation was modified too. In the present algorithm, the generators of chaotic numbers are utilized. The novel algorithm described in this thesis was compared with selected state-of-art algorithms. As the testing problems Pressure vessel design problem, Welded beam design problem, Tension/-Compression Spring design problem and functions from the CEC 2017 benchmark were selected. Based on the experimental results, we can conclude that the innovations implemented to the original algorithm of SOMA significantly improved its performance.

Key Words: Evolutionary algorithms, Swarm Algorithms, Swarm Intelligence, Constrained optimization, Self-organizing migrating algorithm, Crow Search algorithm, Grey Wolf Optimizer algorithm, A Sine-Cosine Algorithm, Chaos, Quadratic Interpolation, CEC 2017, Pressure vessel design problem, Welded beam design problem, Tension/Compression Spring design problem

Obsah

Seznam použitých zkratek a symbolů

ABC	–	Algoritmus umělé kolonie včel
ACO	–	Optimalizace mravenčí kolonií
BOA	–	Motýlí algoritmus
CEC	–	Kongres evolučního výpočtu
CSA	–	Vraní vyhledávací algoritmus
DE	–	Diferenciální evoluce
DNA	–	Deoxyribonukleová kyselina
EA	–	Evoluční algoritmus
ES	–	Evoluční strategie
GA	–	Genetický algoritmus
GWO	–	Algoritmus vlků obecných
ICMIC	–	Iterativní chaotická mapa s neustálými kolizemi
MDE	–	Vícečlenná diferenciální evoluce
PSO	–	Algoritmus roje částic
QI	–	Kvadratická interpolace
RNG	–	Generátor náhodných čísel
SC	–	Algoritmus společenství a civilizace
SCA	–	Sinus-Kosinus algoritmus
SHADE	–	Diferenciální evoluce využívající historii úspěšnosti
SOGMA	–	Samo-organizující se genetický migrační algoritmus
SOMA	–	Samo-organizující se migrační algoritmus
SR	–	Podíl úspěšnosti
UDE	–	Sjednocená diferenciální evoluce
QBL	–	Algoritmus s učící se frontou

Seznam obrázků

Seznam tabulek

Seznam algoritmů

1 Úvod

Hledání globálních extrémů vícerozměrných prostorů, ať už hovoříme o matematických funkcích, či reálných problémech, je již řadu let předmětem intenzivního výzkumu [5]. Existuje mnoho problémů, pro které neexistuje exaktní algoritmus a přistupovat k jejich řešení analyticky je nevhodné, či nereálné. Důvodem může být multimodalita problému, tedy více lokálních optim, kladená omezení, různorodost číselných oborů a pod. Analýza problémů globální optimalizace ukazuje, že neexistuje deterministický algoritmus řešící obecnou úlohu globální optimalizace v polynomiálním čase, tento problém je tedy NP-obtížný [4]. Jedno z možných řešení je bezesporu využití stochastických metod, například hejnových algoritmů napodobujících chování skupin organizmů v přírodě. Těch existuje nepřeberné množství, některé jsou inspirovány chováním vran [11], jiné lovením vlků [20], ostatní zase mohou čerpat z chování hmyzích rojů [15]. Kvalita jednotlivých algoritmů je porovnávána na základě kvality nalezených řešení funkcí a dá se říci, že neexistuje, nebo se alespoň zatím nevynalezl, ideální a univerzální algoritmus pro všechny typy problémů [8].

Cílem této práce je představení a zlepšení hejnového algoritmu SOMA na optimalizačních problémech s omezeními. Proto v první části blíže seznámím čtenáře s definicí optimalizačního problému s omezeními a zaměřím se na metody práce s nimi. Představím evoluční algoritmy, ukážu jak fungují a kde se dají použít. Popíši konkrétní problémy: návrh parametrů tlakového válce, návrh parametrů pružiny, konstrukce svařovaných nosníků a funkce benchmarku CEC 2017. Vysvětlím, jak jsem postupoval při zlepšování SOMA a nakonec pomocí tabulek a grafů představím své dosažené výsledky.

2 Definice optimalizačního problému s omezeními

Mnoho problému z reálného života na poli vědy, obchodu, inženýrství aj., mohou být modelovány jako nelineární optimalizační problémy. Lze je tedy převézt na matematickou úlohu danou vhodným funkčním předpisem. Funkce, určena k optimalizaci s cílem nalezení nejvhodnějších hodnot argumentů se nazývá účelová funkce, označována $f(\vec{x})$, případně $f_{cost}(\vec{x})$. Snahou je pak nalézt řešení problému daného účelovou funkcí s co nejvyšším, či nejnižším ohodnocením [5]. Při zachycování reálného problému matematickou funkcí se přirozeně setkáváme také s omezeními: není například možné odebrat produktu tolik materiálu, aby byl nepoužitelný, nevyhovoval standardům, či porušoval fyzikální zákony. Funkce tedy musí splňovat určitá pravidla [55]. Obecně je můžeme rozdělit na rovnostní (Equality) a nerovnostní (Inequality) omezení. Obecná účelová funkce s omezeními pro minimalizační úlohu lze popsat rovnicí 1.

$$
\begin{aligned}
&\text{Minimalizujte:} \\
&\qquad f(\vec{x}), \qquad \vec{x} = (x_1, x_2, ..., x_n), \\
&\text{S omezeními:} \\
&\qquad g_i(\vec{x}) \leq 0 \qquad i = 1, ..., p \\
&\qquad h_j(\vec{x}) = 0 \qquad j = p+1, ..., m \\
&\qquad a_k \leq x_k \leq b_k \qquad k = 1, 2..., n
\end{aligned}
\tag{1}
$$

kde f je účelová funkce, $g_i(\vec{x}) \leq 0$ jsou nerovnostní omezení, $h_j(\vec{x}) = 0$ jsou omezení rovnostní a $a_k \leq x_k \leq b_k$ označuje hranice parametrů funkce. Řešení problému $\vec{x}$ je označeno za vyhovující, jestliže splňuje veškerá omezení.

Nejčastěji jsou optimalizovány úlohy minimalizační, tedy snažíme se nalézt řešení s co nejnižším ohodnocením účelové funkce. Mezi tyto problémy patří veškeré problémy popsané v této práci. Převod mezi hledáním maxima na hledání minima a obráceně lze provést vynásobením účelové funkce hodnotou -1 [5].

3 Metody pro práci s nevyhovujícími řešeními

Během průběhu optimalizačního algoritmu mohou nabývat parametry jedince hodnot, jež porušují některá omezení. Jestliže jsou překročeny hranice funkce, mohou být tyto situace řešeny dvěma způsoby: Intuitivním a jednoduchým způsobem může být umístění jedince na náhodné místo do prostoru. Tento způsob také zvyšuje diverzitu populace. Druhým je "zastavení"jedince na hranici, což ovšem může za určitých okolností vést k vytváření shluků na hranici a k nízké diverzibilitě populace [5]. Kromě překročení hranic mohou jedinci také nabývat hodnot porušující rovnostní a nerovnostní omezení dané účelové funkce. V těchto případech se setkáváme s následujícími přístupy:

1. Odmítnutí nevyhovujících řešení
   Jedná se o jednoduchý, avšak ne příliš účinný způsob počítající pouze s vyhovujícími řešeními. Je-li během výpočtu nalezeno řešení porušující omezení, potom není akceptováno. Jedná se tedy o tzv. hard-constraints. Zřejmým problémem této metody je neschopnost výpočtu v případě, kdy je akceptovatelná oblast velice malá a není jednoduché nalézt vhodná řešení [2]. S tímto přístupem se setkáváme zejména u algoritmů, jež nejsou primárně určeny pro optimalizaci problémů s malými oblastmi splnitelných řešení, například u základní verze Vraního algoritmu - CSA [11].

2. Použití penalizační funkce
   Jedná se o nejobvyklejší způsob řešení problému s omezeními pomocí evolučních algoritmů, jež byl představen už ve čtyřicátých letech [55]. Myšlenkou je znevýhodnění řešení jež porušují omezení účelové funkce dle určitého předpisu. Nabízí se tak možnost převést nespojitou hyperplochu problému s omezeními na hyperplochu spojitou, kde jsou daná omezení reflektována jako lokálními deformace směrem k opačnému extrému, než který byl hledán. Tento způsob patří do tzv. soft-constraints [5].
   Penalizační funkce mohou být vnější a vnitřní: Vnější metody ohodnocují řešení porušující omezení účelové funkce. Vnitřní metody naopak zabraňují řešením v proveditelné oblasti přístup do neproveditelné tím, že znevýhodňují řešení blížící se k neproveditelné oblasti. Vnitřní přístupy nejsou v praxi využívány, protože nedovedou optimálně prohledat okraje proveditelných oblastí a navíc pracují pouze s proveditelnými řešeními, které však není vždy možné nalézt [56].
   Penalizační funkce, jejíž výpočet je v průběhu algoritmů neměnný se označují jako statické, ty, jež využívají znalost aktuální fáze výpočtu (počet generací, ohodnocení účelové funkce...) jsou potom označovány za dynamické.

3.1 Vybrané statické penalizační funkce

- Rovnice 2 popisuje základní statickou vnější penalizační funkci. S jejím využitím se setkáváme i u moderních algoritmů, například u algoritmu Sjednocené diferenciální evoluce - UDE [80], jež byl představen v roce 2017.

$$F(\vec{x}) = f(\vec{x}) \pm \sum_{m=1}^{M} [r_m * H_m(\vec{x})] + \sum_{k=1}^{K} [c_k * G_k(\vec{x})] \tag{2}$$

  kde $F(\vec{x})$ je upravená hodnota účelové funkce, jež se má optimalizovat, H_m a G_k jsou funkce omezení dané problémem, r_m a c_k jsou pozitivní konstanty.

- V roce 1996 publikovali Hoffemester a Sprave [74] statickou penalizační funkci, popsanou rovnicí 3, jež je díky absenci nutnosti nastavování parametrů velice obecná a ukázala jako účinná po řešení inženýrských optimalizačních problémů.

$$F(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{m} \left[G_i(g_i(\vec{x})) * g_i(\vec{x})^2 \right] \tag{3}$$

  kde G_i je Heavisidův operátor dán: $G_i = 0$, když $g_i(\vec{x}) > 0$ a $G_z = 1$, když $g_i(\vec{x}) < 0$. Vidíme, že tato funkce není uzpůsobena pro práci s rovnostními omezeními, nicméně ty je možné převést na omezení nerovnostní dané formátem $|h(\vec{x})| - \epsilon \leq 0$, kde ϵ je hodnota blízká nule, například v benchamrku CEC 2017 je $\epsilon = 0.0001$ [22].

- O dva roky později (1998) představili Morales a Quezada [57] další statickou funkci, nevyužívající znalost vzdálenosti nevyhovujících jedinců od oblasti splnitelnosti, popsanou rovnicí 4.

$$F(\vec{x}) = \begin{cases} f(\vec{x}) & \text{jestliže je řešení splnitelné} \\ K - \sum_{i=1}^{s} K * m & \text{jinak} \end{cases} \tag{4}$$

  kde s je počet neporušených omezení, m je celkový počet omezení a K je vysoká konstanta, v práci [57] je použitá hodnota $1 * 10^9$. I přes absenci využití znalosti míry porušení ukázala tato metoda v jejich práci schopnost nalézat vhodná řešení, později však bylo ukázáno, že se jedná o metodu vhodnou pouze pro úzkou skupinu problémů [56].

- Jedna z dalších metod je popsána rovnicí 5 (Zelinka, Lampinen 1999 [5]).

$$F(\vec{x}) = (f(\vec{x}) + a) * \prod_{i=1}^{n} c_i^{b_i},$$

kde

$$c_i = \begin{cases} 1 + s_i * g_i(\vec{x}), \text{ jestli } g_i(\vec{x}) > 0 \\ 1 \text{ jinak} \end{cases} \tag{5}$$

$$s_i \geq 1,$$

$$b_i \geq 1$$

$$\min(f(\vec{x})) > 0$$

Konstanta a musí být nastavena dostatečně vysoko, aby funkce nabývala vždy kladných hodnot, tedy $\min(f(\vec{x})) + a > 0$.

- V roce 2000 publikoval Deb [10] neparametrickou penalizační funkci popsanou rovnicí 6. Jejím využitím docílíme vždy upřednostnění splnitelných řešení i plynulého přechodu mezi jednotlivými oblastmi. Během testování a porovnávání jednotlivých algoritmů [10] ukázal tento přístup schopnost optimálně nalézat splnitelná řešení i na samotných okrajích splnitelných oblastí.

$$F(\vec{x}) = \begin{cases} f(x) & \text{jestliže } \vec{x} \in S \\ f_w + \sum_{j-1}^{M} j_j & \text{jestliže } \vec{x} \notin S \end{cases} \tag{6}$$

kde f_w je hodnota účelové funkce nejhoršího vyhovujícího dosaženého řešení, j je míra porušení omezení a S je množina vyhovujících řešení.

3.2 Vybrané dynamické penalizační funkce

- Jonies a Houck [75] představili dynamickou techniku penalizace měnící se v čase výpočtu definovanou rovnicí 7.

$$F(\vec{x}) = f(\vec{x}) + (C * t)^{\alpha} * SVC(\beta, \vec{x}) \tag{7}$$

kde C, α a β jsou konstanty a $SVC(\beta, \vec{x})$ je definováno jako:

$$SVC(\beta, \vec{x}) = \sum_{i=1}^{n} D_i^{\beta}(\vec{x}) + \sum_{j=1}^{p} D_j(\vec{x}) \tag{8}$$

$$D_i(\vec{x}) = \begin{cases} 0 & \text{když } g_i(\vec{x}) \leq 0 \\ |g_i(\vec{x}) & \text{jinak} \end{cases} \tag{9}$$

$$D_j(\vec{x}) = \begin{cases} 0 & \text{když } -\epsilon \leq h_j(\vec{x}) \leq \epsilon \\ |h_j(\vec{x})| & \text{jinak} \end{cases} \tag{10}$$

kde ϵ je číslo blízké nule a t je hodnota zvyšující se v čase (aktuální počet generací).

- Zajímavou skupinou jsou penalizační funkce založeny na myšlence simulovaného žíhání [76], tedy rozkmitání míry znevýhodnění v čase, což může pomoci, jestliže algoritmus uvázne v lokálním extrému. Metoda Michalewicze a Attia [77] (1994) pracuje s rozdělenými omezeními do čtyř skupin: lineární rovnosti, lineární nerovnosti, nelineární rovnosti a nelineární nerovnosti. Výpočet vhodnosti je potom realizován dle rovnice 11.

$$F(\vec{x}) = f(\vec{x}) + 12 * \tau * \sum_{i \in A} \phi_i^2(\vec{x}) \tag{11}$$

kde A je množina aktivních omezení, τ reflektuje chladnutí při žíhání dle zadaných parametrů a ϕ je dáno rovnicí 12.

$$\phi(\vec{(x)}) = \begin{cases} max[0, g(\vec{(x)})] & \text{když } 1 \leq i \leq n \\ |h_i(\vec{x})| & \text{když } n+1 \leq i \leq m \end{cases} \tag{12}$$

kde n je počet nerovnostních omezení a m je celkový počet omezení.
Využití této metody ukázalo v jejich práci velice dobré výsledky při výpočtu inženýrských problémů, pouze však za předpokladu velice pečlivého zvolení parametrů řídících τ.

- Za zmínku také stojí tzv. adaptivní penalizace, které jsou řízeny vývojem jednotlivých algoritmů. Bean a Hadj-Aloune [78] vyvinuli v roce 1992 adaptivní penalizační funkci danou rovnicí 13. Modifikaci této metody lze nalézt i v moderních algoritmech, kupříkladu Adaptivní diferenciální evoluce s historií úspěšnosti s redukcí velikosti populace CAL-SHADE z roku 2017 [82], která rozšiřuje původní adaptivní penalizační funkci tím, že využívá data z n posledních iterací.

$$F(\vec{x}) = f(\vec{x}) + \lambda_t * \sum_{i=1}^{n} g_i(\vec{x})^2 + \sum_{j=1}^{p} |h_j(\vec{x})|,$$

kde

$$\lambda_{t+1} = \begin{cases} \lambda_t * (1/\beta_1) & \text{a)} \\ \lambda_t * \beta_2 & \text{b)} \\ \lambda_t & \text{c)} \end{cases} \tag{13}$$

kde β_1 a β_2 jsou konstanty a λ_{t+1} nabývá tří hodnot když: a) předchozí populace

obsahuje jen vhodná řešení, b) předchozí neobsahuje žádná vhodná řešení, c) všechny ostatní případy. Při správném nastavení vykázala tato metoda velice uspokojivých výsledků při řešení inženýrských problémů, nicméně výkonnost je silně závislá na parametrech β_1 a β_2 [78].

- Mezi moderní způsoby nakládání s omezeními patří Řešení omezení individuální penalizací (CHIP), jež uvedli Datta, Deb a Kim v roce 2019 [79], ve kterém je pro každé omezení individuálně vypočítáván parametr určující, jakou mírou bude ovlivňovat výslednou hodnotu. Jednotlivé parametry jsou normalizovány, což dle testování vedlo k vyšší robustnosti algoritmů. Problém s omezeními je zde také rozdělen jako současná optimalizace dvou funkcí, popsané rovnicí 14, jedna popisuje účelovou funkcí a druhá omezení.

$$\text{minimalizujte } f_1(\vec{x}) = CV(\vec{x}) \tag{14}$$

$$\text{minimalizujte } f_2(\vec{x}) = f(\vec{x}) \tag{15}$$

$$\texttt{kde} \quad CV(\vec{x}) = \sum_{j=1}^{J} R < g_j(\vec{x}) > + \sum_{k=1}^{K} r_k |h_k(\vec{x})| \tag{16}$$

  kde J a K jsou množiny omezení, a R a r jsou parametry.

Coello [56] ve svém přehledu řešení omezení účelových funkcí doporučuje zvolit penalizaci tak, aby byly řešení v nesplnitelné oblasti mírně horší, než ty ve splnitelné. Příliš vysoké znevýhodnění může jedince posunout daleko do splnitelné oblasti, čímž nedojde k prozkoumání hranic obou oblastí. Může tím také dojít k neschopnosti EA prozkoumat více nespojitých splnitelných oblastí, jsou-li daleko od sebe. Tímto se dostáváme k problému vhodné volby penalizační funkce a hlavně k nutnosti vhodně zvolit její parametry. Neexistuje univerzální postup jejich nastavení, které navíc může být u jednotlivých problémů rozdílné. Z těchto důvodů se u velkého množství algoritmů setkáváme s využitím neparametrických penalizačních funkcí [32], [55], [20], [16].

3. Další metody
   Některé algoritmy využívají specifické způsoby pro práci s nevyhovujícími jedinci:

   - U Genetických algoritmů se setkáváme s upravenou selekcí pro práci s omezeními, například C-SOGMA (Deb 2007 [55]), jež bude dále popsán v této práci (v Sekci 6.1), využívá kromě penalizace při výpočtu také upravené turnajové selekce mezi dvěma jedinci, která slouží k výběru vhodných jedinců při křížení:

(a) Jestliže není ani jeden jedinec ve splnitelné oblasti, je vybrán ten, jež porušuje omezení méně. Ohodnocení takovýchto jedinců pak probíhá podle rovnice 17.

$$F(\vec{x}) = \sum_{m=1}^{M} [h_m(\vec{x})]^2 + \sum_{k=1}^{K} [G_k(g_k(\vec{x}))]^2 \tag{17}$$

kde G_k je Heavisidův operátor dán: $G_k = 0$, když $g_k(\vec{x}) > 0$ a $G_k = 1$, když $g_k(\vec{x}) < 0$.

(b) Jestliže je pouze jeden jedinec vyhovující, je právě tento vybrán jako vhodný.

(c) V případě porovnání dvou vyhovujících jedinců, rozhoduje o vybraném hodnota účelové funkce.

Nespornou výhodou tohoto přístupu je absence parametrů nutných k nastavení.

- Výpočty některých algoritmů jsou zase rozděleny na dvě části: první hledá dostatečný počet vyhovujících řešení, jakmile jsou nalezeny začíná samotná optimalizace. Příkladem budiž L-SHADE + UDE (kombinace Adaptivního diferenciálního algoritmu s historií úspěšnosti s redukcí velikosti populace a Sjednocené diferenciální evoluce) z roku 2017 [81], jež v první části výpočtu hledá vhodná řešení ve splnitelné oblasti minimalizací průměru míry porušení omezení a jakmile jich nalezne dostatek (případně je proveden daný počet ohodnocení účelové funkce) postoupí algoritmus do optimalizační fáze.

4 Evoluční algoritmy

Evoluční algoritmy (EA) patří mezi optimalizační techniky vycházející z principů Darwinovy teorie přirozeného výběru. Obvykle se jedná o simulaci vývoje populace, kdy mají úspěšní jedinci vyšší pravděpodobnost přežít a přenést svůj genom do další generace. Tímto dochází k jejímu postupnému zlepšování [1].
Princip fungování EA, ať už se jedná o generování nové populace, či vytváření nových potomků, je založen na využití náhody. Na rozdíl od tradičních exaktních optimalizačních technik nezaručují EA nalezení konkrétního optimálního řešení. Lze je tedy řadit mezi pravděpodobnostní heuristické optimalizační techniky [4].

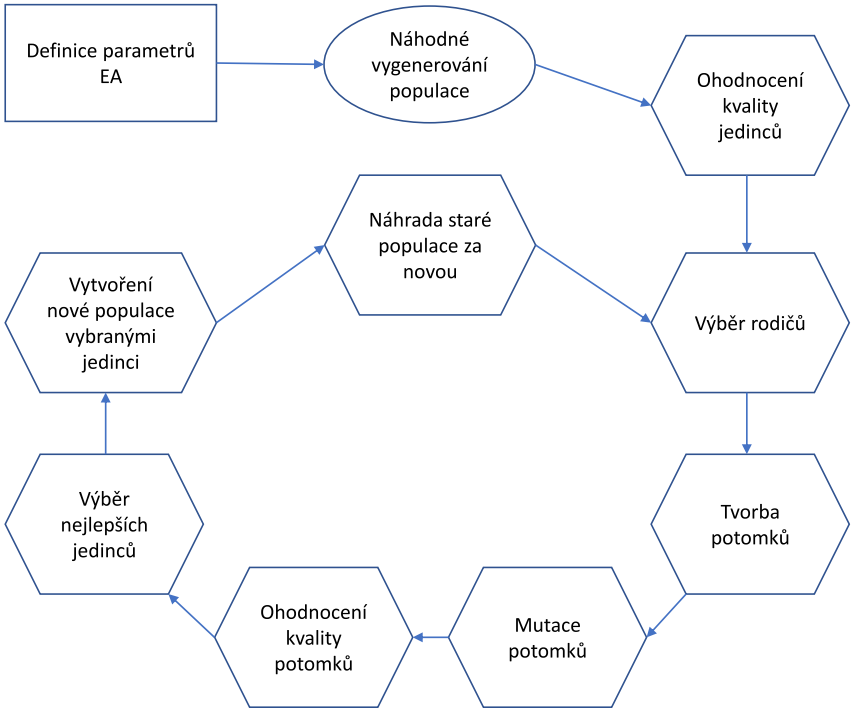
Jedinec	**Parametr** x	**Parametr** y	**Ohodnocení účelové funkce**
1	0,56	21,23	-474,80
2	1,80	55,60	-329,76
3	7,23	-15,20	-182,25
4	-5,12	18,62	-91,01
5	9,51	0,03	-63,52

Tabulka 1: Ukázka populace

Většina pojmů užívaných v EA vychází z Darwinovy teorie. Mezi základní termíny patří [2]:

- Jedinec - jedno řešení daného problému
- Populace - skupina jedinců, ukázka populace je znázorněna v Tabulce 1
- Ohodnocení kvality jedince - udává, jak vhodné jsou jeho parametry
- Účelová funkce - matematické vyjádření řešeného problému
- Křížení - operace mezi více jedinci v populaci (rodiči) jejíž výsledkem je jedinec nový (potomek)
- Mutace - náhodná změna v konfiguraci jedince, jedná se o napodobení biologické mutace genů
- Evoluční cyklus - obměna nevyhovujících jedinců v populaci využitím evolučních technik
- Generace - stav populace během jednoho evolučního cyklu

Princip funkce EA spočívá v cyklickém nahrazování, či úpravě nevyhovujících jedinců. Obecný evoluční cyklus, dle (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009) [5], uvádí Obrázek 1.



Obrázek 1: Obecný cyklus evolučního algoritmu (Zelinka, Oplatková, Šeda, Ošmera, Včelař, 2009) [5]

Před započetím migračního kola je nejprve nutné nastavit parametry. Každý EA musí mít jasně stanoveno za jakých podmínek se má ukončit, nutné je také nastavit účelovou funkci spolu s jejími případnými omezeními, velikost populace a další parametry dle zvoleného algoritmu. Následuje vygenerování nové populace, ta je obvykle generována náhodně. Jedinec je z pravidla reprezentován jako vektor o velikosti počtu dimenzí dané účelové funkce. Nová populace je poté ohodnocena, tedy každému jedinci je na základě jeho argumentů přiřazena hodnota představující jeho kvalitu. V této fázi výpočtu začíná zmiňovaný cyklus, který probíhá tak dlouho, dokud nejsou splněna ukončovací kritéria:

1. Výběr rodičů z populace dle jejich kvality
2. Vytvoření potomků křížením rodičů
3. Mutace potomků
4. Ohodnocení jedinců
5. Výběr se nejlepších jedinců
6. Vytvoření nové populace vybranými jedinci
7. Výměna staré populace za novou

Konkrétní způsob výběru rodičů, tvorby potomků, jejich mutace, či samotného způsobu reprezentace jedinců se mezi jednotlivými typy algoritmů liší. Mezi nejpočetnější skupiny patří:

- Genetické algoritmy (GA), ty jsou inspirovány stavbou DNA a dalšími evolučními procesy probíhající v biologických systémech. Každý jedinec bývá interpretován jako lineárně uspořádané geny, nazývané chromozóm, reprezentovaný jako bitový řetězec. Prohledávání prostoru je realizováno cyklickým prováděním migračního kola. Volba rodičů je u konkrétních algoritmů rozdílná, ukázalo se totiž, že zvolit vždy jedince s nejlepší vhodností může snížit diverzitu populace [5].

- Paralelně s genetickými algoritmy došlo ke vzniku tzv. evolučních strategií. Původně oba tyto druhy evolučních algoritmů sdílely podobný základ, lišily se pouze v reprezentaci jedinců - ES používá reálná čísla, GA binární a ES nepoužívaly operátory křížení. Hlavním principem vyhledávání ES je mutování vybraných jedinců vektorem náhodných čísel. Akceptace nových řešení je potom striktně deterministická.

- Komplexnější způsob tvoření potomků nalezneme u algoritmů diferenciální evoluce (DE), která je v porovnání s GA a ES výrazně mladší. Základ je podobný GA, je zde ovšem využíváno až pěti rodičů najednou, kdy potomek vzniká dle určitých vektorových operací. Je zde také zaměněno pořadí křížení a mutace.

- Hejnové algoritmy - Početná část optimalizačních technik vycházející z kolektivní inteligence, jež popisuje skupinové chování decentralizovaného a samo se organizujícího systému. Inspirací jsou skupiny živočichů v přírodě, které jsou bez jakéhokoliv řízení společně schopny efektivně řešit komplexní problémy [6]. Základem jsou jednoduchá pravidla definující interakci jedinců s okolím [15].
  Princip výpočtu je téměř totožný jako u EA, nicméně získávání nových řešení není realizováno mutací a křížením, ale kooperativním prohledáváním prostoru, ten je obvykle realizován jako posun jedinců na ploše [5]. V této práci budou konkrétní hejnové algoritmy blíže představeny, zejména algoritmus SOMA, který je popsán v Sekci 5 .

5 SOMA

Samoorganizující se migrační algoritmus (SOMA) byl vyvinut roku 1999. K optimalizaci využívá podobně jako algoritmus roje částic operace s vektory. Z technického hlediska lze SOMA řadit mezi algoritmy evoluční, kdy se během každého cyklu pracuje s populací jedinců. Na rozdíl od křížení, či mutace je zde změna realizována geometrickým posunem na hyperploše. Přirozeně se tedy řadí k algoritmům hejnovým [5].
Při vývoji byl algoritmus inspirován rojovou inteligencí v přírodě. Snahou je napodobit chování skupiny inteligentních jedinců, jež spolupracují na řešení daného problému. Tím může být například hledání potravy v přírodě [3]. Od svého vytvoření byl algoritmus mnohokrát upraven a vylepšen (více v Sekci 6).
Stejně jako u ostatních evolučních algoritmů je i SOMA závislá na řídicích a ukončovacích parametrech uvedených v tabulce 2.

Parametr	**Doporučený interval**	**Typ parametru**
PathLength	[1,1;3>]	Řídící
Step	[0,11; PathLength]	Řídící
PRT	[0; 1]	Řídící
D	Různý dle problému	Počet argumentů účelové funkce
PopSize	[10; definován uživatelem]	Řídící
Migrace	[10; definován uživatelem]	Ukončovací
MinDiv	[volitelný, definován uživatelem]	Ukončovací

Tabulka 2: Parametry algoritmu SOMA

Význam parametrů:

- *PathLength* (délka cesty) - určuje jak daleko bude aktivní jedinec prohledávat prostor v závislosti na vzdálenosti k vedoucímu jedinci.
- *Step* (krok) - určuje na kolik částí bude rozdělen *PathLength*, tedy jak jemně bude prostor prohledán
- *PRT* - řídící parametr perturbačního vektoru, ten ovlivňuje, zda se bude aktivní jedinec pohybovat přímo k vedoucímu jedinci. Více v Sekci 5.1.
- *D* - počet dimenzí (argumentů) účelové funkce
- *PopSize* - počet jedinců v populaci
- *Migrace* - maximální počet migrací - provedení hlavního cyklu
- *MinDiv* - ukončovací parametr definující maximální povolený rozdíl mezi nejlepším a nejhorším jedincem v populaci

5.1 Perturbace

Mutace je v SOMA nahrazena tzv. perturbací. Při pohybu jedinců prostorem je pohyb náhodně rušen - perturbován. Intenzita tohoto rušení je dána nastavením PRT parametru na jehož základě je v každém migračním kole pro každého jedince generován perturbáční vektror - $PRTVector$. Tvorba vektoru se řídí rovnicí 18.

$$PRTVector_j = \begin{cases} 1 & jestliže \quad rnd_j < PRT \\ 0 & jestliže \quad rnd_j \geq PRT \end{cases} \tag{18}$$

kde rnd_j je náhodná hodnota v intervalu $[0, 1]$ a PRT je parametr.

5.2 Posun jedinců na ploše

Při pohybu po ploše jedinci procházejí prostor po diskrétních skocích dle rovnice 19. Při tomto pohybu si vždy jedinec pamatuje nejlepší nalezené řešení v rámci své cesty, to je po skončení jeho posunu použito do dalších migračních kol [5].

$$x_{i,j}^{ML+1} = x_{i,j,start}^{ML} + (x_{L,j}^{ML} - x_{i,j,start}^{ML}) * t * PRTVector, \ \ kde\ t \in [0, PathLength] \tag{19}$$

5.3 Strategie

Jedinci v SOMA se mohou ovlivňovat na základě různých pravidel, od těch se odvíjí strategie:

- AllToOne
  Všichni k jednomu je základní strategií ve které dochází během migračního kola k posunu všech jedinců k nejlepšímu z populace. Algoritmus je popsán pseudokódem 1.

Algoritmus 1: Pseudokód SOMA AllToOne

Input:
P: náhodně vygenerovaná populace
řídící a ukončovací parametry `// viz tabluka 5`

```
f_cost: účelová funkce
ohodnoť P
while nejsou splněny ukončovací kritéria do
    leader = vyber nejlepšího z P
    for j <= PopSize do
        i = P[j]
        P[j] = proveď migrační krok i směrem k leader // viz rovnice 19
    end
    return nejlepší jedinec z P
end
```

- AllToAll
  Během migračního kola migrují všichni jedinci ke všem, neexistuje tedy leader. Změny v pozicích jedinců jsou provedeny až na konci migračního kola celé populace. Aplikace této strategie je výpočetně náročnější - během jednoho migračního kola dochází k násobně vyššímu ohodnocení účelové funkce. Na druhou stranu však během něj dojde k prohledání větší oblasti prostoru možných řešení.

- AllToAllAdaptive
  Jedná se o malou obměnu strategie AllToAll s tím rozdílem, že změny v pozicích jedinců se promítají ihned, nikoliv až na konci migrace celé populace. Tímto způsobem můžeme docílit rychlejší konvergace k optimu, nicméně dojde ke zmenšení prohledávané oblasti.

