FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Urychlení evoluční optimalizace pomocí response surface modelů na benchmarkových funkcích z praxe
Student:	Bc. Marek Hanuš
Vedoucí:	Ing. Zbyněk Pitra
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Evoluční algoritmy lze při řešení úloh s drahými black-box cílovými funkcemi efektivně urychlit aproximací těchto funkcí pomocí regresních modelů. Často používaným typem regresních modelů jsou response surface modely, jejichž výzkum v oblasti urychlení evoluční optimalizace empirických funkcí však teprve začíná.

1) Řešitel se nejdříve důkladně seznámí s regresními response surface modely a také s principy optimalizace pomocí evolučních algoritmů. Bude přitom věnovat pozornost i urychlení evoluční optimalizace empirických funkcí pomocí regresního modelu cílové funkce.
2) Ve vývojovém prostředí Matlab naimplementuje response surface modely a jejich zobecnění.
3) Zakomponuje naimplementované response surface modely jako náhradní modely pro evoluční algoritmus DTS-CMA-ES.
4) Urychlení evoluční optimalizace otestuje na benchmarkových funkcích z frameworku pro black-box optimalizaci COCO, který rozšíří o sadu benchmarkových funkcí představujících problémy z praxe (MLDA).

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Karel Klouda, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 30. října 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Urychlení evoluční optimalizace pomocí response surface modelů na benchmarkových funkcích z praxe

Bc. Marek Hanuš

Katedra aplikované matematiky
Vedoucí práce: Ing. Zbyněk Pitra

28. května 2020

Poděkování

Děkuji mému vedoucímu, Ing. Zbyňku Pitrovi, za ochotu a trpělivost při vedení diplomové práce. Dále děkuji organizaci MetaCentrum za možnost využití vypočetního clusteru.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 28. května 2020 .....

Odkaz na tuto práci

Hanuš, Marek. *Urychlení evoluční optimalizace pomocí response surface modelů na benchmarkových funkcích z praxe*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato práce se věnuje optimalizaci drahých black-box funkcí. Zaměřuje se na urychlení evoluční strategie CMA-ES pomocí náhradní modelů založených zejména na response surface modelech. Náhradní modely umožňují ušetření počtu potřebných ohodnocení optimalizovanou funkcí. Úspěšné verze CMA-ESu s náhradními modely jsou popsány a jejich evoluční kontroly a náhradní modely jsou permutovány mezi sebou. Vzniklé permutace různých evolučních kontrol a náhradních modelů jsou posléze otestovány na frameworku BBOB, který byl obohacen o nové funkce založené na reálných problémech. Výsledky testování slouží k posouzení míry podílu jednotlivých evolučních kontrol a náhradních modelů na urychlení původního algoritmu.

Klíčová slova Black-box, Optimalizace, CMA-ES, Náhradní modely, Response surface modely

Abstract

This thesis investigates optimisation of expensive black-box functions. It focuses on extending CMA-ES with surrogate models, which are mostly based on response surface modeling. Surrogate models are used for reducing required number of evaluations of function that is being optimised. Successful versions

of CMA-ES with surrogate models are described and theirs evolution controls are permutated together with surrogate models between each other. Created permutations of different evolution controls and surrogate models are then tested on BBOB framework, which was enhanced with new functions based on real world problems. Results of testing are used to decide how big of a role each evolution control and surrogate model plays in saving evaluations.

Keywords Black-box, Optimisation, CMA-ES, Surrogate models, Response surface models

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Optimalizace black-box funkcí je odvětvím umělé inteligence, které se snaží o nalezení reálných vstupů black-box funkce, jejichž funkční hodnota se nachází v optimu. Black-box funkce jsou specifické tím, že jedinou informací dostupnou v optimalizaci, je vypočtená funkční hodnota. Tato práce se zaměřuje na tzv. drahé black-box funkce, u kterých je samotný výpočet funkční hodnoty nákladný, a proto je cílem kromě zmíněného nalezení optima také minimalizace počtu potřebných ohodnocení touto funkcí.

Existuje celá řada algoritmů využívaných k řešení tohoto optimalizačního problému. Tato práce se zabývá jedním z nich - Covariance Matrix Adaptation Evolution Strategy(CMA-ES), který je v současnosti považován za nejlepší algoritmus v oblasti spojité black-box optimalizace. Jedná se o evoluční strategii, která generuje vstupy pro black-box funkci vzorkováním z normálního rozdělení a snaží se upravovat parametry zmíněného rozdělení tak, aby měly nově vygenerované vstupy menší funkční hodnotu.

Protože vyhodnocení black-box funkce bývají často časově nebo finančně náročná a CMA-ES vyžaduje těchto vyhodnocení mnoho, vznikla celá řada jeho modifikací, které se snaží toto množství snížit. Tento proces zahrnující konstrukci tzv. náhradního modelu, regresního modelu z dat ohodnocených originální black-box funkcí, a rozhodování o četnosti využití tohoto modelu namísto skutečné funkce se nazývá řízení evoluce.

Jak náhradní model, tak řízení evoluce přispívají určitou měrou k urychlení původního algoritmu. Jedním ze základních modelů jsou tzv. response surface modely založené zejména na polynomiální regresi. Tento základní model se může jevit jako velice jednoduchý, avšak nedávné výsledky naznačují, že ve spojení s algoritmem CMA-ES je schopen dosáhnout lepších výsledků než složitější modely založené například na Gaussovských procesech. Tato práce se snaží přispět k nalezení odpovědi na otázku, zda jsou tyto výsledky zásluhou response surface modelu nebo toho, jakým způsobem ho řízení evoluce využívá.

Práce je strukturovaná do tří kapitol. První část vymezuje její cíle.

Druhá kapitola nejdříve představuje optimalizaci black-box funkcí spolu s evolučními algoritmy, které jsou k optimalizaci často používány. Následuje popis populární evoluční strategie Covariance Matrix Adaptation Evolution Strategy. Poté je popsáno náhradní modelování s důrazem na response surface modely a nakonec jsou zmíněny algoritmy, které se snaží skloubit CMA-ES a náhradní modelování.

Třetí kapitola seznamuje čtenáře s metodologií využitou k dosažení cílů práce a vzniklou implementací. Posléze je představen testovací framework BBOB rozšířený o nové funkce založené na reálných problémech spolu s konfigurací využitou při testování. Celou práci uzavírá rozbor výsledků.

Kapitola 1

Cíl práce

Hlavním cílem této práce je implementace vybraných response surface modelů, jejich následná integrace do již existujícího algoritmu DTS-CMA-ES a otestování frameworkem BBOB, který bude rozšířen o funkce založené na reálných problémech. Aby bylo možné tento cíl splnit, tak je potřeba dokončit několik různých podúkolů:

- Seznámení se s již existujicím frameworkem napsaným v Matlabu, který obsahuje CMA-ES spolu s DTS.
- Studium, implementace modelů z algoritmů lmm-CMA-ES a lq-CMA-ES a jejich následná integrace do frameworku.
- Vytvoření nových funkcí a jejich integrace do testovacího frameworku BBOB.
- Otestovaní za pomoci frameworku BBOB.

Kapitola 2

Teoretický základ

Kapitola s názvem Teoretický základ seznamuje čtenáře s teorií využitou v této diplomové práci. Nejdříve jsou zde představené black-box funkce, jejich omezení a cíle, kterých se snažíme dosáhnout při optimalizaci. Následující sekce se zabývá různými druhy evolučních algoritmů, jejich využitím, výhodami i nevýhodami. Algoritmus Covariance Matrix Adaptation Evolution Strategy je popsán v třetí sekci. Poté se práce věnuje náhradnímu modelování včetně vybraných typů modelu s důrazem na response surface modelování. Pátá a poslední sekce teoretické části obsahuje tři vybrané varianty algoritmu CMA-ES využívající náhradní model: Lokální MetaModel CMA-ES, Dvojitě trénovaný CMA-ES a Lineárně kvadratický CMA-ES.

2.1 Black-box optimalizace

Optimalizace black-box funkcí je odvětví umělé inteligence zabývající se hledáním globálního optima (v této práci se zaměříme pouze na hledání minima) tzv. *black-box* funkcí. Formálně hledáme $\mathbf{x} \in \mathbb{R}^n, n \in \mathbb{N}^1$ takové, pro které platí:

$$\arg\min_{\mathbf{x}}(f(\mathbf{x})), \tag{2.1}$$

kde $f : \mathbb{R}^n \to \mathbb{R}^m, m \in \mathbb{N}^1$ je zkoumaná black-box funkce. Za black-box funkci považujeme takovou funkci, u které známe pouze její vstupy a výstupy. Jak již název napovídá, black-box funkci si můžeme představit jako černou krabičku, do které vložíme zkoumané vstupy a ona nám vrátí odpovídající výstupy. O matematické popisu nebo výpočtech potřebných k získání funkčních hodnot nevíme nic. V kontextu optimalizace black-box funkcí se pro funkční hodnotu používá pojem *fitness*.

Z výše popsaných pravidel vyplývá, že se můžeme na každou funkci dívat jako na black-box. Například se může jednat o jednoduché funkce typu sčítání, kdy vstupem funkce jsou dvě čísla a výstupem je jejich součet, nebo i složitější,

jako je předpověď počasí, která na základě historie a momentálních podmínek (vstupy) předpovídá počasí (výstup).

V praxi můžeme narazit na dva druhy problémů. V jednom případě nám není hodnota optimální fitness známa a snažíme se nalézt vstupy, které mají fitness optimálnější než předem stanovenou hodnotu, nebo hodnotu globálního optima známe a máme za cíl nalézt vstupy, jejichž fitness je v předem dané maximální vzdálenosti od fitness optimální.

Nalezení globálního optima pro vícerozměrné reálné funkce ($f : \mathbb{R}^n \to \mathbb{R}^m$, kde $m > 1$) je většinou velmi obtížné, nebo dokonce i výpočetně nemožné a není předmětem zájmu této práce. Tato práce se zabývá optimalizací funkcí, jejichž ohodnocení (výpočet funkční hodnoty) je drahé (ať už výpočetně nebo finančně). Příkladem takové funkce může být výnosnost chemické reakce při hledání optimálního složení katalyzátoru [5]. Optimalizace se v kontextu drahých funkcí snaží o jediné, a tím je minimalizace počtu potřebných ohodnocení black-box funkcí k nalezení uspokojivé fitness. Algoritmus, který dosáhne požadovaného řešení s menším počtem ohodnocení je považován za lepší.

Nahlížení na problémy optikou black-box funkce má své výhody, ale i nevýhody. Mezi výhody patří fakt, že nám stačí vyvinout pouze jeden obecný algoritmus pro hledání optima a ten poté bude fungovat pro všechny black-box funkce. Můžeme se tedy primárně soustředit na zlepšování zmíněného algoritmu a nezabývat se tím, které black-box funkce bude algoritmus optimalizovat. Zároveň se ale jedná i o negativum, protože z principu problému nemůžeme využít případné znalosti o zkoumané black-box funkci. Můžeme například o funkci vědět, že optimum neleží na žádné z os, nebo že tvar funkce je podobný elipse. Algoritmus optimalizovaný pro tuto konkrétní funkci by s největší pravděpodobností nalezl optimum rychleji [6].

Existuje mnoho různých druhů algoritmů pro optimalizaci black-box funkcí, od použití náhodného výběru (generování náhodných řešení) [7], přes genetické algoritmy [8, 9] až po optimalizace hejnem částic [10], mravenčí kolonie [11] nebo evoluční strategie [12].

2.2 Evoluční algoritmy

Evoluční algoritmy tvoří skupinu algoritmů, které se snaží iterativně zlepšovat *generaci* (množinu) *jedinců* (jednotlivá řešení). V každé iteraci nejdříve vygenerují $\lambda \in \mathbb{N}$ jedinců, každého jedince ohodnotí pomocí fitness a vyberou μ nejlepších (*selekce*), ze kterých je následně vytvořena nová generace pomocí operátorů *křížení* (nového jedince vytvořím zkombinováním ostatních) a *mutace* (s určitou pravděpodobností jedince náhodně poupravím). Výběr jedinců s lepší fitness umožňuje postupné zlepšování následujících generacích. Tato myšlenka je založena na Darwinově evoluční teorii, která říká, že se každý druh generaci od generace postupně mění a přizpůsobuje tak, aby měl větší šanci na přežití [13]. Pokud definici šance na přežití převedeme do světa black-

box optimalizace a budeme za ni považovat fitness funkci jedince, tak se využití evolučních algoritmů nabízí jako jedna z možností pro nalezení optima.

Základní princip evolučních algoritmů je vyjádřen v Algoritmu 1. Rozdíly mezi jednotlivými algoritmy spočívají ve způsobu reprezentace jedince nebo v operátorech selekce, křížení a mutace.

Input: λ (počet jedinců v generaci), podmínka pro ukončení algoritmu
1 Vytvoření nulté generace λ jedinců - většinou náhodně;
while *Není splněna podmínka pro ukončení* **do**
2 Ohodnoť jedince v generaci;
3 Vyber jedince s co nejlepší fitness (selekce);
4 Pomocí křížení, mutace nebo kopírování vybraných jedinců vytvoř novou generaci;
end
Result: Jedinec s největš fitness

Algoritmus 1: Obecný průběh evolučního algoritmu

Mezi základní evoluční algoritmy patří *genetické algoritmy*. Ty nahlížejí na jedince jako na binární řetězec konstantní délky. Genom jedince je tedy specifický pro vybraný problém. Křížení dvou vybraných jedinců většinou probíhá náhodnou kombinací jejich genomů a mutace s určitou pravděpodobností zneguje jeden nebo více bitů nezávisle na ostatních.

Genetické programování se snaží o zdokonalení populace počítačových programů určených pro předem daný úkol. Jednotlivé jedince většinou reprezentuje ve stromové struktuře a definuje v ní i jednotlivé operace. Na rozdíl od ostatních evolučních algoritmů je struktura a počet proměnných jednotlivce variabilní. Pokud je počet proměnných předem znám, tak algoritmus optimalizuje pouze jejich číselné hodnoty. Když je ale proměnná i struktura a počet proměnných jedince, tak to většinou vede k důkladnějšímu prohledávání topologií samotných (např. může se ukázat, že přidání další proměnné zjednoduší optimalizaci). Genetické programování se většinou snaží o minimalizaci velikosti jedince spolu s fitness funkcí, aby se zamezil exponenciální růst velikosti jedinců.

Evoluční programování bylo vyvinuto pro simulaci evoluce jako prostředku pro vytvoření umělé inteligence. Inteligence byla vnímána jako schopnost systému upravovat svoje chování, aby byl splněn předem daný cíl. Jako první ho zmiňuje Lawrence Fogel, který využil populaci konečných automatů, jejichž chování se snažil upravit tak, aby předpovídali posloupnost symbolů. Primárním operátorem je mutace, která přidává či ubírá stavy automatu, mění přechodové funkce či upravuje vstupní a výstupní stav [14].

Poslední zde zmíněnou skupinou algoritmů jsou *Evoluční strategie*. Genomem je vektor $\mathbf{x} \in \mathbb{R}^n$, kde $n \in \mathbb{N}$ je dimenze zkoumané black-box funkce. Náhodnost získávají převážně vzorkováním z normálního rozdělení a v každé

následující generaci se snaží posunout parametry zmíněného rozdělení tak, aby nově vygenerovaní jedinci měli lepší fitness. Evoluční strategie využívají převážně operátory selekce a mutace. V každé generaci je vybráno μ nejlepších jedinců, podle kterých se posune střední hodnota a variance rozdělení. Průběh evoluční strategie tedy vypadá následovně. Dokud není splněno potřebné kritérium pro ukončení, tak opakuj:

1. Vytvoř novou generaci λ možných řešení (jedinců) pomocí vzorkování z normálního rozdělení

2. Ohodnoť jedince

3. Uprav parametry normálního rozdělení

Mezi běžná kritéria ukončení může patřit dosažení předem daného počtu iterací nebo například rozptyl vygenerovaných řešení. Aplikaci naleznou evoluční strategie především v optimalizaci funkcí reálných proměnných [15].

2.3 Covariance Matrix Adaptation Evolution Strategy - CMA-ES

Algoritmus CMA-ES [2] je považován za jeden z nejvýznamnějších, ne-li nejvýznamnější algoritmus pro řešení black-box problémů současnosti a vychází z principu evoluční strategie. Historicky se jedná o první algoritmus, který úspěšně upravuje jak střední hodnotu tak i kovarianční matici normálního rozdělení.

Algoritmus by se dal popsat jako robustní metoda lokálního prohledávání. Robustnost spolu s invariancí vůči škálování či rotacím je způsobena stylem výběru jedinců. Po získání fitness hodnoty celé generace jsou jedinci seřazeni v závislosti na jejich fitness a CMA-ES dále pracuje pouze s jejich pořadím. Nevýhodou je občasná pomalejší konvergence.

CMA-ES využívá dva hlavní principy pro úpravu parametrů. Prvním je metoda maximální věrohodnosti, která představuje snahu o zvýšení pravděpodobnosti, že bude vygenerován jedinec s lepší fitness hodnotou. Střední hodnota rozdělení je upravována tak, aby byla co možná největší pravděpodobnost vygenerování jedinců s optimálnější fitness. Úpravami kovarianční matice se algoritmus snaží o častější využití směru, který byl v předchozích iteracích výhodný. Dohromady se dá na úpravu střední hodnoty a kovarianční matice nahlížet jako na metodu gradientního sestupu. Druhým principem je postupné ukládání změn kovarianční matice pro vytvoření tzv. evolučních cest, které jsou následně využity pro lepší úpravy variance ve správném směru a řízení velikosti kroku algoritmu.

CMA-ES ve formě pseudokódu je uveden v Algoritmu 2.

Input: velikost kroku $\sigma^{(0)} \in \mathbb{R}_+$, střední hodnota $m^{(0)} \in \mathbb{R}^n, n \in \mathbb{N}^1$

1 Inicializace všech potřebných parametrů

while *Není splněna podmínka pro ukončení, pro generace* $g = 0, 1, 2, ..$ **do**

2 Vytvoření generace $\mathbf{x}_k^{(g+1)} = \mathcal{N}(\mathbf{m}^{(g)}, (\sigma^{(g)})^2\mathbf{C}^{(g)})$ pro $k = 1, ..., \lambda$

3 Ohodnoť a seřad jedince v generaci

4 Selekce a křížení $\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} \omega_i \mathbf{x}_{i:\lambda}$

5 Rozdíl středních hodnot pro evoluční cestu σ $\mathbf{p}_\sigma^{(g+1)} = (\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)})/(\sigma^{(g)}\sqrt{\mathbf{C}^{(g)}})$

6 Uprav velikost kroku $\sigma^{(g+1)} = ||\mathbf{p}_\sigma^{(g+1)}||$

7 Rozdíl středních hodnot pro evoluční cestu C $\mathbf{p}_c^{(g+1)} = (\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)})/\sigma^{(g)}$

8 Uprav kovarianční matici aktualizací první úrovně podle $\mathbf{p}_c^{(g+1)}$ a úrovně μ podle rozdílu $(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}), i = 1, ..., \mu$

end

Result: $\hat{\mathbf{x}}^{opt}$Jedinec s největším fitness

Algoritmus 2: Zjednodušený algoritmus CMA-ES

2.3.1 Vzorkování

Každá iterace algoritmu začíná vytvořením generace, která se skládá z λ jedinců. Tito jedinci jsou vygenerováni vzorkováním z normálního rozdělení pomocí následujícího předpisu:

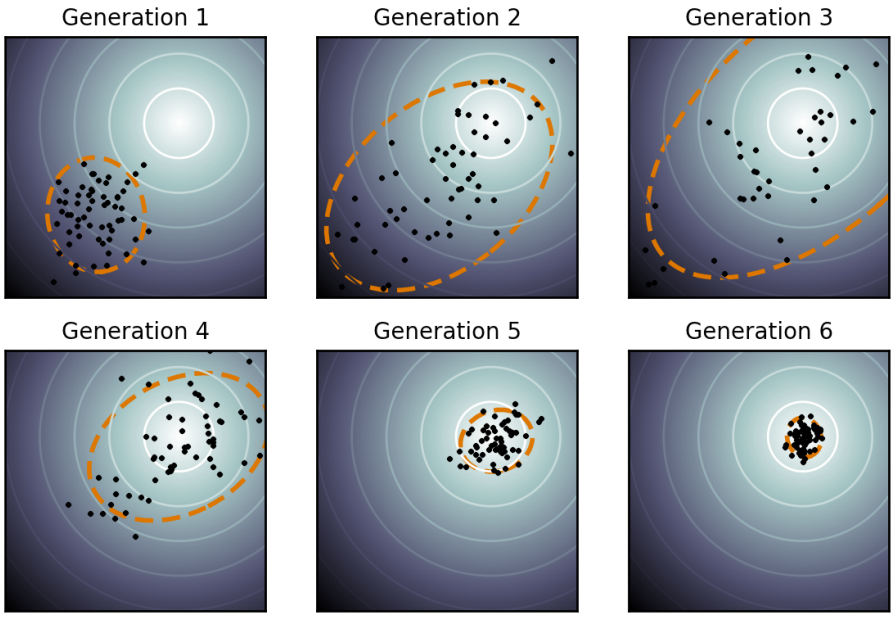
$$\mathbf{x}_k^{(g+1)} = \mathbf{m}^{(g)} + \sigma^{(g)}\mathcal{N}(0, \mathbf{C}^{(g)}) \tag{2.2}$$

kde:

- $\lambda >= 2$ je počet potomků v generaci,
- $k = 1, ..., \lambda$,
- $\mathbf{x}_k^{(g+1)}$ je k-tý potomek z generace $g + 1$,
- $\mathbf{m}^{(g)}$ je střední hodnota rozdělení v generaci g,
- $\sigma^{(g)}$ je směrodatná odchylka (nazývaná také *velikost kroku*) generace g,
- $\mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)})$ je vícerozměrné normální rozdělení se střední hodnotou $\mathbf{0}$ a kovariační maticí $\mathbf{C}$.

2.3.2 Úprava parametrů

Pro generování jedinců, kteří budou mít lepší fitness než jejich předchůdci, je potřeba upravit střední hodnotu $\mathbf{m}$, velikost kroku σ a kovariační matici $\mathbf{C}$. Příklad průběhu algoritmu je znázorněn na Obrázku 2.1.



Obrázek 2.1: Znázornění průběhu algoritmu CMAES při hledání minima v 2D prostoru [1].

Nejdříve je potřeba seřadit nové jedince podle jejich odpovídající fitness. Algoritmus CMA-ES využívá pouze pořadí jedinců a ignoruje absolutní hodnoty jejich fitness. Následně je vybráno nejlepších μ jedinců a jim přiřazena váha splňující předpis:

$$\sum_{i=1}^{\mu} \omega_i = 1, \tag{2.3}$$

$$\omega_1 \geq \omega_2 \geq .. \geq \omega_\mu > 0, \tag{2.4}$$

kde:

- $\mu \leq \lambda$ je počet vybraných bodů,
- $\omega_{i=1,...,\mu}$ jsou pozitivní koeficienty váhy vybraných bodů.

2.3.3 Úprava střední hodnoty

Výpočet střední hodnoty následující generace je poměrně přímočarý. Jedná se o posun současné střední hodnoty ve směru váženého průměru μ nejlepších jedinců. Přesněji:

$$\mathbf{m}^{(g+1)} = \mathbf{m}^{(g)} + c_m \sum_{i=1}^{\mu} \omega_i (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)},) \tag{2.5}$$

kde:

- $\mathbf{m}^{(g)}$ je střední hodnota pro generaci g,
- $c_m \leq 1$ je rychlost učení, většinou 1 (hodnoty menší než 1 můžou být užitečné pro funkce obsahující šum),
- w_i je váha nejlepšího vybraného i-tého jedince,
- $\mathbf{x}_{i:\lambda}^{(g+1)}$ je nejlepší vybraný i-tý jedinec z populace $g+1$ o velikost λ,
- $\mu < \lambda$ je počet vybraných bodů určených ke křížení.

2.3.4 Adaptace kovarianční matice

Existují 2 způsoby jak vypočíst kovarianční matici pro následující generaci. Můžeme se pokusit ji v každé iteraci zcela odhadnout, nebo postupně upravovat kovarianční matici z předchozí generace.

2.3.4.1 Úplný odhad kovarianční matice

Předpokládejme, že stávající generace obsahuje dostatek informací potřebných k spolehlivému odhadu kovarianční matice. Potom můžeme znovu odhadnout originální kovarianční matici $\mathbf{C}^{(g)}$ pomocí bodů $\mathbf{x}_1^{(g+1)}, ..., \mathbf{x}_\lambda^{(g+1)}$ získaných vzorkováním. Pro odhad lepší (lepší v tom smyslu, že vygenerované body mají v průměru optimálnější fitness) kovarianční matice je možné použít vážený selekční mechanismus:

$$\mathbf{C}_\mu^{(g+1)} = \sum_{i=1}^{\mu} w_i (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})(\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})^\top. \tag{2.6}$$

Vzorkování z $\mathbf{C}_\mu^{(g+1)}$ generuje body blížící se k úspěšným jedincům generace. Problémem tohoto postupu je, že aby generace obsahovala dostatečné množství informací pro spolehlivý odhad kovarianční matice, tak je zapotřebí vygenerovat a ohodnotit velké množství bodů (jak velké závisí na dimenzi zkoumané funkce a kovarianční matici). V oblasti optimalizace black-box funkcí si takové množství ohodnocení nemůžeme dovolit, a proto je využita druhá varianta adaptace kovarianční matice.