- AllToOneRand
  Jedinci zde migrují k jednomu řešení, jedná se tedy o obměnu základní strategie AllTo-One. Ta spočívá ve výběru jedince k následování, zatímco zmíněná strategie volí jedince s nejlepším ohodnocením, strategie AllToOneRand volí náhodného jedince z populace.

6 Publikované úpravy SOMA

Od představení SOMA bylo publikováno množství jeho úprav a zlepšení. Některé z nich jsou postaveny na kombinaci SOMA s jinými optimalizačními algoritmy, další zase aplikují inovativní způsoby křížení, či perturbace. Tato kapitola bude prezentovat vybrané úpravy, popíše jak fungují a jakých výsledků bylo jejich aplikací dosaženo.

6.1 SOMGA

Hybridní algortimus založen na kombinaci genetických algoritmů (GA) a SOMA byl představen roku 2007 ve snaze vytěžit z benefitů, jež tyto algoritmy nabízejí. Selekce, crossover a mutace je převzatá z binárních GA, zatímco organizace malé populace a její migrace ze SOMA, algoritmus je popsán v pseudokódu 2.

Algoritmus 2: Pseudokód SOMGA

Input:
P: náhodně vygenerovaná populace
f_{cost}: účelová funkce
P_c: pravděpodobnost křížení
P_m: pravděpodobnost mutace
řídící a ukončovací parametry `// viz tabluka 5`

```
1  ohodnoť P
2  while nejsou splněny ukončovací kritéria do
3      for i= 1 to velikost(P) do
4          rand_chromosomes = vyber 2 náhodné jedince (M)
5          i = proveď křížení(rand_chromosomes, P_c)
6          i = proveď mutaci(i, P_m)
7          ohodnoť(i)
8          P_new += i
9      end
10     P = vyber nejlepší jedince z P_new a P turnajovou selekcí
11     leader = Nejlepší jedinec z P
12     active = Nejhorší jedinec z P
13     P(active) = proveď migrační krok active směrem k leader // viz rovnice 19
14 end
15 return nejlepší jedinec z p
```

SOMGA byla v literatuře [54] porovnána s GA a SOMA na 25 problémech bez omezení mající 10 dimenzí, maximální počet ohodnocení účelové funkce byl nastaven na 40 000. Ve srovnání 30 nezávislých běhů algoritmů vykázala úprava výrazně vyšší spolehlivost, efektivitu i přesnost:

- Žebříček hodnocení spolehlivosti, jež udává počet úspěšných běhů algoritmů na testovaných problémech je GA < SOMA < SOMGA. SOGMA indikoval 195% nárůst celkové spolehlivosti v porovnání s GA a 25% v porovnání se SOMA, GA nevykázal nejlepší spo-

lehlivost na žádném z 25 problémů, SOMA na 16 a SOMGA na 22. 6 problémů pak dalo porovnatelné výsledky algoritmů.

- Porovnání efektivity, tedy nutného počtu evaluací funkce, aby mohl být běh považován za úspěšný, je potom GA < SOMA < SOMGA s 67% zlepšením SOGMA v porovnání s GA a 47% se SOMA. GA mělo nejlepší efektivitu ve 3 problémech, SOMA rovněž ve 3 a SOMGA v 19.

- Přesnost, značící jak blízko je řešení nalezené algoritmem ke globálnímu optimu, byla nejlepší v 6 problémech u GA, SOMA nalezla nejoptimálnější řešení ve 4 problémech a SOMGA v 15. Stejně tak směrodatná odchylka byla vždy nejnižší u SOMGA.

Jako úspěšný běh byl v tomto případě označen ten, jež nalezne řešení v maximální vzdálenosti 1% od známého globálního optima.
SOMGA byl o rok později upraven k řešení problémů s omezeními pod zkratkou C-SOMGA. K práci s nevyhovujícími jedinci je upravena turnajová selekce spolu s penalizační funkcí, postup byl popsán v sekci 3.
Algoritmus byl v literatuře [55] porovnán na 10 problémech s omezeními s algoritmy C-SOMA a C-GA. C-SOMA je verze klasického algoritmu SOMA, užívající k evaluaci stejné neparametrické penalizační funkce. Stejně tak i C-GA vzniklo úpravou GA. Výsledky získané 100 nezávislými běhy na 10 testovacích funkcích s omezeními algoritmů indikují značně vyšší robustnost ve smyslu spolehlivosti, efektivity i přesnosti C-SOMGA v porovnání s oběma algoritmy.

6.2 SOMAQI

Jedná se o hybridní variantu SOMA, publikovanou v roce 2014, rozšířenou o operátor míšení jedinců pomocí kvadratické interpolace. Díky tomu je schopná rychle najít globální extrém s malou velikostí populace na velkém spektru funkcí [12]. Právě z těchto důvodů je SOMAQI primárně určena k optimalizaci velkých multidimenzionálních problémů, kde se počet dimenzí pohybuje v rozmezí od 100 do 3000. S velikostí populace pouze 10 jedinců dosahuje kvalitních optimálních řešení s malým počtem ohodnocení účelové funkce [12] Popis fungování zachycuje pseudokód 3.
Účinnost tohoto přístupu byla testována na 12 škálovatelných problémech bez omezení [12] o počtu dimenzí 100, 500, 1000, 2000 a 3000 s maximálním počtem ohodnocení účelové funkce $50 * počet\_dimenzí$. Pro zjištění účinnosti byly dosažené výsledky srovnány s dalšími evolučními algoritmy na 4 zvolených problémech, konkrétně se jednalo o algoritmus s učicí se frontou (QBL) [60], algoritmus kombinující simulované žíhání a Nedler-Mead algoritmus (SNMRUV) [61], algoritmus umělé kolonie včel (ABC), rozšířený o Lévyho let (L-ABC) [63], GA a PSO [15]. Výsledky dokázaly, že SOMAQI dosahuje lepších výsledků ve většině problémů [12].

Algoritmus 3: Pseudokód SOMAQI

Input:
P: náhodně vygenerovaná populace
f_{cost}: účelová funkce
řídící a ukončovací parametry `// viz tabluka 5`

```
1  ohodnoť(P)
2  while nejsou ukončeny ukončovací kritéria do
3  |  seřaď(P)
4  |  R_1 = nejlepší jedinec(P)
5  |  R_2 = náhodný jedinec(P)
6  |  R_3 = náhodný jedinec(P)
7  |  active = nejhorší jedinec(P)
8  |  active = proveď migrační krok active směrem k R_1 // viz rovnice 19
9  |  i = získej bod // viz rovnice 20
10 |  ohodnoť(i)
11 |  if hodnota účelové funkce i > hodnota účelové funkce active then
12 |  |  active = i
13 |  end
14 |  nejhorší jedinec(P) = active
15 end
16 return Nejlepší jedinec(P)
```

SOMAQI byla také upravena pro použití na funkcích s omezeními využitím neparametrické penalizační funkce, popsané rovnicí 6, označována jako C-SOMAQI [13]. K zjištění výkonnosti algoritmu bylo provedeno porovnání s C-SOMA (SOMA používající neparametrickou penalizační funkci 6) a C-SOMGA (více v sekci 6.1) na 10 problémech s omezeními. Bylo provedeno 100 nezávislých běhů. Srovnání průměrných výsledků algoritmů značí, že C-SOMAQI nalezl v 9 případech nejlepší řešení a v 1 případě to byl C-SOMGA. Žebříček hodnocení algoritmů je C-SOMA < C-SOMGA < C-SOMAQI [13].

6.2.1 Kvadratická interpolace

Metoda kvadratické interpolace aproximuje pozici minima proložením kvadratické paraboly třemi body v prostoru. Jedná se o velice účinnou metodu k vylepšení lokálního prohledávání algoritmů a zpřesnění nalezených optim [28]. K určení interpolované pozice je potřeba znát tři body včetně hodnot účelové funkce. Výpočet pozice je dán rovnicí 20. Přestože je metoda určena pro optimalizaci na funkcích bez omezení, lze ji použít i na funkcích s omezeními nahrazením hodnot účelové funkce hodnotou míry porušení omezení [28].

$$\vec{x} = \frac{1}{2}\frac{(R_2^2 - R_2^3) * f(R_1) + (R_3^2 - R_1^2) f(R_2) + (R_1^2 - R_2^2) * f(R_3)}{(R_2 - R_3) * f(R_1) + (R_3 - R_1) * f(R_2) + (R_1 - R_2) * f(R_3)} \tag{20}$$

Kde R_1 je nejlepší jedinec v populaci, a R_2 a R_3 jsou náhodně vybraní jedinci ze zbytku populace. $f(R_x)$ je hodnota účelové funkce jedince R_x.

6.3 GSOMA

Využití Gaussova rozdělení pravděpodobnosti (též známé jako normální rozdělení) [29], [30], [31] může přinést zpřesnění evolučních algoritmů, či jim dokonce pomoci neuváznout v lokálním extrému [33]. Právě tyto vlastnosti byly inspirací ke vzniku přístupu k SOMA - GSOMA, představeného v roce 2009 [34]. Modifikace algoritmu spočívá v úpravě rovnice pro posun jedinců na ploše, ten je uveden v rovnici 21.

$$x_{i,j}^{ML+1} = \begin{cases} x_{i,j,start}^{ML} + (x_{L,j}^{ML} - x_{i,j,start}^{ML}) * t * PRTVector & u_j < 0.5 \\ x_{i,j,start}^{ML} + G_{i,j}(x_{L,j}^{ML} - x_{i,j,start}^{ML}) * t * PRTVector & u_j \geq 0.5 \end{cases} \tag{21}$$

kde $t \in [0, PathLength]$, u_j je náhodná hodnota rovnoměrného rozdělení v intervalu $[0, 1]$ a $G_{i,j}$ je náhodné číslo Gaussova rozdělení v intervalu $[0, 1]$.
GSOMA byla v literatuře [34] použita pro návrh parametrů volnoběžky (Fortini 1967 [36]). Výsledky byly porovnány s algoritmy SOMA, GA [37] a PSO [15]. Bylo testováno 5 různých nastavení problému návrhu parametrů volnoběžky, při dvou z nich vykázala nejlepších výsledků GSOMA, stejně tak i PSO, ostatní algoritmy byly vhodné právě pro jedno nastavení (při jednom nastavení nalezl PSO a GA stejné parametry).

6.4 SOMA T3A

SOMA Team To Team Adaptive (SOMA T3A), představen na konferenci CEC 2019 [21], je, na rozdíl od původního SOMA, rozdělen na 3 části, jež se během migračního kola opakují:

1. Organizace - má za cíl nalézt vhodné jedince, zvané Migranti, k posunu na ploše. A také vhodného Leadera, k němuž se budou Migranti posouvat. Výběr migrantů probíhá náhodným zvolením m jedinců z populace a následným výběrem n nejlepších z nich. Zvolení Leadera je velice podobné: je zvoleno k náhodných jedinců, nejlepší z nich se stává Leaderem. Pokud je některý jedinec z Migrantů zároveň i Leader, nedochází u toho jedince během daného migračního kola k posunu na poše.

2. Migrace - provádí posun Migrantů k Leaderovi. Na rozdíl od SOMA zde počet skoků není omezen parametrem $PathLength$, ale maximálním počtem skoků N_{jumps}. PRT a Step parametry jsou dále adaptivně upravovány během každé migrace, jak je popsáno rovnicemi 22 a 23.

$$PRT = 0.05 + 0.90\frac{FEs}{MaxFEs} \tag{22}$$

$$Step = 0.15 - 0.08\frac{FEs}{MaxFEs} \tag{23}$$

kde FEs je aktuální počet evaluací účelové funkce, a $MaxFEs$ je maximální počet ohodnocení účelové funkce.

3. Aktualizace - nahrazení původních pozic jedinců nově získanými, pakliže je jejich ohodnocení vhodnější.

Algoritmus byl v literatuře [14] testován na 58 benchmarkových funkcích bez omezení převzatých z CEC 2013 a CEC 2017. Výsledky byly nejprve porovnány s algoritmy SOMA AllToOne, SOMA AllToAll, a poté s ABC [63], algoritmem kombinujícím algoritmus světlušek a roj částic (HFPSO) [64], hejnovým algoritmem SSA [65] a SAMPEJaya [66]. Počet dimenzí všech funkcí byl nastaven na 30 a maximální počet evaluací účelové funkce $MaxFEs$ na 300000. Ve srovnání SOMA T3A s ostatními verzemi SOMA bylo pozorováno zlepšení jak na unimodálních, tak i multimodálních funkcích. Při analýze výsledků s ostatními state-of-art algoritmy bylo pozorováno výrazného zlepšení SOMA T3A ve více, než polovině z 28 funkcí bez omezení z benchmarku CEC 2017 (14 lepších než ABC, 15 než HFPSO, 17 než SSA a 23 než SAMPEJaya).

7 Vybrané hejnové algoritmy

K bližšímu představení byly vybrány algoritmy publikované v posledních letech, které jsou určeny k řešení problému s omezeními. Tyto algoritmy byly také naimplementovány a jejich výsledky jsou v praktické části v sekci 11 srovnávány s vlastním vylepšením.

7.1 Crow Search algoritmus

Vraní vyhledávací algoritmus (CSA), představen v roce 2015, je hejnový evoluční algoritmus inspirován inteligentním chováním vran ve volné přírodě. Inspirací je zejména vlastnost vran schovávat si nadbytečnou potravu do skrýší, jejíž umístění si jsou schopné přesně zapamatovat. Vrány často pozorují ostatní jedince, kteří schovávají své jídlo, aby jim mohly jejich zásoby v nepozorovaný moment ukrást. Vrány si jsou vědomy toho, že je jiná může sledovat, proto pokud zjistí, že je při schovávání potravy nějaká sledovala, změní umístění své skrýše.[11]
Stejně jako chování vran žijící ve volné přírodě i CSA reflektuje následující vlastnosti:

- Vrána (jedinec) žije v hejnu (populaci)
- Vrána si dokáže zapamatovat místo, kam schovala své jídlo
- Vrána zkoumá ostatní při vytváření skrýší a pokouší se je vykrást
- Vrána chrání svou skrýš před vykradením

Hlavní cyklus CSA reflektuje obecný evoluční algoritmus, přesněji je popsán v pseudokódu 4. Každá vrána má svou paměť, ve které je uložena pozice s nejlepší hodnotou, kterou vrána nalezla, tímto se CSA odlišuje od většiny EA. Vrány se pohybují po ploše a hledají místa s nejlepším zdrojem jídla - nejlepší skrýše, tyto nalezené hodnoty si zapamatují - v paměti vran je tedy vždy uložena nejlepší nalezená pozice.
Posun vran v prostoru se provádí následovně: Představme si, že vrána i zná umístění skrýše vrány j a rozhodne se k ní vydat. Potom může dojít ke dvěma situacím:

1. Vrána j neví, že ji vrána i sleduje. Pozice vrány i je v tomto případě počítána podle rovnice 24:

$$x^{i,iter+1} = x^{i,iter} + r_i * fl^{i,iter} * (m^{j,iter} - x^{i,iter}) \tag{24}$$

   Kde r_i je náhodná hodnota v intervalu <0, 1> rovnoměrného rozdělení a $fl^{i,iter}$ udává délku letu vrány i v dané iteraci. $m^{j,iter}$ označuje pozici skrýše sledované vrány.

2. Vrána j ví, že ji vrána i následuje. Proto ve snaze ochránit svou skrýš zmate sledující vránu i tím,že se vydá na jiné náhodné místo v prostoru.

Výkonnost CSA byla v literatuře [11] demonstrována na šesti inženýrských problémech s omezeními, kde byl algoritmus srovnáván, mimo jiné, s algoritmem společenství a civilizace (SC) [42],

Algoritmus 4: Pseudokód CSA

Input:
x: prvopočáteční náhodně vygenerovaná populace N vran v prostoru
řídící a ukončovací parametry 5
f_{cost}: účelová funkce vracející vhodnost aktuálního řešení

```
Proveď evaluaci pozici všech vran
Inicializuj paměť vran
while Nejsou splněny ukončovací kritéria do
    for j <= Počet jedinců v populaci; j++ do
        l = náhodná vrána z populace
        r = náhodné číslo [0, 1]
        if r < AP then
            posuň vránu j směrem k paměti vrány l dle rovnice 24
        else
            posuň vránu j na náhodné místo v prostoru
        end
        Zkontroluj přístupnost nových souřadnic
        Vypočti hodnoty účelových funkcí všech vran
        Aktualizuj paměť vran
    end
end
v = vrána, která navštívila nejlepší místo, má tedy nejlepší uložené místo v paměti
return v
```

PSO-DE [9], vícečlennou diferenciální evolucí s dynamicko-stochastickým výběrem (DSS-MDE) [43], algoritmem simulujícím výbuch nálože (MBA) [44], GA3 [45], GA4 [46], CPSO [50], HPSO [51], učícím se algoritmem (TLBO) [52] a ABC [63]. Výkon CSA se ukázal jako lepší než SC, MBA, GA3, GA4, CPSO a ABC. Stejné výsledky byly pozorovány s algoritmy PSO-DE a DDS-MDE. Mírně lepší potom poskytly algoritmy HPSO a TLBO. Na výsledcích byly také znatelné nízké hodnoty směrodatné odchylky, což indikuje vysokou robustnost CSA.

7.2 Grey Wolf Optimizer algoritmus

Hierarchie a chování smečky při lovu vlků se stala inspirací pro hejnový vyhledávací algoritmus vlků obecných (GWO), jenž byl představen v roce 2013. Vlci žijí ve smečce o průměrné velikosti 5-12 členů, zde panuje hierarchické rozdělení: vůdcové smečky, označování α jsou zodpovědní za provádění rozhodnutí při lovu, určení místa pro přespání a podobně. Pod nimi stojí β, potenciální kandidáti pro nahrazení alfy. Bety často pomáhají alfám v úkonech a dohlíží na dodržování jejich pokynů. Na třetím stupni jsou γ a na posledním ω. Omegy se musí podřídit všem dominantním jedincům, jsou ti poslední jimž je dovoleno jíst, v některých případech jsou omegy určeny ke starání se o výchovu potomků. Vlci nepatřící mezi α, β, ani ω jsou součástí hierarchického stupně γ. Algoritmus reflektuje toto rozdělení a simuluje lov kořisti, kdy vlci následují své dominantní jedince ve snaze nalézt a ulovit kořist [20].

7.2.1 Matematické vyjádření smečky

Rozdělení sociální hierarchie je zachyceno následovně:

- Jako α je označen jedinec s nejlepším ohodnocením
- Druhý nejlepší jedinec je označen β
- Jedinec s třetím nejlepším ohodnocením je potom γ
- Všichni ostatní jedinci jsou ω

Vlci při lovu svou kořist obkličují, matematicky je tato činnost popsána ve rovnicích 25 a 26. Pseudokód 5 popisuje obecný postup GWO.

$$\vec{D} = |\vec{C} * \vec{X}_p(t) - \vec{X}(t)| \tag{25}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} * \vec{D} \tag{26}$$

kde t indikuje současnou iteraci, $\vec{A}$ a $\vec{C}$ jsou koeficienty, $\vec{X}_p$ je pozice vektoru kořisti a $\vec{X}$ je pozice daného vlka.
Vektory $\vec{A}$ a $\vec{C}$ jsou vypočteny následovně:

$$\vec{A} = 2\vec{a} * \vec{r}_1 - \vec{a} \tag{27}$$

$$\vec{C} = 2 * \vec{r}_2 \tag{28}$$

kde $\vec{a}$ je hodnota lineárně klesající od 2 do 0 v průběhu iterací a r_1, r_2 jsou náhodné vektory v rozmezí $[0, 1]$. Při hledání optima - lovení kořisti očekáváme, že vlci α, β a ω znají nejlépe její pozici, proto jsou určující při obklíčení. Vzdálenosti vlka ($\vec{X}$) od α je označena jako D_α, od β jako D_β a od γ potom D_γ a jejich výpočet je uveden v rovnici 29. Efekt dominantních vlků X_1, X_2, X_3 lze spočítat rovnicí 30

$$\vec{D_\alpha} = |\vec{C_1} * \vec{X_\alpha} - \vec{X}|, \vec{D_\beta} = |\vec{C_2} * \vec{X_\beta} - \vec{X}|, \vec{D_\gamma} = |\vec{C_3} * \vec{X_\gamma} - \vec{X}| \tag{29}$$

$$\vec{X_1} = \vec{X_\alpha} - \vec{A_1} * \vec{D_\alpha}, \vec{X_2} = \vec{X_\beta} - \vec{A_1} * \vec{D_\beta}, \vec{X_3} = \vec{X_\gamma} - \vec{A_1} * \vec{D_\gamma} \tag{30}$$

$$\vec{X}(r+1) = \frac{\left(\vec{X_1} + \vec{X_2} + \vec{X_3}\right)}{3} \tag{31}$$

Účinnost tohoto přístupu byla v literatuře [20] porovnávána s algoritmy CPSO [50], GA3 [45],

Algoritmus 5: Pseudokód GWO

Input:
x: prvopočáteční náhodně vygenerovaná populace N vlků v prostoru
f_{cost}: účelová funkce vracející vhodnost aktuálního řešení

```
Vypočítej fitness všech jedinců v populaci
X_α = jedinec s nejlepším fitness
X_β = druhý nejlepší jedinec
X_γ = třetí nejlepší jedinec
Vypočti hodnoty a, A, C
while Nejsou splněny ukončovací kritéria do
    for každý jedinec v populaci do
        Aktualizuj polohu současného jedince dle rovnice 31
    end
    Aktualizuj a, A, C
    Vypočítej fitness všech jedinců v populaci
    Aktualizuj X_α, X_β, X_γ
end
return X_α
```

GA4 [46], DE [43] a ACO [53] na funkcích bez omezení benchmarku CEC 2005 a také na inženýrských problémech s omezeními. Kde byla pozorována vysoká výkonnost GWO, demonstrující schopnost úspěšně se vyhnout uváznutí v lokálnímu optimu a rychle konvergovat k optimu.

7.2.2 Chaos Grey Wolf algoritmus

S cílem zlepšit efektivitu byla v roce 2016 představena verze GWO rozšířená o chaos [16], popsaným v Sekci 9.5.1. Veškeré výpočetní mechanismy jsou identické s algoritmem GWO, jež je uvedeno v matematickém vyjádření smečky 7.2.1. Rozdílným je pouze získávání hodnoty a, jejíž upravená verze je popsána v rovnici 32.

$$\vec{a} = \vec{p} * x_k \tag{32}$$

Kde $\vec{p}$ je hodnota lineárně klesající od 2 do 0 v průběhu iterací a x_k je chaoticky získaná hodnota. Postup algoritmu je přiblížen v pseudokódu 6.
Výkonnost CGWO byla porovnána [16] s původní verzí na inženýrských problémech a ve všech případech ukázal zlepšení v nalezeném řešení.

7.2.3 Chaos

Sekvence náhodných čísel je v EA často využívaná, závisí na ní rozmístění prvotní populace, či rychlost a přesnost jakou se populace blíží k optimu [68]. K jejich generování bývá nejčastěji využíváno generátorů pseudonáhodných čísel (RNG). Další možností je, mimo jiné, využití chaotických sekvencí [69]. Deterministický chaos, objeven v roce 1963 E. Lorenzem, popisuje

Algoritmus 6: Pseudokód CGWO

Input:
x: prvopočáteční náhodně vygenerovaná populace N vlků v prostoru
f_{cost}: účelová funkce vracející vhodnost aktuálního řešení
chaotická mapa

```
Vypočítej fitness všech jedinců v populaci
X_α = jedinec s nejlepším fitness
X_β = druhý nejlepší jedinec
X_γ = třetí nejlepší jedinec
Vypočti hodnoty a, A, C a chaotické číslo x_0 užitím chaotické mapy
while Nejsou splněny ukončovací kritéria do
    for každý jedinec v populaci do
        Aktualizuj polohu současného jedince dle rovnice 31
        Aktualizuj chaotické číslo x_k užitím chaotické mapy
        Aktualizuj a dle rovnice 32
    end
    Aktualizuj A, C
    Vypočítej fitness všech jedinců v populaci
    Aktualizuj X_α, X_β, X_γ
end
return X_α
```

komplexní a nepředvídatelné chování nelineárního deterministického systému [26]. Důležitým znakem je citlivost na extrémně malé změny počátečních podmínek [38]. S jejich využitím se setkáváme v širokém spektru odvětví, ať už hovoříme o zabezpečeném přenosu dat, telekomunikaci, kryptografii, výpočtech DNA, zpracovávaní obrazu a mnoho dalších [40]. Aplikací chaotických posloupností namísto čísel získaných generátorem náhodných čísel je často vhodnou strategií také k zajištění diverzity populace a zlepšení konvergence k optimům vyhledávacích algoritmů [27]. Teorie chaosu byla v minulosti mnohokrát úspěšně kombinována s mnoha meta-heuristickými metodami [70], její využití přineslo zlepšení výkonu mnoha algoritmů, například GWO [16], PSO [71], Algoritmu světlušek (FA) [70], Motýlím algoritmu (BOA) [72], či GA [73].
Matematicky vyjádřeno je chaos náhodnost jednoduchého dynamického deterministického systému. Ke generování chaotických čísel se využívají chaotické mapy, ty při vhodném nastavení parametrů nabývají v každé iteraci unikátních hodnot. Tyto mapy lze popsat rovnicemi, kde a je kontrolní parametr, x_k je chaotické číslo v k-té iteraci [16]. Z velkého množství publikovaných chaotických map využívají optimalizační algoritmy často tyto:

- Bernoulliho mapa

$$x_{k+1} = \begin{cases} \frac{x_k}{1-a} & 0 < x_k < a \\ \frac{x_k-(1-a)}{a} & (1-a) < x_k < 1 \end{cases} \tag{33}$$

- Logistická mapa

$$x_{k+1} = a * x_k(1 - x_k) \tag{34}$$

- Čebyševovova mapa

$$x_{k+1} = \cos(a * \cos^{-1}(x_k)) \tag{35}$$

- Kruhová mapa

$$x_{k+1} = x_k + b - \left(\frac{a}{2 * \pi}\right) * \sin(2 * \pi * x_k) \mod (1) \tag{36}$$

- Iterativní chaotická mapa s neustálými kolizemi (ICMIC)

$$x_{k+1} = |\sin(a/x_k)| \tag{37}$$

- Sinusoidální mapa

$$x_{k+1} = a * x_k^2 * \sin(\pi * x_k) \tag{38}$$

- Stanová mapa

$$x_{k+1} = \begin{cases} \frac{x_k}{0.7} & x_k < 0.7 \\ \frac{10}{3*(1-x_k)} & x_k \geq 0.7 \end{cases} \tag{39}$$

7.3 A Sine Cosine algoritmus

Sinus-Kosinus algoritmus (SCA) využívá při vyhledávání v prostoru mezi dvěma jedinci jednoduchých matematických funkcí: sinus a kosinus [41]. Vyhledávání v prostoru se řídí rovnicemi 40 a 41. Popis postupu algoritmu je uveden v pseudokódu 7.

$$\vec{X}_i^{t+1} = \vec{X}_i^t + r_1 * sin(r_2) * |r_3\vec{P}_i^t - \vec{X}_i^t| \tag{40}$$

$$\vec{X}_i^{t+1} = \vec{X}_i^t + r_1 * cos(r_2) * |r_3\vec{P}_i^t - \vec{X}_i^t| \tag{41}$$

kde $\vec{X}_i^t$ je současná pozice jedince v i-té dimenzi a t-é iteraci. r_1, r_2, r_3 jsou náhodné čísla omezené parametry, P_i je pozice současného cílového řešení problému v i-té dimenzi.
V algoritmu je využívána kombinace rovnic 40 a 41 zobrazena v rovnici 42.

$$\vec{X}_i^{t+1} = \begin{cases} \vec{X}_i^t + r_1 * sin(r_2) * |r_3\vec{P}_i^t - \vec{X}_i^t| & r_4 > 0.5 \\ \vec{X}_i^t + r_1 * cos(r_2) * |r_3\vec{P}_i^t - \vec{X}_i^t| & r_4 \leq 0.5 \end{cases} \tag{42}$$

kde $r4$ je náhodné číslo v $[0, 1]$.

Ve výše uvedených rovnicích jsou uvedeny čtyři hlavní parametry SCA: r_1 určuje směr, kterým se bude v dalších iteracích algoritmus ubírat. r_2 definuje jak daleko se bude jedinec při migraci posouvat. Parametr r_3 přináší náhodnou váhu cílové pozice. A konečně parametr r_4 slouží jako jakýsi přepínač mezi funkcemi sinus a kosinus.

Algoritmus 7: Pseudokód SCA

Input:
x: prvopočáteční náhodně vygenerovaná populace N vlků v prostoru
f_{cost}: účelová funkce vracející vhodnost aktuálního řešení

```
while Nejsou splněny ukončovací kritéria do
    Vypočítej fitness všech jedinců v populaci
    P = jedinec s nejlepším fitness
    Aktualizuj r1, r2, r3, r4
    Posuň všechny jedince na ploše dle rovnice 42
end
P = jedinec s nejlepším fitness
return P
```

SCA byl při svém představení [41] porovnán na funkcích s i bez omezení s algoritmy PSO [15], GA, FPA, kde dosáhl velice nadějných výsledků. Nalezl také optimální konfiguraci při návrhu designu křídla letadla, který se řadí mezi inženýrské problémy s omezeními.

8 Testovací funkce

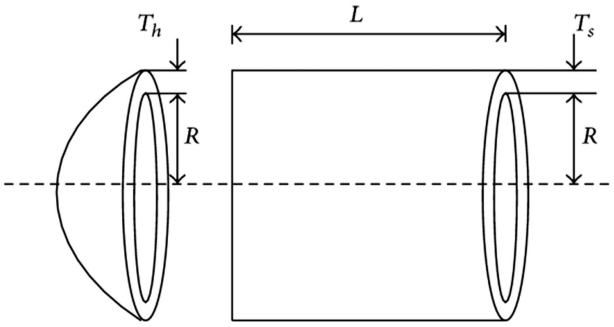
Optimalizační algoritmy jsou porovnávány na základě nalezených řešení na různorodých funkcích. Hovoříme-li se o funkcích s omezeními, nabízí se testovat algoritmy na reálných problémech [39]. Existují ovšem i specializované soubory funkcí tzv. benchmarky. V této práci byly implementovány 3 inženýrské problémy a 28 funkcí z benchmarku CEC 2017[22].

8.1 Funkce návrhu parametrů inženýrských konstrukčních problémů

Byly vybrány tři typické funkce reflektující problémy z inženýrské praxe, jež jsou často využívány při prezentaci nových algoritmů a v dalších srovnáních [20], [16], [13], [55]. Dosažené výsledky vlastních úprav tak lze snadno porovnávat s ostatními EA bez nutnosti jejich implementace.

8.1.1 Návrh parametrů tlakového válce

Navrhnout tlakový válec s co nejnižšími náklady na sváření, výrobu a materiál je jeden z inženýrských problémů [16]. Obrázek 2 zobrazuje čtyři parametry, jež je nutné optimalizovat: šířku stěny obalu (T_s), šířku stěny hlavy válce (T_h), vnitřní poloměr válce (R) a délku válcovitého obalu (L) [39]. Popis matematického vyjádření problémů, spolu s jeho omezeními je zachycen v rovnici 43.



Obrázek 2: Struktura parametrů konstrukce tlakového válce

$$\vec{x} = [x_1, x_2, x_3] = [T_s T_h R L],$$

Minimalizujte:

$$f(\vec{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3,$$

S omezeními:

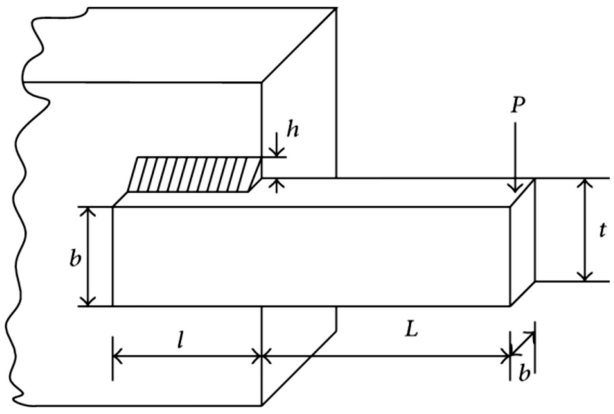
$$\begin{aligned} g_1(\vec{x}) &= -x_1 + 0.0193 x_3 < 0, \\ g_2(\vec{x}) &= -x_2 + 0.00954 x_3 < 0, \\ g_3(\vec{x}) &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 < 0, \\ g_4(\vec{x}) &= x_4 - 240 < 0, \end{aligned} \tag{43}$$

Rozsah parametrů:

$$\begin{aligned} 0 < x_1 < 100, \\ 0 < x_2 < 100, \\ 10 < x_3 < 200 \\ 10 < x_4 < 200 \end{aligned}$$

8.1.2 Návrh konstrukce svařovaných nosníků

Problém nalezení optimálních parametrů svařovaných nosníků s cílem minimalizovat množství užitého materiálu má čtyři proměnné: hrubost sváru (h), délka sváru (l), výška nosníku (t) a šířka nosníku (b) jako je vyobrazeno na Obrázku 3. Při návrhu je nutné dbát na množství omezení, konkrétně únosnost tlaku v ohybu (σ), míra vychýlení při námaze (δ), odolnost v tahu (τ), nosnost (P_c) a další [39]. Matematické formulace jsou uvedeny v rovnici 44.