2.3.4.2 Postupná úprava kovarianční matice

Pro urychlení úprav kovarianční matice je zapotřebí co nejmenšího počtu jedinců v generaci, což ale znamená, že není možné odhadnout kovarianční matici pouze z jedné generace. Řešením je využít informace i z předchozích

generací. Po dostatečném množství iterací je možné využít průměr odhadu kovariančních matic všech předchozích generací:

$$\mathbf{C}^{(g+1)} = \frac{1}{g+1}\sum_{i=0}^{g}\frac{1}{{\sigma^{(i)}}^2}\mathbf{C}_{\mu}^{(i+1)}. \tag{2.7}$$

V takto zadefinované rovnici mají všechny dřívější kovarianční matice stejnou váhu. Pro přiřazení větší váhy novějším generacím se využívá exponenciální vyrovnávání. Zvolením $\mathbf{C}^{(0)} = \mathbf{I}$ jako jednotkové matice a koeficient učení $0 < c_\mu \leq 1$ dostáváme:

$$\mathbf{C}^{(g+1)} = (1-c_\mu)\mathbf{C}^{(g)} + c_\mu\frac{1}{{\sigma^{(g)}}^2}\mathbf{C}_{\mu}^{(g+1)} \tag{2.8}$$

$$= (1-c_\mu)\mathbf{C}^{(g)} + c_\mu\sum_{i=1}^{\mu} w_i\mathbf{y}_{i:\lambda}^{(g+1)}{\mathbf{y}_{i:\lambda}^{(g+1)}}^{\top} \tag{2.9}$$

$$= \mathbf{C}^{(g)^{(1/2)}}(\mathbf{I} + c_\mu\sum_{i=1}^{\mu} w_i(\mathbf{z}_{i:\lambda}^{(g+1)}{\mathbf{z}_{i:\lambda}^{(g+1)}}^{\top} - \mathbf{I}))\mathbf{C}^{(g)^{1/2}}, \tag{2.10}$$

kde:

- $\mathbf{z}\text{'}_{i:\lambda}^{(g+1)} = (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)})/\sigma^{(g)}$,
- $\mathbf{z}_{i:\lambda}^{(g+1)} = \mathbf{C}^{(g)^{-1/2}}\mathbf{z}\text{'}_{i:\lambda}^{(g+1)}$.

Tento styl úpravy kovariační matice je nazýván stupně μ (*rank-μ-update*). Správná volba koeficientu učení c_μ je velmi důležitá. Malé hodnoty vedou k pomalému učení a moc velké zase k úplnému neúspěchu a zdegenerování kovarianční matice.

Druhou možností jak adaptovat kovarianční matici je úprava 1. stupně (*rank-1-update*), který k adaptaci přistupuje opačně. Místo toho, aby využíval všechny vybrané body, tak upravuje kovarianční matici pouze pomocí jednoho bodu.

Vícerozměrné normální rozdělení lze kromě kovarianční matice modelovat i jinak. Nechť existují vektory $\mathbf{y}_1, ..., \mathbf{y}_{g_0} \in \mathbb{R}^n, g_0 \leq n$ a značení $\mathcal{N}(0,1)$ určuje nezávislá náhodná čísla z normálního rozdělení se střední hodnotou 0 a variancí 1. Poté

$$\mathcal{N}(0,1)\mathbf{y}_1 + ... + \mathcal{N}(0,1)\mathbf{y}_{q_0} \sim N(0, \sum_{i=1}^{q_0}\mathbf{y}_i\mathbf{y}_i^{\top}) \tag{2.11}$$

je normálně rozdělený náhodný vektor se střední hodnotou 0 a kovarianční maticí $\sum_{i=1}^{g_0}\mathbf{y}_i\mathbf{y}_i^{\top}$. Rozdělení $\mathcal{N}(0,1)\mathbf{y}_i \sim \mathcal{N}(0, \mathbf{y}_i\mathbf{y}_i^{\top})$ generuje vektor $\mathbf{y}_i$ s maximální pravděpodobností, když se vezmou v potaz všechny normální rozdělení s nulovým průměrem.

Zjednodušením úpravy stupně μ a zadefinováním $\mathbf{y}_{g+1} = \frac{\mathbf{x}_{1:\lambda}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}$ dostáváme úpravu stupně 1

$$\mathbf{C}^{(g+1)} = (1 - c_1)\mathbf{C}^{(g)} + c_1 \mathbf{y}_{g+1} + \mathbf{y}_{g+1}^{\top}, \tag{2.12}$$

která přidává $\mathbf{y}_{g+1}$ do kovarianční matice a tím zvyšuje šanci na jeho vygenerování v příštích generacích.

Úpravy kovarianční matice tvoří tzv. *evoluční cestu* $\mathbf{p}_c \in \mathbb{R}^n$, kterou vypočteme následovně:

$$\mathbf{p}_c^{(g+1)} = (1 - c_c)\mathbf{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{eff}}\frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}, \tag{2.13}$$

kde

- $\mathbf{p}_c^{(g)} \in \mathbb{R}^n$ je evoluční cesta v generaci g,
- $c_c \leq 1$ je konstanta udávající zpětný časový horizont evoluční cesty $\mathbf{p}_c$, který obsahuje přibližně 63% celkové váhy.

Evoluční cesta pro nultou generaci je rovna 0. Úprava 1. stupně s využitím evoluční cesty vypadá následovně:

$$\mathbf{C}^{(g+1)} = (1 - c_1)\mathbf{C}^{(g)} + c_1 \mathbf{p}_c^{(g+1)} \mathbf{p}_c^{(g+1)^{\top}}. \tag{2.14}$$

Běžně využívanou hodnotou je $c_1 = 2/n^2$.

2.3.4.3 Konečná forma úpravy kovarianční matice

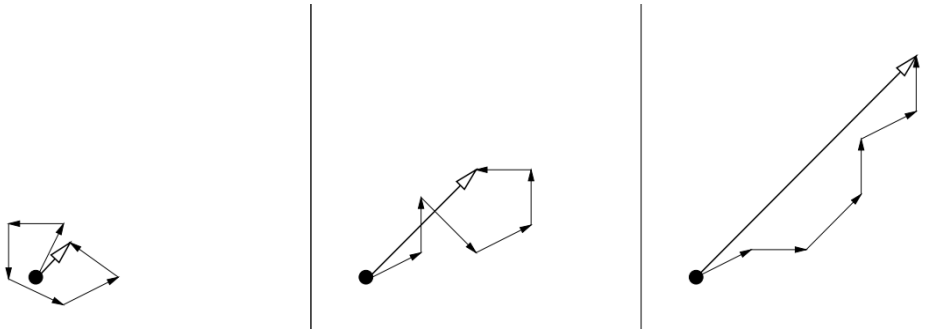
Spojením úprav stupňů 1 a μ získáváme finální předpis pro úpravu kovarianční matice

$$\mathbf{C}^{(g+1)} = (1 - c_1 - c_\mu)\mathbf{C}^{(g)} + c_1 \underbrace{\mathbf{p}_c^{(g+1)} \mathbf{p}_c^{(g+1)^{\top}}}_{\text{úprava stupně 1}} + c_\mu \underbrace{\sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}^{(g+1)} (\mathbf{y}_{i:\lambda}^{(g+1)})^{\top}}_{\text{úprava stupně } \mu}, \tag{2.15}$$

kde

- $c_1 = 2/n^2$,
- $c_\mu = \min(\mu_{eff}/n^2, 1 - c_1)$,
- $\mathbf{y}_{i:\lambda}^{(g+1)} = (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}/\sigma^{(g)})$.

Tato forma adaptace kovarianční matice využívá výhod úprav obou stupňů. Na jednu stranu jsou využity informace obsažené v populaci jedné generace pomocí úpravy stupně μ, které jsou důležité pro populace s velkým počtem jedinců. Na druhou stranu je použita i korelace mezi generacemi díky evolučním cestám, které jsou efektivní převážně pro populace s malým počtem jedinců.



Obrázek 2.2: Ukázka evolučních cest, které vznikly z 6 kroků stejné délky [2].

2.3.5 Úprava velikosti kroku

Adaptace kovarianční matice představená v předchozí sekci se nezabývá úpravou velikosti kroku σ. Existují dva různé důvody proč spolu s kovarianční maticí $\mathbf{C}^{(g)}$ upravovat i velikost kroku:

1. Není možné správně odhadnout velikost kroku pomocí samostatné adaptace kovarianční matice, zvláště když je μ_{eff} větší než jedna.
2. Největší spolehlivá hodnota konstanty rychlost učení c_σ je moc pomalá aby dosáhla podobných výsledků jako úprava velikosti kroku.

Úprava velikost kroku využívá evoluční cestu $\mathbf{p}_\sigma^{(g)}$ a může být využita nezávisle na adaptaci kovarianční matice. Úprava funguje na následujících principech:

- Když je délka evoluční cesty malá, tak jednotlivé kroky ukazují opačným směrem a navzájem se vyruší → velikost kroku by měla být menší.
- Když je délka evoluční cesty velká, tak jednotlivé kroky ukazují podobným směrem, což znamená, že stejnou vzdálenost by bylo možné překonat pomocí méně větších kroků → velikost kroku by měla být vyšší.
- Když je délka evoluční cesty střední, nic se nemění.

Jestli je délka evoluční cesty velká nebo malá zjistíme porovnáním s délkou evoluční cesty, která vznikla náhodným výběrem, protože taková evoluční cesta neobsahuje žádnou korelaci mezi vybranými body. Pokud je tedy hodnota délky reálné evoluční cesty větší než té náhodné, tak selekce bodů vybraných ke křížení způsobila korelaci mezi body a velikost kroku by měla být zvětšena nebo naopak.

Výpočet evoluční cesty pro velikost kroku je následujicí:

$$\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}\mathbf{C}^{(g)^{-1/2}}}\frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}} \qquad (2.16)$$

kde:

- $\mathbf{p}_\sigma^{(g)} \in \mathbb{R}^n$ je evoluční cesta velikosti kroku v generaci g,
- $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$,
- $c_\sigma < 1$ je konstanta udávající zpětný časový horizont evoluční cesty $\mathbf{p}_\sigma^{(g)}$, který obsahuje přibližně 63% celkové váhy,
- $\sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}$ je normalizační konstanta,
- $\mathbf{C}^{(g)^{-1/2}} = \mathbf{B}^{(g)}\mathbf{D}^{(g)^{-1}}\mathbf{B}^{(g)^\top}$, kde $\mathbf{C}^{(g)} = \mathbf{B}^{(g)}(\mathbf{D}^{(g)})^2\mathbf{B}^{(g)^\top}$ je rozložení matice $\mathbf{C}^{(g)}$ na vlastní vektory, $\mathbf{B}^{(g)}$ je ortonormální báze vektorů a elementy v diagonální matici $\mathbf{D}^{(g)}$ jsou odmocniny odpovídajících vlastních čísel.

Úprava velikosti kroku využívá evoluční cestu následovně:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{p}_\sigma^{(g+1)}||}{E||\mathcal{N}(\mathbf{0},\mathbf{I})||} - 1\right)\right) \tag{2.17}$$

kde

- d_σ je tlumící parametr, který škáluje velikost změn σ
- $\mathbb{E}||\mathcal{N}(\mathbf{0},\mathbf{I})||$ je předpokládaná hodnota euklidovské normy z náhodného vektoru $\mathcal{N}(\mathbf{0},\mathbf{I})$. Pokud je $||\mathbf{p}_\sigma^{(g+1)}|| = E||\mathcal{N}(\mathbf{0},\mathbf{I})||$, tak je velikost kroku neměnná, jinak se mění.

2.4 Náhradní modelování

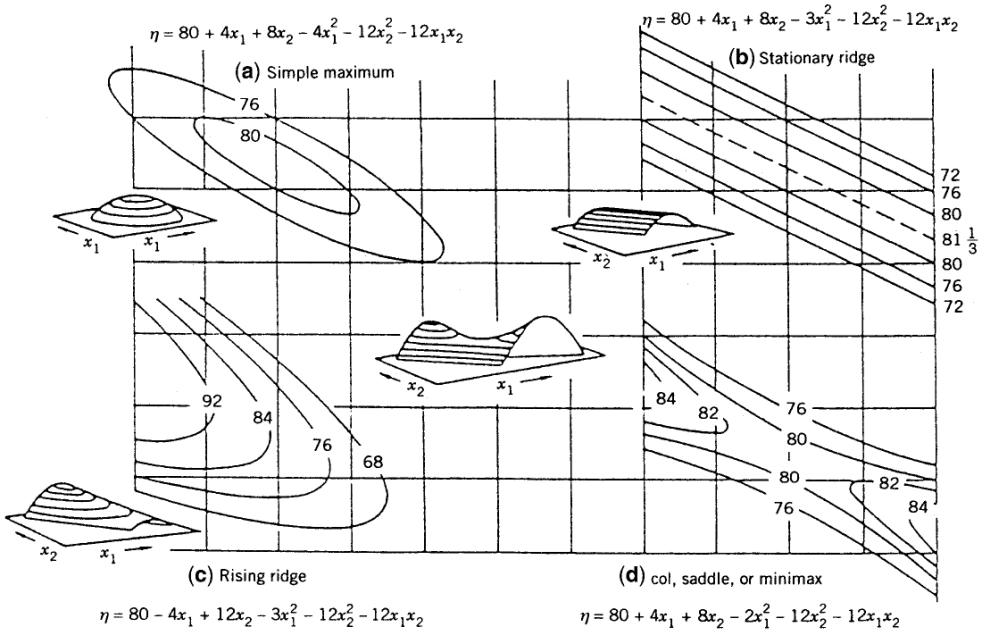
Náhradní modelování je obecné označení procesu, při kterém se snažíme nahradit jednu funkci nějakou jinou (*náhradním modelem*). Jedná se o funkci

$$f_M : \mathbb{R}^a \to \mathbb{R}^b, \tag{2.18}$$

kde a a b odpovídají dimenzím definičního oboru a funkčních hodnot fuknce originální. Cílem takového náhradního modelu je co možná nejpřesněji aproximovat výstupy původní funkce.

2.4.1 Response surface modelování

Response surface modelování [3] je kolekce statistických a matematikých technik určených k efektivnímu vytváření náhradních modelů. Modelování získalo své jméno díky názvu, který používá pro funkční hodnotu modelované funkce - *response*. Vstupní hodnoty modelu se nazývají *nezávislé proměnné*.



Obrázek 2.3: Různé typy response, které je možné vytvořit pomocí polynomů druhého řádu [3].

Jelikož je pravá response funkce f neznámá, tak je potřeba ji aproximovat. Běžně využívané náhradní modely jsou polynomy malého stupně, které jsou vhodné pro odhady menších podprostorů nezávislých proměnných. Většinou jsou použity polynomy prvního či druhého řádu. Polynom prvního řádu lze obecně zapsat jako

$$f_M = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon, \tag{2.19}$$

kde $\boldsymbol{\beta}$ jsou parametry modelu a ϵ je implicitní statistická chyba. Polynom druhého řádu má předpis:

$$f_M = \beta_0 + \sum_{j=1}^{n} \beta_j x_j + \sum_{j=1}^{n} \beta_{jj} x_j^2 + \sum \sum_{i<j=2}^{n} \beta_{ij} x_i x_j + \epsilon. \tag{2.20}$$

Parametry modelu $\boldsymbol{\beta}$ přímo určují tvar povrchu response. Ukázka tvarů, které je možné aproximovat, je vidět na Obrázku 2.3. Cílem response surface modelování je nastavení těchto parametrů takovým způsobem, aby se funkční hodnoty náhradního modelu co možná nejvíce podobaly modelu reálnému. Běžně využívanou algoritmem pro nalezení parametrů je metoda nejmenších čtverců.

Metoda nejmenších čtverců se snaží pro $k > n$ pozorování minimalizovat chybu

$$L = \sum_{i=1}^{k} \epsilon_i^2 = \sum_{i=1}^{k} (y_i - \beta_0 - \sum_{j=1}^{n} \beta_j x_{ij})^2. \tag{2.21}$$

Pokud výpočet polynomiálního modelu převedeme do maticové násobení

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2.22}$$

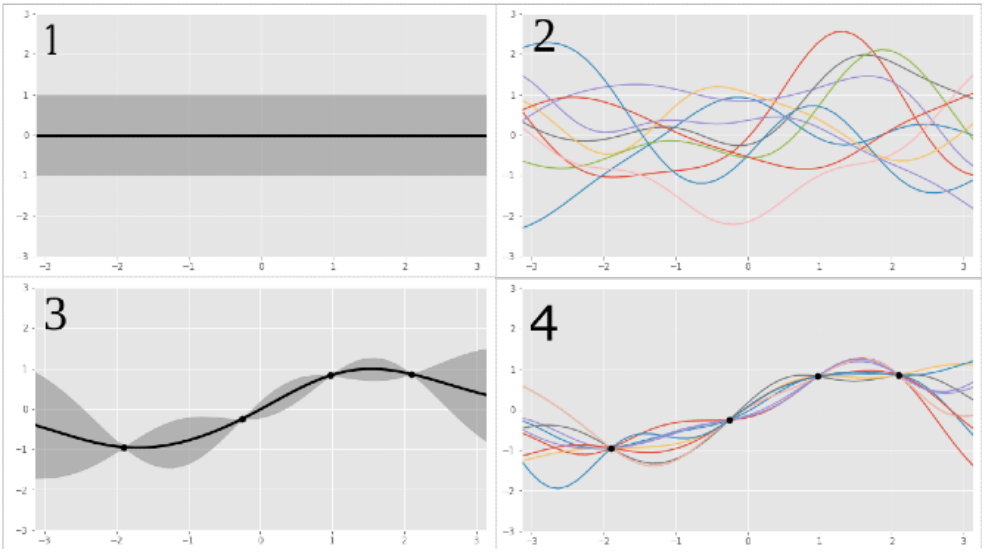
kde $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}$ je výstup modelu, $\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & ... & x_{1n} \\ 1 & x_{21} & x_{22} & ... & x_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{k1} & x_{k2} & ... & x_{kn} \end{bmatrix}$ je matice nezávislých proměnných (první sloupec je roven 1 pro správné započtení konstantního parametru β_0), $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$ je vektor koeficientů modelu a $\boldsymbol{\epsilon} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$ je vektor statistických chyb, tak můžeme parametry modelu $\boldsymbol{\beta}$ vypočítat následovně:

$$\boldsymbol{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}. \tag{2.23}$$

2.4.2 Gaussovské procesy

Gaussovský proces (GP) je nespočetný soubor náhodných veličin $(f_{GP}(\mathbf{x})), \mathbf{x} \in \mathcal{X}, \mathcal{X} \subset \mathbb{R}^n$ takových, že jejich libovolná konečná podmnožina má sdružené N-rozměrné Gaussovské (tj. normální) rozdělení. Každý takový proces je plně určen svou funkcí střední hodnoty $\mu : \mathcal{X} \to \mathbb{R}$ a kovarianční funkcí (jádrem) $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, kde $\mu(\mathbf{x}) = \mathbb{E}[f_{GP}(\mathbf{x})]$ a $K(\mathbf{x}_1, \mathbf{x}_2) = \mathrm{Cov}(f_{GP}(\mathbf{x}_1), f_{GP}(\mathbf{x}_2))$. Funkční hodnoty y jsou většinou dostupné pouze jako pozorování obsahující šum $y = f_{GP}(\mathbf{x}) + \epsilon$, kde ϵ je Gaussovská náhodná veličina se střední hodnotou 0 a variancí σ_n^2.

Jako náhradní model mají GP několik výhod. Hlavní výhodou je fakt, že přímo zachycují svou vlastní nejistotu. Jejich výstupem totiž není pouze funkční hodnota, ale celé rozdělení funkční hodnot, což umožňuje jejich efektivní využití v kritériích nejistoty, které potřebují pro svůj výpočet varianci. Dále se jedná o náhradní model, který nemá předem daný počet parametrů, ale upravuje ho v závislosti na počtu získaných dat (například oproti polynomiálnímu modelu, u kterého musíme specifikovat tvar polynomu). V neposlední řadě také umožňují začlenit znalost předpokládaného tvaru funkce pomocí chytré volby jádra. Hlavní nevýhodou gaussovských procesů je velká



Obrázek 2.4: Znázornění gaussovského procesu. V horní části je gaussovský proces, který zatím nebyl naučen pomocí žádných dat a 10 funkcí, které vznikly z tohoto procesu. Ve spodní části je proces, který byl naučen pomocí 4 různých bodů a 10 funkcí, které by mohl vygenerovat [4].

výpočetní složitost, která roste kubicky s počtem trénovacích bodů. Vizualizace gaussovského procesu je na obrázku 2.4.

2.5 CMA-ES a náhradní modely

Algoritmus CMA-ES je spolehlivý optimalizátor black-box funkcí. Jednou z jeho nevýhod je ale neustálé ohodnocování všech jedinců v generaci reálnou funkcí. To představuje problém v efektivním využití CMA-ESu v kontextu optimalizace drahých black-box funkcí. Z tohoto důvodu se s postupem času objevily modifikace algoritmu, které využívají náhradní modely.

Hlavní využití náhradních modelů v evoluční optimalizaci, tkví v ohodnocení jedinců tímto náhradním modelem namísto reálné black-box funkce. V ideálním případě by měl náhradní model vracet stejné funkční hodnoty jako originální funkce. Běžně využívané náhradní modely jsou tedy takové, které jsou schopny se učit a upravovat svoji predikci na základě tréninkových dat, například response surface modely nebo gaussovské procesy.

Širšímu využití náhradních modelů také napomáhá fakt, že CMA-ES pracuje pouze s pořadím jedinců a nikoliv s absolutní hodnotou fitness. Nezáleží tedy na rozmezí funkčních hodnot náhradního modelu, ale pouze na tom, jestli je pořadí bodů ohodnocených náhradním modelem a seřazených v závislosti na jejich fitness totožné jako kdyby byly ohodnoceny funkcí reálnou. [16]

Varianty CMA-ESu využívající náhradní modely algoritmus CMA-ES obohacují také o tzv. *řízení evoluce*. Proces řízení evoluce je zodpovědný za výběr jedinců k ohodnocení náhradním modelem. Kompletně tedy nahrazuje proces selekce jedinců vybraných k následnému křížení a mutaci.

Metody řízení evoluce můžeme rozdělit do dvou základních kategorií (podle Jinovy teminologie [17]) v závislosti na způsobu výběru jedince k ohodnocení náhradním modelem: *individuální* a *generační*. Individuální řízení evoluce vybírá z množiny jedinců ohodnocených náhradním modelem λ jedinců, které ohodnotí reálnou fitness funkcí. V roce 2002 bylo použito individuální řízení evoluce v článku představující Metamodel-Assisted Evolution Strategy (MA-ES) [18]. Naopak generační řízení evoluce se vždy rozhoduje na úrovní celé generace, takže ohodnotí reálnou fitness funkcí buď všechny jedince v generaci, nebo žádného [19].

Tato práce se zabývá celkem třemi variantami CMA-ESu, které využívají náhradní modely: Lokální MetaModel CMA-ES, Dvojitě trénovaný CMA-ES a Lineárně kvadratický CMA-ES. Řízení evoluce všech zde zmíněných variant nelze jednoznačně kategorizovat, protože se jedná o jakýsi přechod mezi individuální a generační strategií. V každé generaci je v závislosti na momentálním stavu ohodnoceno 1 až λ jedinců.

2.5.1 Lokální MetaModel CMA-ES

Lokální MetaModel (lmm-CMA-ES - *local-meta-model CMA-ES*) [20, 21] je verze CMA-ESu, která využívá lokální metamodel vytvořený specificky pro každého jedince.

Algoritmus si udržuje databázi všech dosud ohodnocených bodů reálnou funkcí. Dokud není ohodnoceno alespoň $n(n+3)/2+1+1$ (počet stupňů volnosti jednoho meta-modelu+1) bodů, tak je využíván čistý CMA-ES bez úprav. Jakmile je dosažen potřebný počet bodů v databázi, tak každá následující generace g využívá algoritmus řízení evoluce pro efektivní zapojení náhradního modelu k tomu, aby určila pořadí jedinců,

Hlavní myšlenkou tohoto řízení evoluce je vytvoření plně kvadratického modelu pro každého jedince $\mathbf{x}_k$, který má $n(n+3)/2+1$ stupňů volnosti. Tento kvadratický model je využit pro získání hodnoty fitness a následné seřazení jedinců v generaci. Kvalita předpovězených pořadí jedinců je zajištěna ohodnocením určité části z nich ($n_{init} \geq 0$ a $n_b > 0$) na originální funkci. Následně jsou všechny náhradní modely vytvořeny znovu a jejich funkční hodnoty jsou využity pouze v případě, že změna v pořadí není moc velká. Pseudokód tohoto řízení evoluce je znázorněný v Algoritmu 12.