Obrázek 3: Struktura parametrů konstrukce svařovaných nosníků

$$\vec{x} = [x_1, x_2, x_3, x_4] = [hltb],$$

Minimalizujte:

$$f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2),$$

S omezeními:

$$\begin{aligned}
g_1(\vec{x}) &= \tau(\vec{x}) - \tau_{max} < 0, \\
g_2(\vec{x}) &= \sigma(\vec{x}) - \sigma_{max} < 0, \\
g_3(\vec{x}) &= \delta(\vec{x}) - \delta_{max} < 0, \\
g_4(\vec{x}) &= x_1 - x_4 < 0, \\
g_5(\vec{x}) &= P - P_c(\vec{x}) < 0, \\
g_6(\vec{x}) &= 0.125 - x_1 < 0, \\
g_7(\vec{x}) &= 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 < 0,
\end{aligned}$$

Rozsah parametrů:

$$\begin{aligned}
0.1 &< x_1 < 2, \\
0.1 &< x_2 < 10, \\
0.1 &< x_3 < 10, \\
0.1 &< x_4 < 2
\end{aligned}$$

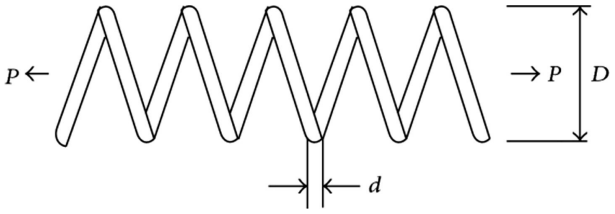
Kde: (44)

$$\begin{aligned}
\tau(\vec{x}) &= \sqrt{\tau'^2 + 2\tau'\tau''\frac{x_2}{2R} + \tau''^2} \\
\tau' &= \frac{P}{\sqrt{2} x_1 x_2} \\
\tau'' &= \frac{MR}{J} \\
M &= P(L + \frac{x_2}{2}) \\
R &= \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \\
J &= 2\left\{\sqrt{2} x_1 x_2 \left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \\
\sigma(\vec{x}) &= \frac{6PL}{x_4 x_3^2} \\
\delta(\vec{x} &= \frac{4PL^3}{E x_3^2 + x_4} \\
P_c(\vec{x}) &= \frac{4.013 E \sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)
\end{aligned}$$

$$P = 6000\,\text{lb}, \quad L = 14\,\text{in.}, \quad \delta_{max} = 0.25\,\text{in.}, \quad E = 30 * 10^6\,\text{psi},$$
$$G = 12 * 10^6\,\text{psi}, \quad \tau_{max} = 13600\,\text{psi}, \quad \sigma_{max} = 30000\,\text{psi}$$

8.1.3 Návrh parametrů pružiny

Cílem funkce je minimalizovat váhu pružiny, tedy množství materiálu, optimalizací jejich parametrů: hrubost použitého drátu (d), průměr vinutí (D) a počet závitů pružiny (P) viz Obrázek 4. Zároveň jsou na návrh kladená omezení [39]. Matematický popis funkce spolu s omezeními je dána rovnicí 45.



Obrázek 4: Struktura parametrů pružiny

$$\vec{x} = [x_1, x_2, x_3] = [dDP],$$

Minimalizujte:

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2,$$

S omezeními:

$$\begin{aligned} g_1(\vec{x}) &= 1 - \frac{x_2^3 x_3}{717,854 x_1^4} < 0, \\ g_2(\vec{x}) &= \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{51108x_1^2} < 0, \\ g_3(\vec{x}) &= 1 - \frac{140.45x_1}{x_2^2x_3} < 0, \\ g_4(\vec{x}) &= \frac{x_1 + x_2}{1,5} - 1 < 0, \end{aligned} \tag{45}$$

Rozsah parametrů:

$$\begin{aligned} 0.05 < x_1 < 2.00, \\ 0.25 < x_2 < 1.30, \\ 2.00 < x_3 < 15.00 \end{aligned}$$

8.2 CEC 2017

V dnešní moderní době se setkáváme s problémy mající stovky proměnných s širokým spektrem omezení. Proto je nutné dbát na škálovatelnost optimalizačních algoritmů, stejně tak i testovacích funkcí [22]. V soutěži CEC 2017 bylo představeno 28 benchmarkových funkcí s množstvím různých omezení, jež jsou volně škálovatelné na 10, 30, 50 a 100 dimenzí. Matematické definice funkcí jsou součástí oficiálního dokumentu soutěže CEC 2017 [22]. Benchmark také určuje postup při zaznamenávání a představení výsledků a definuje maximální počet ohodnocení účelové funkce. Tyto principy jsou reflektovány také při prezentaci výsledků v této práci.

9 Vlastní vylepšení SOMA

Existují obecně známé metody pro zlepšení vyhledávacích algoritmů, jako je využití více populací [17], aplikace chaotických metod [40], [16], či interpolačních metod [12], [28]. Velkou část práce jsem tedy strávil aplikací těchto metod na různé varianty SOMA na účelových funkcích s omezeními. Konkrétně se jednalo o návrh tlakového válce [18], návrh parametrů pružiny [11], návrh konstrukce svařovaných nosníků [19] a na funkcích CEC 2017 benchmarku [22]. Pro srovnání změn chování SOMA po aplikaci úprav bude u každé z nich prezentován graf vývoje průměrného řešení. Průměr bude počítán z 30 nezávislých běhů na funkci optimalizace parametrů konstrukce svařovaných nosníků (viz Sekce 8.1.2) a z 25 nezávislých běhů funkce 5 a 20 benchmarku CEC 2017 na 10 dimenzích. Tyto funkce byly zvoleny, neboť vhodně demonstrují chování algoritmů. Kompletní výsledky všech funkcí jsou součástí přílohy. Osa Y u těchto grafů bude reprezentovat hodnotu účelové funkce daného řešení a osa X udává počet ohodnocení účelové funkce. Kompletní srovnání všech testovacích funkcí a algoritmů s nově navrženou úpravou SOMA, spolu s přehledem nastavením parametrů, bude prezentováno v následujících kapitolách.

9.1 Řešení omezení argumentů ve vlastní implementaci

Jak bylo shrnuto dříve, existuje velké množství způsobů nakládání s jedinci, jež porušují omezení. Vhodné zvolení metody je přitom zásadní pro schopnost algoritmu nalézt splnitelná řešení. Jelikož má být nová verze SOMA použita na škálovatelných funkcích s různou složitostí, bylo vhodné zvolit takovou metodu, která ke své funkčnosti nepotřebuje přesné nastavení parametrů. Volba nejlepšího jedince a případné setřízení populace probíhá turnajovou metodou, stejně jako v C-SOMGA (byla popsána v Sekci 6.1), tedy první jsou brání v úvahu jedinci ve splnitelné oblasti, setřízeni dle hodnoty účelové funkce a následně zbývající jedinci dle míry porušení omezení. Pracuje-li se s hodnotou účelové funkce, například při vytváření nového jedince metodou kvadratické interpolace (viz Rovnice 20), je hodnota získaná neparametrickou penalizační funkcí popsanou rovnicí 6 v sekci 3.

9.2 Dynamická změna strategie

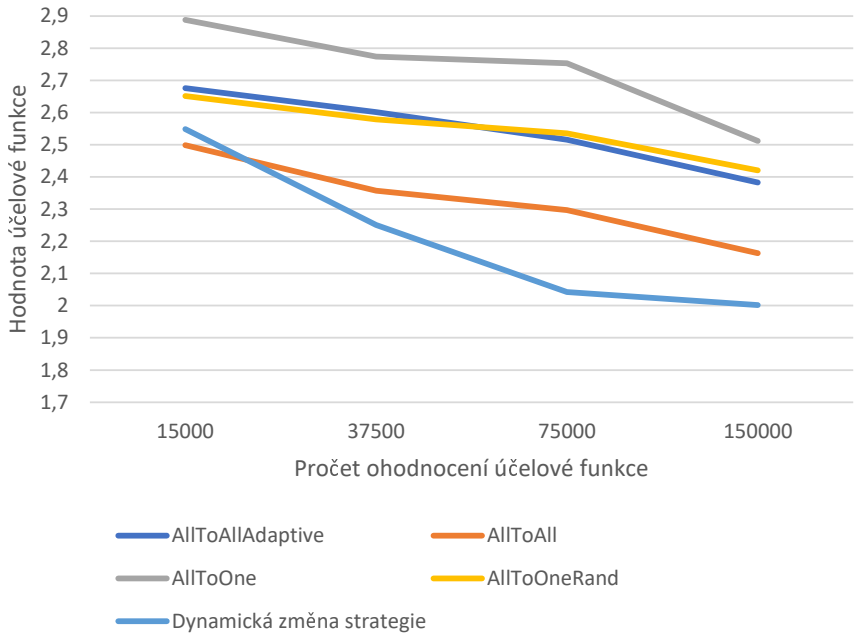
Stejně jako mezi jednotlivými algoritmy, tak i mezi strategiemi SOMA lze pozorovat chování dle no free lunch teorému [8], tedy pro každou strategii existuje optimalizační problém, jež umí vyřešit vhodněji než ostatní. Tento poznatek byl inspirací pro úpravu kombinující vícero strategií v jednom migračním kole. Dvojici kombinovaných strategií je vhodné zvolit tak, aby obě využily během migračního kola stejný počet ohodnocení účelové funkce. Toto splňují dvojice AllToOne, AllToOneRand, a AllToAll, AllToAllAdaptive. Testováním se ukázalo jako vhodnější zvolit dvojici AllToOne a AllToOneRand, proto zde bude prezentována právě tato úprava. Postup algoritmu je popsán v pseudokódu 8.

Algoritmus 8: Pseudokód SOMA s dynamickou změnou strategie

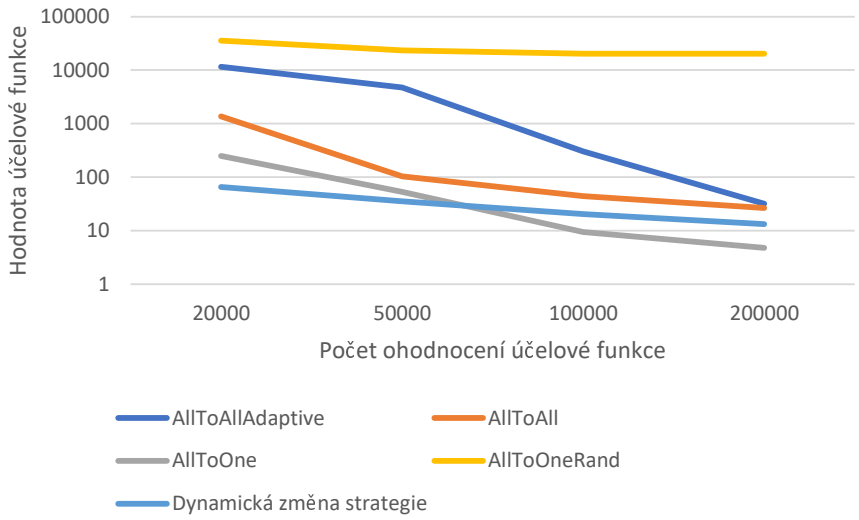
Input:
P: náhodně vygenerovaná populace
f_{cost}: účelová funkce
min_i: minimální počet jedinců v subpopulaci
řídící a ukončovací parametry `// viz tabluka 5`

```
atb = round(size(P) / 2)
while nejsou splněny ukončovací kritéria do
    P = náhodně promíchej(P)
    P_atb = P[: atb]
    P_atr = P[atb :]
    leader = vyber nejlepšího z P_atb
    for j < size(P_atb) do
        // Migrace AllToOne
        i = P_atb[j]
        P_atb[j] = proveď migraci i směrem k leader // viz rovnice 19
    end
    leader = vyber náhodného jedince z P_atr
    for j < size(P_atr) do
        // Migrace AllToOneRand
        i = P_atr[j]
        P_atr[j] = proveď migraci i směrem k leader // viz rovnice 19
    end
    best_atb = nejlepší jedinec z P_atb
    best_atr = nejlepší jedinec z P_atr
    if ohodnocení best_atb > ohodnocení best_atr then
        if (size(P) - atb + 1) > min_i then
            atb = atb + 1
        end
    else
        if (size(P) - atb - 1) > min_i then
            atb = atb - 1
        end
    end
    // Sloučení subpopulací
    P = []
    P.extend(P_atb)
    P.extend(P_atr)
end
return nejlepší jedinec z P
```

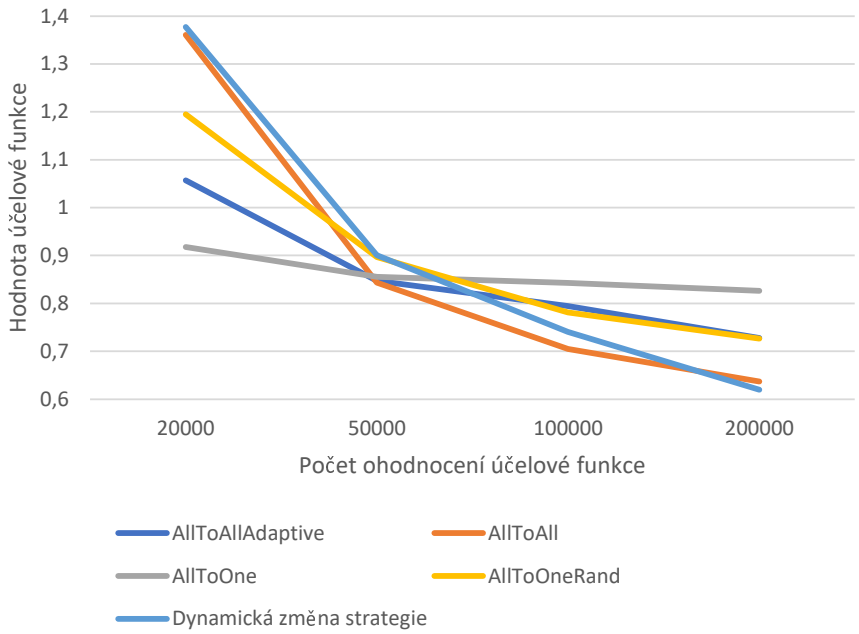
Na začátku migračního kola je populace rozdělena na dvě části tzv. subpopulace. Nejprve je velikost obou částí rovnoměrná, v následujících výpočtech se poměr adaptivně mění dle předchozích výpočtů. Každá nově vzniklá subpopulace provádí migraci dle rozdílné strategie. Následně dochází ke srovnání nalezeného výsledku. Strategie, která se ukázala jako lepší obdrží více jedinců k prohledávání prostoru na úkor strategie horší. Z důvodu udržení diverzity byla implementována podmínka, která zaručí minimální počet min_i jedinců pro horší strategii. Parametr min_i musí být v intervalu $\langle 0, PopSize/2 \rangle$, jako vhodné se ukázalo nastavit jej na 20% velikosti populace.
Grafické porovnání s ostatními strategiemi SOMA na zvolených funkcích je na Obrázcích 5, 6 a 7. Zlepšení průměrné hodnoty účelové funkce, které lze pozorovat na Obrazcích 5 a 7, se projevilo i na většině ostatních testovacích funkcí. Průměrná hodnota Sucess Rate (SR), označující procentuální úspěšnost v nalezení splnitelného řešení, byla 58,45%, což je srovnatelný výsledek se strategiemi AllToOneRand, AllToAll a AllToAllAdaptive, výrazně lepší však byla strategie AllToOne s průměrným SR 72,77%. Vysoká hodnota směrodatné odchylky, vyšší než u všech strategií kromě AllToAllAdaptive, indikuje nízkou robustnost algoritmu, tedy náchylnost na faktory jako je vhodné prvotní náhodné umístění populace. Lze konstatovat, že při zvoleném nastavení parametrů je nový přístup schopen nalézat v průměru lepších hodnot, nicméně obecně nenalezne splnitelná řešení tak často jako strategie AllToOne. S myšlenkou rozdělení populace a následného balancování jsem dále pracoval a testoval i na dalších aspektech SOMA.



Obrázek 5: Porovnání průměrné hodnoty účelové funkce SOMA s dynamickou změnou strategie na funkci návrhu parametrů svařovaných nosníků



Obrázek 6: Porovnání průměrných hodnot účelové funkce SOMA s dynamickou změnou strategie na funkci 5 CEC 2017 na 10 dimenzích



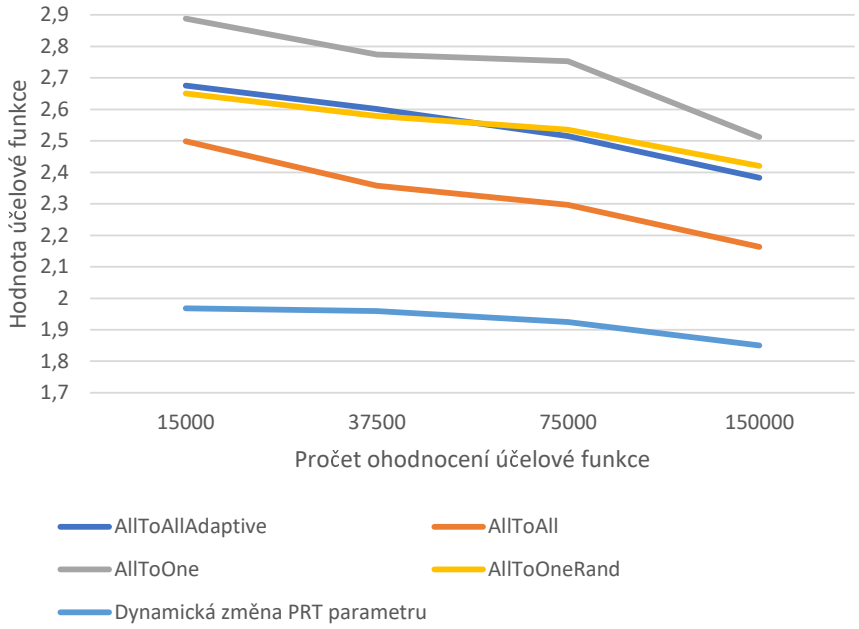
Obrázek 7: Srovnání průměrných hodnot účelové funkce SOMA s dynamickou změnou strategie na funkci 20 CEC 2017 na 10 dimenzích

9.3 Dynamická změna PRT parametru

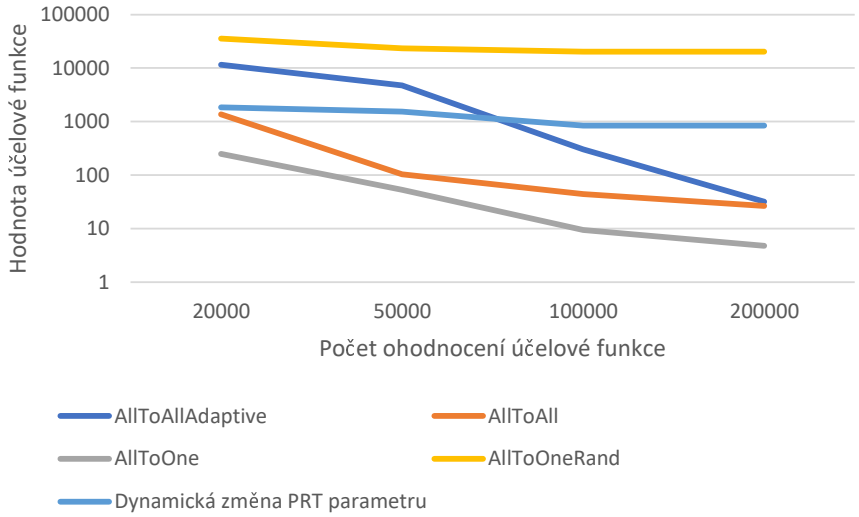
Tato změna vychází z myšlenky dynamické změny strategie. Namísto balancování optimální strategie je balancováno nastavení parametru PRT. Adaptivní ladění parametrů na základě předchozích výsledků je obdobně využito i u řady adaptivních diferenciálních evolucí, například Adaptivní diferenciální evoluce s externím archivem - JADE a Adaptivní diferenciální evoluce s historií úspěšnosti - SHADE [85]. Algoritmus je popsán v pseudokódu 9.
V této úpravě je v každém migračním kole populace náhodně rovnoměrně rozdělena do 3 subpopulací, ke každé z nich je přidělen jiný PRT parametr:

- Subpopulace P_m pracuje s PRT parametrem, který se v minulé generaci ukázal jako nejvhodnější, označeným prt. V první generaci je nastaven náhodně.
- Subpopulace P_l pracuje s PRT parametrem určeným jako náhodné číslo v rozmezí 0 až prt.
- Subpopulace P_h pracuje s PRT parametrem určeným jako náhodné číslo v rozmezí prt až 1.

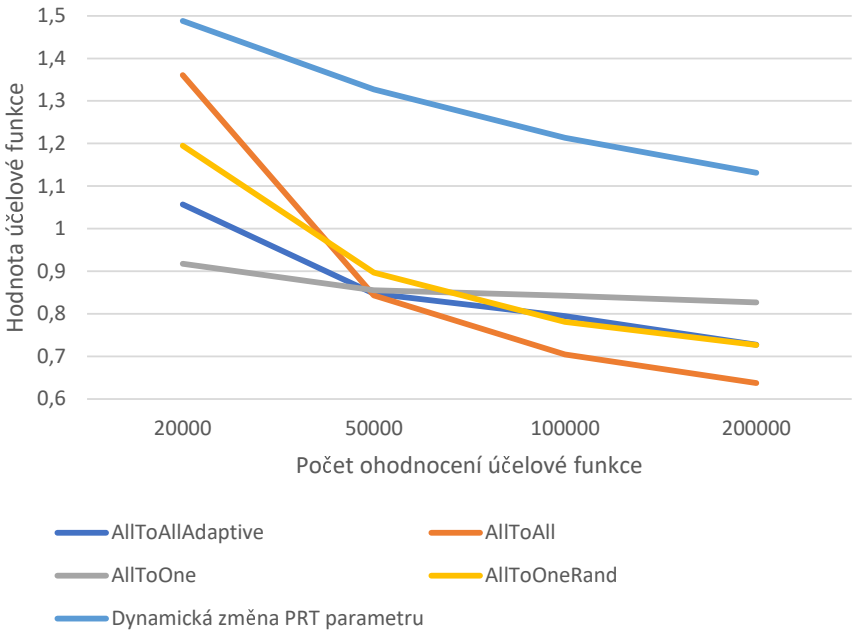
Na každé subpopulaci je poté provedeno migrační kolo dle zvolené strategie. Výsledky nejlepších jedinců v subpopulacích poté určí, kterým směrem se bude hodnota parametru PRT ubírat. Tímto postupem je dosaženo vždy optimálního vybalancování PRT parametru nezávisle na účelové funkci a není třeba jej nastavovat jako parametr algoritmu.
Grafy na obrázcích 8, 9 a 10 zachycují účinnost této strategie. Balancování PRT parametru přineslo zlepšení v hodnotě účelové funkce nalezených řešení v případě jednodušších funkcí. U složitějších funkcí byl algoritmus velice náchylný na vhodné zvolení počáteční populace a PRT parametru, což se projevilo i na extrémně vysokých hodnotách směrodatné odchylky. Například u funkce 20 z CEC 2017 nalezl nejlepší běh z 25 testovacích kvalitnější řešení, než kterákoliv jiná strategie SOMA, nicméně průměr jeho běhů byl výrazně horší. Průměrná hodnota úspěšnosti běhů (SR) byla 61,42%, což můžeme považovat za nevýrazně lepší výsledek, než vykázala předchozí změna. Z důvodu dosti nekonzistentních výsledků nemůžeme tedy považovat tuto úpravu za dostatečně robustní. Byly implementovány a otestovány i další metody založené na myšlence balancování, jako je balancování parametru *step*, či $PathLength$, všechny se však potýkaly se stejným problémem. Dále jsem tedy pracoval pouze s balancováním strategie, zde jsem se dále pokoušel rozvinout více populační přístup.



Obrázek 8: Srovnání průměrných hodnot účelové funkce SOMA s dynamickou změnou PRT parametru na funkci návrhu parametrů svařovaných nosníků



Obrázek 9: Srovnání průměrných hodnot účelové funkce SOMA s dynamickou změnou PRT parametru na funkci 5 CEC 2017 na 10 dimenzích



Obrázek 10: Srovnání průměrných hodnot účelové funkce SOMA s dynamickou změnou PRT parametru na funkci 20 CEC 2017 na 10 dimenzích

Algoritmus 9: Pseudokód SOMA s dynamickou změnou PRT parametru

Input:
P: náhodně vygenerovaná populace
f_{cost}: účelová funkce
řídící a ukončovací parametry `// viz tabluka 5`

```
prt = random(0, 1) // Náhodné číslo v normálním rozdělení
l_prt = random(0, prt)
h_prt = random(prt, 1)
while nejsou splněny ukončovací kritéria do
    P = promíchej P
    sub_size = round(size(P) / 3)
    P_l = P[: sub_size]
    P_m = P[sub_size : size(P) − sub_size]
    P_h = P[size(P) − sub_size :]
    P_l = migruj(P_l, l_prt) // Provede migrační kolo populace P_l s nastavením PRT = l_prt
    P_m = migruj(P_m, prt)
    P_h = migruj(P_h, h_prt)
    l_best = ohodnocení nejlepšího jedince z P_l
    m_best = ohodnocení nejlepšího jedince z P_m
    h_best = ohodnocení nejlepšího jedince z P_h
    if h_best > m_best then
        if h_best > l_best then
            prt = h_prt
        else
            prt = l_prt
        end
    else
        if l_best > m_best then
            prt = l_prt
        end
    end
    l_prt = prt - random(0, 0.1)
    h_prt = prt + random(0, 0.1)
    if l_prt < 0 then
        l_prt = 0
    end
    if h_prt > 1 then
        h_prt = 1
    end
    // Spojení subpopulací
    P = []
    P.extend(P_l)
    P.extend(P_m)
    P.extend(P_h)
end
return nejlepší jedinec z P
```

9.4 Rozšíření o vyhledávání ve více populacích

Při opakovaném spouštění stejného vyhledávacího algoritmu zjišťujeme, že výsledek se často mění. Je to dáno náhodnými faktory jako je náhodné umístění prvotní populace a tak dále. Pokud bychom se chtěli co nejvíce vyvarovat této nahodilosti a pravidelně získávat ty nejlepší řešení, můžeme rozdělit populaci do subpopulací a nad ní spustit daný algoritmus. Tím dojde k obdobě paralelního spuštění více algoritmů za cenu snížení velikosti populace. Intuitivním tedy může být náhodné rozdělení do subpopulací na začátku výpočtu a následné nezávislé spuštění optimalizačního algoritmu na každé z nich. Algoritmus takto upraveného SOMA AllToOne je popsán v pseudokódu 10.

Algoritmus 10: Pseudokód více populační SOMA

Input: P: náhodně vygenerovaná populace
f_{cost}: účelová funkce
pop_count : počet subpopulací
řídící a ukončovací parametry `// viz tabluka 5`

```
1  subpop_size = round(size(P) / pop_count)
2  Pops = []
3  for i<pop_count do
4  |  Pops.append(P[:pop_size])
5  |  P = P[:pop_size]
6  end
7  while nejsou splněny ukončovací kritéria do
8  |  for Pop in Pops do
9  |  |  leader = nejlepší jedinec z Pop
10 |  |  for j < size(Pop) do
11 |  |  |  // Migrace
12 |  |  |  i = Pop[j]
13 |  |  |  Pop[j] = proveď migraci i směrem k leader // viz rovnice 19
14 |  |  end
15 |  end
16 end
17 P = []
18 for Pop in Pops do
19 |  P.extend(Pop)
20 end
21 return Nejlepší jedinec z P
```

Rozdělit populaci na začátku a následně pracovat s každou zvlášť se ukázalo jako nevhodné. Slabé výsledky se projevily zejména u složitějších funkcí, a to z důvodu vyčerpání množství ohodnocení účelové funkce subpopulacemi, které uvázly v lokálním minimu, nebo se navzájem překrývaly. Ke zlepšení došlo u průměrných hodnot účelových funkcí i směrodatné odchylky ve všech inženýrských problémech. Ovšem ve funkcích CEC 2017 algoritmus nedokázal nalézat splnitelná řešení a vykázal průměrnou úspěšnost běhů 50,45%. Průměrné hodnoty účelových

funkcí byly v porovnání s ostatními verzemi SOMA všech případech horší. Pracoval jsem tedy na rozdělování do subpopualcí během každého migračního kola s využitím shlukovacích metod.

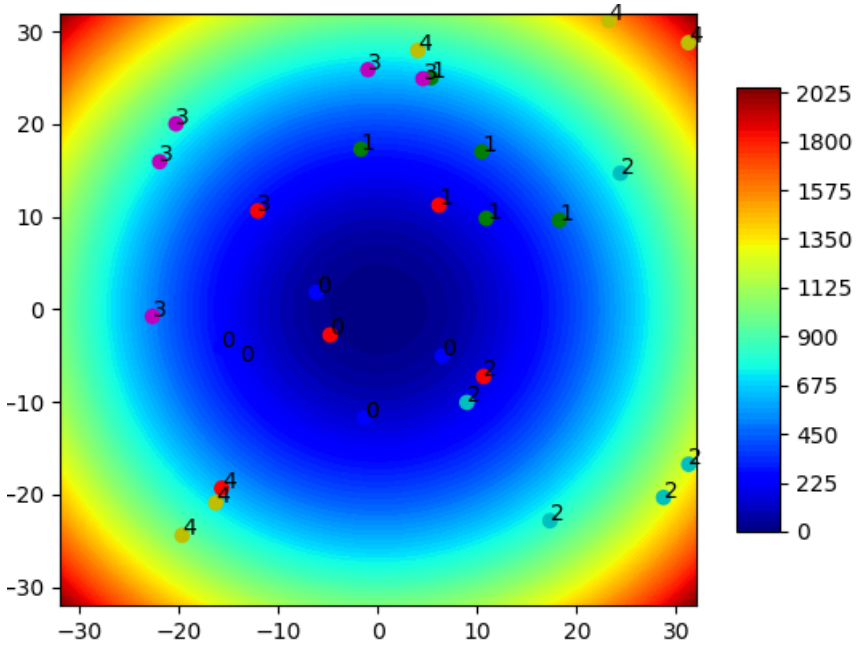
9.4.1 Využití shlukovacích metod

Při rozdělování populace do subpopulací není vždy žádoucí vybírat jedince náhodně. Pro rozdělení se tak užívá shlukovacích metod. Mnoho studií ukázalo, že právě jejich využití je efektivním přístupem k udržení diverzity a pomáhá nalézt optimální řešení [23].
Při hledání vhodné shlukovací metody bylo využito shlukovacích metod pro dynamickou optimalizaci s omezeními (Wang, Yu, Yang, Jiang & Zhao, 2019 [25]) a shlukovací metody uvedené v upravené diferenciální evoluci pro dynamickou optimalizaci (Zhu, Chen, Yuan & Xia, 2018 [24]). Byla vybrána taková, která vytváří subpopulace na základě hodnot účelové funkce, čímž umožňuje subpopulacím s horšími hodnotami účelové funkce zevrubné prohledávání prostoru a zároveň využívá těch lepších ke zpřesňování nejlepšího nalezeného řešení. Tato metoda je popsána v pseudokódu 11.