2.5.1.1 Konstrukce modelu

Model využívaný algoritmem lmm-CMA-ES patří do kategorie response surface modelů. Pro každý bod, který chceme ohodnotit, model nejdříve vybere

Input: $\mathbf{x}_k$ - $k \in \mathbb{N}$ jedinců k ohodnocení, n_{init} - minimální počet jedinců k ohodnocení, err_{max} - maximální povolená chyba modelu, n_b počet dalších jedinců k ohodnocení

1 Vytvoř náhradní model pro každého jedince podle dat k_{nn} nejbližších sousedů;
2 Vytvoř $ranking_0$ pořadí jedinců podle funkčních hodnot z náhradních modelů;
3 Ohodnoť n_{init} nejlepších jedinců reálnou fitness funkcí a ulož výsledek do databáze;
4 Vytvoř náhradní model pro každého jedince podle dat k_{nn} nejbližších sousedů;
5 Vytvoř $ranking_0$ pořadí jedinců podle funkčních hodnot z náhradních modelů;
6 $err = \infty$; i = 0;
7 **while** *nejsou ohodnoceni všichni jedinci* ***and*** $err > err_{max}$ **do**
8 i = i + 1;
9 Ohodnoť n_b nejlepších dosud neohodnocených jedinců reálnou fitness funkcí a ulož výsledek do databáze;
10 Vytvoř náhradní model pro každého jedince podle dat k_{nn} nejbližších sousedů;
11 Vytvoř $ranking_i$ pořadí jedinců podle funkčních hodnot náhradních modelů;
12 $err = \sum_{1<j<\lambda} |ranking_{i-1}(j) - ranking_i(j)|$
end

Result: Pořadí jedinců $ranking_i$

Algoritmus 3: Pseudokód řízení evoluce lokálního metamodelu

k_{nn} nejbližších již ohodnocených bodů. Pro určení vzdálenosti dvou bodů se využívá vzorec vycházející z Mahalanobisovy vzdálenosti, ale doplněný o informace obsažené v kovariační matici a velikosti kroku CMA-ESu. Vypočítá se následovně:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{C}^{-1} \delta^2 (\mathbf{x} - \mathbf{y})} = ||\mathbf{M}(\mathbf{x} - \mathbf{y})||, \tag{2.24}$$

kde

- $\mathbf{C}$ je kovarianční matice,
- δ je velikost kroku,
- $||.||$ je Euklidovská norma,
- $\mathbf{M} = \frac{1}{\delta}\mathbf{D}^{-1/2}\mathbf{B}^T$,
- Ortogonální matice $\mathbf{B}$ a diagonální matice $\mathbf{D}$ vycházejí z rozložení $\mathbf{C} = \mathbf{B}\mathbf{D}\mathbf{B}^T$.

Pokud bod, pro který chceme vytvořit kvadratický metamodel, označíme jako q, tak je předpis metamodelu následující:

$$\hat{f}_\beta = (\mathbf{x} - \mathbf{q})^T \mathbf{M}^T \mathbf{A} \mathbf{M} (\mathbf{x} - \mathbf{q}) + \mathbf{a}^T \mathbf{M} (\mathbf{x} - \mathbf{q}) + a_0 = \mathbf{z}^T \mathbf{A} \mathbf{z} + \mathbf{a}^T \mathbf{z} + a_0 \quad (2.25)$$

kde

- $\mathbf{z} = \mathbf{M}(\mathbf{x} - \mathbf{q})$

Cílem je nalézt parametry $\boldsymbol{\beta} = [\mathbf{A}, \mathbf{a}, a_0]$ ($\mathbf{A}$ je symetrická matice $\in \mathbb{R}^{n \times n}$, $\mathbf{a} \in \mathbb{R}^n$ a $a_0 \in \mathbb{R}^+$).

Pro samotné vytvoření metamodelu vybereme z databáze ohodnocených bodů k_{nn} bodů $(\mathbf{s}_i, y_i = f(\mathbf{s}_i))_{1 \leq i \leq k_{nn})}$, které jsou nejblíže podle vzorce 2.24 a jsou seřazené. Poté můžeme vypočítat koeficienty $\boldsymbol{\beta}$ tak, že se snažíme minimalizovat chybu:

$$LS(\boldsymbol{\beta}) = \sum_{i=1}^{k_{nn}} K \left(\frac{d(\mathbf{s}_i, \mathbf{q})}{d(\mathbf{s}_{k_{nn}}, \mathbf{q})} \right) (\hat{f}_\beta(\mathbf{s}_i) - \mathbf{y}_i)^2, \quad (2.26)$$

kde

- K je kernel funkce $K(\xi) = (1 - \xi^2)^2$.

Stejně jako v publikaci popisující tento model [21] je i v této práci využita soustava lineárních rovnic pro nalezení koeficientů $\boldsymbol{\beta}$. To je možné diky faktu, že zmíněný kvadratický model je lineární v koeficientech $\boldsymbol{\beta}$, přesněji $\hat{f}_\beta(\mathbf{s}_i) = \tilde{\mathbf{z}}_i \tilde{\boldsymbol{\beta}}$, kde $\mathbf{z}_i = \mathbf{M}(\mathbf{s}_i - \mathbf{q})$ a pro vektor $\mathbf{z} = (z_1, ..., z_d)^T$ je vektor $\tilde{\mathbf{z}} \in R^{D(D+3)/2+1}$ roven

$$\tilde{\mathbf{z}} = (z_1^2, ..., z_D^2, z_1 z_2, ..., z_{D-1} z_D, z_1, ..., z_D, 1) \quad (2.27)$$

a

$$\tilde{\boldsymbol{\beta}} = (A_{11}, ..., A_{DD}, 2A_{12}, ..., 2A_{D-1,D}, a_1, ..., a_D, a_0)^T. \quad (2.28)$$

Zadefinování matice $\mathbf{W}$ následujícím způsobem:

$$\mathbf{W} = \operatorname{diag} \sqrt{K(d(\mathbf{s}_i, \mathbf{q}) / d(\mathbf{s}_{k_{nn}}, \mathbf{q}))} \quad (2.29)$$

umožňuje vyjádřit váženou chybu jako:

$$LS(\tilde{\boldsymbol{\beta}}) = ||\mathbf{W}\mathbf{Z}\tilde{\boldsymbol{\beta}} - \mathbf{W}\mathbf{Y}||^2, \quad (2.30)$$

kde $\mathbf{Z} = \begin{pmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ ... \\ \tilde{z}_{k_{nn}} \end{pmatrix}$, $\mathbf{Y} = (y_1, ..., y_{k_{nn}})^T$ a vyřešit soustavu rovnic již existující funkcí v Matlabu:

$$\tilde{\boldsymbol{\beta}} = \mathbf{W}\mathbf{Z}\tilde{\boldsymbol{\beta}} \backslash \mathbf{W}\mathbf{Y} \quad (2.31)$$

Odhad reálné funkční hodnoty v bodě $\mathbf{q}$ odpovídá poslednímu (konstantnímu) koeficientu $\tilde{\boldsymbol{\beta}}$ [21, 20].

Input: g, g_m (maximální počet po sobě jdoucích generací ohodnocených náhradním modelem), $\mathbf{x}_k$, f

```
if g má být ohodnocena reálnou funkcí then
    y_k = f(x_k);
    Ulož ohodnocené dvojice (x_k, y_k);
    if Je uložený dostatečný počet dat then
        Natrénuj model f_M pomocí dat;
        Generaci g+1 ohodnoť pomocí f_M;
    end
else
    Ohodnoť generaci modelem y_k = f_M(x_k);
    if Ohodnoceno již g_m generací po sobě modelem then
        Generaci g+1 ohodnoť reálnou funkcí
    else
        Generaci g+1 ohodnoť náhradním modelem
    end
end
```

Algoritmus 4: Evoluční kontrola S-CMAES

2.5.2 Dvojitě trénovaný CMA-ES

Dvojitě trénovaný CMA-ES (DTS-CMA-ES - *Doubly Trained S-CMA-ES*)[22] je rozšířením algoritmu Surrogate-CMA-ES (S-CMA-ES) [23].

2.5.2.1 Surrogate CMA-ES

S-CMA-ES využívá evoluční strategii, kterou můžeme zařadit mezi generační. Všechny body jsou ohodnoceny reálnou black-box funkcí až do chvíle, kdy je ohodnoceno dostatečné množství bodů, které je potřebné k natrénování náhradního modelu. Následné generace jsou ohodnoceny střídavě buď náhradním modelem nebo reálnou funkcí v závislosti na typu ohodnocení generací předchozích. Pseudokód algoritmu je znázorněn v Algoritmu 4.

Pro zvýšení přesnosti náhradního modelu jsou v jeho trénovací fázi využity pouze body, jejichž Mahalanobisova vzdálenost od stávající střední hodnoty rozdělení CMA-ESu $\mathbf{m}$ je menší nebo rovna předem stanovené hodnotě r. Pokud je velikost trénovací množiny dostatečná, tak je vybráno n_{max} bodů pomocí k-NN metody. Tyto body jsou následně transformovány do báze definované vlastními vektory kovarianční matice $\mathbf{C}$ skrze násobení $(((\sigma^2\mathbf{C})^{-1/2})^\top)$. Body určené k predikci jsou transformovány stejným způsobem pro zachování konzistence.

2.5.2.2 DTS-CMA-ES

DTS-CMA-ES obohacuje výše popsaný S-CMA-ES o několik vylepšení. Hlavní změnou je využití variance náhradního modelu spolu s jeho funkční hodnotou. Algoritmus nejdříve ohodnotí celou generaci náhradním modelem, který byl natrénován za použití dosud nashromážděných dat. Ohodnocené body následně seřadí podle vybraného *kritéria nejistoty* C a ohodnotí n_{orig} bodů s největší nejistotou na originální black-box funkci. Následuje další vytvoření náhradního modelu, který má v průběhu trénování již k dispozici i nově ohodnocené body. Posledním krokem je ohodnocení zbývajících $\lambda - n_{orig}$ jedinců novým modelem. Odpovídající pseudokód algoritmu je znázorněn v Algoritmu 5.

Input: λ, databáze ohodnocených bodů $\mathcal{A}$, populace jedinců $\{x_k\}$, kritérium nejistoty C, počet bodů k ohodnocení n_{orig}
1 Trénuj model f_M
2 Ohodnoť body $(\hat{y}_k, s_k^2) \leftarrow f_M(\mathbf{x}_k)$
3 Vypočítej kritérium nejistoty $c_k \leftarrow C(\hat{y}_k, s_k^2)$
4 Seřaď podle kritéria
5 Ohodnoť n_{orig} bodů $y_k \leftarrow f(\mathbf{x}_k), k \in \{1, ..., n_{orig}\}$
6 Přidej nově ohodnocené body do databáze $\mathcal{A}$
7 Trénuj model f_M
8 Ohodnoť zbývající body novým modelem $y_k \leftarrow f_M(\mathbf{x}_k), k \in \{n_{orig} + 1, ..., \lambda\}$
Output: $y_k, k \in \{1, ..., \lambda\}$

Algoritmus 5: Evoluční kontrola algoritmu DTS-CMA-ES

Existuje několik různých kritérií nejistoty. Patří mezi ně například:

- **Variance.** Variance s^2 bodů, které byly ohodnoceny náhradním modelem. Čím větší variance, tím větší je nejistota předpovězené fitness.

$$C_{s^2} = s^2 \tag{2.32}$$

- **Spodní interval spolehlivosti (LCB - Lower Confidence Bound).** Body s menší hodnotou spodního intervalu spolehlivosti jsou považovány za zajímavější pro přehodnocení.

$$C_{LCB} = \hat{y} - 2s \tag{2.33}$$

- **Pravděpodobnost vylepšení (PoI - Probability of Improvement).** Pravděpodobnost vylepšení pro určitou cílovou hodnotu $T \leq y_{min}$ můžeme vyjádřit následovně:

$$C_{PoI} = P(f(x) \leq T | y_1, ..., y_n) = \phi\left(\frac{T - \hat{y}}{s}\right) \tag{2.34}$$

kde ϕ značí distribuční funkci $\mathcal{N}(0,1)$ a y_{min} je nejlepší dosud nalezená hodnota.

- **Předpokládané zlepšení (EI - Expected Improvement).** Předpokládané zlepšení je popsáno jako:

$$C_{EI} = E((y_{min} - f(\mathbf{x}))I(f(\mathbf{x}) < y_{min})|y_1, ..., y_n), \tag{2.35}$$

kde

$$I(f(\mathbf{x}) < y_{min}) = \begin{cases} 1 \leftarrow f(\mathbf{x}) < y_{min} \\ 0 \leftarrow f(\mathbf{x}) \geq y_{min}. \end{cases} \tag{2.36}$$

Podobně můžeme popsat C_{EI} jako

$$C_{EI} = (y_{min} - \hat{y})\phi\left(\frac{y_{min} - \hat{y}}{s}\right) + s\varphi\left(\frac{y_{min} - \hat{y}}{s}\right), \tag{2.37}$$

kde φ značí hustotu rozdělení $\mathcal{N}(\mathbf{0}, \mathbf{1})$.

2.5.3 Lineárně kvadratický CMA-ES

Lineárně kvadratický CMA-ES (lq-CMA-ES - *Linear-Quadratic CMA-ES*)[24] je verze CMA-ESu využívající jednoduchý polynomiální model v lineární či kvadratické formě v závislosti na počtu již ohodnocených dat.

Podobně jako ostatní verze CMA-ESu si i ten lineárně kvadratický udržuje databázi již ohodnocených bodů, pouze v tomto případě se jedná o frontu s omezenou kapacitou, kde jsou staré body postupně nahrazovány těmi nově ohodnocenými. V každé iteraci algoritmus nejdříve vytvoří náhradní model, ohodnotí všechny jedince z aktuální generace tímto modelem a následně vyhodnotí reálnou fitness několika nejlepších jedinců. Dokud je Kendallův korelační koeficient mezi pořadím předpovězeným náhradním modelem a pořadím reálně ohodnocených jedinců této generace menší než předem daná konstanta (autory nastavená na 0.85 [24]), je počet jedinců vybraných pro ohodnocení reálnou funkcí opakovaně zvyšován a model znovu trénován i s nově ohodnocenými jedinci.

Celkové pořadí populace je určeno výhradně reálnou fitness funkcí, nebo náhradním modelem podle toho, jestli byli ohodnoceni všichni jedinci v generaci reálnou funkcí. Nemůže tedy nastat situace, že by byly zkombinovány hodnoty náhradního modelu a reálné fitness. Pseudokód algoritmu je znázorněn v Algoritmu 6.

2.5.3.1 Lineárně kvadratický model

Lineárně kvadratický model je response surface model, který ale mění svůj stupeň a tvar v závislosti na počtu získaných dat.

Input: Populace jedinců x_k, fronta $\mathcal{M}$, model f_M

$k \leftarrow \lfloor 1 + \max(\lambda * 0.02, 4 - |\mathcal{M}|) \rfloor$

while $|X| > 0$ **do**

  Přesuň $k - (\lambda - |X|)$ nejoptimálnějších bodů podle f_M z X do $\mathcal{M}$

  Seřaď $\min(k, \lambda)$ naposledy přidaných bodů v $\mathcal{M}$ podle skutečné funkční hodnoty

  $\mathbf{y}_1, ..., \mathbf{y}_j \leftarrow$ posledních $\max(15, \min(1.2k, 0.75\lambda))$ z $\mathcal{M}$

  if $KendallCoeff(f_M(\boldsymbol{y}_1, ..., \boldsymbol{y}_j), f(\boldsymbol{y}_1, ..., \boldsymbol{y}_j)) \geq 0.85$ **then**

    break while

  end

  $k \leftarrow \lfloor 1.5k \rfloor$

end

if $|X| \geq 0$ **then**

  return $f_M(x_1), ..., f_M(x_\lambda)$ posunuté o $\min_{\mathbf{x} \in \text{posledních } k \text{ bodů z } \mathcal{M}}(f(\mathbf{x})) - \min_{i=1,...,\lambda}(f_M(\mathbf{x}_i))$

end

else

  return $f(\mathbf{x}_1), ..., f(\mathbf{x}_\lambda)$

end

Algoritmus 6: Evoluční kontrola algoritmu lq-CMA-ES

Model využívá obecné zobrazení $z : \mathbb{R}^n \to \mathbb{R}^m, m \in \mathbb{R}, \mathbf{x} \to z(\mathbf{x})$ a cílem modelu je pro data $\mathbf{x}_i \in \mathbb{R}^n$ odhadnout reálnou funkční hodnotu $f(\mathbf{x}_i)$ pomocí skalárního násobení

$$f' = \mathbf{m}^T z(\mathbf{x}_i), \tag{2.38}$$

kde $\mathbf{m} \in \mathbb{R}^m$ jsou koeficienty modelu.

K nalezení zmíněných koeficientů $\mathbf{m}$ je potřeba minimalizovat chybu:

$$\sum_i w_i^2 (\mathbf{m}^T z(\mathbf{x}_i) - f(\mathbf{x}_i))^2, \tag{2.39}$$

kde w_i je kladná váha bodu. Zadefinováním $\mathbf{Z} := [z(x_1), ..., z(x_N)]^T, \mathbf{f} := [f(x_1), ..., f(x_N)]^T$ a $\mathbf{w} = [w_1, ..., w_m]^T$ můžeme chybový předpis přepsat jako:

$$\arg\min_{\mathbf{m}} ||\mathbf{w} \circ (\mathbf{Z}\mathbf{m} - \mathbf{f})||^2, \tag{2.40}$$

kde $\circ$ je Hadamardův součin. Samotný výpočet řešení využívá Moore–Penroseovu pseudoinverzi:

$$\mathbf{m} = \mathbf{Z}(\text{diag}(\mathbf{w}))^+. \tag{2.41}$$

Proces zadefinovaný tímto způsobem je zcela nezávislý na výběru zobrazení z. Tato práce využívá stejná zobrazení jako [24], která se mění v závislosti na počtu uložených bodů. Jedná se o:

$$z_{lin} : x \to [1, x_1, x_2, ..., x_n]^T \tag{2.42}$$

$$z_{quad} : x \rightarrow [z_{lin}(x)^T, x_1^2, ..., x_n^2]^T \tag{2.43}$$

$$z_{full} : x \rightarrow [z_{quad}(x)^T, x_1x_2, x_1x_3, ..., x_1x_n, x_2x_3, ..., x_2x_n, x_3x_4, ..., x_{n-1}x_n]^T \tag{2.44}$$

Model začíná s lineárním zobrazením z a přepne na složitější, když je počet dat ohodnocených reálnou fitness o 10% větší než počet stupňů volnosti složitějšího modelu.

Kapitola 3

CMA-ES s response surface modely

Tato kapitola se zabývá převážně praktickou částí práce. Nejdříve je zde představena použitá metodologie a implementace potřebná k dosažení cílů práce. Následuje popis testovacího frameworku BBOB spolu s novými funkcemi, které vychází z reálných problémů a byly v rámci této práce implementovány. Veškerá konfigurace použitá při testování je popsána ve čtvrté sekci. Nakonec jsou prezentovány výsledky testování spolu s vyvozenými závěry.

3.1 Metodologie

Pro realizaci cílů práce jsem si vybral následující postup. Kromě implementace response surface modelů vycházejících z vybraných variant CMA-ESu a jejich následné integrace do frameworku, jsem se rozhodl také pro implementaci evolučních kontrol ze stejných algoritmů. Každá z tří vybraných variant algoritmu CMA-ES je rozdělena na dvě komponenty: evoluční kontrolu a náhradní model. Jejich kombinací vznikne celkem šest dvojic:

- **DTS + lmm** → Evoluční kontrola z DTS-CMA-ES spolu s lokálním metamodelem.
- **DTS + linQuad** → Evoluční kontrola z DTS-CMA-ES spolu s lineárně kvadratickým modelem.
- **LinQuad + gp** → Evoluční kontrola z lq-CMA-ES spolu s Gaussovskými procesy.
- **LinQuad + lmm** → Evoluční kontrola z lq-CMA-ES spolu s lokálním metamodelem.
- **Lmm + gp** → Evoluční kontrola z LMM-CMA-ES spolu s Gaussovskými procesy.

- **Lmm + linQuad** → Evoluční kontrola z LMM-CMA-ES spolu s lineárně kvadratickým modelem.

Všechny dvojice jsou následně otestovány v různých dimenzích pomocí testovacího frameworku BBOB. Toto testování poskytne potřebná data ke komparaci, jak jsou jednotlivé kombinace úspěšné pro funkce s různou obtížností a tvarem povrchu funkčních hodnot.

3.2 Implementace

Hlavní část implementace spočívala v naprogramování algoritmů lmm-CMA-ES a lq-CMA-ES, které bylo specifikováno dvěma požadavky:

1. Algoritmus musí být plně integrován do již existujícího frameworku.
2. Musí být možné využít libovolnou evoluční kontrolu s jakýmkoli náhradním modelem.

Při implementaci jednotlivých modelů bylo také zapotřebí zajistit, aby byly modely schopné nalézt bod, pro který bude jejich vlastní funkční hodnota minimální, protože ho využívá evoluční kontrola lq-CMA-ES. Model získaný z lq-CMA-ESu kopíruje originální implementaci a minimum získává s využitím Hessovy matice. Pro gaussovské procesy nebo model z lmm-CMA-ESu neexistuje jednoduchý způsob, jak takové minimum vypočítat. Jelikož se jedná o optimalizační úlohu hledání minima, tak je i k jeho nalezení využit algoritmus CMA-ES.

Z důvodu specifické práce algoritmu lq-CMA-ES s databází již ohodnocených bodů bylo také zapotřebí upravit archiv(databázi), který je společný pro všechny varianty CMA-ESu.

V neposlední řadě byly implementovány a integrovány do testovacího frameworku nové funkce založené na reálných problémech, které jsou detailně popsány níže.

Jedna z nově implementovaných funkcí využívá mapu, která byla vytvořena redukováním originální mapy zveřejněné organizací NASA [25] na požadované rozlišení. Originální mapa je poskytována jako 22 912 archivů, každý obsahující celkem 3601x3601 bodů a celková velikost datasetu je 387GB, což je pro potřeby této práce nepraktické. Z tohoto důvodu jsem vytvořil program, který slouží k redukci rozlišení, viz přiložené CD.

3.2.1 Validace implementovaných algoritmů

Součástí této práce je i samotná implementace algoritmů lmm-CMA-ES a lq-CMA-ES. Pro ověření správnosti implementace byly obě varianty otestovány na frameworku BBOB s nastavením, které se co možná nejvíce podobalo konfiguraci využitou autory originální práce (identické funkce a instance). Detailní

porovnání vlastní implementace vůči té originální je dostupné online, resp online, nebo na přiloženém CD.

Vlastní implementace lmm-CMA-ESu má v podstatě totožné výsledky s tou originální. Jediné rozdíly jsou v tom, že vlastní implementace inicializuje hodnotu velikosti kroku $\sigma = 8/3$ a původní na $\sigma = 2$. Dalším rozdílem je nastavení prvotních souřadnic jedince, kdy lmm-CMA-ES využívá nulový vektor, což hraje podstatnou roli např. ve funkci f_{19}, která má minimum v nule.

Větší rozdíl je mezi implementacemi lq-CMA-ESu. První rozdíl je opět ve využití nulového vektoru pro inicializaci prvního jedince. Důsledek nulové inicializace je znázorněn na obrázku 3.14. Dalším rozdílem je využití starší verze CMA-ESu, který je neustále zdokonalován. Tato práce využívá schodně s [22] verzi z roku 2012, zatímco originální lq-CMA-ES tu nejnovější. Ačkoliv jsou výstupy náhradního modelu a jedinci vybraní k ohodnocení evoluční kontrolou totožní, tak je celkový průběh algoritmu jiný z důvodu využitých vylepšení v úpravě kovarianční matice a velikosti kroku σ. Tento fakt ale nebrání ve splnění stanovených cílů této práce, protože všechny kombinace řízení evoluce a náhradního modelu mají totožné podmínky.

3.3 Testovací framework

Pro běh testů byl využit testovací framework BBOB [26], který poskytuje celkem 54 různých black-box funkcí - 24 bez šumu a 30 se šumem. Každá funkce je jednoznačně definována svým indexem (jménem) spolu s číslem instance. Číslo instance slouží k inicializaci náhodného generátoru, který využívají funkce pro posun funkční hodnoty globálního optima, jeho souřadnic a různé rotace prostoru funkčních hodnot podle jednotlivých os. Využití čísla instance dává frameworku 2 důležité vlastnosti:

1. Možnost vygenerovat různé black-box funkce, jejich obtížnost je podobná. Funkce, která má tvar koule, bude mít podobný tvar pro všechny její instance, ale bude se lišit jejich optimum funkční hodnoty nebo rotace zmíněné koule.

2. Funkce se stejným indexem a číslem instance jsou vždy totožné, což umožňuje replikovat experimenty.

Níže popsané předpisy funkcí využívají následující definice:

- $T_{osz} : \mathbb{R}^n \to \mathbb{R}^n$ je rovno

$$y = \text{sign}(\mathbf{x})\exp(\hat{\mathbf{x}} + 0.049(\sin(c_1\hat{\mathbf{x}}) + \sin(c_2\hat{\mathbf{x}}))), \tag{3.1}$$

  kde

  - $\hat{x} = \begin{cases} \log(|x|), & \text{když } x \neq 0, \\ 0 \text{ jinak}, \end{cases}$

- $\text{sign}(x) = \begin{cases} -1, \text{ když } x < 0 \\ 0, \text{ když } x = 0 \\ -1 \text{ jinak,} \end{cases}$
- $c_1 = \begin{cases} 10, \text{ když } x > 0 \\ 5.5 \text{ jinak,} \end{cases}$
- $c_2 = \begin{cases} 7.9, \text{ když } x > 0 \\ 3.1 \text{ jinak.} \end{cases}$

- $T_{\text{asy}}^{\beta} : \mathbb{R}^n \to \mathbb{R}^n$ je rovno

$$y = \begin{cases} x_i^{1+\beta \frac{i-1}{n-1}\sqrt{x_i}}, \text{ když } x_i > 0 \\ x_i \text{ jinak.} \end{cases} \tag{3.2}$$

- A^{α} je diagonální matice dimenze n, jejichž í-tý diagonální prvek je roven $\lambda_{ii} = \alpha^{\frac{1}{2}\frac{i-1}{n-1}}$,
- f_{opt} je náhodný posun prostoru funkčních hodnot v ose y,
- $\mathbf{x}^{\text{opt}}$ jsou souřadnice optima,
- $f_{\text{pen}} : \mathbb{R}^n \to \mathbb{R}$ je rovno $y = \sum_{i=1}^{n} \max(0, |x_i| - 5)^2$.