Algoritmus 11: Pseudokód shlukovací metody více populačního přístupu

```
   Input: P: náhodně vygenerovaná populace
   pop_count : počet subpopulací
 1 subpop_size = round(size(P) / pop_count)
 2 S = [ ] // seznam subpopulací
 3 P_used = [ ] // seznam použitých jedinců
 4 while size(S) < size(P) do
 5 |   best_ind = najdi nejlepšího jedince v populaci, který zároveň není v P_used
 6 |   i = 0
 7 |   P_temp = [ ]
 8 |   while i <subpop_size do
 9 |   |   new_ind = najdi nejbližšího jedince k best_ind, který není v P_used
10 |   |   P_used.append(new_ind)
11 |   |   P_temp.append(new_ind)
12 |   end
13 |   S.append(P_temp)
14 |   i = i + 1
15 end
16 return S
```

Na grafické ukázce vzniklých shluků, která je vyobrazena na obrázku 11, představuje každý bod na grafu jedince v populaci. Červeně je označen vždy nejlepší jedinec v subpopulaci, tedy ten, který byl dle pseudokódu 11 určen jako nejlepší nepoužitý v dané iteraci. Index u bodu značí, do které subpopulace daný jedinec patří. Cílový počet clusterů byl nastaven na 5, velikost populace je 30, každá subpopulace tedy obsahuje 6 jedinců. V prvním kroku algoritmu byl vybrán nejlepší jedinec celé populace (na grafu červený bod s indexem 0) a k němu bylo přiřazeno 5 nejbližších (na grafu modře označení jedinci s indexem 0). V dalším kroku byl zase vybrán nejlepší jedinec,



Obrázek 11: Shluky populací na sférické funkci

který již ovšem není zahrnut v jiné subpopulaci (červeně označený jedinec s indexem 1), k němu bylo přiřazeno 5 nejbližších volných jedinců (zeleně označeni s indexem 1). Ve stejném duchu pokračoval algoritmus do té doby, dokud nejsou všichni jedinci přiřazeni do některé subpopulace.

9.4.2 Zvýšení diverzity překrývajících se populací

Ve složitějších funkcích je žádoucí udržovat určitou míru diverzity - různorodosti populace. S tímto cílem lze snadno rozšířit shlukovací algoritmus který při zjištění příliš blízké polohy dvou shluků posune jedince v horším z nich na náhodné pozice. Algoritmus pro zvýšení diverzity více populační SOMA se shlukováním je popsán v pseudokódu 12. V tomto přístupu porovnáváme vzdálenost nejlepšího a nejhoršího jedince v subpopulaci se vzdáleností nejlepších jedinců v obou porovnávaných subpopulací. Jestliže jsou shluky příliš blízko, jsou všichni jedinci v horší subpopulaci náhodně rozmístěni po ploše. Kritická vzdálenost je daná rovnicí 46. Tímto přístupem neztrácíme drahocenné jedince umístěné na vhodných pozicích a zároveň udržujeme větší diverzitu populace, což může být u některých funkcí velice účinná metoda.

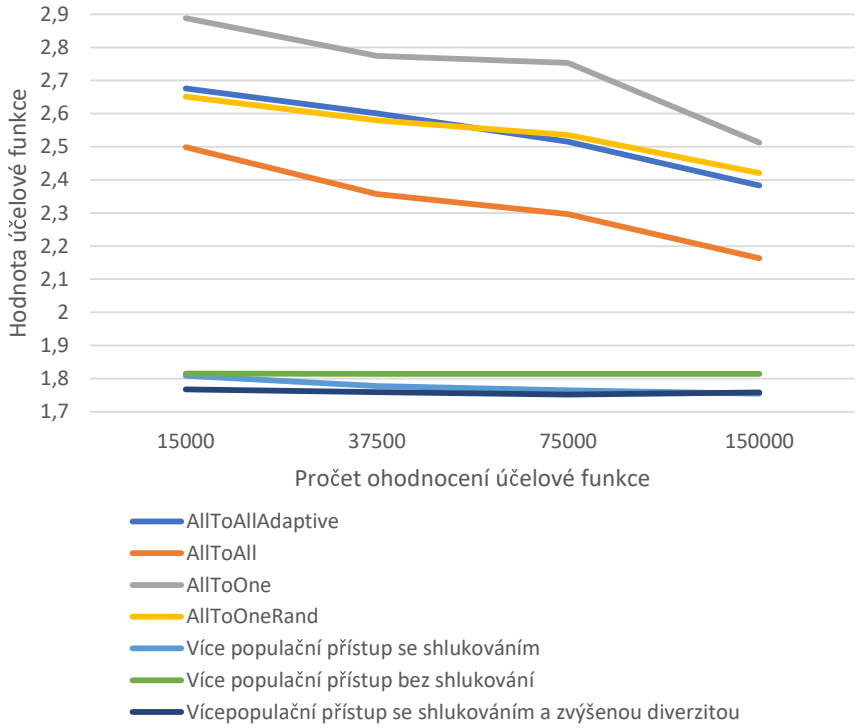
$$|best\_P_1, worst\_P_1|/2 < |best\_P_1, best\_P_2| \tag{46}$$

kde $best\_P_1$ označuje nejlepšího jedince z populace P_1, $best\_P_2$ označuje nejlepšího jedince z populace P_2 a $worst\_P_1$ označuje nejhoršího jedince z populace P_1. K výpočtu vzdálenosti mezi jedinci je využito Euklidovské vzdálenosti.

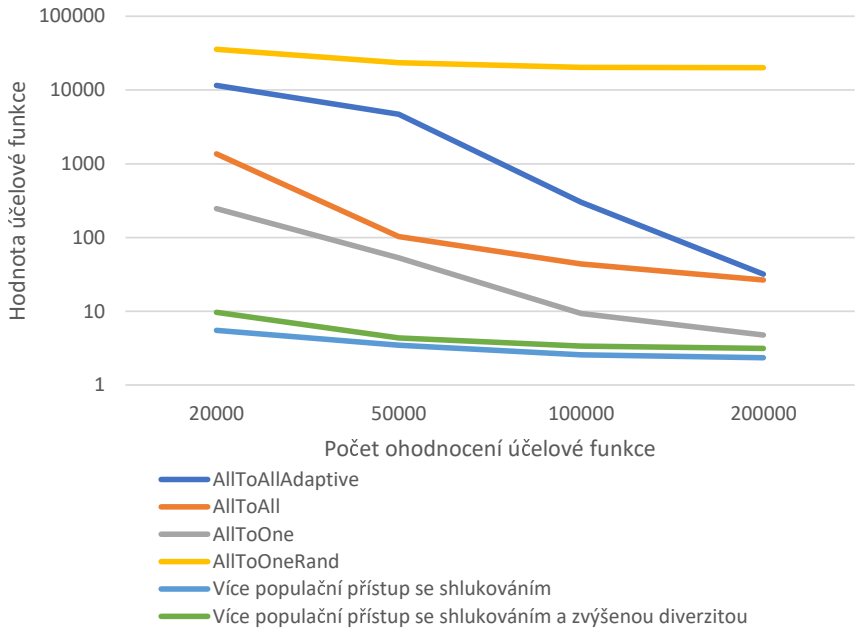
Algoritmus 12: Pseudokód zvýšení diverzity při využití shlukovacích algoritmů

```
   Input: S = seznam subpopulací // viz pseudokód 11
 1 for P_a in S do
 2     for P_b in S do
 3         if P_a == P_b then
 4             continue
 5         end
 6         if P_b je lepší P_a then
 7             P_t = P_b
 8             P_b = P_a
 9             P_a = P_t
10         end
11         dist_a = vzdálenost nejlepšího jedince z P_a a nejhoršího jedince P_a
12         dist_b = vzdálenost nejlepšího jedince ze P_a a nejlepšího jedince P_b
13         if dist_a < dist_b/2 then
14             for ind in P_b do
15                 ind = vytvoř nového jedince na náhodném místě ind = evaluuj ind
16             end
17         end
18     end
19 end
20 return S
```

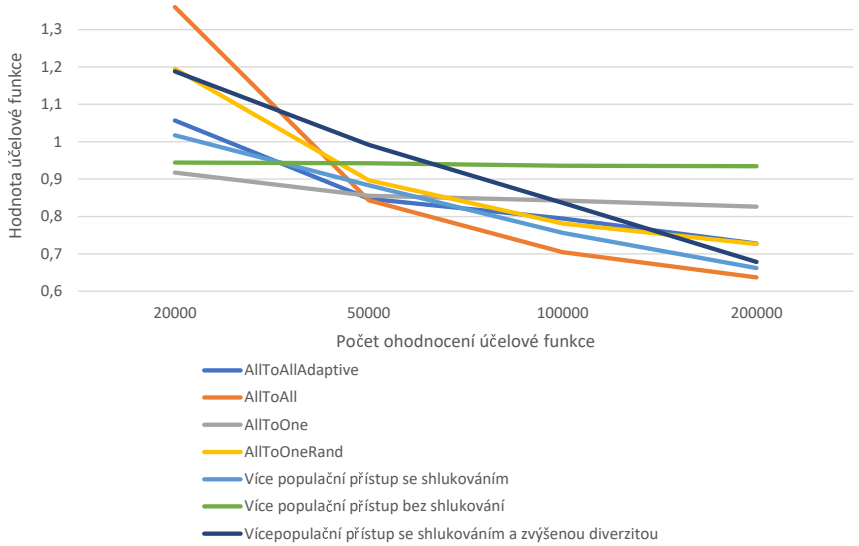
Obrázky 12, 13 a 14 porovnávají průměrné výsledky klasických strategií SOMA s více populačními přístupy. Obrázek 13 nezobrazuje graf více populačního přístupu bez shlukování, jelikož nebylo nalezeno řešení ve splnitelné oblasti. Více populační přístup bez shlukování popsaný výše byl v porovnání s původními strategiemi SOMA zlepšením pouze na inženýrských funkcích. Oba více populační přístupy se shlukováním přinesly ve srovnání s původními strategiemi SOMA zlepšení na většině funkcí. A to jak v hodnotách účelových funkcí, tak v hodnotách směrodatné odchylky. Využití přístupu pro zvýšení diverzity v porovnání s více populačním přístupem se shlukováním nepřineslo výrazné zlepšení a výsledky byly srovnatelné. Průměrná úspěšnost běhů (SR) byla velice uspokojivá s hodnotou 71,16% u shlukovací metody a 72,52% u shlukovací metody s metodou pro zvýšení diverzity. Což je výrazně lepší než strategie AllToAll, AllToAllAdaptive a AllToOneRand a srovnatelné se strategií AllToOne. Velkou robustnost ukázaly i hodnoty směrodatné odchylky, které byly v porovnání s ostatními strategiemi SOMA řádově nižší.



Obrázek 12: Srovnání průměrných hodnot účelové funkce SOMA s více populačním přístupem se shlukováním na funkci návrhu parametrů svařovaných nosníků



Obrázek 13: Srovnání průměrných hodnot účelové funkce SOMA s více populačním přístupem se shlukováním na funkci 5 CEC 2017 na 10 dimenzích



Obrázek 14: Srovnání průměrných hodnot účelové funkce SOMA s více populačním přístupem se shlukováním na funkci 20 CEC 2017 na 10 dimenzích

9.5 Úprava kroku SOMA

Úspěšnost úpravy balancování strategie viz sekce 9.2 mě přivedla na myšlenku vytěžit z obou strategií maximum již během migrace jedinců. Tím by bylo možné vyhnout se rozdělování populace do subpopulací. Také by mohlo dojít ke zlepšení úspěšnosti běhů, které bylo u balancování strategií nízké. Při provádění posunu k jedinci není nově následován pouze jeden jedinec, ale dva. Jeden z těchto jedinců je vždy nejlepší jedinec v populaci, volba druhého jedince byla testována na třech možnostech:

- Náhodný jedinec z populace
- Nově vzniklý jedinec umístěním na náhodné místo v prostoru
- Nový jedinec, umístěný na souřadnice získané kvadratickou interpolací (viz rovnice 20 v Sekci 6.2.1)

Během provádění kroku se vždy jedinec pokusí o skok k oběma cílovým bodům a také k jejich průměrným souřadnicím. Během každého kroku dojde k porovnání výsledků a posunu cílových bodů. Dochází tedy k jejich adaptaci v průběhu migrace. Krok této úpravy algoritmu je uveden také v pseudokódu 13.

Algoritmus 13: Pseudokód úprava migračního kroku

```
Input: P: náhodně vygenerovaná populace
       f_cost: účelová funkce
       cshift, crange: parametry
       řídící a ukončovací parametry // viz tabluka 5
leader_b = nejlepší jedinec v populaci // první cílový jedinec
for i in P do
    if i == leader_b then
        continue
    end
    leader_r = vytvoř jedince dle zvolené strategie // druhý cílový jedinec
    b_c = parametry jedince leader_b
    r_c = parametry jedince leader_r
    a_c = průměr souřadnic b_c a r_coordiantes
    q = step
    best = i // udržuje nejlepšího nalezeného jedince
    while q < path_length do
        b_step = proveď migrační krok i směrem k b_c // viz rovnice 48
        q = q + step
        r_step = proveď migrační krok i směrem k r_c // viz rovnice 48
        q = q + step
        a_step = proveď migrační krok i směrem k a_c // viz rovnice 48
        q = q + step
        t_b = vytvoř nového jedince na souřadnicích b_step
        t_r = vytvoř nového jedince na souřadnicích r_step
        t_a = vytvoř nového jedince na souřadnicích a_step
        ohodnoť t_b, t_r, t_a
        if t_a > t_b and t_a > t_r then
            best_t = t_a
            b_c = získej nové souřadnice krokem k a_c // viz rovnice 49
            r_c = získej nové souřadnice krokem k a_c // viz rovnice 49
        end
        if t_b > t_a and t_b > t_r then
            best_t = t_b
            r_c = získej nové souřadnice krokem k b_coordiantes // viz rovnice 49
        end
        if t_r > t_a and t_r > t_a then
            best_t = t_r
            b_c = získej nové souřadnice krokem k r_c // viz rovnice 49
        end
        if best_t > best then
            best = best_t
        end
    end
    i= best
end
```

Rovnice posunu jedinců po ploše byla na základě mnoha pokusů a iterací upravena do podoby zachycené rovnicí 48. Využití původního způsobu migrace na nově upraveném kroku nepřinášelo na testovacích funkcích, v porovnání s ostatními strategiemi SOMA, výrazně lepších výsledků. Problém byl identifikován ve způsobu perturbace, kdy bylo pozorováno nejuspokojivějších řešení při nastavení PRT parametru na hodnotu 1, tedy když migrace nebyla rušena. Nabízela se možnost nahrazení PRT vektoru náhodně vygenerovaným číslem, obdobně jako je rušení řešeno v CSA, uvedeném v Sekci 7, či PSO [15]. Tento krok se ukázal jako správný, nicméně výkonnost algoritmu byla silně ovlivněna nastavením rozsahu generovaného náhodného čísla. Vznikl proto vzorec, uveden v rovnici 47, umožňující za pomocí parametrů měnit míru i posun čísla udávající rušení. Zakomponování nové perturbace do migrace jedinců popisuje rovnice 48. Adaptace cílových bodů, uvedena v rovnici 49, nijak rušená není.

$$C = c * crange + cshift \tag{47}$$

Kde c je náhodná hodnota v rozmezí 0 až 1, $cshift$ je parametr určující posun hodnoty perturbace jedince v rozmezí -2 až 2 a $crange$ je parametr určující míru velikosti posunu v rozmezí 0 až 3. Záporná hodnota parametru $cshift$ umožňuje jedinci skočit v dané dimenzi směrem od cílového bodu, což může v některých případech pomoci najít vhodné řešení. Správné fungování algoritmu je ovšem postaveno na provádění skoků směrem k cílovému bodu, proto je podmínkou nastavit oba parametry tak, aby v součtu vždy dávaly kladné číslo, tedy $cshift + crange > 0$.

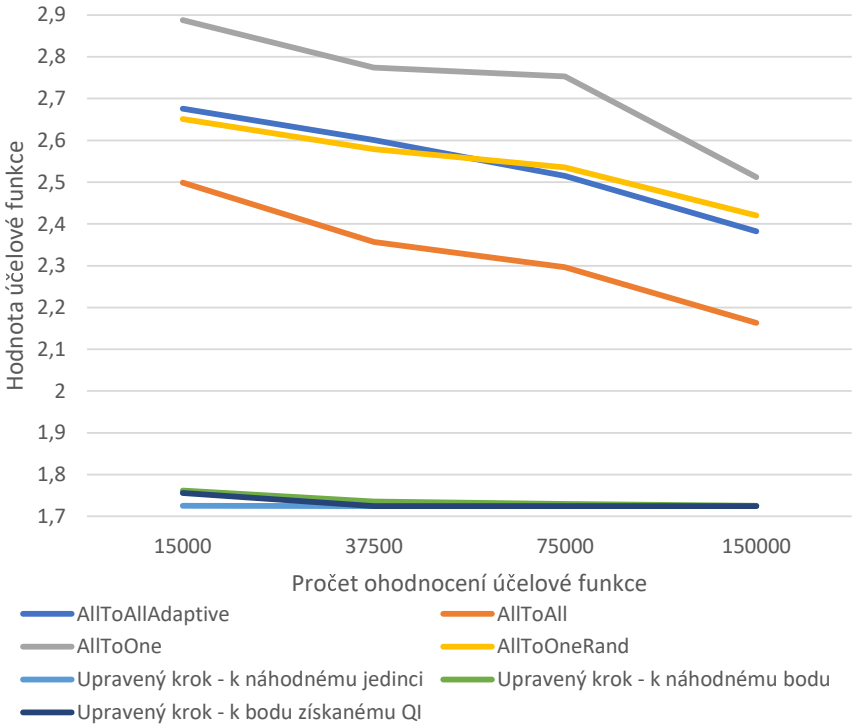
$$\vec{x}^{ML+1} = \vec{x}^{ML}_{start} + (\vec{x}^{ML}_{T} - \vec{x}^{ML}_{start}) * t * C \tag{48}$$

$$\vec{x}^{ML+1} = \vec{x}^{ML} + (\vec{x}^{ML}_{T} - \vec{x}^{ML}) * t \tag{49}$$

kde $t \in [0, PathLength]$ inkrementována hodnotu parametru $step$ a x_T označuje konkrétní cílový bod. Určení, který cílový bod bude při adaptaci posunut se řídí následujícími pravidly:

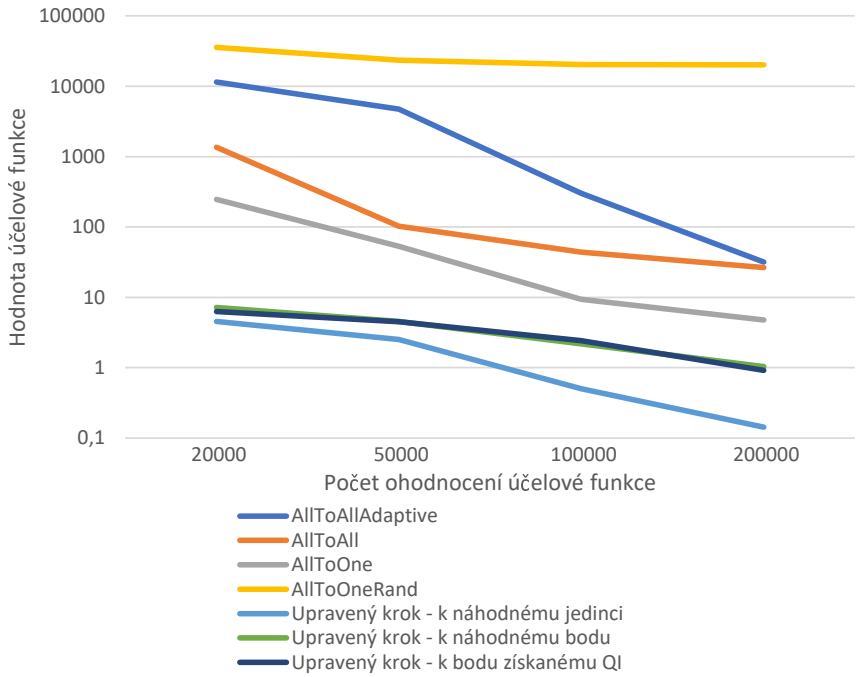
- Jestliže bylo nalezeno nejlepší řešení při posunu směrem k cílovému bodu, označme jej a, je posunut druhý cílový bod směrem k bodu a.
- Jestliže bylo nalezeno nejlepší řešení při posunu k průměru cílových bodů, jsou oba cílové body posunuty směrem k tomu druhému.

Po skončení migrace se daný jedinec, stejně jako v původní SOMA, posune na nejlepší nalezené místo. Následující grafy 15, 16 a 17 porovnávají průměrné hodnoty účelových funkcí všech tří variant upravené migrace. Na grafech 15 a 17 lze pozorovat výrazné zlepšení nové strategie v porovnání se strategiemi původními, tento trend se opakoval i u většiny ostatních funkcí. Mezi výjimky patří funkce 20 z benchmarku CEC 2017 zachycená na Obrázku 17, kde dokázaly tradiční metody nalézt přesnějších průměrných výsledků. Obecně však nová metoda ukázala

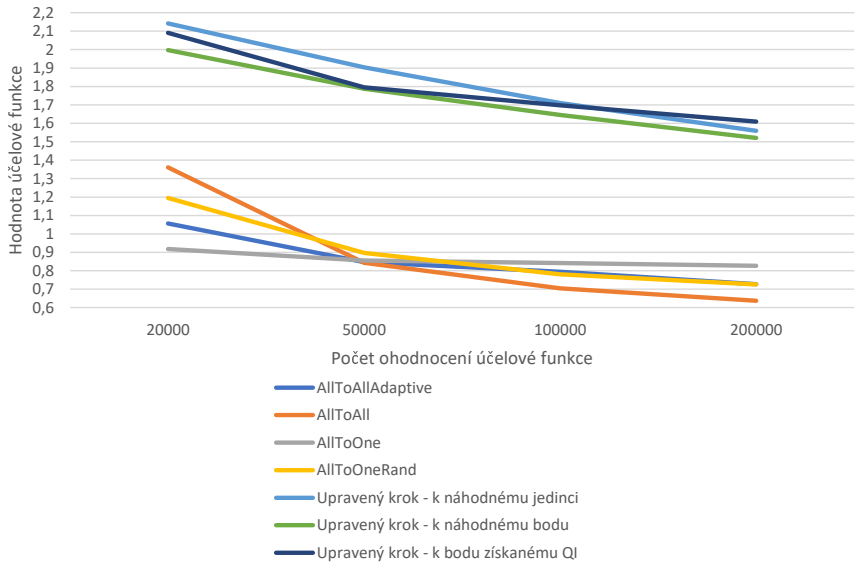


Obrázek 15: Srovnání průměrných hodnot účelové funkce SOMA s upraveným migračním krokem na funkci návrhu parametrů svařovaných nosníků

zlepšení jak v kvalitě hodnot účelových funkcí nalezených řešení, tak v úspěšností běhů. Výrazně lepší byly také hodnoty směrodatné odchylky, což ukazuje vysokou robustnost nové strategie.



Obrázek 16: Srovnání průměrných hodnot účelové funkce SOMA s upraveným migračním krokem na funkci 5 CEC 2017 na 10 dimenzích



Obrázek 17: Srovnání průměrných hodnot účelové funkce SOMA s upraveným migračním krokem na funkci 20 CEC 2017 na 10 dimenzích

9.5.1 Využití chaosu

S cílem dalšího zlepšení algoritmu bylo otestováno využití chaotických hodnot, ty byly dosazovány do proměnné c, viz rovnice 47, která předtím byla získávána generátorem náhodných čísel (RNG). Testovány byly veškeré systémy uvedeny v sekci 7.2.3. Jedná se o systémy, které patří dle literatury (Kohli a Arora 2018 [16]) k nejpoužívanějším a vhodným pro adaptaci evolučními algoritmy. Konkrétně se jedná o Bernoulliho mapu (rovnice 33), Logistickou mapu (rovnice 34), Čebyševovu mapu (rovnice 35), Kruhovou mapu (rovnice 36), ICMIC (rovnice 37), Sinusiodální mapu (rovnice 38) a Stanovou mapu (rovnice 39). Provedeno bylo 30 nezávislých běhů u inženýrských problémů a 25 u funkcí z CEC 2017. Vliv chaotických posloupností na úspěšnost běhů zachycuje tabulka úspěšnosti běhů 3, kde hodnota značí procentuální úspěšnost v nalezení splnitelných řešení. V tabulce byly vynechány problémy, na kterých vykázaly všechny přístupy hodnoty 100%. Využití Bernoulliho mapy nalezlo nejlepší úspěšnost (SR) ve 12 funkcích, Logistická mapa ve 13, Čebyševova ve 12, Kruhová v 1, ICMIC ve 14, Sinusoidální ve 13, Stanová v 11 a využití klasického generátoru pseudonáhodných čísel ve 12 případech. V tabulce je také uveden průměr SR, v němž jsou zahrnuty výsledky všech testovací funkcí (tedy i těch, u kterých mají všechny algoritmy 100% úspěšnost a nejsou uvedeny v Tabulce 3).
Srovnání v Tabulce 4 ukazuje průměrnou hodnotu účelových funkcí jednotlivých metod, a to pouze v případě, že byly všechny běhy úspěšné, tedy když SR = 100%. Nejlepší nalezené výsledky jsou v tabulkách zvýrazněny **tučně**. V tabulce 4 je pro problém Návrhu parametrů tlakového válce užito zkratky PV, Návrhu parametrů pružiny SD a Návrhu parametrů svařovaných nosníků WB, funkce z benchmarku CEC 2017 jsou označeny jejich identifikačním číslem. Využití Čebyševovy mapy přineslo nejlepší průměrnou hodnotu účelových funkcí v 10 případech, přičemž využití Logistické mapy ve 4, ICMIC, Stanová mapa, Sinusoidální a generátor pseudonáhodných čísel ve 3 případech a Bernoulliho mapa ve 2.
Je tedy zřejmé, že pro dané problémy s omezeními na nové úpravě SOMA je vhodné využít Čebyševovu mapu pro generování chaotických čísel namísto těch generovaných RNG v těch případech, kdy bylo nalezeno splnitelné řešení. Pakliže se snažíme nalézt splnitelné řešení, je nejvhodnější použít ke generování náhodných čísel ICMIC. Díky tomuto pozorování byla v nové strategii s upraveným migračním krokem provedena následující změna: V každém migračním kole je kontrolováno, zda v populaci existuje jedinec umístěný ve splnitelné oblasti, pakliže ano, je použito ke generování náhodného čísla c Čebyševova mapa, v opačném případě je použit ICMIC.

Funkce	Bernoulliho	Logistická	Čebyševova	Kruhová	ICMIC	Sinusoidální	Stanová	RNG
f_{CEC_3}	48	24	16	**64**	44	20	44	24
f_{CEC_6}	92	**96**	0	32	**96**	88	**96**	92
f_{CEC_7}	52	52	0	24	**64**	44	60	56
f_{CEC_8}	**100**	**100**	**100**	0	**100**	**100**	**100**	**100**
f_{CEC_9}	**100**	**100**	**100**	52	**100**	**100**	**100**	**100**
$f_{CEC_{10}}$	**100**	**100**	**100**	0	**100**	**100**	80	**100**
$f_{CEC_{11}}$	24	12	12	0	16	**32**	0	24
$f_{CEC_{12}}$	**100**	**100**	**100**	0	**100**	**100**	**100**	**100**
$f_{CEC_{13}}$	64	68	**100**	0	56	76	92	56
$f_{CEC_{14}}$	**100**	**100**	**100**	0	**100**	**100**	**100**	**100**
$f_{CEC_{15}}$	**100**	**100**	**100**	76	**100**	**100**	**100**	**100**
$f_{CEC_{16}}$	**100**	**100**	**100**	80	**100**	**100**	**100**	**100**
$f_{CEC_{17}}$	0	0	0	0	0	0	0	0
$f_{CEC_{18}}$	**100**	**100**	**100**	0	**100**	**100**	64	**100**
$f_{CEC_{19}}$	0	0	0	0	0	0	0	0
$f_{CEC_{21}}$	**100**	**100**	**100**	0	**100**	**100**	**100**	**100**
$f_{CEC_{22}}$	28	40	**100**	0	52	60	24	48
$f_{CEC_{23}}$	**100**	**100**	**100**	0	**100**	**100**	**100**	**100**
$f_{CEC_{24}}$	**100**	**100**	92	44	**100**	**100**	**100**	**100**
$f_{CEC_{25}}$	**100**	**100**	96	44	**100**	**100**	**100**	**100**
$f_{CEC_{26}}$	0	0	0	0	0	0	0	0
$f_{CEC_{27}}$	72	52	84	0	72	**76**	4	72
$f_{CEC_{28}}$	0	0	0	0	0	0	0	0
Průměr	76,8	76,3	72,9	39,2	77,4	77,3	73,0	76,5

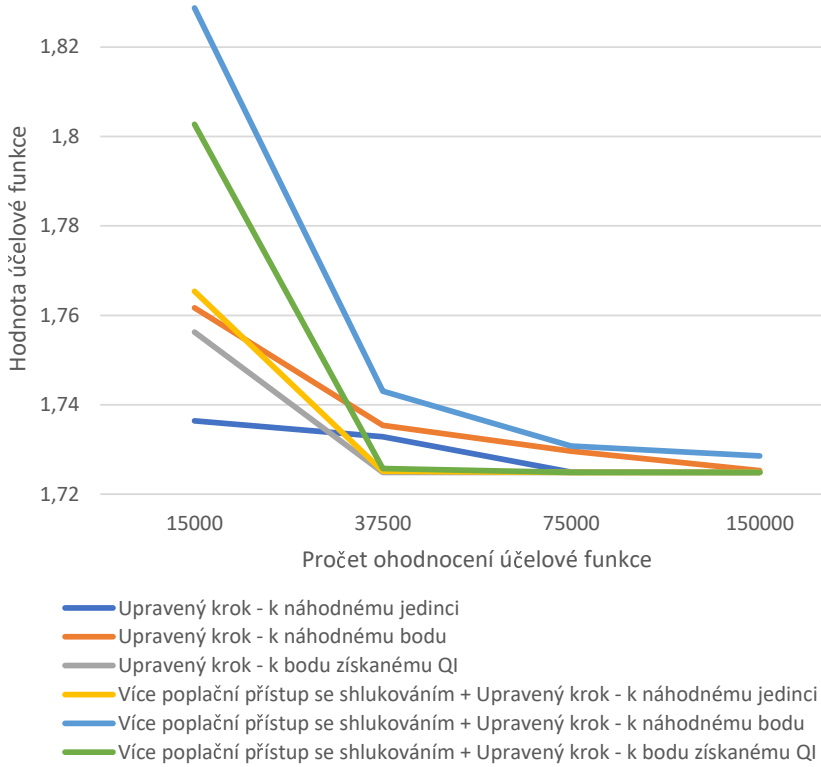
Tabulka 3: Porovnání procentuální úspěšnosti běhů v % SOMA s různými způsoby generování chaotických a náhodných čísel

Funkce	Bernoulliho	Logistická	Čebyševova	ICMIC	Sinusoidální	Stanová	RNG
f_{WB}	**1,73E+00**	1,79E+00	**1,73E+00**	**1,73E+00**	**1,73E+00**	**1,73E+00**	**1,73E+00**
f_{SD}	1,23E−02	1,30E−02	1,31E−02	1,27E−02	1,28E−02	1,29E−02	**1,27E−02**
f_{PV}	7,11E+03	**6,38E+03**	6,46E+03	6,65E+03	6,45E+03	6,49E+03	6,55E+03
f_{CEC_1}	1,36E−19	1,73E−19	**2,62E−27**	8,82E−21	2,45E−20	1,18E−20	1,64E−18
f_{CEC_2}	4,49E−19	5,73E−19	**3,17E−27**	2,90E−20	2,32E−20	6,73E−22	2,26E−19
f_{CEC_8}	**−1,35E−03**	**−1,35E−03**	**−1,35E−03**	**−1,35E−03**	**−1,35E−03**	−1,33E−03	−1,35E−03
f_{CEC_9}	1,51E−01	7,02E−01	1,14E+00	5,78E−02	3,48E−01	1,76E−01	**2,44E−02**
$f_{CEC_{10}}$	−5,10E−04	−5,10E−04	−5,10E−04	**−5,09E−04**	−5,09E−04	–	−5,09E−04
$f_{CEC_{12}}$	1,03E+01	**7,85E+00**	1,03E+01	1,47E+01	7,19E+00	7,94E+00	1,06E+01
$f_{CEC_{13}}$	–	–	**6,38E−01**	–	–	–	–
$f_{CEC_{14}}$	2,60E+00	**2,59E+00**	2,60E+00	2,61E+00	2,62E+00	3,01E+00	2,60E+00
$f_{CEC_{15}}$	1,49E+01	1,61E+01	**7,63E+00**	1,51E+01	1,35E+01	1,28E+01	1,42E+01
$f_{CEC_{16}}$	6,23E+01	5,93E+01	**4,82E+01**	5,88E+01	5,59E+01	5,61E+01	5,68E+01
$f_{CEC_{18}}$	1,13E+02	1,34E+02	1,21E+02	1,37E+02	**4,87E+01**	–	5,14E+01
$f_{CEC_{20}}$	6,86E−01	9,81E−01	1,26E+00	7,82E−01	9,53E−01	**5,99E−01**	8,18E−01
$f_{CEC_{21}}$	8,69E+00	7,09E+00	**3,99E+00**	1,13E+01	9,97E+00	1,37E+01	7,44E+00
$f_{CEC_{22}}$	–	–	**3,20E+03**	–	–	–	–
$f_{CEC_{23}}$	2,76E+00	2,86E+00	3,28E+00	2,79E+00	2,89E+00	**2,73E+00**	2,80E+00
$f_{CEC_{24}}$	1,49E+01	1,42E+01	**8,01E+00**	1,47E+01	1,40E+01	1,15E+01	1,52E+01
$f_{CEC_{25}}$	6,72E+01	6,43E+01	–	6,27E+01	**6,23E+01**	–	6,54E+01