3.3.1 Funkce bez šumu

Funkce bez šumu jsou dále rozděleny do následujících kategorií:

- oddělitelné (1-5),
- málo nebo středně podmíněné (6-9),
- vysoce podmíněné a unimodální (10-14),
- multimodální s přiměřenou globální strukturou (15-19),
- multimodální se slabou globální strukturou (20-24).

3.3.1.1 Funkce oddělitelné

Oddělitelné funkce jsou následující:

1. **Koule** - Koule patří mezi nejjednodušší zkoumané spojité funkce. Jedná se o funkci unimodální, vysoce symetrickou a invariantní vůči rotacím a škálování. Její předpis je:

$$f_1(\mathbf{x}) = ||\mathbf{z}||^2 + f_{\text{opt}}, \tag{3.3}$$

kde $\mathbf{z} = \mathbf{x} - \mathbf{x}^{\text{opt}}$.

2. **Elipsoid** - Globálně kvadratická funkce s hladkými lokálními nerovnostmi.

$$f_2(\mathbf{x}) = \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} z_i^2 + f_{\text{opt}}, \tag{3.4}$$

kde $\mathbf{z} = T_{osz}(\mathbf{x} - \mathbf{x}^{\text{opt}})$.

3. **Rastrigin** - Vysoce multimodální funkce s poměrně pravidelnou strukturou umístění optima.

$$f_3(\mathbf{x}) = 10\left(n - \sum_{i=1}^{n} \cos(2\pi z_i)\right) + ||\mathbf{z}||^2 + f_{\text{opt}}, \tag{3.5}$$

kde $\mathbf{z} = A^{10} T_{\text{asy}}^{0.2}(T_{\text{osz}}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$

4. **Büche-Rastrigin** - Vysoce multimodální funkce se strukturovaným ale vysoce asymetrickým umístěním optima. Hlavní využití najde jako funkce určená ke klamání algoritmů, které prohledávají prostor symetricky.

$$f_4(\mathbf{x}) = 10\left(n - \sum_{i=1}^{n} \cos(2\pi z_i)\right) + \sum_{i=1}^{n} z_i^2 + 100 f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}, \tag{3.6}$$

kde

- $z_i = s_i T_{\text{osz}}(x_i - x_i^{\text{opt}})$ pro $i = 1, ..., n$,
- $s_1 = \begin{cases} 10 \times 10^{\frac{1}{2}\frac{i-1}{n-1}}, \text{ když } z_i > 0 \text{ a } i = 1, 3, 5, ..., n, \\ 10^{\frac{1}{2}\frac{i-1}{n-1}} \text{ jinak.} \end{cases}$

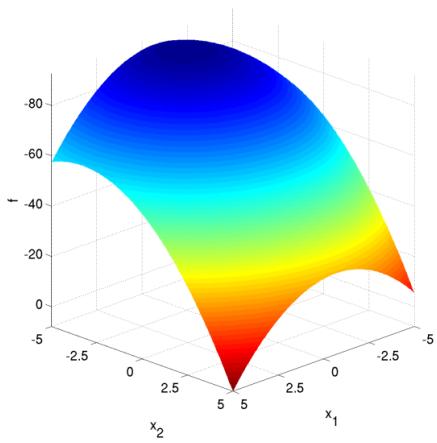
5. **Lineární sklon** - Funkce testující, zda vyhledávání může nalézt optimum umístěné na samém okraji definičního oboru.

$$f_5(\mathbf{x}) = \sum_{i=1}^{n} 5|s_i| - s_i z_i + f_{\text{opt}}, \tag{3.7}$$

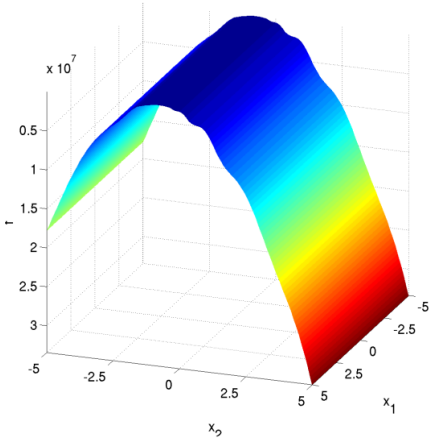
kde

- $z_i = x_i$ pokud $x_i^{\text{opt}} x_i < 25$, jinak $z_i = x_i^{\text{opt}}$,
- $s_i = \text{sign}(x_i^{\text{opt}}) 10^{\frac{i-1}{n-1}}$.

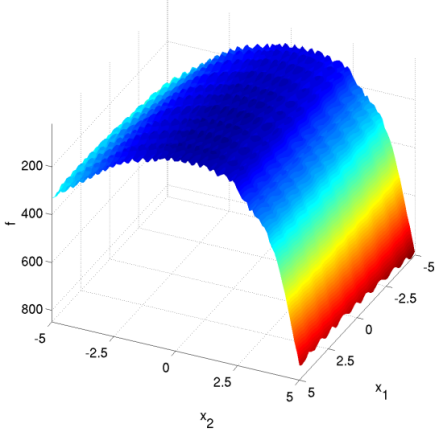
Ukázku prostorů funkčních hodnot pro oddělitelné funkce můžeme vidět v Obrázku 3.1.



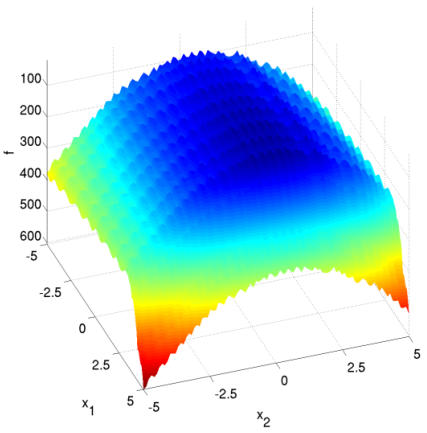
f_1 - Koule



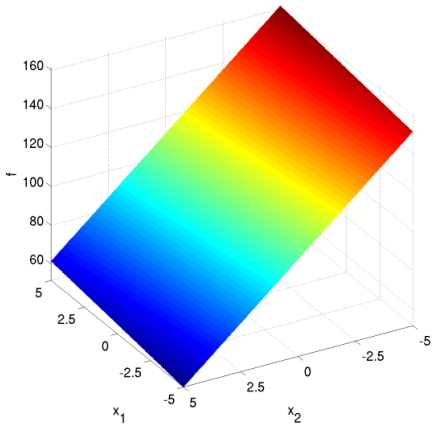
f_2 - Elipsoid



f_3 - Rastrigin



f_4 - Büche-Rastrigin



f_5 - Lineární sklon

Obrázek 3.1: Ukázka prostoru funkčních hodnot pro jednotlivé oddělitelné funkce ve 2D.

3.3.1.2 Funkce málo nebo středně podmíněné

Málo nebo středně podmíněné funkce jsou následující:

6. **Atraktivní sektor** - Vysoce asymetrická funkce, kde pouze jeden „hyperkužel“ vrací nízké funkční hodnoty.

$$f_6(\mathbf{x}) = T_{\text{osz}}\left(\sum_{i=1}^{n}(s_i z_i)^2\right)^{0.9} + f_{\text{opt}}, \tag{3.8}$$

   kde

   - $\mathbf{z} = \mathbf{Q}A^{10}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$,
   - $s_i = \begin{cases} 100, \text{ když } z_i \times x_i^{\text{opt}} > 0, \\ 1 \text{ jinak.} \end{cases}$

7. **Step Elipsoid** - Funkce obsahující mnoho plošin různých velikostí. S výjimkou malé plochy blízko globálního optima se gradient blíží nule skoro všude.

$$f_7(\mathbf{x}) = 0.1 \max\left(|\hat{z}_1|/10^4, \sum_{i=1}^{n} 10^{2\frac{i-1}{n-1}} z_i^2\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}, \tag{3.9}$$

   kde

   - $\hat{\mathbf{z}} = A^{10}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$,
   - $\tilde{z}_i = \begin{cases} \lfloor 0.5 + \hat{z}_i \rfloor, \text{ když } |\hat{z}_i| > 0.5, \\ \lfloor 0.5 + 10\hat{z}_i \rfloor / 10 \text{ jinak,} \end{cases}$
   - $\mathbf{z} = \mathbf{Q}\tilde{\mathbf{z}}$.

8. **Rosenbrock** - Funkce obsahující několik údolí.

$$f_8(\mathbf{x}) = \sum_{i=1}^{n-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{\text{opt}}, \tag{3.10}$$

   kde

   - $\mathbf{z} = \max\left(1, \frac{\sqrt{n}}{8}\right)(\mathbf{x} - \mathbf{x}^{\text{opt}}) + 1$.

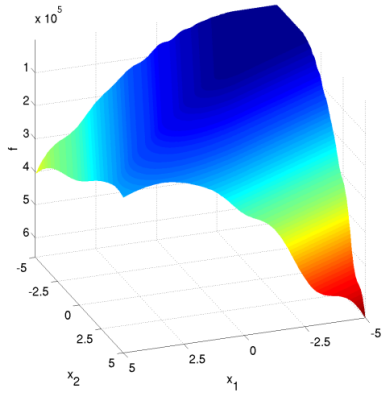
9. **Rosenbrock rotovaný** - Zrotovaná verze originální Rosenbrockovy funkce.

$$f_9(\mathbf{x}) = \sum_{i=1}^{n-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{\text{opt}}, \tag{3.11}$$

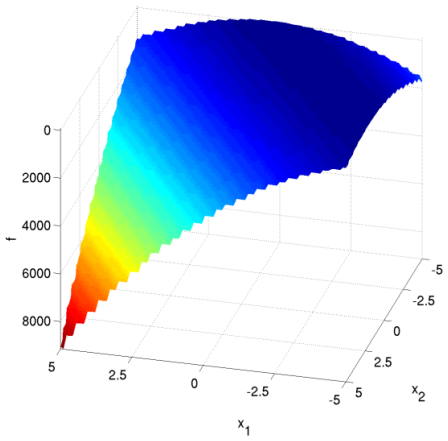
   kde

   - $\mathbf{z} = \max\left(1, \frac{\sqrt{n}}{8}\right)\mathbf{R}\mathbf{x} + 1/2$.

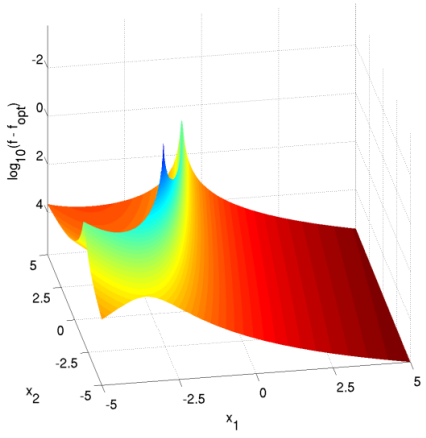
Ukázku prostorů funkčních hodnot pro funkce málo nebo středně podmíněné můžeme vidět na Obrázku 3.2.



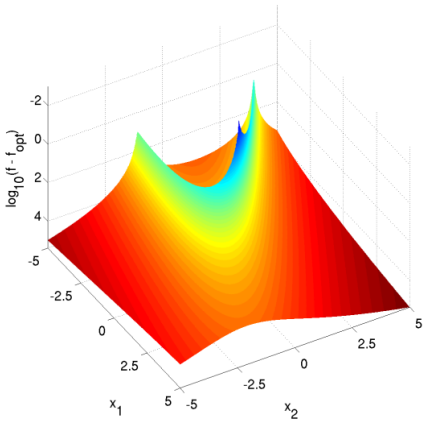
f_6 - Atraktivní sektor



f_7 - Step Elipsoid



f_8 - Rosenbrock



f_9 - Rosenbrock rotovaný

Obrázek 3.2: Ukázka prostoru funkčních hodnot pro jednotlivé málo nebo středně podmíněné funkce ve 2D.

3.3.1.3 Funkce velmi podmíněné a unimodální

Funkce velmi podmíněné a unimodální jsou následující:

10. **Elipsoid** - Globálně kvadratická funkce s hladkými lokálními nerovnostmi. Neseparovatelná varianta k funkci f_2.

$$f_{10}(\mathbf{x}) = \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} z_i^2 + f_{\text{opt}}, \tag{3.12}$$

kde

- $\mathbf{z} = T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$.

11. **Disk** - Globálně kvadratická funkce lokálními nerovnostmi. Jeden směr v prostoru funkčních hodnot je tisíckrát citlivější než všechny ostatní.

$$f_{11}(\mathbf{x}) = 10^6 z_1^2 + \sum_{i=2}^{n} z_i^2 + f_{\text{opt}}, \tag{3.13}$$

kde

- $\mathbf{z} = T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$.

12. **Hřbet** - Funkce připomínající hladký, ale velmi prudký hřbet.

$$f_{12}(\mathbf{x}) = z_1^2 + 10^6 \sum_{i=2}^{n} z_i^2 + f_{\text{opt}}, \tag{3.14}$$

kde

- $\mathbf{z} = \mathbf{R}T_{\text{asy}}^{0.5}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$.

13. **Ostrý hřbet** - Ostrý hřbet, který není diferencovatelný a gradient funkce je konstantní s přibližujícím se vrcholkem hřbetu.

$$f_{13}(\mathbf{x}) = z_1^2 + 100\sqrt{\sum_{i=2}^{n} z_i^2} + f_{\text{opt}}, \tag{3.15}$$

kde

- $\mathbf{z} = \mathbf{Q}\mathbf{A}^{10}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$.

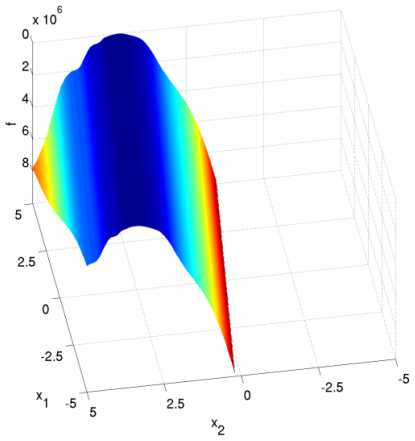
14. **Rozdílné mocniny** - Vzhledem k různým mocninám se funkční hodnota stává citlivější přibližováním se k globálnímu minimu.

$$f_{14}(\mathbf{x}) = \sqrt{\sum_{i=1}^{n} |z_i|^{2+4\frac{i-1}{n-1}}} + f_{\text{opt}}, \tag{3.16}$$

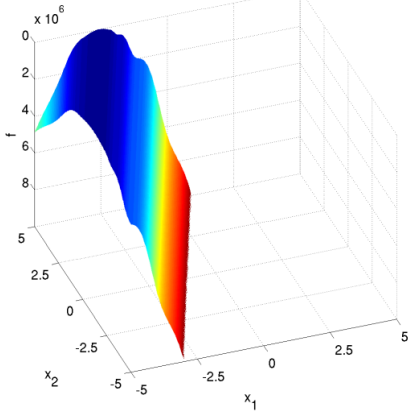
kde

- $\mathbf{z} = \mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$.

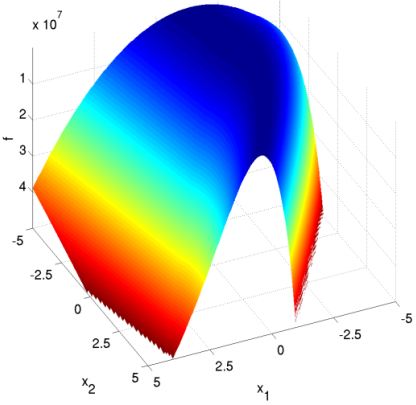
Ukázku prostorů funkčních hodnot pro funkce s velkou podmíněností a unimodalitou můžeme vidět na Obrázku 3.3.



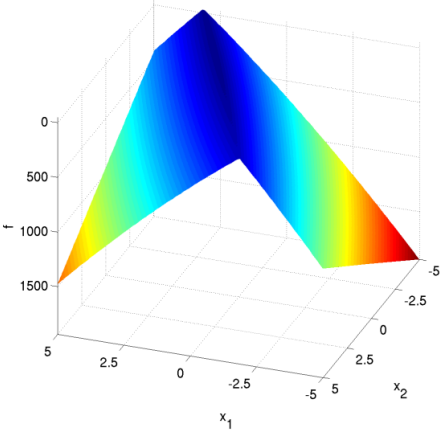
$f10$ - Elipsoid



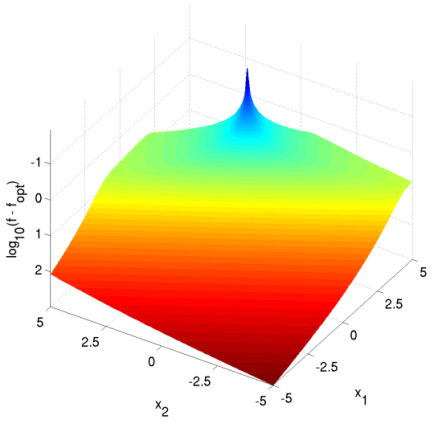
$f11$ - Disk



$f12$ - Hřbet



$f13$ - Ostrý hřbet



$f14$ - Rozdílné mocniny

Obrázek 3.3: Ukázka prostoru funkčních hodnot pro jednotlivé funkce vysoce podmíněné a unimodální.

3.3.1.4 Multimodální funkce s odpovídající globální strukturou

Vícemodální funkce s odpovídající globální strukturou jsou následující:

15. **Rastrigin** - Transformovaná Rastriginova funkce se sníženou symetrií a regularitou.

$$f_{15}(\mathbf{x}) = 10\left(n - \sum_{i=1}^{n}\cos(2\pi z_i)\right) + ||\mathbf{z}||^2 + f_{\text{opt}}, \tag{3.17}$$

kde

- $\mathbf{z} = \mathbf{R}\mathbf{A}^{10}\mathbf{Q}T_{\text{asy}}^{0.2}(T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})))$

16. **Weierstrass** - Vysoce nerovná a poměrně se opakující funkce s několika globálními optimy.

$$f_{16}(\mathbf{x}) = 10\left(\frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{11} 1/2^k \cos(2\pi 3^k(z_i + 1/2)) - f_0\right) + \frac{10}{n} f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}, \tag{3.18}$$

kde

- $\mathbf{z} = \mathbf{R}\mathbf{A}^{1/100}\mathbf{Q}T_{\text{osz}}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$,
- $f_0 = \sum_{k=0}^{11} 1/2^k \cos(2\pi 3^k 1/2)$.

17. **Schaffers F7** - Vysoce multimodální funkce s proměnnou frekvencí a amplitudou.

$$f_{17}(\mathbf{x}) = \left(\frac{1}{n-1}\sum_{i=1}^{n-1}\sqrt{s_i} + \sqrt{s_1}\sin^2(50 s_i^{1/5})\right)^2 + 10 f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}, \tag{3.19}$$

kde

- $\mathbf{z} = \mathbf{A}^{10}\mathbf{Q}T_{\text{asy}}^{0.5}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$,
- $s_i = \sqrt{z_i^2 + z_{i+1}^2}$.

18. **Schaffers F7 - špatně podmíněná** - Špatně podmíněná varianta předchozí funkce.

$$f_{18}(\mathbf{x}) = \left(\frac{1}{n-1}\sum_{i=1}^{n-1}\sqrt{s_i} + \sqrt{s_1}\sin^2(50 s_i^{1/5})\right)^2 + 10 f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}, \tag{3.20}$$

kde

- $\mathbf{z} = \mathbf{A}^{1000}\mathbf{Q}T_{\text{asy}}^{0.5}(\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}}))$,

- $s_i = \sqrt{z_i^2 + z_{i+1}^2}$.

19. **Složená Griewank-Rosenbrock F8F2** - Podobná Rosenbrockově funkci ve velmi multimodální variantě.

$$f_{19}(\mathbf{x}) = \frac{10}{n-1} \sum_{i=1}^{n-1} \left(\frac{s_i}{4000} - \cos(s_i) \right) + 10 + f_{\text{opt}}, \tag{3.21}$$

kde

- $\mathbf{z} = \max(1, \frac{\sqrt{n}}{8})\mathbf{R}\mathbf{x} + 0.5$,
- $s_i = 100(z_i^2 - z_{i+1}) + (z_i - 1)^2$,
- $\mathbf{z}^{\text{opt}} = \mathbf{1}$.

Ukázku prostorů funkčních hodnot pro vícemodální funkce s odpovídající globální strukturou můžeme vidět na Obrázku 3.4.

3.3.1.5 Multimodální se slabou globální strukturou

Multimodální funkce se slabou globální strukturou jsou následující:

20. **Schwefel** - Nejslibnější minima jsou umístěna blízko rohů oblasti, která není penizována.

$$f_{20}(\mathbf{x}) = -\frac{1}{100n} \sum_{i=1}^{n} z_i \sin(\sqrt{|z_i|}) + 4.189828872724339 \\ +100 f_{\text{pen}}(\mathbf{z}/100) + f_{\text{opt}}, \tag{3.22}$$

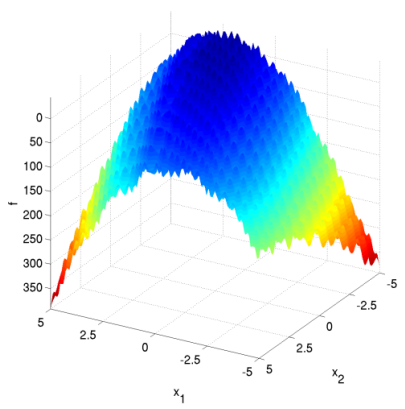
kde

- $\hat{\mathbf{x}} = 2 \times \mathbf{1}_-^+ \bigotimes \mathbf{x}$,
- $\hat{z}_1 = \hat{x}_1, \hat{z}_{i+1} = \hat{x}_{i+1} + 0.25(\hat{x}_i - 2|x_i^{\text{opt}}|)$,
- $\mathbf{z} = 100(\mathbf{A}^{10}(\hat{z} - 2|x^{\text{opt}}|) + 2|x^{\text{opt}}|)$,
- $\mathbf{x}^{\text{opt}} = 4.2096874633/2$.

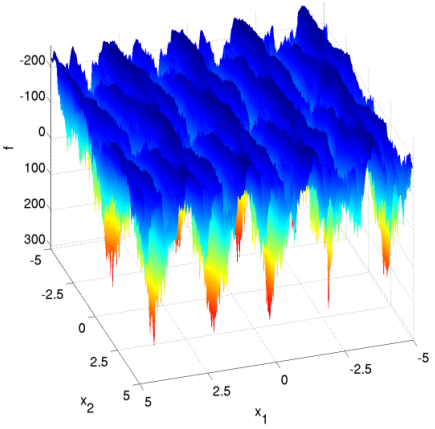
21. **Gallagherových 101 vrcholů** - Funkce se skládá ze 101 lokálních optim, u kterých neexistuje spojitost mezi jejich pozicí a výškou.

$$f_{21}(\mathbf{x}) = T_{\text{osz}} \left(10 - \max_{i=1}^{101} w_i \exp\left(-\frac{1}{2n} (\mathbf{x} - \mathbf{y}_i)^\top \mathbf{R}^\top \mathbf{C}_i \mathbf{R} (\mathbf{x} - \mathbf{y}_i) \right) \right)^2 \\ + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}, \tag{3.23}$$

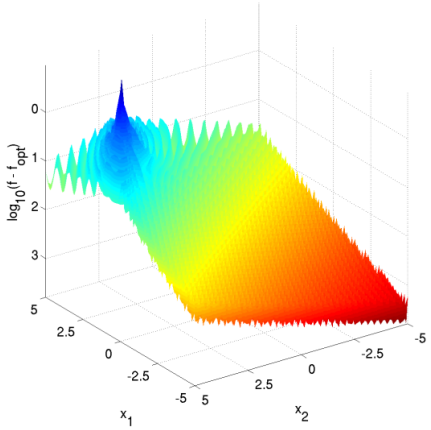
kde



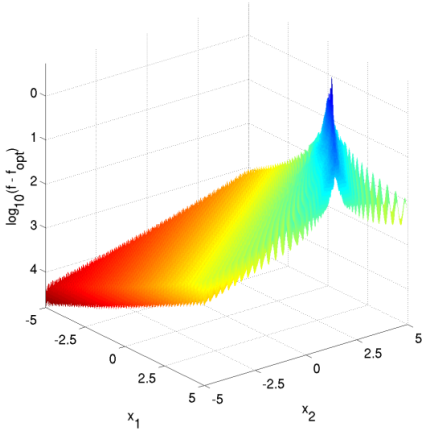
f_{15} - Rastrigin



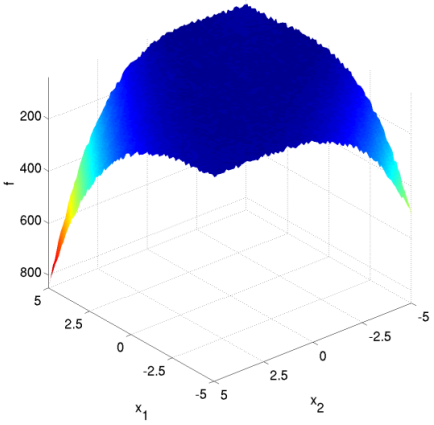
f_{16} - Weierstrass



f_{17} - Schaffers F7



f_{18} - Schaffers F7 - špatně podmíněná



f_{19} - Složená Griewank-Rosenbrock F8F2

Obrázek 3.4: Ukázka prostoru funkčních hodnot pro jednotlivé funkce vysoce podmíněné a unimodální ve 2D.

- $w_i = \begin{cases} 1.1 + 8 \times \frac{i-2}{99} \text{ pro } i = 2, ..., 101, \\ 10 \text{ pro } i = 1, \end{cases}$
- $\mathbf{C}_i = \mathbf{A}^{\alpha_i}/\alpha_i^{1/4}$, kde $\alpha_1 = 1000$, α_i je náhodná hodnota z $\{1000^{2\frac{j}{99}} | j = 0, ..., 99\}$ pro $i = 2, ..., 101$.

22. **Gallagherových 21 vrcholů** - Funkce se skládá ze 21 lokálních optim. Oproti předchozí je podstatně více podmíněná v okolí globálního optima.

$$f_{22}(\mathbf{x}) = T_{\text{osz}}\left(10 - \max_{i=1}^{21} w_i \exp\left(-\frac{1}{2n}(\mathbf{x} - \mathbf{y}_i)^\top \mathbf{R}^\top \mathbf{C}_i \mathbf{R}(\mathbf{x} - \mathbf{y}_i)\right)\right)^2 + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}, \tag{3.24}$$

kde

- $w_i = \begin{cases} 1.1 + 8 \times \frac{i-2}{19} \text{ pro } i = 2, ..., 101, \\ 10 \text{ pro } i = 1, \end{cases}$
- $\mathbf{C}_i = \mathbf{A}^{\alpha_i}/\alpha_i^{1/4}$, kde $\alpha_1 = 1000^2$, α_i je náhodná hodnota z $\{1000^{2\frac{j}{19}} | j = 0, ..., 19\}$ pro $i = 2, ..., 21$.

23. **Katsuura** - Velmi hrbolatá a opakující se funkce s více než 10^n globálními optimy.

$$f_{23}(\mathbf{x}) = \frac{10}{n^2}\Pi_{i=1}^n \left(1 + i \sum_{j=1}^{32} \frac{|2^j z_i - [2^j z_i]|}{2^j}\right)^{10/n^{1.2}} - \frac{10}{n^2} + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}}, \tag{3.25}$$

kde

- $\mathbf{z} = \mathbf{Q}\mathbf{A}^{100}\mathbf{R}(\mathbf{x} - \mathbf{x}^{\text{opt}})$.

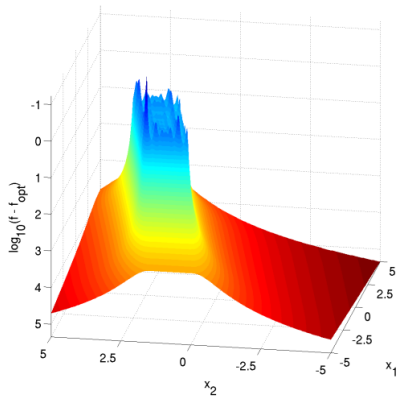
24. **Lunacek bi-Rastrigin** - Velmi multimodální funkce s 2 krátery.

$$f_{24}(\mathbf{x}) = \min\left(\sum_{i=1}^n (\hat{x}_i - \mu_0)^2, dn + s\sum_{i=1}^n (\hat{x}_i - \mu_1)^2\right) + 10\left(n - \sum_{i=1}^n \cos(2\pi z_i)\right) + 10^4 f_{\text{pen}} + f_{\text{opt}}, \tag{3.26}$$

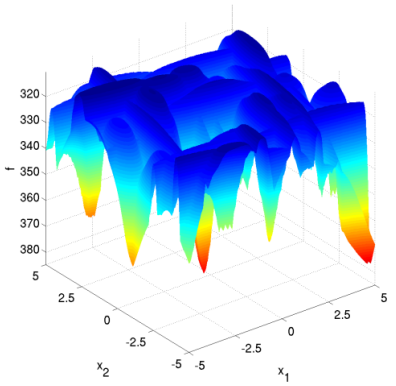
kde

- $\hat{\mathbf{x}} = 2\text{sign}(\mathbf{x}^{\text{opt}}) \bigotimes \mathbf{x}, \mathbf{x}^{\text{opt}} = \frac{\mu_0}{2}\mathbf{1}_-^+$,
- $\mathbf{z} = \mathbf{Q}\mathbf{A}^{100}\mathbf{R}(\hat{\mathbf{x}} - \mu_0\mathbf{1})$,
- $\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}, s = 1 - \frac{1}{2\sqrt{n+20}-8.2}, d = 1$.

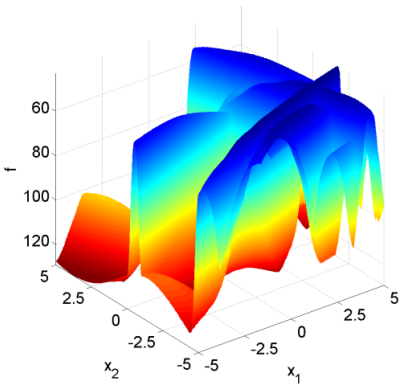
Ukázku prostorů funkčních hodnot pro multimodální funkce se slabou globální strukturou můžeme vidět na Obrázku 3.5.



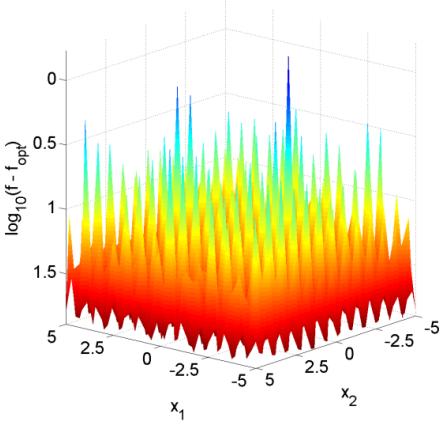
f_{20} - Schwefel



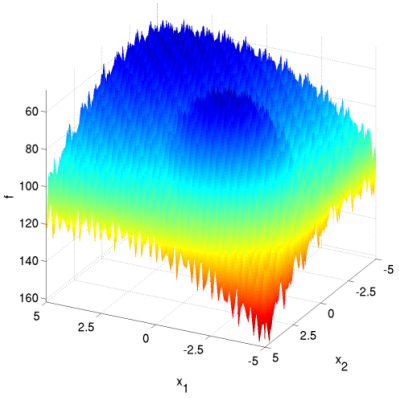
f_{21} - Gallagherových 101 vrcholů



f_{22} - Gallagherových 21 vrcholů



f_{23} - Katsuura



f_{24} - Lunacek bi-Rastrigin

Obrázek 3.5: Ukázka prostoru funkčních hodnot pro jednotlivé funkce multimodální se slabou globální strukturou ve 2D.

3.3.2 Funkce se šumem

Funkce se šumem jsou rozděleny následovně:

- Funkce se středním šumem (101-106)
- Funkce s velkým šumem (107-121)
- Funkce multimodální s velkým šumem (122-130)

Níže popsané předpisy funkcí se šumem dále využívají následující definice:

- $f_{\mathrm{GN}}(f, \beta) = f \times \exp(\beta \mathcal{N}(0,1))$
- $f_{\mathrm{UN}}(f, \alpha, \beta) = f \times \mathcal{U}(0,1)^{\beta} \max\left(1, \left(\frac{10^9}{f+\epsilon}\right)^{\alpha \mathcal{U}(0,1)}\right)$
- $f_{\mathrm{CN}}(f, \alpha, p) = f + \alpha \max\left(0, 1000 + \mathbb{I}_{\{\mathcal{U}(0,1)<p\}} \frac{\mathcal{N}(0,1)}{|\mathcal{N}(0,1)|+\epsilon}\right)$

3.3.2.1 Funkce se středním šumem

Funkce se středním šumem jsou následující:

1. **Koule se středním gaussovským šumem**

$$f_{101}(\mathbf{x}) = f_{\mathrm{GN}}(f_1(\mathbf{x}), 0.01) + f_{\mathrm{pen}}(\mathbf{x}) + f_{\mathrm{opt}} \tag{3.27}$$

2. **Koule se středním rovnoměrným šumem**

$$f_{102}(\mathbf{x}) = f_{\mathrm{UN}}\left(f_1(\mathbf{x}), 0.01\left(0.49 + \frac{1}{n}\right), 0.01\right) + f_{\mathrm{pen}}(\mathbf{x}) + f_{\mathrm{opt}} \tag{3.28}$$

3. **Koule se středním Cauchyho šumem**

$$f_{103}(\mathbf{x}) = f_{\mathrm{CN}}(f_1(\mathbf{x}), 0.01, 0.05) + f_{\mathrm{pen}}(\mathbf{x}) + f_{\mathrm{opt}} \tag{3.29}$$

4. **Rosenbrock se středním gaussovský šumem**

$$f_{104}(\mathbf{x}) = f_{\mathrm{GN}}(f_8(\mathbf{x}), 0.01) + f_{\mathrm{pen}}(\mathbf{x}) + f_{\mathrm{opt}} \tag{3.30}$$

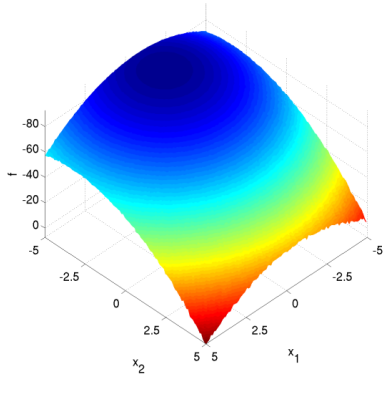
5. **Rosenbrock se středním rovnoměrným šumem**

$$f_{105}(\mathbf{x}) = f_{\mathrm{UN}}\left(f_8(\mathbf{x}), 0.01, \left(0.49 + \frac{1}{n}\right), 0.01\right) + f_{\mathrm{pen}}(\mathbf{x}) + f_{\mathrm{opt}} \tag{3.31}$$

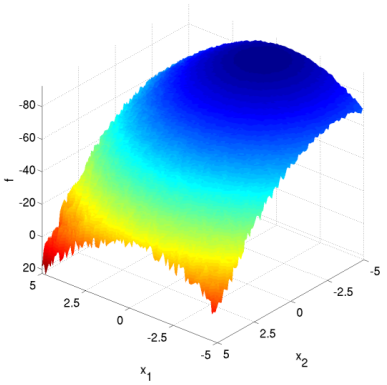
6. **Rosenbrock se středním Cauchyho šumem**

$$f_{106}(\mathbf{x}) = f_{\mathrm{CN}}\left(f_8(\mathbf{x}), 0.01, 0.05\right) + f_{\mathrm{pen}}(\mathbf{x}) + f_{\mathrm{opt}} \tag{3.32}$$

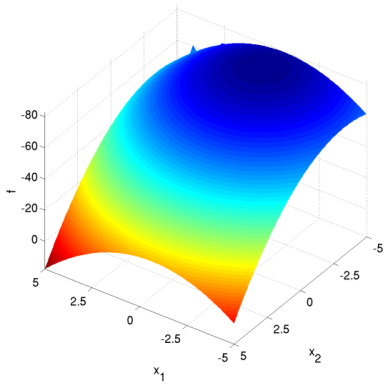
Ukázku prostorů funkčních hodnot pro funkce se středním šumem můžeme vidět na Obrázku 3.6.



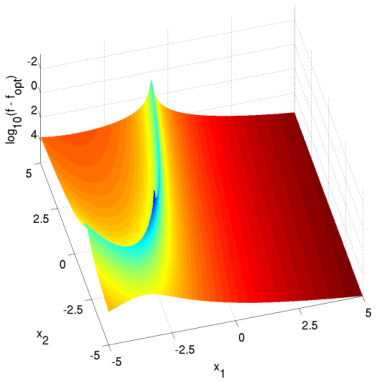
(a) f_{101} - Koule se středním gaussovským šumem



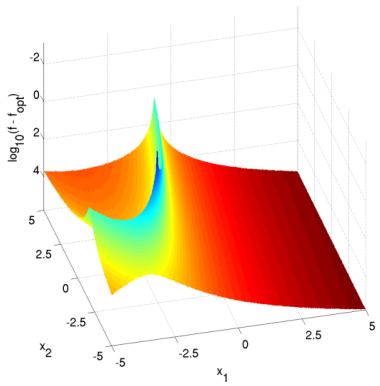
(b) f_{102} - Koule se středním rovnoměrným šumem



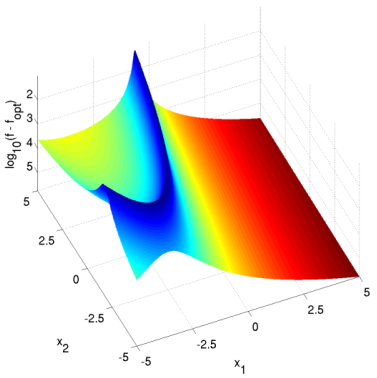
(c) f_{103} - Koule se středním Cauchyho šumem



(d) f_{104} - Rosenbrock se středním gaussovský šumem



(e) f_{105} - Rosenbrock se středním rovnoměrným šumem



(f) f_{106} - Rosenbrock se středním Cauchyho šumem

Obrázek 3.6: Ukázka prostoru funkčních hodnot pro jednotlivé funkce se středním šumem.

3.3.2.2 Funkce s velkým šumem

Funkce s velkým šumem jsou následující:

1. **Koule s velkým gaussovským šumem**

$$f_{107}(\mathbf{x}) = f_{\text{GN}}(f_1(\mathbf{x}), 1) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.33}$$

2. **Koule s velkým rovnoměrným šumem**

$$f_{108}(\mathbf{x}) = f_{\text{UN}}\left(f_1(\mathbf{x}), 0.49 + \frac{1}{n}, 1\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.34}$$

3. **Koule s velkým cauchyho šumem**

$$f_{109}(\mathbf{x}) = f_{\text{CN}}(f_1(\mathbf{x}), 1, 0.2) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.35}$$

4. **Rosenbrock s velkým gaussovský šumem**

$$f_{110}(\mathbf{x}) = f_{\text{GN}}(f_8(\mathbf{x}), 1) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.36}$$

5. **Rosenbrock s velkým rovnoměrným šumem**

$$f_{111}(\mathbf{x}) = f_{\text{UN}}\left(f_8(\mathbf{x}), 0.49 + \frac{1}{n}, 1\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.37}$$

6. **Rosenbrock s velkým Cauchyho šumem**

$$f_{112}(\mathbf{x}) = f_{\text{CN}}\left(f_8(\mathbf{x}), 1, 0.2\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.38}$$

7. **Step Elipsoid s velkým gaussovský šumem**

$$f_{113}(\mathbf{x}) = f_{\text{GN}}(f_7(\mathbf{x}), 1) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.39}$$

8. **Step Elipsoid s velkým rovnoměrným šumem**

$$f_{114}(\mathbf{x}) = f_{\text{UN}}\left(f_7(\mathbf{x}), 0.49 + \frac{1}{n}, 1\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.40}$$

9. **Step Elipsoid s velkým Cauchyho šumem**

$$f_{115}(\mathbf{x}) = f_{\text{CN}}\left(f_7(\mathbf{x}), 1, 0.2\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.41}$$

10. **Elipsoid s velkým gaussovský šumem**

$$f_{116}(\mathbf{x}) = f_{\text{GN}}(f_2(\mathbf{x}), 1) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.42}$$

11. **Elipsoid s velkým rovnoměrným šumem**

$$f_{117}(\mathbf{x}) = f_{\text{UN}}\left(f_2(\mathbf{x}), 0.49 + \frac{1}{n}, 1\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.43}$$

12. **Elipsoid s velkým Cauchyho šumem**

$$f_{118}(\mathbf{x}) = f_{\text{CN}}\left(f_2(\mathbf{x}), 1, 0.2\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.44}$$

13. **Rozdílné mocniny s velkým gaussovský šumem**

$$f_{119}(\mathbf{x}) = f_{\text{GN}}(f_{14}(\mathbf{x}), 1) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.45}$$

14. **Rozdílné mocniny s velkým rovnoměrným šumem**

$$f_{120}(\mathbf{x}) = f_{\text{UN}}\left(f_{14}(\mathbf{x}), 0.49 + \frac{1}{n}, 1\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.46}$$

15. **Rozdílné mocniny s velkým Cauchyho šumem**

$$f_{121}(\mathbf{x}) = f_{\text{CN}}\left(f_{14}(\mathbf{x}), 1, 0.2\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.47}$$

Ukázku prostorů funkčních hodnot pro vícemodální funkce s velkým šumem můžeme vidět na Obrázku 3.7.

3.3.2.3 Multimodální funkce s velkým šumem

Multimodální funkce s velkým šumem jsou následující:

1. **Schaffer F7 s velkým gaussovským šumem**

$$f_{122}(\mathbf{x}) = f_{\text{GN}}(f_{17}(\mathbf{x}), 1) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.48}$$

2. **Schaffer F7 s velkým rovnoměrným šumem**

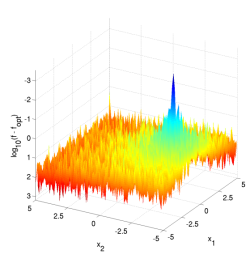
$$f_{123}(\mathbf{x}) = f_{\text{UN}}\left(f_{17}(\mathbf{x}), 0.49 + \frac{1}{n}, 1\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.49}$$

3. **Schaffer F7 s velkým cauchyho šumem**

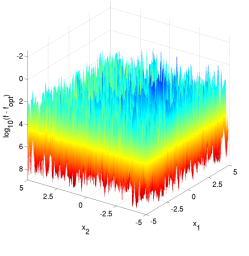
$$f_{124}(\mathbf{x}) = f_{\text{CN}}(f_{17}(\mathbf{x}), 1, 0.2) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.50}$$

4. **Griewank-Rosenbrock s velkým gaussovským šumem**

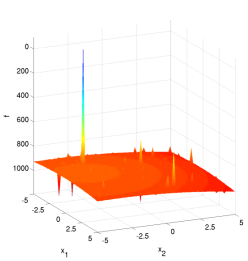
$$f_{125}(\mathbf{x}) = f_{\text{GN}}(f_{19}(\mathbf{x}), 1) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.51}$$



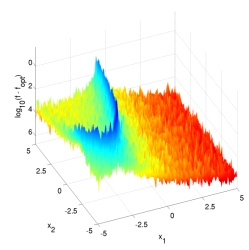
f_{107} - Koule s velkým gaussovským šumem



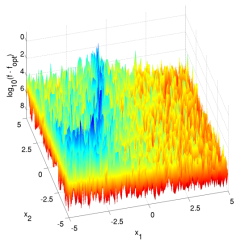
f_{108} - Koule s velkým rovnoměrným šumem



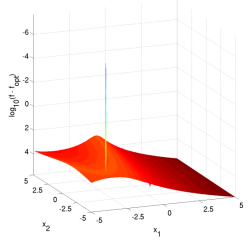
f_{109} - Koule s velkým cauchyho šumem



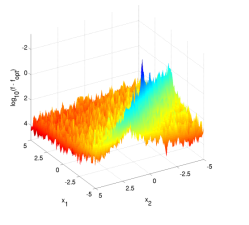
f_{110} - Rosenbrock s velkým gaussovským šumem



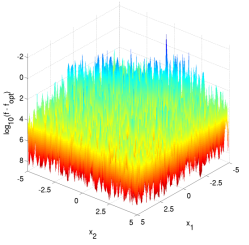
f_{111} - Rosenbrock s velkým rovnoměrným šumem



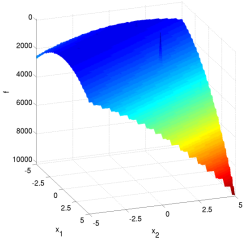
f_{112} - Rosenbrock s velkým Cauchyho šumem



f_{113} - Krokovitá elipsa s velkým gaussovským šumem



f_{114} - Krokovitá elipsa s velkým rovnoměrným šumem



f_{115} - Krokovitá elipsa s velkým Cauchyho šumem

Obrázek 3.7: Ukázka prostoru funkčních hodnot pro jednotlivé funkce s velkým šumem.

5. **Griewank-Rosenbrock s velkým rovnoměrným šumem**

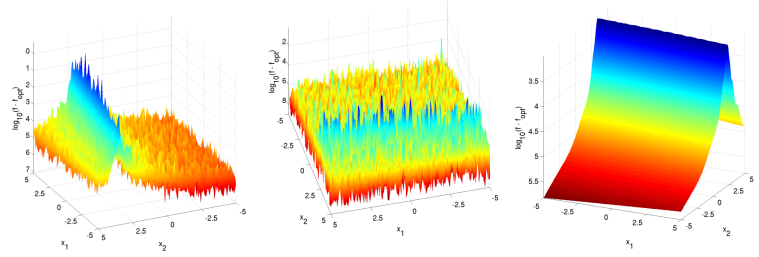
$$f_{126}(\mathbf{x}) = f_{\text{UN}}\left(f_{19}(\mathbf{x}), 0.49 + \frac{1}{n}, 1\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.52}$$

6. **Griewank-Rosenbrock s velkým cauchyho šumem**

$$f_{127}(\mathbf{x}) = f_{\text{CN}}(f_{19}(\mathbf{x}), 1, 0.2) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.53}$$

7. **Gallagherových 101 vrcholů s velkým gaussovským šumem**

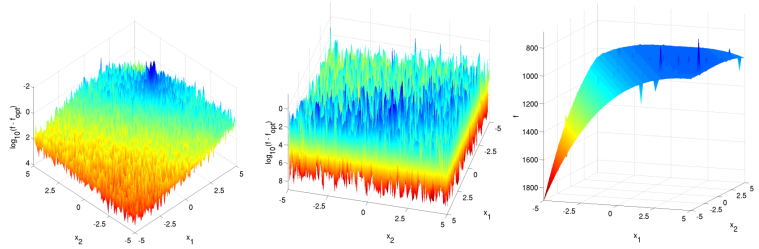
$$f_{128}(\mathbf{x}) = f_{\text{GN}}(f_{21}(\mathbf{x}), 1) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.54}$$



f_{116} - Elipsa s velkým gaussovským šumem

f_{117} - Elipsa s velkým rovnoměrným šumem

f_{118} - Elipsa s velkým Cauchyho šumem



f_{119} - Rozdílné mocniny s velkým gaussovským šumem

f_{120} - Rozdílné mocniny s velkým rovnoměrným šumem

f_{121} - Rozdílné mocniny s velkým Cauchyho šumem

Obrázek 3.7: Ukázka prostoru funkčních hodnot pro jednotlivé funkce s velkým šumem.

8. **Gallagherových 101 vrcholů s velkým rovnoměrným šumem**

$$f_{129}(\mathbf{x}) = f_{\text{UN}}\left(f_{21}(\mathbf{x}), 0.49 + \frac{1}{n}, 1\right) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.55}$$

9. **Gallagherových 101 vrcholů s velkým cauchyho šumem**

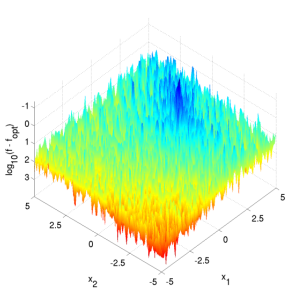
$$f_{130}(\mathbf{x}) = f_{\text{CN}}(f_{21}(\mathbf{x}), 1, 0.2) + f_{\text{pen}}(\mathbf{x}) + f_{\text{opt}} \tag{3.56}$$

Ukázku prostorů funkčních hodnot pro vícemodální funkce s velkým šumem můžeme vidět v Obrázku 3.8.

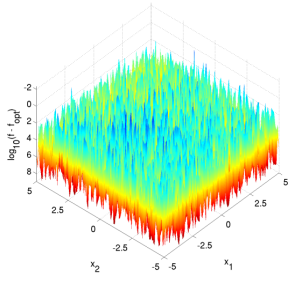
3.3.3 Nové funkce založené na reálných problémech

Framework BBOB obsahuje velké množství funkcí, které zkoumají různé vlastnosti testovaných algoritmů. Všechny funkce jsou ale uměle vytvořené a čistě teoretické. Ačkoliv jsou znalosti získané za pomoci těchto funkcí nesmírně důležité, tak velmi podstatná zůstává výkonnost algoritmu na reálných datech.