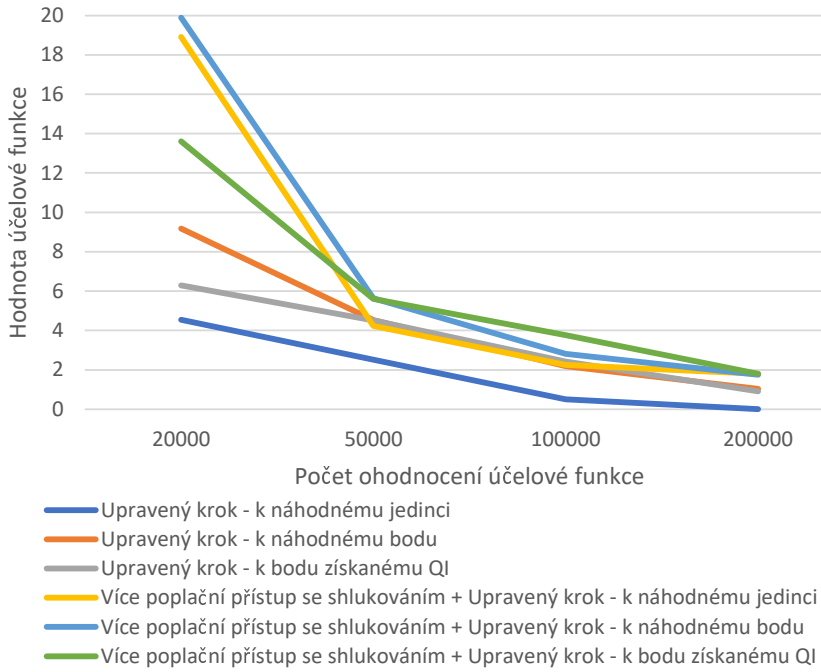
Tabulka 4: Porovnání průměrných hodnot účelové funkce získaných SOMA různými způsoby generování chaotických a náhodných čísel

9.6 Zkombinování více změn

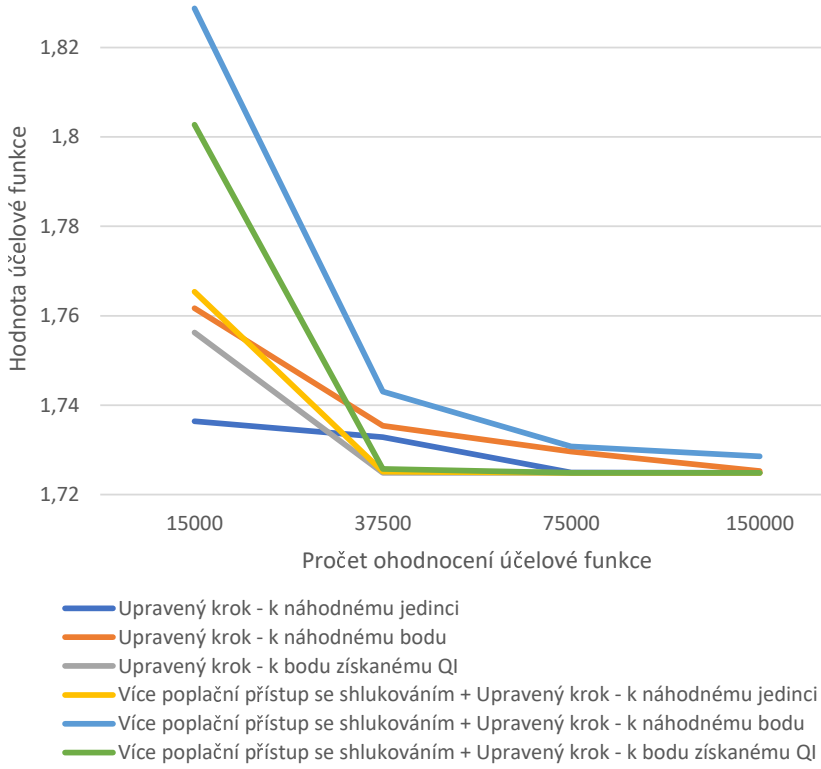
Vybízejícím se přístupem je bezesporu zkombinování výše uvedených změn. Při balancování strategie či parametrů, stejně jako u více populačních přístupů dochází k rozdělování populace do subpopulací, není proto vhodné tyto metody navzájem kombinovat. Vyzkoušel jsem ovšem zlepšit kvalitu výsledků upraveného kroku popsaného v sekci 9.5 a více populačního přístupu se shlukováním 9.4.1. Srovnání průměrných hodnot účelové funkce této kombinace s úpravou kroku na zvolených funkcích je prezentováno na obrázcích 18, 19 a 20.
Na funkci nastavení parametrů svařovaných nosníků(na Obrázku 18) vykázala kombinace zlepšení. U zvolených funkcí z CEC 2017 k výrazně lepším výsledkům nedošlo. Při analýze kombinací se ukázala jako nejlepší metoda SOMA s upraveným krokem, kdy se druhý jedinec volí dle kvadratické interpolace a její kombinace s víceúčelovým přístupem se shlukováním. Tyto dvě metody proto budou dále porovnávány s dalšími algoritmy v Kapitole 11.



Obrázek 18: Srovnání průměrných hodnot účelové funkce SOMA s upraveným krokem s kombinací více populačního přístupu a upraveného kroku na funkci návrhu parametrů svařovaných nosníků



Obrázek 19: Srovnání průměrných hodnot účelové funkce SOMA s upraveným krokem s kombinací více populačního přístupu a upraveného kroku na funkci 5 CEC 2017 na 10 dimenzích



Obrázek 20: Srovnání průměrných hodnot účelové funkce SOMA s upraveným krokem s kombinací více populačního přístupu a upraveného kroku na funkci 20 CEC 2017 na 10 dimenzích

10 Nastavení parametrů

Nastavení parametrů algoritmů je zásadní pro jejich výkonnost, hodnoty jsou proto převzaty z dostupné literatury. Velikost smečky v GWO je nastavena na 30 vlků. CGWO, s velikostí smečky 30 vlků využívá ke generování čísel stanovou (tent) chaotickou mapu. CSA operuje s velikostí hejna 50 vran a parametry $ap = 0.1$ a $fl = 2$. Algoritmus SCA pracuje s velikostí populace 30 jedinců a parametry $r_1 = 3$, $r_2 = 2\pi$, $r3 = 3$ a $r_4 = 0.5$. SOMA a CSOMA-QI pracuji s velikostí populace 10 jedinců, $path\_length = 3$, $step = 0.11$ a $prt = 0.9$. SOMA T3A má velikost populace 30 jedinců, parametry $k = 10$, $n = 4$, $m = 10$ a $N_{jumps} = 45$. Vlastní úpravy SOMA jsou potom nastaveny následovně:

- SOMA s balancováním PRT: velikost populace = 20, strategie migrace AllToOne, $path\_length = 3$, $step = 0.11$ a $prt = 0.9$.
- SOMA s balancováním strategií: balancuje se mezi strategiemi AllToOne a AllToOneRand, velikost populace = 20, minimální velikost populace jedné strategie = 5, $path\_length = 3$, $step = 0.11$ a $prt = 0.9$.
- Více populační SOMA velikost populace = 6, počet populací = 5, $path\_length = 3$, $step = 0.11$ a $prt = 0.9$.
- Více populační SOMA se shlukováním: strategie AllToOne, velikost populace = 6, počet populací = 5, vzdálenost dvou bodů je vypočítávána Euklidovskou vzdáleností, $path\_length = 3$, $step = 0.11$ a $prt = 0.5$.
- SOMA s migrací ke dvěma bodům: velikostí populace = 10, chaotická mapa = Čebyševova - je li v populaci jedinec ve splnitelné oblasti, jinak ICMIC, $path\_length = 3$, $step = 0.11$, $chaotic\_range = 1$ a $chaotic\_shift = 0$ pro CEC 2017.
  Velikostí populace = 10, chaotická mapa = Čebyševova, $path\_length = 3$, $step = 0.17$, $chaotic\_range = 3$ a $chaotic\_shift = -0.5$ pro inženýrské problémy.

11 Prezentace výsledků

Jako nejlepší nová strategie se ukázala úprava kroku SOMA, kdy je druhý cílový bod určen dle kvadratické interpolace. Slibných výsledků bylo také dosaženo kombinací této metody s více populačním přístupem se shlukováním. Tyto zlepšení proto budou porovnávány s původními strategiemi SOMA a s variantami SOMA T3A a C-SOMA QI, jež byly dříve představeny v Sekci 6. Dojde také ke srovnání s vybranými state-of-art algoritmy. Algoritmy SOMA, CSA, GWO, CGWO a SCA byly naprogramovány v jazyku Python 3.6, výpočet byl proveden na procesoru Intel i7-6700HQ s pamětí 16 GB RAM, výsledky ostatních uvedených algoritmů byly převzaty z literatury, a to tak, aby byl počet nezávislých běhů a nastavení terminálních podmínek shodný. Pro snadnější orientaci v tabulkách bude pro strategie algoritmu SOMA využíváno následujících zkratek:

- SOMA AllToOne: SOMA ATB
- SOMA AllToOneRand: SOMA ATR
- SOMA AllToAll: SOMA ATA
- SOMA AllToAllAdaptive: SOMA ATAA
- Více populační SOMA se shlukováním viz 9.4.1: SOMA MPC
- SOMA s upraveným krokem viz 9.5, kdy je druhý cílový bod generován kvadratickou interpolací: SOMA M3-QI
- Kombinace SOMA M3-QI s více populačním přístupem se shlukováním:
  SOMA M3-QI+MPC, případně S M3-QI+MPC

11.1 Výsledky inženýrských problémů

Nové strategie SOMA jsou porovnány na inženýrských problémech s původními strategiemi SOMA, dále s algoritmy představenými v této práci: C-SOMA QI, C-SOMA T3A, GWO [20], CGWO [16], CSA [11], SCA [41]. A dalšími state-of-ar algoritmy, jejichž výsledky byly převzaty z literatury [16], [20]: algoritmy roje částic C-PSO, H-PSO a U-PSO [15], genetickými algoritmy GA4 [46] a GA3 [45], algoritmem kombinujícím PSO a diferenciální evoluci PSODE [47], evoluční strategií $(\mu + \lambda)$ES a učícím se algoritmem TLBO [52]. Maximální počet ohodnocení účelové funkce je nastaven tak, aby byl v souladu s převzatými výsledky, tedy: 150 000. Prezentované výsledky vychází z 30 nezávislých běhů.

Nově představené algoritmy jsou v následujících tabulkách zvýrazněny **tučně**, chybějící hodnoty jsou označeny jako N.A. Sloupec Min označuje nejhorší nalezené řešení, Max potom nejlepší, STD je označení pro směrodatnou odchylku. Významnost rozdílu průměrné hodnoty účelových funkcí všech algoritmů na jednotlivých inženýrských problémech byla porovnána na základě t-testu na hladině významnosti 95%. V posledním sloupci tabulek je na základě p-hodnoty uvedeno, zda byla průměrná hodnota výrazně lepší, horší, ši srovnatelná s hodnotou nově představeného algoritmu SOMA M3-QI. Kompletní výsledky t-testu jsou součástí přílohy.

Na funkci návrhu parametrů pružiny, jejichž výsledky jsou uvedeny v Tabulce 5, dosahoval nejlepších výsledků algoritmus PSODE a TLBO. Rozdíl ve výsledcích SOMA M3-QI a PSODE je navíc dle t-testu statisticky významný. Průměrné hodnoty účelových funkcí všech ostatních algoritmů jsou naopak horší než nově představené algoritmy. Mezi nejnižší také patří hodnota směrodatné odchylky, což ukazuje jejich vysokou robustnost.

Výsledky algoritmů na funkci návrhu parametrů tlakového válce uvádí Tabulka 6. I zde nalezly nové metody průměrně třetí a čtvrté nejlepší řešení. Předčily je algoritmy PSODE a TLBO. Opět lze pozorovat, v porovnání s původními strategiemi SOMA, zlepšení v hodnotách směrodatné odchylky.

Hodnoty dosažené na funkci návrhu parametrů svařovaných nosníků jsou uvedeny v Tabulce 7. Dominantní byly algoritmy PSODE a TLBO, následované nově prezentovanými SOMA M3-QI a SOMA M3-QI+MPC. Průměrné výsledky všech ostatních algoritmů byly statisticky významně horší.

Výsledky nově prezentovaných strategií SOMA na inženýrských problémech indikují výrazné zlepšení ve srovnáním s ostatními strategiemi SOMA. Je to dáno zejména novou schopností adaptace během migrace, která umožňuje velice rychle nalézt na inženýrských problémech přibližné umístění optima. Algoritmus má potom velký prostor ke zpřesňování výsledků. Nově představená metoda perturbace zase zajišťuje, aby algoritmus neuvázl v lokálním extrému a došlo k dostatečnému prohledání prostoru řešení.

Algoritmus	Min	Průměr	Max	STD	
PSODE	1,2665E-02	1,2665E-02	1,2665E-02	1,2000E-08	+
TLBO	N.A.	1,2666E-02	1,2665E-02	N.A.	
SOMA M3-QI	1,2770E-02	1,2684E-02	1,2665E-02	2,7232E-05	
SOMA M3-QI + MPC	1,3066E-02	1,2701E-02	1,2665E-02	7,5683E-05	≈
HPSO	1,2719E-02	1,2707E-02	1,2665E-02	1,5800E-05	−
CPSO	1,2924E-02	1,2733E-02	1,2675E-02	5,2000E-04	≈
GA4	1,2973E-02	1,2742E-02	1,2681E-02	5,9000E-05	−
GA3	1,2822E-02	1,2769E-02	1,2705E-02	3,9400E-05	−
$(\mu + \lambda)$ES	1,2670E-02	1,2769E-02	1,2665E-02	1,3570E-06	−
GWO	1,3112E-02	1,2776E-02	1,2671E-02	1,0333E-04	−
CGWO	1,3269E-02	1,2805E-02	1,2676E-02	1,6004E-04	−
SOMA ATA	1,7773E-02	1,2935E-02	1,2666E-02	5,0271E-04	−
CSA	1,3482E-02	1,2960E-02	1,2707E-02	1,8180E-04	−
SOMA ATAA	1,6379E-02	1,3194E-02	1,2665E-02	7,9980E-04	−
SOMA ATR	1,7709E-02	1,3241E-02	1,2665E-02	9,1990E-04	−
SOMA QI	2,3561E-02	1,3457E-02	1,2665E-02	3,0873E-03	≈
SOMA T3A	1,4718E-02	1,3470E-02	1,2665E-02	5,6042E-04	−
SOMA ATB	1,7746E-02	1,4094E-02	1,2666E-02	1,3827E-03	−
SCA	3,5384E-02	1,7726E-02	1,2957E-02	4,9233E-03	−
UPSO	N.A.	2,2940E-02	1,3120E-02	7,2000E-03	−

Tabulka 5: Výsledky hodnot účelové funkce algoritmů na funkci návrhu parametrů pružiny, kde +, − a ≈ označují výsledky t-testu, tedy zda jsou na hladině významnosti 95% statisticky významně lepší, horší, či jsou srovnatelné.

Algoritmus	Min	Průměr	Max	STD	
PSODE	N.A.	6,0597E+03	6,0597E+03	N.A.	
TLBO	N.A.	6,0597E+03	6,0597E+03	N.A.	
SOMA M3-QI	6,4101E+03	6,0905E+03	6,0597E+03	7,1860E+01	
SOMA M3-QI + MPC	6,3191E+03	6,0905E+03	6,0597E+03	4,5012E+01	$\approx$
HPSO	6,2887E+03	6,0999E+03	6,0597E+03	8,6200E+01	$\approx$
CPSO	6,3638E+03	6,1471E+03	6,0611E+03	8,6450E+01	$-$
GA4	6,4693E+03	6,1773E+03	6,0599E+03	1,3093E+02	$-$
CGWO	7,3359E+03	6,2606E+03	6,0601E+03	3,3712E+02	$-$
GA3	6,3085E+03	6,2938E+03	6,2887E+03	7,4133E+00	$-$
CSA	6,8204E+03	6,3326E+03	6,0776E+03	1,8744E+02	$-$
SOMA ATA	7,3328E+03	6,3708E+03	6,0597E+03	4,5933E+02	$-$
SOMA T3A	7,2737E+03	6,3708E+03	6,0905E+03	3,1481E+02	$-$
$(\mu + \lambda)$ES	N.A.	6,3799E+03	6,0597E+03	2,1000E+02	$-$
SOMA ATAA	7,5445E+03	6,4101E+03	6,0597E+03	5,3885E+02	$-$
GWO	7,5482E+03	6,4215E+03	6,0602E+03	5,2792E+02	$-$
SOMA ATR	7,5445E+03	6,6365E+03	6,0597E+03	4,6357E+02	$-$
SOMA QI	1,1678E+04	6,6703E+03	6,1157E+03	2,2456E+03	$\approx$
SOMA ATB	8,0510E+03	6,6795E+03	6,0597E+03	5,4951E+02	$-$
UPSO	9,3878E+03	8,0164E+03	6,1547E+03	7,4587E+02	$-$
SCA	2,2801E+04	1,2289E+04	7,0162E+03	3,9761E+03	$-$

Tabulka 6: Výsledky hodnot účelové funkce algoritmů na funkci návrhu parametrů tlakového válce, kde $+$, $-$ a $\approx$ označují výsledky t-testu, tedy zda jsou na hladině významnosti 95% statisticky významně lepší, horší, či jsou srovnatelné.

Algoritmus	Min	Průměr	Max	STD	
PSODE	1,2665E-02	1,2665E-02	1,2665E-02	1,2000E-08	+
TLBO	N.A.	1,2666E-02	1,2665E-02	N.A.	
SOMA M3-QI	1,2770E-02	1,2684E-02	1,2665E-02	2,7232E-05	
SOMA M3-QI + MPC	1,3066E-02	1,2701E-02	1,2665E-02	7,5683E-05	−
HPSO	1,2719E-02	1,2707E-02	1,2665E-02	1,5800E-05	−
CPSO	1,2924E-02	1,2733E-02	1,2675E-02	5,2000E-04	−
GA4	1,2973E-02	1,2742E-02	1,2681E-02	5,9000E-05	−
GA3	1,2822E-02	1,2769E-02	1,2705E-02	3,9400E-05	−
$(\mu + \lambda)$ES	1,2670E-02	1,2769E-02	1,2665E-02	1,3570E-06	−
GWO	1,3112E-02	1,2776E-02	1,2671E-02	1,0333E-04	−
CGWO	1,3269E-02	1,2805E-02	1,2676E-02	1,6004E-04	−
SOMA ATA	1,7773E-02	1,2935E-02	1,2666E-02	5,0271E-04	−
CSA	1,3482E-02	1,2960E-02	1,2707E-02	1,8180E-04	−
SOMA ATAA	1,6379E-02	1,3194E-02	1,2665E-02	7,9980E-04	−
SOMA ATR	1,7709E-02	1,3241E-02	1,2665E-02	9,1990E-04	−
SOMA QI	2,3561E-02	1,3457E-02	1,2665E-02	3,0873E-03	−
SOMA T3A	1,4718E-02	1,3470E-02	1,2665E-02	5,6042E-04	−
SOMA ATB	1,7746E-02	1,4094E-02	1,2666E-02	1,3827E-03	−
SCA	3,5384E-02	1,7726E-02	1,2957E-02	4,9233E-03	−
UPSO	N.A.	2,2940E-02	1,3120E-02	7,2000E-03	−

Tabulka 7: Výsledky hodnot účelové funkce algoritmů na funkci návrhu parametrů svařovaných nosníků, kde +, − a ≈ označují výsledky t-testu, tedy zda jsou na hladině významnosti 95% statisticky významně lepší, horší, či jsou srovnatelné.

11.2 Výsledky CEC 2017

Výsledky dvou nových strategií SOMA M3-QI a SOMA M3-QI+MPC jsou srovnávány s algoritmy, jež byly představeny a soutěžily na samotné konferenci CEC 2017 [22]: CAL-SHADE [82], L-SHADE+UDE [81], UDE [80] a LSHADE44 [83], dále s hejnovými algoritmy GWO [20], CGWO [16], SCA [41], CSA [11], SOMA (všechny původní strategie), C-SOMA QI a C-SOMA T3A.
Nastavení ukončovacích podmínek, stejně jako způsob zaznamenávání a srovnávání výsledků, vychází z pravidel soutěže CEC 2017, tedy maximální počet ohodnocení účelové funkce je nastaven na $20000 * D$ a na každé funkci je provedeno 25 nezávislých běhů. Daný je také způsob ohodnocení výkonu algoritmů, určuje ho tzv. *Rank*, jehož výpočet je uveden v rovnici 50. Čím nižší ohodnocení *Rank* algoritmus získá, tím lepšího výsledku dosáhl.

$$Rank = \sum_{i=1}^{28} rank_i(průměr) + \sum_{i=1}^{28} rank_i(medián) \tag{50}$$

kde se dílčí ranky, pro každou funkci zvlášť, získávají následovně:

- Procedura určení *ranku* pro průměrné hodnoty se řídí pravidly:
  1. Algoritmy jsou ohodnoceny dle úspěšnosti běhů (SR)
  2. Poté dle průměrné míry porušení omezení
  3. Konečně dle průměrné hodnoty účelové funkce
- Procedura určení *ranku* pro hodnoty mediánu se řídí pravidly:
  1. Algoritmus, jehož medián neporušuje omezení účelové funkce je vždy lepší než ty, které je porušují
  2. Algoritmy, jejichž medián porušuje omezení účelové funkce jsou seřazeny dle míry porušení
  3. Algoritmy, jejichž medián neporušuje omezení účelové funkce jsou seřazeny dle hodnoty účelových funkcí

V tabulce 8 jsou uvedeny součty *ranků* na všech funkcích, získaných dle výše uvedených pravidel. Další tabulky potom prezentují hodnoty potřebné k výpočtu těchto ranků: Tabulky 9, 10, 11, 12, 13, 14, 15, 16, 17 a 18 prezentují průměrné výsledky na 10 dimenzích. Tabulky 29, 30, 31, 32, 33, 34, 35, 36, 37 a 38 průměrné výsledky na 30 dimenzích. *Průměr* v těchto tabulkách značí průměrné ohodnocení účelové funkce v případě, kdy algoritmus nalezl všechna řešení ve splnitelné oblasti, v opačném případě označuje průměrné porušení omezení. SR je označením úspěšnosti běhů uváděných v %, r značí *rank* algoritmu na dané funkci. Tabulky 19, 20, 21, 22, 23, 24, 25, 26, 27 a 28 prezentují výsledky mediánu na 10 dimenzích a tabulky 39, 40, 41, 42, 43, 44, 45, 46, 47 a 48 zase výsledky mediánů na 30 dimenzích. Hodnota *medián* zde označuje

medián ohodnocení účelové funkce v případě, že bylo nalezeno řešení ve splnitelné oblasti, v opačném případě je uvedena míra porušení omezení. K určení, zda je medián umístěn ve splnitelné oblasti slouží sloupec označený F (feasible), který může nabývat hodnot A − řešení je umístěno ve splnitelné oblasti a N − v opačném případě.

Dle metody ohodnocení algoritmu z CEC 2017 můžeme konstatovat, že algoritmy založené na principu diferenciální evoluce překonaly výkony všech hejnových. Při analýze 10 dimenzí však dokázala nově představená strategie SOMA M3-QI překonat výsledek algoritmu CAL-SHADE, ten ovšem zase ukázal lepších hodnot na 30 dimenzích a v celkovém pořadí se byl určen jako vhodnější. Nově představená strategie SOMA M3-QI ovšem dokázala na většině funkcí nalézat lepších výsledků, než ostatní hejnové algoritmy. Nejmarkantnějších zlepšení lze pozorovat u funkcí, kde nebylo složité nalézt globální optimum, ale rozhodovalo nalezení přesnější hodnoty účelové funkce, například $f_{CEC_1}, f_{CEC_2}, f_{CEC_5}$, což je dáno zejména schopností adaptace a také využitím kvadratické interpolační metody, která právě k nalezení lokálního optima slouží. Tato vlastnost se ovšem projevuje negativně na jiných funkcích, kde často k nalezení optima nedošlo, například $f_{CEC_9}, f_{CEC_{14}}, f_{CEC_{20}}$. Adaptivní změna způsobu generování chaotických čísel umožnila algoritmu SOMA M3-QI nalézat častěji, než ostatní algoritmy, splnitelná řešení, dobře to demonstrují výsledky na funkcích $f_{CEC_{13}}, f_{CEC_{15}}, f_{CEC_{18}}, f_{CEC_{23}}, f_{CEC_{27}}$. Konečné pořadí algoritmů dle pravidel CEC 2017 je: UDE > L-SHADE44 > L-SHADE + UDE > CAL-SHADE > SOMA M3-QI > SOMA T3A > SOMA M3-QI+MPC > CSA > SOMA-QI > SOMA ATB > SOMA ATA > SOMA ATAA > SOMA ATR > CGWO > GWO > SCA.

	$D = 10$		$D = 30$			
Algoritmus	Medián	Průměr	Medián	Průměr	$\sum$	Výsledný *rank*
UDE	94	122	87	116	419	1
L-SHADE44	113	132	96	98	439	2
L-SHADE+UDE	108	118	113	117	456	3
CAL-SHADE	160	189	161	138	648	4
SOMA M3-QI	171	174	178	175	698	5
SOMA T3A	175	185	172	176	708	6
SOMA M3-QI+MPC	187	170	171	193	721	7
CSA	193	196	236	210	835	8
SOMA-QI	221	244	194	188	847	9
SOMA ATB	222	255	249	257	983	10
SOMA ATA	249	269	247	266	1031	11
SOMA ATAA	279	282	284	329	1174	12
SOMA ATR	304	303	303	334	1244	13
CGWO	369	355	383	374	1481	14
GWO	386	379	389	384	1538	15
SCA	444	413	443	415	1715	16

Tabulka 8: Součty ranků průměrů a mediánů algoritmů dle pravidel CEC 2017 na funkcích 1-28 CEC 2017 10 a 30 dimenzí

Algoritmus	f_{CEC_1} Průměr	SR	r	f_{CEC_2} Průměr	SR	r
CAL-SHADE	0,00E+00	100	1	0,00E+00	100	1
L-SHADE44	0,00E+00	100	1	0,00E+00	100	1
UDE	0,00E+00	100	1	0,00E+00	100	1
L-SHADE+UDE	0,00E+00	100	1	0,00E+00	100	1
CGWO	4,66E+02	100	14	6,02E+02	100	14
CSA	1,42E−08	100	9	9,46E−09	100	9
GWO	8,29E+02	100	15	7,79E+02	100	15
SCA	7,01E+03	100	16	6,27E+03	100	16
SOMA ATA	1,07E+01	100	13	1,12E+01	100	12
SOMA ATAA	7,68E+00	100	12	1,85E+01	100	13
SOMA ATB	2,61E−13	100	7	3,26E−15	100	6
SOMA ATR	1,10E+00	100	11	2,42E+00	100	11
SOMA M3-QI	3,97E−22	100	5	1,79E−21	100	5
S M3-QI+MPC	8,87E−10	100	8	2,43E−11	100	8
SOMA-QI	1,21E−19	100	6	1,92E−13	100	7
SOMA T3A	1,75E−05	100	10	4,21E−07	100	10

Tabulka 9: Výsledky algoritmů na funkcích 1−2 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	f_{CEC_3} Průměr	SR	r	f_{CEC_4} Průměr	SR	r	f_{CEC_5} Průměr	SR	r
CAL-SHADE	3,87E+01	100	8	9,57E−01	100	5	5,37E−03	96	1
L-SHADE44	1,36E+01	100	1	0,00E+00	100	1	3,23E−02	0	10
UDE	2,44E+01	100	5	1,59E−01	100	3	6,80E−03	96	2
L-SHADE+UDE	1,44E+01	100	3	0,00E+00	100	1	3,77E−02	0	11
CGWO	6,15E+01	100	10	3,87E+03	100	12	2,54E−01	0	14
CSA	1,48E+01	100	4	2,50E+00	100	8	1,06E−01	4	9
GWO	6,77E+01	100	11	6,93E+03	100	13	2,23E−01	0	12
SCA	1,15E+02	100	16	3,63E+01	56	16	1,58E+00	0	16
SOMA ATA	4,78E+01	100	9	2,65E+01	100	10	5,15E−02	52	6
SOMA ATAA	9,26E+01	100	14	3,18E+01	100	11	2,04E−01	16	8
SOMA ATB	8,22E+01	100	13	1,98E−02	96	15	5,64E−02	84	5
SOMA ATR	9,68E+01	100	15	2,02E+04	100	14	3,36E−01	0	15
SOMA M3-QI	2,65E+01	100	7	6,41E−01	100	4	2,88E−03	88	3
S M3-QI+MPC	2,53E+01	100	6	1,80E+00	100	6	5,28E−01	88	4
SOMA-QI	6,93E+01	100	12	4,59E+00	100	9	1,05E−01	48	7
SOMA T3A	1,38E+01	100	2	2,15E+00	100	7	2,45E−01	0	13

Tabulka 10: Výsledky algoritmů na funkcích 3−5 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	f_{CEC_6} Průměr	SR	r	f_{CEC_7} Průměr	SR	r	f_{CEC_8} Průměr	SR	r
CAL-SHADE	−1,35E−03	100	5	1,25E−01	100	4	−5,10E−04	100	1
L-SHADE44	−1,35E−03	100	1	−4,97E−03	100	3	−5,10E−04	100	4
UDE	−1,35E−03	100	1	−4,98E−03	100	2	−5,10E−04	100	2
L-SHADE+UDE	−1,35E−03	100	3	−4,98E−03	100	1	−5,10E−04	100	3
CGWO	3,35E+01	0	14	4,10E+00	12	14	1,97E+03	0	14
CSA	−1,33E−03	100	6	6,16E−01	100	5	−5,05E−04	100	8
GWO	4,26E+01	0	15	1,50E+01	12	15	2,91E+03	0	15
SCA	1,56E+02	0	16	1,77E+02	0	16	9,09E+03	0	16
SOMA ATA	6,48E+00	0	13	3,47E−05	96	11	7,17E−02	4	13
SOMA ATAA	6,77E+00	8	11	1,71E+00	64	13	3,23E−03	64	12
SOMA ATB	1,93E−05	100	9	9,20E+00	100	10	−2,48E−04	100	10
SOMA ATR	5,53E+00	0	12	2,18E+00	76	12	1,66E−02	72	11
SOMA M3-QI	−1,35E−03	100	4	5,63E+00	100	8	−5,10E−04	100	5
S M3-QI+MPC	−1,28E−03	100	7	2,66E+00	100	6	−5,09E−04	100	7
SOMA-QI	1,69E−04	100	10	7,70E+00	100	9	−3,89E−04	100	9
SOMA T3A	−1,19E−03	100	8	3,15E+00	100	7	−5,09E−04	100	6

Tabulka 11: Výsledky algoritmů na funkcích 6−8 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	f_{CEC_9} Průměr	SR	r	$f_{CEC_{10}}$ Průměr	SR	r	$f_{CEC_{11}}$ Průměr	SR	r
CAL-SHADE	1,25E−01	100	4	−5,10E−04	100	1	−1,56E−01	100	3
L-SHADE44	−4,97E−03	100	3	−5,10E−04	100	4	−1,69E−01	100	2
UDE	−4,98E−03	100	2	−5,10E−04	100	2	3,00E−03	0	7
L-SHADE+UDE	−4,98E−03	100	1	−5,10E−04	100	3	−1,69E−01	100	1
CGWO	4,10E+00	12	14	1,97E+03	0	14	2,15E+01	0	14
CSA	6,16E−01	100	5	−5,05E−04	100	8	8,55E−03	0	8
GWO	1,50E+01	12	15	2,91E+03	0	15	3,37E+01	0	15
SCA	1,77E+02	0	16	9,09E+03	0	16	6,79E+01	0	16
SOMA ATA	3,47E−05	96	11	7,17E−02	4	13	1,14E+01	0	12
SOMA ATAA	1,71E+00	64	13	3,23E−03	64	12	9,73E+00	0	11
SOMA ATB	9,20E+00	100	10	−2,48E−04	100	10	2,44E−03	4	6
SOMA ATR	2,18E+00	76	12	1,66E−02	72	11	1,69E+01	0	13
SOMA M3-QI	5,63E+00	100	8	−5,10E−04	100	5	1,06E−09	36	4
S M3-QI+MPC	2,66E+00	100	6	−5,09E−04	100	7	1,21E−01	0	10
SOMA-QI	7,70E+00	100	9	−3,89E−04	100	9	2,51E−04	4	5
SOMA T3A	3,15E+00	100	7	−5,09E−04	100	6	9,23E−02	0	9