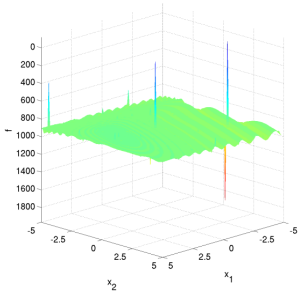
Tato práce představuje jeden z pokusů rozšířit framework BBOB o nové funkce, které odpovídají reálným problémům a zároveň jsou vhodné k ben-



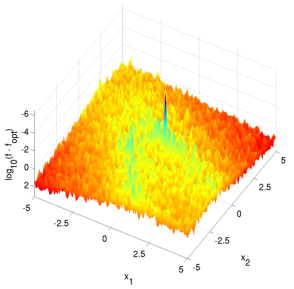
f_{122} - Schaffer F7 s velkým gaussovským šumem



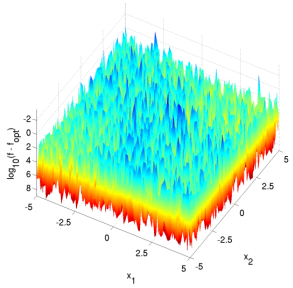
f_{123} - Schaffer F7 s velkým rovnoměrným šumem



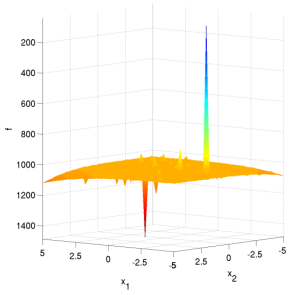
f_{124} - Schaffer F7 s velkým cauchyho šumem



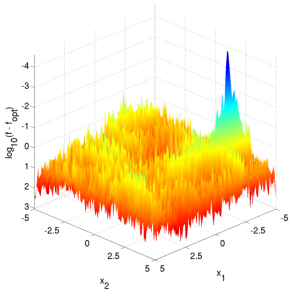
f_{125} - Griewank-Rosenbrock s velkým gaussovským šumem



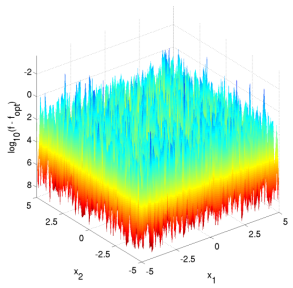
f_{126} - Griewank-Rosenbrock s velkým rovnoměrným šumem



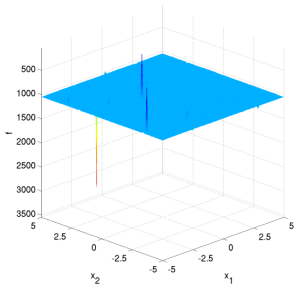
f_{127} - Griewank-Rosenbrock s velkým Cauchyho šumem



f_{128} - Gallagherových 101 vrcholů s velkým gaussovským šumem



f_{129} - Gallagherových 101 vrcholů s velkým rovnoměrným šumem



f_{130} - Gallagherových 101 vrcholů s velkým Cauchyho šumem

Obrázek 3.8: Ukázka prostoru funkčních hodnot pro jednotlivé multimodální funkce s vysokým šumem.

chmarkování. Všechny funkce pochází z práce vydané v roce 2018 [27]. Integrace do frameworku BBOB vyžaduje zachování jeho základních principů (např. vztah funkcí a instancí popsaný na začátku sekce 3.1). Proto jsem musel tyto funkce upravit, aby tyto principy splňovaly a mohly být začleněny do frameworku v maximální možné míře. Zachováno bylo i pravidlo, že číslo instance pevně určuje podobu funkce. Definice funkcí a jejich instancí jsou uvedeny na následujících řádcích.

3.3.3.1 Součet čtverců - shlukování

Spojitý problém součtu čtverců při shlukování může být popsán následovně. Pro danou množinu $\mathcal{X} = \mathbf{x}_1, ..., \mathbf{x}_n \subseteq \mathbb{R}^d$ obsahující p bodů je cílem nalézt souřadnice k středů shluků v prostoru $\mathcal{C} = \mathbf{c}_1, ..., \mathbf{c}_k \in \mathbb{R}^d$ takových, aby byl následující součet minimální:

$$f(\mathcal{C}|\mathcal{X}) = \sum_{i=1}^{n}\sum_{j=1}^{k} b_{i,j}.||\mathbf{x}_i - \mathbf{c}_j||^2, \tag{3.57}$$

kde

$$b_{i,j} = \begin{cases} 1, \text{pokud} ||\mathbf{x}_i - \mathbf{c}_j|| = \min_{l \in 1,...,k} ||\mathbf{x}_i - \mathbf{c}_l||, \\ 0, \text{jinak} \end{cases} \tag{3.58}$$

a $||.||$ je euklidovská vzdálenost. Neznámé tohoto problému jsou tedy souřadnice středů. Pokud označíme souřadnice í-tého středu jako $\mathbf{c}_i = (y_i, y_{i+1}, ..., y_d)$, tak můžeme problém zapsat jako spojitý neomezený optimalizační problém dimenze $n = dk$, kde cílem je:

$$\min f(\mathbf{y}), \mathbf{y} \in \mathbb{R}^n. \tag{3.59}$$

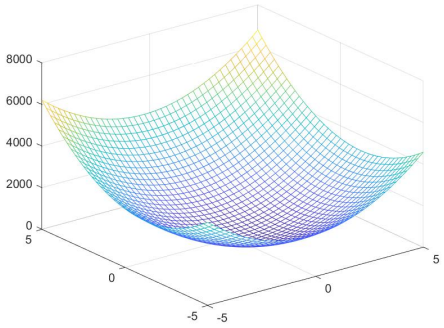
Každá instance problému je určena množinou dat $\mathcal{X}$ a počtem středů k. Celková dimenze problému roste úměrně s dimenzí dat d a počtem středů k, ale je nezávislá na počtu bodů p.

Jelikož dataset přímo definuje funkci, tak je každá množina dat reprezentována vlastní funkcí. Jednotlivé instance náhodně rotují celý dataset, tudíž optimální řešení zůstává pro všechny instance stejné. Tato práce využívá tři různé množiny dat [28]:

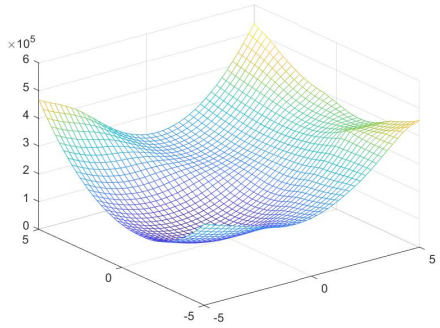
1. Iris - Celkem 150 bodů dimenze $d = 4$.
   $f_{201}(\mathbf{x})$ = předpis 3.57, kde $x_i \in \mathcal{X}_{\text{Iris}}$
2. Ruspiny = Celkem 75 bodů dimenze $d = 2$.
   $f_{202}(\mathbf{x})$ = předpis 3.57, kde $x_i \in \mathcal{X}_{\text{Ruspiny}}$
3. Německá města = Celkem 89 bodů dimenze $d = 3$.
   $f_{203}(\mathbf{x})$ = předpis 3.57, kde $x_i \in \mathcal{X}_{\text{German}}$

Požadovaná dimenze funkce spolu s datasetem přimo určuje počet středů, takže např. dataset Iris spolu s dimenzí 12 vytvoří funkci, která hledá 3 středy.

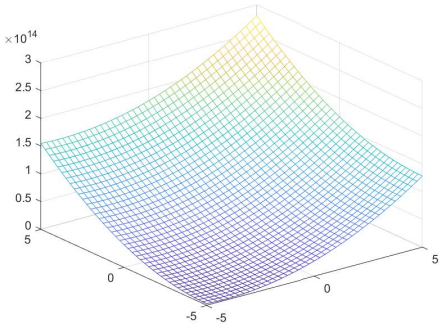
Graf znázorňující prostor funkčních hodnot je uveden na Obrázku 3.9.



(a) Iris



(b) Ruspiny



(c) Německá města

Obrázek 3.9: Ukázka prostoru funkčních hodnot pro jednotlivé množiny dat funkce shlukování součtu čtverců. Jelikož mají datasety Iris a Německá města dimenzi větší než 2, tak jsou jejich grafy projekcí dvou proměnných zatímco zbytek byl zafixován na optimální hodnotě.

3.3.3.2 Trénování vícevrstvého perceptronu

Vícevrstvý perceptron s jedním vstupem, jednou skrytou vrstvou o $k \geq 1 \in \mathbb{N}$ perceptronech a jedním výstupem mapuje vstupní data podle následujícího vzorce:

$$f(\mathbf{x}) = w_0 + \sum_{i=0}^{k-1} w_{3i+1}(w_{3i+2} + xw_{3i+3}) \tag{3.60}$$

Cílem je nalézt parametry $\mathbf{w} = \{w_0, ..., w_{3k+1}\} \in \mathbb{R}$ takové, že chyba

$$\epsilon = \frac{1}{p}\sum_{i=1}^{p}(f(x_i) - y_i)^2 \tag{3.61}$$

je minimální, kde $p \in \mathbb{N}$ je počet dat.

Trénování perceptronové sítě je funkce $f_{204}(\mathbf{x}) =$ předpis 3.61, kde $f(x_i)$ a y_i je dáno vybraným datasetem. Deset datasetů vygenerovaných pomocí

Číslo instance	Funkce	Předpis
1	Druhá mocnina	$f(x) = x^2$
2	Sinus	$f(x) = \sin(x)$
3	Absolutní hodnota	$f(x) = \lvert x \rvert$
4	Hyperbolický sinus	$f(x) = \sinh(x)$
5	Cosinus	$f(x) = \cos(x)$
6	Tangens	$f(x) = \tan(x)$
7	Hyperbolický cosinus	$f(x) = \cosh(x)$
8	Hyperbolický tangens	$f(x) = \tanh(x)$
9	Cotangens	$f(x) = \cot(x)$
10	Hyperbolický cotangens	$f(x) = \coth(x)$

Tabulka 3.1: Jednotlivé datasety funkce f_{204}.

jednoduchých funkcí definuje jednotlivé instance. Příslušnosti instancí k jednotlivým funkcím jsou uvedeny v tabulce 3.1.

Graf znázorňující prostor funkčních hodnot je zobrazen na Obrázku 3.10.

3.3.3.3 Vícedimenzionální škálování - Sammonovo mapování

Metody vícedimenzionálního škálování se snaží o nalezení nízkodimenzionální reprezentace dat - například nalezení 2-D či 3-D reprezentace datasetu. Jedno z kritérií pro určení podobnosti reprezentace je Sammonovo mapování. Cílem optimalizace je minimalizovat funkci danou předpisem

$$E(z) = \sum_{r,s} \frac{(||\mathbf{z}^r - \mathbf{z}^s|| - ||\mathbf{x}^r - \mathbf{x}^s||)^2}{||\mathbf{x}^r - \mathbf{x}^s||^2}, \tag{3.62}$$

kde $\mathbf{x}^r$ a $\mathbf{x}^s$ je dvojice bodů z originálního prostoru a $\mathbf{z}^r$ a $\mathbf{z}^s$ je dvojice bodů z nového redukovaného prostoru. Výsledná dimenze optimalizačního problému je rovna

$$\text{n} = \text{počtu bodů} \times \text{požadovaná dimenze.} \tag{3.63}$$

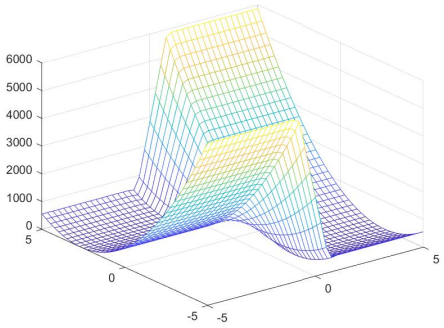
Sammonovo mapování představuje funkci $f_{205}(\mathbf{x})$ = předpis 3.62, kde $\mathbf{x}$ jsou body vybraného datasetu. Dostupné jsou dva různé datasety [27]:

1. **Virus** - Celkem 38 bodů
2. **Banka** - Celkem 80 bodů

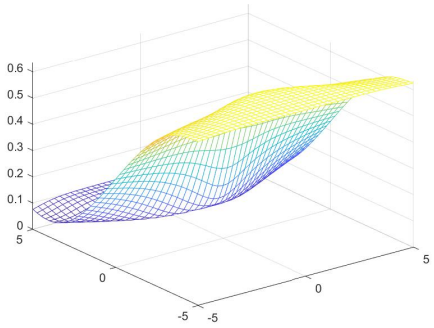
Tato práce experimentuje pouze s mapováním do 2D prostoru. Jelikož počet bodů určených k mapování přímo ovlivňuje dimenzi funkce

$$n = \text{počet bodů} \times 2, \tag{3.64}$$

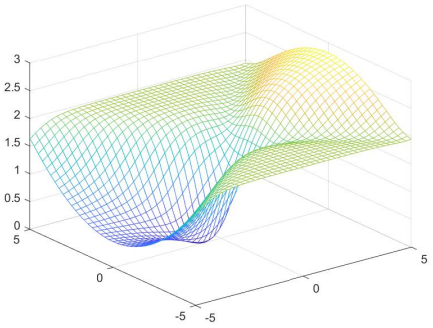
tak je z celého datasetu vždy vybrán pouze potřebný počet bodů. Číslo instance slouží k inicializaci náhodného generátoru. Čísla instancí menší než 250 využívají dataset Virus, čísla větší zase Banku.



(a) Absolutní hodnota



(b) Druhá mocnina



(c) Sinus

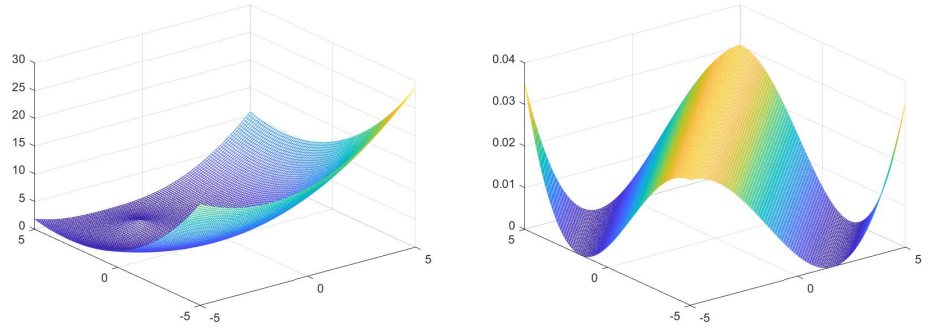
Obrázek 3.10: Ukázka prostoru funkčních hodnot pro vybrané množiny dat funkce f_{204}. Grafy znázorňují různé druhy prostorů, které je možné optimalizovat změnou instance.

Graf znázorňující prostor funkčních hodnot je zobrazen na Obrázku 3.11.

3.3.3.4 Mapa země

NASA poskytuje pro vědecké účely výškovou mapu země (Global Digital Elevation Map - GDEM [25]). Na této mapě je možné zadefinovat optimalizační úlohu, která má za cíl najít nejvyšší bod. Specifické na této úloze je to, že gradient je nulový v celém oboru hodnot, protože nadmořské výšky existují pouze pro diskrétní hodnoty souřadnic.

Mapa země představuje funkci $f_{206}(\mathbf{x}) = y[x_1, x_2]$ dimenze 2, pro kterou je výsledná funkční hodnota rovna prvku v tabulce y s indexem x_1 a x_2. Instance číslo 1 přímo odpovídá úloze představené v [27]. Ostatní čísla instancí využívají vlastní mapu, která byla vytvořena redukováním té originální s rozlišením na konečné rozlišení 11520x5344. Tato mapa je v dalších instancích znovu zmenšena (číslo instance -1) krát. Jednotlivá rozlišení jsou vypsána v tabulce 3.2.



(a) Virus (b) Banka

Obrázek 3.11: Ukázka prostoru funkčních hodnot pro vybrané množiny dat

Číslo instance	Rozlišení mapy
1	2160x4320
2	5344x11520
3	2672x5760
4	1782x3840
5	1366x2880
6	1069x2304
7	891x1920
8	764x1646
9	668x1440
10	594x1280
11	535x1152
12	486x1048
13	446x960
14	412x887
15	382x823

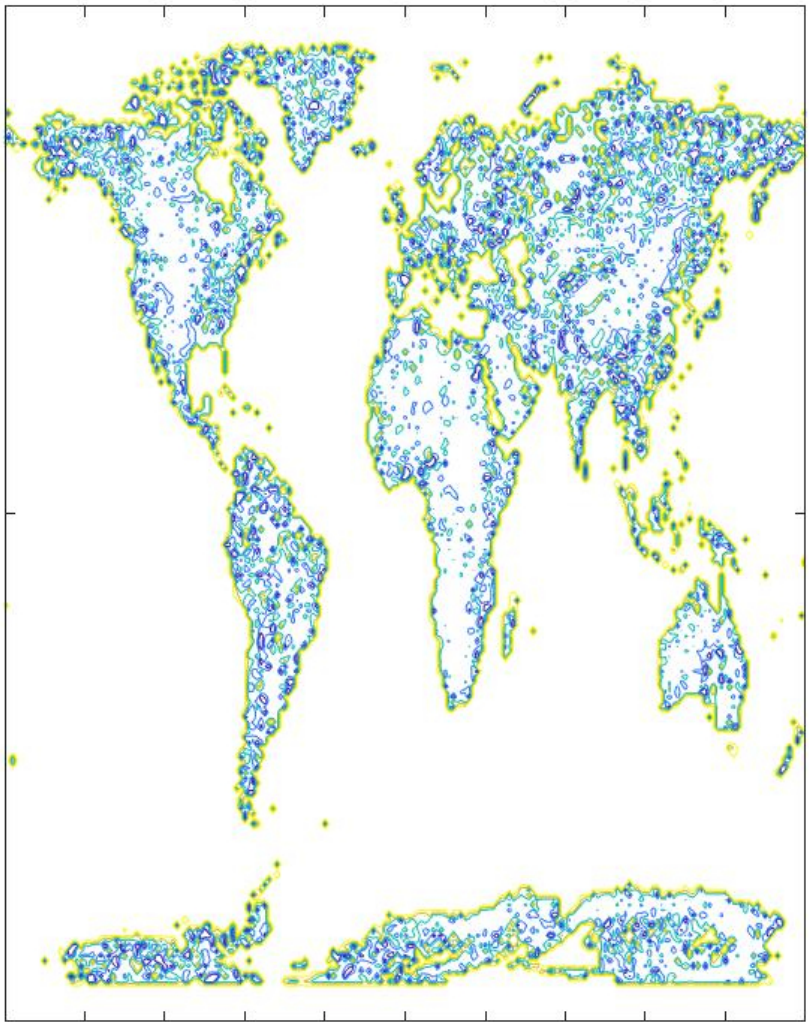
Tabulka 3.2: Použitá rozlišení mapy v závislosti na vybrané instanci funkce f_{206}.

Graf znázorňující prostor funkčních hodnot je znázorněn na Obrázku 3.12.

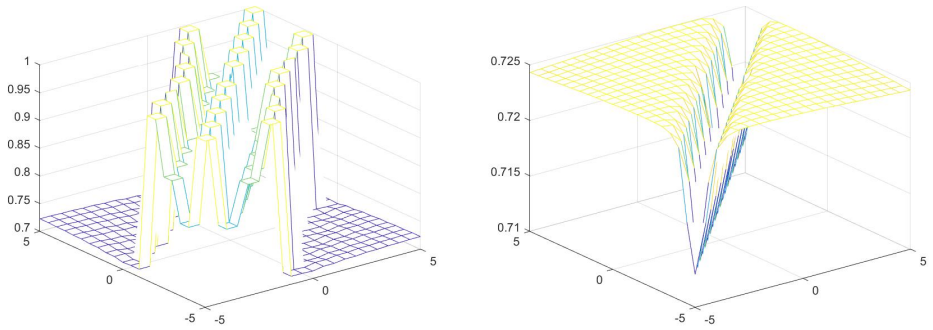
3.3.3.5 Rozmístění bójek

Skupina univerzity v Adelaide zaměřená na optimalizaci a logistiku [29] poskytuje kód určený k simulaci energie z vln získané pomocí bójek rozmístěných v oceánu. Cílem optimalizace je nalézt rozmístění bójek takové, aby byla vygenerovaná energie co největší.

Nalezení ideální pozice jedné bójky je jednoduchý úkol, ale jeho dimenze i obtížnost se zvětšují s rostoucím počtem bójek, protože se navzájem ovlivňují a



Obrázek 3.12: Ukázka prostoru funkčních hodnot výškové mapy. Funkční hodnota souřadnic je přímo úměrná nadmořské výšce. Žlutá barva znázorňuje funkční hodnoty menší než nula a modrá zase pohoří (čím tmavší, tím větší funkční hodnota). Oceány mají funkční hodnotu 0, protože NASA tyto data neposkytuje.



Obrázek 3.13: Ukázka prostoru funkčních hodnot pro umístění 2 bójek. Jedná se o projekci dvou různých proměnných, zatímco zbylé dvě jsou zafixované v optimu.

mění tak celkový vzhled funkce. Jelikož je simulace získané energie výpočetně náročná, tak se jedná o ideálního kandadíta pro využití náhradních modelů.

Umístění bójek představuje funkci $f_{207}(\mathbf{x}) = \text{buoyes}(\mathbf{x}, \text{ssp}, \text{vsp}, \text{rjb}, \text{fs})$, kde buoyes je funkce simulátoru a ssp, vsp, rjb a fs jsou parametry dané číslem instance. Poskytnutý simulátor obsahuje několik různých nastavení a číslo instance umožňuje jejich využití následovně:

1. Číslo na místě jednotek - Šířka simulované plochy (ssp). Číslo představuje index(začínající od 1) do pole šířek: $[10, 25, 50, 100, 200, 350, 500, 1000, 2500]$
2. Číslo na místě desítek - Výška simulované plochy (vsp). Číslo představuje index(začínající od 1) do pole výšek: $[10, 25, 50, 100, 200, 350, 500, 1000, 2500]$
3. Číslo na místě stovek - Radius jedné bójky (rjb). Číslo představuje index(začínající od 1) do pole rádiusů: $[2.0, 2.5, 3.2, 4.0, 5.0]$
4. Číslo na místě tisíců - Frekvence simulace (fs). Číslo představuje index(začínající od 1) do pole simulací: $[1, 2, 3, 5, 10, 25]$

Například instance číslo 2211 znamená, že simulátor bude využívat frekvenci 2, velikost bójky 2.5 a vstupy budou škálovány do prostoru $[10, 10]$.

Graf znázorňující prostor funkčních hodnot je znázorněn na Obrázku 3.13.

3.4 Konfigurace

Za účelem komplexnějšího srovnání byly kromě 6 kombinací evolučních kontrol a náhradních modelů vypsaných výše otestovány také 4 další kombinace. Tři z nich reprezentují originální varianty CMA-ESu: DTS-CMA-ES, lmm-CMA-ES a lq-CMA-ES. Jelikož původní verze DTS-CMA-ES pracuje s rozptylem

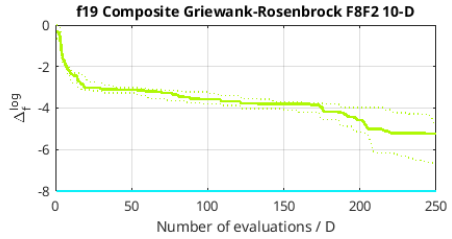
náhradního modelu a zbylé varianty CMA-ES ne, tak byla pro lepší porovnání přidána také varianta DTS-CMA-ES, která využívá pouze funkční hodnoty modelu.

Celkem bylo otestováno následujících 10 kombinací:

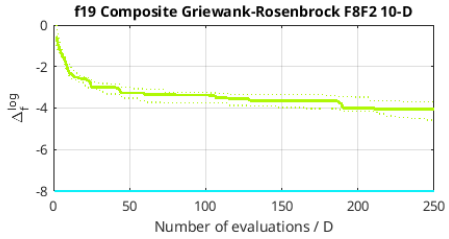
- **DTS + gp** → Evoluční kontrola z DTS-CMA-ES spolu s Gaussovskými procesy využívající pravděpodobnost vylepšení jako kritérium nejistoty.
- **DTS + gp (fvalues)** → Evoluční kontrola z DTS-CMA-ES spolu s Gaussovskými procesy využívající pouze funkční hodnoty modelu jako kritérium nejistoty.
- **DTS + lmm** → Evoluční kontrola z DTS-CMA-ES spolu s lokálním metamodelem.
- **DTS + linQuad** → Evoluční kontrola z DTS-CMA-ES spolu s lineárně kvadratickým modelem.
- **LinQuad + gp** → Evoluční kontrola z lq-CMA-ES spolu s Gaussovskými procesy.
- **LinQuad + lmm** → Evoluční kontrola z lq-CMA-ES spolu s lokálním metamodelem.
- **LinQuad + linQuad** → Evoluční kontrola z lq-CMA-ES spolu s lineárně kvadratickým modelem. Duplikuje celý lq-CMA-ES.
- **Lmm + gp** → Evoluční kontrola z LMM-CMA-ES spolu s Gaussovskými procesy.
- **Lmm + lmm** → Evoluční kontrola z LMM-CMA-ES spolu s lokálním metamodelem. Duplikuje celý LMM-CMA-ES.
- **Lmm + linQuad** → Evoluční kontrola z LMM-CMA-ES spolu s lineárně kvadratickým modelem.

Každá funkce frameworku BBOB byla testována na instancích 1-5 a 71-80 a výsledkem je průměr ze všech různých běhů. Funkce nově integrované byly otestovány na následujících instancích:

- f_{201} = 1-5 a 71-80
- f_{202} = 1-5 a 71-80
- f_{203} = 1-5 a 71-80
- f_{204} = 1-10
- f_{205} = 1-10 a 251-260



(a) LinQuad



(b) lmm

Obrázek 3.14: Ukázka vlivu inicializace nulovým vektorem. Zelená křivka znázorňuje škálovaný logaritmus vzdálenosti od optima vlastní implementace a zelené té originální. Můžeme zde vidět, že díky inicializaci nulovým vektorem algoritmus začíná přímo v globálním optimu.

- f_{206} = 1-15
- f_{207} = 1111, 1122, 1133, 1144, 1211, 1222, 1233, 1244, 1311, 1322, 1333, 1344, 2111, 2122, 2133, 2144, 2211, 2222, 2233, 2244, 2311, 2322, 2333, 2344

Všechny kombinace řízení evoluce a náhradního modelu využívají totožné nastavení CMA-ESu, které jsou následující:

- Počáteční velikost kroku $\sigma = 8/3$.
- Počet jedinců v generaci je $\lambda = 8 + (6\log(n))$.
- Vektor počáteční generace je roven $8\text{rand}(n) - 4$, kde $\text{rand}(x)$ je funkce vracející vektor délky x s rovnoměrným rozdělením v intervalu 0-1.

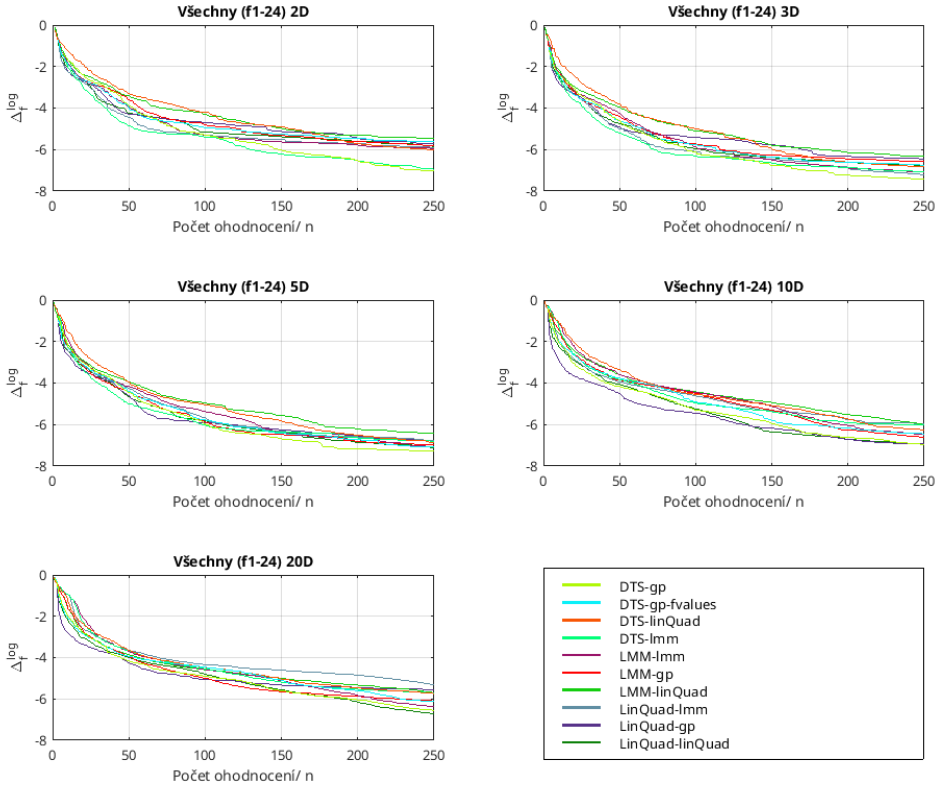
3.5 Výsledky

V této sekci jsou uvedeny vybrané výsledky, které nejlépe reprezentují data získaná z experimentů. Jedná se o grafy znázorňující průměrnou vzdálenost od optima nebo tabulky s počtem dosažených prvních míst. Kompletní sada grafů je dostupná online, případně na přiloženém CD.

3.5.1 Funkce bez šumu

Nejlepších výsledků pro funkce bez šumu dosáhla kombinace dvojité evoluční kontroly a gaussovských procesů. V žádné z testovaných skupin funkcí výrazně nezaostávala a v mnoha byla tou nejlepší kombinací.

Zajímavé jsou výsledky dvojitě trénované evoluční kontroly a lokálního metamodelu, protože tato kombinace dopadla podstatně lépe, než originální



Obrázek 3.15: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro funkce bez šumu.

LMM-CMA-ES. Také měla výrazně nejlepší výsledky na začátku optimalizace (počet evaluací/$n < 50$) pro malé dimenze ($n <= 5$).

Lineárně kvadratické řízení evoluce spolu s gaussovskými procesy předčilo všechny v oblasti multimodálních funkcí se slabou strukturou.

Dvojitě trénované evoluční kontrole s lineárně kvadratickým modelem se dařilo především ve funkcích obsahujících mnoho vrcholů, případně vysokých dimenzích f_{17} a f_{18}.

Originální LMM-CMA-ES byl ve všech případech průměrný. Žádné kategorii úplně nedominoval, ale zase v žádné ani nepropadl.

Řízení evoluce převzaté z lmm-CMA-ESu spolu s lineárně kvadratickým modelem dopadlo nejhůře, v žádné funkci nepatřil mezi nejlepší kombinace.

Algoritmus lq-CMA-ES obsadil první místo u funkcí málo nebo středně podmíněných a obecně patřil mezi ty úspěšnější kombinace.

Grafy vývoje vzdálenosti od optima, tabulku se součtem pořadí a jednotlivé skupiny můžeme vidět na 3.3, 3.15, 3.16, 3.17, 3.18, 3.19 a 3.20.

Tabulka 3.3: Součet prvních míst algoritu pro různé poměry počtu evaluace funkcí / dimenze (FE/D) a dimenze pro funkce bez šumu. V případě, že prvního místa dosáhne více algoritmů, tak je první místo započteno všem. Většinou je to z důvodu nalezení globálního optima.

	2D				3D				5D			
FE/D	25	50	100	200	25	50	100	200	25	50	100	200
DTS-gp	138.5	134	118	**103.5**	139.5	136	118.5	**108**	135.5	125	**97.5**	**99**
DTS-gp-fvalues	127.5	148	149	154.5	112.5	140	128.5	124.5	124.5	132	123.5	116.5
DTS-linQuad	214.5	183	164.5	148.5	218.5	193	179	154.5	197.5	194	187	144.5
DTS-lmm	**62.5**	**77.5**	**108**	113	**68.5**	**78**	105	122	**77.5**	**83**	126	144
LMM-lmm	118.5	111	112	130.5	146.5	135	118	110	149.5	148	142.5	120.5
LMM-gp	135.5	150	149.5	131.5	109.5	133	133	140.5	110.5	113	115.5	124.5
LMM-linQuad	185.5	179	166.5	164.5	180.5	180	185	173	177.5	169	188	183.5
LinQuad-lmm	96.5	108	125	128.5	103.5	89	**99.5**	122.5	126.5	131	133.5	141
LinQuad-gp	111.5	128.5	145.5	145	121.5	134	164.5	160.5	110.5	119	128.5	142
LinQuad-linQuad	129.5	133	138	138.5	119.5	126	125	126.5	110.5	106	126	124.5
	10D				20D				$\sum$			
FE/D	25	50	100	200	25	50	100	200	25	50	100	200
DTS-gp	78	85.5	100.5	**105**	103	**91**	**102.5**	**100.5**	594.5	571.5	**537**	**516**
DTS-gp-fvalues	89	104.5	113.5	118.5	85	95	108.5	107.5	538.5	619.5	623	621.5
DTS-linQuad	194	173.5	148.5	142	147	128	122.5	135.5	971.5	871.5	801.5	725
DTS-lmm	104	108.5	133.5	147.5	139	138	140.5	134.5	**451.5**	**485**	613	661
LMM-lmm	210	190.5	161.5	143.5	208.5	189	152.5	126.5	833	773.5	686.5	631
LMM-gp	157.5	143	138	120	139.5	127	108.5	119.5	652.5	666	644.5	636
LMM-linQuad	184	192.5	177.5	156.5	141.5	148	156.5	148.5	869	868.5	873.5	826
LinQuad-lmm	120	140.5	148.5	165	171.5	177	175.5	177.5	618	645.5	682	734.5
LinQuad-gp	**72**	**75.5**	98.5	124	**78**	91.5	129	156	493.5	548.5	666	727.5
LinQuad-linQuad	107	101.5	**95.5**	105.5	114.5	131	119.5	109.5	581	597.5	604	604.5

3.5.2 Funkce se šumem

V kategorii funkcí se šumem dosáhl celkově největší úspěšnosti algoritmus lq-CMA-ES, který byl v průměru nejlepší od $\mathrm{FE}/D > 100$.

Podobně jako u funkcí bez šumu dominovala kombinace dvojitě trénované evoluční kontroly s lokálním metamodelem počátečním průběhům optimalizace.

Dvojitě trénovaná evoluční kontrola s gaussovskými procesy obsadila druhé místo a např. na funkcích $f_{105} - 3D$, $f_{107} - 2D$ a $f_{116} - 2D$ byla nejlepší.

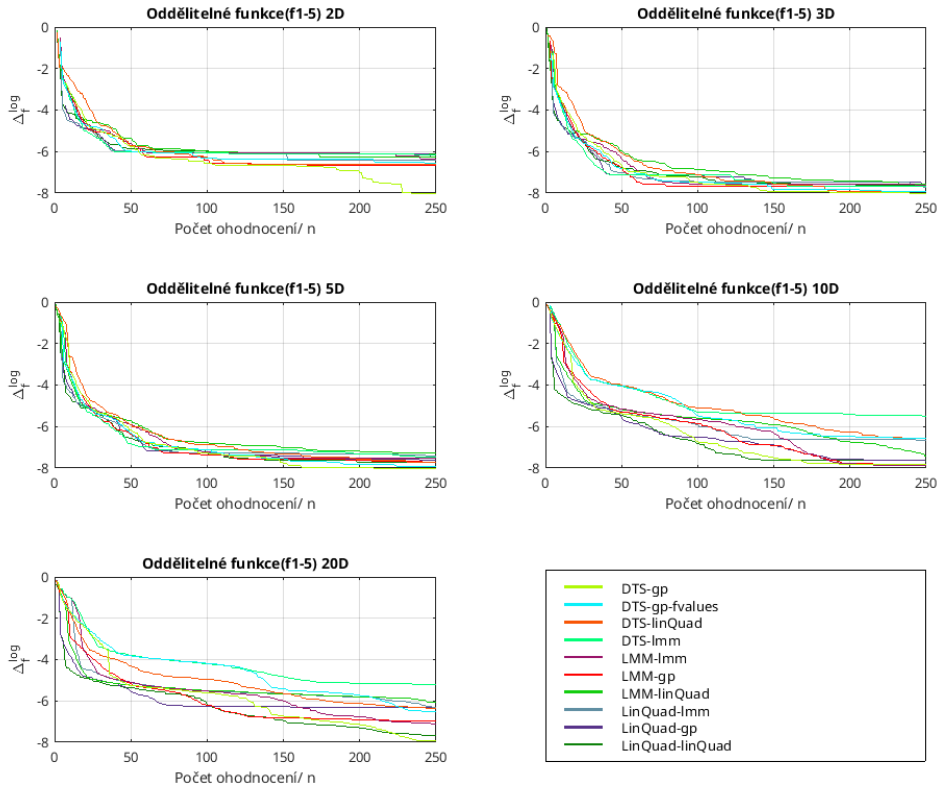
Evoluční kontrole z lmm-CMA-ESu spolu s gaussovskými procesy se dařilo u menších dimenzí ($n <= 5$) a funkcí s velkým šumem.

Multimodální funkce s velkým šumem v dimenzi 20 zvládla nejlépe kombinace lineárně kvadratické evoluční kontroly s gaussovskými procesy.

Grafy vývoje chyby, tabulku s počtem potřebných evaluací nebo součet pořadí můžeme vidět na obrázcích 3.4, 3.15, 3.22, 3.23 a 3.24.

3.5.3 Funkce založené na reálných problémech

Nejlepších výsledků na nově implementovaných funkcích založených na reálných problémech dosáhly gaussovské procesy. Na malých dimenzích byly součástí nejlepší kombinace u 3 funkcí a u velkých dimenzí dokonce u všech. Důvodem

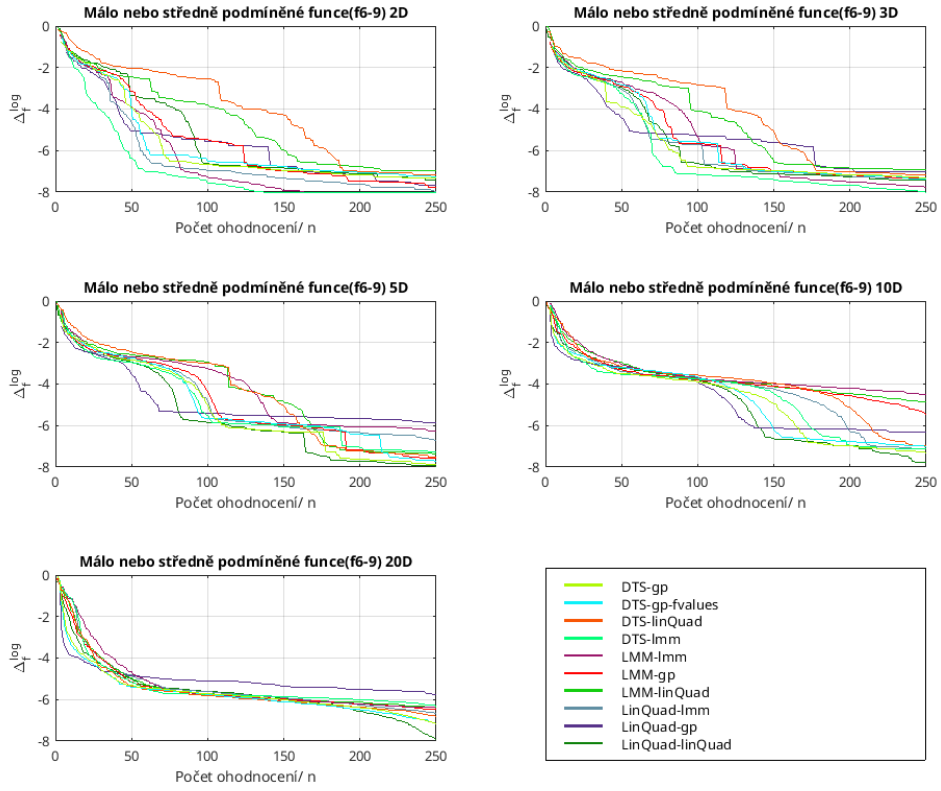


Obrázek 3.16: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro oddělitelné funkce bez šumu.

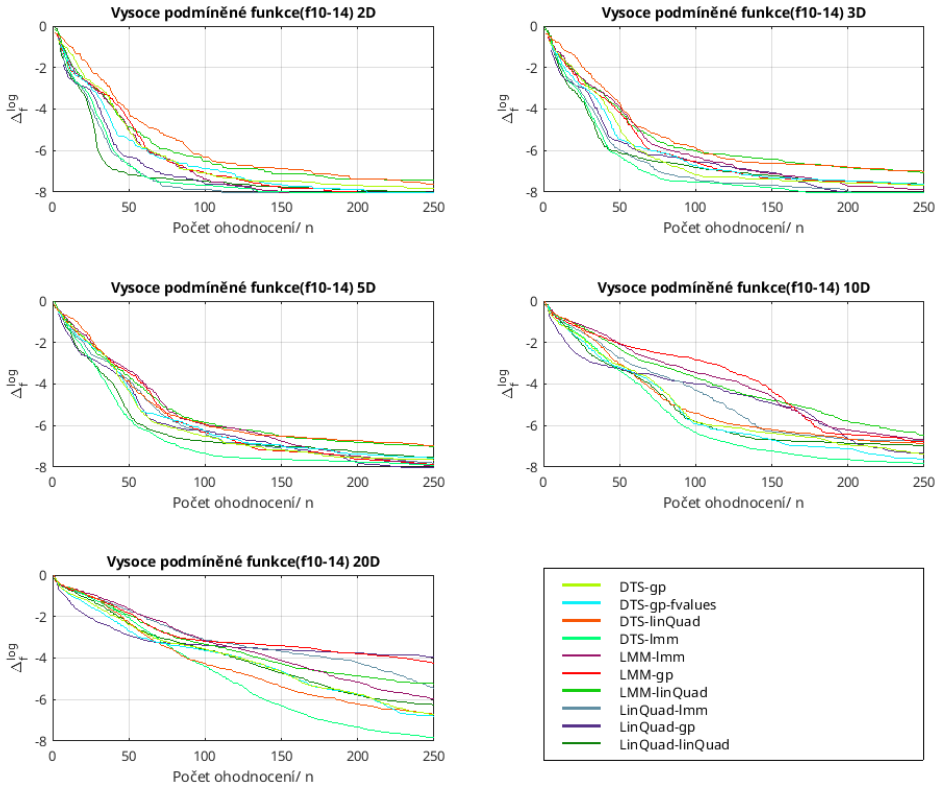
bude nejspíše větší složitost black-box funkcí, kterou nebyly ostatní modely schopny zachytit.

Za zmíňku také stojí originální lq-CMA-ES, který dokázal nalézt optimum u nízkých dimenzí f_{201}, f_{202}, f_{203} a f_{204} téměř ihned v prvních generacích.

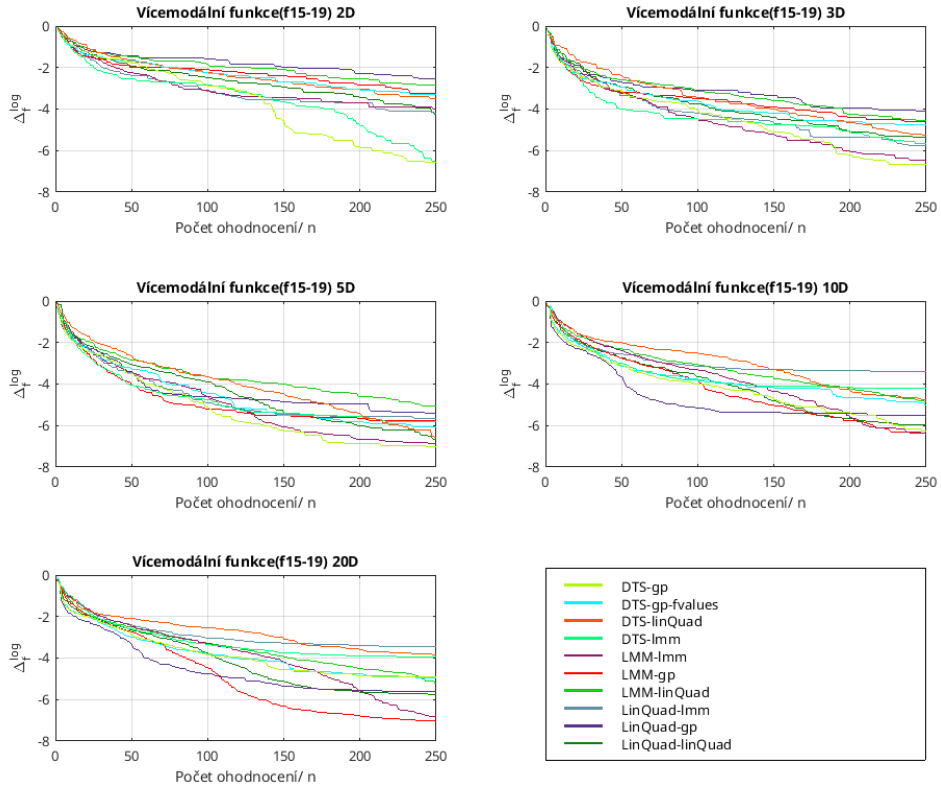
Průběhy algoritmů jsou vidět na obrázcích 3.25 a 3.26.



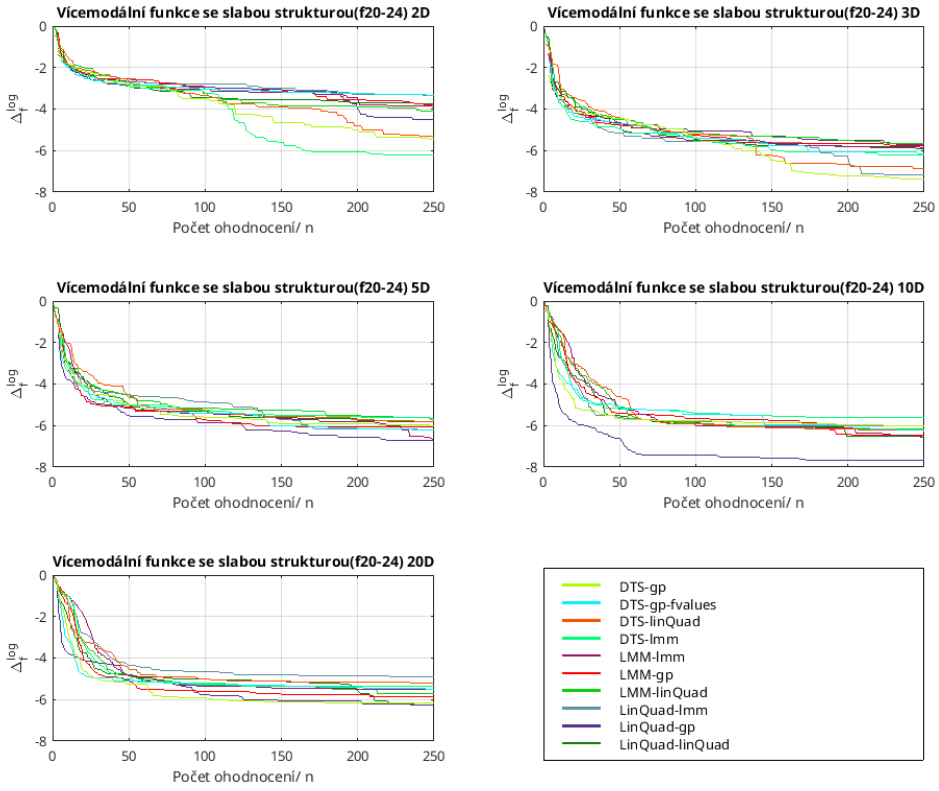
Obrázek 3.17: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro málo nebo středně podmíněné funkce bez šumu.



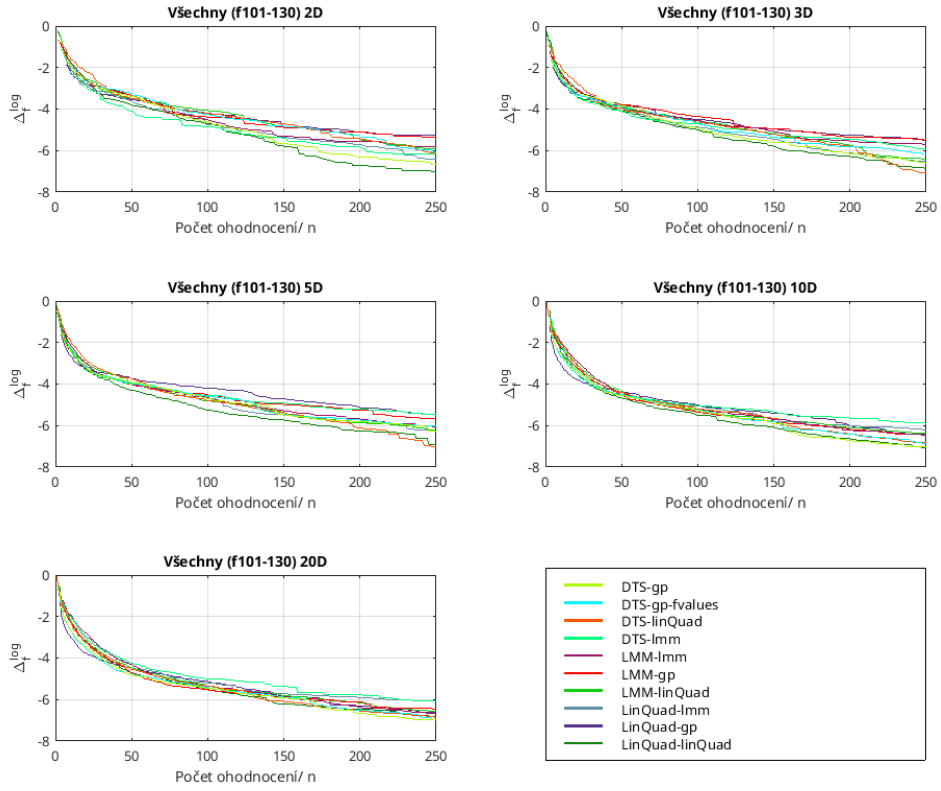
Obrázek 3.18: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro vysoce podmíněné a unimodální funkce bez šumu.



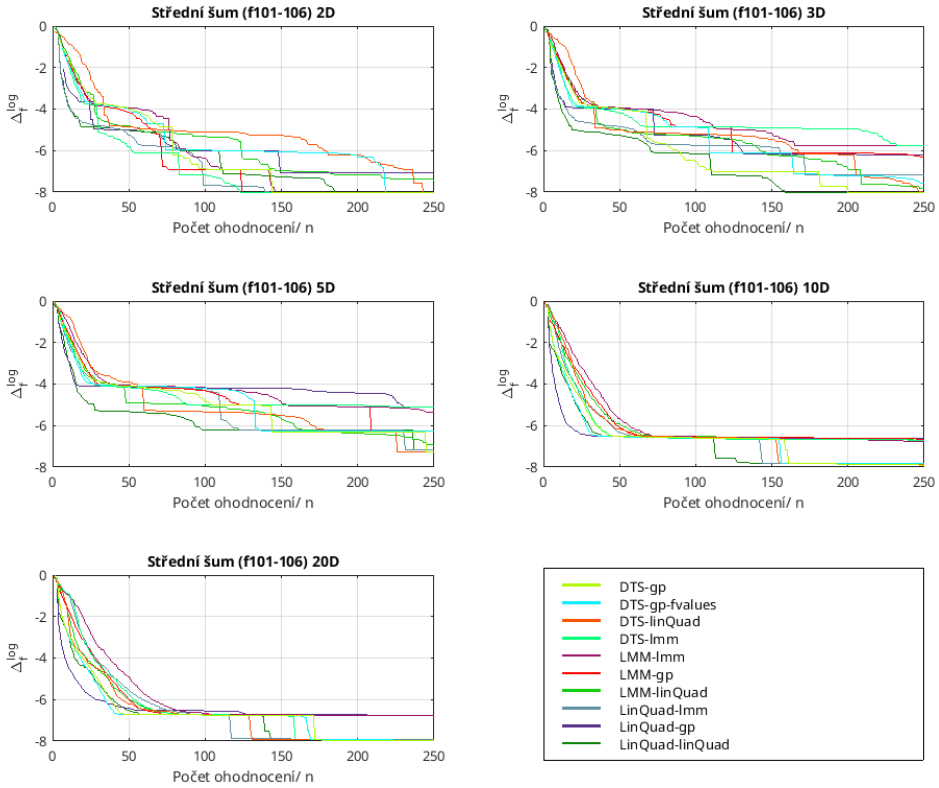
Obrázek 3.19: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima vícemodálních funkcí bez šumu s odpovídající globální strukturou.