Tabulka 12: Výsledky algoritmů na funkcích 9−11 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{12}}$ Průměr	SR	r	$f_{CEC_{13}}$ Průměr	SR	r	$f_{CEC_{14}}$ Průměr	SR	r
CAL-SHADE	3,99E+00	100	1	1,12E+00	100	8	2,64E+00	100	2
L-SHADE44	3,99E+00	100	8	0,00E+00	100	1	2,88E+00	100	6
UDE	3,99E+00	100	1	3,19E−01	100	5	2,47E+00	100	1
L-SHADE+UDE	3,99E+00	100	3	0,00E+00	100	1	3,00E+00	100	7
CGWO	1,62E+02	32	14	1,08E+02	24	11	4,25E+02	4	14
CSA	3,99E+00	100	5	3,27E−01	100	7	3,33E+00	100	9
GWO	2,89E+02	24	15	1,68E+02	8	14	4,93E+02	0	15
SCA	1,09E+03	0	16	7,51E+02	0	16	2,12E+03	0	16
SOMA ATA	1,17E+01	100	10	2,14E+01	52	9	4,82E−01	40	11
SOMA ATAA	2,82E+01	100	12	9,99E+01	16	13	6,23E−01	24	13
SOMA ATB	1,47E+01	100	11	8,41E+01	0	15	2,74E+00	100	3
SOMA ATR	3,14E+01	100	13	8,76E+01	20	12	8,57E−01	28	12
SOMA M3-QI	3,99E+00	100	7	3,19E−01	100	6	3,26E+00	100	8
S M3-QI+MPC	3,99E+00	100	4	2,76E−02	100	4	3,45E+00	100	10
SOMA-QI	8,81E+00	100	9	3,27E+01	36	10	2,79E+00	100	4
SOMA T3A	3,99E+00	100	6	1,09E−03	100	3	2,79E+00	100	5

Tabulka 13: Výsledky algoritmů na funkcích 12−14 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{15}}$ Průměr	SR	r	$f_{CEC_{16}}$ Průměr	SR	r	$f_{CEC_{17}}$ Průměr	SR	r
CAL-SHADE	1,51E+01	68	13	1,30E−02	60	14	5,54E+00	0	11
L-SHADE44	9,18E−05	48	15	4,07E+01	100	4	5,30E+00	0	7
UDE	9,56E−05	88	12	6,09E+00	100	1	5,38E+00	0	8
L-SHADE+UDE	1,13E+01	100	3	4,04E+01	100	3	5,22E+00	0	5
CGWO	4,18E+01	92	11	5,55E+01	72	13	1,57E+02	0	12
CSA	7,13E+00	100	1	5,32E+01	100	5	4,85E+00	0	1
GWO	1,24E+02	60	14	6,87E+01	48	15	2,31E+02	0	13
SCA	5,18E+02	0	16	7,19E+02	0	16	1,15E+03	0	14
SOMA ATA	1,78E+01	100	8	6,18E+01	100	8	5,22E+00	0	4
SOMA ATAA	2,97E−02	92	9	6,83E+01	100	10	5,29E+00	0	6
SOMA ATB	1,76E+01	100	7	7,02E+01	100	11	5,50E+00	0	10
SOMA ATR	3,12E−02	92	10	6,71E+01	100	9	5,10E+00	0	3
SOMA M3-QI	1,43E+01	100	5	5,65E+01	100	7	5,42E+00	0	9
S M3-QI+MPC	8,14E+00	100	2	5,58E+01	100	6	4,94E+00	0	2
SOMA-QI	1,69E+01	100	6	7,28E+01	100	12	4,39E+04	0	16
SOMA T3A	1,24E+01	100	4	3,90E+01	100	2	1,98E+04	0	15

Tabulka 14: Výsledky algoritmů na funkcích 15−17 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{18}}$ Průměr	SR	r	$f_{CEC_{19}}$ Průměr	SR	r	$f_{CEC_{20}}$ Průměr	SR	r
CAL-SHADE	1,40E+08	0	16	4,42E+03	0	1	8,69E−01	100	10
L-SHADE44	2,38E+06	0	12	6,63E+03	0	3	1,93E−01	100	1
UDE	8,29E+03	0	10	6,63E+03	0	9	1,50E+00	100	12
L-SHADE+UDE	8,23E+06	0	14	6,63E+03	0	5	4,16E−01	100	2
CGWO	1,91E+06	0	11	6,66E+03	0	16	1,03E+00	100	11
CSA	9,78E+01	100	2	6,63E+03	0	7	1,51E+00	100	14
GWO	7,28E+06	0	13	6,66E+03	0	15	8,50E−01	100	9
SCA	3,35E+07	0	15	6,49E+03	0	2	1,82E+00	100	16
SOMA ATA	4,23E+02	0	9	6,63E+03	0	11	6,37E−01	100	5
SOMA ATAA	1,51E+05	16	7	6,63E+03	0	10	7,28E−01	100	7
SOMA ATB	1,01E+00	96	4	6,64E+03	0	13	8,26E−01	100	8
SOMA ATR	3,00E+03	4	8	6,63E+03	0	12	7,26E−01	100	6
SOMA M3-QI	9,29E+01	100	1	6,63E+03	0	6	1,66E+00	100	15
S M3-QI+MPC	3,74E+01	96	6	6,63E+03	0	4	1,50E+00	100	13
SOMA-QI	1,01E+00	96	5	6,64E+03	0	14	6,36E−01	100	4
SOMA T3A	2,58E−05	96	3	6,63E+03	0	8	5,60E−01	100	3

Tabulka 15: Výsledky algoritmů na funkcích 18−20 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{21}}$ Průměr	SR	r	$f_{CEC_{22}}$ Průměr	SR	r	$f_{CEC_{23}}$ Průměr	SR	r
CAL-SHADE	6,86E+00	100	8	6,67E+00	100	6	2,53E+00	100	2
L-SHADE44	3,99E+00	100	3	6,38E−01	100	2	2,54E+00	100	3
UDE	4,41E+00	100	6	5,31E+00	100	4	2,47E+00	100	1
L-SHADE+UDE	3,99E+00	100	1	1,59E−01	100	1	3,02E+00	100	5
CGWO	6,02E+02	8	14	4,45E+02	12	11	1,26E+03	0	15
CSA	3,99E+00	100	2	6,18E+00	100	5	3,56E+00	100	10
GWO	9,40E+02	4	15	6,95E+02	16	10	1,25E+03	0	14
SCA	3,89E+03	0	16	2,39E+03	0	16	6,15E+03	0	16
SOMA ATA	8,15E+00	100	9	7,34E+01	8	12	1,10E−01	76	13
SOMA ATAA	1,56E+01	100	13	2,64E+02	0	15	3,63E−02	92	12
SOMA ATB	1,11E+01	100	11	1,68E+02	4	13	2,65E+00	100	4
SOMA ATR	1,21E+01	100	12	2,40E+02	0	14	2,53E−02	92	11
SOMA M3-QI	4,45E+00	100	7	1,06E+00	84	8	3,33E+00	100	7
S M3-QI+MPC	3,99E+00	100	4	1,03E+00	100	3	3,43E+00	100	9
SOMA-QI	9,76E+00	100	10	1,33E+02	16	9	3,18E+00	100	6
SOMA T3A	4,29E+00	100	5	1,23E+01	100	7	3,37E+00	100	8

Tabulka 16: Výsledky algoritmů na funkcích 21−23 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{24}}$ Průměr	SR	r	$f_{CEC_{25}}$ Průměr	SR	r	$f_{CEC_{26}}$ Průměr	SR	r
CAL-SHADE	1,01E−01	40	13	1,22E−02	60	13	5,50E+00	0	12
L-SHADE44	8,64E+00	100	4	3,87E+01	100	3	5,06E+00	0	3
UDE	5,50E+00	100	2	6,28E+00	100	1	5,42E+00	0	6
L-SHADE+UDE	8,77E+00	100	5	3,77E+01	100	2	5,05E+00	0	2
CGWO	4,85E+02	20	14	5,48E+02	32	14	7,98E+02	0	14
CSA	2,25E−05	72	12	5,54E+01	100	5	5,02E+00	0	1
GWO	7,36E+02	8	15	6,58E+02	28	15	7,28E+02	0	13
SCA	3,30E+03	0	16	3,45E+03	0	16	3,33E+03	0	15
SOMA ATA	1,50E+01	100	7	7,48E+01	100	9	5,46E+00	0	7
SOMA ATAA	1,63E+01	100	9	7,84E+01	100	10	5,50E+00	0	8
SOMA ATB	1,69E+01	100	11	7,85E+01	100	11	5,50E+00	0	8
SOMA ATR	1,67E+01	100	10	7,90E+01	100	12	5,50E+00	0	8
SOMA M3-QI	1,19E+01	100	6	5,69E+01	100	6	5,50E+00	0	8
S M3-QI+MPC	5,42E−06	100	1	5,89E+01	100	7	5,08E+00	0	4
SOMA-QI	1,54E+01	100	8	7,43E+01	100	8	4,40E+04	0	16
SOMA T3A	8,14E+00	100	3	5,01E+01	100	4	5,18E+00	0	5

Tabulka 17: Výsledky algoritmů na funkcích 24−26 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{27}}$ Průměr	SR	r	$f_{CEC_{28}}$ Průměr	SR	r
CAL-SHADE	1,26E+07	0	14	4,44E+03	0	1
L-SHADE44	4,13E+05	0	11	6,65E+03	0	12
UDE	1,05E+04	0	8	6,64E+03	0	9
L-SHADE+UDE	4,05E+07	0	15	6,64E+03	0	10
CGWO	9,57E+06	0	12	6,35E+03	0	2
CSA	5,93E+02	64	3	6,65E+03	0	11
GWO	1,00E+07	0	13	6,58E+03	0	7
SCA	3,94E+08	0	16	6,40E+03	0	3
SOMA ATA	3,78E+04	0	10	6,66E+03	0	14
SOMA ATAA	3,85E+03	0	7	6,44E+03	0	4
SOMA ATB	7,61E+02	64	4	6,67E+03	0	15
SOMA ATR	2,88E+04	0	9	6,67E+03	0	16
SOMA M3-QI	3,83E+02	84	1	6,57E+03	0	6
S M3-QI+MPC	1,03E−02	68	2	6,48E+03	0	5
SOMA-QI	4,59E+02	48	5	6,58E+03	0	8
SOMA T3A	4,53E+02	16	6	6,66E+03	0	13

Tabulka 18: Výsledky algoritmů na funkci 27 a 28 CEC 2017 D = 10 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	f_{CEC_1} Medián	F	r	f_{CEC_2} Medián	F	r
CAL-SHADE	0,00E+00	A	1	0,00E+00	A	1
L-SHADE44	0,00E+00	A	1	0,00E+00	A	1
UDE	0,00E+00	A	1	0,00E+00	A	1
L-SHADE+UDE	0,00E+00	A	1	0,00E+00	A	1
CGWO	3,35E+02	A	14	5,88E+02	A	15
CSA	1,36E−08	A	8	6,25E−09	A	8
GWO	5,35E+02	A	15	5,24E+02	A	14
SCA	6,51E+03	A	16	5,35E+03	A	16
SOMA ATA	7,91E+00	A	13	4,32E+00	A	13
SOMA ATAA	1,00E−01	A	11	2,32E−01	A	12
SOMA ATB	5,16E−21	A	7	1,19E−19	A	7
SOMA ATR	1,20E−01	A	12	2,13E−01	A	11
SOMA M3-QI	1,19E−22	A	9	1,85E−23	A	9
S M3-QI+MPC	6,38E−07	A	5	1,78E−07	A	5
SOMA-QI	3,83E−22	A	10	2,86E−22	A	10
SOMA T3A	1,24E−07	A	6	2,99E−08	A	6

Tabulka 19: Výsledky algoritmů na funkcích 1−2 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	f_{CEC_3} Medián	F	r	f_{CEC_4} Medián	F	r	f_{CEC_5} Medián	F	r
CAL-SHADE	1,03E−04	N	10	3,58E+01	A	8	0,00E+00	A	1
L-SHADE44	2,11E+04	A	2	1,36E+01	A	4	0,00E+00	A	1
UDE	7,47E+01	A	1	1,89E+01	A	5	0,00E+00	A	1
L-SHADE+UDE	2,26E+05	A	8	1,36E+01	A	1	0,00E+00	A	1
CGWO	5,41E−04	N	14	6,22E+01	A	10	3,32E+02	A	14
CSA	5,79E−04	N	15	1,36E+01	A	2	2,43E+00	A	9
GWO	6,69E−04	N	16	6,82E+01	A	12	1,66E+03	A	15
SCA	1,85E−04	N	12	1,18E+02	A	16	1,56E+04	A	16
SOMA ATA	2,11E+04	A	3	4,48E+01	A	9	7,88E+00	A	13
SOMA ATAA	5,36E+04	A	5	7,96E+01	A	15	7,11E+00	A	11
SOMA ATB	6,70E+04	A	6	7,46E+01	A	13	8,60E−01	A	7
SOMA ATR	3,22E+04	A	4	7,66E+01	A	14	7,18E+00	A	12
SOMA M3-QI	1,65E−04	N	9	2,55E+01	A	3	3,35E−03	A	8
S M3-QI+MPC	3,28E−04	N	11	3,46E+01	A	6	2,55E+00	A	5
SOMA-QI	9,21E+04	A	13	6,67E+01	A	7	6,27E−01	A	10
SOMA T3A	8,05E−05	N	7	1,36E+01	A	11	1,73E+00	A	6

Tabulka 20: Výsledky algoritmů na funkcích 3−5 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	f_{CEC_6} Medián	F	r	f_{CEC_7} Medián	F	r	f_{CEC_8} Medián	F	r
CAL-SHADE	3,08E+02	A	4	−6,52E+01	A	3	−1,35E−03	A	5
L-SHADE44	2,97E−02	N	8	1,28E+01	A	5	−1,35E−03	A	1
UDE	7,69E+01	A	1	−6,99E+01	A	1	−1,35E−03	A	1
L-SHADE+UDE	3,83E−02	N	9	−2,68E+01	A	4	−1,35E−03	A	3
CGWO	4,33E−01	N	12	2,06E+00	N	9	3,31E+01	N	14
CSA	2,64E−01	N	11	4,33E+00	N	11	−1,33E−03	A	6
GWO	4,99E−01	N	13	1,21E−02	N	8	4,24E+01	N	15
SCA	4,49E+00	N	16	8,24E+01	N	16	1,61E+02	N	16
SOMA ATA	1,58E+02	A	2	5,34E+00	N	12	9,94E−01	N	13
SOMA ATAA	1,65E−01	N	10	1,94E+01	N	13	2,44E−02	N	12
SOMA ATB	9,16E+02	A	6	−6,83E+01	A	2	−4,75E−04	A	10
SOMA ATR	6,20E−01	N	14	2,18E+01	N	14	6,29E−03	N	11
SOMA M3-QI	3,94E+02	A	15	2,52E+00	N	6	−1,35E−03	A	8
S M3-QI+MPC	1,72E+02	A	5	2,60E−03	N	10	−1,33E−03	A	4
SOMA-QI	8,10E−04	N	3	3,57E+01	N	7	−7,97E−04	A	7
SOMA T3A	7,62E−01	N	7	4,73E−04	N	15	−1,31E−03	A	9

Tabulka 21: Výsledky algoritmů na funkcích 6−8 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	f_{CEC_9} Medián	F	r	$f_{CEC_{10}}$ Medián	F	r	$f_{CEC_{11}}$ Medián	F	r
CAL-SHADE	−4,98E−03	A	4	−5,10E−04	A	1	−1,69E−01	A	2
L-SHADE44	−4,98E−03	A	1	−5,10E−04	A	4	−1,69E−01	A	2
UDE	−4,98E−03	A	3	−5,10E−04	A	2	3,00E−03	N	10
L-SHADE+UDE	−4,98E−03	A	2	−5,10E−04	A	3	−1,69E−01	A	1
CGWO	3,14E−03	N	14	2,15E+03	N	14	1,16E+01	N	14
CSA	−4,97E−03	A	5	−5,05E−04	A	8	1,39E−03	N	8
GWO	4,29E+00	N	15	2,73E+03	N	15	2,15E+01	N	15
SCA	1,43E+02	N	16	9,80E+03	N	16	6,80E+01	N	16
SOMA ATA	9,38E+00	A	11	3,75E−03	N	13	4,98E+00	N	12
SOMA ATAA	7,43E+00	A	10	1,81E−03	A	12	3,36E+00	N	11
SOMA ATB	9,44E+00	A	12	−3,11E−04	A	10	4,71E−04	N	7
SOMA ATR	1,20E+01	A	13	1,78E−03	A	11	5,62E+00	N	13
SOMA M3-QI	5,37E+00	A	6	−5,10E−04	A	6	5,03E−10	N	9
S M3-QI+MPC	4,90E+00	A	8	−5,07E−04	A	5	2,89E−05	N	4
SOMA-QI	5,41E+00	A	7	−4,48E−04	A	7	1,91E−04	N	5
SOMA T3A	2,29E+00	A	9	−5,09E−04	A	9	1,70E−03	N	6

Tabulka 22: Výsledky algoritmů na funkcích 9−11 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{12}}$ Medián	F	r	$f_{CEC_{13}}$ Medián	F	r	$f_{CEC_{14}}$ Medián	F	r
CAL-SHADE	3,99E+00	A	2	0,00E+00	A	1	2,63E+00	A	2
L-SHADE44	3,99E+00	A	11	0,00E+00	A	1	2,91E+00	A	6
UDE	3,99E+00	A	1	0,00E+00	A	1	2,38E+00	A	1
L-SHADE+UDE	3,99E+00	A	3	0,00E+00	A	1	3,01E+00	A	7
CGWO	1,35E+02	N	14	1,09E+02	N	14	2,56E+02	N	14
CSA	3,99E+00	A	8	1,33E−01	A	8	3,32E+00	A	9
GWO	1,76E+02	N	15	1,97E+02	N	15	4,75E+02	N	15
SCA	1,08E+03	N	16	1,20E+03	N	16	2,00E+03	N	16
SOMA ATA	3,99E+00	A	10	3,59E−01	A	9	1,22E−01	N	11
SOMA ATAA	2,29E+01	A	12	5,75E+01	N	12	5,81E−01	N	12
SOMA ATB	3,99E+00	A	7	7,34E+01	N	13	2,66E+00	A	3
SOMA ATR	2,38E+01	A	13	5,04E+01	N	11	7,40E−01	N	13
SOMA M3-QI	3,99E+00	A	6	1,66E−05	A	5	3,30E+00	A	5
S M3-QI+MPC	3,99E+00	A	4	1,76E−02	A	6	3,46E+00	A	8
SOMA-QI	3,99E+00	A	9	3,51E+01	N	7	2,67E+00	A	10
SOMA T3A	3,99E+00	A	5	3,12E−06	A	10	2,79E+00	A	4

Tabulka 23: Výsledky algoritmů na funkcích 12−14 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{15}}$ Medián	F	r	$f_{CEC_{16}}$ Medián	F	r	$f_{CEC_{17}}$ Medián	F	r
CAL-SHADE	1,49E+01	A	8	5,10E−05	N	14	5,50E+00	N	13
L-SHADE44	5,79E−05	N	15	3,93E+01	A	3	5,50E+00	N	4
UDE	5,50E+00	A	1	6,28E+00	A	1	5,50E+00	N	4
L-SHADE+UDE	1,18E+01	A	3	3,93E+01	A	4	5,50E+00	N	4
CGWO	1,49E+01	A	10	6,91E+01	A	12	1,16E+02	N	15
CSA	5,50E+00	A	2	5,18E+01	A	6	4,57E+00	N	3
GWO	1,18E+01	A	6	5,32E−05	N	15	6,03E+01	N	14
SCA	5,05E+02	N	16	5,01E+02	N	16	1,13E+03	N	16
SOMA ATA	1,81E+01	A	11	6,28E+01	A	8	5,50E+00	N	4
SOMA ATAA	1,49E+01	A	9	6,91E+01	A	9	5,50E+00	N	4
SOMA ATB	1,81E+01	A	11	6,91E+01	A	11	5,50E+00	N	4
SOMA ATR	2,12E+01	A	14	6,91E+01	A	9	5,50E+00	N	4
SOMA M3-QI	1,18E+01	A	7	5,65E+01	A	2	5,50E+00	N	1
S M3-QI+MPC	1,18E+01	A	5	4,56E+01	A	7	4,50E+00	N	4
SOMA-QI	1,81E+01	A	3	7,54E+01	A	5	5,50E+00	N	1
SOMA T3A	1,18E+01	A	11	3,77E+01	A	13	4,50E+00	N	4

Tabulka 24: Výsledky algoritmů na funkcích 15−17 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{18}}$ Medián	F	r	$f_{CEC_{19}}$ Medián	F	r	$f_{CEC_{20}}$ Medián	F	r
CAL-SHADE	9,72E+05	N	14	4,42E+03	N	1	7,58E−01	A	8
L-SHADE44	3,33E+03	N	10	6,63E+03	N	2	1,91E−01	A	1
UDE	5,45E+03	N	11	6,63E+03	N	12	1,38E+00	A	12
L-SHADE+UDE	8,23E+04	N	12	6,63E+03	N	4	4,01E−01	A	2
CGWO	4,46E+05	N	13	6,66E+03	N	14	9,91E−01	A	11
CSA	3,66E+01	A	2	6,63E+03	N	9	1,55E+00	A	13
GWO	3,20E+06	N	15	6,66E+03	N	15	8,23E−01	A	10
SCA	4,92E+07	N	16	6,67E+03	N	16	1,79E+00	A	16
SOMA ATA	3,80E+01	N	7	6,63E+03	N	4	5,92E−01	A	3
SOMA ATAA	1,62E+02	N	8	6,63E+03	N	8	6,92E−01	A	6
SOMA ATB	4,16E+01	A	4	6,63E+03	N	11	8,22E−01	A	9
SOMA ATR	7,91E+02	N	9	6,63E+03	N	10	7,20E−01	A	7
SOMA M3-QI	3,66E+01	A	6	6,63E+03	N	3	1,71E+00	A	4
S M3-QI+MPC	4,09E+01	A	1	6,63E+03	N	7	1,69E+00	A	15
SOMA-QI	4,34E+01	A	3	6,64E+03	N	4	6,05E−01	A	14
SOMA T3A	4,34E+02	A	5	6,63E+03	N	13	6,03E−01	A	5

Tabulka 25: Výsledky algoritmů na funkcích 18−20 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{21}}$ Medián	F	r	$f_{CEC_{22}}$ Medián	F	r	$f_{CEC_{23}}$ Medián	F	r
CAL-SHADE	3,99E+00	A	3	0,00E+00	A	1	2,63E+00	A	3
L-SHADE44	3,99E+00	A	1	3,49E−27	A	3	2,40E+00	A	2
UDE	3,99E+00	A	1	3,49E−27	A	4	2,38E+00	A	1
L-SHADE+UDE	3,99E+00	A	4	0,00E+00	A	1	3,06E+00	A	5
CGWO	4,72E+02	N	14	2,04E+02	N	13	1,18E+03	N	15
CSA	3,99E+00	A	7	1,43E+00	A	7	3,66E+00	A	11
GWO	7,75E+02	N	15	1,33E+03	N	15	2,91E+02	N	14
SCA	3,91E+03	N	16	3,44E+03	N	16	5,75E+03	N	16
SOMA ATA	4,12E+00	A	10	4,75E+01	N	9	3,77E+00	A	13
SOMA ATAA	1,48E+01	A	13	1,97E+02	N	12	3,64E+00	A	10
SOMA ATB	3,99E+00	A	5	1,86E+02	N	11	2,64E+00	A	4
SOMA ATR	1,46E+01	A	12	2,96E+02	N	14	3,71E+00	A	12
SOMA M3-QI	3,99E+00	A	11	1,90E−01	A	8	3,35E+00	A	8
S M3-QI+MPC	3,99E+00	A	8	8,27E−01	A	5	3,54E+00	A	7
SOMA-QI	3,99E+00	A	9	5,40E+01	N	6	3,19E+00	A	9
SOMA T3A	4,17E+00	A	6	2,87E+00	A	10	3,40E+00	A	6

Tabulka 26: Výsledky algoritmů na funkcích 21−23 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{24}}$ Medián	F	r	$f_{CEC_{25}}$ Medián	F	r	$f_{CEC_{26}}$ Medián	F	r
CAL-SHADE	3,99E+00	A	3	0,00E+00	A	1	2,63E+00	A	3
L-SHADE44	3,99E+00	A	1	3,49E−27	A	3	2,40E+00	A	2
UDE	3,99E+00	A	1	3,49E−27	A	4	2,38E+00	A	1
L-SHADE+UDE	3,99E+00	A	4	0,00E+00	A	1	3,06E+00	A	5
CGWO	4,72E+02	N	14	2,04E+02	N	13	1,18E+03	N	15
CSA	3,99E+00	A	7	1,43E+00	A	7	3,66E+00	A	11
GWO	7,75E+02	N	15	1,33E+03	N	15	2,91E+02	N	14
SCA	3,91E+03	N	16	3,44E+03	N	16	5,75E+03	N	16
SOMA ATA	4,12E+00	A	10	4,75E+01	N	9	3,77E+00	A	13
SOMA ATAA	1,48E+01	A	13	1,97E+02	N	12	3,64E+00	A	10
SOMA ATB	3,99E+00	A	5	1,86E+02	N	11	2,64E+00	A	4
SOMA ATR	1,46E+01	A	12	2,96E+02	N	14	3,71E+00	A	12
SOMA M3-QI	3,99E+00	A	11	1,90E−01	A	8	3,35E+00	A	8
S M3-QI+MPC	3,99E+00	A	8	8,27E−01	A	5	3,54E+00	A	7
SOMA-QI	3,99E+00	A	9	5,40E+01	N	6	3,19E+00	A	9
SOMA T3A	4,17E+00	A	6	2,87E+00	A	10	3,40E+00	A	6

Tabulka 27: Výsledky algoritmů na funkcích 24−26 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{27}}$ Medián	F	r	$f_{CEC_{28}}$ Medián	F	r
CAL-SHADE	1,96E+06	N	15	4,44E+03	N	1
L-SHADE44	1,03E+04	N	11	6,65E+03	N	6
UDE	8,05E+03	N	10	6,63E+03	N	3
L-SHADE+UDE	3,68E+05	N	12	6,63E+03	N	2
CGWO	5,15E+05	N	14	6,66E+03	N	10
CSA	3,66E+01	A	3	6,64E+03	N	4
GWO	4,75E+05	N	13	6,66E+03	N	11
SCA	3,46E+08	N	16	6,67E+03	N	16
SOMA ATA	2,54E+03	N	8	6,66E+03	N	8
SOMA ATAA	4,85E+03	N	9	6,66E+03	N	12
SOMA ATB	3,66E+01	A	4	6,67E+03	N	13
SOMA ATR	1,22E+03	N	7	6,67E+03	N	15
SOMA M3-QI	3,66E+01	A	6	6,66E+03	N	7
S M3-QI+MPC	3,66E+01	A	1	6,65E+03	N	9
SOMA-QI	8,46E−02	N	2	6,67E+03	N	5
SOMA T3A	6,01E−01	N	5	6,66E+03	N	14

Tabulka 28: Výsledky algoritmů na funkci 27 a 28 CEC 2017 D = 10 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	f_{CEC_1} Průměr	SR	r	f_{CEC_2} Průměr	SR	r
CAL-SHADE	9,63E−29	100	4	8,66E−29	100	4
L-SHADE44	7,23E−30	100	2	4,74E−30	100	2
L-SHADE+UDE	0,00E+00	100	1	0,00E+00	100	1
UDE	7,34E−29	100	3	7,39E−29	100	3
CGWO	5,38E+03	100	14	7,52E+03	100	15
CSA	1,66E+01	100	8	8,36E+00	100	8
GWO	6,13E+03	100	15	7,30E+03	100	14
SCA	3,67E+04	100	16	3,91E+04	100	16
SOMA ATA	1,42E+03	100	12	1,14E+03	100	11
SOMA ATAA	7,24E+02	100	11	1,45E+03	100	12
SOMA ATB	2,31E+02	100	9	2,88E+02	100	10
SOMA ATR	2,52E+03	100	13	2,33E+03	100	13
SOMA QI	5,00E+00	100	7	2,31E−01	100	7
SOMA T3A	6,65E−02	100	6	2,15E−02	100	6
SOMA M3-QI	2,34E−03	100	5	1,47E−03	100	5
S M3-QI+MPC	3,95E+02	100	10	1,40E+02	100	9

Tabulka 29: Výsledky algoritmů na funkcích 1−2 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	f_{CEC_3} Průměr	SR	r	f_{CEC_4} Průměr	SR	r	f_{CEC_5} Průměr	SR	r
CAL-SHADE	3,69E+05	100	3	1,38E+01	100	2	1,60E−01	100	4
L-SHADE44	3,17E+05	100	2	1,36E+01	100	1	3,65E−31	100	2
L-SHADE+UDE	6,70E+06	100	7	1,39E+01	100	3	0,00E+00	100	1
UDE	7,33E+01	100	1	8,24E+01	100	6	2,32E−17	100	3
CGWO	1,31E−03	4	15	2,87E+02	100	11	8,19E+04	100	11
CSA	6,15E−03	0	16	6,09E+01	100	5	2,55E+01	100	5
GWO	8,49E−04	8	14	3,04E+02	100	12	1,10E+05	100	12
SCA	6,51E−04	20	12	4,40E+02	100	16	3,46E+01	68	16
SOMA ATA	1,16E−05	96	9	2,33E+02	100	9	1,81E+03	100	9
SOMA ATAA	4,31E−06	96	8	3,60E+02	100	13	2,57E+01	72	15
SOMA ATB	8,73E+05	100	4	3,85E+02	100	14	3,53E+00	96	13
SOMA ATR	9,00E+05	100	5	4,39E+02	100	15	3,97E+01	84	14
SOMA QI	1,05E+06	100	6	2,18E+02	100	7	5,60E+03	100	10
SOMA T3A	4,74E−04	20	11	1,50E+01	100	4	3,88E+01	100	8
SOMA M3-QI	7,49E−05	44	10	2,30E+02	100	8	3,09E+01	100	7
S M3-QI+MPC	2,58E−04	16	13	2,75E+02	100	10	2,56E+01	100	6

Tabulka 30: Výsledky algoritmů na funkcích 3−5 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	f_{CEC_6} Průměr	SR	r	f_{CEC_7} Průměr	SR	r	f_{CEC_8} Průměr	SR	r
CAL-SHADE	2,08E−02	0	11	−1,50E+02	100	2	−2,80E−04	100	1
L-SHADE44	1,78E−02	0	10	5,51E−06	96	4	−2,80E−04	100	1
L-SHADE+UDE	2,57E−02	0	12	4,06E−06	96	3	−2,60E−04	100	4
UDE	3,04E+02	100	2	−5,98E+02	100	1	−2,80E−04	100	1
CGWO	3,04E−01	0	14	8,97E+01	0	13	1,78E+02	0	11
CSA	−1,74E+01	100	1	3,27E+01	0	11	8,19E−01	0	9
GWO	1,92E−01	0	13	9,36E+01	0	14	2,00E+02	0	12
SCA	3,87E+00	0	16	4,50E+01	0	12	1,29E+03	0	16
SOMA ATA	2,62E+03	100	3	2,76E+01	4	7	2,73E+02	0	13
SOMA ATAA	3,18E−02	76	8	1,10E+02	0	16	4,77E+02	0	14
SOMA ATB	4,23E−03	92	6	1,10E+02	4	10	4,13E+01	0	10
SOMA ATR	4,84E−02	60	9	9,65E+01	4	8	9,62E+02	0	15
SOMA QI	2,34E−05	96	4	1,09E+02	0	15	6,28E−03	48	7
SOMA T3A	4,25E−01	0	15	5,61E−02	76	5	4,87E−03	0	8
SOMA M3-QI	1,16E−04	92	5	1,01E+02	4	9	5,09E−02	72	6
S M3-QI+MPC	2,49E−04	88	7	8,28E+00	4	6	2,35E−03	100	5

Tabulka 31: Výsledky algoritmů na funkcích 6−8 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	f_{CEC_9} Průměr	SR	r	$f_{CEC_{10}}$ Průměr	SR	r	$f_{CEC_{11}}$ Průměr	SR	r
CAL-SHADE	−2,67E−03	100	1	−1,00E−04	100	1	1,00E−19	88	3
L-SHADE44	−2,67E−03	100	1	−1,00E−04	100	1	−8,62E−01	100	2
L-SHADE+UDE	−2,67E−03	100	1	−9,80E−05	100	4	−8,65E−01	100	1
UDE	−2,67E−03	100	1	−1,00E−04	100	1	2,07E−02	0	4
CGWO	2,82E+02	0	15	2,27E+04	0	14	2,95E+02	0	13
CSA	7,11E+00	100	5	1,34E−03	0	9	1,24E+01	0	9
GWO	2,14E+02	0	14	2,60E+04	0	15	3,02E+02	0	14
SCA	3,96E+03	0	16	4,63E+04	0	16	6,36E+02	0	16
SOMA ATA	1,89E+01	28	11	8,03E+02	0	11	6,68E+01	0	10
SOMA ATAA	8,88E+01	8	13	3,36E+03	0	12	1,65E+02	0	12
SOMA ATB	1,01E+00	80	10	1,45E+02	0	10	8,31E+01	0	11
SOMA ATR	1,14E+01	16	12	3,63E+03	0	13	4,79E+02	0	15
SOMA QI	1,37E+01	100	7	1,06E−02	36	8	8,69E−01	0	6
SOMA T3A	2,13E−09	96	9	−5,79E−05	100	5	2,69E+00	0	8
SOMA M3-QI	1,42E+01	100	8	2,01E−03	100	7	2,18E+00	0	7
S M3-QI+MPC	9,14E+00	100	6	1,60E−04	100	6	4,10E−01	0	5

Tabulka 32: Výsledky algoritmů na funkcích 9−11 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{12}}$ Průměr	SR	r	$f_{CEC_{13}}$ Průměr	SR	r	$f_{CEC_{14}}$ Průměr	SR	r
CAL-SHADE	2,12E+01	100	9	1,50E+01	100	2	2,05E+00	100	7
L-SHADE44	3,99E+00	100	2	4,84E+00	100	1	1,83E+00	100	5
L-SHADE+UDE	6,07E+00	100	3	2,60E+01	100	3	1,91E+00	100	6
UDE	1,87E+01	100	8	8,15E+01	100	5	1,53E+00	100	1
CGWO	3,53E+03	0	14	2,58E+03	0	14	6,50E+03	0	14
CSA	1,00E+01	100	5	1,65E+00	88	6	2,17E+00	100	9
GWO	3,77E+03	0	15	2,75E+03	0	15	6,74E+03	0	15
SCA	6,96E+03	0	16	4,79E+03	0	16	1,36E+04	0	16
SOMA ATA	3,64E+00	92	10	6,84E+02	0	11	3,00E+00	4	11
SOMA ATAA	5,75E+00	80	11	7,33E+02	0	12	2,88E+02	0	12
SOMA ATB	3,85E+01	80	12	6,55E+02	0	10	1,52E+01	40	10
SOMA ATR	3,90E+02	32	13	9,63E+02	0	13	6,81E+02	0	13
SOMA QI	1,29E+01	100	7	4,13E+02	0	8	1,66E+00	100	2
SOMA T3A	3,98E+00	100	1	5,56E+01	100	4	1,71E+00	100	3
SOMA M3-QI	1,07E+01	100	6	4,89E+02	0	9	1,74E+00	100	4
S M3-QI+MPC	9,21E+00	100	4	1,38E+00	84	7	2,10E+00	100	8

Tabulka 33: Výsledky algoritmů na funkcích 12−14 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{15}}$ Průměr	SR	r	$f_{CEC_{16}}$ Průměr	SR	r	$f_{CEC_{17}}$ Průměr	SR	r
CAL-SHADE	4,30E−04	16	13	1,61E+02	100	5	1,55E+01	0	3
L-SHADE44	1,86E−05	84	8	1,49E+02	100	4	1,55E+01	0	3
L-SHADE+UDE	1,29E+01	100	1	1,43E+02	100	3	1,55E+01	0	3
UDE	2,62E−05	88	7	8,42E+00	100	1	1,55E+01	0	3
CGWO	2,10E+03	0	14	1,98E+03	0	14	3,59E+03	0	13
CSA	9,54E−06	88	6	2,02E+02	100	11	1,55E+01	0	7
GWO	2,63E+03	0	15	2,48E+03	0	15	3,70E+03	0	14
SCA	5,07E+03	0	16	5,43E+03	0	16	6,93E+03	0	15
SOMA ATA	9,01E−02	68	10	1,77E+02	100	6	1,73E+01	0	9
SOMA ATAA	1,27E−01	64	11	3,17E+01	96	13	2,04E+01	0	10
SOMA ATB	6,97E−02	84	9	1,80E+02	100	8	4,97E+01	0	11
SOMA ATR	1,88E−01	44	12	1,32E+01	96	12	2,26E+02	0	12
SOMA QI	2,45E+01	100	5	1,99E+02	100	10	1,55E+01	0	1
SOMA T3A	1,86E+01	100	3	1,30E+02	100	2	1,56E+01	0	8
SOMA M3-QI	2,32E+01	100	4	1,79E+02	100	7	1,55E+01	0	1
S M3-QI+MPC	1,64E+01	100	2	1,84E+02	100	9	4,40E+04	0	16

Tabulka 34: Výsledky algoritmů na funkcích 15−17 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{18}}$ Průměr	SR	r	$f_{CEC_{19}}$ Průměr	SR	r	$f_{CEC_{20}}$ Průměr	SR	r
CAL-SHADE	1,11E+06	0	9	2,14E+04	0	5	1,18E+00	100	1
L-SHADE44	1,58E+04	0	7	2,14E+04	0	5	1,40E+00	100	2
L-SHADE+UDE	1,33E+06	0	10	2,14E+04	0	5	2,25E+00	100	4
UDE	3,36E+07	0	12	2,14E+04	0	9	4,58E+00	100	10
CGWO	1,47E+08	0	14	2,15E+04	0	16	5,99E+00	100	12
CSA	1,06E+01	4	4	2,14E+04	0	4	8,82E+00	100	14
GWO	1,66E+08	0	15	2,15E+04	0	15	5,56E+00	100	11
SCA	3,46E+08	0	16	2,04E+04	0	1	9,28E+00	100	16
SOMA ATA	7,33E+04	0	8	2,14E+04	0	10	2,40E+00	100	6
SOMA ATAA	4,71E+07	0	13	2,14E+04	0	11	2,64E+00	100	7
SOMA ATB	1,08E+03	0	6	2,14E+04	0	12	2,25E+00	100	5
SOMA ATR	3,28E+06	0	11	2,14E+04	0	13	2,65E+00	100	8
SOMA QI	5,35E+01	4	5	2,14E+04	0	14	2,13E+00	100	3
SOMA T3A	1,10E+01	44	3	2,14E+04	0	2	3,74E+00	100	9
SOMA M3-QI	1,14E+01	48	2	2,14E+04	0	2	8,25E+00	100	13
S M3-QI+MPC	9,67E+00	64	1	2,14E+04	0	8	8,90E+00	100	15

Tabulka 35: Výsledky algoritmů na funkcích 18−20 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{21}}$ Průměr	SR	r	$f_{CEC_{22}}$ Průměr	SR	r	$f_{CEC_{23}}$ Průměr	SR	r
CAL-SHADE	1,95E+01	100	6	4,04E−01	88	4	2,06E+00	100	6
L-SHADE44	1,87E+01	100	5	3,06E+03	100	3	1,66E+00	100	3
L-SHADE+UDE	2,75E+01	100	8	8,44E+02	100	2	1,86E+00	100	4
UDE	1,48E−12	92	9	9,41E+01	100	1	1,43E+00	100	1
CGWO	1,16E+04	0	14	7,48E+03	0	14	2,46E+04	0	14
CSA	2,32E+01	100	7	3,19E+01	0	6	8,10E−06	92	9
GWO	1,41E+04	0	15	8,74E+03	0	15	2,58E+04	0	15
SCA	3,18E+04	0	16	2,19E+04	0	16	6,15E+04	0	16
SOMA ATA	1,16E+02	0	11	2,00E+03	0	11	1,26E+02	0	11
SOMA ATAA	6,05E+02	0	12	3,48E+03	0	12	1,01E+03	0	12
SOMA ATB	9,08E+01	40	10	1,71E+03	0	10	9,07E+01	24	10
SOMA ATR	1,94E+03	0	13	3,65E+03	0	13	4,35E+03	0	13
SOMA QI	1,50E+01	100	4	1,31E+03	0	8	1,52E+00	100	2
SOMA T3A	9,38E+00	100	2	1,07E+01	8	5	1,95E+00	100	5
SOMA M3-QI	1,48E+01	100	3	1,58E+03	0	9	2,00E−02	96	8
S M3-QI+MPC	4,54E+00	100	1	3,63E+01	0	7	2,13E+00	100	7

Tabulka 36: Výsledky algoritmů na funkcích 21−23 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{24}}$ Průměr	SR	r	$f_{CEC_{25}}$ Průměr	SR	r	$f_{CEC_{26}}$ Průměr	SR	r
CAL-SHADE	1,50E+01	100	5	1,66E+02	100	5	1,55E+01	0	1
L-SHADE44	1,22E+01	100	2	1,41E+02	100	3	1,55E+01	0	1
L-SHADE+UDE	1,39E+01	100	4	1,41E+02	100	2	1,55E+01	0	1
UDE	8,39E+00	100	1	1,60E+01	100	1	1,55E+01	0	1
CGWO	9,92E+03	0	14	1,01E+04	0	14	1,23E+04	0	15
CSA	2,49E−05	84	12	7,52E−06	88	11	1,55E+01	0	9
GWO	1,16E+04	0	15	1,10E+04	0	15	1,16E+04	0	14
SCA	3,37E+04	0	16	3,14E+04	0	16	3,23E+04	0	16
SOMA ATA	2,11E+01	100	10	2,39E+02	100	8	8,02E+01	0	10
SOMA ATAA	3,25E+01	92	11	3,27E+02	88	12	7,01E+02	0	12
SOMA ATB	1,97E+01	100	8	2,43E+02	100	10	9,58E+01	0	11
SOMA ATR	9,98E+02	68	13	5,23E+02	64	13	2,22E+03	0	13
SOMA QI	2,10E+01	100	9	2,39E+02	100	9	1,55E+01	0	5
SOMA T3A	1,38E+01	100	3	1,55E+02	100	4	2,51E+01	0	5
SOMA M3-QI	1,61E+01	100	7	1,97E+02	100	6	1,55E+01	0	5
S M3-QI+MPC	1,59E+01	100	6	2,00E+02	100	7	1,55E+01	0	5

Tabulka 37: Výsledky algoritmů na funkcích 24−26 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	$f_{CEC_{27}}$ Průměr	SR	r	$f_{CEC_{28}}$ Průměr	SR	r
CAL-SHADE	3,83E+07	0	10	2,15E+04	0	12
L-SHADE44	1,00E+07	0	8	2,15E+04	0	9
L-SHADE+UDE	1,08E+08	0	11	2,15E+04	0	10
UDE	3,41E+08	0	13	2,14E+04	0	8
CGWO	1,42E+09	0	14	2,12E+04	0	4
CSA	1,86E+03	0	4	2,08E+04	0	2
GWO	2,08E+09	0	15	2,06E+04	0	1
SCA	1,08E+10	0	16	2,12E+04	0	7
SOMA ATA	9,98E+06	0	7	2,15E+04	0	13
SOMA ATAA	2,43E+07	0	9	2,15E+04	0	16
SOMA ATB	1,47E+05	0	6	2,11E+04	0	3
SOMA ATR	1,55E+08	0	12	2,12E+04	0	6
SOMA QI	2,18E+04	0	5	2,12E+04	0	5
SOMA T3A	5,01E+02	8	3	2,15E+04	0	14
SOMA M3-QI	3,79E+02	12	2	2,15E+04	0	14
S M3-QI+MPC	2,74E+01	44	1	2,15E+04	0	11

Tabulka 38: Výsledky algoritmů na funkcích 27−28 CEC 2017 D = 30 potřebné pro určení *ranku* průměru: průměrná hodnota účelové funkce, respektive míry porušení omezení, úspěšnost běhů v % (SR) a *rank* průměru (r)

Algoritmus	f_{CEC_1} Medián	F	r	f_{CEC_2} Medián	F	r
CAL-SHADE	0,00E+00	A	1	0,00E+00	A	1
L-SHADE+UDE	0,00E+00	A	1	0,00E+00	A	1
L−SHADE 44	0,00E+00	A	1	0,00E+00	A	1
UDE	3,70E−29	A	4	4,42E−29	A	4
CGWO	5,80E+03	A	15	7,44E+03	A	15
CSA	1,82E+01	A	9	6,96E+00	A	9
GWO	5,68E+03	A	14	7,03E+03	A	14
SCA	3,52E+04	A	16	3,89E+04	A	16
SOMA ATA	1,10E+03	A	12	1,14E+03	A	12
SOMA ATAA	4,99E+02	A	11	8,88E+02	A	11
SOMA ATB	5,11E−01	A	8	2,12E+00	A	8
SOMA ATR	2,28E+03	A	13	1,92E+03	A	13
SOMA QI	1,09E−02	A	6	1,87E−02	A	7
SOMA T3A	2,35E−02	A	7	1,34E−02	A	6
SOMA M3-QI	1,25E−03	A	5	6,45E−04	A	5
S M3-QI+MPC	3,95E+02	A	10	1,35E+02	A	10

Tabulka 39: Výsledky algoritmů na funkcích 1−2 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	f_{CEC_3} Medián	F	r	f_{CEC_4} Medián	F	r	f_{CEC_5} Medián	F	r
CAL-SHADE	1,44E−03	N	15	1,13E+02	A	6	0,00E+00	A	1
L-SHADE44	6,58E+06	A	8	1,36E+01	A	1	0,00E+00	A	1
L-SHADE+UDE	2,06E+05	A	2	1,36E+01	A	2	0,00E+00	A	1
UDE	7,47E+01	A	1	7,76E+01	A	5	8,78E−25	A	4
CGWO	7,43E−04	N	14	2,83E+02	A	10	8,35E+04	A	14
CSA	4,69E−03	N	16	5,59E+01	A	4	2,36E+01	A	7
GWO	4,60E−04	N	13	3,08E+02	A	12	9,70E+04	A	15
SCA	1,85E−04	N	11	4,46E+02	A	16	3,34E+05	A	16
SOMA ATA	5,01E+05	A	3	2,23E+02	A	8	1,93E+02	A	11
SOMA ATAA	6,00E+05	A	6	3,60E+02	A	13	3,22E+02	A	12
SOMA ATB	8,24E+05	A	7	3,93E+02	A	14	1,38E+02	A	10
SOMA ATR	5,48E+05	A	5	4,40E+02	A	15	9,72E+03	A	13
SOMA QI	5,47E+05	A	4	2,15E+02	A	7	7,50E+01	A	9
SOMA T3A	3,17E−04	N	12	1,36E+01	A	3	2,46E+01	A	8
SOMA M3-QI	5,41E−05	N	9	2,32E+02	A	9	2,18E+01	A	5
S M3-QI+MPC	1,78E−04	N	10	2,90E+02	A	11	2,34E+01	A	6

Tabulka 40: Výsledky algoritmů na funkcích 3−5 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	f_{CEC_6} Medián	F	r	f_{CEC_7} Medián	F	r	f_{CEC_8} Medián	F	r
CAL-SHADE	3,83E+03	A	9	6,70E−05	N	5	−2,84E−04	A	1
L-SHADE44	2,55E−02	N	11	−8,08E+01	A	3	−2,70E−04	A	4
L-SHADE+UDE	1,27E−02	N	10	−1,34E+02	A	2	−2,84E−04	A	2
UDE	2,78E+02	A	1	−5,70E+02	A	1	−2,83E−04	A	3
CGWO	6,98E−01	N	13	4,01E+02	N	15	1,73E+02	N	11
CSA	9,01E+00	N	15	2,97E+02	N	9	7,25E−01	N	10
GWO	3,85E−01	N	12	3,60E+02	N	12	1,92E+02	N	12
SCA	1,07E+01	N	16	4,88E+02	N	16	1,13E+03	N	16
SOMA ATA	2,56E+03	A	5	9,00E+01	N	7	2,05E+02	N	13
SOMA ATAA	2,95E+03	A	6	3,02E+02	N	10	3,51E+02	N	14
SOMA ATB	3,74E+03	A	8	3,68E+02	N	14	7,45E−02	N	9
SOMA ATR	2,46E+03	A	4	3,03E+02	N	11	7,93E+02	N	15
SOMA QI	3,24E+03	A	7	3,64E+02	N	13	5,98E−05	N	7
SOMA T3A	1,19E+00	N	14	−1,75E+01	A	4	2,87E−03	N	8
SOMA M3-QI	2,33E+03	A	3	2,26E+02	N	8	7,52E−03	A	6
S M3-QI+MPC	1,53E+03	A	2	3,19E+01	N	6	2,21E−03	A	5

Tabulka 41: Výsledky algoritmů na funkcích 6−8 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	f_{CEC_9} Medián	F	r	$f_{CEC_{10}}$ Medián	F	r	$f_{CEC_{11}}$ Medián	F	r
CAL-SHADE	−2,67E−03	A	1	−1,03E−04	A	1	−9,20E−01	A	3
L-SHADE44	−2,67E−03	A	3	−9,91E−05	A	4	−9,25E−01	A	2
L-SHADE+UDE	−2,67E−03	A	2	−1,03E−04	A	2	−9,25E−01	A	1
UDE	−2,67E−03	A	4	−1,02E−04	A	3	4,26E−03	N	4
CGWO	2,03E+02	N	14	2,33E+04	N	14	2,56E+02	N	13
CSA	6,02E+00	A	5	1,11E−03	N	9	1,00E+01	N	9
GWO	2,27E+02	N	15	2,69E+04	N	15	3,21E+02	N	15
SCA	5,16E+03	N	16	4,43E+04	N	16	6,28E+02	N	16
SOMA ATA	1,83E+01	N	12	5,06E+02	N	11	5,69E+01	N	11
SOMA ATAA	2,82E+01	N	13	1,38E+03	N	12	1,47E+02	N	12
SOMA ATB	1,41E+01	A	10	1,57E+00	N	10	3,33E+01	N	10
SOMA ATR	1,71E+01	N	11	2,74E+03	N	13	2,82E+02	N	14
SOMA QI	1,35E+01	A	8	2,05E−04	N	8	5,36E−01	N	6
SOMA T3A	1,14E+01	A	7	−6,54E−05	A	5	1,47E+00	N	8
SOMA M3-QI	1,40E+01	A	9	1,47E−03	A	7	9,06E−01	N	7
S M3-QI+MPC	8,67E+00	A	6	1,60E−04	A	6	1,39E−01	N	5

Tabulka 42: Výsledky algoritmů na funkcích 9−11 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{12}}$ Medián	F	r	$f_{CEC_{13}}$ Medián	F	r	$f_{CEC_{14}}$ Medián	F	r
CAL-SHADE	9,78E+00	A	4	1,45E+02	A	5	1,50E+00	A	2
L-SHADE44	3,98E+00	A	1	0,00E+00	A	1	1,90E+00	A	7
L-SHADE+UDE	3,99E+00	A	3	5,91E+00	A	3	1,84E+00	A	6
UDE	9,78E+00	A	5	3,99E+00	A	2	1,50E+00	A	1
CGWO	3,62E+03	N	14	3,76E+03	N	14	6,31E+03	N	14
CSA	9,96E+00	A	7	1,63E+02	A	6	2,18E+00	A	9
GWO	3,77E+03	N	15	4,15E+03	N	15	6,89E+03	N	15
SCA	6,91E+03	N	16	7,19E+03	N	16	1,43E+04	N	16
SOMA ATA	3,68E+01	A	12	9,95E+02	N	11	1,92E+00	N	11
SOMA ATAA	3,42E+01	A	11	1,07E+03	N	12	2,18E+00	N	12
SOMA ATB	2,69E+01	A	10	9,51E+02	N	10	8,67E−01	N	10
SOMA ATR	4,16E+01	N	13	1,22E+03	N	13	4,49E+01	N	13
SOMA QI	9,78E+00	A	6	5,71E+02	N	8	1,63E+00	A	3
SOMA T3A	3,98E+00	A	2	7,96E+01	A	4	1,73E+00	A	5
SOMA M3-QI	9,96E+00	A	8	6,98E+02	N	9	1,71E+00	A	4
S M3-QI+MPC	1,06E+01	A	9	7,53E+03	A	7	2,10E+00	A	8

Tabulka 43: Výsledky algoritmů na funkcích 12−14 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{15}}$ Medián	F	r	$f_{CEC_{16}}$ Medián	F	r	$f_{CEC_{17}}$ Medián	F	r
CAL-SHADE	2,36E+01	A	10	2,82E−03	N	13	1,55E+01	N	13
L-SHADE44	1,18E+01	A	3	1,45E+02	A	4	1,55E+01	N	1
L-SHADE+UDE	1,81E+01	A	7	1,51E+02	A	5	1,55E+01	N	1
UDE	8,64E+00	A	1	6,28E+00	A	1	1,55E+01	N	1
CGWO	1,80E+03	N	14	1,90E+03	N	14	3,85E+03	N	15
CSA	1,49E+01	A	5	2,03E+02	A	12	1,55E+01	N	12
GWO	2,71E+03	N	15	2,45E+03	N	15	3,64E+03	N	14
SCA	5,20E+03	N	16	5,72E+03	N	16	7,05E+03	N	16
SOMA ATA	2,12E+01	A	8	1,88E+02	A	8	1,55E+01	N	1
SOMA ATAA	1,18E+01	A	2	1,51E+02	A	6	1,55E+01	N	1
SOMA ATB	2,75E+01	A	12	1,88E+02	A	8	1,55E+01	N	1
SOMA ATR	3,51E−02	N	13	1,45E+02	A	3	1,55E+01	N	1
SOMA QI	2,43E+01	A	11	2,01E+02	A	11	1,55E+01	N	1
SOMA T3A	1,81E+01	A	6	1,27E+02	A	2	1,55E+01	N	1
SOMA M3-QI	2,12E+01	A	9	1,82E+02	A	7	1,55E+01	N	1
S M3-QI+MPC	1,49E+01	A	4	1,88E+02	A	10	1,55E+01	N	1

Tabulka 44: Výsledky algoritmů na funkcích 15−17 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{18}}$ Medián	F	r	$f_{CEC_{19}}$ Medián	F	r	$f_{CEC_{20}}$ Medián	F	r
CAL-SHADE	7,68E+05	N	11	1,42E+04	N	1	2,08E+00	A	2
L-SHADE44	1,01E+06	N	12	2,14E+04	N	3	2,29E+00	A	4
L-SHADE+UDE	2,08E+03	N	7	2,14E+04	N	2	1,45E+00	A	1
UDE	7,86E+04	N	9	2,14E+04	N	7	5,15E+00	A	10
CGWO	2,26E+08	N	14	2,15E+04	N	15	6,18E+00	A	12
CSA	1,82E−03	N	2	2,14E+04	N	6	8,81E+00	A	14
GWO	2,28E+08	N	15	2,15E+04	N	14	5,74E+00	A	11
SCA	5,05E+08	N	16	2,15E+04	N	16	9,39E+00	A	16
SOMA ATA	1,59E+04	N	8	2,14E+04	N	8	2,39E+00	A	6
SOMA ATAA	4,16E+06	N	13	2,14E+04	N	9	2,58E+00	A	8
SOMA ATB	1,74E+02	N	6	2,14E+04	N	10	2,34E+00	A	5
SOMA ATR	5,23E+05	N	10	2,14E+04	N	11	2,49E+00	A	7
SOMA QI	3,18E+01	N	5	2,14E+04	N	13	2,16E+00	A	3
SOMA T3A	2,80E+01	N	4	2,14E+04	N	12	3,82E+00	A	9
SOMA M3-QI	1,10E−02	N	3	2,14E+04	N	3	8,78E+00	A	13
S M3-QI+MPC	4,71E+01	A	1	2,14E+04	N	5	9,01E+00	A	15

Tabulka 45: Výsledky algoritmů na funkcích 18−20 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{21}}$ Medián	F	r	$f_{CEC_{22}}$ Medián	F	r	$f_{CEC_{23}}$ Medián	F	r
CAL-SHADE	9,78E+00	A	2	1,80E+04	A	4	1,49E+00	A	3
L-SHADE44	2,83E+01	A	8	4,78E+02	A	2	1,85E+00	A	5
L-SHADE+UDE	2,83E+01	A	7	7,96E+02	A	3	1,74E+00	A	4
UDE	9,78E+00	A	2	8,06E+01	A	1	1,41E+00	A	1
CGWO	1,09E+04	N	14	1,09E+04	N	14	2,50E+04	N	14
CSA	2,87E+01	A	9	4,90E+01	N	6	2,16E+00	A	8
GWO	1,42E+04	N	15	1,41E+04	N	15	2,62E+04	N	15
SCA	3,38E+04	N	16	3,36E+04	N	16	6,30E+04	N	16
SOMA ATA	8,99E+01	N	11	2,93E+03	N	11	7,01E+01	N	11
SOMA ATAA	3,58E+02	N	12	5,45E+03	N	13	8,76E+02	N	12
SOMA ATB	1,55E+00	N	10	2,76E+03	N	10	5,50E+00	N	10
SOMA ATR	1,10E+03	N	13	5,26E+03	N	12	3,93E+03	N	13
SOMA QI	1,46E+01	A	6	1,92E+03	N	8	1,48E+00	A	2
SOMA T3A	9,78E+00	A	4	1,78E+01	N	5	1,96E+00	A	6
SOMA M3-QI	1,46E+01	A	5	2,42E+03	N	9	2,21E+00	A	9
S M3-QI+MPC	3,98E+00	A	1	5,00E+01	N	7	2,15E+00	A	7

Tabulka 46: Výsledky algoritmů na funkcích 21−23 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{24}}$ Medián	F	r	$f_{CEC_{25}}$ Medián	F	r	$f_{CEC_{26}}$ Medián	F	r
CAL-SHADE	4,34E−02	N	13	9,66E−04	N	13	1,55E+01	N	10
L-SHADE44	1,49E+01	A	4	1,40E+02	A	2	1,55E+01	N	1
L-SHADE+UDE	1,18E+01	A	2	1,46E+02	A	3	1,55E+01	N	1
UDE	8,64E+00	A	1	1,26E+01	A	1	1,55E+01	N	1
CGWO	8,57E+03	N	14	9,87E+03	N	14	1,25E+04	N	15
CSA	1,49E+01	A	5	2,03E+02	A	7	1,55E+01	N	9
GWO	1,11E+04	N	15	1,07E+04	N	15	1,15E+04	N	14
SCA	3,38E+04	N	16	3,12E+04	N	16	3,23E+04	N	16
SOMA ATA	2,12E+01	A	12	2,39E+02	A	9	1,55E+01	N	1
SOMA ATAA	1,81E+01	A	8	2,45E+02	A	12	3,49E+02	N	12
SOMA ATB	1,81E+01	A	8	2,40E+02	A	10	1,55E+01	N	1
SOMA ATR	1,81E+01	A	8	2,32E+02	A	8	1,15E+03	N	13
SOMA QI	1,81E+01	A	8	2,40E+02	A	10	1,55E+01	N	1
SOMA T3A	1,49E+01	A	3	1,51E+02	A	4	2,06E+01	N	11
SOMA M3-QI	1,49E+01	A	7	1,96E+02	A	5	1,55E+01	N	1
S M3-QI+MPC	1,49E+01	A	6	2,01E+02	A	6	1,55E+01	N	1

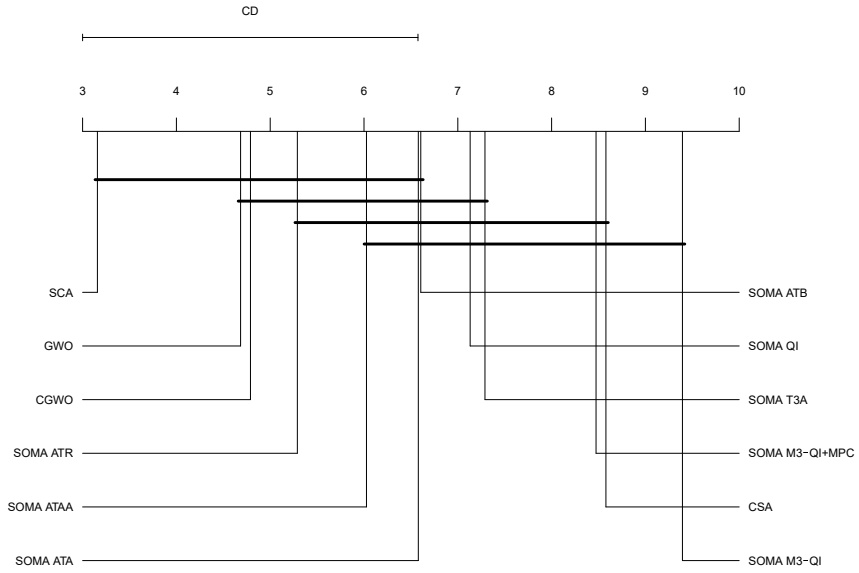
Tabulka 47: Výsledky algoritmů na funkcích 24−26 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

Algoritmus	$f_{CEC_{27}}$ Medián	F	r	$f_{CEC_{28}}$ Medián	F	r
CAL-SHADE	4,09E+06	N	10	1,43E+04	N	1
L-SHADE44	6,83E+07	N	13	2,15E+04	N	3
L-SHADE+UDE	5,10E+06	N	11	2,15E+04	N	4
UDE	5,79E+05	N	7	2,15E+04	N	2
CGWO	1,86E+09	N	14	2,15E+04	N	10
CSA	3,63E−01	N	2	2,15E+04	N	15
GWO	2,95E+09	N	15	2,15E+04	N	7
SCA	1,57E+10	N	16	2,15E+04	N	16
SOMA ATA	8,44E+05	N	8	2,15E+04	N	6
SOMA ATAA	3,58E+06	N	9	2,15E+04	N	12
SOMA ATB	1,56E+04	N	6	2,15E+04	N	14
SOMA ATR	6,32E+07	N	12	2,15E+04	N	13
SOMA QI	5,98E+02	N	5	2,15E+04	N	11
SOMA T3A	3,78E+01	N	4	2,15E+04	N	8
SOMA M3-QI	1,12E+00	N	3	2,15E+04	N	9
S M3-QI+MPC	7,36E−05	N	1	2,15E+04	N	5

Tabulka 48: Výsledky algoritmů na funkcích 27−28 CEC 2017 D = 30 potřebné pro určení *ranku* mediánu: hodnota účelové funkce mediánu, respektive míry porušení omezení, označení, zda je řešení ve splnitelné oblasti (F, A−ano, N−ne) a *rank* mediánu (r)

11.3 Statistické srovnání

Výsledky nově představené strategie SOMA M3-QI budou statisticky srovnány se všemi původními strategiemi SOMA, s novou strategií SOMA M3-QI+MPC, dále s algoritmy SOMA T3A, SOMA QI, SCA, GWO, CGWO a CSA na funkcích návrhu parametrů tlakového válce, návrhu kosntrukce svařovaných nosníků, návrhu parametrů pružiny a na funkcích f_{CEC_1}, f_{CEC_2}, $f_{CEC_{11}}$, $f_{CEC_{17}}$, $f_{CEC_{19}}$, $f_{CEC_{20}}$, $f_{CEC_{26}}$, $f_{CEC_{28}}$ z CEC 2017 na 10 a 30 dimenzích. Funkce z CEC 2017 byly vybrány tak, aby byla úspěšnost běhů všech algoritmů 100%, nebo 0%. Lze tak srovnávat průměrné hodnoty účelové funkce, respektive průměrné hodnoty míry porušení omezení. Srovnávat hodnotu účelové funkce v případě, kdy pouze část algoritmů nenalezla řešení ve splnitelné oblasti možné není a proto musely být tyto funkce ze statistického srovnání vypuštěny.
Pro určení, zda-li se algoritmy chovají rozdílně je využit Iman-Davenportův test [87]. P-hodnota provedeného testu byla $9.064E-09$, mezi průměrnými výsledky tedy existují statisticky významné rozdíly.
Může být přistoupeno k post-hoc analýze. Algoritmy byly rozděleny do ranků Friedmanovým testem a následně byl proveden Nemenyiův test, jehož výsledek znázorňuje Obrázek 21. Kritická vzdálenost je $3,8654$. Lze tedy konstatovat, že algoritmus SOMA M3-QI nalezl statisticky významně lepších výsledků, než algoritmy SOMA AllToOneRand, CGWO, GWO a SCA.



Obrázek 21: Grafická prezentace výsledků Nemenyiova testu

Pro přesnější post-hoc analýzu byl proveden Wilcoxnův test. Výsledky p-hodnot state-of-art algoritmů s algoritmem SOMA M3-QI uvádí tabulka 49. Kompletní výsledky post-hoc analýzy Wilcoxnova testu jsou součástí přílohy. Na hladině významnosti 95% můžeme konstatovat, že SOMA M3-QI dosáhl statisticky významně lepších výsledků, než algoritmy CGWO, CSA, GWO, SCA, všechny původní varianty SOMA a SOMA QI. Rozdíly ve výsledcích SOMA T3A, SOMA M3-QI+MPC a SOMA M3-QI nebyly statisticky významné. Tyto slibné výsledky ukazují ve-

lice dobrou schopnost nově prezentovaného algoritmu SOMA M3-QI nalézat optimální výsledky. Jak již bylo zmíněno dříve, hlavním zlepšení přináší adaptivní posun cílových bodů, který umožňuje rychlou migraci. Díky změně perturbace využívající chaos algoritmus obvykle neuvázne v lokálním extrému, přesto však existují funkce, které dokáží řešit původní strategie úspěšněji.

	Průměrné	Nemenyiho test		Wilcoxnův test	
Algoritmy	ranky	CI	Sign.	p-hodnota	Sign.
CGWO	7,2	[3,3; 11,1]	†	0,004	†
CSA	3,4	[-0,4; 7,3]		0,042	†
GWO	7,3	[3,5; 11,2]	†	0,002	†
SCA	8,8	[5,0; 12,7]	†	0,005	†
SOMA ATA	5,4	[1,6; 9,3]		0,004	†
SOMA ATAA	6,0	[2,1; 9,8]		0,006	†
SOMA ATB	5,4	[1,5; 9,3]		0,009	†
SOMA ATR	6,7	[2,8; 10,6]	†	0,005	†
SOMA M3-QI	2,6	[-1,3; 6,5]		N.A.	
SOMA M3-QI+MPC	3,5	[-0,3; 7,4]		0,397	
SOMA QI	4,9	[1,0; 8,7]		0,023	†
SOMA T3A	4,7	[0,8; 8,6]		0,011	

Tabulka 49: Statistické srovnání state-of-art algoritmů s nově představeným algoritmem SOMA M3-QI. Znak † je uveden, jestliže bylo daným testem nalezeno statisticky významného rozdílu.

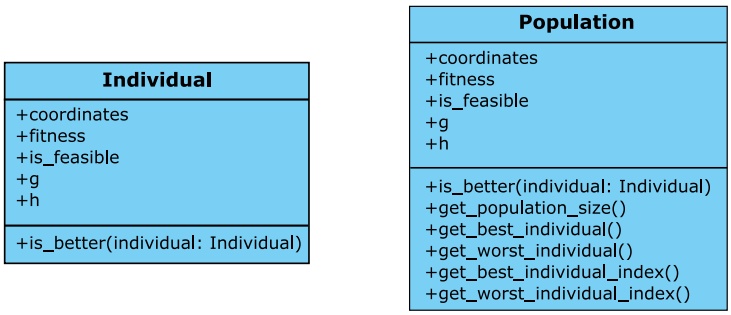
12 Objektový návrh aplikace

V jazyce Python byla vyvinuta aplikace schopná paralelně provádět výpočty vyhledávacích algoritmů na testovacích funkcích a snadno porovnávat jejich výsledky. Z důvodu nutnosti implementace vícero vyhledávacích algoritmů a účelových funkcí byly pro tyto případy definované abstraktní třídy, sloužící jako předpis pro jednotlivé implementace.
Program se skládá z několika částí:

- Třídy reprezentující účelové funkce
- Algoritmy pro vyhledávání optima
- Metody pro výpočet a porovnání výsledků jednotlivých algoritmů a jejich vizualizaci
- Modelové třídy jedinců a populace
- Pomocné třídy a metody

12.1 Modelové třídy

Evoluční algoritmy pracují s jedinci, kteří tvoří populaci. Teto vztah je zachycen v modelových třídách, které jsou zobrazeny na obrázku 22. Základní třídou je třída `Individual` znázorňující jedince v populaci, jeho polohu, kvalitu a metodu pro porovnání s ostatními. Skupinu jedinců, tedy populaci modeluje třída `Popuation` obsahující seznam jedinců a metody pro práci s nimi.



Obrázek 22: Diagram modelových tříd

Atributy:

c	Souřadnice na nichž se jedinec nachází
fitness	Hodnota účelové funkce jedince
is_feasible	Příznak, zda se jedinec nachází v proveditelném prostoru
g	Seznam obsahující hodnotu porušení proveditelnosti g
h	Seznam obsahující hodnotu porušení proveditelnosti h

Metody:

is_better(individual: Individual)	Určuje zda je konkrétní jedinec umístěn na lepších souřadnicích, než jedinec předaný parametrem

Tabulka 50: Třída `Individual`

Atributy:

population	Seznam jedinců v populaci

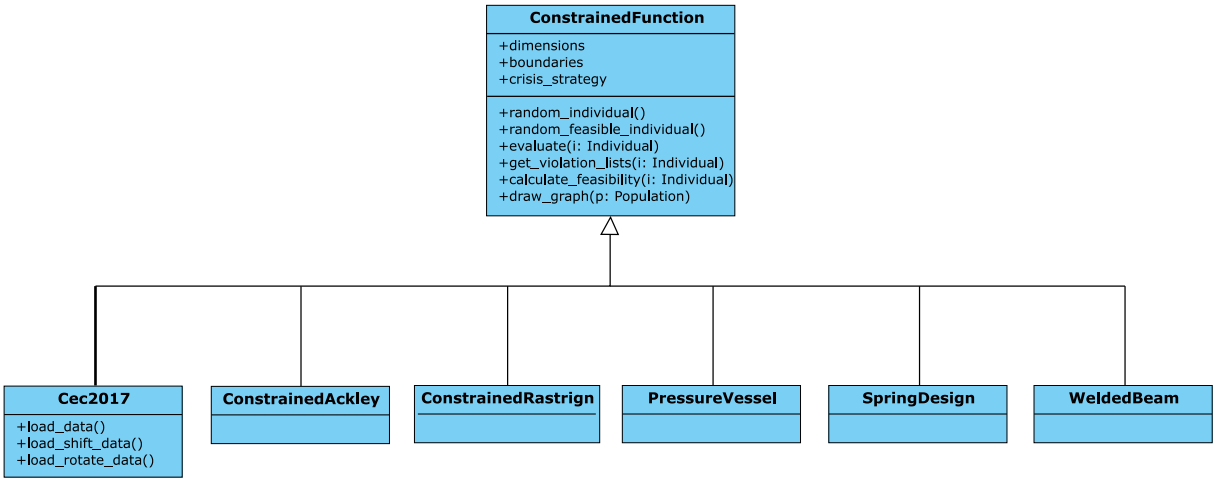
Metody:

add_individual(individual: Individual)	Přidá jedince do populace
get_population_size()	Vrací počet jedinců v populaci
get_best_individual()	Vrací nejlépe ohodnoceného jedince v populaci
get_worst_individual()	Vrací nejhůře ohodnoceného jedince v populaci
get_best_individual_index()	Vrací index nejlépe ohodnoceného jedince v populaci
get_worst_individual_index()	Vrací index nejhůře ohodnoceného jedince v populaci

Tabulka 51: Třída `Population`

12.2 Třídy reprezentující účelové funkce

Účelové funkce s omezeními jsou součástí balíčku `constrained_functions` jehož diagram je zobrazen na obrázku 23. Obecnou účelovou funkci definuje abstraktní třída `ConstrainedFunction`, která je děděna konkrétními účelovými funkcemi.



Obrázek 23: Diagram tříd účelových funkcí

Atributy:

dimensions	Počet dimenzí funkce
boundaries	Definuje hranice funkce
crisis_strategy	Způsob řešení výpočtu pokud se jedinec nachází mimo proveditelný prostor

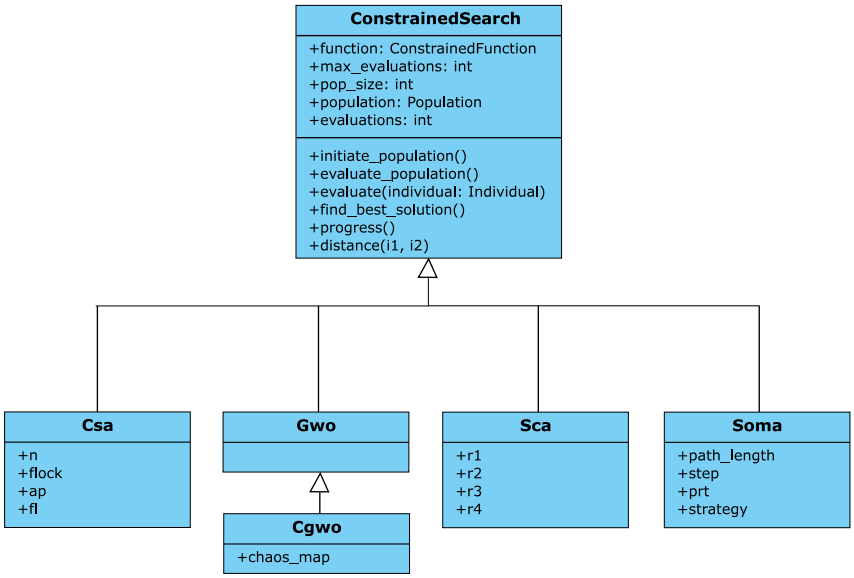
Metody:

random_individual()	Vrátí jedince umístěného na náhodných souřadnicích uvnitř hranic funkce
random_feasible_individual()	Vrátí jedince umístěného na náhodných souřadnicích uvnitř proveditelného prostoru
evaluate(i: Individual)	Vrátí hodnotu účelové funkce jedince
get_violation_lists(i: Individual)	Vrátí seznamy obsahující porušení podmínek proveditelnosti jedince
calculate_feasibility(i: Individual)	Vrátí hodnotu určující míru splnitelnosti na základě **crisis_strategy**
draw_graph(p: Population)	Vykreslí graf zobrazující populaci v prostoru účelové funkce

Tabulka 52: Abstraktní třída `ConstrainedFunction`

12.3 Třídy obsahující vyhledávací algoritmy

Vyhledávací algoritmy jsou obsaženy v balíčku `search_algs` zobrazeném na obrázku 24. Jako předpis pro vyhledávací algoritmus slouží abstraktní třída `ConstrainedSearch`. Hlavní funkcí je `find_best_solution()`, která provede kompletní evoluční cyklus a vrátí nejlepšího jedince, je tedy jedinou nutnou metodou k implementaci.



Obrázek 24: Diagram tříd vyhledávacích algoritmů

Atributy:

function: ConstrainedFunction	Účelová funkce
max_evaluations	Maximální počet výpočtů hodnoty účelové funkce
pop_size	Počet jedinců v populaci
population	Populace
evaluations	Počet evaluací účelové funkce

Metody:

initiate_population()	Vytvoří novou populaci na náhodných souřadnicích
evaluate_population()	Vypočte hodnotu účelové funkce všech jedinců v populaci
evaluate(individual: Individual)	Vypočte hodnotu účelové funkce daného jedince
find_best_solution()	Provede výpočet a vrátí nejlepšího nalezeného jedince
progress()	Vypočte v jaké fázi výpočtu se aktuálně algoritmus nachází
distance(i1: Individual, i2: Individual)	vrací euklidovskou vzdálenost dvou jedinců

Tabulka 53: Abstraktní třída `ConstrainedSearch`

12.4 Třídy zodpovědné za zaznamenávání a srovnávaní výsledků

Program je schopen spouštět jednotlivé evoluční algoritmy na testovacích funkcích a zaznamenávat nejlepší jedince v průběhu. Z těchto záznamů pak vytvoří podrobné srovnání a vše uloží do souboru. O měření výsledků se stará třída `Measure`. Hlavní metodou je `get_full_stats()`, jež

opakovaně volá daný vyhledávací algoritmus, monitoruje jeho postup a porovná jeho výsledky, vše nakonec uloží do souboru. Jelikož je potřeba spouštět stejný algoritmus vícekrát, bylo využito balíčku *multiprocessing* umožňující parametrické spouštění algoritmů, jakožto separátních procesů.
Třída `Measure` však pracuje pouze s jedním konkrétním algoritmem. Porovnání výsledků jednotlivých algoritmů s ostatními na dané funkci, spolu s možností srovnávacího vykreslení grafu je součástí třídy `SumStats`.

Atributy:

algorithm: ConstrainedSearch	Vyhledávací algoritmus
name	Jméno pod kterým budou statistiky uloženy
number_of_runs	Počet běhů algoritmu
save_statistics_interval	Intervaly ukládání nejlepšího jedince

Metody:

get_full_stats()	Provede daný počet měření, vypočte jejich průměr, medián, směrodatnou odchylku atp. a uloží výsledky do souboru
run_search()	Provede daný počet měření a uloží výsledky
print_stats()	Vypíše do konzole získané porovnání výsledků
save_results()	Provede uložení dat do souboru
set_violations_stats()	Provede výpočet četnosti a míry poručení jedinců v neproveditelné oblasti
set_solution_stats()	Provede porovnání účelových hodnot jedinců
get_eval_duration_time()	Vrací dobu trvání běhu algoritmu ve vteřinách

Tabulka 54: Třída `Measure`

Atributy:

func_name	Název funkce
alg_name	Název algoritmu

Metody:

create_stats_for_best_solutions()	Vytvoří porovnání nejlepších jedinců daných algoritmů
create_stats_for_median_solutions()	Vytvoří porovnání mediánu řešení daných algoritmů
create_stats_for_Průměr_solutions()	Vytvoří porovnání průměrných jedinců daných algoritmů
save_results()	Uloží výsledky do souboru
plot()	Vytvoří graf srovnávající jednotlivé algoritmy

Tabulka 55: Třída `SumStats`

12.5 Ostatní třídy a metody

V programu jsou využity ještě pomocné třídy a metody. Tyto třídy obsahují zpravidla statické metody pro práci s grafy, měření vzdáleností dvou bodů, respektive dvou jedinců, chaotické mapy a podobně. Součástí je také jednoduchá vizualizace postupu algoritmů na inženýrských funkcích s omezeními, které jsou popsány v sekci 8.1. Ta je součástí přílohy.

13 Závěr

Cílem práce bylo nalézt zlepšení Samo-organizujícího se algoritmu na funkcích s omezeními. Inkrementálním vývojem vycházejícím z nápadu balancování strategií bylo nalezeno inovativního přístupu k migraci jedinců. Hlavní změna spočívá v pohybu (skocích) ke dvěma cílovým jedincům (řešením), jež mají schopnost se během migrační fáze posouvat - adaptovat. Prvním takovým jedincem je jedinec s nejlepším ohodnocením, druhý jedinec je pak výsledkem kvadratické interpolace. Tento přístup spolu s upraveným způsobem perturbace přináší na mnoha funkcích výrazná zlepšení. Ke zlepšení výkonu SOMA rovněž přispěla aplikace chaotických metod.
Výsledný algoritmus byl porovnán na inženýrských problémech, tedy: návrhu parametrů tlakového válce, návrhu konstrukce svařovaných nosníků a návrhu parametrů pružiny s 14 state-of-art algoritmy. Na základě t-testu nalezla nově představená metoda na každém z těchto problému statisticky významně lepších výsledků než původní strategie a také než většina dalších algoritmů. Proběhlo i porovnávání na 28 funkcích benchmarku CEC 2017 na 10 a 30 dimenzích s 11 algoritmy. Při srovnání dle pravidel CEC 2017 se nový algoritmus umístil na pátém místě, přitom nejlepší původní strategie SOMA AllToBest dosáhla až na místo desáté. Tím předčil veškeré srovnávané hejnové algoritmy, překonaly jej však algoritmy pracující na principu diferenciální evoluce. Ověření průměrných výsledků algoritmů proběhlo také statistickými testy. Post-hoc analýza dle Wilcoxonova testu potvrdila statisticky významného zlepšení v porovnáním s původními strategiemi SOMA.
Součástí práce je také program v jazyce Python umožňující kromě spouštění implementovaných algoritmů na zvolených funkcích, zaznamenávání a prezentaci výsledků také vizualizovat postup nejlepšího řešení na inženýrských problémech.
Mnoho optimalizačních algoritmů mění své chování úpravou parametrů v průběhu výpočtu. Může tím například dojít k zevrubnější exploraci během počáteční fáze a k podrobnému prozkoumávání malého prostoru ve fázi konečné. Aplikací takové metody na nové parametry prezentovaného algoritmu, které zásadně ovlivňují jeho výkon, by mohlo přinést další zlepšení dosahovaných výsledků. Dalším přínosem by mohla být kombinace více způsobů volby cílových jedinců, což by umožňovalo vetší kontrolu nad mírou diverzity v populaci.
Algoritmy řešící globální optimalizaci jsou již dlouhou dobu předmětem intenzivního výzkumu. Výjimkou není ani algoritmus SOMA, jehož inovace byly prezentovány i v nedávné době. Například verze týmové adaptace (SOMA T3A) a samo-adaptivní SOMA (SASOMA) byly představeny v roce 2019. Je tedy velice pravděpodobné, že budou i nadále objevovány nové optimalizační techniky s různorodou oblastí působení.

Literatura

[1] BACK, Thomas. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford university press, 1996.

[2] DASGUPTA, Dipankar; MICHALEWICZ, Zbigniew (ed.). Evolutionary algorithms in engineering applications. *Springer Science & Business Media*, 2013.

[3] ZELINKA, Ivan. SOMA—self-organizing migrating algorithm. In: *New optimization techniques in engineering.* Springer, Berlin, Heidelberg, 2004. p. 167-217.

[4] VOLNÁ, Eva. *Evoluční algoritmy a neuronové sítě.* Ostrava: Ostravská univerzita, 2012

[5] ZELINKA, Ivan. *Evoluční výpočetní techniky: principy a aplikace.* Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-218-3.

[6] KAMAL, Nashwa A.; IBRAHIM, Ahmed M. Conventional, intelligent, and fractional-order control method for maximum power point tracking of a photovoltaic system: a review. In: *Fractional Order Systems.* Academic Press, 2018. p. 603-671.

[7] DUTOT, Antoine; OLIVIER, Damien. Swarm Problem-Solving. In: *Agent-based Spatial Simulation with NetLogo*, Volume 2. Elsevier, 2017. p. 117-172.

[8] WOLPERT, David H.; MACREADY, William G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1997, 1.1: 67-82.

[9] GARG, Harish. A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation*, 2016, 274: 292-305.

[10] DEB, Kalyanmoy. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering,* 2000, 186.2-4: 311-338.

[11] ASKARZADEH, Alireza. A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & Structures*, 2016, 169: 1-12.

[12] SINGH, Dipti; AGRAWAL, Seema. Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems. *Applied Soft Computing,* 2016, 38: 1040-1048.

[13] SINGH, Dipti; AGRAWAL, Seema; DEEP, Kusum. C-SOMAQI: self organizing migrating algorithm with quadratic interpolation crossover operator for constrained global optimization. In: *Self-Organizing Migrating Algorithm.* Springer, Cham, 2016. p. 147-165.

[14] DIEP, Quoc Bao. Self-Organizing Migrating Algorithm Team To Team Adaptive–SOMA T3A. In: *2019 IEEE Congress on Evolutionary Computation (CEC).* IEEE, 2019. p. 1182-1187.

[15] MARINI, Federico; WALCZAK, Beata. Particle swarm optimization (PSO). A tutorial. *Chemometrics and Intelligent Laboratory Systems,* 2015, 149: 153-165.

[16] KOHLI, Mehak; ARORA, Sankalap. Chaotic grey wolf optimization algorithm for constrained optimization problems. *Journal of computational design and engineering*, 2018, 5.4: 458-472..

[17] BRANKE, Jürgen, et al. A multi-population approach to dynamic optimization problems. In: *Evolutionary Design and Manufacture.* Springer, London, 2000. p. 299-307.

[18] STEWART, Maurice; LEWIS, Oran T. MARINI, Federico; WALCZAK, Beata. Particle swarm optimization (PSO). A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 2015, 149: 153-165.. Gulf Professional Publishing, 2012.

[19] DEB, Kalyanmoy. Optimal design of a welded beam via genetic algorithms. *AIAA journal*, 1991, 29.11: 2013-2015.

[20] MIRJALILI, Seyedali; MIRJALILI, Seyed Mohammad; LEWIS, Andrew. Grey wolf optimizer. *Advances in engineering software*, 2014, 69: 46-61.

[21] YIN, J., et al. 2019 IEEE Congress on Evolutionary Computation (CEC). 2019.

[22] WU, Guohua; MALLIPEDDI, R.; SUGANTHAN, P. N. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2017.

[23] LI, Changhe; YANG, Shengxiang. A general framework of multipopulation methods with clustering in undetectable dynamic environments. *IEEE Transactions on Evolutionary Computation,* 2012, 16.4: 556-577.

[24] ZHU, Zhen, et al. Global replacement-based differential evolution with neighbor-based memory for dynamic optimization. *Applied Intelligence*, 2018, 48.10: 3280-3294.

[25] WANG, Yong, et al. Evolutionary dynamic constrained optimization: Test suite construction and algorithm comparisons. *Swarm and Evolutionary Computation,* 2019, 50: 100559.

[26] WANG, Guanyu; HE, Sailing. A quantitative study on detection and estimation of weak signals by using chaotic Duffing oscillators. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* 2003, 50.7: 945-953.

[27] COELHO, Ld S.; MARIANI, Viviana Cocco. Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect. *IEEE Transactions on power systems*, 2006, 21.2: 989-996.

[28] LI, Hong; JIAO, Yong-Chang; WANG, Yuping. Integrating the simplified interpolation into the genetic algorithm for constrained optimization problems. In: I*nternational Conference on Computational and Information Science.* Springer, Berlin, Heidelberg, 2005. p. 247-254.

[29] CHELLAPILLA, Kumar. Combining mutation operators in evolutionary programming. *IEEE transactions on Evolutionary Computation,* 1998, 2.3: 91-96.

[30] DOS SANTOS COELHO, Leandro; LEE, Chu-Sheng. Solving economic load dispatch problems in power systems using chaotic and Gaussian particle swarm optimization approaches. *International Journal of Electrical Power & Energy Systems,* 2008, 30.5: 297-307.

[31] YAO, Xin; LIU, Yong; LIN, Guangming. Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, 1999, 3.2: 82-102.

[32] SINGH, Dipti; AGRAWAL, Seema. Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems. *Applied Soft Computing,* 2016, 38: 1040-1048.

[33] DOS SANTOS COELHO, Leandro. Self-organizing migrating strategies applied to reliability-redundancy optimization of systems. *IEEE Transactions on Reliability*, 2009, 58.3: 501-510.

[34] DOS SANTOS COELHO, Leandro. Self-organizing migration algorithm applied to machining allocation of clutch assembly. *Mathematics and Computers in Simulation,* 2009, 80.2: 427-435.

[35] AGRAWAL, Seema; SINGH, Dipti. Modified Nelder-Mead self organizing migrating algorithm for function optimization and its application. *Applied Soft Computing,* 2017, 51: 341-350.

[36] FORTINI, Earlwood T. Dimensioning for interchangeable manufacture. *Industrial Press,* 1967.

[37] HAQ, A. Noorul, et al. Tolerance design optimization of machine elements using genetic algorithm. *The international journal of advanced manufacturing technology,* 2005, 25.3-4: 385-391.

[38] KRATOCHVÍL, Ctirad, Pavel HERIBAN. *Dynamické systémy a chaos.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky, 2010, 80 s. ISBN 978-80-214-4056-2. Dostupné z: `http://www.umt-old.fme.vutbr.cz/images/stories/opory/2009_dynamicke_systemy_a_chaos.pdf`

[39] GANDOMI, Amir Hossein; YANG, Xin-She. Benchmark problems in structural optimization. In: *Computational optimization, methods and algorithms.* Springer, Berlin, Heidelberg, 2011. p. 259-281.

[40] ALATAS, Bilal; AKIN, Erhan; OZER, A. Bedri. Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons & Fractals,* 2009, 40.4: 1715-1734.

[41] MIRJALILI, Seyedali. SCA: a sine cosine algorithm for solving optimization problems. *Knowledge-based systems,* 2016, 96: 120-133.

[42] RAY, Tapabrata; LIEW, Kim-Meow. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation,* 2003, 7.4: 386-396.

[43] ZHANG, Min; LUO, Wenjian; WANG, Xufa. Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences,* 2008, 178.15: 3043-3074.

[44] SADOLLAH, Ali, et al. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing,* 2013, 13.5: 2592-2612.

[45] COELLO, Carlos A. Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry,* 2000, 41.2: 113-127.

[46] COELLO, Carlos A. Coello; MONTES, Efrén Mezura. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics,* 2002, 16.3: 193-203.

[47] WANG, Xiaochen; YANG, Quan; ZHAO, Yuntao. Research on hybrid PSODE with three populations based on multiple differential evolutionary models. In: *2010 International Conference on Electrical and Control Engineering.* IEEE, 2010. p. 1692-1696.

[48] BEYER, Hans-Georg. *The theory of evolution strategies.* Springer Science & Business Media, 2013.

[49] SINGH, Manohar; PANIGRAHI, B. K.; ABHYANKAR, A. R. Optimal coordination of directional over-current relays using Teaching Learning-Based Optimization (TLBO) algorithm. *International Journal of Electrical Power & Energy Systems,* 2013, 50: 33-41.

[50] HE, Qie; WANG, Ling. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering applications of artificial intelligence,* 2007, 20.1: 89-99.

[51] HE, Qie; WANG, Ling. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied mathematics and computation,* 2007, 186.2: 1407-1422.

[52] RAO, R. Venkata; SAVSANI, Vimal J.; VAKHARIA, D. P. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 2011, 43.3: 303-315.

[53] KAVEH, A.; TALATAHARI, S. An improved ant colony optimization for constrained engineering design problems. *Engineering Computations,* 2010.

[54] DEEP, Kusum, et al. A new hybrid self organizing migrating genetic algorithm for function optimization. In: *2007 IEEE Congress on Evolutionary Computation.* IEEE, 2007. p. 2796-2803.

[55] DEEP, Kusum, et al. A self-organizing migrating genetic algorithm for constrained optimization. *Applied Mathematics and Computation,* 2008, 198.1: 237-250.

[56] COELLO, Carlos A. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering,* 2002, 191.11-12: 1245-1287.

[57] MORALES, Angel Kuri; QUEZADA, Carlos Villegas. A universal eclectic genetic algorithm for constrained optimization. In: *Proceedings of the 6th European congress on intelligent techniques and soft computing.* 1998. p. 518-522.

[58] DEB, Kalyanmoy; AGRAWAL, Samir. A niched-penalty approach for constraint handling in genetic algorithms. In: *Artificial Neural Nets and Genetic Algorithms.* Springer, Vienna, 1999. p. 235-243.

[59] AKHTAR, Shamim; TAI, Kang; RAY, Tapabrata. A socio-behavioural simulation model for engineering design optimization. *Engineering Optimization,* 2002, 34.4: 341-354.

[60] KAZIMIPOUR, Borhan; LI, Xiaodong; QIN, A. Kai. Initialization methods for large scale global optimization. In: *2013 IEEE Congress on Evolutionary Computation.* IEEE, 2013. p. 2750-2757.

[61] ALI, Ahmed Fouad. Hybrid simulated annealing and Nelder-Mead algorithm for solving large-scale global optimization problems. *International Journal of Research in Computer Science,* 2014, 4.3: 1.

[62] GROSAN, Crina; ABRAHAM, Ajith; HASSAINEN, Aboul Ella. A line search approach for high dimensional function optimization. *Telecommunication Systems,* 2011, 46.3: 217-243.

[63] RAJASEKHAR, Anguluri; ABRAHAM, Ajith; PANT, Millie. Levy mutated artificial bee colony algorithm for global optimization. In: *2011 IEEE International Conference on Systems, Man, and Cybernetics.* IEEE, 2011. p. 655-662.

[64] AYDILEK, Ibrahim Berkan. A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. *Applied Soft Computing,* 2018, 66: 232-249.

[65] MIRJALILI, Seyedali, et al. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software,* 2017, 114: 163-191.

[66] RAO, Ravipudi Venkata. Jaya: an advanced optimization algorithm and its engineering applications. *Cham: Springer International Publishing*, 2019.

[67] NELDER, John A.; MEAD, Roger. A simplex method for function minimization. *The computer journal,* 1965, 7.4: 308-313.

[68] CAPONETTO, Riccardo, et al. Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE transactions on evolutionary computation*, 2003, 7.3: 289-304.

[69] ÖZKAYNAK, Fatih. A novel method to improve the performance of chaos based evolutionary algorithms. *Optik,* 2015, 126.24: 5434-5438.

[70] GANDOMI, Amir Hossein, et al. Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, 2013, 18.1: 89-98.

[71] GANDOMI, Amir Hossein, et al. Chaos-enhanced accelerated particle swarm optimization. *Communications in Nonlinear Science and Numerical Simulation,* 2013, 18.2: 327-340.

[72] ARORA, Sankalap; SINGH, Satvir. An improved butterfly optimization algorithm with chaos. *Journal of Intelligent & Fuzzy Systems,* 2017, 32.1: 1079-1088.

[73] HAN, XiaoHong; CHANG, XiaoMing. An intelligent noise reduction method for chaotic signals based on genetic algorithms and lifting wavelet transforms. *Information Sciences,* 2013, 218: 103-118.

[74] HOFFMEISTER, Frank; SPRAVE, Joachim. *Problem-independent handling of constraints by use of metric penalty functions.* 1996.

[75] JOINES, Jeffrey A.; HOUCK, Christopher R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In: *Proceedings of the First IEEE Conference on Evolutionary Computation.* IEEE World Congress on Computational Intelligence. IEEE, 1994. p. 579-584.

[76] KIRKPATRICK, Scott; GELATT, C. Daniel; VECCHI, Mario P. Optimization by simulated annealing. *science*, 1983, 220.4598: 671-680.

[77] MICHALEWICZ, Zbigniew; ATTIA, Naguib. Evolutionary optimization of constrained problems. In: *Proceedings of the 3rd annual conference on evolutionary programming.* Singapore: World Scientific, 1994. p. 98-108.

[78] MICHALEWICZ, Zbigniew, et al. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering*, 1996, 30.4: 851-870.

[79] DATTA, Rituparna; DEB, Kalyanmoy; KIM, Jong-Hwan. CHIP: Constraint Handling with Individual Penalty approach using a hybrid evolutionary algorithm. *Neural Computing and Applications*, 2019, 31.9: 5255-5271.

[80] TRIVEDI, Anupam, et al. A unified differential evolution algorithm for constrained optimization problems. In: 2*017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017. p. 1231-1238.

[81] TVRDÍK, Josef; POLÁKOVÁ, Radka. A simple framework for constrained problems with application of L-SHADE44 and IDE. In: *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017. p. 1436-1443.

[82] ZAMUDA, Aleš. Adaptive constraint handling and success history differential evolution for CEC 2017 constrained real-parameter optimization. In: *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017. p. 2443-2450.

[83] POLAKOVA, Radka. L-SHADE with competing strategies applied to constrained optimization. In: *2017 IEEE congress on evolutionary computation (CEC)*. IEEE, 2017. p. 1683-1689.

[84] TRIVEDI, Anupam; SRINIVASAN, Dipti. Empirical Investigations Into the Composite Differential Evolution on CEC 2017 Constrained Optimization Problems. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2018. p. 280-285.

[85] TANABE, Ryoji; FUKUNAGA, Alex. Success-history based parameter adaptation for differential evolution. In: *2013 IEEE congress on evolutionary computation.* IEEE, 2013. p. 71-78.

[86] CALVO, Borja; SANTAFÉ RODRIGO, Guzmán. scmamp: Statistical comparison of multiple algorithms in multiple problems. *The R Journal,* Vol. 8/1, Aug. 2016, 2016.

[87] IMAN, Ronald L.; DAVENPORT, James M. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics-Theory and Methods,* 1980, 9.6: 571-595.