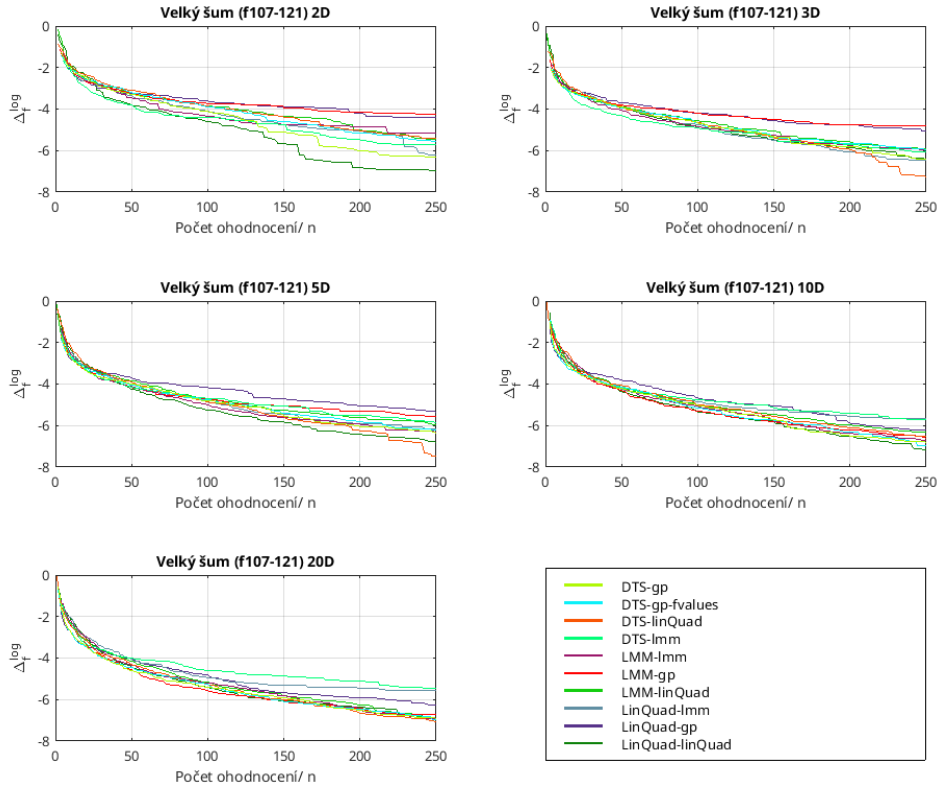
Obrázek 3.20: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima multimodálních funkcí bez šumu se slabou globální strukturou.



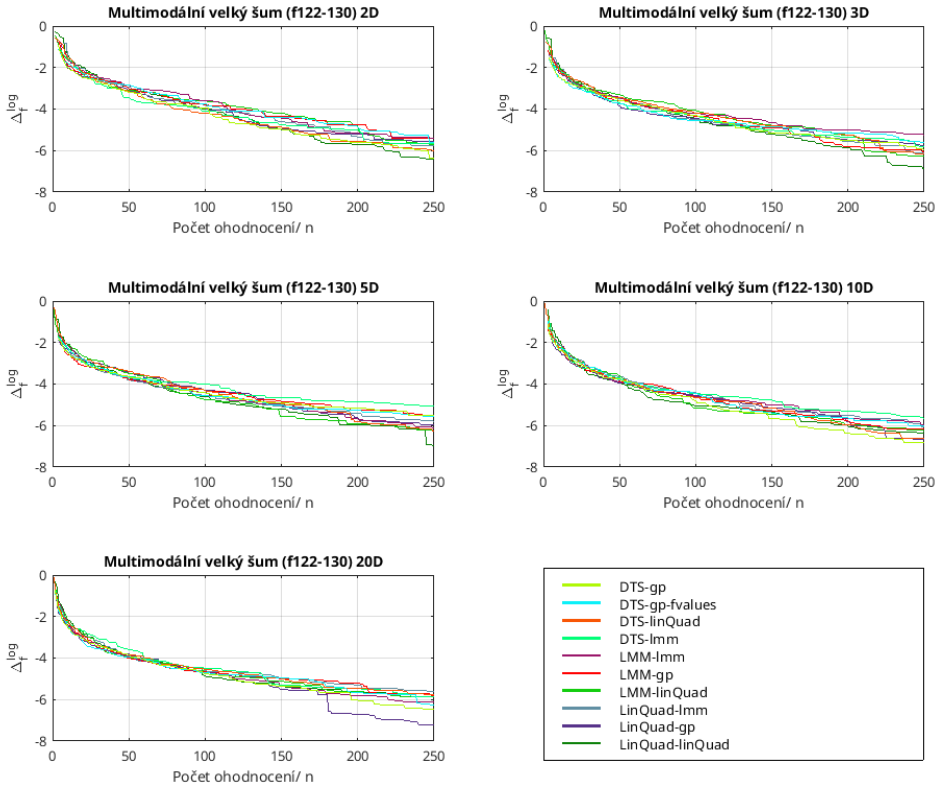
Obrázek 3.21: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro funkce se šumem.



Obrázek 3.22: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro funkce se středním šumem.



Obrázek 3.23: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro funkce s velkým šumem.

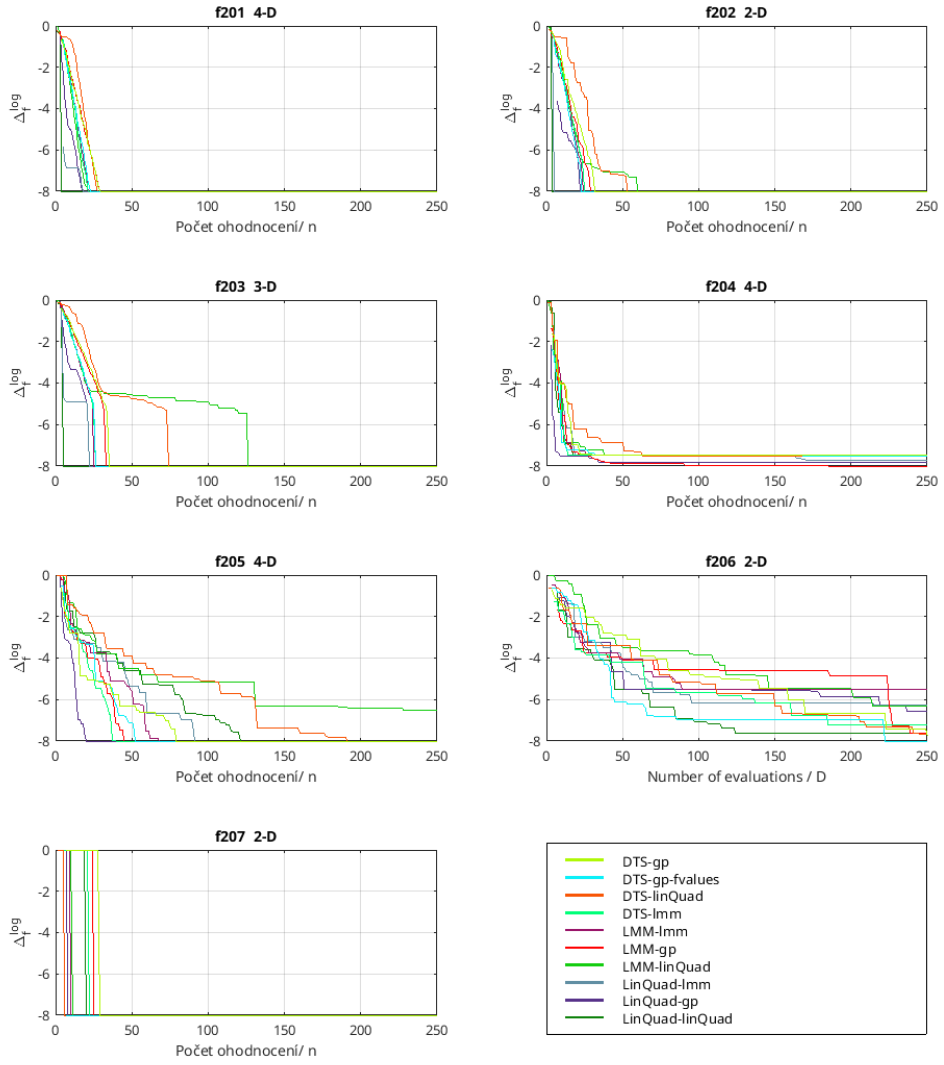


Obrázek 3.24: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro multimodální funkce s velkým šumem.

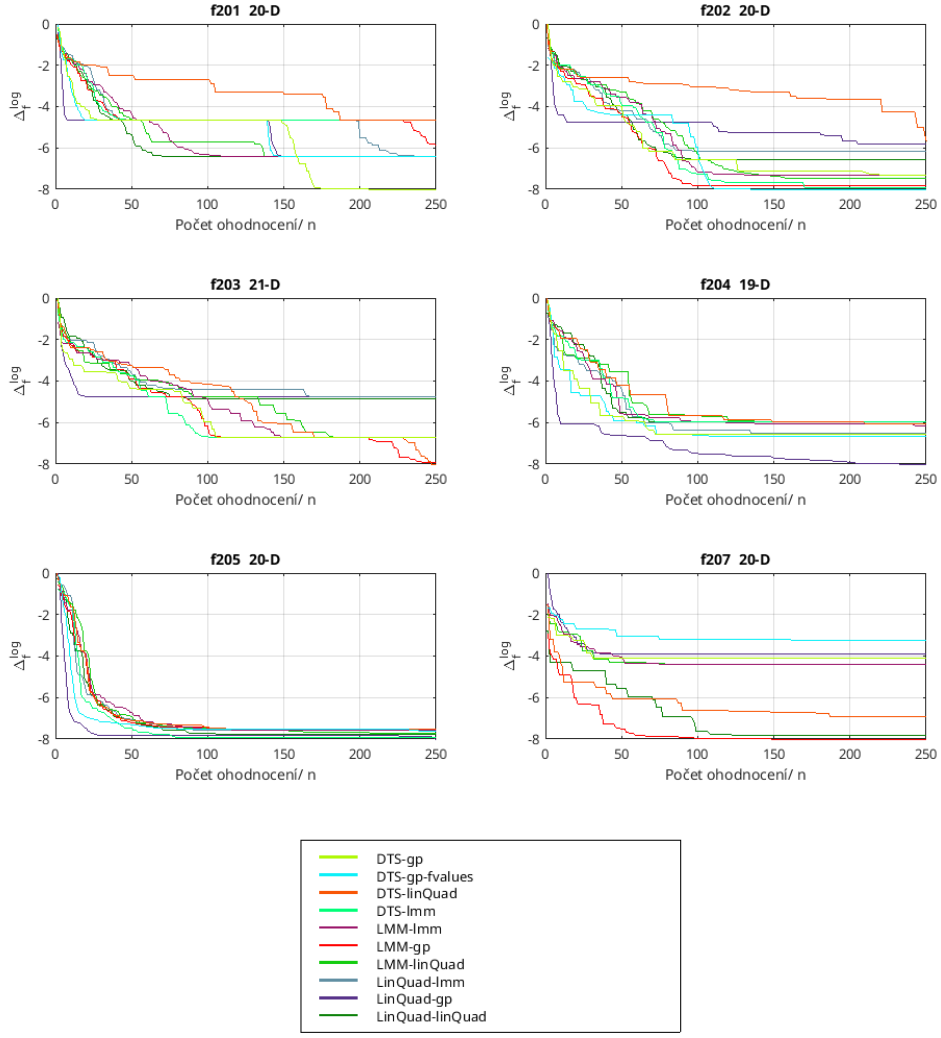
Tabulka 3.4: Součet prvních míst algoritu pro různé poměry počtu evaluace funkcí / dimenze (FE/D) a dimenze pro funkce se šumem. V případě, že prvního místa dosáhne více algoritmů, tak je první místo započteno všem. Většinou je to z důvodu nalezení globálního optima.

	2D				3D				5D			
FE/D	25	50	100	200	25	50	100	200	25	50	100	200
DTS-gp	168	158	**137**	**136**	178.5	153	147.5	151.5	158.5	167	165	151.5
DTS-gp-fvalues	164	188	172	165	140.5	148	165.5	170.5	137.5	158	144	167.5
DTS-linQuad	242	207.5	164.5	163	225	195.5	190	144	240.5	224	183.5	133.5
DTS-lmm	**96**	**100.5**	140.5	160	**90.5**	**116**	142.5	182.5	**104.5**	133	188.5	206
LMM-lmm	159	160	170.5	176	177	165	164	181.5	164	147	151	162
LMM-gp	143	169	206.5	221.5	181	183	207.5	221	149	136	188	188
LMM-linQuad	208	190	199.5	184	199	211	188	167	201.5	214	166	179.5
LinQuad-lmm	149	144	153.5	**136**	166.5	157	**135.5**	120.5	180.5	162	151	143.5
LinQuad-gp	135	177.5	191	201	144.5	166	181	214	157.5	190	221	216
LinQuad-linQuad	186	167.5	147	139.5	159.5	163.5	140.5	**117.5**	164.5	**119**	**108**	**114.5**

	10D				20D				$\sum$			
FE/D	25	50	100	200	25	50	100	200	25	50	100	200
DTS-gp	**129**	152.5	156	132	116	**120.5**	129.5	**122.5**	750	751	735	693.5
DTS-gp-fvalues	130	160.5	157	155	**106**	122.5	141.5	145.5	678	777	780	803.5
DTS-linQuad	205	189.5	166	151	150	172	154.5	139.5	1062.5	988.5	858.5	731
DTS-lmm	149	154.5	190	207	190	207	213.5	223.5	**630**	711	875	979
LMM-lmm	219	171.5	186	141	214	200	186.5	137	933	843.5	858	797.5
LMM-gp	164	158.5	152	173	158	129	**117.5**	168	795	775.5	871.5	971.5
LMM-linQuad	210	203.5	146	170	172	185	164.5	162	990.5	1003.5	864	862.5
LinQuad-lmm	178	173.5	186	198	238	200	200.5	221.5	912	836.5	826.5	819.5
LinQuad-gp	139.5	185.5	211	202	138	183	205.5	199	714.5	902	1009.5	1032
LinQuad-linQuad	134.5	**100.5**	**100**	**121**	168	139	136.5	143.5	812.5	**689.5**	**632**	**636**



Obrázek 3.25: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro nově přidané funkce založené na reálných problémech testované na malých dimenzí. Dimenze zobrazených grafů jsou rozdílné, protože nově přidané funkce jsou definované pouze v předem daných dimenzích.



Obrázek 3.26: Zprůměrované grafy vývoje škálovaného logaritmu vzdálenosti od optima pro nově přidané funkce založené na reálných problémech testované na větších dimenzí. Dimenze zobrazených grafů jsou rozdílné, protože nově přidané funkce jsou definované pouze v předem daných dimenzích. Graf funkce f_{206} není přítomen, protože funkce je definovaná pouze ve 2D.

Závěr

Tato práce si kladla za cíl zjistit, jakou mírou přispívají jednotlivé algoritmy řízení evoluce a náhradní modely, založené na response surface modelech, třech vybraných variant CMA-ESu, k uspoření potřebného počtu ohodnocení minimalizovanou funkcí. Vycházel jsem z teorie týkající se optimalizace black-box funkcí, evolučních strategií, algoritmu CMA-ES i všech tří zkoumaných variant - lmm-CMA-ES, lq-CMA-ES a DTS-CMA-ES. Každá z variant byla implementována a rozdělena na dvě části: evoluční kontrolu a náhradní model. Všechny permutace kombinací evoluční kontroly s náhradním modelem byly otestovány na testovacím frameworku BBOB, který jsem rozšířil o funkce z praxe.

Každá varianta CMA-ESu obohacuje algoritmus o nějakou klíčovou myšlenku. Lokální metamodel buduje unikátní náhradní model pro každého jedince v generaci a trénuje ho pouze pomocí nejbližších již dříve ohodnocených bodů. Lineárně kvadratický model zase mění svojí složitost v závislosti na počtu ohodnocených bodů a využívá souřadnice minimální funkční hodnoty náhradního modelu k vygenerováního jednoho z jedinců příští generace. Dvojitě trénované řízení evoluce je dynamické díky kritériím nejistoty a dokáže využívat i rozptyl náhradního modelu.

Ve výsledcích testování můžeme pozorovat, že žádná z kombinací řízení evoluce a náhradního modelu není ve všem nejlepší. Zatímco pro funkce bez šumu se jako nejlepší varianta jeví originální DTS s gaussovskými procesy, tak funkce se šumem ovládl lq-CMA-ES. Kombinace dvojitě trénovaného řízení evoluce a lokálního metamodelu zase excelovala ze začátku optimalizace, ale nebyla schopná v této efektivnosti pokračovat.

U všech algoritmů platí, že nejlepších výsledků dosáhly v kombinaci s náhradním modelem, který byl publikován v práci představující celý algoritmus.

Jednou z možností jak pokračovat v rozvíjení variant CMA-ESu je sloučení všech myšlenek do jednoho algoritmu. Náhradní model nemusí měnit pouze stupeň použitého polynomu, ale celý model by mohl by být nahrazen jiným

typem. Podle provedených experimentů je lokální metamodel velmi úspěšný v malých dimenzích, ale větší dimenze již nezvládne namodelovat. Gaussovské procesy zase excelují ve velkých dimenzích. Výměna celého modelu by tedy mohla minimalizovat nevýhody, které každý z nich má. Stavba vlastního náhradního modelu pro každý ohodnocený bod zase snižuje velikost prostoru, kterou se náhradní model snaží replikovat, a dá se využít u všech náhradních modelů. To samé platí pro využívání souřadnic s minimální funkční hodnotou. Bylo by zajímavé prozkoumat variantu, která toto minimum využije pouze v případě, že se náhradní model dokázal správně naučit a nebyli ohodnoceni všichni jedinci generace.

Dále by bylo dobré všechny varianty CMA-ESu porovnávat vždy pouze s těmi, které využívají stejnou verzi algoritmu CMA-ES. CMA-ES je neustále vylepšován a rychlejší optimalizace s využitím nejnovější verze ještě nemusí být způsobena novou evoluční kontrolou nebo náhradním modelem. Proto by se v dalších studiích měli autoři zaměřit na využití aktuální verze CMA-ESu.

Literatura

[1] Commons, W.: File:Concept of directional optimization in CMA-ES algorithm.png. 2008. Dostupné z: https://en.wikipedia.org/wiki/File:Concept_of_directional_optimization_in_CMA-ES_algorithm.png

[2] Hansen, N.: The CMA Evolution Strategy: A Tutorial. 2016, 1604.00772.

[3] Myers, R.; Montgomery, D.; Anderson-Cook, C.: *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley Series in Probability and Statistics, Wiley, 2009, ISBN 9780470174463.

[4] Knagg, O.: An intuitive guide to Gaussian processes. 2019. Dostupné z: https://towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d

[5] Pitra, Z.: *Rozšíření systému evoluční optimalizace GENACAT o náhradní modelování pomocí gaussovských procesů*. Diplomová práce, České vysoké učení technické v Praze Fakulta jaderná a fyzikálně inženýrská, 2014.

[6] Cassioli, A.: A Tutorial on Black–Box Optimization. 2013. Dostupné z: https://www.lix.polytechnique.fr/~dambrosio/blackbox_material/Cassioli_1.pdf

[7] Auger, A.; Brockhoff, D.; Hansen, N.; aj.: Benchmarking the Pure Random Search on the Bi-Objective BBOB-2016 Testbed. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, New York, NY, USA: Association for Computing Machinery, 2016, ISBN 9781450343237, str. 1217–1223, doi:10.1145/2908961.2931704. Dostupné z: https://doi.org/10.1145/2908961.2931704

[8] Sawyerr, B.; Adewumi, A.; Ali, M.: Benchmarking projection-based real coded genetic algorithm on BBOB-2013 noiseless function testbed. 07 2013, doi:10.1145/2464576.2482698.

[9] Holtschulte, N. J.; Moses, M.: Benchmarking Cellular Genetic Algorithms on the BBOB Noiseless Testbed. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, New York, NY, USA: Association for Computing Machinery, 2013, ISBN 9781450319645, str. 1201–1208, doi:10.1145/2464576.2482699. Dostupné z: https://doi.org/10.1145/2464576.2482699

[10] Tang, Y.; Chen, J.; Wei, J.: A surrogate-based particle swarm optimization algorithm for solving optimization problems with expensive black box functions. *Engineering Optimization*, ročník 45, 05 2013: s. 557–576, doi:10.1080/0305215X.2012.690759.

[11] Speranskii, D. V.: Ant colony optimization algorithms for digital device diagnostics. *Automatic Control and Computer Sciences*, ročník 49, č. 2, Mar 2015: s. 82–87, ISSN 1558-108X, doi:10.3103/S0146411615020078. Dostupné z: https://doi.org/10.3103/S0146411615020078

[12] Li, Z.; Zhang, Q.; Lin, X.; aj.: Fast Covariance Matrix Adaptation for Large-Scale Black-Box Optimization. *IEEE Transactions on Cybernetics*, 2018: s. 1–11, ISSN 2168-2275, doi:10.1109/TCYB.2018.2877641.

[13] Serrano, J. I.; del Castillo, M. D.: On the origin of the evolutionary computation species influences of Darwin's theories on computer science. *Artificial Intelligence Review*, ročník 38, č. 1, Jun 2012: s. 41–54, ISSN 1573-7462, doi:10.1007/s10462-011-9246-6. Dostupné z: https://doi.org/10.1007/s10462-011-9246-6

[14] Eiben, A. E.; Smith, J. E.: *Introduction to Evolutionary Computing*. Springer Publishing Company, Incorporated, druhé vydání, 2015, ISBN 3662448734.

[15] Loshchilov, I.: *Surrogate-Assisted Evolutionary Algorithms*. Theses, Université Paris Sud - Paris XI ; Institut national de recherche en informatique et en automatique - INRIA, Leden 2013. Dostupné z: https://tel.archives-ouvertes.fr/tel-00823882

[16] Loshchilov, I.; Schoenauer, M.; Sebag, M.: Self-Adaptive Surrogate-Assisted Covariance Matrix Adaptation Evolution Strategy. *GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation*, 04 2012, doi:10.1145/2330163.2330210.

[17] Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, ročník 9, 10 2005: s. 3–12, doi:10.1007/s00500-003-0328-5.

[18] Emmerich, M.; Giotis, A.; Özdemir, M.; aj.: Metamodel—Assisted Evolution Strategies. 09 2002, doi:10.1007/3-540-45712-7_35.

[19] Pitra, Z.; Bajer, L.; Repicky, J.; aj.: Overview of surrogate-model versions of covariance matrix adaptation evolution strategy. 07 2017, s. 1622–1629, doi:10.1145/3067695.3082539.

[20] Kern, S.; Hansen, N.; Koumoutsakos, P.: Local Meta-models for Optimization Using Evolution Strategies. 01 2006, s. 939–948, doi:10.1007/11844297_95.

[21] Auger, A.; Brockhoff, D.; Hansen, N.: Benchmarking the Local Metamodel CMA-ES on the Noiseless BBOB'2013 Test Bed. In *GECCO (Companion), workshop on Black-Box Optimization Benchmarking (BBOB'2013)*, Amsterdam, Netherlands, Červenec 2013. Dostupné z: `https://hal.inria.fr/hal-00825840`

[22] Pitra, Z.; Bajer, L.; Holeňa, M.: Doubly Trained Evolution Control for the Surrogate CMA-ES. In *Parallel Problem Solving from Nature – PPSN XIV*, editace J. Handl; E. Hart; P. R. Lewis; M. López-Ibáñez; G. Ochoa; B. Paechter, Cham: Springer International Publishing, 2016, ISBN 978-3-319-45823-6, s. 59–68.

[23] Bajer, L.; Pitra, Z.; Holena, M.: Benchmarking Gaussian Processes and Random Forests Surrogate Models on the BBOB Noiseless Testbed. 07 2015, doi:10.1145/2739482.2768468.

[24] Hansen, N.: A Global Surrogate Assisted CMA-ES. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, New York, NY, USA: Association for Computing Machinery, 2019, ISBN 9781450361118, str. 664–672, doi:10.1145/3321707.3321842. Dostupné z: `https://doi.org/10.1145/3321707.3321842`

[25] ASTER Global Digital Elevation Map. Dostupné z: `https://asterweb.jpl.nasa.gov/gdem.asp`

[26] COmparing Continuous Optimisers: COCO. Dostupné z: `https://coco.gforge.inria.fr/`

[27] Gallagher, M.; Preuss, M.; Kerschke, P.: PPSN'18 Machine Learning and Data Analysis (MLDA) Problem Set, v1.0. 2018.

[28] Euclidean Sum of Squares Clustering Problems. 2017. Dostupné z: `http://realopt.uqcloud.net/ess_clustering.html#german`

[29] Renewable Energy. 2016. Dostupné z: https://cs.adelaide.edu.au/~optlog/research/energy.php

Příloha **A**

Seznam použitých zkratek

CMA-ES Covariance Matrix Adaptation Evolution Strategy

lmm Lokální metamodel

LQ Lineárně kvadratický

BBOB Black-Box Optimalization Benchmarking

FE Počet ohodnocení funkce (Function Evaluations)

gp Gaussovské procesy

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF