VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

EVOLUČNÍ NÁVRH ULTRAZVUKOVÝCH OPERAČNÍCH PLÁNŮ

EVOLUTIONARY DESING OF ULTRASOUND OPERATIONAL PLANS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAKUB CHLEBÍK**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. JIŘÍ JAROŠ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Chlebík Jakub, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Evoluční návrh ultrazvukových operačních plánů**
Evolutionary Design of Ultrasound Treatment Plans
Kategorie: Umělá inteligence
Zadání:

1. Seznamte se s principy evolučního návrhu ultrazvukových operačních plánů, zaměřte se na použité fyzikální modely.
2. Prostudujte nejpoužívanější evoluční algoritmy. Zaměřte se na efektivitu s ohledem na velikost populace a počet evaluací fitness funkce.
3. Navrhněte vhodnou množinu testovacích úloh vycházejících z klinické praxe.
4. Navrhněte sadu kritérií pro vyhodnocení efektivity zvolených evolučních algoritmů.
5. Poveďte experimentální porovnání zvolených evolučních algoritmů.
6. Statisticky vyhodnoťte získané výsledky.
7. Diskutujte přínos práce pro klinickou praxi.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 4 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Jaroš Jiří, doc. Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 25. října 2019

Abstrakt

Tato práce se zabývá studiem vybraných evolučních systémů pro jejich použití při návrhu plánu pro ultrazvukové operace. Tyto algoritmy statisticky analyzuje a dle vhodných kritérií je experimentálně srovnává a diskutuje přínos pro klinickou praxi.

Abstract

The thesis studies selected evolution systems to use in planning of high intensity focused ultrasound sonications. Considered algorithms are statistically analysed and compared by appropriate criteria to find the one that adds the most value to the potential real world medical problems.

Klíčová slova

optimalizace, evoluční algoritmy, genetický algoritmus, tabu prohledávání, simulované žíhání, diferenciální evoluce, optimalizace rojem částic, CMA-ES, optimalizace HIFU operací

Keywords

optimization, evolutionary algorithms, genetic algorithm, tabu search, simulated annealing, differential evolution, partile swarm optimization, CMA-ES, HIFU surgery optimization

Citace

CHLEBÍK, Jakub. *Evoluční návrh ultrazvukových operačních plánů.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Jiří Jaroš, Ph.D.

Evoluční návrh ultrazvukových operačních plánů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Doc. Ing. Jiřího Jaroše, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Chlebík
3. června 2020

Poděkování

Tímto bych chtěl poděkovat Doc. Ing. Jiřímu Jarošovi, Ph.D. za pomoc a rady, které mi poskytl při psaní této práce a v průběhu konzultací. Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy z Národního programu udržitelnosti II (NPU II) v rámci projektu IT4Innovations excellence in science - LQ1602. Výsledky byly získány s využitím výzkumné infrastruktury podpořené z programu Velkých infrastruktur pro výzkum, experimentální vývoj a inovace v rámci projektu IT4Innovations národní superpočítačové centrum - LM2015070.

Obsah

Kapitola 1

Úvod

Evoluční algoritmy jsou moderním způsobem řešení náročných optimalizačních problémů, pro které doposud neexistuje přesná, matematickou analýzou podložená, metoda. Jsou součástí tématu nazývaného anglicky „Soft Computing“, od ostatních metod této rodiny se ovšem liší svou tolerancí nepřesností a šumu, schopností pracovat s aproximacemi a jistou náhodností.

Cílem práce je statistickou analýzou nalézt takový evoluční algoritmus, který dle vybraných kritérií nejlépe navrhne plán ulrazvukové operace. V tomto textu bude představeno několik vybraných evolučních algoritmů, otestována jejich schopnost hledat optimum nad vhodnou testovací sadou a diskutována jejich vhodnost pro optimalizaci hledání ultrazvukové trajektorie. Následně jsou tyto algoritmy aplikovány na model šíření tepelné energie a představeny statistické výsledky. Jak již je u optimalizačních problémů složitějších systémů zvykem, i zde platí tzv. „No Free Lunch“ teorém a je tedy třeba podstoupit jisté kompromisy. Ty jsou vyjádřeny formou kritérií, dle kterých jsou algoritmy hodnoceny.

V následující kapitole je představen problém optimalizace plánů systému ultrazvukových operací. Zde budou prezentována kritéria, kterými je třeba se řídit při hodnocení vhodnosti jednotlivých optimalizačních metod. Diplomová práce navazuje na článek [20], jehož stěžejní body budou v následující kapitole představeny. V kapitole 3 budou detailněji popsány jednotlivé zkoumané algoritmy - principy jejich fungování, motivace k jejich vzniku a kladné i stinné stránky použití. Jedná se o optimalizační metody *genetického algoritmu*,*simulovaného žíhání*, *metoda tabu prohledávání*, *optimalizace rojem částic*, *diferenciální evoluce* a *evoluční strategie založená na adaptaci kovarianční matice*. Kapitola 4 představí implementaci jednotlivých algoritmů, jejich rozhraní a výstupní protokol. Následně je představeno několik typických problémů pro hodnocení optimalizačních metod a statistickým způsobem ukázáno, jak dobře byly schopny zmíněné vybrané algoritmy tyto optimalizační testy řešit. V závěru kapitoly je provedeno zběžné zhodnocení implementovaných optimalizačních metod pro řešení problémů v testovací sadě a představena kritéria hodnocení těchto metod v kontextu optimalizace trajektorie HIFU operací. Kapitola 5 se bude věnovat samotné simulaci šíření tepla. Představí testovací sadu několika modelů cílené tkáně pro ablaci, které jsou založeny na získaných poznatcích o problému optimalizace trajektorie. Následně jsou provedeny experimenty a výsledky statisticky prezentovány a zhodnoceny. V závěru je poté vybráno několik vhodných algoritmů pro další studium.

Kapitola 2

Optimalizovaný systém

Následující kapitola si klade za cíl seznámit čtenáře se systémem HIFU - Fokusovaný ultrazvuk o vysoké intenzitě. Se svolením autora je téměř celý obsah této kapitoly, včetně obrázků 2.5, 2.6 a zdrojů [10], [17], [19] převzat, přeložen a nebo parafrázován ze článku `Design of HIFU Treatment Plans using an Evolutionary Strategy` autorů *Jiří Jaroš, Marta Jaroš* a *Bradley E. Treeby* [20]. Toto dílo se má práce snaží rozšířit a je tedy vhodné jej zde i představit. Nejdříve je ovšem třeba udělat krok zpět a představit pojmy optimalizace a operační výzkum - této problematice jsem se již věnoval ve své bakalářské práce [5] a proto zde i uvádím původní část tohoto textu.

2.1 Matematická optimalizace

Optimalizace je podoborem matematické analýzy a numerické matematiky, zabývající se hledáním extrému zobrazení $f : \mathbb{R}^n \to \mathbb{R}$ v prostoru $\Omega = \mathbb{R}^n$. Přesněji, vyhledává množinu vstupních parametrů $x = x_1, ..., x_n$ problému, které odpovídají minimu, případně maximu, hodnotící funkce $f(x) = f(x_1, ..., x_n)$.

Definice 1. Minimum funkce.
Nechť $f : \mathbb{R}^n \to \mathbb{R}$ je funkce n proměnných. Tvrdíme, že $x^* \in \mathbb{R}$ je minimem f (dále v textu pouze "minimum"), pokud existuje $\delta > 0$ taková, že pro všechna x ve vztahu $\|x - x^*\| \leq \delta$ platí $f(x^*) \leq f(x)$. Analogicky lze odvodit definici maxima.

O takovémto minimu tvrdíme, že je lokálním, pokud je nejmenším bodem v daném okolí. O globálním minimu hovoříme, pokud je takovýto bod nejmenším na celém definičním oboru. Znázorněno na obrázku 2.1.

Díky vztahu

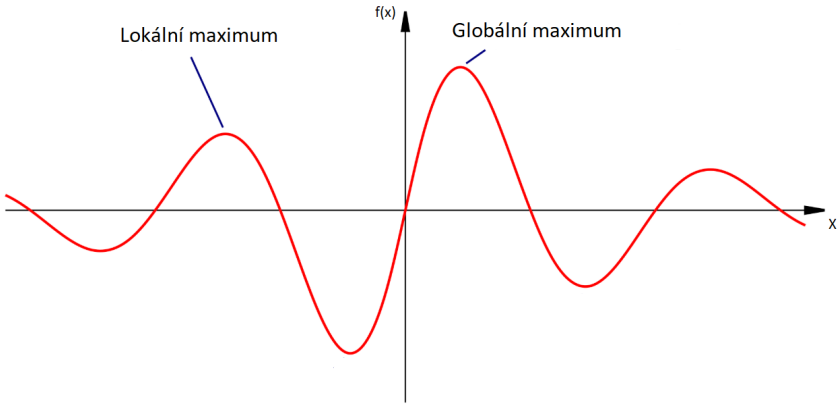
$$\max_{x \in \Omega}(f(x)) = -\min_{x \in \Omega}(-f(x)) \tag{2.1}$$

můžeme vždy předpokládat, že optimalizační problém je problémem minimalizačním a maximalizací se dále nemusíme zabývat.

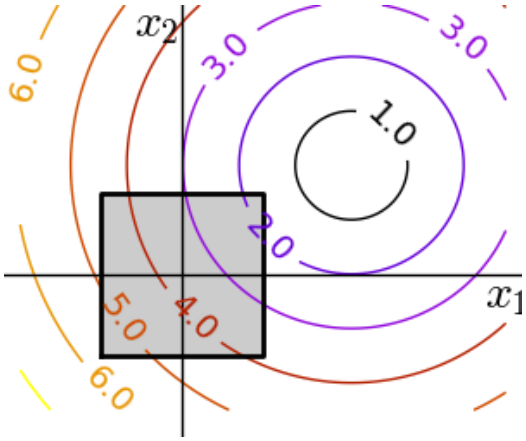
2.1.1 Klasifikace optimalizačních úloh

Dle definičního oboru prostoru Ω účelové funkce se optimalizační úlohy klasifikují na:

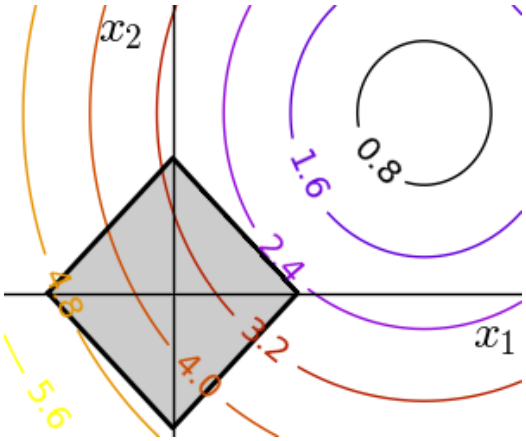
- Úlohy volného extrému. Tento případ nastává, pokud $\Omega = \mathbb{R}^n$.



Obrázek 2.1: Ukázka globálního a lokálního minima na jednoduché matematické funkci. Zdrojem je internet.



Obrázek 2.2: Omezení intervalem.



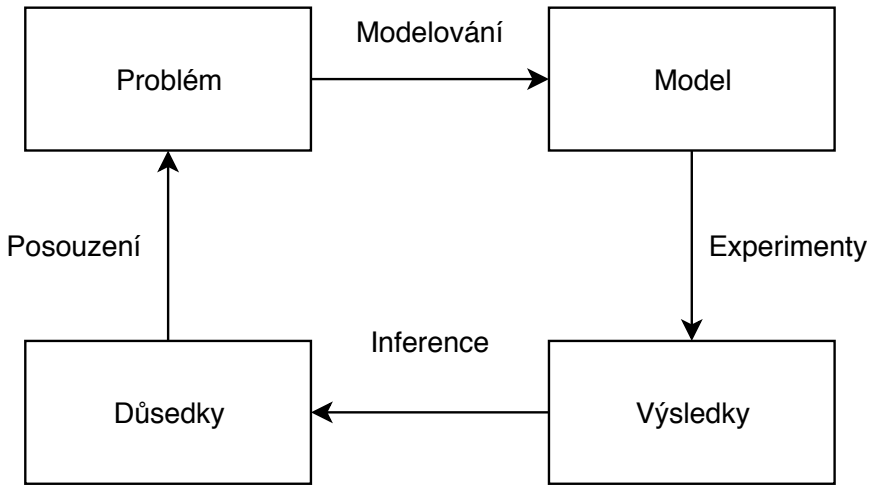
Obrázek 2.3: Omezení funkcí.

- Úlohy vázaného extrému - případ kdy $\Omega \subset \mathbb{R}^n$. Zde nás zajímají pouze řešení, která splňují další podmínky, omezující buďto jeden nebo více vstupních parametrů $x_0, ...x_n$ na interval, nebo funkcemi $g_0(x_i, ...x_n) \geq 0$. O takto definovaných úlohách se mluví jako o úlohách matematického programování. Jednoduché příklady omezení vázaného extrému jsou znázorněna na obrázcích 2.2 a 2.3.
- Funkce jednoho extrému - tzv. „unimodální“.
- Funkce více extrémů stejné váhy - tzv. „multimodální“.

2.1.2 Operační výzkum

Pojmem *operační výzkum* (často označován také jako operační analýza) se všeobecně rozumí převod komplexní inženýrské úlohy reálného světa na matematický, či jiný, model a následné provádění experimentů pro získání informací. Problémy tohoto typu jsou často spojeny s hledáním minima (např. provozní riziko), maxima (např. zisk), či jiného optimálního výsledku. Teoretickým základem operačního výzkumu je tedy matematická optimalizace a ke hledání řešení se využívá metody optimalizací, modelování a simulací [3].

Průběh operačního výzkumu je znázorněn na obrázku 2.4.



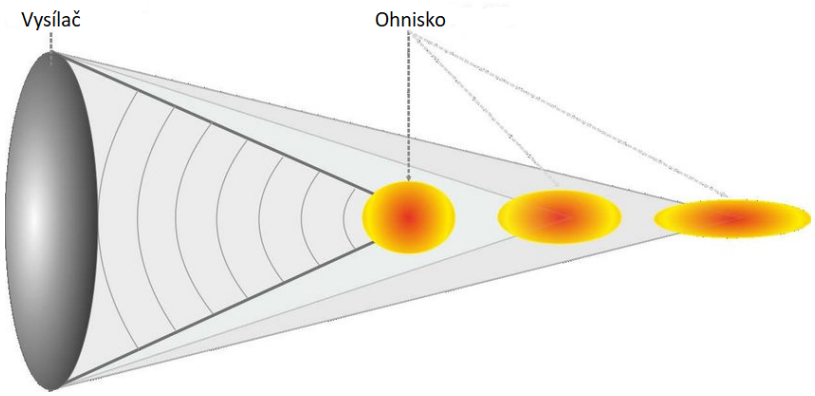
Obrázek 2.4: Zjednodušený náhled na proces operačního výzkumu

2.2 Fokusovaný ultrazvuk o vysoké intenzitě

Fokusovaný ultrazvuk o vysoké intenzitě je moderní technika pro ne-invazivní operace nádorových onemocnění, při které je maligní tkáň odstraněna kumulovanou tepelnou energií vyzářenou ultrazvukovými vysílači. Tato takzvaná termální ablace je provedena pod dohledem profesionála a úkolem je zvýšit teplotu v cíleném místě o několik desítek stupňů, čímž dojde ke zničení tkáně. Na rozdíl od standardních postupů léčby rakoviny se jedná o ne-invazivní a ne-ionizující řešení, které již bylo aplikováno ve více než 100,000 případech.

Jedno toto ultrazvukové ozáření - tzv. sonikace - dokáže odstranit pouze malou oblast cílené tkáně. V závislosti na velikosti oblasti ošetřované části může být potřeba absolvovat sérii těchto sonikací (běžně v řádu desítek). Každá takováto sonikace může způsobit různá další zranění - od popálenin kůže až po možné vážné poškození tkáně poblíž fokálního místa a nebo někde mezi vysílačem a nádorem.

Hlavním problémem tohoto přístupu je tedy toto: jak rozmístit sérii ohnisek vysílačů v prostoru tak, aby se minimalizovalo riziko zranění i počet sonikací a zároveň byla zničena všechna nežádoucí tkáň. Navíc je třeba dát pozor na různé komplikace, jako například krevní řečiště sousedící s cíleným tumorem, které může odvést velkou část indukované tepelné energie, nebo může být zářením poškozeno [10]. Série ohnisek sonikací poté vytváří trajektorii. Toto je optimalizační, přesněji minimalizační, problém, který svou povahou připomíná problém batohu - snažíme se co nejmenší vahou (zatížením těla sonikacemi) zaplnit co největší prostor (odstranit celou nežádoucí tkáň). Bohužel, běžné matematicko-fyzikální rovnice pro šíření zvukových vln a tepla spolu s metodami optimalizace v tak komplikovaném a heterogenním prostředí, jako může být i lidský mozek, a pro tak individuální systém selhávají, a je třeba přistoupit k metodám operační analýzy a soft comuputingu - heuristiky a nebo algoritmy umělé inteligence. Tento přístup obětuje exaktní přesnost za rychlejší proces s možností opakované, parametrizovatelné simulace spolu se zpětnou verifikací.



Obrázek 2.5: Vizualizace regionů, ve kterém se indukuje teplo v závislosti na vzdálenosti ohniska od vysílače. Převzato z [20].

2.3 Rozšiřované řešení

Jak již bylo nastíněno, k řešení tohoto problému je třeba vytvořit model pro šíření tepla ve tkáních. Takovýto model byl již implementován v nástroji k-Wave [17] a využit v řešení [20].

Model

Průběh simulace modelu se skládá z několika stádií. V následujícím výpisu pouze nastíním jejich funkci, pro detailnější popis včetně referencí doporučuji již zmíněnou zdrojovou práci [20]:

- *Výpočet přidané tepelné energie ve tkáni* - v závislosti na délce sonikace, pozici, velikosti a tvaru ohniska (znázorněno na obrázku 2.5) je třeba zjistit, kolik tepelné energie bude vydáno zářením. Přesné výpočty jsou možné například za použití modelu [19]. Bohužel, tyto modely jsou příliš výpočetně náročné pro efektivní použití optimalizační metody, především pak pro použitou evoluční metodu. Proto bylo zavedeno několik zjednodušení a předpokladů. Tyto alternace jsou řádně ocitovány v již zmíněném článku [20].

- *Šíření a rozptyl tepla ve tkáni* - dále je třeba zjistit, jak se všechno přidané teplo rozšířilo ve tkáních. Jedná se o model Penneho biotepelné rovnice šíření, jejíž simulace je prováděna v nástroji k-Wave [17]. Výstupem tohoto modelu je prostorová termální mapa (tzv. CEM_{43} - Cumulative Equivalent Minutes at 43°) kumulovaného tepla za sérii sonikací.

- *Přesnost řešení* - tato tepelná mapa je v posledním kroku převedena na binární masku podle hranice. Následně je spočten integrál nad celou touto mapou, sčítající hodnoty oblastí, které byly ošetřeny a neměly být, spolu s hodnotami oblastí, které neměly být ošetřeny a byly. Dále byla zavedena nevýznamná oblast - oblast ve které nás možná abraze nezajímá. Názorně zobrazeno na obrázku 2.6.

Plánování trajektorie

K návrhu a optimalizaci plánu byl použit algoritmus „Covariance Matrix Adaptation - Evolutionary Strategy“ - zkráceně CMA-ES. Jedná se o moderní variantu evoluční strategie,

pro nelineární nekonvexní systém černé skříňky definované na spojité veličině. Nevyžaduje náročné ladění parametrů - naopak výběr strategie vnitřních parametrů si algoritmus zvolí a postupně vylepšuje sám. Více se tímto algoritmem zabývá sekce 3.6.

Kódování řešení

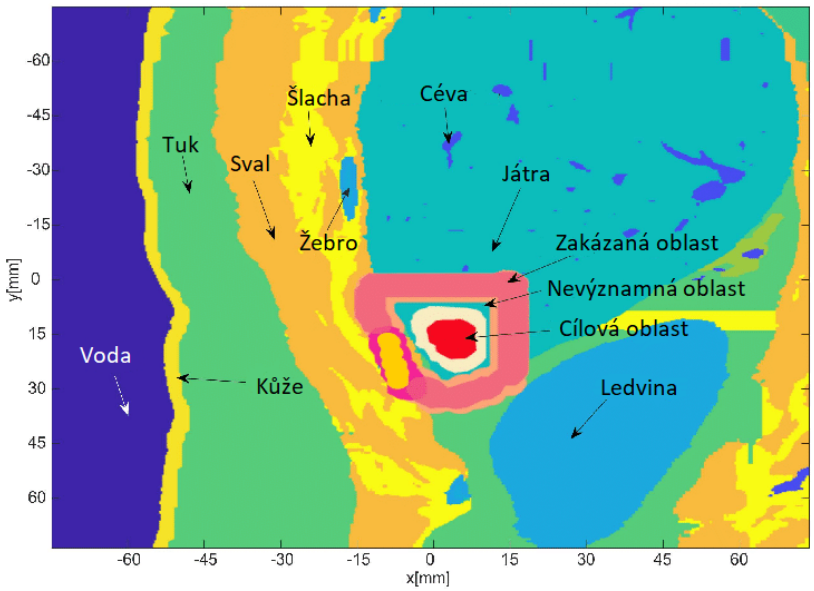
Dále bylo také již navrženo kódování hledaných parametrů do chromozomu. Kandidátní řešení I je poté trajektorie sonikací, kde i odpovídá pořadí sonikací [20]:

$$I = (S_1, S_2, ..., S_N) \tag{2.2}$$

kde

$$S_i = (x(i), y(i), t_{on}(i), t_{off}(i)) \tag{2.3}$$

- S_i - parametry sonikace i
- $x(i), y(i)$ - 2D souřadnice ohniska sonikace i
- $t_{on}(i)$ - délka sonikace i
- $t_{off}(i)$ - interval ochlazení po dokončení sonikace i



Obrázek 2.6: Segmentová mapa AustinWoman s vyznačenou cílovou, zakázanou a nevýznamnou oblastí. Převzato z [20].

Kapitola 3

Evoluční algoritmy

Evoluční algoritmy (zkráceně EA) patří do kategorie přírodou inspirovaných algoritmů. Jedna se o zastřešující množinu pro různé přístupy, které se inspirují v evolučním procesu - pokud je populace jedinců vystavena dlouhodobě selekčnímu tlaku, začíná se v následujících generací na tento tlak lépe adaptovat, aby druh přežil. Takto adaptovaní jedinci dále šíří své geny a neustále vylepšují zdatnost následujících populací. Z technického pohledu je množina jedinců (generace) iterativně vystavována evolučním operacím (v genetickém algoritmu jsou to operace selekce nejlepších jedinců, jejich křížení a případně mutace potomků) pro vytvoření generace nové. Zda-li se původní jedinci budou či nebudou nacházet v nové populaci již záleží na implementaci samotné.

Z matematického hlediska se řadí mezi stochastické metody prohledávání stavového prostoru a *metaheuristiky* - strategie, či procedura na nějaké vyšší úrovni, jejíž cílem je nalézt proceduru nebo heuristiku o úroveň níže, která bude schopná řešit zadaný problém bez bližších údajů o optimalizovaném systému [21].

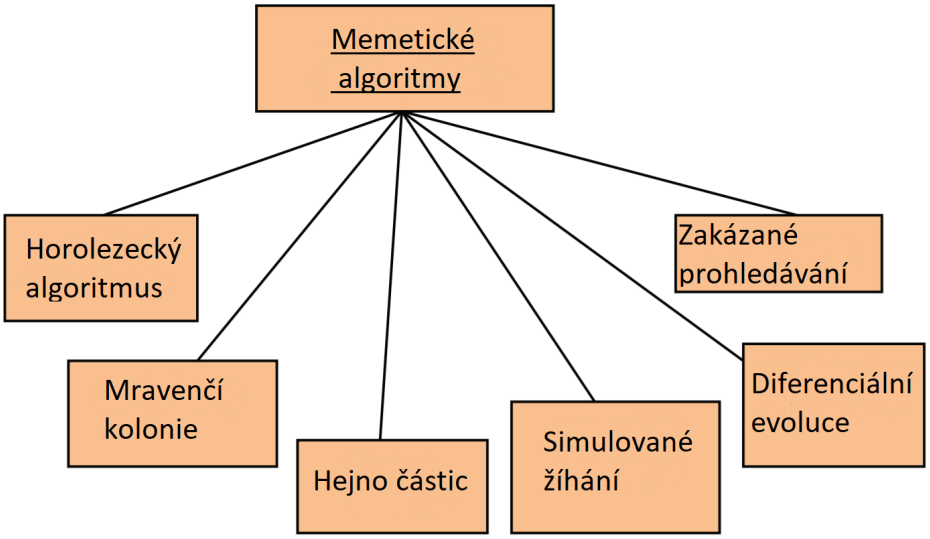
Již mnohokrát se ukázalo, že EA dokáží najít inovativní či zcela nová řešení inženýrských úloh, která konvenční algoritmy nejsou schopna poskytnout [2].

Oproti konvenčním optimalizačním metodám vykazují EA především tyto vlastnosti:

- Výhody
  - Jednodušší a flexibilnější návrh - není třeba pokročilých znalostí matematiky či fyzikálních principů k použití těchto algoritmů.
  - Umožňují optimalizovat systémy o kterých nemají žádné informace.
  - Pokud ovšem tyto informace máme, jsme schopni je využít ke zrychlení/zefektivnění optimalizace.
  - Robustnost a schopnost přizpůsobit se změnám v prostředí či prostředí samotného a šumu.
- Nevýhody
  - Samotná povaha metod je stochastická a tedy nezaručuje optimálnost řešení.
  - Algoritmy této rodiny jsou často velice náchylné na nastavené parametry a špatné zvolení těchto parametrů vede na selhání.

Další rodina algoritmů, která se do skupiny evolučních algoritmů zařadila nedávno, se nazývá *Memetické algoritmy*, nebo také kulturní algoritmy či genetické lokální prohledá-

vání [21]. Na rozdíl od evoluce celých populací se zaměřují na evoluci jedinců - selekčním tlakem zde není schopnost přežití v přírodě nýbrž ve společnosti.



Obrázek 3.1: Memetické algoritmy. Převzato z [21]

Selekční tlak

Při prohledávání stavového prostoru je neustále třeba hledat rovnováhu mezi prohledáváním a „vykořisťováním“ [2]. Prohledávání nutí jedince v populaci pokrýt celou plochu prohledávaného prostoru a vykořisťování umožní jedincům specializovat se v nějaké podoblasti prostoru a skutečně konvergovat k optimálnímu řešení v tomto podprostoru [22]. Tato síla se nazývá selekční tlak a pohání každý EA. Velikost selekčního tlaku ovlivňuje tendence algoritmu prohledávat a vykořisťovat. Čím větší je selekční tlak, tím menší stavový prostor algoritmus prohledává a tím více je hnán ke konvergenci k optimu v tomto podprostoru a naopak [22].

Pojmy

Následuje výčet a vysvětlení používaných pojmů v evolučních algoritmech [2]. Tyto pojmy mají svůj základ v biologii, přesněji v DNA a RNA [21]:

- *Fenotyp/jedinec* - objekt a potencionální řešení daného problému.
- *Chromozom/genotyp* - struktura řešení pro algoritmus. Konkrétní podoba chromozomu představuje jeden stav prohledávaného prostoru. Jeho interpretací získáváme fenotyp.
- *Gen* - element chromozomu .
- *Alela* - hodnota genu.
- *Lokus* - pozice genu v chromozomu.
- *Populace* - struktura n chromozomů. Multimnožina.

- *Generace* - jedna iterace. Každá generace má svou populaci, na kterou jsou v průběhu generace aplikovány genetické operátory. Výsledkem těchto operací je nová populace.
- *Genetické operátory* - způsob a postup tvorby potomků z rodičovských chromozomů.
- *Ukončovací podmínky* - podmínka, při které je evoluce prohlášena za dokončenou. Typicky dosažení požadovaného výsledku nebo dosažení maximálního počtu generací.
- *Fitness funkce* - funkce vyhodnocující stupeň adaptace jedince s ohledem na kritéria a cíle evolučního procesu, zjednodušeně schopnost jedince přežít.
- *Prohledávaný prostor* - množina všech potencionálních řešení problému. Každý bod v tomto prostoru má jistý potenciál pro přežití; svou fitness. Velikost a tvar tohoto prostoru je závislý na množině potencionálních řešení optimalizovaného problému. Zároveň ze znalosti řešeného problému mohou vyplynout další podmínky, které tento prostor dále omezí - tzv. omezující podmínky. Tato omezení nemusí a často nebývají lineární - například omezení křivkou nelineární funkce derivace některé z proměnných.

Příklad: Pokud je fenotypem ASCII znak, chromozomem bude struktura osmi bitů, genem bit, alela bude hodnota 1 či 0 a lokus bude pozice zvoleného genu v rámci znaku (0 - 7).

3.1 Genetický algoritmus

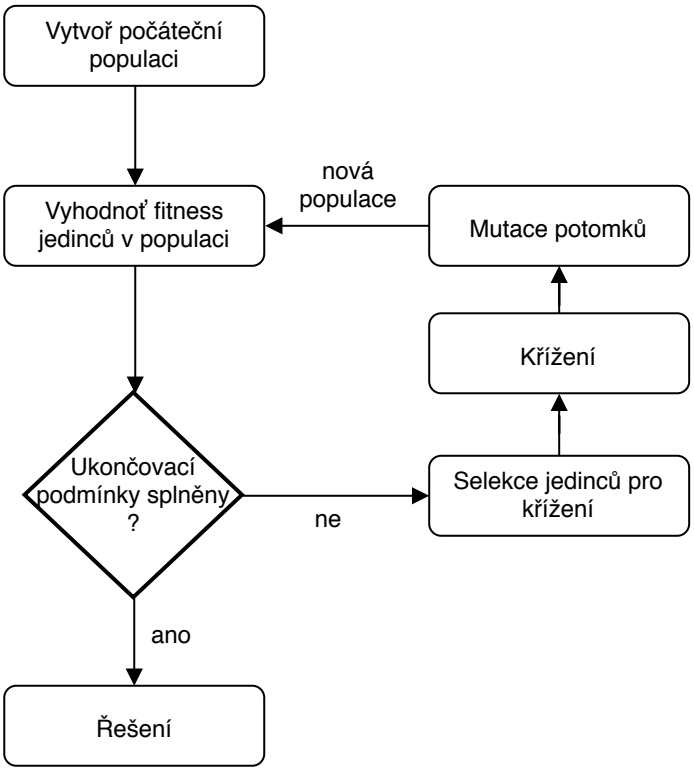
Genetický algoritmus (GA) je jedním z nejpopulárnějších z rodiny EA. Byl inspirované Darwinovou teorii evoluce a jako první jej popsal John Holland v roce 1975 [14]. Často je používán při strojovém učení, rozpoznávání a klasifikace a pro optimalizace. GA je populačně zaměřený EA a jako evoluční operace používá křížení a mutaci. Genetický algoritmus je běžný ve dvou variantách:

- *Generační model*, v literatuře často „generation GA“ - vždy je celá populace jedinců nahrazena následující populací.
- *Ustálený model*, v literatuře „steady-state GA“ - v každé generaci je nahrazován jen nejhorší jedinec.

Algoritmus začíná s nějakou, typicky náhodně vygenerovanou, populací chromozomů. Každý jedinec populace je jedním řešením daného problému a principem je postupně se v průběhu generací za pomoci genetických operátorů dostávat k novým a kvalitnějším jedincům. Ohodnocení kvality každého z jedinců probíhá za pomoci fitness funkce, která je specifická pro řešený problém. Toto je prováděno do té doby, dokud není splněno nějaké ukončovací kritérium. Flowchart 3.2 ukazuje rozhodovací proces algoritmu.

3.1.1 Kódování chromozomu

Jedná se o způsob, jakým je reprezentována struktura chromozomu. Lze uvažovat například kódování binární - 8 bitů tvoří chromozom a každý bit je jedním genem. Nebo také ale bity nemusí být pro nás významnou hodnotou a chromozomem je například binární strom, či seznam prvků nebo i obyčejné číslo (reálné nebo i přirozené).



Obrázek 3.2: Flowchart průběhu genetického algoritmu.

Binární kódování

Původní varianta GA používala právě toto kódování. Typickým případem použití tohoto kódování jsou problémy, jejichž jedinci nejsou definováni na množině čísel, případně potřebujeme velkou kontrolu nad evolucí jedinců. Nevýhodou použití tohoto kódování pro číselné typy, respektive typy, na nichž je definována relace uspořádání je tzv. *Hammingova bariéra* - malá změna v genotypu jedince může způsobit velké změny fenotypu. Například čísla $127d = 01111111b$ a $128d = 10000000b$. Toto může snižovat výkonnost algoritmu [22]. Jedním z možných řešení je tzv. *Grayův kód* - binární kódování čísel, ve kterém se sousední hodnoty vždy liší pouze v jednom bitu. Pro ukládání reálných čísel je nejdříve třeba tyto čísla převést na celá čísla.

Číselné

Typicky používáno pro optimalizační úlohy definované na množině $\mathbb{N}$ či $\mathbb{R}$. V takovýchto úlohách se běžně pracuje s číselnými fenotypy a není třeba nižší úrovně ukládání hodnot. Dalším příkladem použití celočíselných chromozomů je *pořadové kódování*, které se používá pro úlohy hledání cest. Poté může být každý uzel reprezentován jedním číslem a chromozom obsahovat seznam čísel, jejichž pořadí udává pořadí, ve kterém jsou uzly navštíveny.

3.1.2 Selekce

Představuje postup, jakým jsou vybírány chromozomy aktuální populace pro proces křížení [2, 22]. Následně jsou uvedeny nejběžněji používané selekční operátory.

Turnaj

Náhodně vybráno m jedinců, kteří se účastní turnaje. Vítěz tohoto turnaje je poté vybrán pro křížení. Podmínky pro vítězství v turnaji se mohou lišit, typické je ovšem použít hodnotu fitness.

Vážená ruleta

Pravděpodobnost výběru každého jedince závisí na jeho fitness hodnotě a na fitness hodnotě zbytku populace. Představme si ruletu, kterou roztáčíme pokaždé, když chceme vybrat jedince. Čím lepší fitness má jedinec vzhledem ke zbytku populace, tím větší je jeho část rulety a tedy tím větší šanci má, že bude vybrán [2, 22].

Rank

Jedinci v populaci jsou seřazení podle své fitness od nejlepšího k nejhoršímu. Každému jedinci je přiřazeno číslo k, které například určuje, kolik jedinců v populaci je horších, než je on sám. Následně je dle tohoto čísla spočítána pravděpodobnost každého jedince k výběru [2, 22].

Elitismus

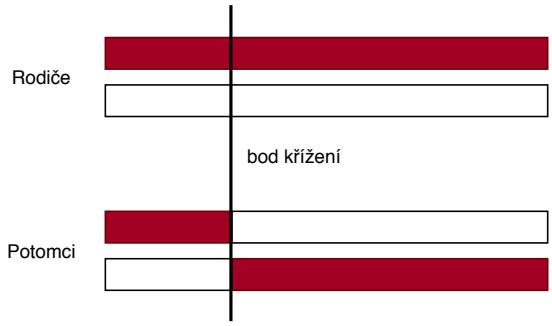
Nejlepšího jedinec (případně n nejlepších) přežije svou generaci a dostane šanci uplatnit se v následující. Takovýto jedince přeskakuje evoluční operátory. Elitismus má potenciál snižovat diverzitu populace a naklánět selekční tlak spíše k vykořisťování [2]. Jedná se o emergentní 3 vlastnost některých selekčních operátorů. V případech, kdy nepoužíváme tyto operátory, můžeme tuto vlastnost algoritmu zavést explicitně.

Incest

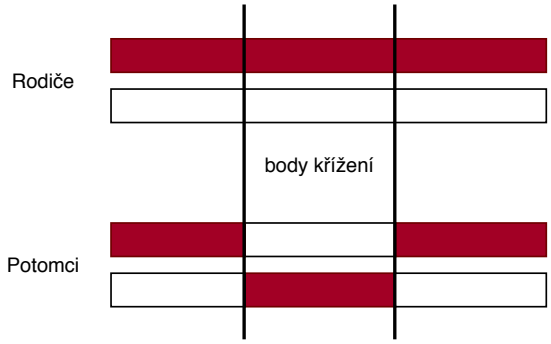
Jedince s podobnými chromozomy lze považovat za příbuzné a pokud by mělo dojít ke křížení takovýchto jedinců, nevznikne žádná nová vlastnost. Tento jev snižuje diverzitu populace a může vést k předčasné konvergenci a je třeba se mu bránit. Tuto situaci lze detekovat za pomoci *Hammingovy vzdálenosti*, případně *Euklidovské metriky*. Pokud jsou dva jedinci vybraní k páření příbuzní, je třeba zamezit křížení a provést selekci nového jedince [22].

3.1.3 Křížení

Je kombinací rodičovských genů za cílem vytvoření potomstva. Má význam pro zvýšení schopnosti explorace algoritmu a zajištění diverzity, ovšem také ke schopnosti algoritmu konvergovat. Spolu s vhodnou selekční metodikou tlačí populaci k jedincům s lepšími vlastnostmi. V základním provedení 2 rodiče produkují 2 potomky. Následující varianty operace křížení předpokládají standardní binární či celočíselnou reprezentaci. Pokročilejší zakódování vyžadují speciální návrhy těchto operátorů. V případě číselných chromozomů je možné pro křížení použít například operace průměru či sečtení [2, 22].



Obrázek 3.3: Jednobodové křížení.



Obrázek 3.4: Vícebodové křížení.

Jednobodové

V chromozomech o délce n je náhodně zvolen index $i, 1 \leq i \leq n$. Prvních i genů je poté získáno z prvního rodiče, zatímco i až n z druhého rodiče. Záměnou pořadí rodičů můžeme získat až dva potomky [5]. Znázorněno na obrázku 3.3.

Vícebodové

Analogicky k jednobodovému křížení, je vygenerováno až k různých indexů $(i_0, ...i_k)$, pro které platí $(1 \leq i_1 \leq i_2 \leq ... \leq i_k \leq n)$. Poté se vybere náhodný rodič, ze kterého je zkopírována část i_j až i_{j+1}, posune se index j a rodiče alternují. Záměnou pořadí je i zde možné získat dva potomky [5]. Znázorněno na obrázku 3.4.

Uniformní

Každému genu je vygenerována rovnoměrným rozložením zvoleno, ze kterého z rodičů bude gen zkopírován. Záměna rodičů opět umožní získání druhého potomka.

3.1.4 Mutace

Klíčový genetický operátor z pohledu přínosu nových vlastností jedinců [2, 22]. Obvyklým postupem je zcela náhodná změna několika náhodně vybraných genů u náhodně vybraných jedinců. Smyslem mutace je zvyšovat diverzitu populace. Pravděpodobnost výskytu mutace se typicky volí jako nízká; při neúměrně vysoké mutaci totiž může docházet k narušování dříve slibných řešení před konvergencí k optimu. Ovšem příliš nízká mutace může vést na nedostatečnou diverzitu populace a s tím spojené omezené schopnosti GA prohledávat stavový prostor. V binárních chromozomech můžeme náhodně měnit bity, v číselných například náhodně přičíst konstantu nebo vynásobit chromozom nějakou váhou či maskou [5].

3.2 Simulované žíhání

Metoda simulovaného žíhání patří mezi memetické algoritmy a její základ je v metalurgii, přesněji ve fyzikálním jevu žíhání - procesu, při kterém je těleso umístěno do pece vyhřáté na vysokou teplotu a postupným pomalým ochlazováním jsou odstraňovány vnitřní defekty tělesa. Po dokončení je ocel stejně pevná ovšem ohebnější a odolnější vůči poškození.

Při vysokých teplotách je těleso v tekutém stavu, krystalová mřížka náhodně uspořádána s krystalky kmitajícími v prostoru - systém je ve stavu vysoké entropie. Postupným

ochlazováním se celková entropie systému snižuje - krystalky mřížky přestávají kmitat a pomalu se usazují do pozic s nižší energií. Celý systém se tak dostává do rovnovážného stavu s pevnou mřížkou bez defektů.

Algoritmus simulovaného žíhání následuje tento princip. Tělesem je náš optimalizovaný systém, u kterého se předpokládá, že začíná ve stavu vysoké entropie. Kandidátní řešení algoritmu je potom energie systému, přesněji uspořádání krystalků mřížky. Postupem času se systém ochlazuje dle nějakého předem zvoleného chladícího rozvrhu a mřížka se ustaluje - v každém kroku jsou zaváděny náhodné poruchy momentálního stavu (je prohledáváno blízké „okolí“ současného stavu) a je porovnávána energie při těchto defektech s energií momentální. V případě, že některá z poruch má nižší energii, než stav momentální, určí jej algoritmus za nejlepší řešení v rámci této iterace a přijme jej za nový výchozí stav. V případě, že se nenašel lepšího kandidáta, je proveden test na *Metropolisovo kritérium*. Pokud je test úspěšný je i horší stav přijat za momentální. Toto kritérium je založeno na Boltzmannové pravděpodobnostním rozdělení a šance na přijetí klesá spolu s teplotou [11]:

$$W_T(E_i) = \frac{1}{Z(T)} \exp(\frac{-E_i}{k_B T}) \tag{3.1}$$

kde T je teplota žíhaného tělesa, E_i je energie systému ve stavu i, k_B je Boltzmannova konstanta. Partiční funkce Boltzmannova rozdělení $Z(T)$ je rovnice 3.2:

$$Z(T) = \sum_j \exp(\frac{-E_j}{k_B T}) \tag{3.2}$$

Znázorněno na obrázku 3.5.

3.2.1 Boltzmannovo rozdělení

Za podmínky, že proces ochlazování je dostatečně pomalý je žíhaný systém vždy rovnovážném stavu - tento jev popisuje *Boltzmannovo rozdělení pravděpodobnosti* (rovnice 3.1 a 3.2). V případě, kdy dochází k ochlazování systému příliš rychle mohou defekty zamrznout a vzniknout tak metastabilní struktury - lokální minima - které SA nedokáže překonat [11]. Pro simulaci tohoto rozdělení lze použít, a v metodě simulovaného žíhání se používá, algoritmus *Metropolis*, který je implementací metody *MCMC - Markov Chain Monte-Carlo* [4].

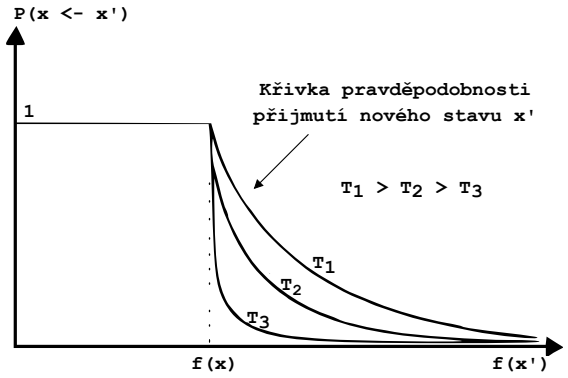
3.2.2 Monte Carlo a Markovské řetězce

Monte Carlo je stochastická numerická metoda pro získávání znalostí o simulovaných systémech pracující s teorií pravděpodobnosti. Metoda provádí n náhodných pokusů se systémem a po dokončení statisticky odhaduje charakteristiku systému. Metoda **Monte Carlo** je modelování takové náhodné veličiny X, že její střední hodnota $E(X)$ je rovna hledané hodnotě a. Pak, jestliže vypočteme n nezávislých realizací $X_1, ..., X_n$ náhodné veličiny X, můžeme odhadnout a pomocí aritmetického průměru [6].

Předpokládejme, že máme diskrétní množinu hodnot a jí odpovídající diskrétní množinu výsledků - stavů. Poté můžeme **Markovův řetězec** použít pro popis modelu tohoto systému [1, 2]. Zjednodušeně - jedná se o model série událostí, které jsou v nějakém pravděpodobnostním vztahu. Na základě statistiky tento model předpovídá, jaká událost nastane

příště bez znalosti předchozích událostí - model nemá paměť. (tzv. *Markovská vlastnost*).

Spojením těchto dvou modelů získáváme **Markov Chain Monte-Carlo (MCMC)**. Metodu, která stejně jako Monte Carlo získává znalosti o systému za pomoci náhodných pokusů, ovšem tyto náhodné pokusy vytváří stavy markovského řetězce, který vykazuje vlastnost zvanou *ergodicita* [2]. Ergodicitní modely, mimo jiné, naleznou rovnováhu v náhodném rozložení a ustálí se kolem tohoto stavu (tzv. stacionární rozdělení). Markovské řetězce také zavádějí závislosti mezi momentálním a následujícím stavem ve zkoumané distribuční funkci. Analýzou takto vzniklého řetězce jsme schopni získat informace o systému, které obyčejné Monte Carlo nedokáže. MCMC je také mnohem vhodnější pro získávání vzorků z více-dimenzionálních distribučních funkcí. Problémem tohoto přístupu je hlavně citlivost na zvolený počáteční stav. Dále pro ně platí „halting problem“ - nevíme, jestli řetěz již konvergoval [4].



Obrázek 3.5: Křivka pravděpodobnosti přijetí horšího řešení v závislosti na teplotě systému. Zdroj: internet.

3.2.3 Metropolis

Algoritmus Metropolis [4] je MCMC metoda pro získávání náhodných vzorků z pravděpodobnostní distribuce, u které je toto vzorkování složité a nebo je distribuční funkce vícedimenzionální. Algoritmus vznikl jako způsob, jak počítačovou simulací zajistit vlastnosti termodynamického systému za pomoci MCMC metody. Pseudokód algoritmu je možné vidět na 2. Algoritmus cyklicky provádí následující:

- Vygeneruj kandidátní řešení za pomoci funkce zvané „kernel“. Tato funkce je poskytnuta uživatelem a může být třeba i funkcí generující náhodné čísla. V případě SA je to funkce **perturberance** - 3.3 - která je volená tak, aby byla symetrická; pravděpodobnost toho, že malou poruchou se ze stavu A stane stav B je stejná, jako že ze stavu B se stane stav A. Symetričnost kernel funkce je podmínkou algoritmu Metropolis (ovšem SA funguje i bez splnění tohoto kritéria)
- Vygeneruj akceptační kritérium - funkce pravděpodobnosti, která určuje, jak moc se navržené řešení liší od skutečného stavu navrženého Markovským řetězem (stacionární distribuce zmíněná dříve). v případě SA se jedná o **Metropolisovo kritérium** 3.4 [11] [4].

- Pokud jsou si dostatečně podobné, vygenerovaný stav je přidán do Markovského řetězu [4].

$$x_{i+1} = x_i + u \ * |x^{max} - x^{min}| * \frac{T}{T^{max}} \tag{3.3}$$

3.1: Funkce perturberance - výběru kandidátního stavu.

kde u je náhodné číslo rovnoměrného rozložení mezi -1 a 1, x_i je fitness momentálního stavu, x^{max} a x^{min} jsou fitness hodnoty prozatím nejhoršího a nejlepšího nalezeného stavu resp., T je momentální teplota systému a T^{max} je počáteční teplota.

$$\exp(-(\frac{\Delta E}{T_i})) \tag{3.4}$$

3.2: Metropolisovo kritérium pro přijetí horšího stavu v simulovaném žíhání.

3.2.4 Rozvrh chlazení

Většina parametrů simulovaného žíhání - definiční obor, výběr následujícího stavu, atd. - jsou dány již v definici a za běhu se nemění. O to více důležitý je pro nalezení optima výběr správného chladícího rozvrhu. Běžně je nejpoužívanější lineární rozvrh chlazení. Ten lze vyjádřit následovně [5]:

$$T(i) = T_0 * \alpha^i \tag{3.5}$$

3.3: Běžný lineární rozvrh chlazení systému SA. $(0 \leq \alpha \leq 1)$

Pro vhodně malé α, je tento rozvrh dostatečně pomalý a umožní globální konvergenci.

3.2.5 Algoritmus

Pseudokód algoritmu simulovaného žíhaní [5] využívající metodu Metropolis je definován v algoritmech 1 a 2. Je vhodné podotknout, že parametry funkcí například *Perturb* se považují za proměnné v globálním prostoru.

Algoritmus 1 Algoritmus simulovaného žíhání

```
T := T_max;
x_best := náhodně vygenerovaný stav;
while T > T_min do
    x_best := Metropolis();
    T := T * α;
end while
return x_best;
```

Algoritmus 2 Metropolis

$k := 0$;
$x_{new} := x_{best}$;
while $k < k_{max}$ **do**
  $k := k + 1$;
  $x_{new} := Perturb()$;
  $P := \min(1, \frac{exp(-(f(x_{new}) - f(x_{best}))}{T}))$;
  if $random() \leq P$ **then**
    $x := x_{new}$;
  end if
end while
return x_{new};

3.3 Tabu prohledávání

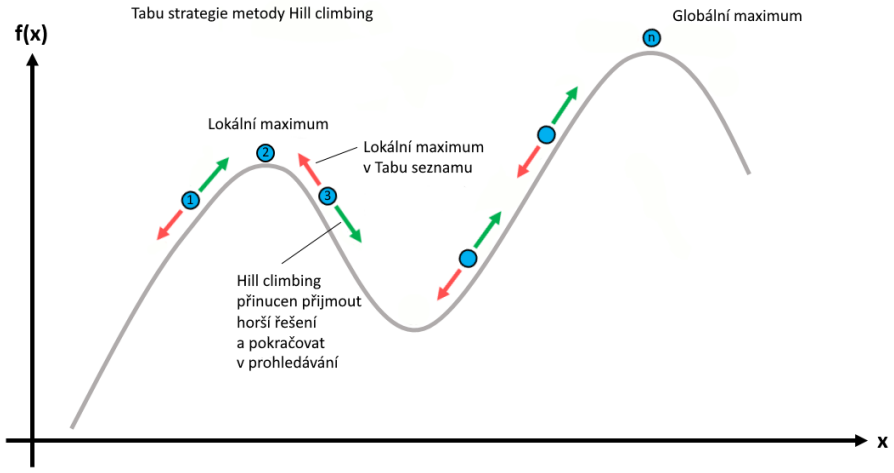
Zakázané prohledávání - častěji v originálním názvu „Tabu prohledávání“ či „Tabu strategie“ - je globální optimalizační metoda a meta-heuristika zastřešující rodinu metod, které vnášejí do optimalizačních algoritmů paměťové struktury pro překonání lokálního optima [8]. Tabu prohledávání je založeno na premise, že každé řešení problému, které lze klasifikovat jako inteligentní, musí začlenit do svého systému adaptivní paměť, a neslepé, reagující prohledávání. Poté, pokud systém s pamětí udělá špatnou volbu, je možné se z této volby díky paměti poučit a upravit strategii. To je v kontrastu s návrhem dalších optimalizačních metod založených na přírodních jevech, jakou je například *Simulované žíhání* [8]. Tabu prohledávání dostalo své jméno od slova Tabu - *zakázaný* - kvůli své inspiraci v kulturní evoluci (tabu prohledávání je memotický a evoluční algoritmus). V kontextu společnosti jsou „tabu“ věci taková témata, o která společnost nestojí a jedinec, který na toto nedbá, bude na nižším společenském postavení a tedy nemusí přežít.

3.3.1 Horolezecká metoda

Horolozecký algoritmus, iterativně prochází stavový prostor. Vždy se z momentálního stavu rozhlíží po okolí a pokud je některý z těchto okolních stavů lepší, přijme jej za současný. Tato logika je opakována po k iterací nebo dokud není dosaženo některé z ukončovacích podmínek. Takto nejlepší nalezený jedinec je prohlášen za optimum a algoritmus ukončen. Tato metodika často vede na uváznutí v oblastech lokálního optima - stav, kdy všechny relativně blízké stavy mají horší fitness. Tento problém je elegantně řešený Tabu strategií zavedením paměti - tzv. Tabu seznamu. Tabu úpravu běžného horolezeckého algoritmu je možné vidět na obrázku 3.6.

3.3.2 Adaptivní paměť

Na rozdíl od běžných lokálních metod tedy Tabu strategie pracuje se skutečně dynamickým sousedstvím. Tabu prohledávání posouvá lokální heuristiky tak, že upravuje okolí $N(x)$ (x je momentální stav), ze kterého mohou vybírat následující řešení. Toto nové okolí - $N*(x)$ - je omezováno třemi různými typy pamětí [7]:



Obrázek 3.6: Příklad využití strategie Tabu prohledávání nad lokální heuristikou Horolezecké metody (Hill climbing). Bez použití tabu strategie by se Hill climbing dopracoval do stavu 3, ve kterém by lokálním prohledáním znovuobjevil stav 2 a přesunul by se do něj.

- *krátkodobá paměť* - jednoduchý seznam obsahující n posledně navštívených stavů, kde n je vstupním parametrem algoritmu. Všechny stavy v tomto seznamu jsou vyloučeny ze sousedství.
- *střednědobá paměť* - pravidla, jejichž účelem je vést vyhledávání ke slibným oblastem stavového prostoru. Zde se reálně budou nacházet omezující podmínky specifické pro řešený problém, případně pravidla urychlující konvergenci zakázáním jistých vzorů při vyhledávání.
- *dlouhodobá paměť* - pravidla pro udržení diverzity.

Rozdíly, mezi těmito druhy pamětí jsou často velice mlhavé a co je pro jeden problém údaj střednědobý, může být v jiné údajem dlouhodobým nebo naopak. Toto je navíc pouze zjednodušený náhled; koncept těchto pamětí je mnohem složitější a její rozbor pro tuto práci není podstatný. Je pouze vhodné dodat, že adaptivní paměť tabu strategie se dále dělí na nedávnou a opakující - *nedávná* ukládá samotné vlastnosti řešení, které se změnily v nedávné době (jedná se o jistou úroveň granulity - nezakazujeme celé stavy ale například stavy, jejichž jedna vlastnost je omezena). *Opakující* poté, jak název napovídá, zamezuje cesty, které vedou na cyklus či předčasné konvergenci jedné vlastnosti . Pro zájemce doporučují například tento článek [7] samotného autora algoritmu, ve kterém se této problematice věnuje do hloubky.

3.3.3 Algoritmus

Pseudokód zakázaného prohledávání využívající Horolezeckou metodu je definován v algoritmu 4. Je vhodné podotknout, že se jedná o vysokou abstrakci, která prezentuje pouze hlavní smyčku optimalizace a vynechává implementace metod pracujících s tabu seznamem. $N(x)$ je sousední funkce tak, jak je definována optimalizovaným problémem, $StopCondition()$ je metoda ukončující optimalizace - například dosažením maximálního počtu iterací nebo nalezením optima, pokud je známé.

Algoritmus 3 Strategie Tabu prohledávání využívající Hill-Climbing

$TabuList := InicializujTabuList()$;
x_{best} := náhodně vygenerovaný stav;
while $!UkoncovaciPodminka()$ **do**
  $N^*(x) := Sousedstvi(N(x), TabuList)$
  $x := HillClimbingKrok(x, N^*(x))$;
  if $Fitness(x) \leq Fitness(x_{best})$ **then**
    $x_{best} := x$
  end if
  $AktualizujTabuList(TabuList, x)$
end while
return x_{best};

3.4 Diferenciální evoluce

Diferenciální evoluce, dále DE, je variantou genetického algoritmu specializovaná na problémy řešení spojitých problémů v reálné doméně (existují ovšem i celočíselné varianty) [22]. Snahou bylo vyvinout robustní, snadno použitelný evoluční algoritmus pro komplexní reálné problémy a to takové u nichž neznáme přesný předpis účelové funkce [2]. Na rozdíl od ostatních evolučních algoritmů pro řešení úloh v reálné doméně, se DE vzdává myšlenky, ve které je odvozování nových bodů „slepou funkcí náhody“ a namísto ní používá k odvození informace z bodů současných. Inspiraci nalézá v deterministické metodě *Nelder-Mead*.

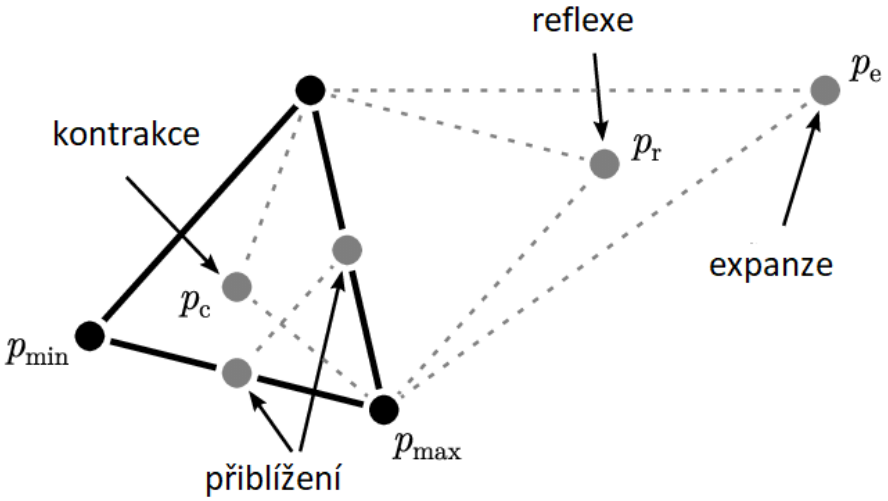
DE vykazuje následující vlastnosti:

- minimální počet řídicích parametrů.
- aplikovatelnost na nediferencovatelné funkce.
- dobrá a rychlá konvergence.
- snadná paralelizace.

3.4.1 Nelder-Meadův algoritmus

Jedná se o známou lokální heuristickou optimalizačních metod pro více-dimenzionální neomezenou optimalizaci bez použití derivací. Myšlenkou metody je obalit stavový prostor do konvexní obálky - simplexu, kterému se postupně za pomocí heuristik budou posouvat body směrem k optimu. Po dokončení optimalizace bude tento simplex ohraničovat lokální optimum systému. Body simplexu jsou posouvány podle pozice tzv. *centroidu*, což je průměrná hodnota ohodnocení všech bodů simplexu. Dle této hodnoty a pozice nejhoršího bodu se dále vypočítá *reflexní bod*. V posledním kroku heuristika určí nejvhodnější z akcí - reflexe, kontrakce, expanze, přiblížení - v závislosti na fitness hodnotách nových a dosavadních bodů [2]. Ukázka jednoho stavu a fungování metody je znázorněn na obrázku 3.7.

Definice 2. Simplex
Simplex $\mathbb{S} \in \mathbb{R}^n$ je definován jako konvexní obal o $n+1$ bodech $x_0, ..., x_n$ $x \in \mathbb{R}^n$. Pro $\mathbb{R}^2$ se jedná o trojúhelník, pro $\mathbb{R}^3$ o čtyřstěn.



Obrázek 3.7: Ukázka simplexu ve dvourozměrném prostoru a nelder-mead heuristik, které je možné nad tímto simplexem provést. Bod p_{min} může být expandován na bod p_e, kontrakcí se z něj stane centroid - bod p_c, reflexí bod p_r. Pokud ani jedna z těchto operací nevylepší fitness hodnotu tohoto bodu v prostoru, nastane operace přiblížení simplexu. Při této operaci je bod p_{max} posunut na střed jedné ze sousedních hran, podle toho, která je lepším řešením. Body p_{min} a p_{max} označují (při minimalizaci) nejlepší a nejhorší bod v rámci simplexu.

3.4.2 Řídící parametry a inicializace

DE pracuje s populací velikosti N, kde chromozom každého jedince je D-dimenzionální vektor x_i, kde $1 \leq i \leq N$ udává číslo jedince v populaci. Velikost dimenze D je dána dimenzí optimalizovaného problému. Dalšími parametry pro optimální fungování DE je *faktor zesílení* β a *pravděpodobnost křížení* CR. Špatné zvolení těchto parametrů má detrimentální vliv na schopnost algoritmu nalézt optimum v rozumném čase [22].

- **Velikost populace** má přímý vliv na velikost prohledávaného prostoru. S rostoucí velikostí populace roste pokrytí stavového prostoru, ale také časová složitost algoritmu. Empiricky bylo zjištěno, že velikost populace by měl být přibližně desetinásobek dimenze problému D pro pokrytí celého stavového prostoru. Z důvodu zrychlení konvergence algoritmu je ovšem častěji použit vztah $N \geq 2n_v + 1$, kde n_v je parametrem selekčního schéma - počet diferenčních vektorů [22].

- **Faktor zesílení** určuje velikost mutace. Doporučená hodnota je $\beta = 0.5$. Příliš velká hodnota s sebou přináší riziko přeskočení optima, zatímco příliš malá vede k pomalé konvergenci a uváznutí. Velikost faktoru zesílení by měla být v nepřímé úměře k velikosti populace.

- **Pravděpodobnost křížení** má markantní vliv na diverzitu populace. Při operaci křížení určuje počet parametrů v rámci jedince, které nebudou zděděny z rodičovského vektoru. Velká hodnota vede na vysokou diverzitu a snížení schopností lokálního prohledávání prostoru [22].

- **Strategie** - poslední a velice důležitý parametr, určující jakým způsobem bude vytvářen vektor diferencí a jak bude probíhat křížení. Tato strategie je zapisována ve

formátu $DE/x/y/z$, a platí že x označuje způsob výběru bázového vektoru, y počet vektorů pro výpočet vektoru diferencí a z je typ křížení. Mezi nejpoužívanější (ovšem zdaleka ne jediné) strategie patří $DE/rand/1/bin$ nebo $DE/best/2/bin$ [2]. Každá ze strategií s sebou přináší výhody a nevýhody a jejich studium přesahuje rámec této práce. Pro zájemce doporučuji tento článek [12], který se této problematice věnuje.

Inicializace počáteční populace probíhá náhodně, pro každý parametr vždy v rozmezí jeho definičního oboru.

3.4.3 Mutace

Mutace je v kontextu DE chápána jako vytvoření tzv. vektoru diferencí (v literatuře také šumový vektor) $\vec{v_i}$, a je to základní stavební kámen celého algoritmu. Tento vektor slouží při křížení jako druhý rodič a způsob jeho vzniku je určen dříve zmíněným parametrem DE - strategie. Následující příklad popisuje vytvoření vektoru diferencí několika běžnými strategiemi [16, 22]. Parametr typu křížení nemá na generování mutací vliv a proto je zde vynechán.

- **DE/rand/1** - šumový vektor je počítán ze tří náhodně vybraných jedinců z aktuální populace. Váhovaný rozdíl (diference) mezi dvěma náhodně vybranými jedinci $\vec{x_{r_2}}$ a $\vec{x_{r_3}}$ je připočten ke třetímu náhodnému jedinci $\vec{x_{r_1}}$. Platí, že $r_1 \neq r_2 \neq r_3$. Graficky zobrazeno na obrázku 3.8. Rovnice tohoto vztahu:

$$\vec{v_i} = \vec{x_{r_1}} + \beta * (\vec{x_{r_2}} - \vec{x_{r_3}}) \tag{3.6}$$

- **DE/rand/2** - modifikace $DE/rand/1/$. Vektor je počítán ze 4 náhodných unikátních jedinců:

$$\vec{v_i} = \vec{x_{r_1}} + \beta * (\vec{x_{r_2}} + \vec{x_{r_3}} - \vec{x_{r_4}} - \vec{x_{r_5}}) \tag{3.7}$$

- **DE/best/1** - je podobný $DE/rand/1/$; liší se pouze ve vektoru, ke kterému je připočítáván výsledek. V této strategii se jedná o x_{best}, tedy doposud nejlepší nalezený jedinec.

$$\vec{v_i} = \vec{x_{best}} + \beta * (\vec{x_{r_2}} - \vec{x_{r_3}}) \tag{3.8}$$

- **DE/best/2** - analogicky k $DE/rand/1$ a $DE/best/1$ vznikl $DE/best/2$ modifikací $DE/rand/2$.

$$\vec{v_i} = \vec{x_{best}} + \beta * (\vec{x_{r_2}} + \vec{x_{r_3}} - \vec{x_{r_4}} - \vec{x_{r_5}}) \tag{3.9}$$

Jak je vidět, ve skutečnosti nic nebrání použití více než jednoho páru vektorů pro lineární kombinaci. Obecný tvar takové kombinace lze vyjádřit takto :

$$\vec{v_i} = \vec{x_0} + \beta * \sum_{n=1}^{n_v} (\vec{x_{r_1}^n} - \vec{x_{r_2}^n}) \tag{3.10}$$

kde n_v bude počet dvojic vektorů, které si přejeme kombinovat a x_0 bude nějaký bázový vektor, který se liší od ostatních - nejlepší v *best* strategiích, náhodný v *rand* [22].

Další známe strategie jsou [22]:

- **DE/current-to-best/**n_v, které pracuje s momentálním řešením i jako bázovým, ke kterému je připočítána váhovaná vzdálenost od nejlepšího *best* spolu s váhovanou vzdáleností dalších n_v párů dvou náhodných unikátních r_1 a r_2.

$$\vec{v_i} = \vec{x_i} + \beta * (\vec{x_{best}} - \vec{x_i}) + \beta * \sum_{n=1}^{n_v} (\vec{x_{r_1^n}} - \vec{x_{r_2^n}}) \tag{3.11}$$

- **DE/rand-to-best/**n_v, které kombinuje přístup strategií *rand* a *best* - bázový vektor je poměr nejlepšího a náhodně vybraného vektoru udávaný náhodným parametrem γ.

$$\vec{v_i} = \gamma * \vec{x_{best}} + (1 - \gamma) * \vec{x_{r_1}} + \beta * \sum_{n=1}^{n_v} (\vec{x_{r_2^n}} - \vec{x_{r_3^n}}) \tag{3.12}$$

  Opět zde platí, že náhodně generované vektory musí být unikátní a $0 \leq \gamma \leq 1$.

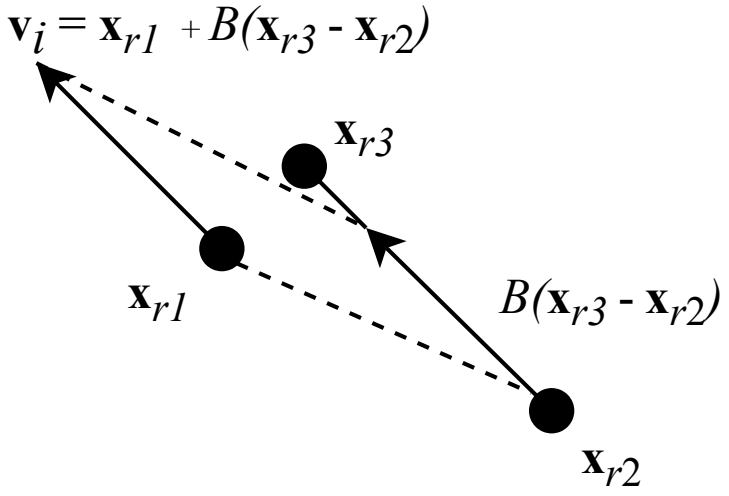
3.4.4 Křížení a selekce

Operace křížení následuje po vytvoření vektoru diferencí. Slouží k vygenerování potomka - tzv. *zkušebního vektoru* x_{new} [22]. Ten je vytvořen za pomoci následujícího vztahu:

$$\vec{x_{new}}[k] = \begin{cases} \vec{v_i}[k] & N(0,1) \leq CR \\ \vec{x_i}[k] & \text{jinak} \end{cases} \tag{3.13}$$

kde $\vec{x_i}[k]$ je gen k rodiče $\vec{x_i}$, $\vec{v_i}[k]$ je gen k vektoru diferencí $\vec{v_i}$ a $N(0,1)$ je náhodně vygenerované číslo od 0 do 1 normálním pravděpodobnostním rozložením.

Do nové populace je poté vybrán ten jedinec, který má lepší výsledky účelové funkce.



Obrázek 3.8: Ukázka generování vektoru diferencí v_i ve 2D prostoru strategií $DE/rand/1$. Převzato z [2].

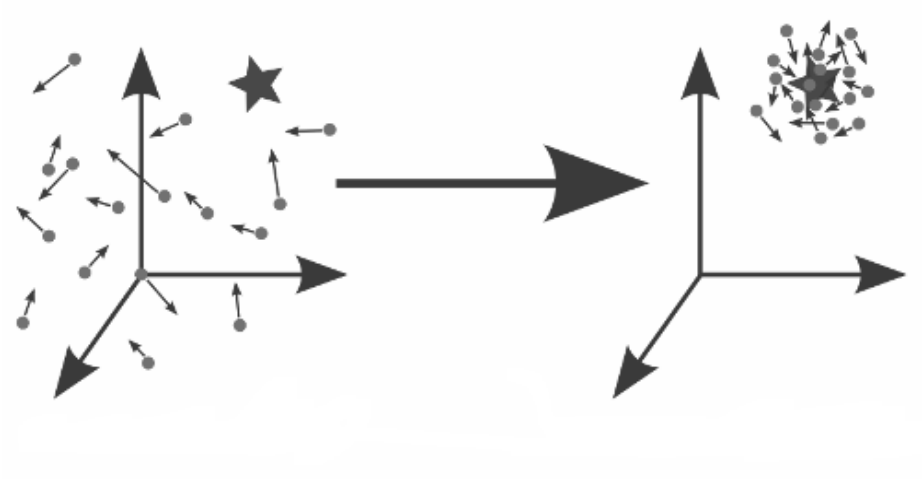
3.5 Optimalizace hejnem částic

Optimalizace hejnem částic [2], anglicky Particle Swarm Optimization - PSO, je evoluční optimalizační technika inspirovaná pohybem zvířat v hejnech, zejména ptáků. Bylo zjištěno, že schopnosti kolektivní spolupráce jedinců při vykonávaných činnostech (například hledání potravy) poskytuje těmto druhům evoluční výhodu. Metoda patří do skupiny zvané částicové systémy a od svého vzniku v roce 1995 se pro tuto skupinu stala, společně s metodou optimalizace mravenčí kolonií (ACO), hlavním představitelem. Na rozdíl od ACO, které je vhodnější spíše pro problémy diskrétního charakteru, je PSO navrženo pro problémy definované na spojité doméně.

V algoritmu je počet jednoduchých agentů - částic - umístěn do stavového prostoru optimalizovaného problému a každému z nich je spočtena jeho fitness hodnota. Každý z agentů poté spočítá svůj následující pohyb tak, že vezme v potaz historii jeho vlastního pohybu, historii pohybu zbytku agentů v hejnu a přidáním náhodné odchylky. Další iterace nastane poté, co se všechny částice pohnou na své nové pozice. Nakonec celé hejno, stejně jako hejno ptáků společně hledající jídlo, konverguje k optimu [15]. Princip zobrazen na obrázku 3.9.

Samotná částice by nebyla schopná nalézt optimum, jedná se o emergentní vlastnost začlenění částic do společnosti a komunikovat. Tato společnost nemá vůdce ani žádnou centrální inteligence, přesto je schopna býti více než pouze součtem svých prvků.

Definice 3. Emergence
Spontánní vznik makroskopických vlastností a struktur složitých systémů, jež není snadné odvodit z vlastních jejich složek.



Obrázek 3.9: Vizualizace stavového prostoru s částicemi na začátku (vlevo) a konci (vpravo) vykonávání PSO. Zdroj: internet.

3.5.1 Částice

Každá částice v hejnu se skládá ze tří D-dimenzionálních vektorů, kde D je dimenze prohledávaného problému [15].

- **Momentální pozice $\vec{x}$.**

- **Dosavadní nejlepší pozice.** $\vec{p}$**.**
- **Rychlost a směr pohybu.** $\vec{v}$**.**

Momentální pozice je inicializována náhodnou hodnotou v rámci stavového prostoru, dosavadní nejlepší pozice na zástupnou hodnotu, jejíž hodnota musí být horší než hodnota libovolného stavu ve stavovém prostoru. Rychlost a směr je inicializován náhodně ovšem v rozumném rozmezí.

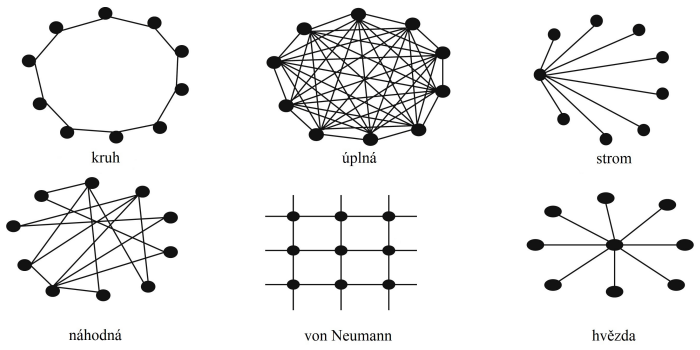
3.5.2 Topologie

Aby byla možná komunikace mezi jedinci, populace je uspořádána do jisté topologie [15] - sociální sítě. Tuto síť si lze představit jako propojení dvou částic neorientovanou hranou, celou topologii si je poté možné představit jako neorientovaný graf. Částice má poté své *sousedy*, kteří vytváří její *sousedství* (nejedná se o tranzitivní relaci). Toto rozdělení je důležité - částice ve stejném sousedství tíhnou k prohledávání stejné oblasti. Každá částice objeví svou část prostoru a informuje o tom sousedy. Ostatní částice o této oblasti dostanou informaci až od svých sousedů - informační zpoždění.

Originální PSO využívalo Euklidovského sousedství. To se ovšem prokázalo jako výpočetně náročná topologie s nevhodnými konvergenčními vlastnostmi [15]. V současné době nejpoužívanější topologie jsou založeny na sociálním sousedství. Jedinci jsou sousedy bez ohledu na to, ve které části stavového prostoru se zrovna nacházejí.

Typ sousedství se obecně rozděluje na dva druhy [15]:

- **Statická sousedství** - jak název napovídá, jedná se o sousedství která se nemění v průběhu vykonávání algoritmu. Typickým příkladem takového sousedství je tzv. *gbest* - úplná - topologie (obrázek 3.10). Dalším je *lbest* topologie - kruh. Toto sousedství dovoluje paralelní prohledávání - rozdělení na regiony podle sousedství. Tato topologie konverguje pomaleji než gbest, ovšem lépe dokáže rozpoznat lokální optima a vyhnout se jim. Kompromisem mezi těmito topologiemi je *von Neumann*, který dokáže kombinovat výhody obou. Obecně platí, že pro unimodální problémy jsou vhodnější gbest topologie, zatímco pro multimodální lbest.
- **Dynamická sousedství** - topologie měnící počet sousedů i sousedy samotné v průběhu vykonávání algoritmu. Například uveďme techniku, při které se začíná s lbest topologie pro podporu prohledávání a postupem času se přelíná do gbest pro podporu exploitace.



Obrázek 3.10: Nejznámější PSO topologie.

3.5.3 Výpočet směru

Nová pozice jedince je v kanonickém PSO udána následující rovnicí [15] (grafické znázornění je na obrázku 3.11):

$$\vec{x_i}(g+1) = \vec{x_i}(g) + \vec{v_i}(g+1) \tag{3.14}$$

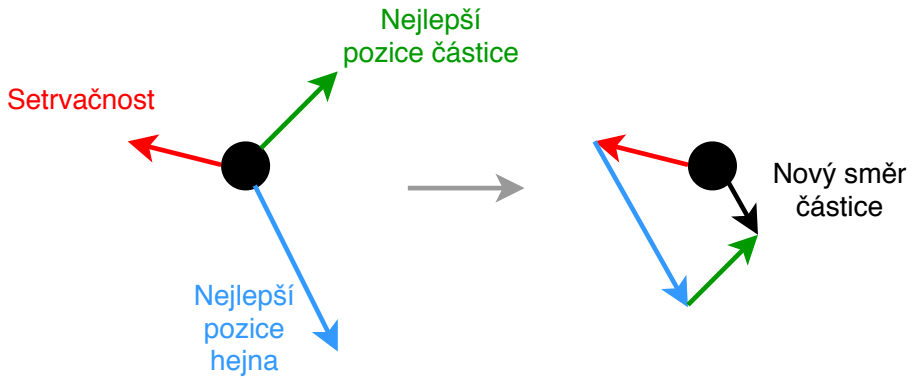
kde i udává jedince v populaci, $\vec{x_i}$ jeho pozici a $\vec{v_i}$ jeho rychlost; g je současná generace. Výpočet $\vec{v_i}$ pro gen j v generaci g:

$$\vec{v_i}[j](g+1) = \vec{v_i}[j](g) + C_1 + C_2 \tag{3.15}$$

$$C_1 = c_1 * N(0,1) * (pBest_i[j](g) - \vec{x_i}[j](g)) \tag{3.16}$$

$$C_2 = c_2 * N(0,1) * (gBest[j](g) - \vec{x_i}[j](g)) \tag{3.17}$$

- $pBest_i[j](g)$ je nejlepší nalezená pozice jedince i pro gen j v generaci g.
- $gBest[j](g)$ je nejlepší nalezená pozice v celém sousedství pro gen j v generaci g.
- c_1 a c_2 - akcelerační koeficienty.
- $N(0,1)$ - náhodně vygenerované číslo od 0 do 1 normálním pravděpodobnostním rozložením.



Obrázek 3.11: Proces výpočtu následujícího směru pohybu částice. Dosavadní směr pohybu částice je přebit směrem pohybu hejna. Zdroj: [18].

3.5.4 Parametry

PSO vyžaduje pro správné fungování nastavení při nejmenším následujících parametrů:

Akcelerační koeficienty

Koeficienty c_1 a c_2 určují, jak velkou váhu dává algoritmus nejlepším nalezeným pozicím agentem samotným ($pBest$ - c_1) a v rámci sousedství ($gBest$ - c_2). Špatně nastavené koeficienty naruší rovnováhu prohledávání a vykořisťování a neumožní algoritmu konvergovat ke globálnímu optimu. Obecně se nastavuje $c_1 = c_2 = 2$ - částice přeletí optimum v polovině případů. Čím vyšší je součet koeficientů, tím stoupá četnost oscilací kolem optima [15].

Počet částic

Vysoce závislé na typu problému, ovšem obecně se udává mezi 20 - 50 částicemi.

Maximální rychlost částice

Částicím je třeba omezit rychlost; při příliš velkých rychlostech by se mohlo stát, že částice bude provádět příliš velké kroky a optimum přeskakovat. Proto byla zaveden parametr V_{max} a V_{min}, tedy minimální a maximální rychlost, kterou se může částice pohybovat. Dále se doporučuje, aby $V_{min} = -V_{max}$. V případě, že částice povolenou rychlost překročí, je zastavena a je jí vygenerována nová náhodná rychlost ve stejném směru. I zde je volba těchto parametrů závislá na problému, příliš nízká maximální rychlost omezuje prohledávání, zatímco příliš vysoká vykořisťování.

3.5.5 Algoritmus

Pseudokód algoritmu PSO vypadá následovně:

Algoritmus 4 Optimalize hejnem částic

```
InicializujHejnoCastic();
while !UkoncovaciPodminka() do
    for all p ∈ Hejno do
        x = Pozice(p);
        if Fitness(x) ≤ Fitness(pBest(p)) then
            pBestUpdate(p, x)
        end if
    end for
    gBestUpdate(pBest, Hejno)
    for all p ∈ Hejno do
        VypocitejRychlostCastice(p)
        VypocitejPoziciCastice(p)
    end for
end while
return gBest;
```

3.6 Evoluční strategie založená na adaptaci kovarianční matice

Metoda známá pod zkratkou „CMA-ES“, z anglického „Covariance Matrix Adaptation - Evolution Strategy“ [9], je moderní state-of-the-art metoda pro optimalizaci v doméně reálných čísel. Stejně jako ostatní EA, i ze je využíváno stochastických prvků. Základní kanonická verze *evoluční strategie* využívá jako svůj hnací motor mutaci svých jedinců o náhodnou hodnotu z rozložení s pevně danou směrodatnou odchylku σ. CMA-ES tento princip rozšiřuje použitím tzv. kovarianční matice, na základě které generuje nové jedince. Tato matice je neustále adaptována podle dosavadního stavu populace a umožňuje tak dynamicky upravovat velikost kroku a tím i prohledávaný stavový prostor. Tato adaptace poté udává tzv. cestu evoluce, kterou algoritmus používá k učení a stanovení následujícího kroku

směrem k optimu. Algoritmus samotný nevyžaduje hluboké nastavování řídicích parametrů, což může být výhodou.

3.6.1 Kovariance

Statistická charakteristika, určující míru lineární závislosti náhodných veličin X a Y.

$$c(X,Y) = E[(X - E[X]) * (Y - E[Y])]$$

kde $E[X]$ je střední hodnotou náhodné veličiny X.

- $c(X,Y) > 0$ - veličiny se pohybují stejným směrem.
- $c(X,Y) < 0$ - veličiny se pohybují opačným směrem.
- $c(X,Y) = 0$ - veličiny jsou nezávislé.

Kovarianční matice C pro N proměnných je poté $N \times N$ matice, jejíž prvky jsou kovariance mezi proměnnými (C_{ij} je spočtena jako $c(I,J)$). Na diagonále této matice je poté standardní odchylka náhodné veličiny I [9]. Obecně lze vytvořit kovarianční matici pro libovolnou distribuční funkci, nejčastěji (jako i v CMA-ES) se setkáme s rozložením normálním.

3.6.2 Adaptace

Adaptace matice kovariancí zhruba odpovídá tzv. rekurzivní *rank update* funkci. Jedná se o přibližný, ovšem při vhodné volbě parametrů, dostatečně přesný odhad, abychom mohli považovat každou iteraci kovarianční matice za validní. Bylo zjištěno, že rank-one metoda je nejlepším estimátorem pro velké populace (obecně $\lambda/4 > 1 + \log n$) [9].

Rank-One Update

Základní operace nad maticí, odpovídající transformaci 3.18.

$$A \; + \vec{u} * \vec{v}^T \tag{3.18}$$

kde A je $K \times K$ matice a u a v dva $K \times 1$ vektory. Rank-One dostal svůj název kvůli hodnosti matice vzniklé vynásobením vektorů (matice hodnosti 1).

Při adaptaci vypadá rank-one update následovně [9]. Začínáme s $m \in \mathbb{R}^n$, $C = \mathbf{I}$, $\sigma = 1$, rychlost učení $c_{cov} \approx \frac{2}{n^2}$, w_i je váha, která náleží vektoru $\vec{y_i}$. Platí, že $\sum_{i=1}^{\mu} w_i = 1$ a zároveň je $w_{i=1\dots\lambda}$ nastaveno tak, aby platilo, že 3.19d $\approx 0.4\lambda$.

$$\vec{x_i} = \vec{m} + \vec{\sigma y_i}, \qquad \vec{y_i} \sim N_i(\vec{0}, C) \tag{3.19a}$$

$$\vec{m} \leftarrow \vec{m} + \vec{\sigma y_w}, \qquad \vec{y_w} = \sum_{i=1}^{\mu} w_i \vec{y_i} \tag{3.19b}$$

$$C \leftarrow (1 - c_{cov}) * C + c_{cov} * \mu_w * \underbrace{\vec{y_w}\vec{y_w}^T}_{\text{rank-one}} \tag{3.19c}$$

$$\mu_w = \frac{1}{\sum_{i=0}^{\mu} w_i^2} \approx 0.4\lambda \tag{3.19d}$$

V těchto rovnicích je skryt téměř celý algoritmus CMA-ES. Zjednodušeně:

- Je provedeno ohodnocení všech jedinců v populaci, rovnice 3.19a.
- Je vybráno μ jedinců, jejichž váhová suma, společně se standardní odchylkou posouvá nový průměr příští generace. Přiřazování vah si lze představit jako selekční mechanismus tohoto algoritmu. Je nutno zmínit, že **je vybráno μ nejlepších řešení** (získáno fitness ohodnocením $\vec{x_i}$), nikoliv náhodných. Rovnice 3.19b.
- Za pomoci zmíněné sumy je adaptována kovarianční matice C. Rovnice 3.19c.

Adaptace matice způsobí, že CMA-ES [9]

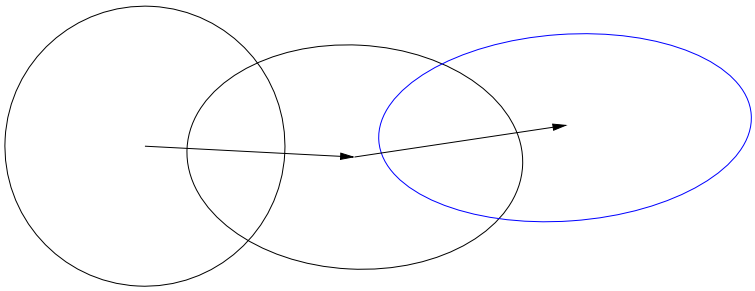
- postupně zjistí všechny párové závislosti mezi proměnnými.
- provede analýzu hlavních komponent kroků y_w tak, jak šli za sebou - rozložení na charakteristické vektory lineárních transformací kovarianční matice (rank update matice je symetrická, což zaručuje rozklad) - umožní rozpoznat libovolnou rotaci stavového prostoru proměnných a řešit takto rotovaný problém.

a pokud je $\mu = 1$ poté je pro průměr nové generace vždy vybrán nejlepší jedinec v populaci a CMA-ES provádí populační hill-climbing algoritus nad distribucí proměnných N.

Toto je v krocích téměř celý algoritmus CMA-ES. Zásadní rozdíl mezi rovnicemi 3.20 a skutečným algoritmem je v samotné adaptaci matice C. Pro její aktualizaci je ve skutečnosti vypočtena *evoluční cesta*. Dále je třeba dát CMA-ES dříve zmíněnou dynamičnost postupnou úpravou standardní odchylky σ.

Evoluční cesta

Konceptuálně se jedná o cestu, kterou se strategie vydala při hledání řešení. Znázorněno na obrázku 3.12. Rekonstrukce této cesty se nazývá **kumulací** (nebo také například „momentum“ při fázi zpětné propagace umělých neuronových sítích - učení). Kumulace je v CMA-ES spočtena dvakrát, tzv. **„isotropická“** 3.20a, pro adaptaci zmíněné odchylky a tím velikosti kroku algoritmu; a poté cesta **„anisotropická“** 3.20b. Tato cesta slouží k adaptaci kovarianční matice a směru, kterým se algoritmus vydá při zkoumání stavového prostoru [9].



Obrázek 3.12: Názorná ukázka evoluční cesty. Elipsoidy znázorňují distribuční rozsah, tvar i střed je udán kovarianční maticí v odpovídajícím kroku evoluce.

Nechť k označuje generaci

$$\vec{p_\sigma}^{(k+1)} \leftarrow (1-c_\sigma)\vec{p_\sigma}^{k} + \sqrt{1-(1-c_\sigma)^2}\sqrt{\mu_w}C^{-\frac{1}{2}}\vec{y_w} \tag{3.20a}$$

$$\vec{p_c}^{(k+1)} \leftarrow (1-c_c)\vec{p_c}^{k} + h_\sigma * \sqrt{1-(1-c_c)^2}\sqrt{\mu_w}\vec{y_w} \tag{3.20b}$$

$$\mu_w = \frac{1}{\sum_{i=0}^{\mu} w_i^2}, c_{cov} \ll c_c \approx c_\sigma \ll 1 \tag{3.20c}$$

$$h_\sigma = \begin{cases} 1 & pokud \quad \|p_\sigma\| < 1.5\sqrt{n} \\ 0 & jinak \end{cases} \tag{3.20d}$$

- c_c a c_σ jsou učící faktory pro anisotropickou a isotropickou cestu resp. Bylo ovšem zjištěno, že tyto hodnoty iterativně degradovaly přirozený rozptyl proměnných. Z tohoto důvodu byly přidány normalizační výrazy (odmocnina $\sqrt{1-(1-c_x)^2}$), které tento efekt zastavují (nastavením hodnot na 1 se z rovnic vytratí i zmíněné mocniny).
- $C^{-\frac{1}{2}}$ - symmetrická odmocnina z matice preciznosti (inverze kovarianční matice - obecně udává, jak moc je třeba transformovat originální data, aby byla nezávislá).
- h_σ je zdržovací funkce. V případě, že $\|p_\sigma\|$ je příliš vysoké, by docházelo k příliš rychlému růstu elipsoidu daného maticí C a tím i nechtěné diverzifikaci populace.

Rank-μ update

Pokud problém vyžaduje rychlou konvergenci (i za cenu globálního prohledávání), je třeba volit menší velikost populace. V takovém případě ovšem již estimátor rank-one update není validní a je třeba využít jiné metodiky. Zde přichází na řadu „Rank-μ“. K výpočtu následující kovariance je použita informace z celé generace, na rozdíl od informace o korelacích mezi generacemi.

$$C^{(k+1)} = (1-c_\mu \sum w_i)C^k + c_\mu \sum_{i=1}^{\lambda} w_i \vec{z_i}^{(k+1)}(\vec{z_i}^{(k+1)})^T \tag{3.21}$$

$$z_i^{(k+1)} = \frac{(y_i^{(k+1)} - m^k)}{\sigma_i^k} \tag{3.22}$$

3.6.3 Konečná podoba rovnic

Kombinací rank-one a rank-μ metodik vzniká výsledná podoba adaptace kovarianční matice:

$$\begin{aligned} C^{(k+1)} \leftarrow & (1-c_{cov}-c_\mu \sum w_i)C^k \\ & + \underbrace{c_{cov} * \vec{p_c}^{(k+1)}(\vec{p_c}^{(k+1)})^T}_{\text{rank-one}} + \underbrace{c_\mu \sum_{i=1}^{\lambda} w_i \vec{z_i}^{(k+1)}(\vec{z_i}^{(k+1)})^T}_{\text{rank-}\mu} \end{aligned} \tag{3.23}$$

Přidáme-li i rovnici adaptace velikosti kroku σ, získáváme kompletní algoritmus *CMA-ES*:

$$\vec{x_i} \leftarrow \vec{m} + \vec{\sigma}\vec{y_i}, \qquad \vec{y_i} \sim N_i(\vec{0}, C)$$

$$sort(\vec{y_i}) \text{ podle hodnoty } f(\vec{x_i})$$

$$\vec{m} \leftarrow \vec{m} + \vec{\sigma}\vec{y_w}, \qquad \vec{y_w} \leftarrow \sum_{i=1}^{\mu} w_i \vec{y_i}$$

$$\vec{p_\sigma}^{(k+1)} \leftarrow (1 - c_\sigma)\vec{p_\sigma}^{k} + \sqrt{1 - (1 - c_\sigma)^2}\sqrt{\mu_w} C^{-\frac{1}{2}} \vec{y_w}$$

$$\vec{p_c}^{(k+1)} \leftarrow (1 - c_c)\vec{p_c}^{k} + h_\sigma * \sqrt{1 - (1 - c_c)^2}\sqrt{\mu_w}\vec{y_w}$$

$$\mu_w \leftarrow \frac{1}{\sum_{i=0}^{\mu} w_i^2} \quad h_\sigma \leftarrow \begin{cases} 1 & \text{pokud} \quad \|p_\sigma\| < 1.5\sqrt{n} \\ 0 & \text{jinak} \end{cases}$$

$$z_i^{(k+1)} = \frac{(y_i^{(k+1)} - m^k)}{\sigma_i^k}$$

$$C^{(k+1)} \leftarrow (1 - c_{cov} - c_\mu \sum w_i) * C^k + \underbrace{c_{cov} * \vec{p_c}^{(k+1)}(\vec{p_c}^{(k+1)})^T}_{\text{rank-one}} + \underbrace{c_\mu \sum_{i=1}^{\lambda} w_i \vec{z_i}^{(k+1)}(\vec{z_i}^{(k+1)})^T}_{\text{rank-}\mu}$$

$$\vec{\sigma}^{(k+1)} \leftarrow \vec{\sigma}^k * \exp\left(\frac{c_\sigma}{1 + \sqrt{\frac{\mu_w}{n}}} * \left[\frac{\|p_\sigma\|}{E\|N(\vec{0}, \mathbf{I})\|} - 1\right]\right)$$

Empiricky bylo zjištěno:

- $\lambda \approx 4 + 3 * \log N$
- $\mu \approx \frac{\lambda}{4}$
- $c_{cov} \approx \frac{2}{n^2}$
- $c_c \approx c_\sigma \approx \frac{4}{n}$
- $c_\mu \approx \min(\frac{\mu_w}{n^2}, 1 - c_{cov})$
- $c_{cov} + c_\mu \leq 1$
- $\sum_{\frac{-c_{cov}}{c_\mu}} w_i = \sum_{j=1}^{\lambda} w_j \approx$

Iniciální hodnoty:

- $C = \mathbf{I}$
- $p_c = \vec{0}$
- $p_\sigma = \vec{0}$

Vstupy:

- $\vec{m} \in \mathbb{R}^n$ - počáteční průměrná hodnota
- $\vec{\sigma} \in \mathbb{R}_+$ - počáteční standardní odchylky jednotlivých proměnných
- k_{max} - maximální počet generací

Kapitola 4

Implementace a testování

Následující kapitola si nejprve klade za cíl představit vytvořený programový základ, který byl použit pro řešení problému hledání trajektorie sonikací. Následně představí několik běžných matematických funkcí pro testování optimalizačních metod, na kterých bude ověřena optimalizační schopnost vytvořeného řešení. Funkce nebyly vybrány náhodně, každá z nich má odlišné vlastnosti a testuje algoritmy podle jiných kritérií. V poslední části budou prezentovány výsledky testovacích běhu, vedeny úvahy nad těmito výsledky a diskutována schopnost optimalizovat trajektorii sonikací *HIFU*.

4.1 Implementace

Každý z algoritmů představených v kapitole 3 byl implementován jako spustitelný optimalizér v jazyce *C++11* za pomoci externích knihoven *GAUL*[1] pro metody *Genetického algoritmu*, *Simulovaného žíhání*, *Tabu prohledávání* a *Diferenciální evoluce*, *LibOPT*[2] pro metodu *Optimalizace rojem částic* a *CMA-ESpp*[3] pro metodu *CMA-ES*. Všechny tyto knihovny byly v malé míře upraveny od jejich běžně dostupné varianty pro potřeby této práce.

Dále vzniklo několik skriptů v prostředí *Bash* pro paralelní spuštění a sběr výsledků nezávislých optimalizací na clusterech *IT4Innovation*. Skript v jazyce *Python3* pro generování relevantních statistických grafů z těchto výsledků za pomoci balíčku *MatplotLib*[4]. Pro překlad je poskytnut vzorový `Makefile` soubor.

Implementované optimalizační programy a skripty pro kolekci dat vznikly pouze jako nezbytný nástroj pro potřeby této práce. Pro úpravy se předpokládá znalost programování, optimalizačních metod a alespoň mírná znalost rozhraní a datových struktur použitých externích knihoven. Všechen kód je řádně dokumentován v doxygen komentářích ve zdrojových souborech.

4.1.1 Sjednocení rozhraní externích knihoven

Každá z optimalizačních knihoven má výrazně odlišné rozhraní spuštění. Nastavení fitness funkce, podmínky konvergence a přerušení. I výstupního protokol průběhových dat procesu optimalizace se výrazně liší. Z toho důvodu bylo třeba vytvořit novou vrstvu mezi uživatelem a začátkem procesu optimalizace. Pro snadnou úpravu a jednotné spuštění procesu byl

[1] http://gaul.sourceforge.net/
[2] https://github.com/jppbsi/LibOPT
[3] https://github.com/AlexanderFabisch/CMA-ESpp
[4] https://matplotlib.org/index.html

vytvořen modulární návrh využívající jednotných deklarací rozhraní v hlavičkovém souboru. Tyto funkce jsou poté zakomponovány v rámci unifikované mezivrstvy mezi spouštěnou aplikací a funkcí optimalizace externí knihovny. Samotná implementace metod je optimalizačnímu programu poskytnuta při linkovacím procesu v průběhu překladu. Implementace nové fitness funkce je tedy možná definicí funkcí z jednotného hlavičkového souboru a následné přilinkování tohoto souboru při překladu. Je nutné podotknout, že vzniklé optimalizační programy minimalizují.

Modul fitness funkce

Jak již bylo zmíněno, závazné deklarace pro implementaci nové účelové funkce jsou dostupné v hlavičkovém souboru `FitnessFunction.h`. Soubor deklaruje následující funkce:

- `double fitnessFunction(int, double*)` pro výpočet fitness hodnoty. Funkce očekává na vstupu délku testovaného řešení a data ve formě ukazatele na typ double.
- `int isInConstraints(int, double**)` pro ověření splnění omezujících podmínek. Pokud vstupní data splňují všechny uživatelem definované podmínky, funkce vrací -1, v opačném případě pozici první proměnné, která omezení porušuje.
- `void applyConstraints(int, double**)` pro aplikaci omezení a úpravu dat nalezeného řešení. I zde je vstupem délka testovaného řešení a data tohoto řešení, které tato funkce může upravit. Tato funkce je vždy volána před funkcí ohodnocující fitness hodnotu řešení. Jedinou výjimkou je algoritmus CMA-ES, který tuto metodu nepoužívá.
- `void getConstraints(int, double**, double**)` pro definici omezení stavového prostoru proměnných. Tato omezení jsou vrácena do optimalizace přes vstupní argumenty typu ukazatel na ukazatel na datový typ double. Jeden pro horní omezení, druhý pro spodní omezení. Funkce je zavolána před spuštěním optimalizace pro nastavení všech potřebných dat.
- `bool isConverged(int, int, double, double, int, double**)` je volána na začátku každé generace pro kontrolu ukončovacích podmínek algoritmu. Vstupem metody jsou údaje o počtu provedených generací, počtu evaluací fitness funkce, hledaná optimální fitness, nejlepší nalezená fitness, momentální velikost populace a fitness hodnoty celé dosavadní populace. Návratová hodnota indikuje, zda-li má být optimalizační proces ukončen.

4.1.2 Mezivrstvy optimalizačních metod

Samotný optimalizační algoritmus jednotlivých knihoven je volán z mezivrstvy, která specializuje jednotné rozhraní účelové funkce na konkrétní požadavky externí knihovny. Tato vrstva je vstupním bodem každého vytvořeného programu.

GAUL optimalizační programy

Knihovna *GAUL - Genetic Algorightm Utility Library* je známou optimalizační knihovnou. Poskytuje několik běžně používaných optimalizačních algoritmu, z nichž je v této práci využito genetického algoritmu, simulovaného žíhání, tabu prohledávání a diferenciální evoluce.

Zároveň je pro tyto metody (a nejvíce pro samotný genetický algoritmus) umožněna vysoká míra přizpůsobení - uživatel si může definovat vlastní genetické operátory. Toho bylo využito při implementaci mezivrstev zmíněných metod. Byl implementován vlastní operátor `randomUniformSeedInBounds(population*, entity*)` pro inicializaci populace. Tato inicializace respektuje omezeními pro každou z dimenzí, namísto pouze jednoho globálního omezení, tak jak je to v základu v knihovně. Vstupní parametry jsou vnitřní datové struktury knihovny.

Dále vznikl mutační operátor `mutateDoubleMultipointCustomStddev(population*, entity*, entity*)`, který je schopen mutovat jedince v populaci o náhodně generované číslo z normálního rozložení a vlastní standardní odchylkou, namísto standardních nabízených knihovnou, které mutují vždy s odchylkou 1. Tyto funkce se nachází v souborech `CustomGaulFunctions.cc` a `CustomGaulFunctions.h`. Výpis textového výstupu a předčasné ukončení obstarává funkce `generationHook(int, population*)`, případně `iterationHook(int, entity*)`. Knihovna maximalizuje, a funkce `scoreFunction(population*, entity*)` byla upravena adekvátně k této skutečnosti. Vstupy pro spuštění optimalizace algoritmů z knihovny GAUL jsou následující:

- **Genetický algoritmus** - v implementované mezivrstvě je k selekci použit turnaj, křížení je prováděno dvoubodově na náhodně vygenerovaných pozicích a mutace upravuje náhodný počet alel za pomoci vlastního mutačního operátoru. Implementace této mezivrstvy se nachází v souboru `GA_Gaul.cc`. Pro spuštění optimalizace je nezbytné poskytnout následující hodnoty v uvedeném pořadí:
  - maximální počet generací.
  - velikost populace.
  - dimenze problému.
  - míra křížení - hodnoty 0 až 1.
  - míra mutace - hodnoty 0 až 1.
  - forma mezigeneračního přežití - celočíselné hodnoty 1 až 3 (1 - neudržuje se informace o generaci, po vygenerování všech potomků jsou rodiče i děti seřazeny dle fitness a nejhorších n je zabito, 2 - lepší z rodiče či dítěte je vybírán již při ohodnocení potomstva, 3 - rodiče vždy umírají).
  - optimální fitness - libovolné číslo z $\mathbb{R}$.
  - seed pro generátor náhodných čísel - libovolná posloupnost čísel.
  - příznak, přejeme-li si výpis průběhu optimalizace.
- **Simulované žíhání** - ke chlazení byl použit lineární rozvrh, tedy v každém kroku iterace je teplota snížena o daný teplotní krok. Pro generování kandidátního řešení je využita perturberance, která posune každou hledanou proměnnou řešení o náhodnou hodnotu. Tento náhodný posun je generován normálním rozložením s průměrem 0 a standardní odchylkou empiricky zvolenou na přibližně 1/5 rozmezí perturbované proměnné. I zde je použit vlastní mutační operátor, popsaný na začátku sekce. Implementace mezivrstvy se nachází v souboru `SA_Gaul.cc`. Parametry spuštění v uvedeném pořadí jsou následující:
  - maximální počet iterací.
  - dimenze problému.

  - počáteční teplota.
  - teplotní krok.
  - optimální fitness - libovolné číslo z $\mathbb{R}$.
  - seed pro generátor náhodných čísel - libovolná posloupnost čísel.
  - příznak, přejeme-li si výpis průběhu optimalizace.

- **Tabu prohledávání** - generování kandidátních řešení je zde provedeno stejnou metodou, jako u simulovaného žíhání. Knihovna nepodporuje jinou, něž dlouhodobou paměť. Zdrojovým souborem s mezivrstvou je `Tabu_Gaul.cc`. Pro spuštění optimalizace zakázaným prohledáváním je nutné poskytnout následující hodnoty v uvedeném pořadí
  - maximální počet iterací.
  - dimenze problému.
  - délka tabu seznamu - maximální počet zakázaných stavů.
  - počet testovaných kandidátních řešení.
  - optimální fitness - libovolné číslo z $\mathbb{R}$.
  - seed pro generátor náhodných čísel - libovolná posloupnost čísel.
  - příznak, přejeme-li si výpis průběhu optimalizace.

- **Diferenciální evoluce** - knihovna GAUL nabízí 3 níže uvedené strategie. Mezivrstva je implementována ve zdrojovém souboru `DE_Gaul.cc`. Parametry spuštění v uvedeném pořadí jsou tyto:
  - maximální počet generací.
  - velikost populace.
  - dimenze problému.
  - počet generovaných diferenciálních vektorů.
  - míra křížení - hodnoty 0 až 1.
  - minimální faktor zesílení - libovolná kladná celočíselná hodnota.
  - maximální faktor zesílení - hodnoty od minimálního faktoru dále.
  - typ strategie - celočíselné hodnoty 1 až 3 (1 - strategie *BEST*, 2 - strategie *RAND*, 3 - strategie *RAND-TO-BEST*).
  - typ křížení - celočíselné hodnoty 1 nebo 2 (1 - křížení *BIN*, 2 - křížení *EXP*).
  - optimální fitness - libovolné číslo z $\mathbb{R}$.
  - seed pro generátor náhodných čísel - libovolná posloupnost čísel.
  - příznak, přejeme-li si výpis průběhu optimalizace.

LibOpt optimalizační program

Knihovna *LibOpt* byla použita pro její realizaci optimalizace rojem částic. Pro využití této knihovny nebylo třeba příliš velkých úprav, pouze přizpůsobení rozhraní. Metoda využívá vlastních interních struktur `SearchSpace` a `Agent`. Stejně jako při GAUL optimalizačních programech, i zde byly implementovány funkce `scoreFunction(Agent*, ...)` a `generationHook(int, SearchSpace*)`, které plní stejné poslání. Knihovna minimalizuje.

- **Optimalizace rojem částic** - knihovna poskytuje kanonickou verzi, která uvažuje jeden roj s úplnou topologií. Mezivrstva implementována souborem `PSO_LibOpt.cc`. Pro spuštění optimalizace je nutné poskytnout následující hodnoty v uvedeném pořadí
  - počet generací.
  - počet částic.
  - dimenze problému.
  - akcelerační koeficient $c1$.
  - akcelerační koeficient $c2$.
  - faktor hybnosti.
  - optimální fitness - libovolné číslo z $\mathbb{R}$.
  - seed pro generátor náhodných čísel - libovolná posloupnost čísel.
  - příznak, přejeme-li si výpis průběhu optimalizace.

CmaESpp optimalizační program

I zde nebylo třeba větších úprav. Standardní funkce `scoreFunction(int, double*)` a `generationHook(CMAES)` jsou přítomny i zde. `CMAES` je interní datovou strukturou této knihovny. Dále byla přidána metoda, umožňující převzorkování jednotlivých genů, pro podporu hledání v omezeném prostoru. *CmaESpp* je hlavičkovou knihovnou, což zjednodušilo sestavovací a linkovací proces. I zde se jedná o minimalizaci.

- **CMA-ES** - mezivrstvu lze najít v souboru `CMAES_CmaESpp.cc`. Parametry spuštění v uvedeném pořadí jsou následující:
  - maximální počet generací.
  - dimenze problému.
  - optimální fitness - libovolné číslo z $\mathbb{R}$.
  - seed pro generátor náhodných čísel - libovolná posloupnost čísel.
  - příznak, přejeme-li si výpis průběhu optimalizace.

4.1.3 Výstupní protokol

Výstup každého z optimalizačních algoritmů byl sjednocen. Text s údaji je tištěn na standardní výstup a rozdělen do tří sekcí, oddělených posloupností symbolů `"@@@"`. Každá sekce se skládá z posloupnosti dvojic `klič:hodnota`, kde jednotlivé dvojice jsou poté odděleny symbolem `'$'` a libovolným počtem bílých znaků. V případě, kdy `hodnota` je seznam, prvky tohoto seznamu jsou oddělený znakem `','`. Tento protokol byl zvolen, jelikož jej lze jednoduše převést do formátu `JSON`, čehož je následně využito ve skriptech generujících statistické grafy. Vzor pro ukázku struktury:

```
$kl:hodnota$kl:hodnota$...$kl:hodnota$
@@@
$kl:hodnota$kl:hodnota$...$kl:hodnota1,hodnota2,...$
$kl:hodnota$kl:hodnota$...$kl:hodnota1,hodnota2,...$
@@@
$kl:hodnota$kl:hodnota$...$kl:hodnota$
```

Obsah sekcí je následující:

- První sekce je dedikována pro údaje o spuštění. Zde se nachází všechny vstupní parametry, včetně omezení stavového prostoru.
- Druhá sekce obsahuje průběžné údaje o prováděné optimalizaci (v případě, že je výpis těchto údajů povolen při spuštění optimalizéru). Zde se, mimo jiné, nachází údaje o nejlepším a nejhorším řešení v populaci, statistická data o populaci jako jsou kvartály, medián, průměr a dosavadní počet fitness vyhodnocení.
- V poslední sekci jsou poté výsledky optimalizace. Nejlepší nalezená fitness, řešení a čas, po který optimalizace běžela.

Pro získání statistických dat vzniklo několik modulů, které jsou schopny extrahovat relevantní informace z datových struktur jednotlivých knihoven. Tyto funkce jsou implementovány v souborech `LoggingHooks_[jméno knihovny].cc` a využity ve funkcích `iterationHook()`, případně `generationHook()`, která je volána vždy na začátku iterace, resp. generace.

4.1.4 Skripty pro práci s výsledky

V poslední řadě vznikly spolu s optimalizačními programy i následující skripty:

- `statsGenerator.py` - skript, který je schopen generovat statistické grafy z textových výstupů několika běhů optimalizace. Grafy jsou generovány za pomoci knihovny *matplotlib*. Veškeré výsledné grafy, které se nachází v této a následující kapitole, byly generovány tímto souborem.
- `[opt]_jobscript.pbs` - skripty pro systém PBS na clusterech *IT4Innovation*. Umožňují automatizované masivní spuštění více běhů nezávislé optimalizace. Zkratka [opt] představuje zkratku identifikující jednotlivé optimalizační metody - ga, sa, tabu, de, pso či cmaes.
- `paramOpt.py` - experimentální skript pro nalezení optimálních řídících parametrů jednotlivých optimalizačních metod za pomoci meta-optimalizace algoritmem Nelder-Mead. Ve finálním výsledku nebylo tohoto přístupu využito z důvodu přílišné výpočetní náročnosti. Pro určení parametrů byl namísto toho zvolen empirický přístup.

4.2 Sada testů

V následující sekci bude představeno několik běžných matematických funkcí pro testování optimalizačních metod. Tyto funkce byly vybrány záměrně, každá z nich má odlišné vlastnosti a testuje optimalizační algoritmus na jiné kritéria. Následně jsou ke každé funkci prezentovány statistické výsledky optimalizace provedené dříve představenými metodami. V poslední části kapitoly budou zhodnoceny metody holisticky v kontextu optimalizovaného systému *HIFU*.

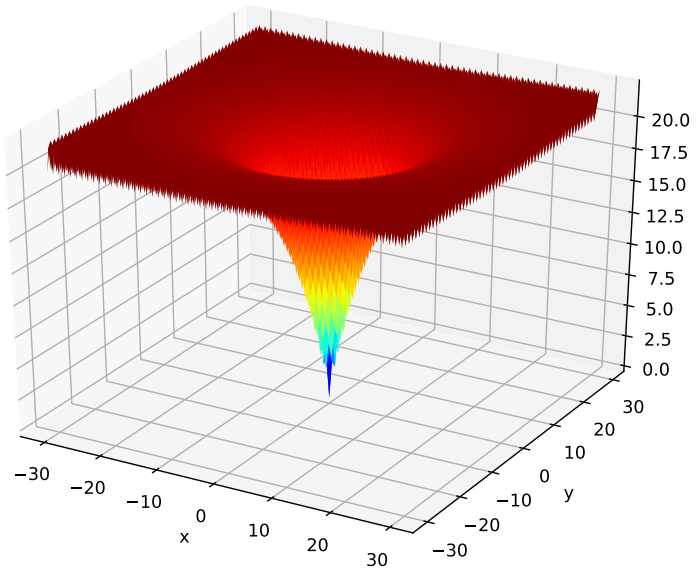
4.2.1 Acleyho funkce

První funkce v testovací sadě je *Acleyho funkce* - jedna z snad o nejznámější funkci pro testování optimalizačních metod. Je specifická svou podobou *černé díře* - jedna hluboká

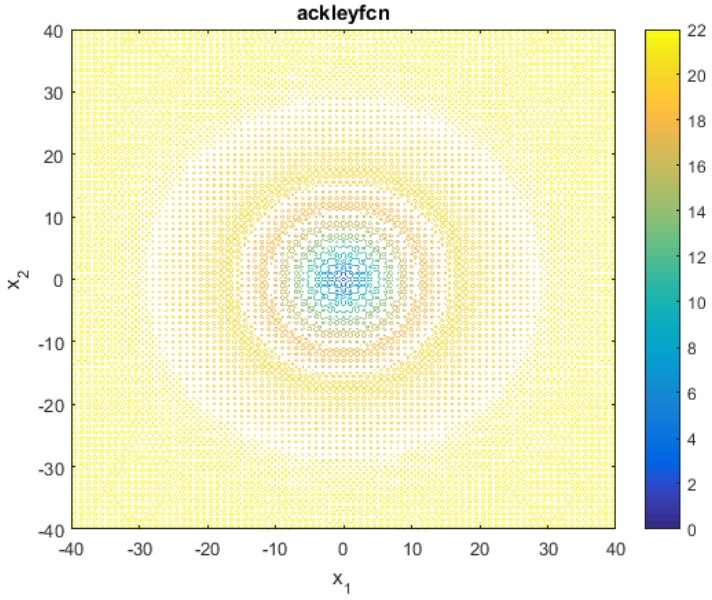
propast ve středu funkce, ve které se nachází globální minimum, zatímco okolí této propasti je poseto lokálními minimy, ve kterých mohou optimalizační metody uváznout. Vizualizace ve dvou rozměrném prostoru je možné vidět na obrázcích 4.1 a 4.2. Funkční předpis:

$$f(\mathbf{x}) = f(x_1, ..., x_n) = -a.exp(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)) + a + exp(1) \quad (4.1)$$

Pro optimalizační účely jsou argumenty funkce omezeny intervalem $[-30, 30]$ a jejím globálním optimem je $f(0, \cdots, 0) = 0$. Konstanty a, b a c jsou běžně voleny jako $a = 20$, $b = 0.2$ a $c = 2\pi$



Obrázek 4.1: Ackleyho funkce, vizualizace pro 2D.



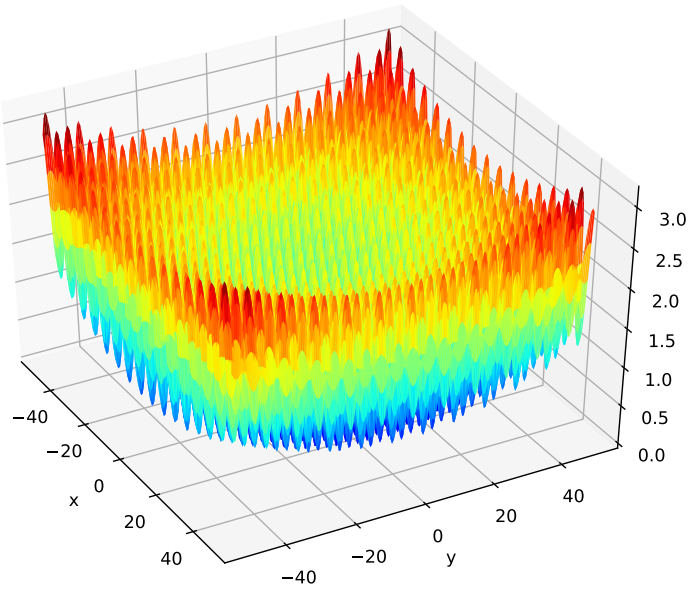
Obrázek 4.2: Ackleyho funkce, pohled na rovinu x_1,x_2.

4.2.2 Griewankova funkce

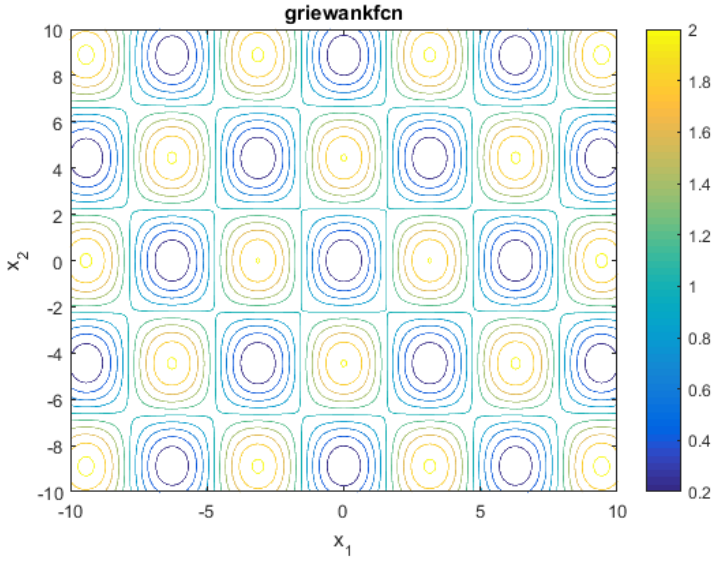
Další funkcí v testovací sadě je *Griewankova funkce* - jedna se o funkci modelující šum. Celá funkce je poseta postupně se snižujícími lokálními minimy. Nejnižší z nich se nachází ve středu; globální minimum. Tato funkce testuje především schopnost algoritmu prohledávat. Vizualizace ve dvou rozměrném prostoru je možné vidět na obrázcích 4.3 a 4.4. Funkční předpis:

$$f(\mathbf{x}) = f(x_1, ..., x_n) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) \tag{4.2}$$

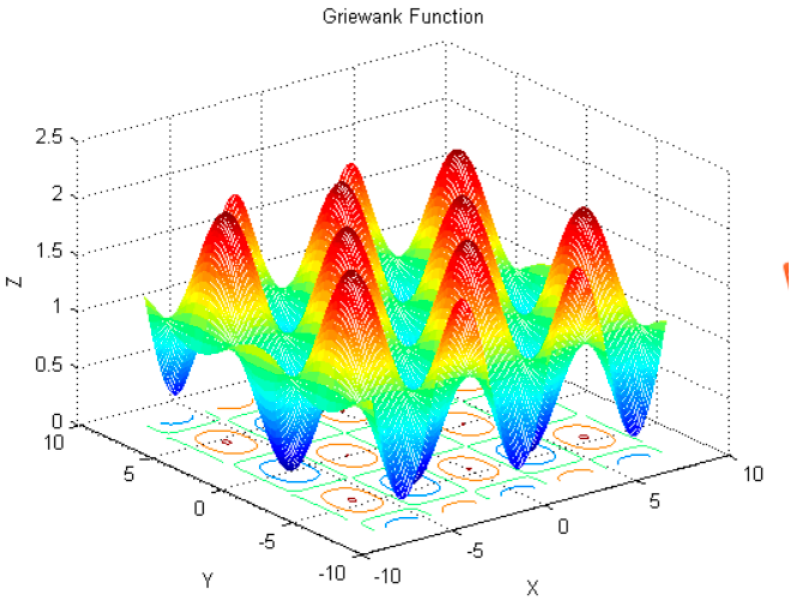
Pro optimalizační účely jsou argumenty funkce omezeny intervalem $[-60, 60]$ a jejím globálním optimem je $f(0, \cdots, 0) = 0$.



Obrázek 4.3: Griewankova funkce, vizualizace pro 2D.



Obrázek 4.4: Griewankova funkce, pohled na rovinu x_1,x_2.



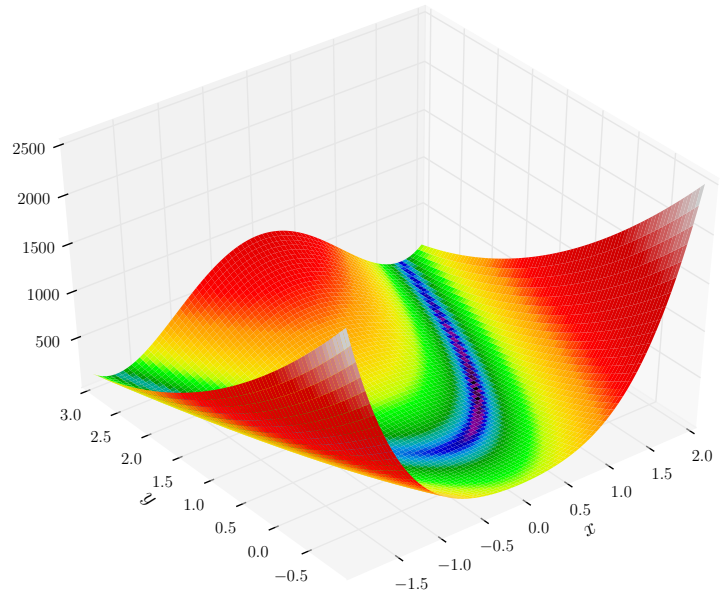
Obrázek 4.5: Griewankova funkce, náhled na okolí optima.

4.2.3 Rosenbrockova funkce

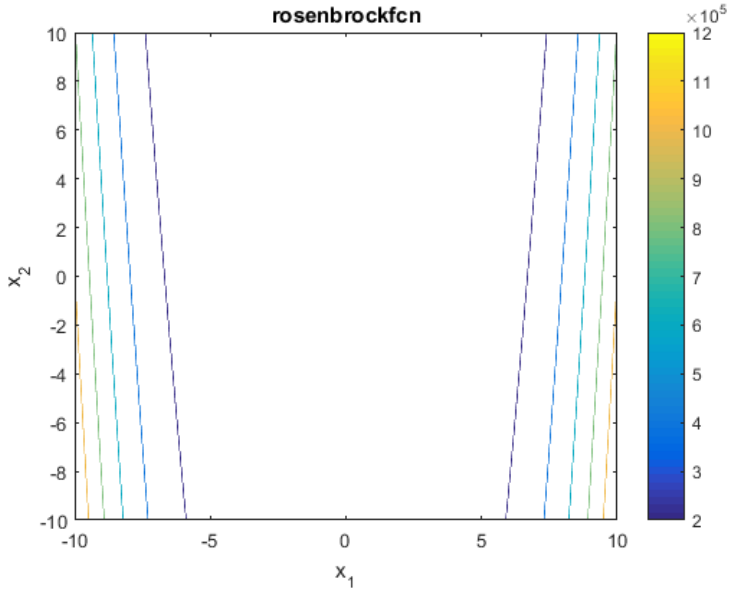
Poslední v testovací sadě je *Rosenbrockova funkce*, označována také za „banánovou“ funkci. Jedná se o jednu z nejpoužívanějších funkcí pro testování optimalizačních algoritmů. Funkce je unimodální, její globální minimum leží v úzkém, parabolickém údolí. Toto údolí je většinou jednoduché najít, funkce testuje spíše schopnost metod konvergovat k minimu. Vizualizace ve dvou rozměrném prostoru je možné vidět na obrázcích 4.6 a 4.7. Funkční předpis:

$$f(\vec{x}) = \sum_{i=1}^{d-1} [100 * (x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{4.3}$$

Pro optimalizační účely jsou argumenty funkce omezeny intervalem $[-5, 10]$ a jejím globálním optimem je $f(1, \cdots, 1) = 0$.



Obrázek 4.6: Rosenbrockova funkce, vizualizace pro 2D.



Obrázek 4.7: Rosenbrockova funkce, pohled na rovinu x_1,x_2.

4.3 Testování

Následující sekce prezentuje výsledky optimalizace představené testovací sady. Nejprve je sada použita pro validaci jednotlivých implementovaných metod a následně jsou experimenty provedeny znovu, tentokrát s ohledem na vybraná kritéria. V poslední části jsou diskutovány výsledky těchto omezených běhů a přínos pro cíl této práce.

4.3.1 Parametry spuštění

V předešle kapitole bylo nastíněno, že jedním z hlavních problémů, které je třeba vyřešit při použití evolučních metod, je vhodné nastavení řídicích parametrů. K tomu je třeba znalost metody i konkrétního optimalizovaného problému. V případech, kdy je problém typu *black-box* se dostáváme do situace, ve které je třeba experimentovat a parametry určit empiricky nebo za pomoci meta-optimalizačních technik. Tato technika popisuje využít další, nadřazené optimalizační metody (jsou typicky voleny ty, které nejsou na řídicí parametry příliš citlivé) pro nalezení vhodných parametrů. Tento přístup má svou stinnou stránku - počet vyhodnocení fitness funkce roste exponenciálně. Tohoto přístupu si ovšem práce nemůže dovolit. Vyhodnocení simulace šíření tepla je velice výpočetně náročné a extenzivně experimentovat, či využít jiných strojových metodik pro nalezení řídicích parametrů si nemůžeme dovolit. Z tohoto důvodu si tato sekce klade za cíl i nalezení jisté vhodné *redukce* tohoto modelu na některou (či kombinaci některých) z testovacích funkcí. Odpovídající funkci můžeme považovat za formu aproximace a úvodní řídicí parametry nastavit podle poznatků o této metodě.

4.3.2 Kritéria hodnocení

Snahou práce je ovšem nalézt vhodný algoritmus pro optimalizaci systému HIFU (prezentován v kapitole 2). Pro tento cíl je nutné navrhnout kritéria, podle kterých budeme algoritmy posuzovat. Jak již bylo nastíněno, tyto kritéria musí vyjadřovat poměr mezi počtem evaluací simulace šíření tepla ve tkáních a přesností nalezení trajektorie. Tyto kritéria jsou reprezentována zavedeným omezením optimalizace na maximálně 2000 vyhodnocení účelové funkce a tím i evaluací simulace (obecně jedna simulace se skládá z provedení n sonikací). Toto omezení se na první pohled může zdát relativně přísné, ovšem je třeba zvážit situaci. Pacient čekající na odstranění zhoubného nádoru si nemůže dovolit dlouho čekat, samotná simulace je přizpůsobena jeho případu, který se každým dnem vyvíjí. Zhoubná tkáň se den za dnem rozšiřuje, mění tvar nebo i otáčí a i malá změna znehodnotí nalezené výsledky. A přesto, že jednomu vyhodnocení simulace nelze přiřadit časovou hodnotu - je vysoce závislá na počtu sonikací a i parametrech samotné sonikace, je třeba se pokusit o včasnost výsledku. 2000 je čistě empirická hodnota - jedná se o 8 hodin výpočtu 20 sonikací pro relativně průměrné hodnoty dob pálení a čekáni. Výpočty byly navíc prováděny na clusterech *IT4Innovation*, což jsou vysoce výkonné výpočetní zařízení, na kterých je možné uplatnit masivního paralelizmu.

V ideálním případě by pro minimalizaci rizika zranění pacienta hrálo roli i nezbytný počet sonikací. Toto je ovšem problém, jehož řešení se může výrazně lišit model od modelu a jakékoliv nalezení řešení pro jednu specifickou situaci nelze považovat za obecně odpovídající. Cílem je proto nalézt takový algoritmus, který zvládá nízké i vysoké počty sonikací.

4.3.3 Testy

Nejprve byla pro ověření validity nastavena dimenze problému na **5** proměnných bez dodatečných omezení. Výsledky této validace lze vidět v přílohách na obrázcích A.1, A.2 a A.3. Následně byly testy provedeny znovu, tentokrát s omezením na 2000 evaluací účelové funkce a dimenzí problému nastavenou na 16. Tato omezení odpovídají hledání trajektorie o 4 sonikacích. Běžně se při HIFU operacích používají desítky, 4 slouží pouze pro ověření konceptu.

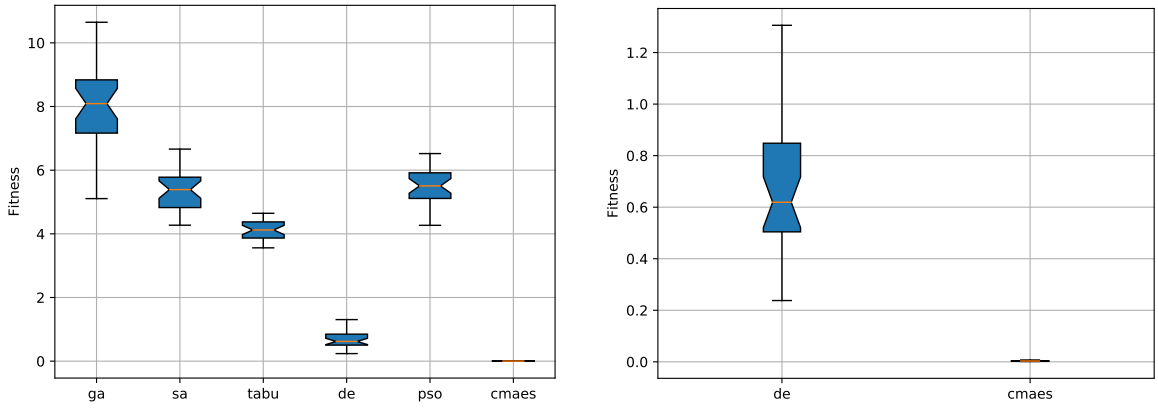
Ackleyho funkce

Následující výsledky jsou agregací 30 běhu každého algoritmu. Parametry spuštění byly následující:

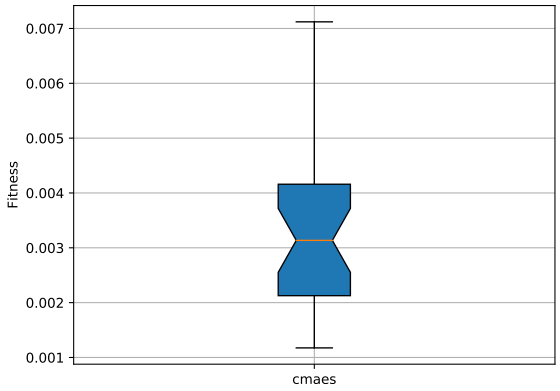
GA	64 jedinců v populaci	Turnajový výběr	Dvoubodové křížení 80%	Mutace 15%	Přežití nejlepších bez ohledu na generaci
SA	Počáteční teplota 2000	Krok 1%	Lineární chladící rozvrh		
Tabu	20 prvků v tabu seznamu	5 kandidátních řešení			
DE	40 jedinců v populaci	Strategie BEST/1/BIN	Šance rekombinace 80%	Faktor zesílení 0.5 až 0.9	
PSO	100 částic	Akcelerační koeficienty c1 = c2 = 2		Koeficient setrvačnosti 0.5	
CMA ES	*Nevyžaduje vstupní parametry.*				

Tabulka 4.1: Řídící parametry jednotlivých optimalizačních metod pro omezenou optimalizaci Ackleyho funkce.

Výsledky je možné vidět na grafech 4.8 a 4.9. Je patrné, že až na *CMA-ES* nebyl žádný z algoritmů schopen dosáhnout skutečného optima. Zastavili se v jednom z mnohých lokálních extrémů při strmém sestupu a nedokázali jej překonat. Nejhůře dopadl genetický algoritmus, což není příliš překvapující; 2000 vyhodnocení účelové funkce je obecně pro populačně založené algoritmy málo. Genetické algoritmy typicky vyžadují mnohonásobně vyšší počet vyhodnocení k dosažení optima. Z grafu posledních generací (přílohy, graf A.5) je možné vidět, že populace si nebyly schopny udržet diverzitu. Za zmínku stojí také metoda diferenciální evoluce, která se zdá být zdravá (grafy A.13) a jednoduše nestihla dokončit evoluci.



Obrázek 4.8: Výsledky omezené optimalizace Ackleyho funkce. Vlevo všechny algoritmy, vpravo detail na diferenciální evoluci a CMA-ES.



Obrázek 4.9: Detail na výsledky algoritmu CMA-ES omezené optimalizace Ackleyho funkce.

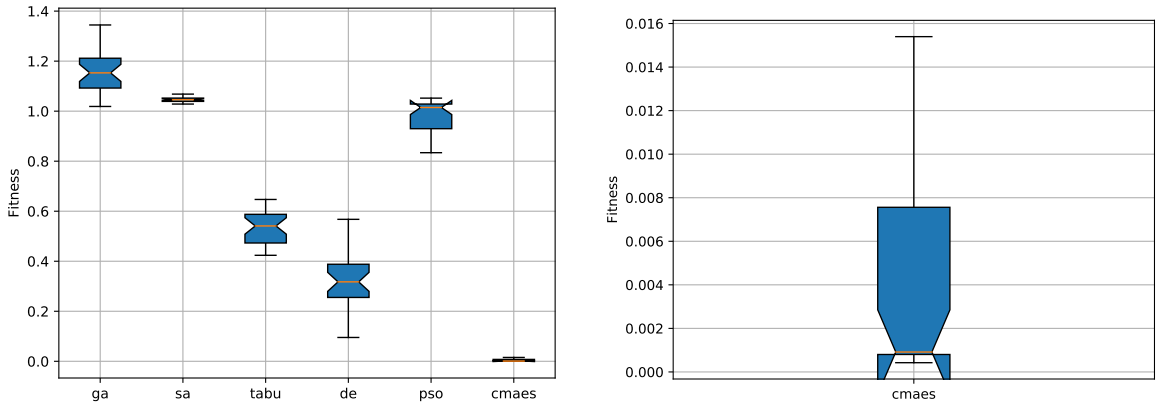
Griwankova funkce

Následující výsledky jsou agregací 30 běhu každého algoritmu. Parametry spuštění byly následující:

GA	50 jedinců v populaci	Turnajový výběr	Dvoubodové křížení 80%	Mutace 20%	Přežití nejlepších bez ohledu na generaci
SA	Počáteční teplota 2000	Krok 1%	Lineární chladící rozvrh		
Tabu	20 prvků v tabu seznamu	5 kandidátních řešení			
DE	40 jedinců v populaci	Strategie RAND-TO-BEST/1/BIN	Šance rekombinace 75%	Faktor zesílení 0.5	
PSO	75 částic	Akcelerační koeficienty c1 = c2 = 2		Koeficient setrvačnosti 0.5	
CMA ES	*Nevyžaduje vstupní parametry.*				

Tabulka 4.2: Řídící parametry jednotlivých optimalizačních metod pro omezenou optimalizaci Griewankovi funkce.

Výsledky je možné vidět na grafech 4.10. Funkce testuje globálnost metod a bylo tedy třeba podpořit tuto vlastnost vhodnými řídícími parametry algoritmů. Bohužel, i zde se omezení na 2000 vyhodnocení ukázalo těžké. Diferenciální evoluce ani optimalizace rojem částic nestihli dokončit evoluci. Genetický algoritmus zdegeneroval, pravděpodobně nedostatečnou diverzitou způsobenou malou populací.



Obrázek 4.10: Výsledky omezené optimalizace Griewankovi funkce. Vlevo výsledky všech algoritmů, vpravo detail na CMA-ES.

Rosenbrockova funkce

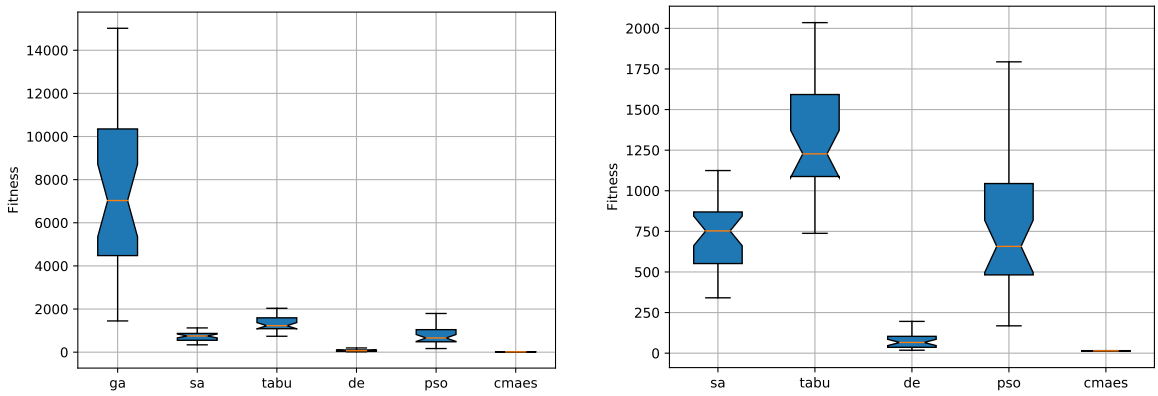
Následující výsledky jsou agregací 30 běhu každého algoritmu. Parametry spuštění byly následující:

GA	64 jedinců v populaci	Turnajový výběr	Dvoubodové křížení 80%	Mutace 5%	Přežití nejlepších bez ohledu na generaci
SA	Počáteční teplota 2000	Krok 1%	Lineární chladící rozvrh		
Tabu	20 prvků v tabu seznamu	5 kandidátních řešení			
DE	30 jedinců v populaci	Strategie BEST/1/BIN	Šance rekombinace 80%	Faktor zesílení 0.5 až 0.7	
PSO	75 částic	Akcelerační koeficienty c1 = c2 = 2		Koeficient setrvačnosti 0.5	
CMA ES	*Nevyžaduje vstupní parametry.*				

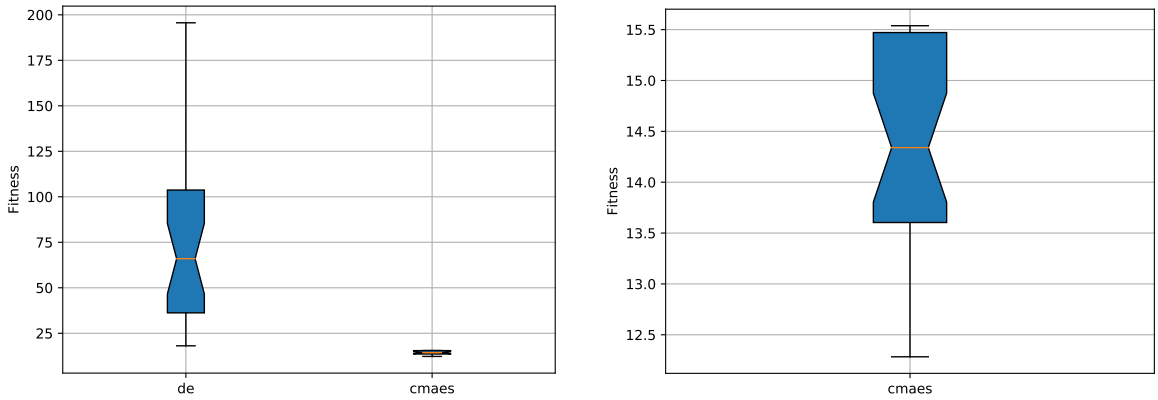
Výsledky je možné vidět na grafech 4.11, 4.11 a 4.12. Optimum rosenbrockovi funkce leží v hlubokém údolí, kde každý krok směrem k optimu výrazně vylepšuje fitness. Na tomto testu ukázali svou sílu algoritmy využívající informace o trasách či směru - diferenciální evoluce, pso a cma-es. Na druhou stranu je z výsledků na první pohled patrné, že genetický algoritmus neuspěl. I zde se ukázala jako problém diverzita (grafy A.37, přestože jiným způsobem, než v předchozích případech. Algoritmus neuvázl v lokálním extrému; nedokázal dostatečně konvergovat, protože populace neobsahovala lepší jedince. Tabu prohledávání i simulované žíhání také nebyly schopný cíle dosáhnout, pravděpodobně z důvodů špatně nastavené délky kroku. Optimalizace rojem částic a diferenciální evoluce opět nedokázali dokončit evoluci - v momentě ukončení optimalizace stále vylepšovali svá řešení (patrné z grafů A.48 a A.45).

Bohužel, i zde se omezení na 2000 vyhodnocení ukázalo těžké. Diferenciální evoluce ani optimalizace rojem částic nestihli dokončit evoluci. Genetický algoritmus zdegeneroval, pravděpodobně nedostatečnou diverzitou způsobenou malou populací. Větší populace vy-

žadují více vyhodnocení a tím zase méně generací k vylepšení řešení. Z tohoto důvodu byla i nastavena nezvykle vysoká pravděpodobnost mutace; jako snaha o podporu diverzity.



Obrázek 4.11: Výsledky omezené optimalizace Rosenbrockovi funkce. Vlevo všechny algoritmy, vpravo vynechán genetický algoritmus.



Obrázek 4.12: Výsledky omezené optimalizace Rosenbrockovi funkce. Vlevo detail na diferenciální evoluci a CMA-ES, vpravo CMA-ES samotná.

4.3.4 Zhodnocení

Testovací sada funkcí nebyla vybrána náhodně, každá vystavuje optimalizační metody jiným úskalím. Od konce:

- **Rosenbrockova funkce** testuje exploitační schopnosti metod - jak dobře dokáží konvergovat k optimu.
- **Griewankova funkce** testovala explorační schopnost algoritmu. Funkce připomíná šum a algoritmus musel překonat množství lokálních optim, než se dostane ke globálnímu.
- **Ackleyho funkce** - první testovaná funkce měla za úkol otestovat obecnou schopnost optimalizačních metod. K její optimalizaci je zapotřebí obou vlastností, exploraci i exploitace.

Cílem ovšem nebylo najít nejlepší obecný optimalizační algoritmus ale takovou evoluční metodu, která má největší potenciál řešit úlohu představenou v kapitole 2 - plánování HIFU trajektorie. Platí tzv. „No free lunch“ teorém.

No free lunch teorém

Teorém definovaný matematikem *David H. Wolpert*. Všechny algoritmy, které hledají extrém účelové funkce, mají stejnou obecnou výkonost - pokud bychom zprůměrovaly jejich výsledky pro všechny možné účelové funkce, nenašli bychom rozdíl. Pokud je jeden algoritmus výkonnější nad množinou funkcí A, jiný bude výkonnější nad množinou funkcí B, která má stejnou kardinalitu.

Závěry

Z výsledků 4.3.3, 4.3.3 a 4.3.3 je patrné, že zavedená omezení a zvýšení dimenze problému jsou těžce překonatelné nástrahy. Genetický algoritmus trpí na problémy s diverzitou populace, diferenciální evoluce a optimalizace rojem částic nedokáží dostatečně rychle konvergovat. Simulované žíhání i tabu prohledávání naopak nepoužívají populace řešení, nýbrž stále vylepšují své jedno nalezené. To jim dává jistou výhodu při omezeném množství vyhodnocení účelové funkce, ovšem i přes to nejsou díky povaze generování následujících stavů schopny konvergovat k úzkému středu a nebo překonat lokální extrém v pozdějších fázích optimalizace. Jako nejlepší algoritmus při řešení všech optimalizačních testů se ukázala metoda CMA-ES, jejíž flexibilní generování kandidátních řešení na základě předchozí evoluční cesty umožňuje dobrou konvergenci i škálovatelnost s velikostí dimenze.

Transformace problému

Vhodný algoritmus pro řešení testovací množiny ovšem ještě nezaručuje jeho schopnost řešit i problém hledání trajektorie sonikací, stále platí „No free lunch“ teorém. Kvůli této skutečnosti je třeba zjistit, jestli se tento problém alespoň částečně nepodobá nějakému z testovací sady.

Jak bylo popsáno v kapitole 2, systém HIFU je heterogenní systém, ve kterém se vyskytuje šum. Také bylo zmíněno, že cévy v blízkosti cíleného místa odvádějí energii. Není příliš těžké si představit transformaci tohoto problému - pokud budeme tyto cévy považovat za lokální minima a šum a ne-homogennost za jisté „spády“ na povrchu takovéto funkce. Dále je přitěžující skutečnost, že hledané proměnné jsou na sobě závislé, a díky povaze kódování a průběhu simulace - sonikace jsou prováděny sekvenčně za sebou a pořadí těchto sonikací je důležité jen do jisté míry - lze prohlásit, že každé řešení problému má i svou variantu, která se liší pouze pořadím provedení. Sonikace jsou na sobě také velice závislé, malá změna v parametrech jedné sonikace může mít za následek velké skoky v hodnotách fitness - sonikace pokrývají kruhovou oblast, kterou je jednoduché přesáhnout okraje nebo například způsobit, že jiná sonikace, která doposud vylepšovala fitness, je najednou zbytečná. To vše má za následek obtížnou konvergenci k optimu. V poslední řadě jsou kraje stavového prostoru penalizovány (přesah na zdravou tkáň je nežádoucí). Tento popis není příliš podobný žádné z funkcí. Testuje schopnost prohledávat i konvergovat ke skutečně optimální kombinaci. Těžká konvergence odpovídá nejlépe funkci **Rosenbrock**, zatímco případné lokální extrémy a strmý spád funkci odpovídá testu **Ackley**.

Kapitola 5

Model šíření tepla ve tkáních

Tato kapitola naváže na popis z kapitoly 2 a více představí implementovaný model. Zdrojový článek [20] pracoval s implementací v prostředí MATLAB, která byla později přepsána do formy *mex* funkce pro urychlení výpočtu za pomoci paralelizačních technik - *multi-threadingu* a vektorizace za pomoci *OMP*. Toto řešení bylo pro potřeby mé práce dále upraveno takto:

- Definovaná funkce byla základem pro výpočetní model v jádru účelové funkce. Pro potřeby této práce byl model zbaven *mex* a tím i *MATLAB* rozhraní, a upraven pro použití v jazyce C++.
- Přidáno načítání dat média ze souborů. Jedná se o jednoduché čtení matice čísel z několika souborů, kde každý soubor reprezentuje jinou relevantní informaci o médii, jako například hustotu či tepelnou vodivost.
- Implementováno podpůrné jádro pro práci s maticemi, které slouží pro definice cílové a penalizační mapy. Za tímto účelem vzniklo několik výpočetních funkcí a obal, jehož vstupem jsou soubory `prohibitedMapCircles.dat` a `targetMapCircles.dat`, které definují své mapy ve formě libovolného množství geometrických kruhů. Tyto kruhy jsou popsány čtveřicí (X, Y, R, V), kde X a Y jsou souřadnice středu, R je poloměrem a V je hodnotou váhy. Hodnoty jsou odděleny čárkou a jednotlivé kruhy odděleny novým řádkem. Váha kruhu reprezentuje hodnotu penalizace či závažnost cíle. Modul postupně tyto kruhy vykreslí do mřížky o velikosti média a vyplní právě zadanou hodnotou. V případě, že se kruhy překryjí, se hodnoty na těchto místech:
  - při definici penalizační mapy sečtou
  - při definici cíle je nad těmito hodnotami provedena funkce $max(...)$

  Tím je možné vytvořit penalizační i cílenou mapu nad libovolným médiem. Pro takto vytvořené mapy vznikly funkce umožňující jejich sečtení, produkt, či práhování hodnot aplikací filtru.
- Z *MATLAB* implementace přepsáno generování matice přidané tepelné energie. Tato matice reprezentuje jednu sonikaci a její tvar je aproximací za pomoci Gaussova rozložení ve $2D$ prostoru. Takto definované sonikace jsou následně vstupem samotné simulace.
- V poslední řadě byla proveden analýza výpočetního výkonu samotné simulace za pomoci nástrojů *Intel VTune*^{TM} *Profiler*. Na základě těchto výsledků byly přepracovány

již existující paralelizační techniky na modernější a čitelnější variantu direktiv *OMP* pro překladač *Intel Compiler 2019a*.

Tyto části, spolu se vstupním bodem splňujícím rozhraní `FitnessFunction.h` z kapitoly 4, vytváří optimalizovanou účelovou funkci.

5.1 Data

Pro spuštění simulace sonikací jsou potřeba dva druhy dat. Za prvé data definující médium, ve kterém se teplo šíří, a za druhé segmentová mapa D, která každému bodu média přiřazuje hodnotu. Ta je po dokončení simulace použita pro výpočet fitness.

5.1.1 Médium

Médium, se kterým model pracuje, je reprezentováno jako soustava několika matic $X \times Y$ a hodnoty dx, která ukazuje skutečnou vzdálenost (v mm) mezi jednotlivými body matice:

- matice teploty T - momentální teplota v C v jednotlivých bodech média.
- matice hustoty ρ - fyzikální hustota látek v médiu (v $kg * m^{-3}$).
- matice měrné tepelné kapacity C - množství tepelné energie potřebné k ohřátí $1kg$ látky o jeden teplotní stupeň ($J * kg^{-1} * K^{-1}$).

Jako médium byl využit open-source voxelový model *Austin Woman* [13]. Jedná se o dataset pořízený jako součást projektu *Visible Human* ze *U.S. National Library of Mediciny*. Model pracuje s reprezentací tohoto média v prostoru 495×495 a $dx = 0.2e - 3$. Tyto data jsou umístěna ve vlastních souborech pro případnou snadnou úpravu či náhradu za jiný dataset.

5.1.2 Omezení

Jak již bylo zmíněno v předchozí kapitole, z důvodu výpočetní náročnosti bylo omezeno maximální počet vyhodnocení kompletní simulace na 2000[1] na jeden běh optimalizační metody. Kompletní simulací je myšleno provedení všech n sonikací obsažených v chromozomu (tedy dohromady $2000 * n$ výpočtů šíření tepla). Zároveň jsou pro každý z vybraných problémů definována globální omezení tohoto typu:

- Stavový prostor pro souřadnice sonikace byl omezen na hodnoty $x_l < x(i) < x_h$ a $y_l < y(i) < y_h$.
- Stavový prostor proměnných určujících dobu záření a chlazení byl omezen následovně : $0 < t_{on}$ a $0 < t_{off}$.
- Stavový prostor každé proměnné byl spojen do tvaru toroidu - překročením hranice se metoda dostává na druhou stranu.
- Optimalizace je prohlášena za dokončenou v případě, kdy $f(I) < 10$ (fitness na začátku optimalizce se pohybuje v řádech $10^3 - 10^4$ a vzhledem k rasterizaci problému, nepřesnosti čísel s plovoucí desetinou čárkou a použitým aproximacím, lze takto nízkou fitness považovat za optimum), nebo dosažením zmíněné hodnoty 2000 fitness vyhodnocení.

[1] Hodnota nebyla kontrolována před každým provedením simulace, nýbrž na začátku generací. Následkem je, že populační metody mohou tuto hodnotu přesáhnout pro dokončení generace

5.2 Průběh simulace

V momentě, kdy jsou k dispozici data popisující médium, segmentová mapa i vektor sonikací, může začít simulace. Pro každou sonikaci se vytvoří matice přidaného tepla Q. Tato matice (opět $X \times Y$) je vygenerována za pomoci dvourozměrného normálního rozložení se středem v bodě matice $Q_{x(i)y(i)}$[2]. Takto vytvořené matice se sekvenčně aplikují na model média tak, že pro každou sonikaci je spuštěna simulace šíření tepla v médii a spočteno kumulované teplo - prostorová termální mapa $CEM43$. Na základě této mapy je poté provedeno práhování, které označí všechny body média, které byly zničeny termální ablací. Práhování, resp. výsledná maska, je popsána vztahem 5.1. Rovnice převzaty ze [20].

$$C_{ij} = \begin{cases} 0 & \text{pokud } CEM43_{ij} \leq 240 \\ 1 & \text{jinak} \end{cases} \tag{5.1}$$

Fitness funkce je poté vypočtena vztahem 5.2:

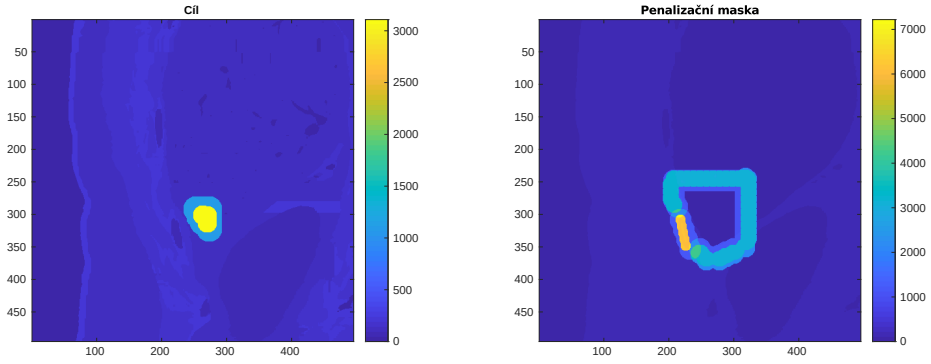
$$f = \int_0^X \int_0^Y (D * \overline{C}) + (D * C))dxdy \tag{5.2}$$

5.3 Testované útvary

V následující sekci jsou popsány dva vybrané útvary pro pokus o optimalizaci trajektorie. Útvary představují dva typické modely cílené tkáně při praktickém použití. Tyto útvary byly pojmenovány jako „skvrna“ a „květina“, podle tvarů, které připomínají autorovi. Obě tyto mapy byly vytvořeny na médiu z *AustinWoman*.

5.3.1 Skrvna

Cíl, označený jako skvrna, je monolitická cílová oblast definována v rozmezích $272 < x(i) < 342$ a $233 < y(i) < 293$. Spolu s penalizační mapou je tento model zobrazen na obrázku 5.1. Jedná se o citelně jednodušší z obou testů, především z důvodu pozitivního efektu akumulovaného tepla ve tkáních. Ablace středu tohoto útvaru má prioritu před okraji. Útvar tohoto typu si není příliš těžké představit i v reálném světě.

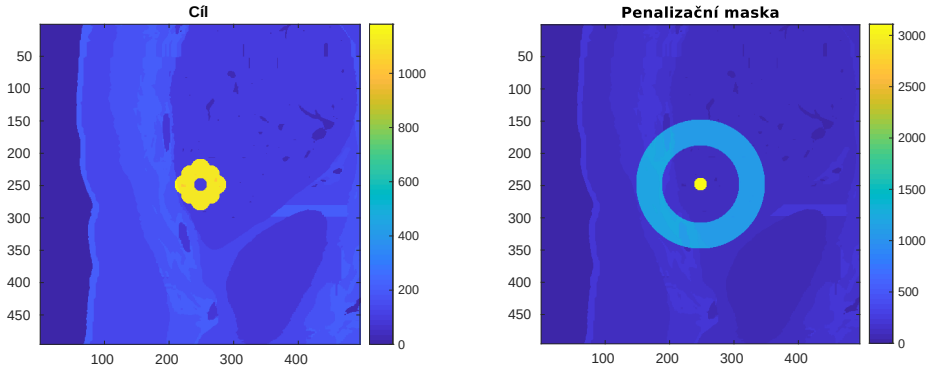


Obrázek 5.1: Diskrétní barevná mapa útvaru skvrna pro testování optimalizace trajektorie HIFU sonikací. Vlevo mapa definující cíl, který je třeba vypálit. Vpravo mapa ukazující penalizovanou oblast, kterou je třeba uchránit.

[2]Tato distribuce je aproximací skutečnosti, jelikož modely schopny precizní simulace jsou výpočetně náročné a přidávat je do již dost výpočetně náročné simulace šíření by způsobilo rapidní zpomalení výpočtu

5.3.2 Květina

Cíl, označený jako květina, je pravidelný, symterický útvar, v jehož středu se nachází zdravá tkáň, kterou je nutno ochránit. Celá cílová oblast je definována v rozmezích $208 < x(i) < 288$ a $208 < y(i) < 288$. Spolu s penalizační mapou je tento model zobrazen na obrázku 5.2. Z experimentů se ukazuje, že se jedná o těžší z modelů, především kvůli efektu akumulovaného tepla právě ve zdravém středu objektu. Prováděné sonikace se překrývají a v okolí středu postupně akumulují teplo. Tím vzniká kladná zpětná vazba a vysoce se zvyšuje závislost na pořadí provedení řetězu sonikací. Tento jev existuje sice i v případě problému typu skvrna, tam je ovšem pozitivním faktorem, jelikož je tvar ucelený a střed útvaru je cílem. V reálném světě tento problém reprezentuje zhoubnou tkáň, obrostlou například okolo močové trubice.



Obrázek 5.2: Diskrétní barevná mapa útvaru květina pro testování optimalizace trajektorie HIFU sonikací. Vlevo mapa definující cíl, který je třeba vypálit. Vpravo mapa ukazující penalizovanou oblast, kterou je třeba uchránit.

5.4 Výsledky

V následující sekci jsou prezentovány agregované výsledky 15 nezávislých běhů optimalizační metody. V první části je ukázána schopnost algoritmů hledat optimum při malém počtu sonikací (4 sonikace). Tento test je prováděn na problému typu skvrna a není relevantní pro reálné případy, při kterých jsou použity desítky sonikací, nýbrž jako důkaz koncepce použití těchto metod. Následně budou prezentovány výsledky na vyšších počtech sonikací. Skvrna bude vyžita znovu, tentokrát pro hledání trajektorie o 20 sonikacích. Dále pak bude ukázáno 15 sonikací pro oblast typu květina. Tyto hodnoty byly vybrány empiricky s ohledem na daný problém a časovou náročnost výpočtů.

Výpočty byly provedeny na superpočítači *Salomon* v rámci projektu *IT4Innovations*. V součtu, včetně experimentů s parametry, bylo využito **64956 hodin** normovaného procesorového času. U grafů prezentujících poslední generace, případně rozptyl běhu, lze pozorovat výřezy na stranách - tzv. „notche“. Tento notch představuje interval spolehlivosti 95% kolem mediánu. Pokud se notche dvou rozdílných generací *nepřekrývají* (tedy spodní konec výřezu není níže, než svrchní konec jiného), lze s 95% důvěrou tvrdit, že se od sebe liší.

5.4.1 Skvrna - malý počet sonikací

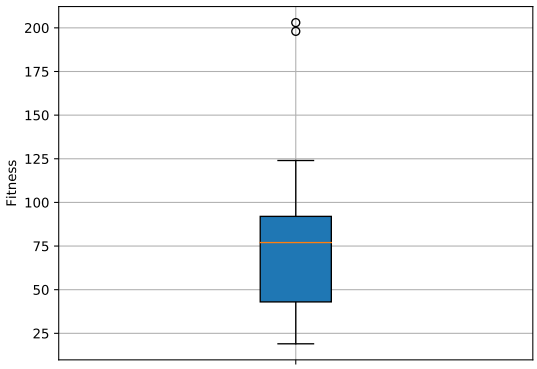
Hledání optimální trajektorie pro HIFU operaci s nízkým počtem sonikací bylo spuštěno s následujícími parametry:

GA	64 jedinců v populaci	Turnajový výběr	Dvoubodové křížení 80%	Mutace 20%	Přežití nejlepších bez ohledu na generaci
SA	Počáteční teplota 2000	Krok 1%	Lineární chladící rozvrh		
Tabu	20 prvků v tabu seznamu	5 kandidátních řešení			
DE	25 jedinců v populaci	Strategie BEST/1/BIN	Šance rekombinace 90%	Faktor zesílení 0.5 až 0.9	
PSO	64 částic	Akcelerační koeficienty c1 = c2 = 2		Koeficient setrvačnosti 0.5	
CMA ES	*Nevyžaduje vstupní parametry.*				

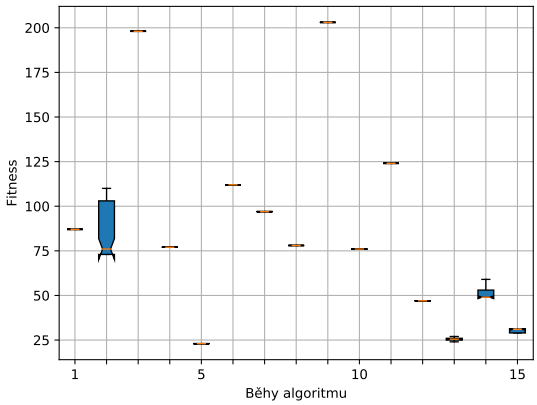
Tabulka 5.1: Tabulka řídících parametrů použitých při optimalizaci modelu skvrna malým počtem sonikací.

Genetický algoritmus

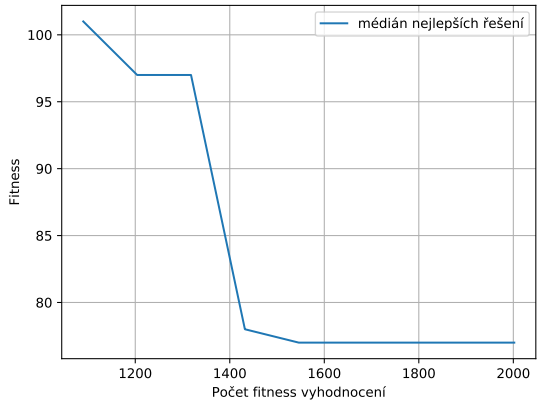
Lze vidět, že genetický algoritmus nebyl příliš úspěšný při hledání optima. Populace většiny běhů degenerovala - graf 5.4 - a ani přes nadmíru vysokou pravděpodobnost mutace se nepodařilo udržet diverzitu. Bohužel, jedním z hlavních prvků pro udržení různorodosti v GA je velikost populace v kontextu dimenze problému. S velikostí populace ovšem rychle roste počet kandidátních řešení a tím i počet evaluací. Zvětšením populace by mohla rozrůst diverzita, algoritmus by ale nemusel mít dostatečný počet evaluací ke konvergenci.



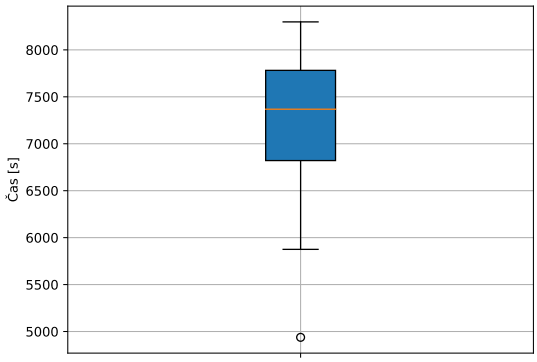
Obrázek 5.3: Boxplot nejlepších výsledků všech 15 běhů GA.



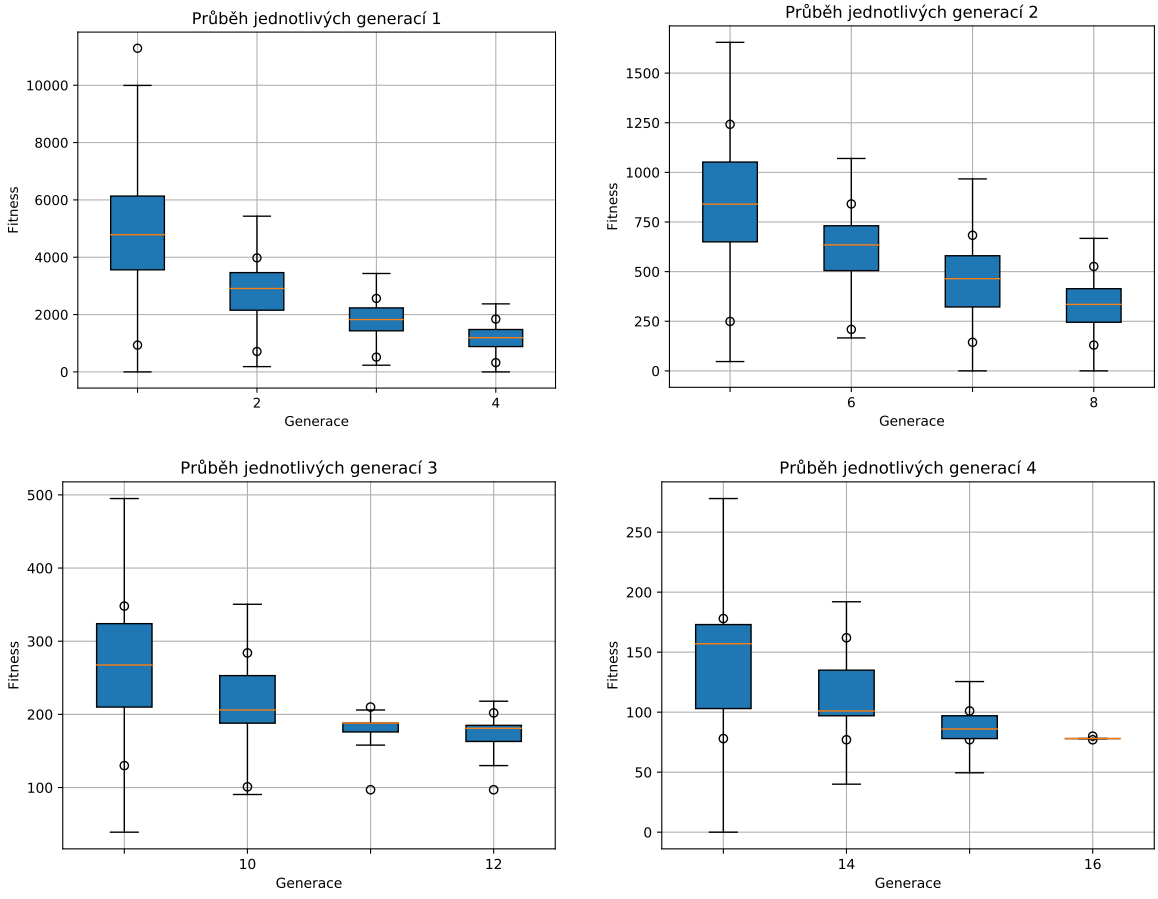
Obrázek 5.4: Boxplot stavu poslední generace všech 15 běhů GA. Lze vidět, že docházelo k degeneraci populace.



Obrázek 5.5: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



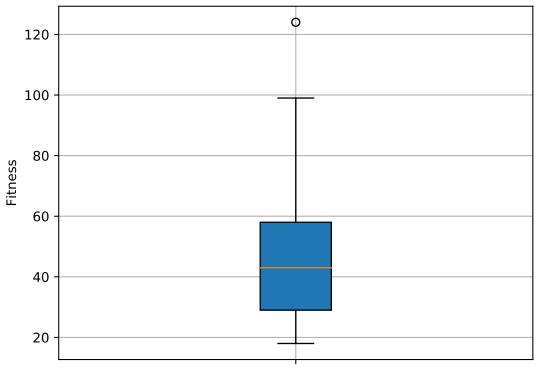
Obrázek 5.6: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($7200s = 2h$)



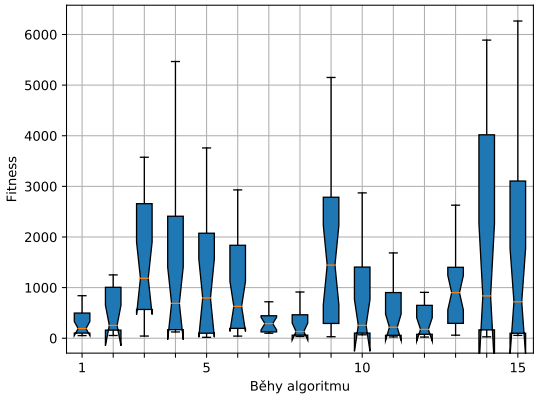
Obrázek 5.7: Evoluční průběh GA. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Simulované žíhání

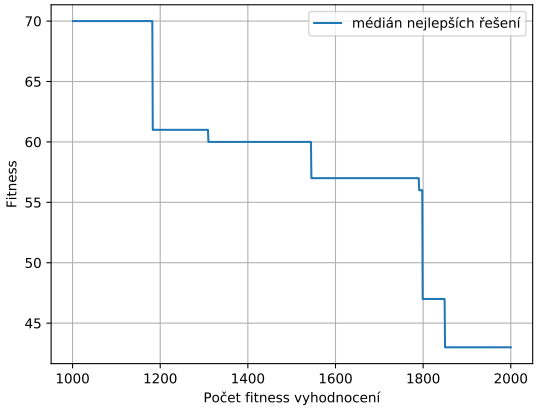
Simulované žíhání bylo překvapivě úspěšné. Vzhledem k tomu, že se nejedná o populační algoritmus nebylo třeba řešit problémy s diverzitou. Vyzdvihněme, že na mediánovou hodnotu 70 se dostalo již za tisíc iterací - 5.10. V případech, kdy by taková fitness byla dostatečná, by se SA dalo považovat za velice efektivní algoritmus pro optimalizaci problému hledání trajektorie.



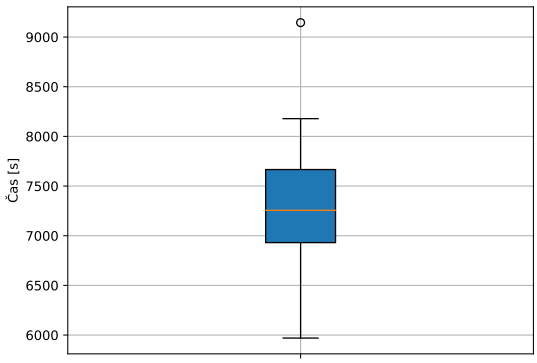
Obrázek 5.8: Boxplot nejlepších výsledků všech 15 běhů SA.



Obrázek 5.9: Boxplot ukazující rozptyl fitness, jaký každý běh SA prohledal.



Obrázek 5.10: Poměr médiánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.

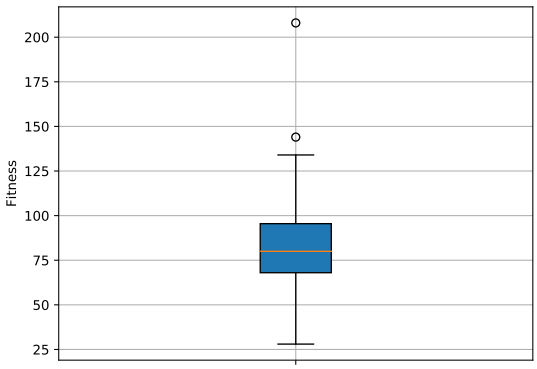


Obrázek 5.11: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($7200s = 2h$)

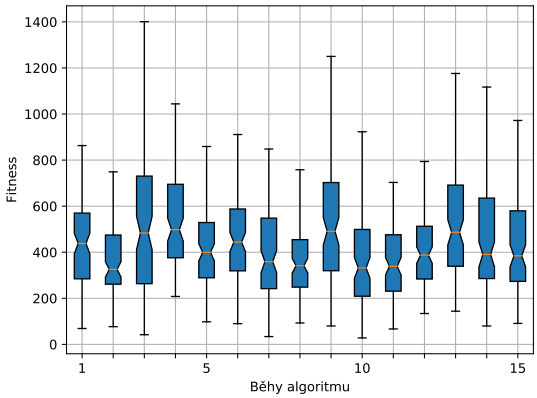
Tabu prohledávání

Výsledky tabu prohledávání se zdají být objektivně horší, než výsledky simulovaného žíhání; jediným dalším zkoumaným optimalizačním algoritmem, který nevyužívá populace. Zdá se, že TS vyžaduje více evaluací a nebo uvázlo v lokálním minimu. Je nutné podotknout, že tato implementace pracuje pouze s dlouhodobým seznamem. Přidání dalších paměťových struktur by pravděpodobně urychlilo konvergenci a vylepšilo i globálnost algoritmu. Na

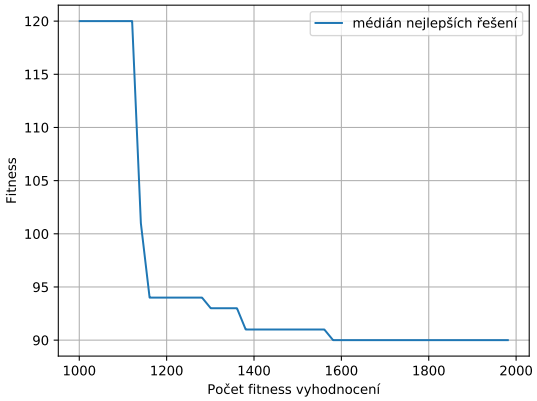
grafu 5.13 lze vidět rozdíl mezi SA a TS - tabu prohledávání nepokrylo ani zdaleka tak velký rozptyl fitness, jako simulované žíhání.



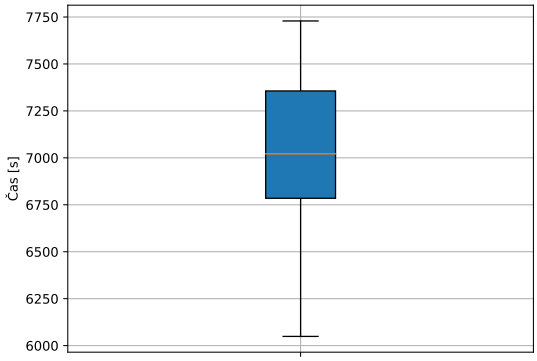
Obrázek 5.12: Boxplot nejlepších výsledků všech 15 běhů TABU.



Obrázek 5.13: Boxplot ukazující rozptyl fitness, jaký každý běh TABU prohledal.



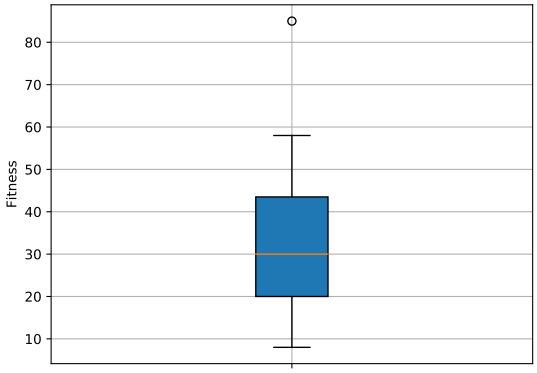
Obrázek 5.14: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



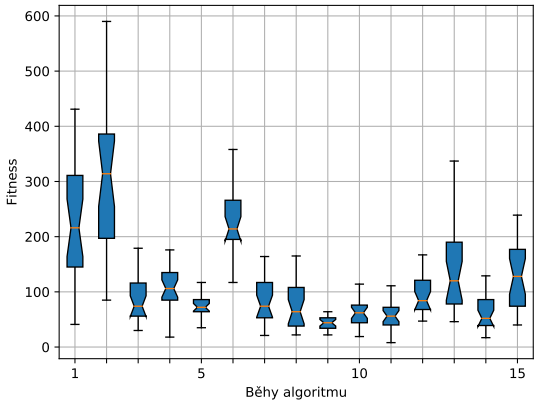
Obrázek 5.15: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($7200s = 2h$)

Diferenciální evoluce

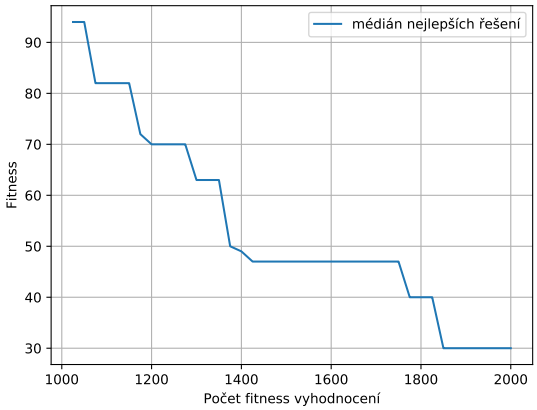
Diferenciální evoluce se zdá být lepší, než simulované žíhání. Mohlo by se zdát, že se zde stále vyskytuje problém, který se objevil při optimalizace testovacích funkcí - algoritmus nedokončil konvergenci. Populace se jeví být po dokončení stále různorodá a docházelo k pravidelnému vylepšení nejlepšího řešení i několik generací před ukončením - viz grafy 5.20. Pokud budeme tento graf studovat detailněji, zjistíme, že je možné pozorovat vliv strategie *best/1/bin* - algoritmus vylepšuje populaci za použití nejlepšího jedince a medián se tedy přibližuje nejlepšímu řešení. Šlo by tedy předpovědět, že by netrvalo příliš mnoho generací, než by populace degenerovala.



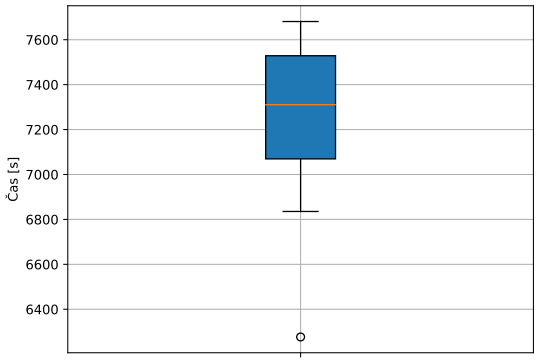
Obrázek 5.16: Boxplot nejlepších výsledků všech 15 běhů DE.



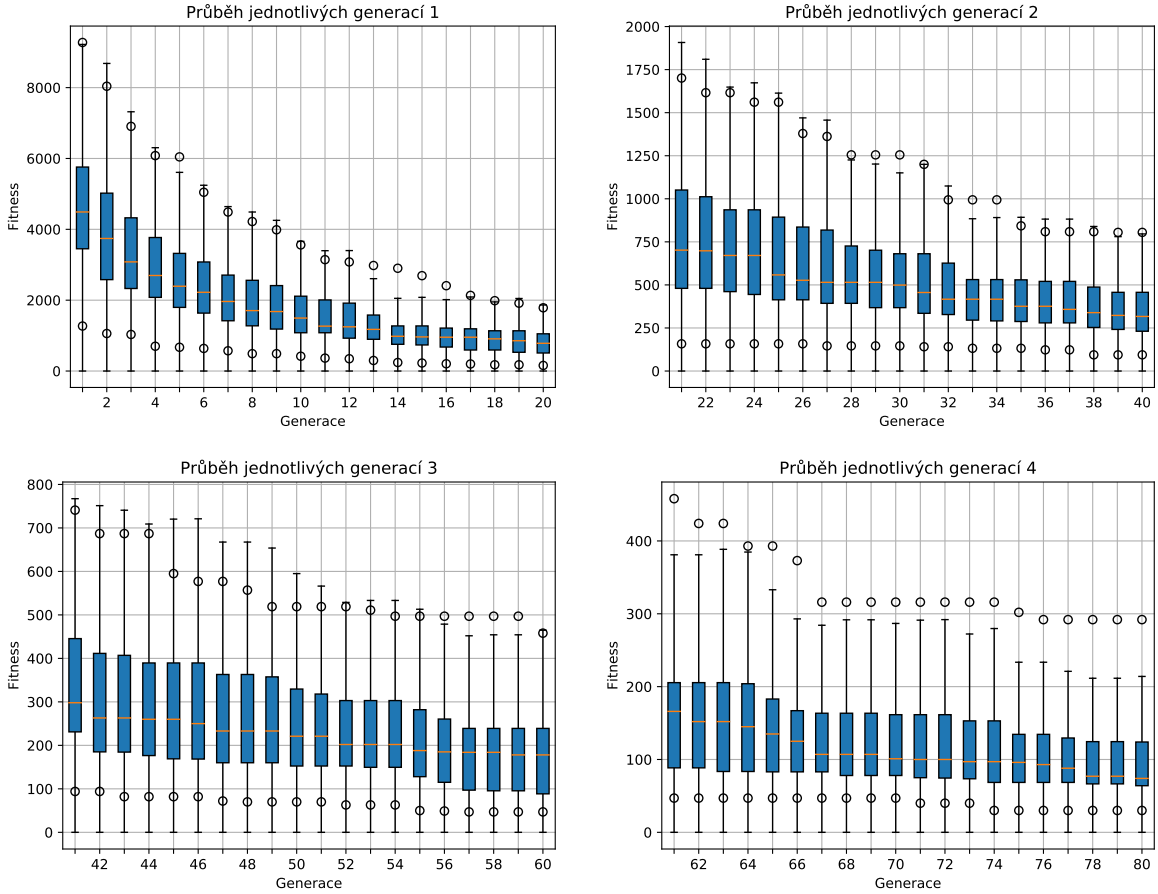
Obrázek 5.17: Boxplot stavu poslední generace všech 15 běhů DE. Zvláštní konce ve tvaru vidlice indikují, že medián je příliš blízko kvartálu. Typicky předzvěst degenerace či indikátor příliš malé velikosti populace.



Obrázek 5.18: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



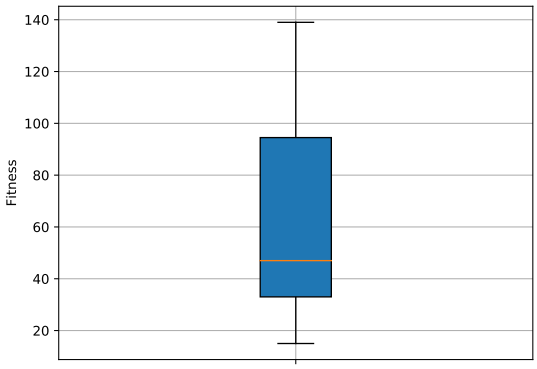
Obrázek 5.19: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($7200s = 2h$)



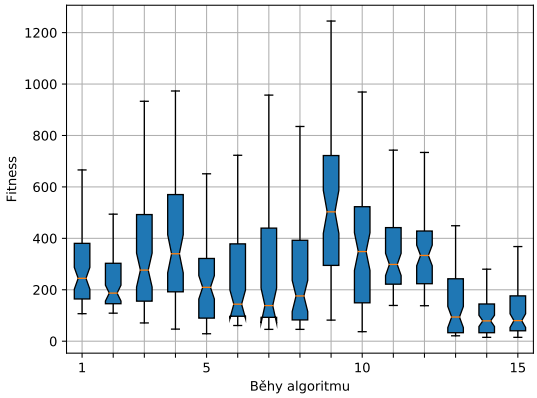
Obrázek 5.20: Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Optimalizace rojem částic

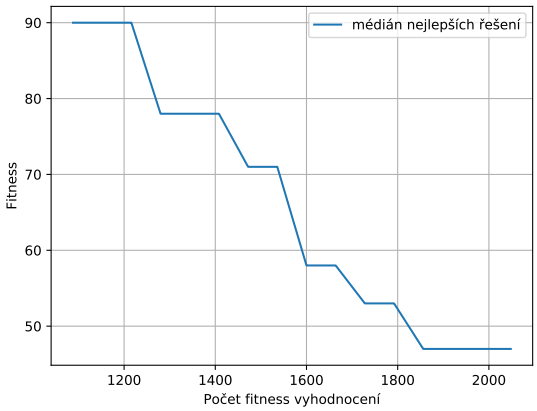
Výkon PSO je dostatečný. Podle grafu poslední generace 5.22 lze soudit, že populace byly dostatečně různorodé a metoda pouze ve většině případů nestihla konvergovat.



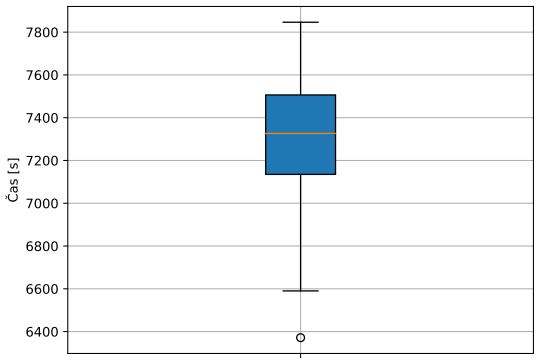
Obrázek 5.21: Boxplot nejlepších výsledků všech 15 běhů PSO.



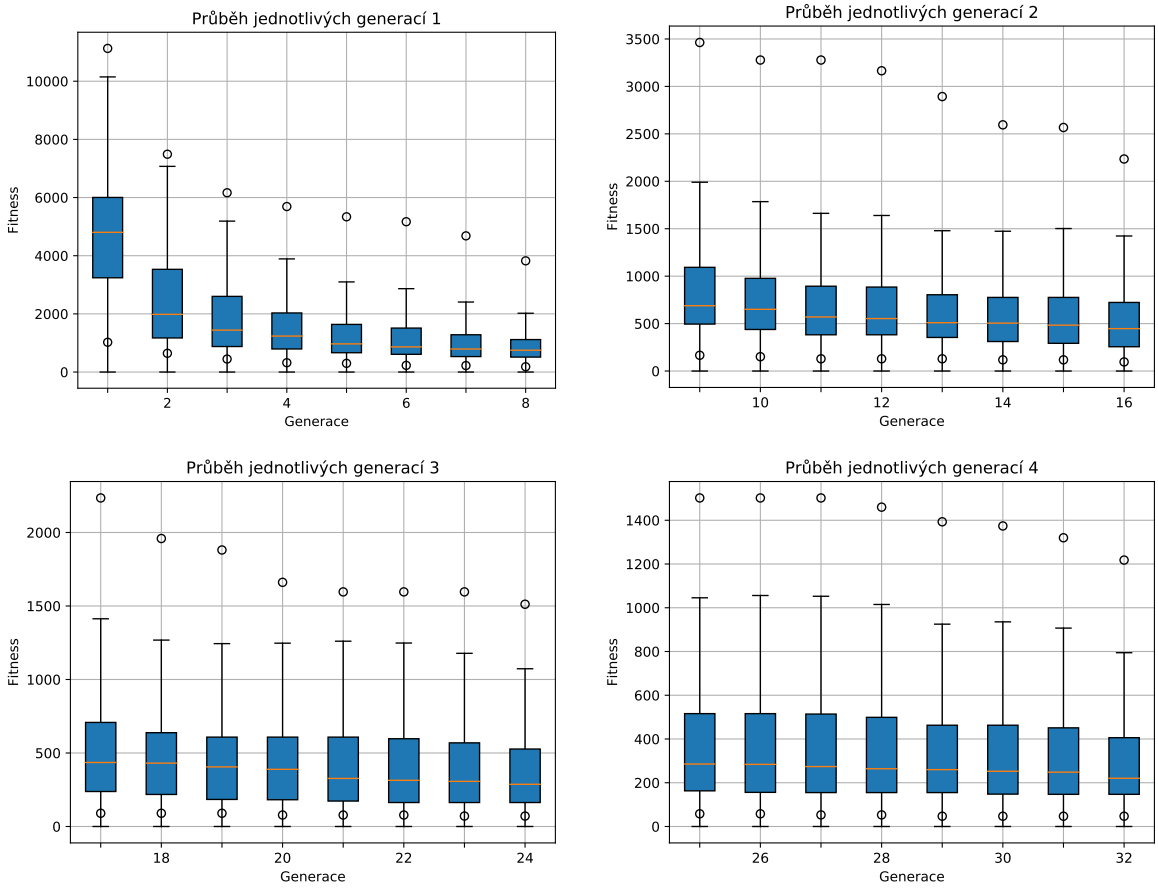
Obrázek 5.22: Boxplot stavu poslední generace všech 15 běhů PSO. Zvláštní konce ve tvaru vidlice indikují, že medián je příliš blízko kvartálu. Typicky předzvěst degenerace či indikátor příliš malé velikosti populace.



Obrázek 5.23: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



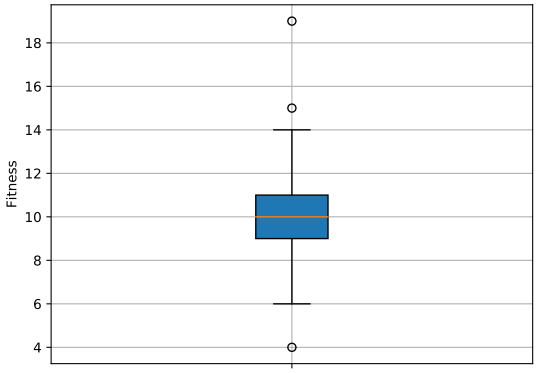
Obrázek 5.24: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($7200s = 2h$)



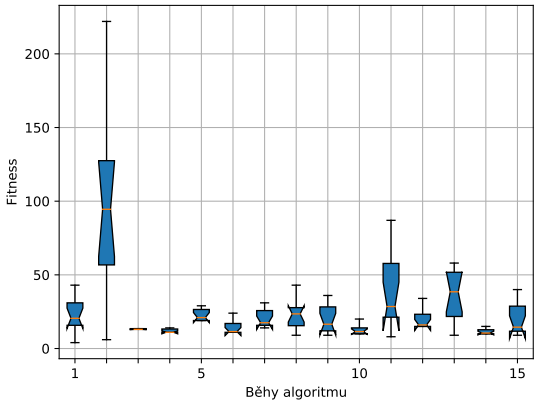
Obrázek 5.25: Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

CMA-ES

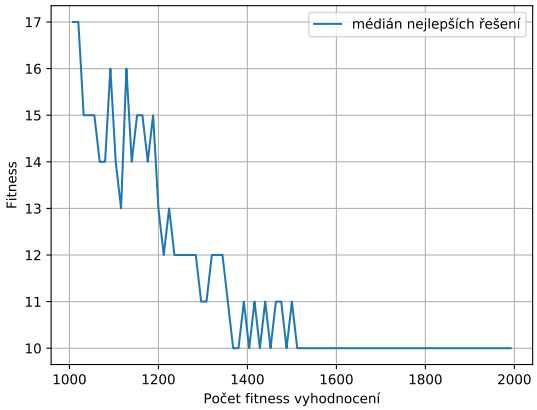
CMA-ES jako jediné dokázalo najít hledané optimum (5.26). Navíc toho docílilo i před využitím všech 2000 fitness evaluací (5.28). Na grafu 5.28 je možné pozorovat ostré skoky způsobenou průměrovou povahu algoritmu. Metoda vytváří nové populace na základě doposud nalezených statistických distribucí pro jednotlivé proměnné, což může při příliš vysoké deviaci způsobit právě tento profil pohybu mediánu populace.



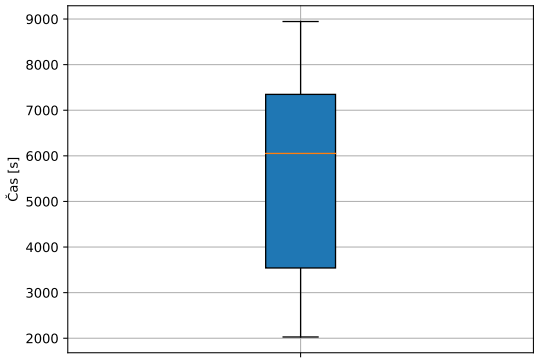
Obrázek 5.26: Boxplot nejlepších výsledků všech 15 běhů CMA-ES.



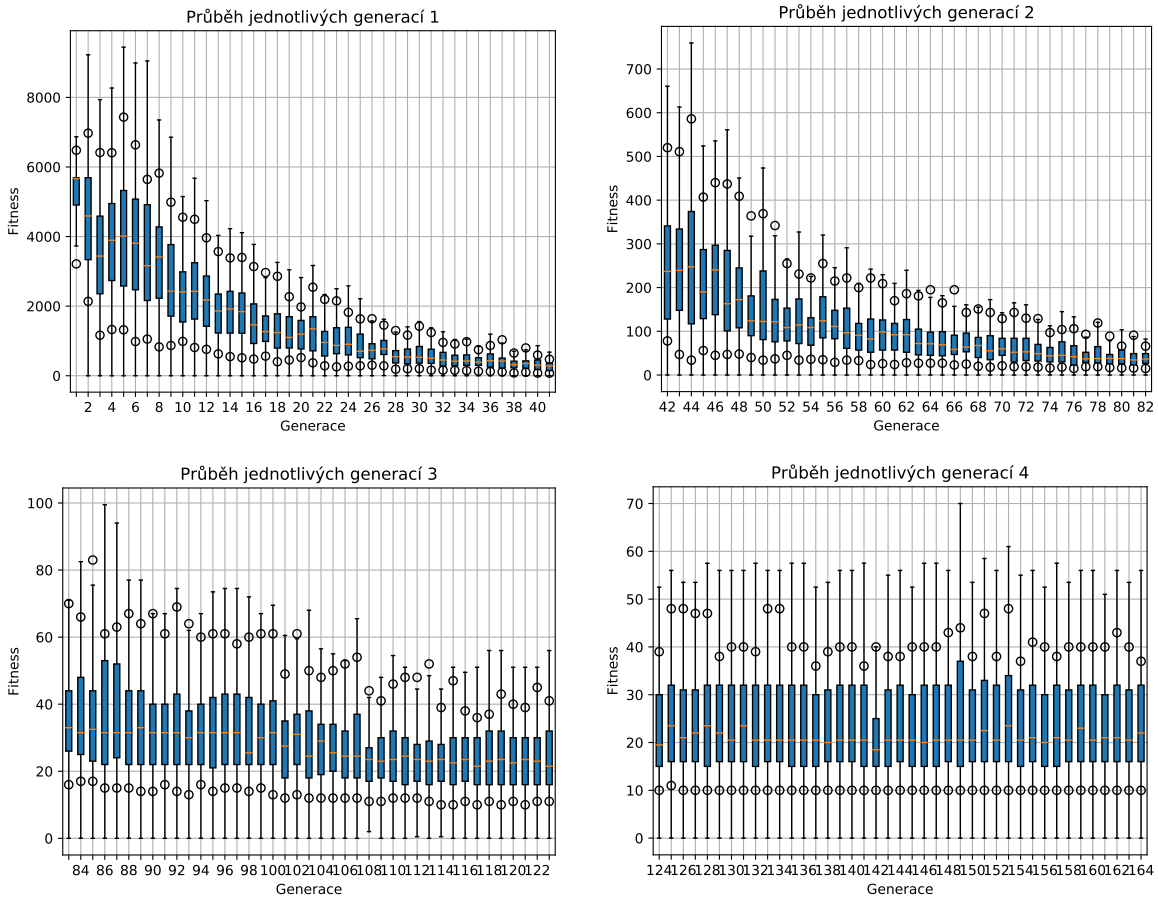
Obrázek 5.27: Boxplot stavu poslední generace všech 15 běhů CMA-ES. Zvláštní konce ve tvaru vidlice indikují, že medián je příliš blízko kvartálu. Typicky předzvěst degenerace či indikátor příliš malé velikosti populace.



Obrázek 5.28: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



Obrázek 5.29: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($7200s = 2h$)

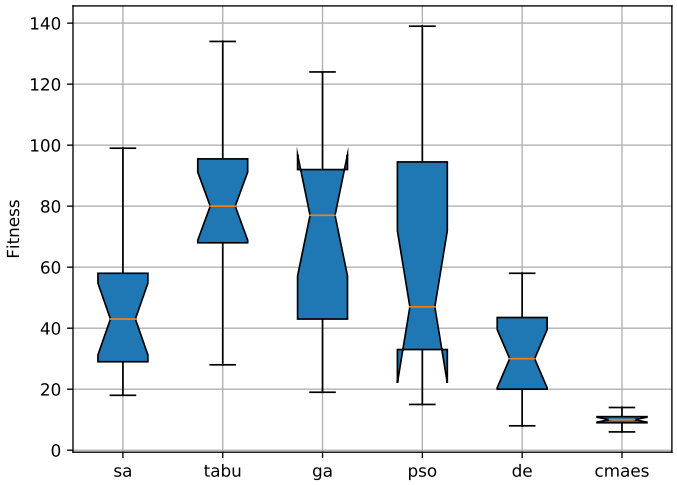


Obrázek 5.30: Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Souhrn

Tato sekce prezentuje grafy pro porovnání výsledků jednotlivých metod. Graf 5.31 porovnává nejlepší výsledky nalezené jednotlivými metodami. Z grafu je patrné, že nejlepším algoritmem se ukázala metoda **CMA-ES**. Jedním z hlavních důvodů je pravděpodobně skutečnost, že nevyžaduje nastavování parametrů. Tato metoda navíc vznikla právě se zaměřením na optimalizaci více-dimenzionálních problémů. Nejhůře dopadlo tabu prohledávání a genetický algoritmus. **Tabu prohledávání** využívalo pouze dlouhodobé paměti předchozích stavů a je pravděpodobné, že 16 dimenzí problému bylo příliš mnoho, aby byl takovýto seznam efektivní. **Genetický algoritmus** trpí problémy s různorodostí populace a omezením maximálního počtu evaluací fitness funkce. **Simulované žíhání** se ukazuje jako efektivní kompromis. Zdá se, že lineární chladící rozvrh je vhodnou variantou. **Diferenciální evoluce** je i zde, stejně jako u testovacích funkcí, na druhém místě. Spolu s **PSO** má příliš pomalou konvergenci. Dále; přestože PSO má horší médián výsledků, notche zařadí tyto výsledky s 95% jistotou na stejnou úroveň, jako DE a SA (a také GA, které je ovšem prokazatelně odlišné od SA a DE).

Vizualizaci nejlepšího nalezeného výsledku je možné vidět v přílohách na obrázku B.1.



Obrázek 5.31: Porovnání výsledků všech algoritmů pro malé množství sonikací.

5.4.2 Skvrna - vysoký počet sonikací

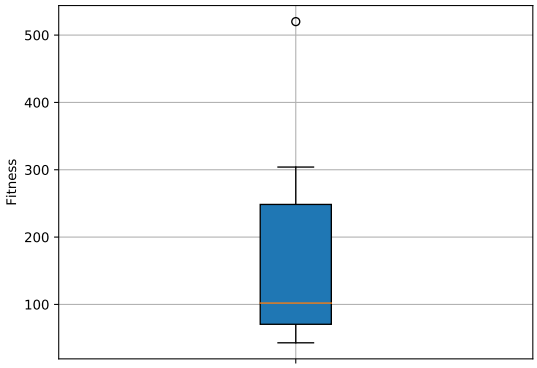
Optimalizace byly spuštěny s následujícími řídícími parametry. Došlo k menším úpravám na základě znalostí získaných při optimalizaci s malým počtem sonikací.

GA	64 jedinců v populaci	Turnajový výběr	Dvoubodové křížení 80%	Mutace 20%	Přežití nejlepších bez ohledu na generaci
SA	Počáteční teplota 2000	Krok 1%	Lineární chladící rozvrh		
Tabu	60 prvků v tabu seznamu	20 kandidátních řešení			
DE	25 jedinců v populaci	Strategie BEST/1/BIN	Šance rekombinace 90%	Faktor zesílení 0.5 až 0.9	
PSO	64 částic	Akcelerační koeficienty c1 = c2 = 2		Koeficient setrvačnosti 0.5	
CMA ES	*Nevyžaduje vstupní parametry.*				

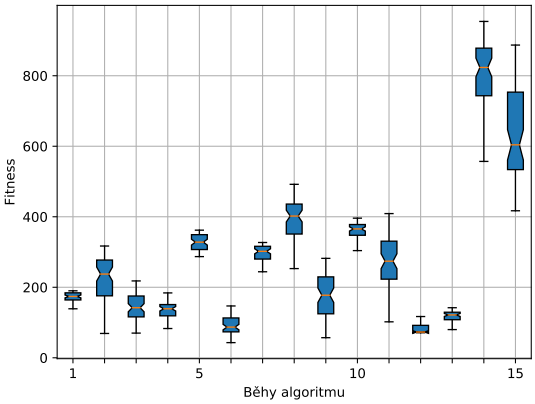
Tabulka 5.2: Tabulka řídících parametrů použitých při optimalizaci modelu skvrna vysokým počtem sonikací.

Genetický algoritmus

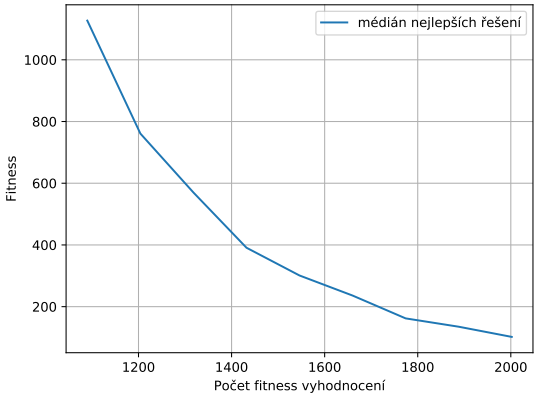
S rostoucí dimenzí roste i schopnost GA udržet si diverzitu. Genetický algoritmus byl schopný přiblížit se optimu a populace již nedegenerují - graf 5.33. Nejlepší řešení se ovšem nacházelo mimo „vousky“ boxplotu poslední generace a nelze tedy vyloučit, že se jedná o náhodu.



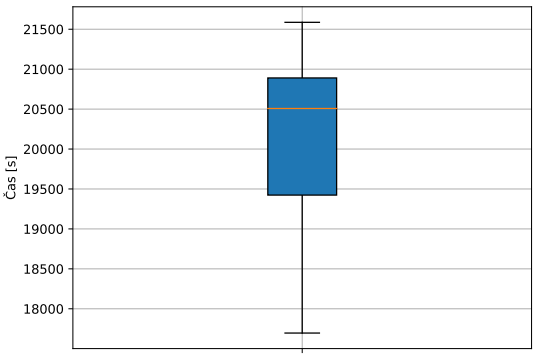
Obrázek 5.32: Boxplot nejlepších výsledků všech 15 běhů GA.



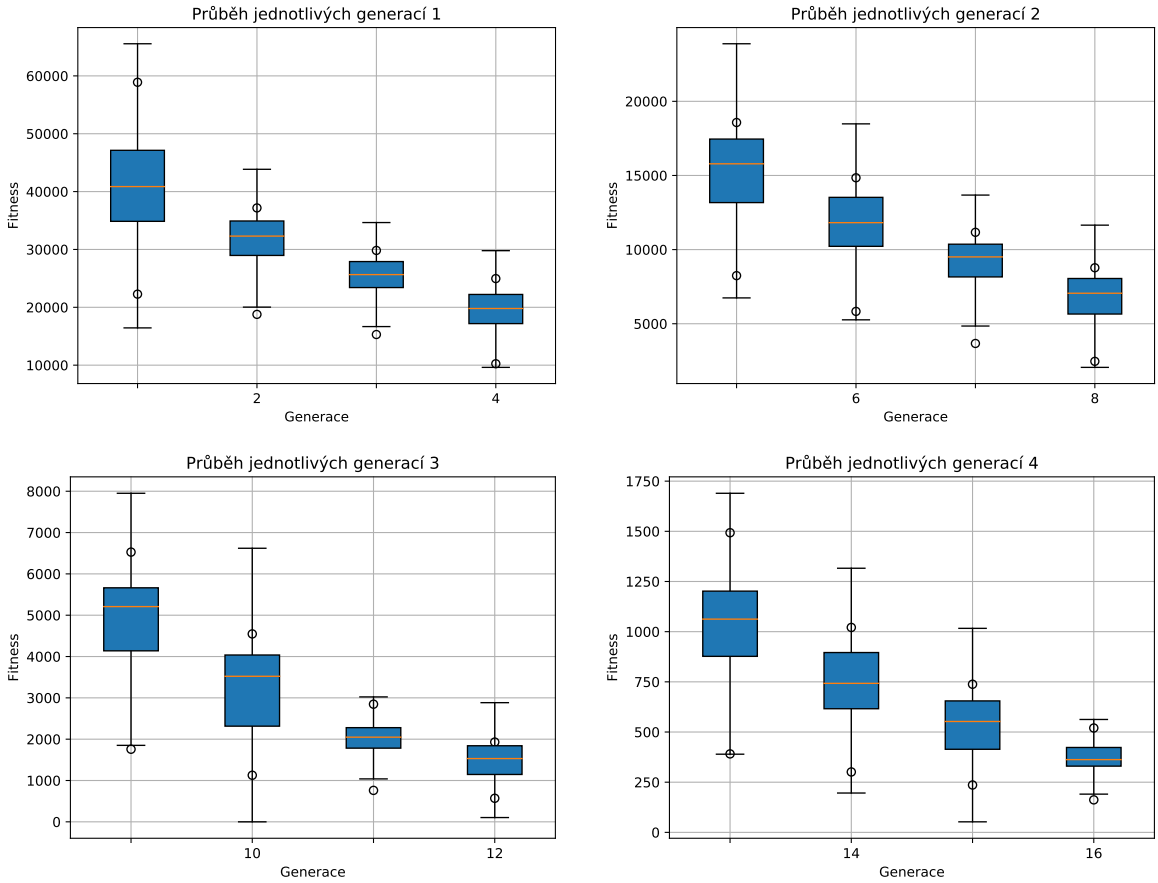
Obrázek 5.33: Boxplot stavu poslední generace všech 15 běhů GA.



Obrázek 5.34: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



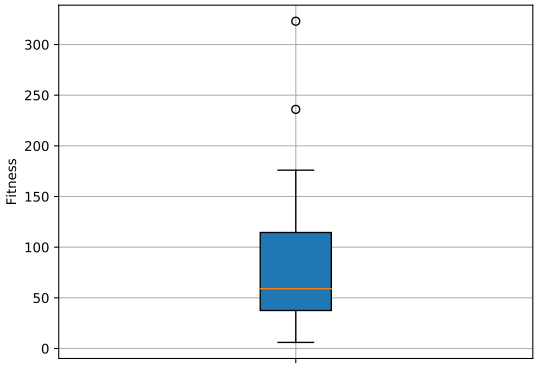
Obrázek 5.35: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800 = 8h$)



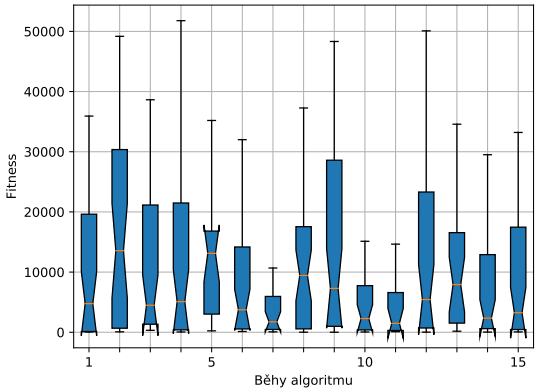
Obrázek 5.36: Evoluční průběh GA. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Simulované žíhání

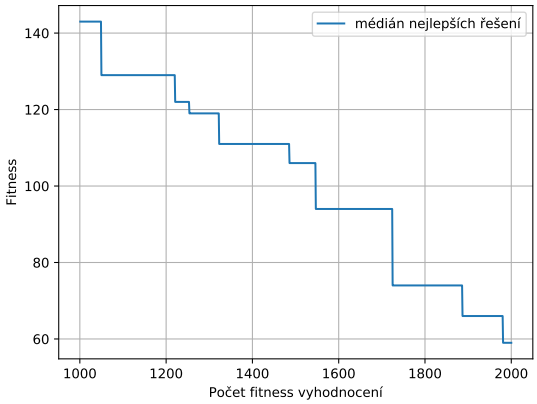
Simulované žíhání bylo i zde úspěšné. Zdá se, že zvýšení dimenze problému nemělo příliš velký vliv na schopnosti algoritmu. Navíc vylepšení fitness probíhalo i těsně před ukončením optimalizace - graf 5.39 - a lze tedy předpokládat, že přidáním dalších evaluací by bylo nalezeno lepší řešení.



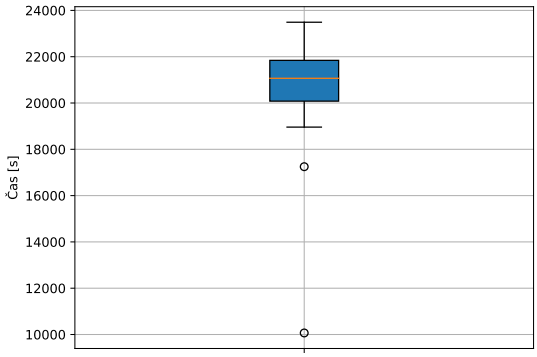
Obrázek 5.37: Boxplot nejlepších výsledků všech 15 běhů SA.



Obrázek 5.38: Boxplot ukazující rozptyl fitness, jaký každý běh SA prohledal.



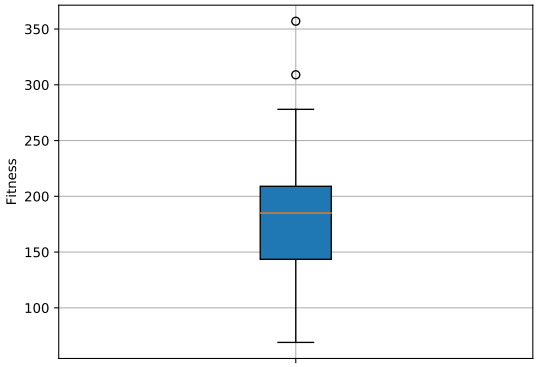
Obrázek 5.39: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



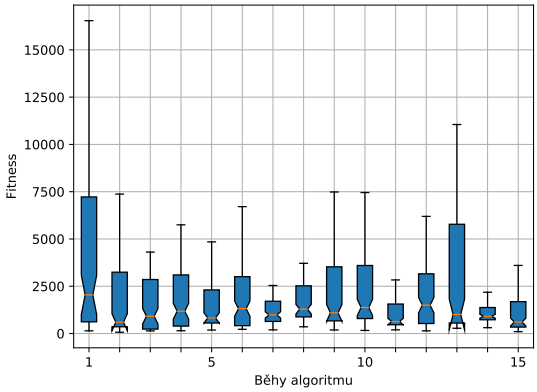
Obrázek 5.40: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800s = 8h$)

Tabu prohledávání

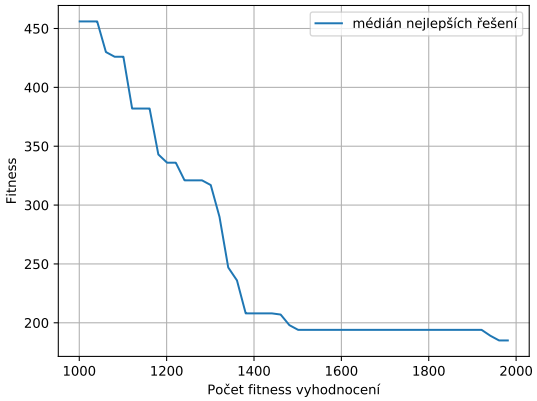
Tabu prohledávání byla jedna z metod, které byly upraveny řídící parametry, jako pokus vylepšit výkon. Z výsledků 5.42 a 5.43 se ovšem zdá, že bez většího efektu. Stále nevykazuje tak dobré výsledky, jako přímý rival v algoritmu Simulovaného žíhání. Závěr o TS z předchozího testu stále platí - vyžaduje více evaluací a nebo uvázlo v lokálním minimu. Také stále platí dodatek, že tato implementace pracuje pouze s dlouhodobým seznamem. Jedná se o omezení samotné implementace.



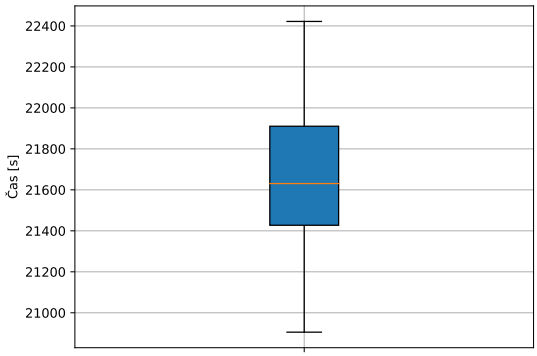
Obrázek 5.41: Boxplot nejlepších výsledků všech 15 běhů TABU.



Obrázek 5.42: Boxplot ukazující rozptyl fitness, jaký každý běh TABU prohledal.



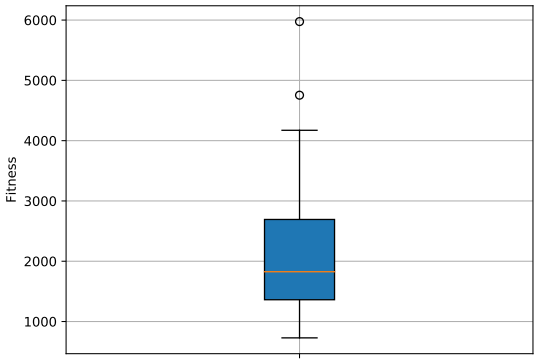
Obrázek 5.43: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



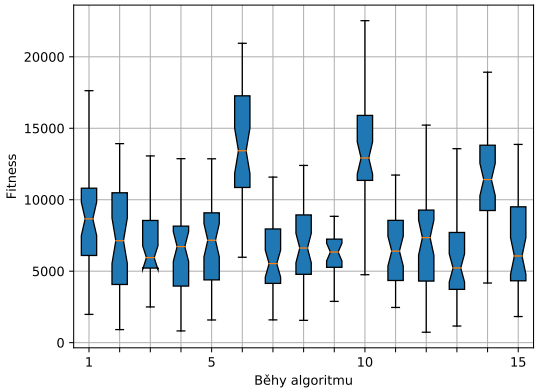
Obrázek 5.44: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800s = 8h$)

Diferenciální evoluce

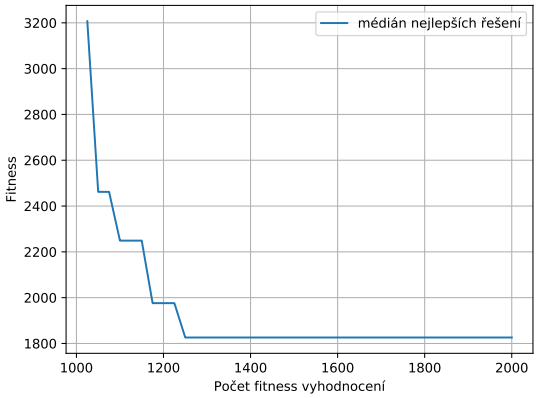
Diferenciální evoluce vykazuje markantní zhoršení výsledků se zvýšením dimenze. Z grafu 5.47 a 5.49 se zdá, že došlo k uváznutí ještě před 1300 evaluacemi účelové funkce. Jak bylo dříve zmíněno, řetěz sonikací v různých pořadích mohou mít stejnou fitness. Například řetěz, který vypálí všechny cílené body ovšem zároveň s nimi i část penalizované mapy. Dokážeme si představit, že takových řetězů může být v populací obecně více, každý zasahující na jinou část penalizační mapy. Rekombinací těchto řešení pravděpodobně vznikne řešení horší, což značně stěžuje schopnost algoritmu konvergovat při strategiích využívajících pouze fitness jako selekční tlak. Je možné, že strategie *rand*, či jiná, pracující s náhodnými prvky, by mohla zvýšit globálnost.



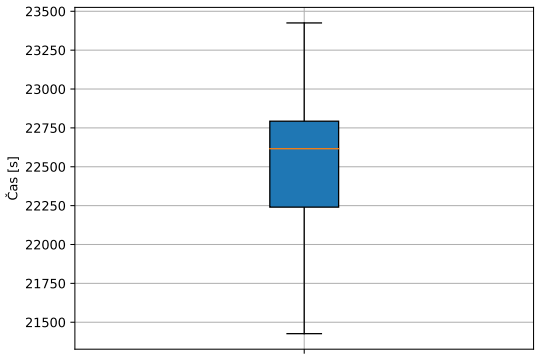
Obrázek 5.45: Boxplot nejlepších výsledků všech 15 běhů DE.



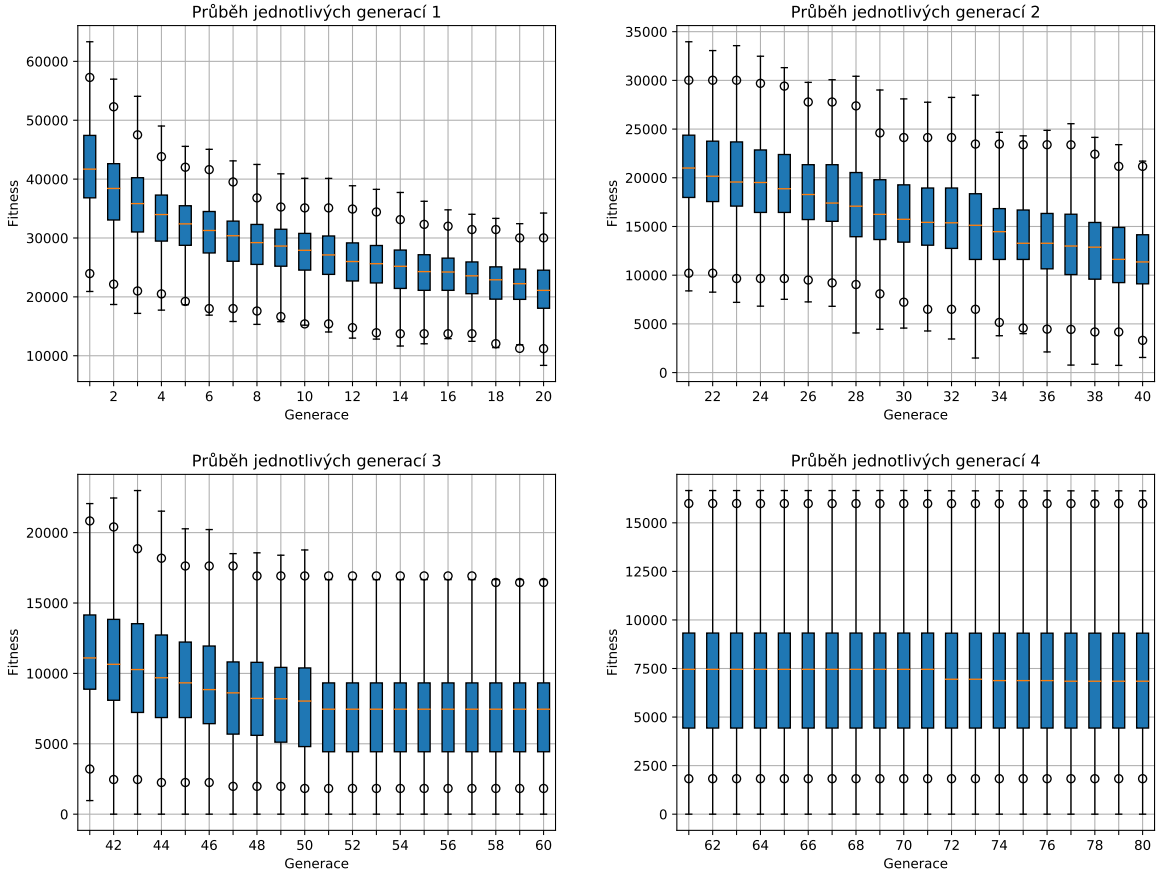
Obrázek 5.46: Boxplot stavu poslední generace všech 15 běhů DE.



Obrázek 5.47: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



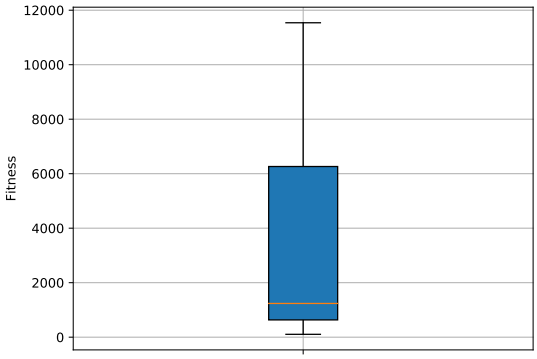
Obrázek 5.48: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800s = 8h$)



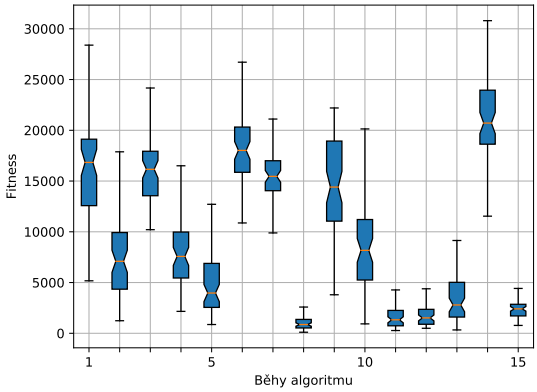
Obrázek 5.49: Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Optimalizace rojem částic

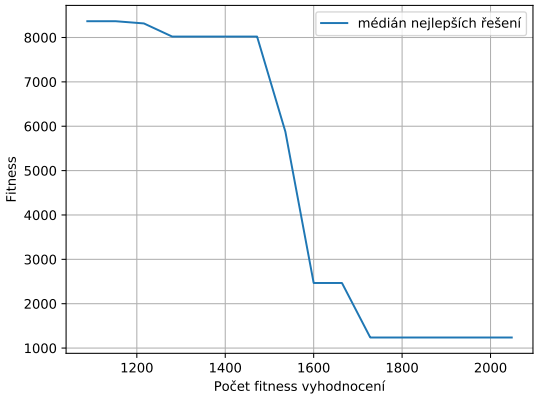
Dalším zklamáním po zvýšení dimenze problému je metoda PSO. Optimalizace rojem částic má spolu s metodou DE ve strategii *best* společnou důležitou vlastnost - svou populaci přibližuje k nějakému podprostoru, na základě směru vypočteném z nejlepších řešení. V případě, že v populaci existuje více přibližně stejných nejlepších řešení zakódovaných výrazně odlišně, algoritmus postrádá schopnost konvergovat. Multi-swarm by mohl tento problém vyřešit.



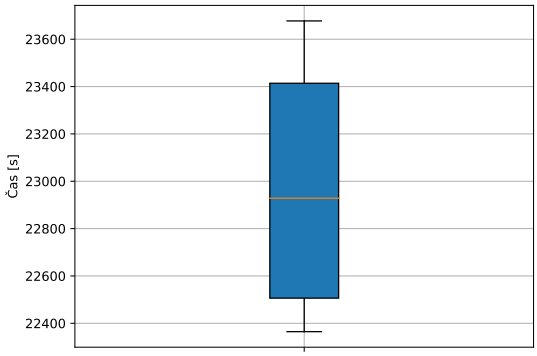
Obrázek 5.50: Boxplot nejlepších výsledků všech 15 běhů PSO.



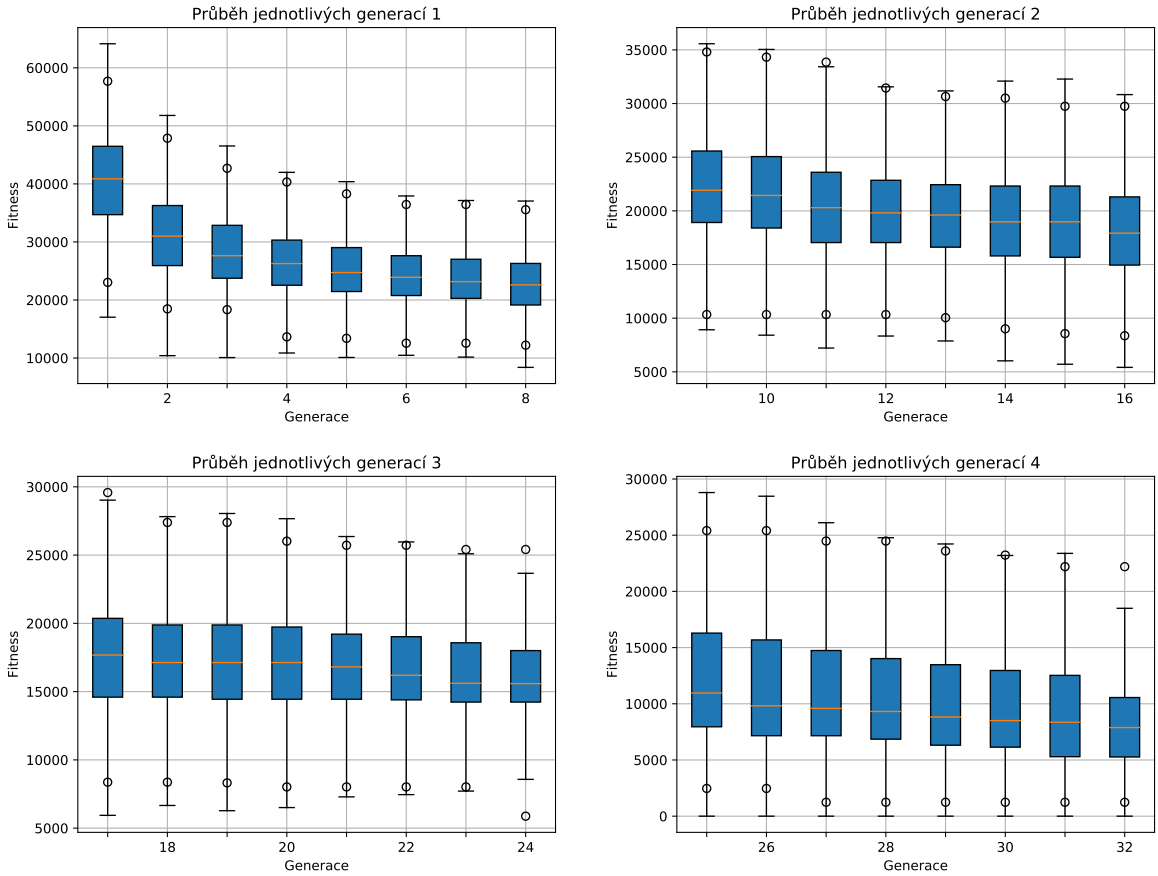
Obrázek 5.51: Boxplot stavu poslední generace všech 15 běhů PSO.



Obrázek 5.52: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



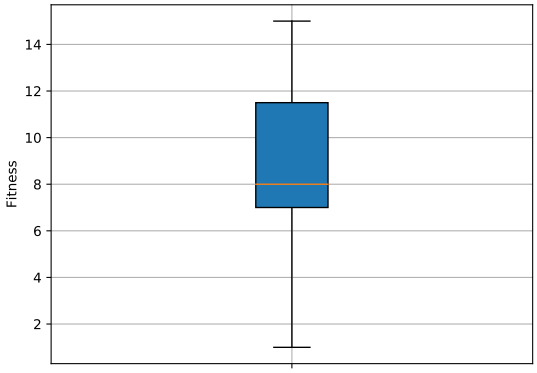
Obrázek 5.53: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800s = 8h$)



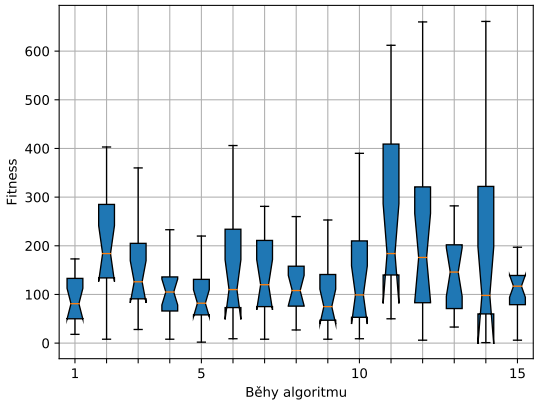
Obrázek 5.54: Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

CMA-ES

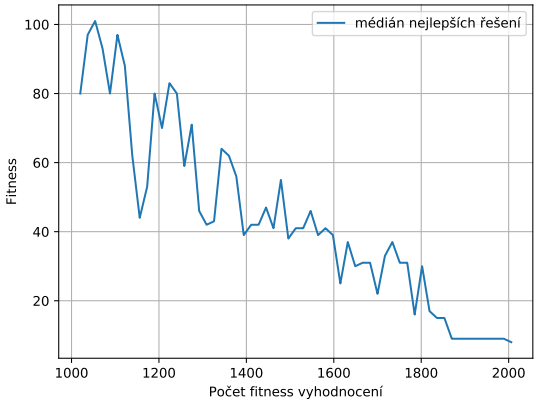
I zde se CMA-ES ukázala jako jediná metoda, schopná konzistentně najít, nebo se alespoň přiblížit, ke hledanému optimu (5.26).



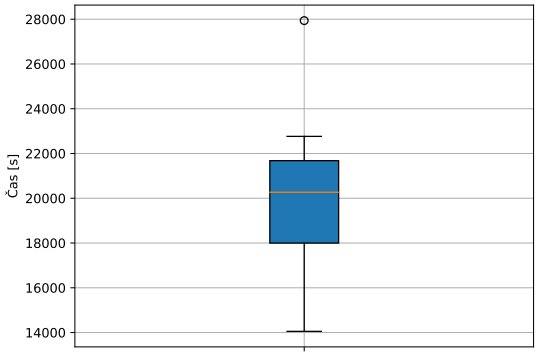
Obrázek 5.55: Boxplot nejlepších výsledků všech 15 běhů CMA-ES.



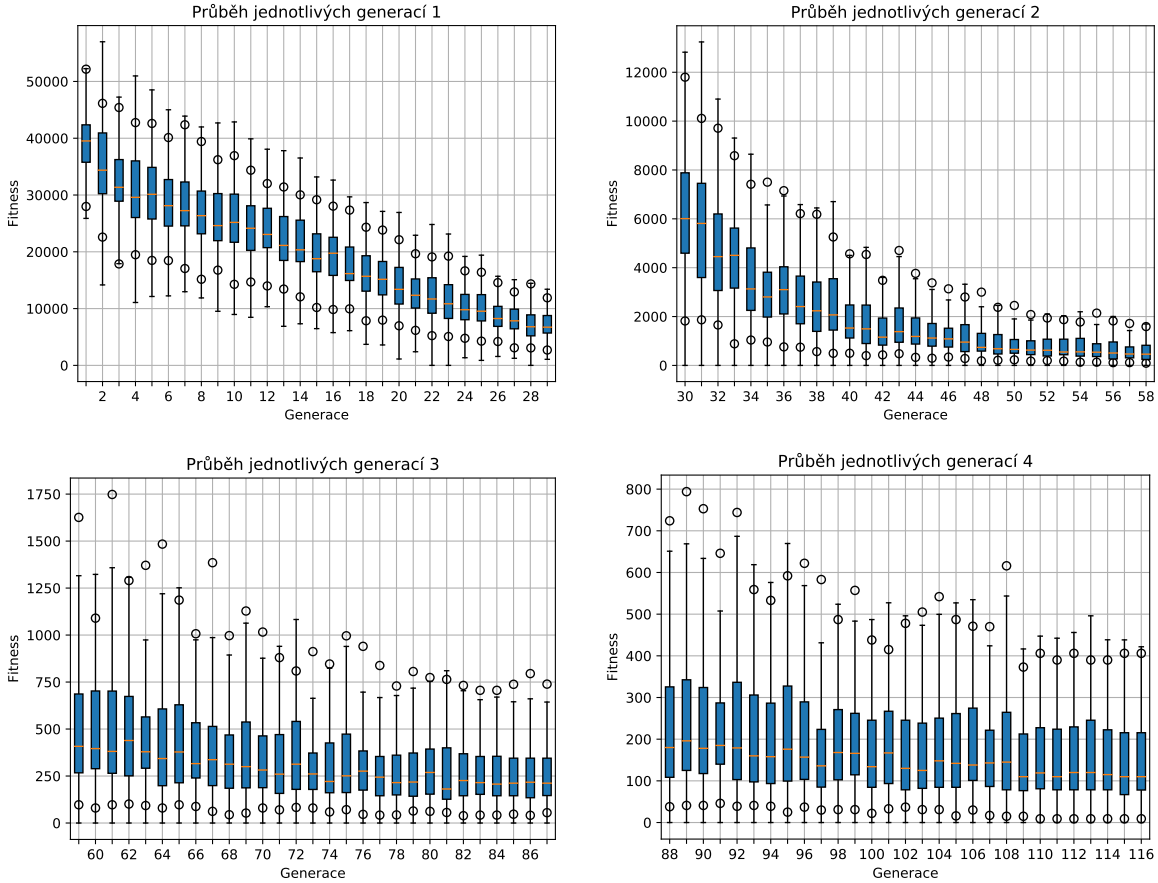
Obrázek 5.56: Boxplot stavu poslední generace všech 15 běhů CMA-ES.



Obrázek 5.57: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



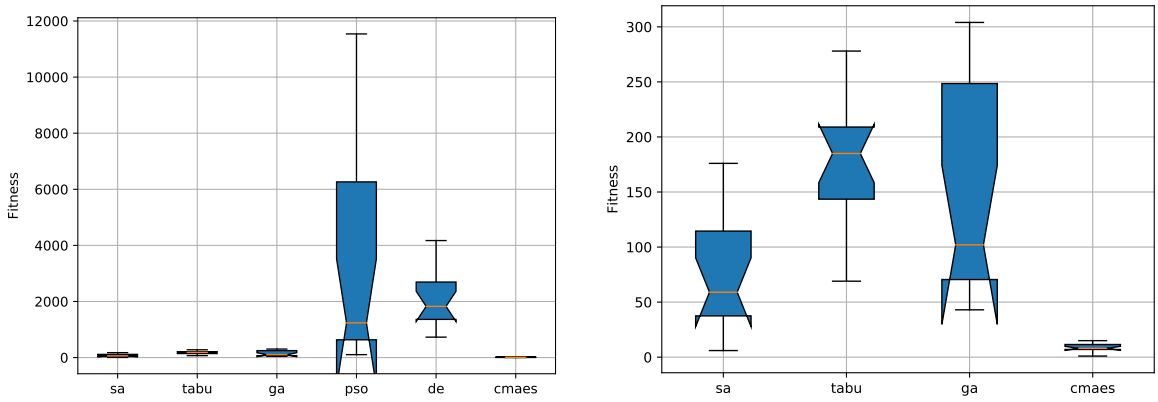
Obrázek 5.58: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800s = 8h$)



Obrázek 5.59: Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Souhrn

Grafy 5.60 jasně ukazují, že i zde vyhrává metoda **CMA-ES**. Jak již bylo zmíněno, metoda nevyžaduje vstupní řídící parametry a optimalizace více-dimenzionálních problémů je její „forte“. Nejhůře dopadlo **PSO** a **Diferenciální evoluce**, pravděpodobně z důvodu pomalé, nebo i zamrzlé konvergence. Jak bylo zmíněno, obě metody posouvají svou populaci směrem, který je udáván nejlepšími jedinci v populaci. Pokud je takto silných jedinců v populaci mnoho a každý udává jiný/opačný směr, algoritmy nekonvergují. Lze předpokládat, že tato situace nastala - algoritmus dokázal odstranit celou cílenou část, ovšem za cenu velké penalizace zasažením zakázaných oblastí. **Tabu prohledávání** je i nadále horší než **Simulované žíhání**, které se zdá být invariantní vůči dimenzi problému. Naopak **Genetický algoritmus** byl díky zvýšení dimenze schopen překonat problémy s diverzitou a velice rychle se posunul směrem k optimu. Vizualizaci nejlepšího nalezeného výsledku je možné vidět v přílohách na obrázku B.2.



Obrázek 5.60: Porovnání výsledků všech algoritmů pro 20 sonikací. Vlevo všechny metody, vpravo detail na úspěšnější z nich.

5.4.3 Květina

Pro optimalizaci tvaru květina bylo přistoupeno k ústupkům vzhledem k časové náročnosti a předchozím poznatkům. Hlavním z těchto ústupků je vynechání **Tabu prohledávání** a **Optimalizace rojem částic**. Ani jeden z těchto algoritmů nebyl schopen překonat svého přímého konkurenta (Simulované žíhání a Diferenciální evoluci resp.). Další z ústupků plyne ze znalosti kvality výsledků a časové náročnosti výpočtů - každému algoritmu bylo provedeno pouze 10 nezávislých běhů, namísto předchozích 15, z důvodu šetření procesorových hodin. Model květina je složitějším problémem, než skrvna především z důvodu reziduálního akumulovaného tepla ve středu modelu. Pro úsporu času bylo také použito 15, namísto předešlých 20 sonikací. Částečné řešení problému reziduálního tepla je možné za pomoci vysokých hodnot proměnné t_{off} v rámci sonikací. Vysoké hodnoty mají ovšem za následek delší simulace, které při původních 20 sonikacích přesáhli hranici 12 hodin na superpočítači. Takto nalezené řešení by nesplnilo dříve představena kritéria na včasnost výsledků.

Optimalizace byly spuštěny s následujícími řídícími parametry.

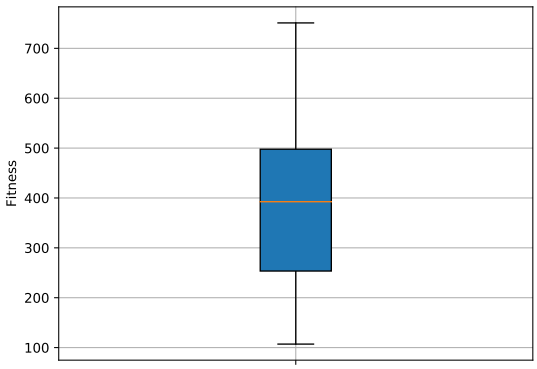
GA	64 jedinců v populaci	Turnajový výběr	Dvoubodové křížení 80%	Mutace 20%	Přežití nejlepších bez ohledu na generaci
SA	Počáteční teplota 2000	Krok 1%	Lineární chladící rozvrh		
DE	25 jedinců v populaci	Strategie BEST/1/BIN	Šance rekombinace 90%	Faktor zesílení 0.5 až 0.9	
CMA ES	*Nevyžaduje vstupní parametry.*				

Tabulka 5.3: Tabulka řídících parametrů použitých při optimalizaci modelu květina větším počtem sonikací.

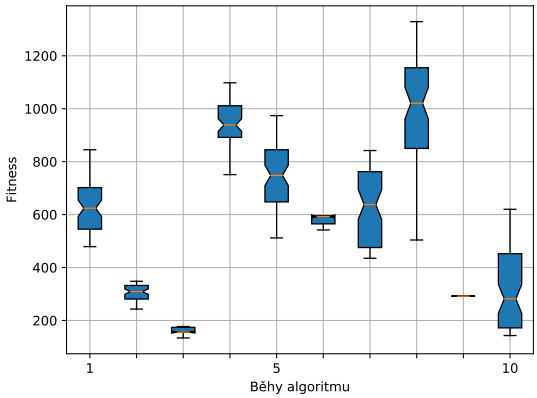
Genetický algoritmus

Genetický algoritmus nedokázal nalézt optimální řešení při omezeném množství evaluací, dokázal ovšem překonat lokální extrémy a přiblížit se. Při pohledu na grafy 5.62 a 5.65 lze

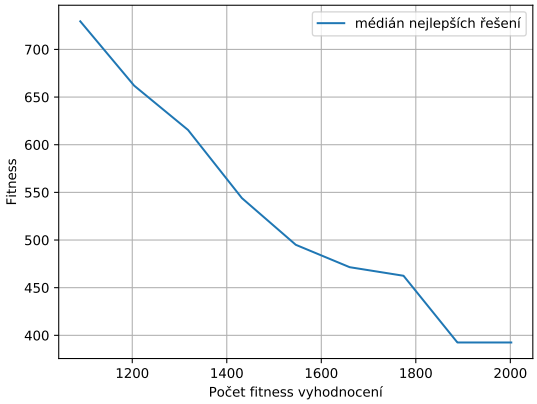
ovšem předpokládat, že při případném uvolnění tohoto omezení by GA bylo schopno dále pokračovat a řešení upřesnit.



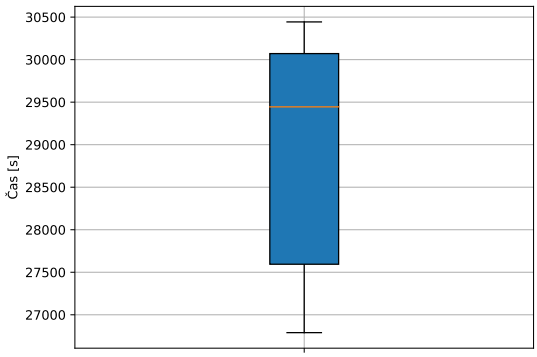
Obrázek 5.61: Boxplot nejlepších výsledků všech 10 běhů GA.



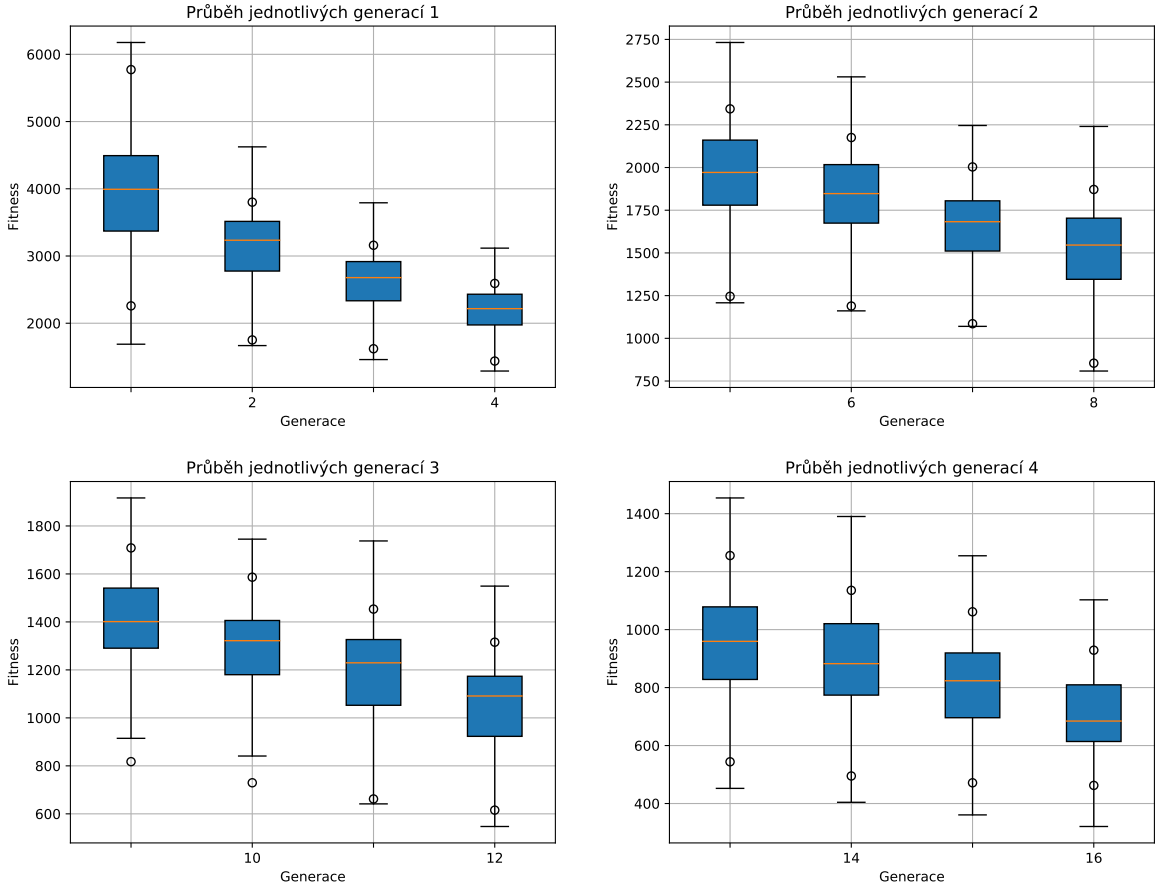
Obrázek 5.62: Boxplot stavu poslední generace všech 10 běhů GA.



Obrázek 5.63: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



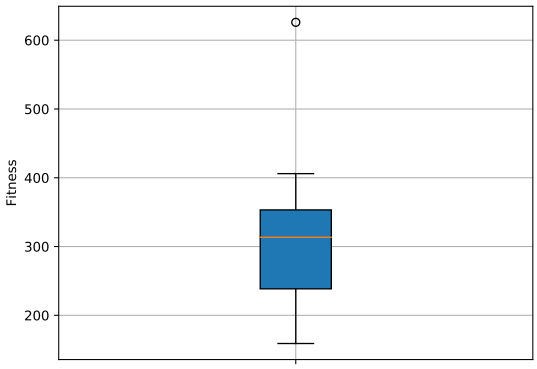
Obrázek 5.64: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800 = 8h$)



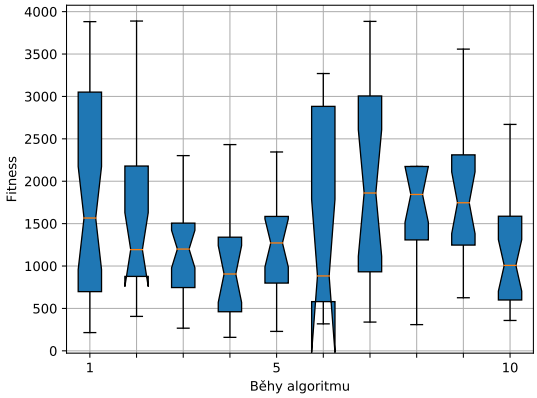
Obrázek 5.65: Evoluční průběh GA. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Simulované žíhání

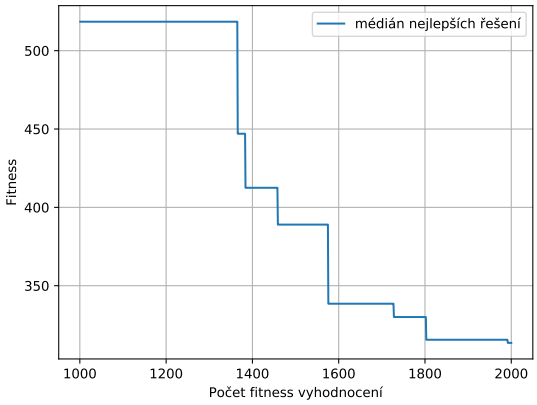
Simulované žíhání bylo nejúspěšnější z testovaných metod, ovšem samotné výsledky jsou stále neuspokojivé - algoritmus se pohybuje v okolí optima ale nedokáže konvergovat. Toto chování algoritmus vyznačoval již při předchozích testech. Na vinně je pravděpodobně velikost kroku, která je generována normální distribuční funkcí s standardní odchylkou 1/5 rozmezí proměnné, která je právě perturbována. Tento krok se zdá být příliš velký v pozdějších fázích prohledávání.



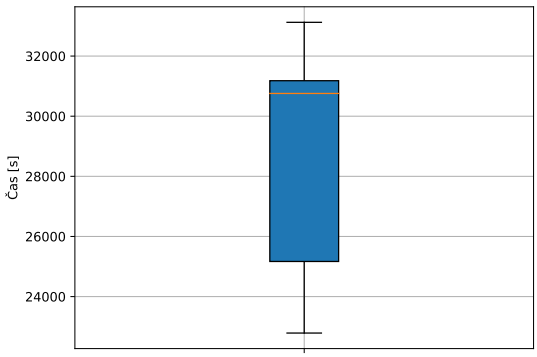
Obrázek 5.66: Boxplot nejlepších výsledků všech 10 běhů SA.



Obrázek 5.67: Boxplot ukazující rozptyl fitness, jaký každý běh SA prohledal.



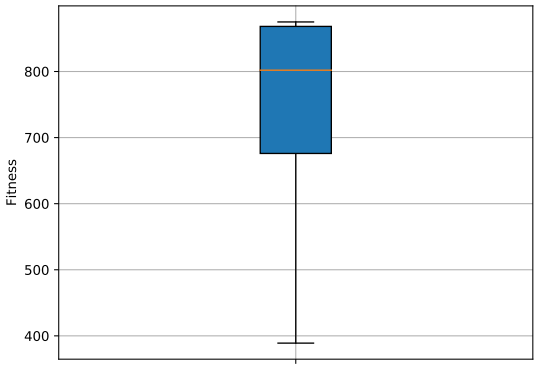
Obrázek 5.68: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



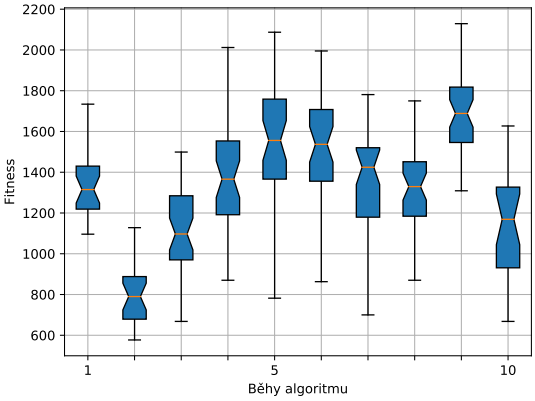
Obrázek 5.69: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800s = 8h$)

Diferenciální evoluce

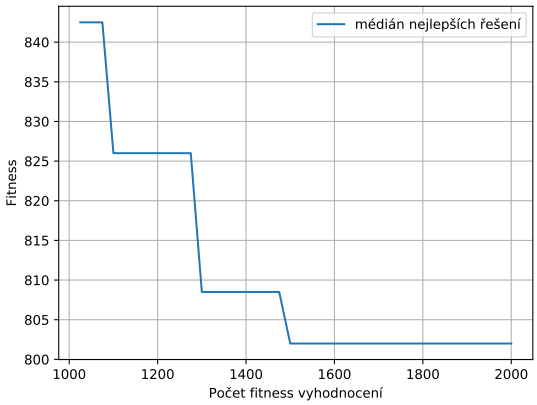
Diferenciální evoluce vykazuje stejné tendence, jako při předchozím testu. Problém reziduálního tepla způsobí, že stavový prostor je plný lokálních extrémů, které se od ostatních jedinců liší především pořadím provedení sonikací. Na grafu 5.72 je vidět, že přibližně od 1500 evaluací již nedocházelo ke změnám. Pravděpodobně se jedná o stav, který jako jeden z mála nezničí prostřední penalizovanou oblast, ovšem ani nedokáže pokrýt cíl.



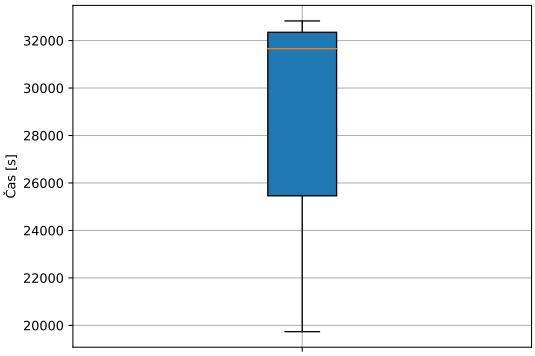
Obrázek 5.70: Boxplot nejlepších výsledků všech 10 běhů DE.



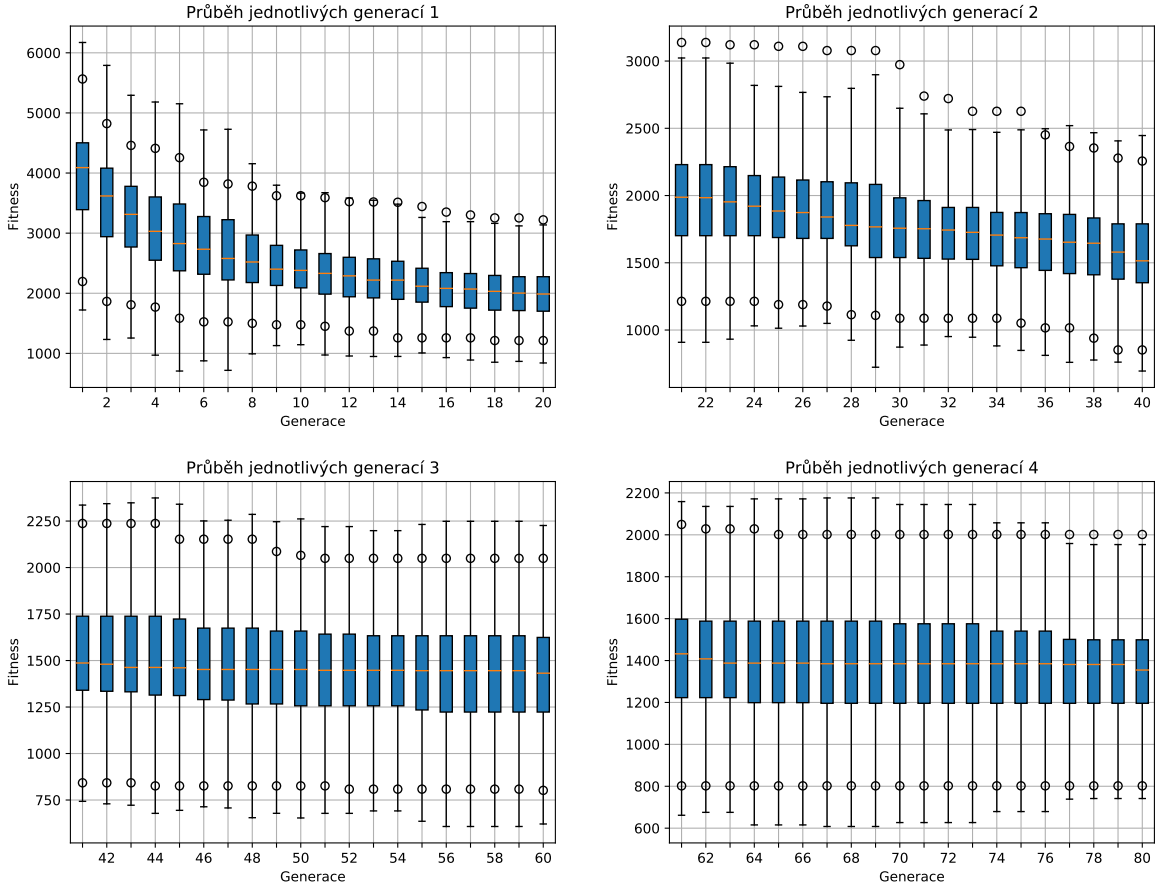
Obrázek 5.71: Boxplot stavu poslední generace všech 10 běhů DE.



Obrázek 5.72: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



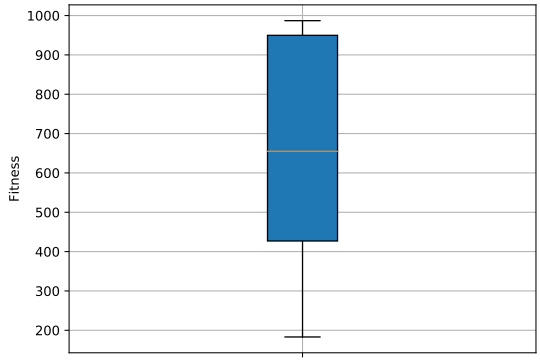
Obrázek 5.73: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800s = 8h$)



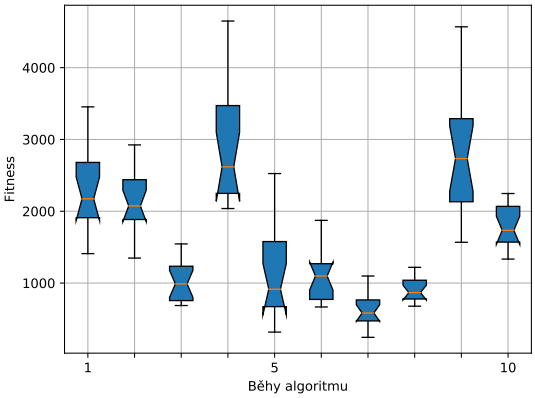
Obrázek 5.74: Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

CMA-ES

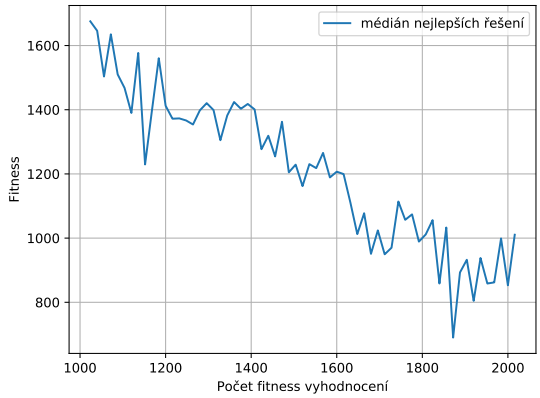
Tento problém se ukázal jako příliš náročný i pro CMA-ES. Zdá se, že metoda se přeučí na stavy na počátku hledání a není následně schopna řešit pozdější lokální extremy. V počátečních fázích jsou nalezena lepší řešení kolem hodnot, které jsou dlouho lokálním optimem a metoda bohužel upraví kovariance do stavů, které nejsou schopny při případném úniku nalézt globální optimum.



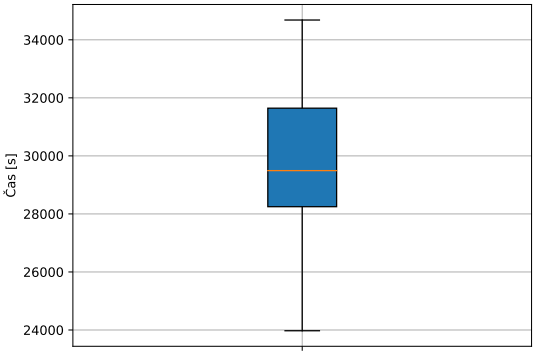
Obrázek 5.75: Boxplot nejlepších výsledků všech 10 běhů CMA-ES.



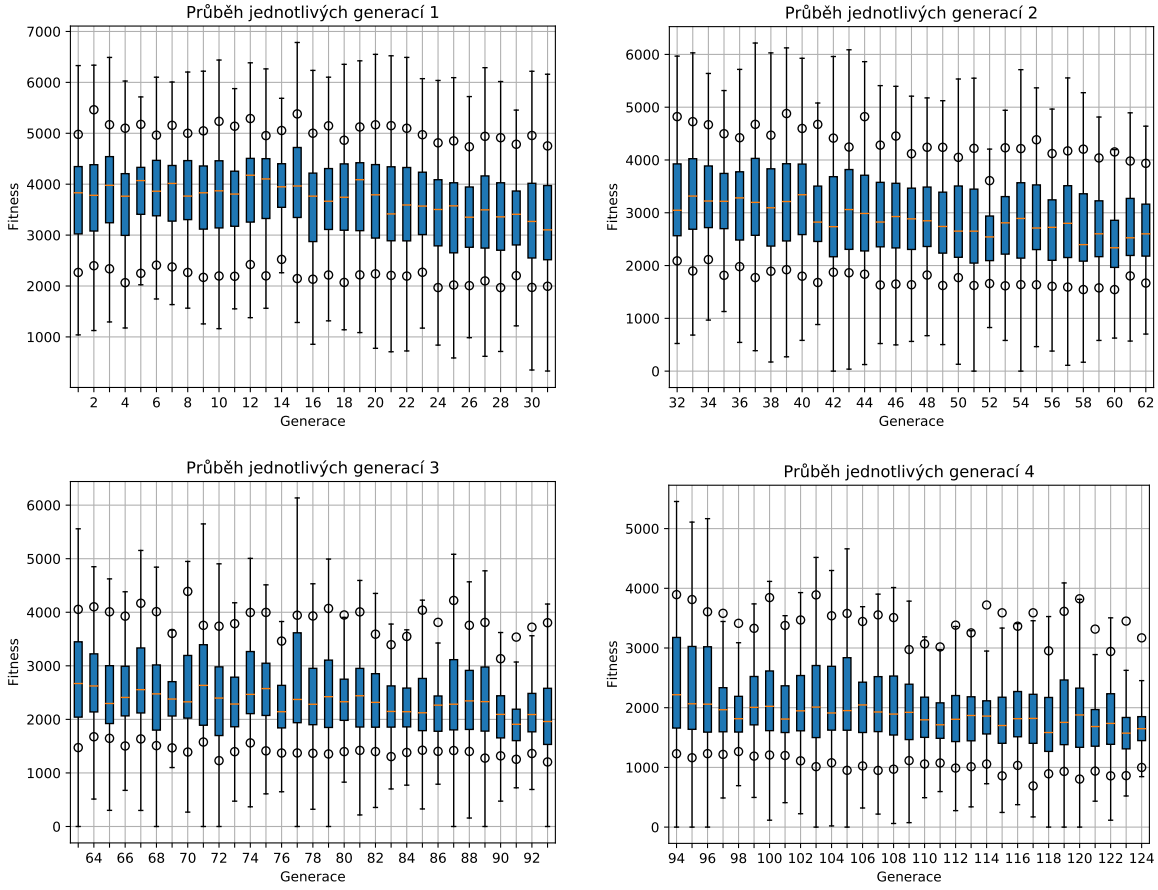
Obrázek 5.76: Boxplot stavu poslední generace všech 10 běhů CMA-ES.



Obrázek 5.77: Poměr mediánu nejlepších nalezených řešení vůči počtu evaluací fitness funkce. Zobrazena až druhá poloviny optimalizace.



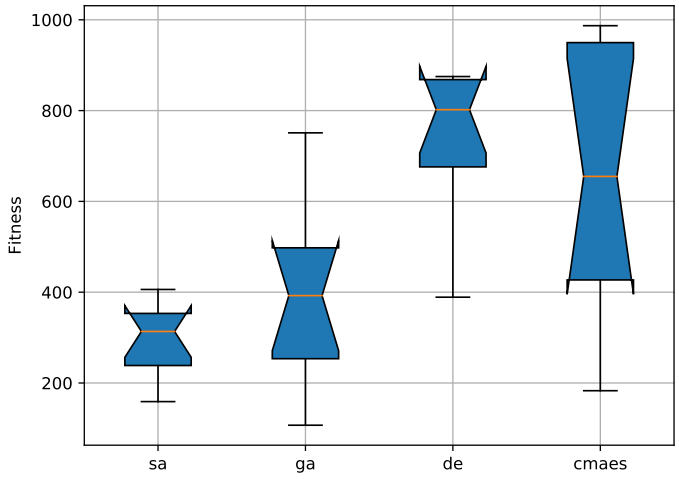
Obrázek 5.78: Boxplot času potřebného k dokončení optimalizace ve vteřinách. ($28800s = 8h$)



Obrázek 5.79: Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Souhrn

Grafy 5.60 ukazují, že ani jeden z algoritmů nebyl úspěšný při řešení tohoto typu problému. Největším zákoutím je nutnost a tlak na algoritmy vyhnout se střednímu bodu, který je velice snadno spálen díky kladné zpětné vazbě vznikajícího reziduálního tepla při pálení okolí. Genetický algoritmus je příliš omezen kritériem 2000 evaluací, Diferenciální evoluce uvázne v lokálním minimu, Simulované žíhání nemá dostatečně flexibilní krok pro přesnou konvergenci a CMA-ES se přeučí na neoptimální počátečním prostředí. Vhodným řešením by mohla být kombinace SA a CMA-ES, při které je prvních 1000 evaluací použito k nalezení přibližné pozice optima algoritmem SA a následně výsledek předán CMA-ES jako počáteční stav. Vizualizaci nejlepšího nalezeného výsledku je možné vidět v přílohách na obrázku B.3.



Obrázek 5.80: Porovnání výsledků všech algoritmů pro problém květiny.

5.5 Shrnutí a přínos pro klinickou praxi

Bylo bezesporu ukázáno, že pro problém připomínající typ „skvrna“ je nejvhodnějším kandidátem algoritmus **CMA-ES**. Splňuje kritéria včasnosti výpočtu i přesnosti - jako jediný byl schopen nalézt optimum ve vymezeném rozsahu evaluací. Problém typu „květina“ nebyl řešitelný v rámci definovaných kritérii - nepodařilo se nalézt optimum. Pro přibližné řešení tohoto typu oblasti je ovšem možné využít algoritmu **Simulované žíhání**.

Hlavním rysem obou metod je menší množství vstupních parametrů, případně citlivost metody na tyto parametry. Přejeme-li si najít vhodnou evoluční metodu pro řešení problému hledání trajektorie HIFU operací, jedná se o žádaný efekt. Platí „no-free-lunch“ teorém a problém je závislý na mnoha externích faktorech.

Stále je nutno dalších experimentů s dodatečnými typy prostorů a pokusy k ověření teorií prezentovaných při hodnocení výsledků jednotlivých algoritmů.

Kapitola 6

Závěr

Byla představena problematika optimalizace obecně a optimalizace plánování trajektorie HIFU operací. Následně bylo podrobně prostudováno několik známých evolučních metod optimalizace systémů a procesu operačního výzkumu. O těchto evolučních algoritmech je pojednáno v kapitole 3.

Vzniklo několik optimalizačních programů s jednotným rozhraním pro využití implementací externích knihoven. Dále několik skriptů transformujících data popisující průběh optimalizace do podoby statistických grafů. Rozhraní těchto vzniklých programů a skriptů je popsáno v kapitole **??**. Následně byly tyto programy otestovány na navržené testovací sadě, která byla vybrána tak, aby vhodně reprezentovala základní rozdělení spektra optimalizačních problémů. Nakonec byla představena kritéria pro výběr vhodné metody s ohledem na tzv. „no-free-lunch“ teorém. Tyto kritéria jsou reprezentována jako omezení počtu evaluací simulace šíření tepla a přesnost v bezpečném rozmezí. Následně bylo pro porovnání metod a ověření hypotéz vykonáno 30 experimentů nad každou z navržených funkcí a výsledná data prezentována v této práci - kapitola 4. Ve stejné části byla následně diskutována povaha modelu šíření tepla v kontextu „tvaru“ fitness funkce, a provedena přibližná redukce na jeden z problému prezentovaných v testovací sadě - *Rosenbrockovu funkci*. V poslední kapitole - 5 - byla podrobněji popsána implementace simulace šíření tepla ve tkáních. Na základě znalostí implementace modelu a datasetu byla představena omezení a úpravy pro převod na vzniklé rozhraní optimalizačních programů. Posléze byly definovány dva odlišné modely cíle pro optimalizaci - model „skvrny“ a model „květiny“. Následně byly provedeny experimenty, které využívají poznatků získaných z navržené testovací sady a empirických dat získaných při studii modelu. Následně bylo na modelu „skvrny“ provedeno 15 nezávislých běhů pro každý algoritmus, pro malé i vysoké počty sonikací - 4 a 20 resp. Tím bylo ukázáno, že algoritmy jsou funkční i pro problémy připomínající reálné případy. Výsledky jednotlivých metod byly diskutovány v kontextu samotného modelu i vlastností metod samotných a navržena případná vylepšení. V další části kapitoly 5 bylo na základě těchto výsledků vybráno několik nejlepších algoritmu pro pokus o optimalizaci trajektorie v modelu tvaru „květina“, který byl předpokládán jako složitější k nalezení trajektorie vyhovující podmínkám. Bohužel, pro tento model se nepodařilo nalézt optimum, nýbrž pouze přibližnou aproximaci. V kontextu *no-free-lunch* teorému byly poté pro další studium a úpravy doporučeny metody **CMA-ES** a **Simulované žíhání**.

Všechny experimenty byly prováděny na superpočítači *Salomon* projektu *IT4Innovation*.

Literatura

[1] Baštinec, J. a Svoboda, Z. *Náhodné procesy*. Brno: Ústav matematiky, Fakulta elektrotechniky a komunikačních technologií, Vysoké učení technické v Brně, 2014.

[2] Bidlo, M. *Materiály kurzu EVO19 - Aplikované evoluční algoritmy* [Stránka kurzu <https://www.fit.vut.cz/study/course/13232/.cs> Navštíveno 2020.01.05].

[3] Brent, A. a Labuschagne, C. A REVIEW OF OPERATIONAL RESEARCH AND MATHEMATICAL METHODS FOR ENVIRONMENTAL MANAGEMENT APPLICATIONS IN INDUSTRY. *The South African Journal of Industrial Engineering*. Leden 2005, roč. 16.

[4] Brownlee, J. *A Gentle Introduction to Markov Chain Monte Carlo for Probability* [Machine Learning Mastery Pty. Ltd.]. [Online; navštíveno 07.01.2020]. Dostupné na: <https://machinelearningmastery.com/markov-chain-monte-carlo-for-probability/>.

[5] Chlebík, J. *Optimalizační metody pro knihovnu SIMLIB/C++*. Brno: Vysoké učení technické v Brně. Fakulta informačních technologií. Ústav inteligentních systémů, 2017. Bakalářská práce. Dostupné na: <http://hdl.handle.net/11012/69674>.

[6] Dřímal, D. *Úvod do metody Monte-Carlo*. Brno: Masarykova univerzita, 2000. 122 s. ISBN 80-210-0228-X.

[7] Glover, F. Tabu Search Fundamentals and Uses. Leden 1994.

[8] Glover, F. a Marti, R. *Tabu Search*. Boston, MA: Springer US, 2006. S. 53–69. Dostupné na: <https://doi.org/10.1007/0-387-33416-5_3>. ISBN 978-0-387-33416-5.

[9] Hansen, N. *The CMA Evolution Strategy: A Tutorial*. duben 2016. Dostupné na: <http://arxiv.org/abs/1604.00772>.

[10] Ichihara, M., Sasaki, K., Umemura, S.-I. et al. Blood Flow Occlusion Via Ultrasound Image-Guided High-Intensity Focused Ultrasound and Its Effect on Tissue Perfusion. *Ultrasound in medicine & biology*. Duben 2007, roč. 33. S. 452–9.

[11] Jaroš, J. et al. *Simulované žíhání* [FIT VUT v Brně]. [Online; navštíveno 07.01.2020]. Dostupné na: <http://www.fit.vutbr.cz/~jarosjir/groups/eva/sa.html.cz>.

[12] Leon, M. a Xiong, N. Investigation of Mutation Strategies in Differential Evolution for Solving Global Optimization Problems. In Rutkowski, L., Korytkowski, M.,

Scherer, R. et al. (ed.). *Artificial Intelligence and Soft Computing*. Cham: Springer International Publishing, 2014. S. 372–383. ISBN 978-3-319-07173-2.

[13] Massey, J. a Yilmaz, A. AustinMan and AustinWoman: High-fidelity, anatomical voxel models developed from the VHP color images. In. Srpen 2016. S. 3346–3349.

[14] Mitchell, M. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998. ISBN 0262631857.

[15] Poli, R., Kennedy, J. a Blackwell, T. Particle Swarm Optimization: An Overview. *Swarm Intelligence*. říjen 2007, roč. 1.

[16] Sun, C., Zhou, H. a Chen, L. Improved differential evolution algorithms. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*. May 2012. S. 142–145. ISSN null.

[17] Suomi, V., Jaros, J., Treeby, B. et al. Full Modeling of High-Intensity Focused Ultrasound and Thermal Heating in the Kidney Using Realistic Patient Models. *IEEE Transactions on Biomedical Engineering*. Září 2018, PP. S. 1–1.

[18] T, T. *Particle Swarm Optimisation in Machine Learning* [Medium, Towards Data Science]. [Online; navštíveno 08.01.2020]. Dostupné na: <https://towardsdatascience.com/particle-swarm-optimisation-in-machine-learning-b01b1d2ad8a8>.

[19] Treeby, B., Jaros, J., Rendell, A. et al. Modeling nonlinear ultrasound propagation in heterogeneous media with power law absorption using a k-space pseudospectral method. *The Journal of the Acoustical Society of America*. červen 2012, roč. 131. S. 4324–36.

[20] Čudová, M., Treeby, E. B. a Jaroš, J. Design of HIFU Treatment Plans using Evolutionary Strategy. In *GECCO'18 Companion: Genetic and Evolutionary Computation Conference Companion*. [b.m.]: Association for Computing Machinery, 2018. S. 1568–1575. Dostupné na: <https://www.fit.vut.cz/research/publication/11696>. ISBN 978-1-4503-5764-7.

[21] Vikhar, P. A. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. Dec 2016. S. 261–265. ISSN null.

[22] Weisser, R. *Evoluční optimalizace řídicích algoritmů*. Brno: Vysoké učení technické v Brně. Fakulta strojního inženýrství. Ústav automatizace a informatiky, 2010. Disertační práce. Dostupné na: <http://hdl.handle.net/11012/7799>.

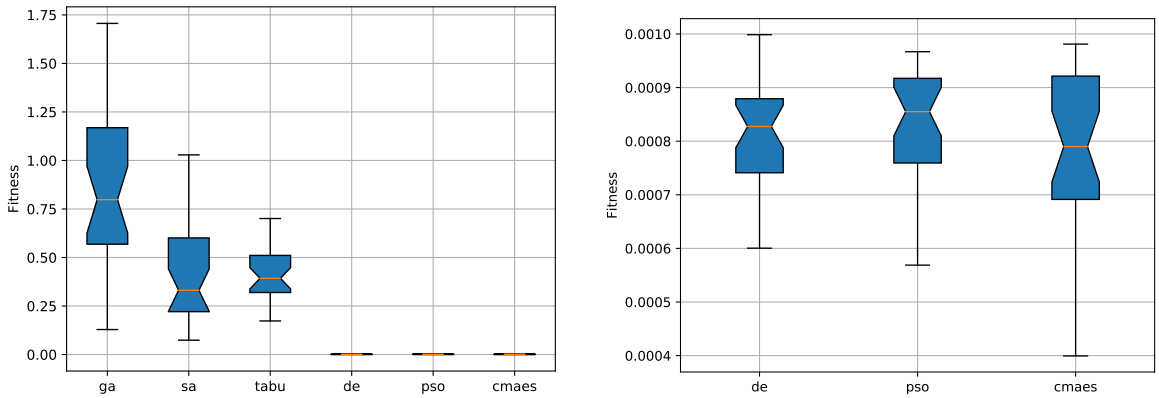
Příloha A

Výsledky optimalizace testovacích funkcí

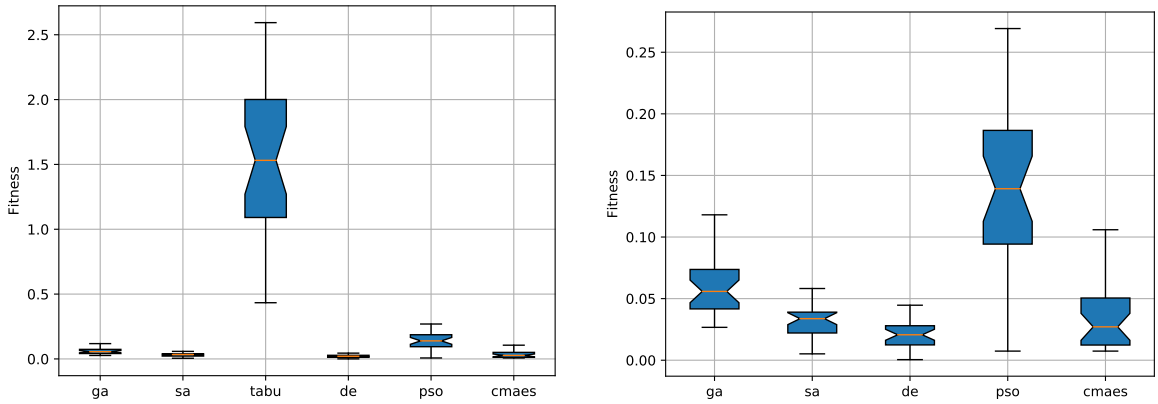
Tato kapitola obsahuje kompletní výsledky optimalizací testovacích funkcí Ackley, Griewank a Rosenbrock.

A.1 Validace

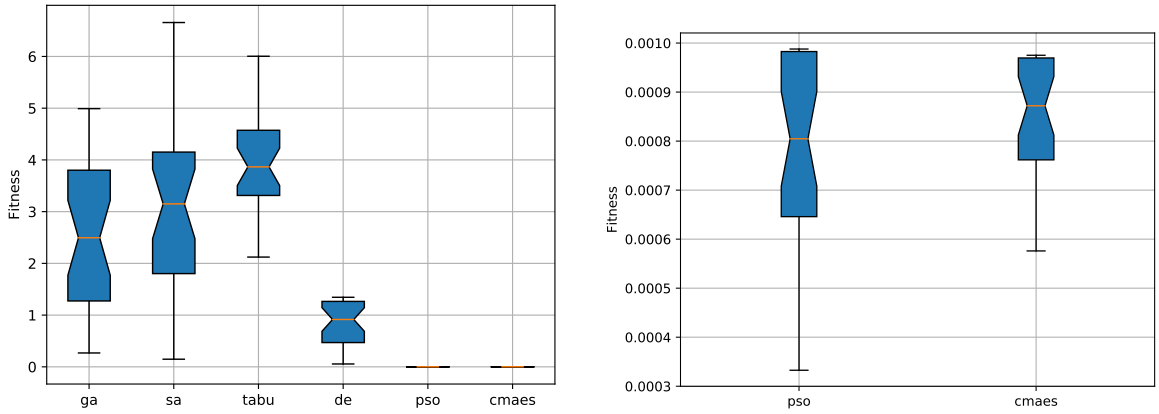
Tato sekce obsahuje výsledné grafy validace optimalizačních algoritmů nad testovací sadou.



Obrázek A.1: Výsledky optimalizace Ackleyho funkce.



Obrázek A.2: Výsledky optimalizace Griewankovi funkce.



Obrázek A.3: Výsledky optimalizace Rosenbrockovi funkce.

A.2 Omezené testy

Tato sekce obsahuje statisticky zpracované data popisující průběh testů omezené optimalizace Ackleyho funkce.

A.2.1 Ackleyho funkce

Následující výsledky jsou agregací 30 běhů každého optimalizačního algoritmu.

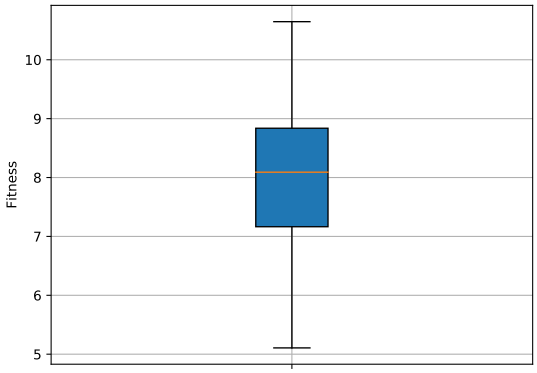
Genetický algoritmus

Pro optimalizaci byly zvoleny parametry:

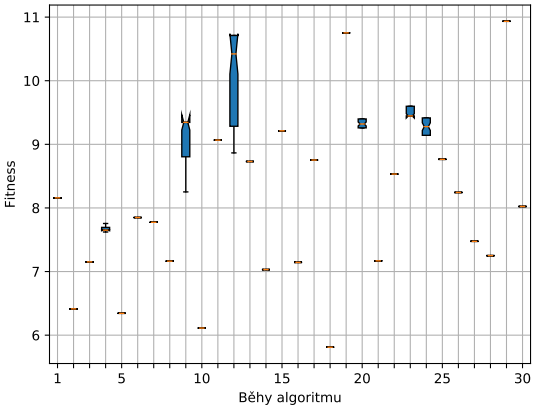
- Velikost populace - 64.
- Turnajová selekce.
- Dvoubodové křížení - 80%.
- Mutace - 15%.

- Mezigenerační přežití povoleno.

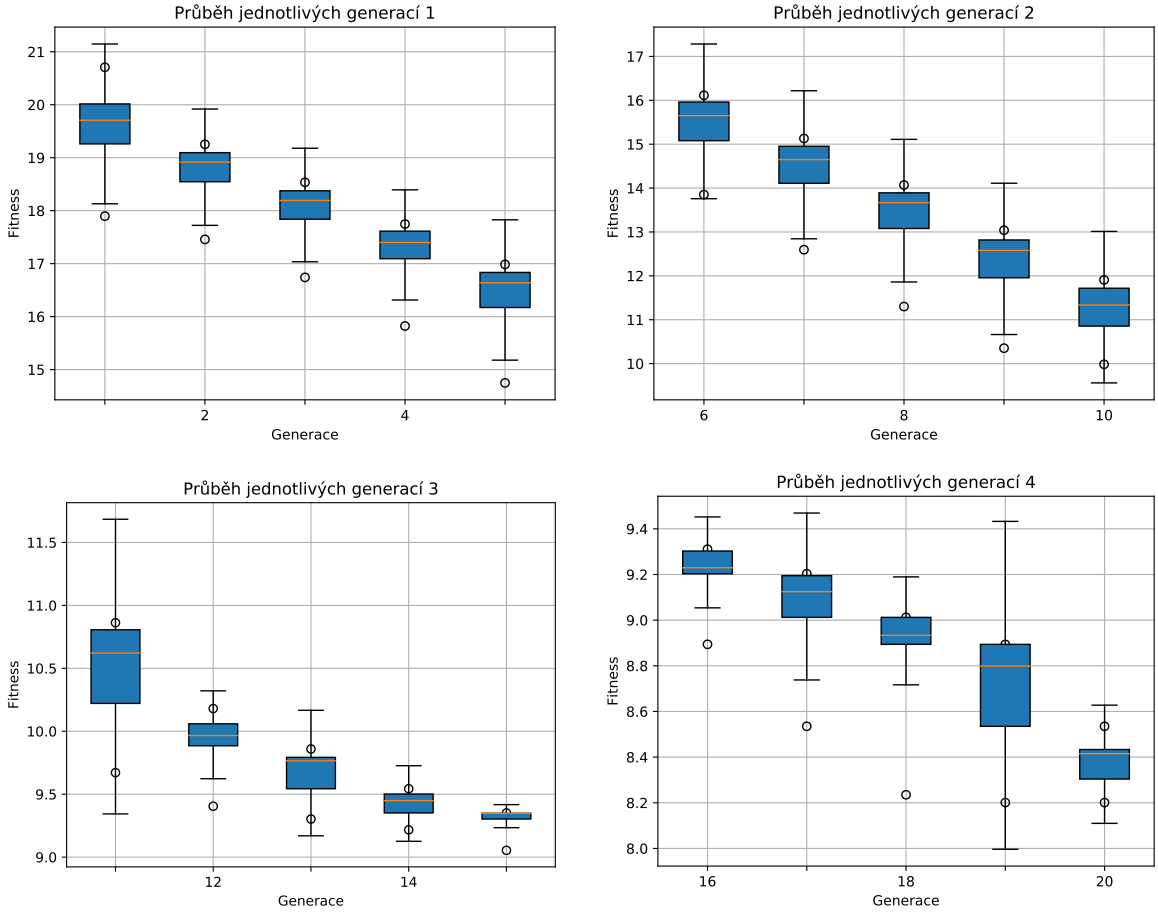
Počet generací byl omezen maximálním počtem vyhodnocení fitness funkce.



Obrázek A.4: Boxplot nejlepších výsledků všech 30 běhů GA.



Obrázek A.5: Boxplot stavu poslední generace všech 30 běhů GA. Lze vidět, že populace často degenerovali.



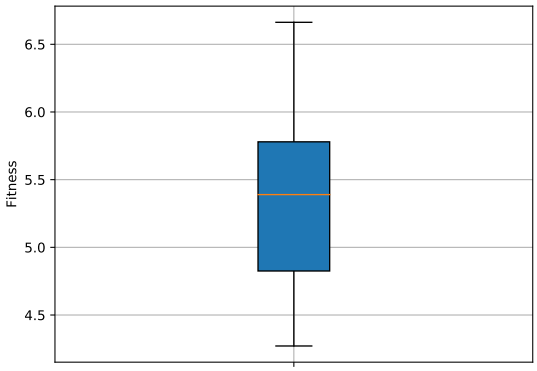
Obrázek A.6: Evoluční průběh GA. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Simulované žíhání

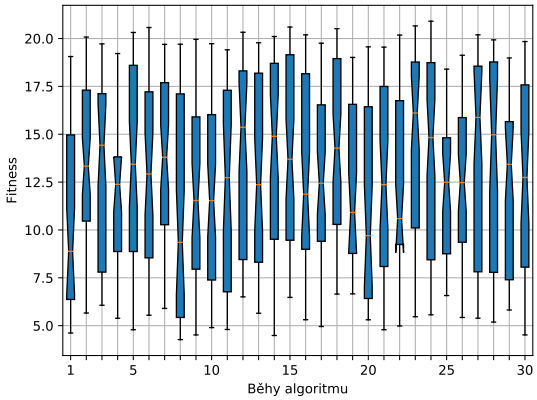
Pro nalezení řešení byly zvoleny následující parametry:

- Počáteční teplota - 2000.
- Chladící krok - 1%.

Teplotní rozvrh byl zvolen jako lineární. Perturberance maximálně o pětinu z rozsahu.



Obrázek A.7: Boxplot nejlepších výsledků všech 30 běhů SA.



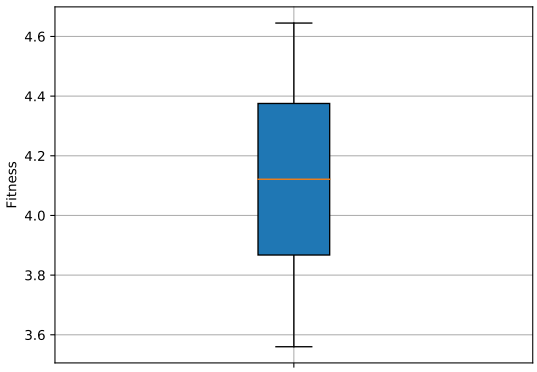
Obrázek A.8: Boxplot fitness hodnot navštívených stavů u všech 30 běhů SA.

Tabu prohledávání

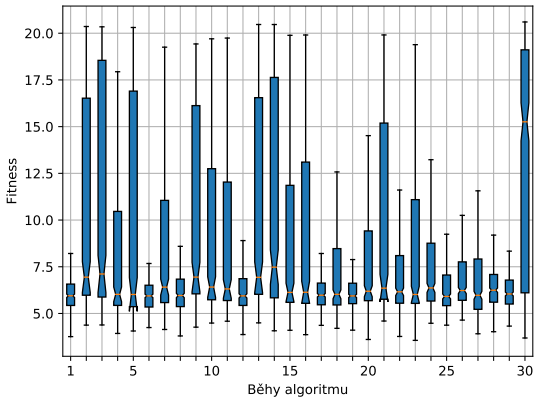
Pro nalezení řešení byly zvoleny následující parametry:

- Velikost tabu seznamu - 20 stavů.
- Velikost prohledávaného sousedství - 5 stavů

Použita pouze dlouhodobá paměť.



Obrázek A.9: Boxplot nejlepších výsledků všech 30 běhů Tabu prohledávání.



Obrázek A.10: Boxplot fitness hodnot navštívených stavů u všech 30 běhů Tabu prohledávání.

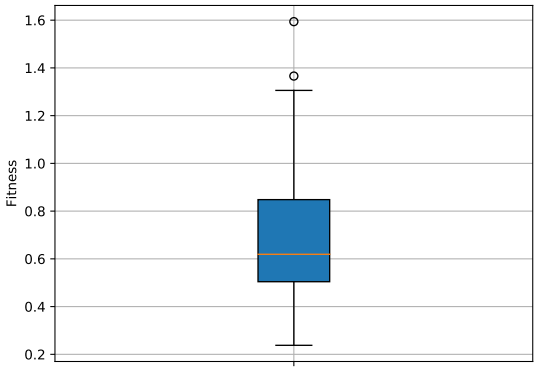
Diferenciální evoluce

Pro nalezení řešení byly zvoleny následující parametry:

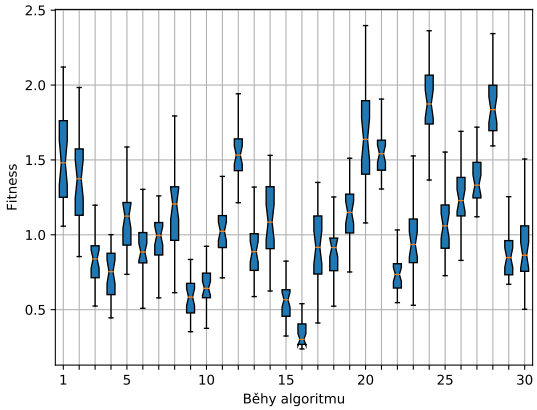
- Velikost populace - 40.
- Rekombinace 80%.
- Strategie - *best/1/bin*.

- Faktor zesílení - $0.5 - 0.9$.

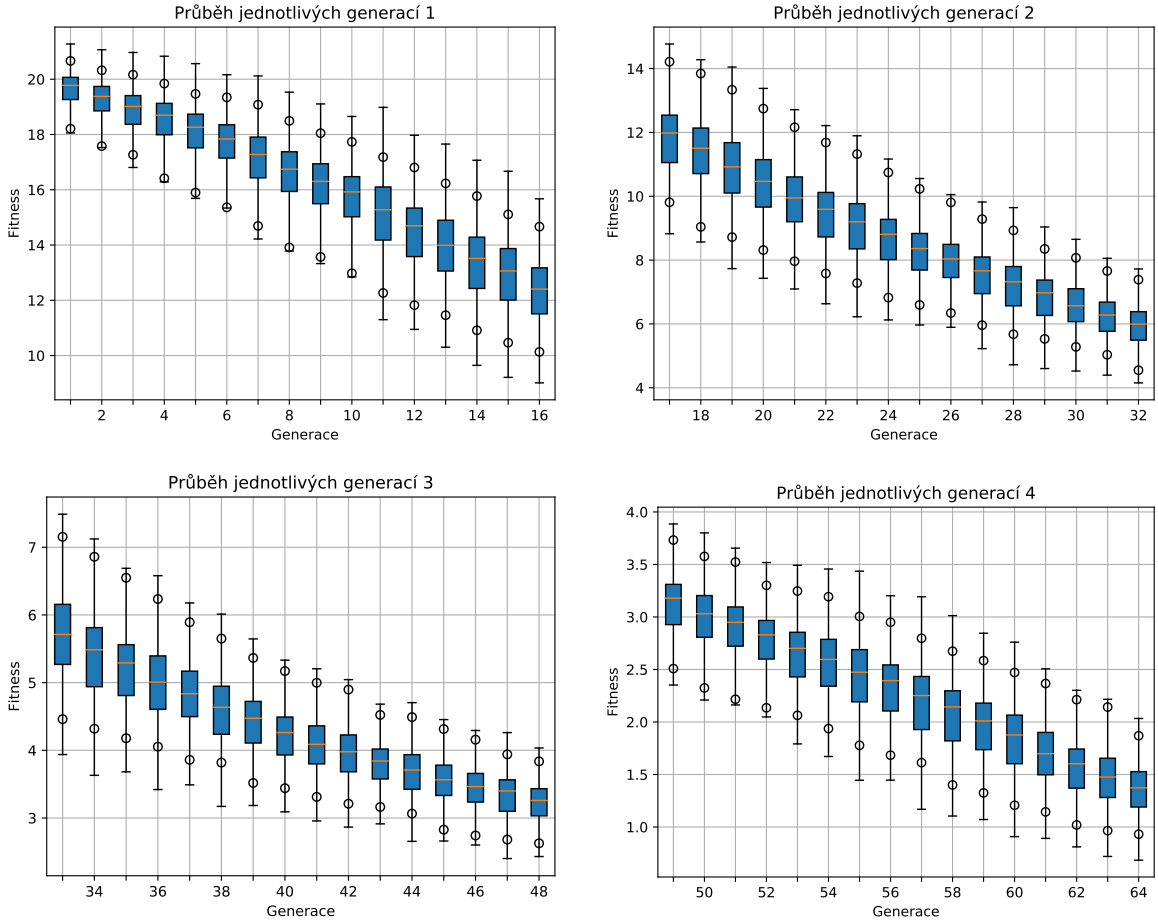
Počet generací nadhodnocen tak, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness.



Obrázek A.11: Boxplot nejlepších výsledků všech 30 běhů DE.



Obrázek A.12: Boxplot stavu poslední generace všech 30 běhů DE. Lze vidět, že populace jsou různorodé.



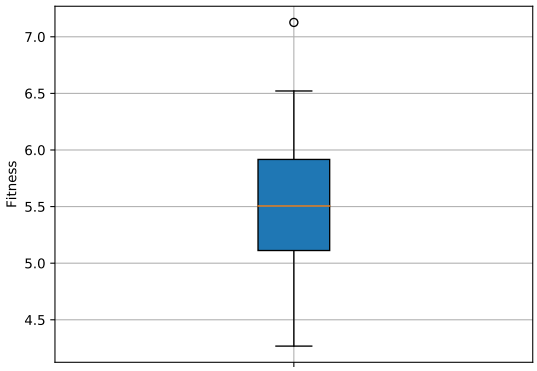
Obrázek A.13: Evoluční průběh DE. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Optimalizace rojem částic

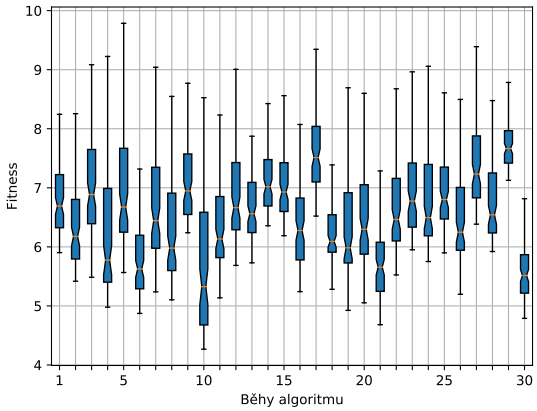
Pro optimalizaci byly použity tyto parametry:

- Velikost populace - 100 částic.
- Akcelerační koeficienty $c1 = c2 = 2$.
- Koeficient setrvačnosti 0.5.
- Úplná topologie.

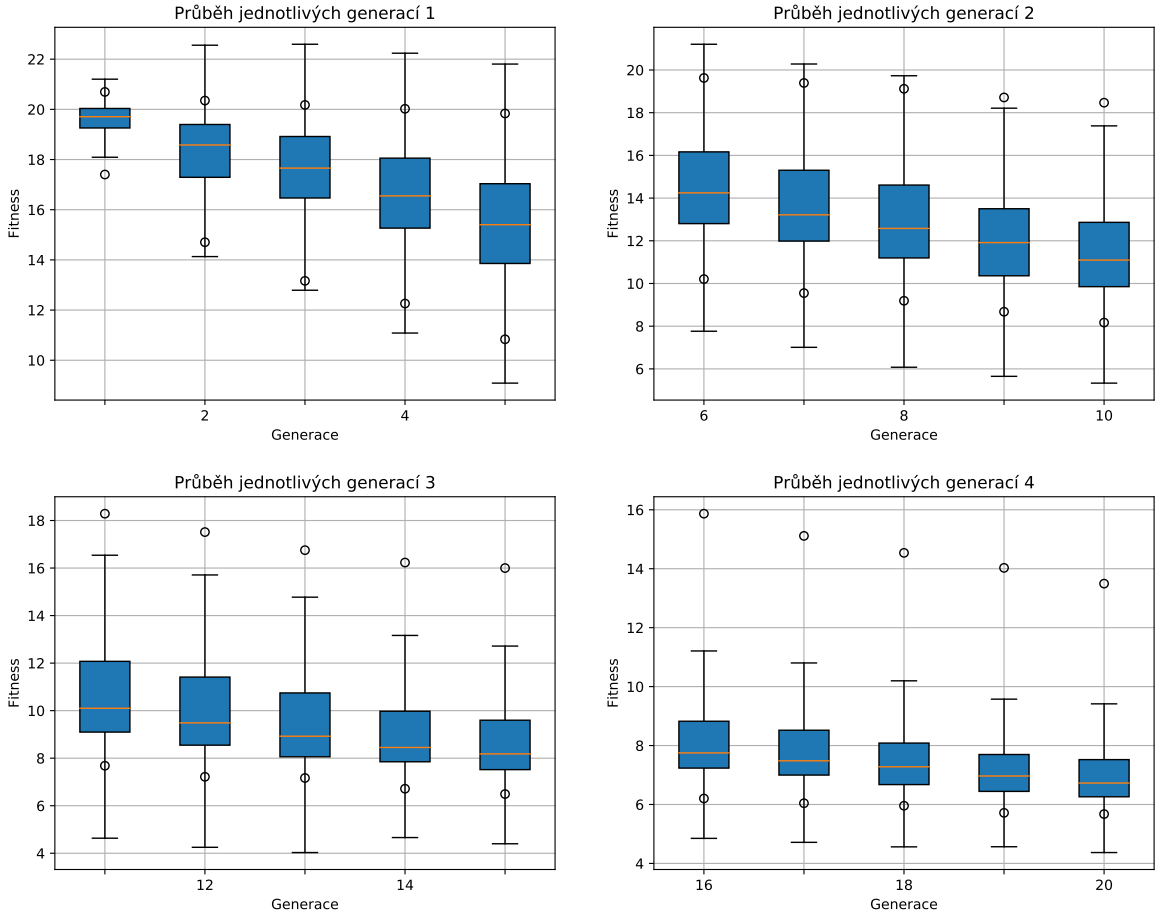
Počet generací nadhodnocen, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness.



Obrázek A.14: Boxplot nejlepších výsledků všech 30 běhů PSO.



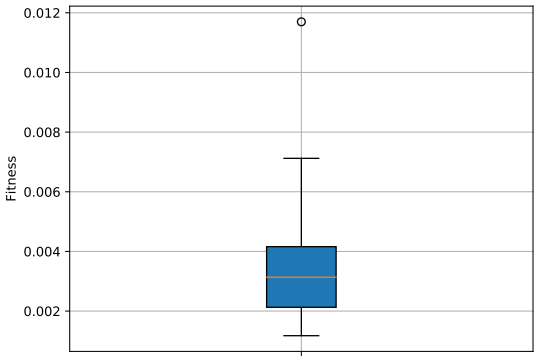
Obrázek A.15: Boxplot stavu poslední generace všech 30 běhů PSO. Lze vidět, že populace jsou různorodé.



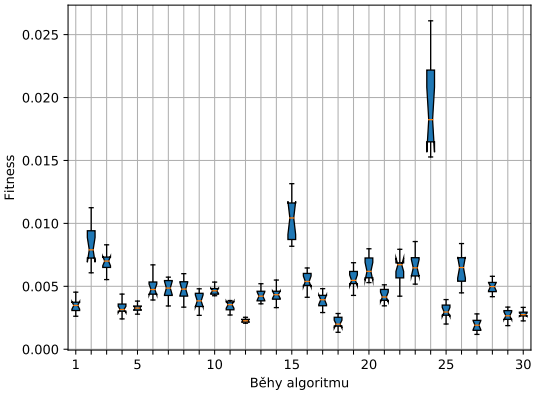
Obrázek A.16: Evoluční průběh PSO. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

CMA-ES

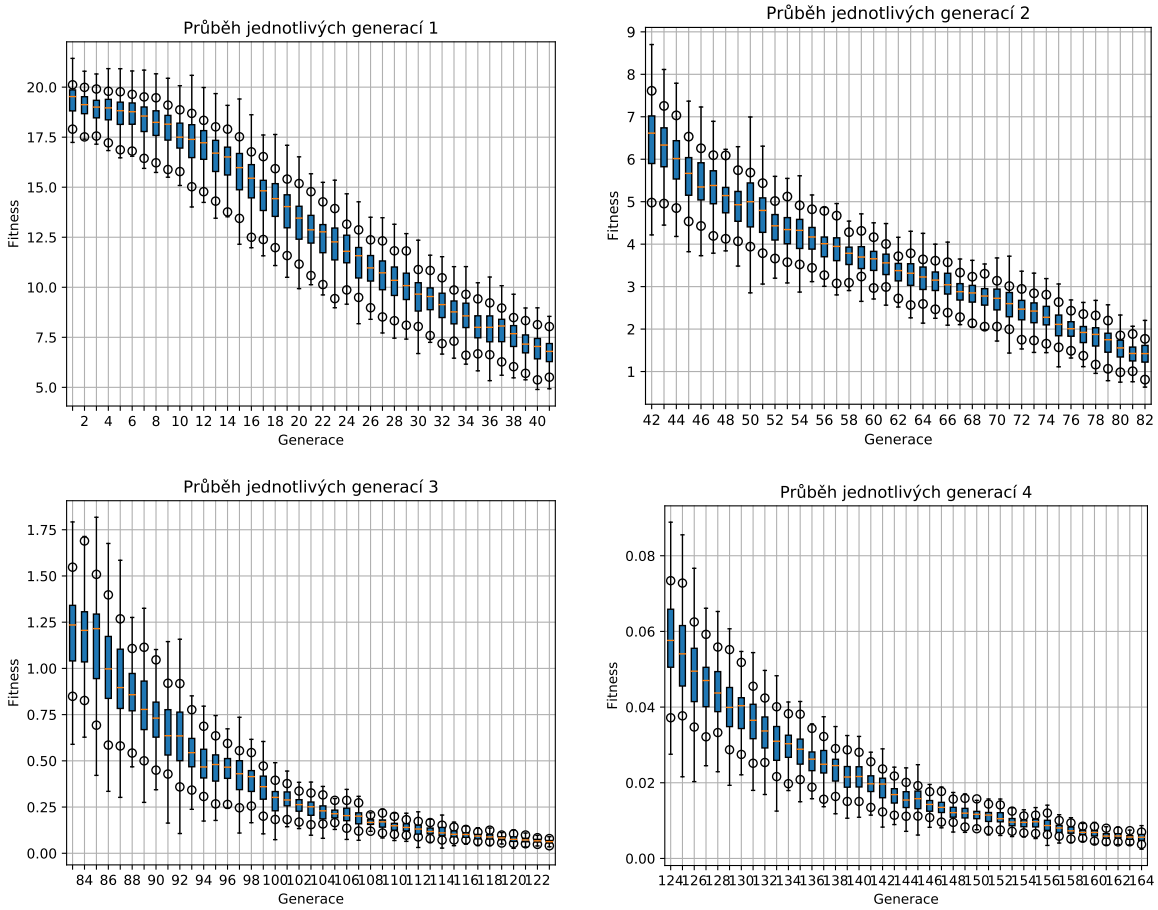
Výhodou *CMA-ES* je skutečnost, že téměř všechny vstupní parametry jsou závislé pouze na dimenzi problému, který je předem daný. Maximální počet generací nadhodnocen, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness. Počáteční průměrná hodnota byla určena jako střední hodnota v mezích proměnných, standardní odchylka jako třetina absolutní hodnoty tohoto rozmezí.



Obrázek A.17: Boxplot nejlepších výsledků všech 30 běhů CMA-ES.



Obrázek A.18: Boxplot stavu poslední generace všech 30 běhů CMA-ES. Lze vidět, že populace jsou různorodé.



Obrázek A.19: Evoluční průběh CMA-ES. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

A.2.2 Griewankova funkce

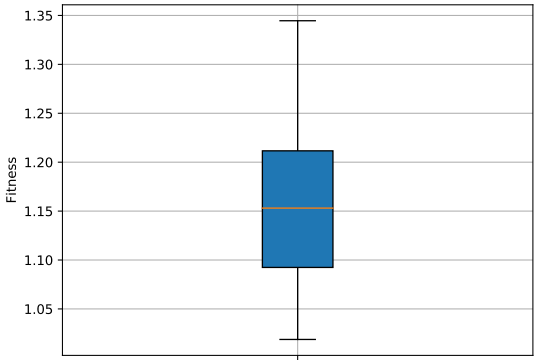
Statisticky zpracované výsledky 30 běhů každého optimalizačního algoritmu.

Genetický algoritmus

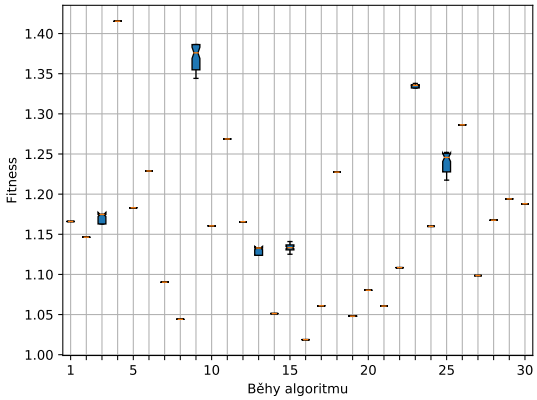
Pro optimalizace byly použity následující parametry:

- Velikost populace - 50.
- Turnajový výběr.
- Dvoubodové křížení - 80%.
- Mutace - 20%.
- Přežití nejlepších bez ohledu na generaci.

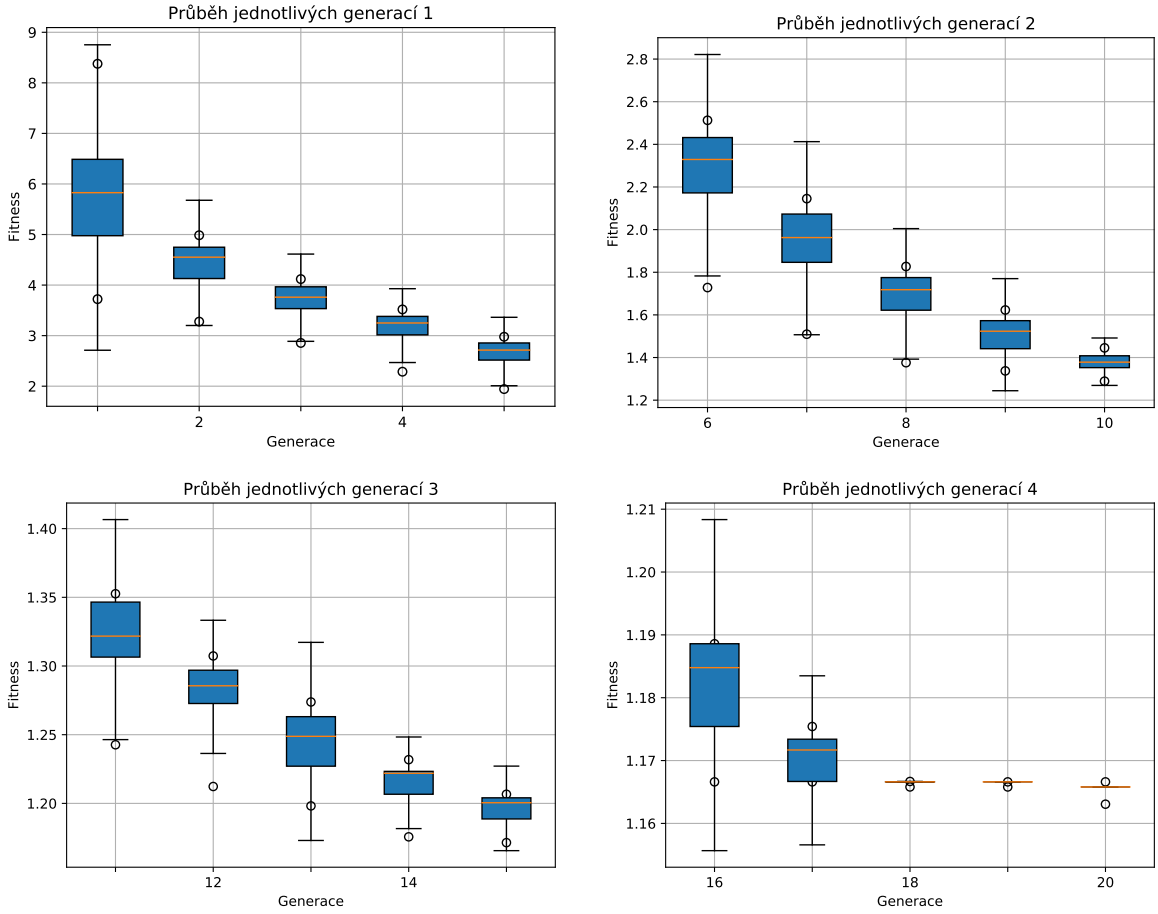
Maximální počet generací nadhodnocen, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness.



Obrázek A.20: Boxplot nejlepších výsledků všech 30 běhů GA.



Obrázek A.21: Boxplot stavu poslední generace všech 30 běhů GA.



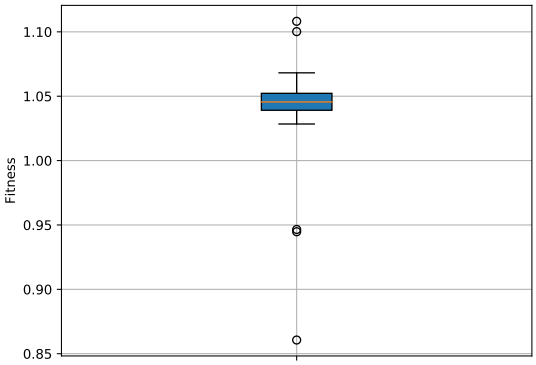
Obrázek A.22: Evoluční průběh GA. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Simulované žíhání

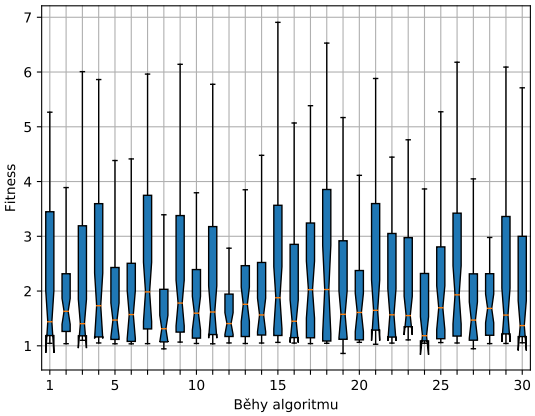
Pro nejoptimálnější řešení byly nalezeny parametry

- Počáteční teplota - 2000
- Chladící krok - 1%

Teplotní rozvrh byl zvolen jako lineární. Perturberance stavu je o číslo z Gaussova rozložení s průměrem 0 a odchylkou pětina rozmezí prostoru proměnné.



Obrázek A.23: Boxplot nejlepších výsledků všech 30 běhů SA.



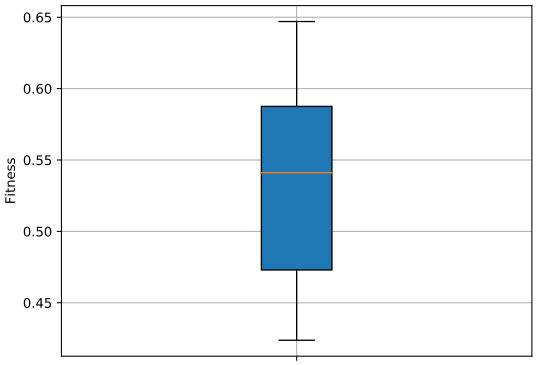
Obrázek A.24: Boxplot stavu poslední generace všech 30 běhů SA.

Tabu prohledávání

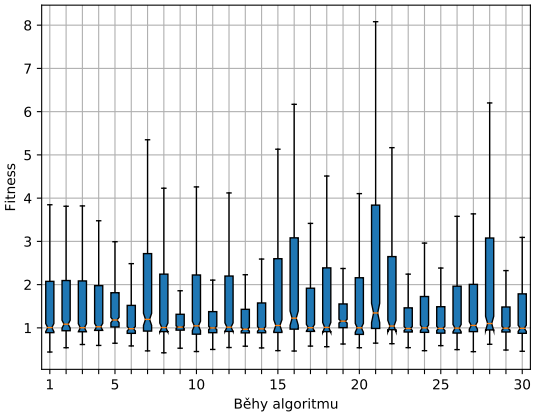
Pro nalezení řešení byly zvoleny následující parametry:

- Velikost tabu seznamu - 20
- Velikost prohledávaného sousedství - 5

Použita pouze dlouhodobá paměť.



Obrázek A.25: Boxplot nejlepších výsledků všech 30 běhů Tabu prohledávání.



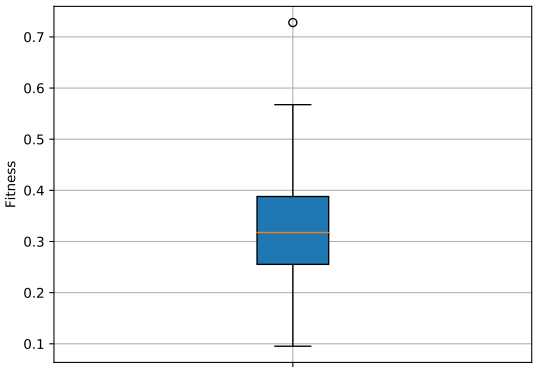
Obrázek A.26: Boxplot stavu poslední generace všech 30 běhů Tabu prohledávání.

Diferenciální evoluce

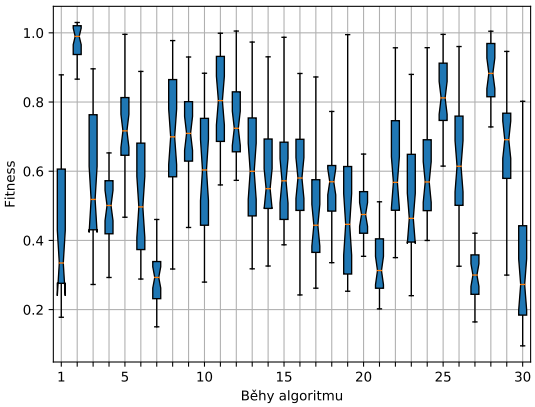
Pro optimalizaci byly použity tyto parametry:

- Velikost populace - 40.
- Rekombinace - 75%.
- Strategie *rand-to-best/1/bin*.
- Faktor zesílení 0.5.

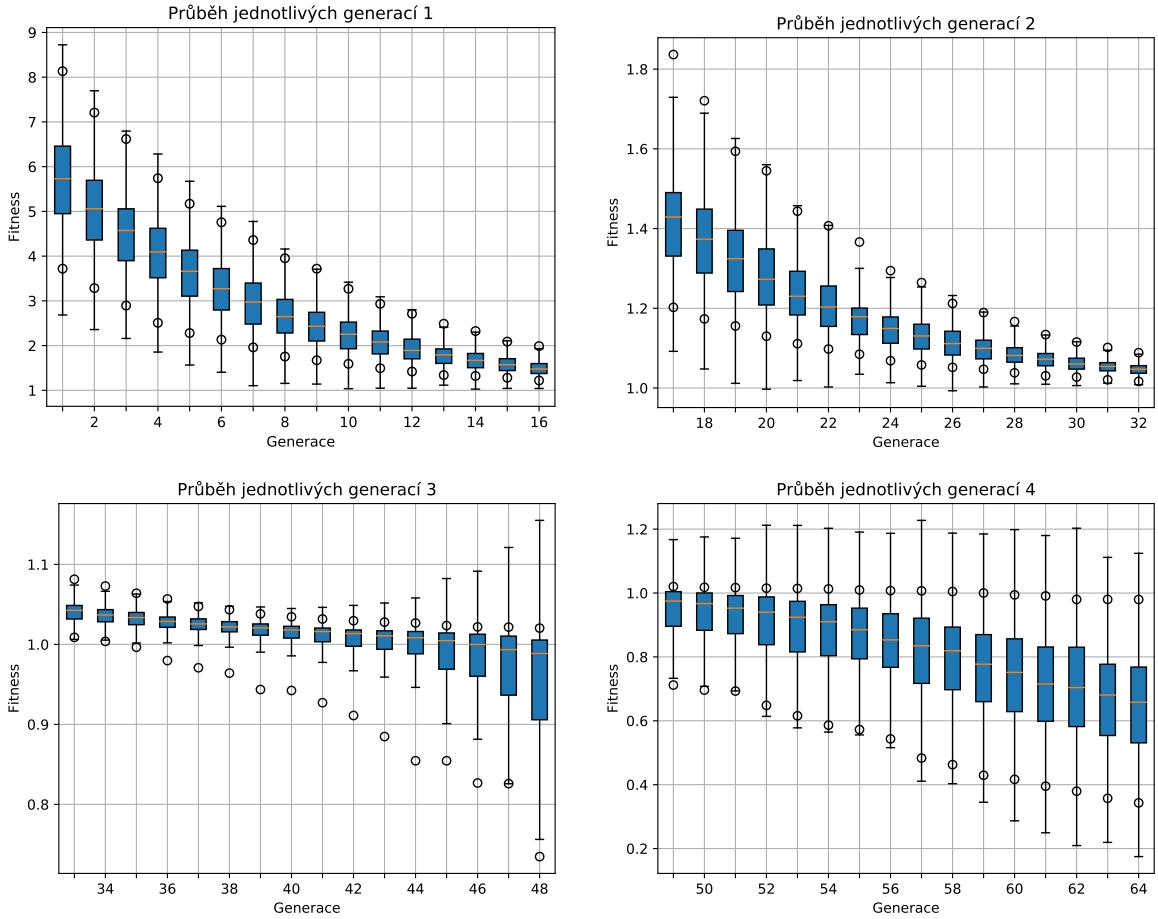
I zde byl počet generací nadhodnocen tak, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness.



Obrázek A.27: Boxplot nejlepších výsledků všech 30 běhů DE.



Obrázek A.28: Boxplot stavu poslední generace všech 30 běhů DE.



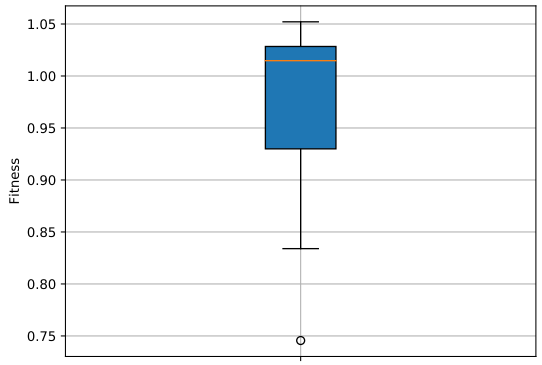
Obrázek A.29: Evoluční průběh DE. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Optimalizace rojem částic

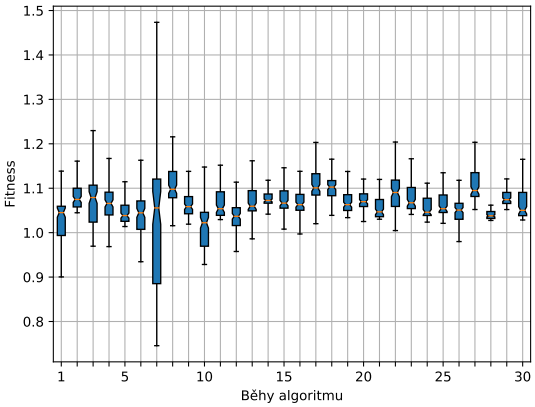
Pro nejoptimálnější řešení byly nalezeny parametry

- Velikost populace - 75.
- Akcelerační koeficienty $c1 = c2 = 2$.
- Váhový faktor setrvačnosti 0.5.
- Roj s úplnou topologií.

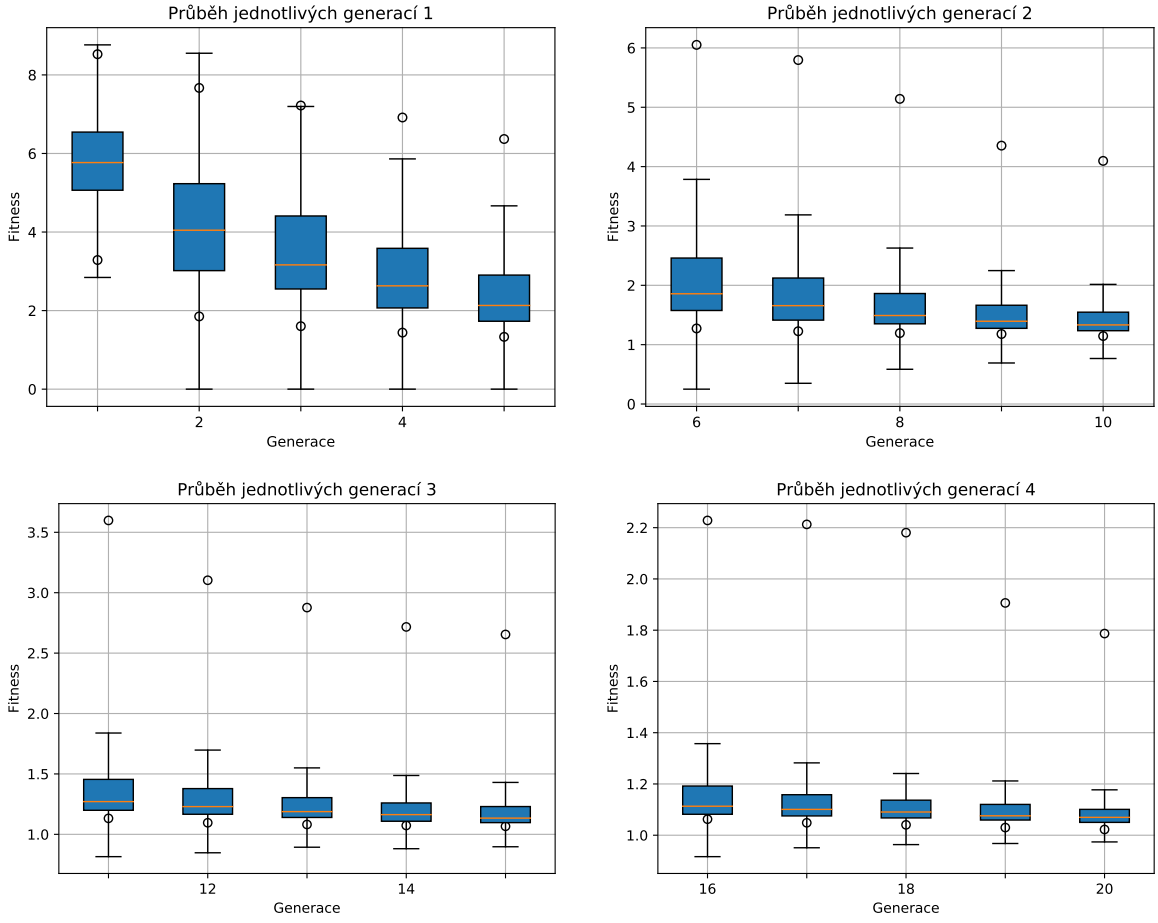
Počet generací nadhodnocen, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness.



Obrázek A.30: Boxplot nejlepších výsledků všech 30 běhů PSO.



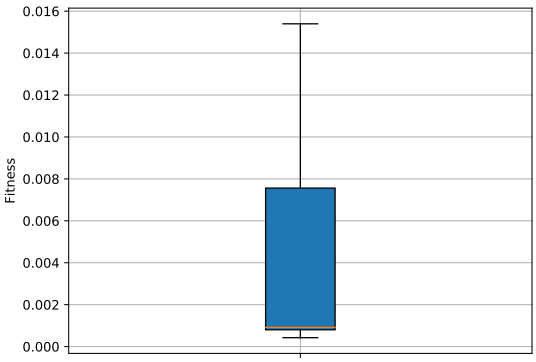
Obrázek A.31: Boxplot stavu poslední generace všech 30 běhů PSO.



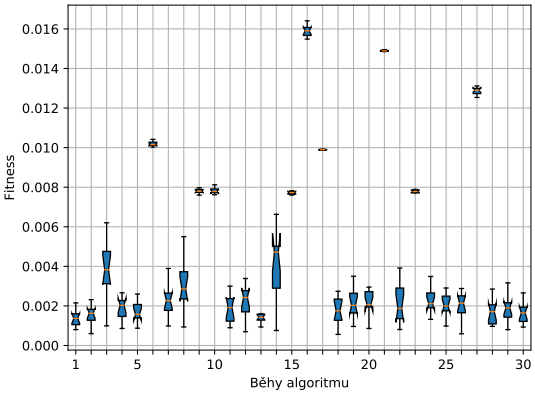
Obrázek A.32: Evoluční průběh PSO. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

CMA-ES

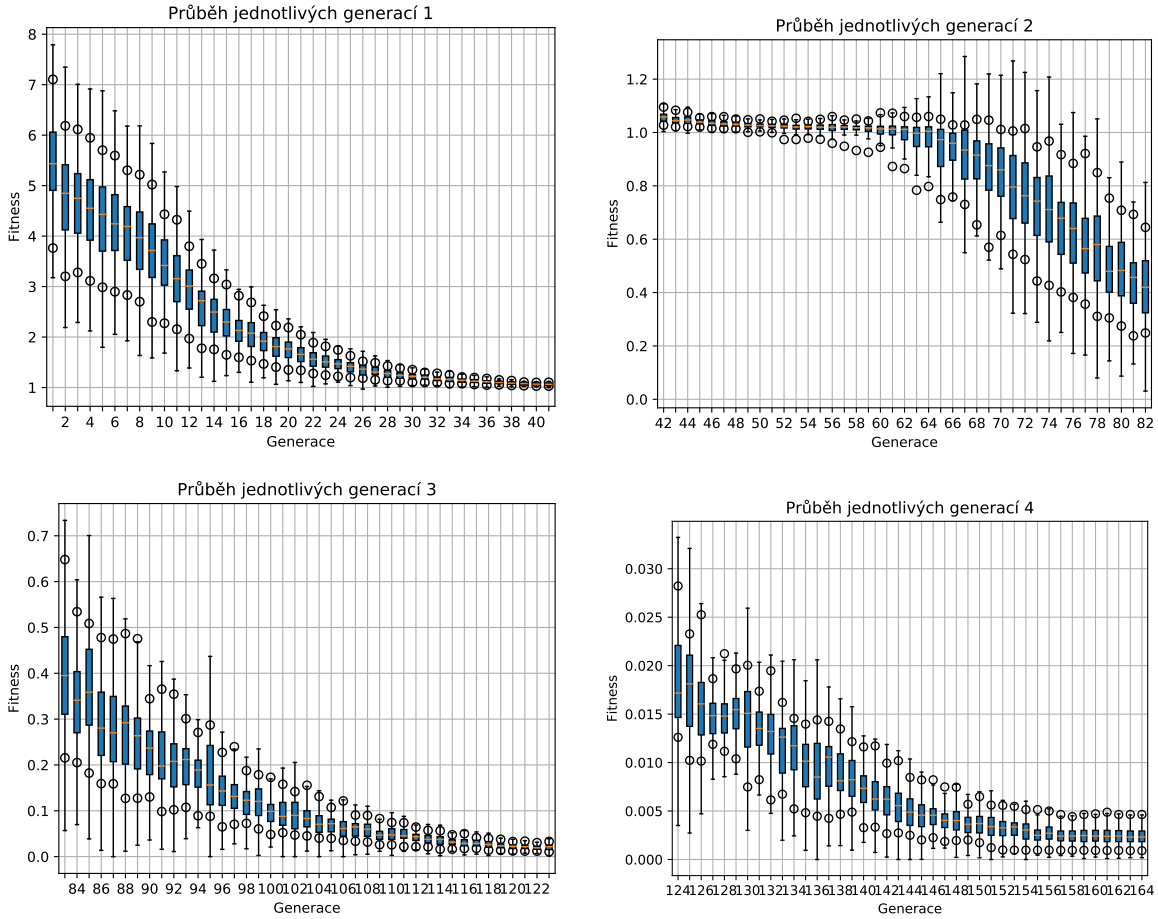
Výhodou *CMA-ES* je skutečnost, že téměř všechny vstupní parametry jsou závislé pouze na dimenzi problému, který je předem daný. Maximální počet generací nadhodnocen, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness. Počáteční průměrná hodnota byla určena jako střední hodnota v mezích proměnných, standardní odchylka jako třetina absolutní hodnoty tohoto rozmezí.



Obrázek A.33: Boxplot nejlepších výsledků všech 30 běhů CMA-ES.



Obrázek A.34: Boxplot stavu poslední generace všech 30 běhů CMA-ES.



Obrázek A.35: Evoluční průběh CMA-ES. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

A.2.3 Rosenbrockova funkce

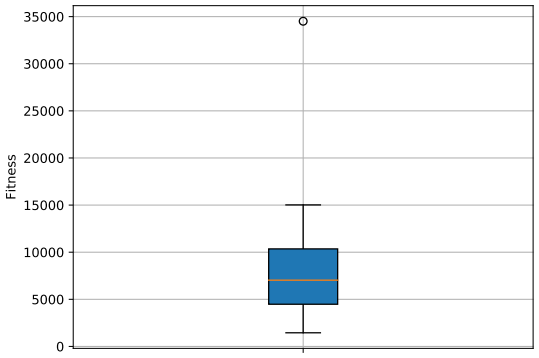
Následující grafy jsou kolekcí výsledků 30 běhů každého optimalizačního algoritmu.

Genetický algoritmus

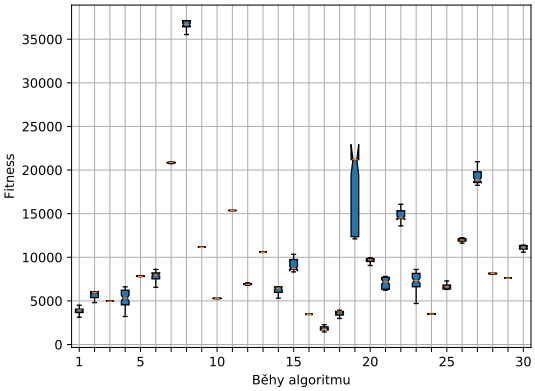
Pro optimalizaci byly použity následující parametry:

- Velikost populace - 64.
- Turnajový výběr.
- Dvoubodové křížení - 80%
- Mutace - 5%
- Vždy přežívají nejlepší řešení bez ohledu na to, ze které generace pochází.

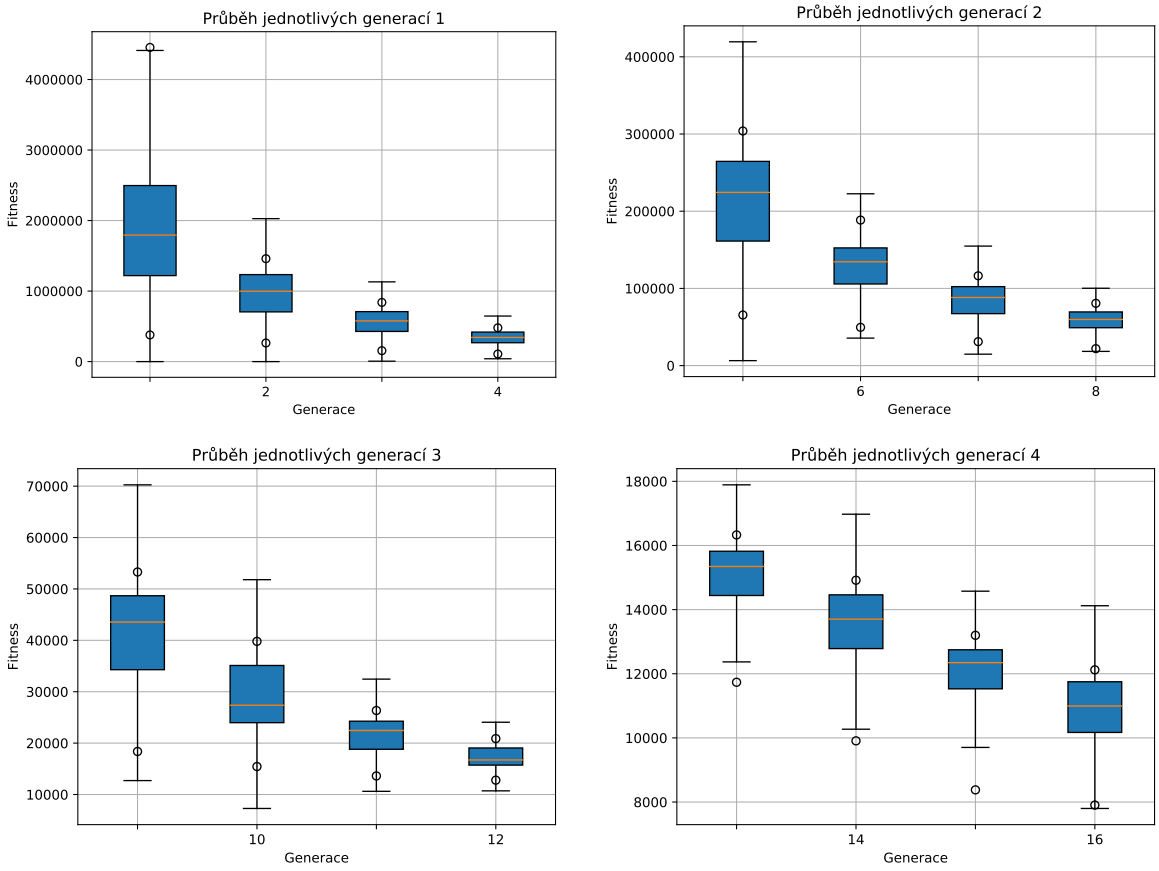
Vzhledem k omezení na maximální počet evaluací byl počet generací fixně nadhodnocen, aby nebyl nikdy důvodem k zastavení algoritmu.



Obrázek A.36: Boxplot nejlepších výsledků všech 30 běhů GA.



Obrázek A.37: Boxplot stavu poslední generace všech 30 běhů GA.



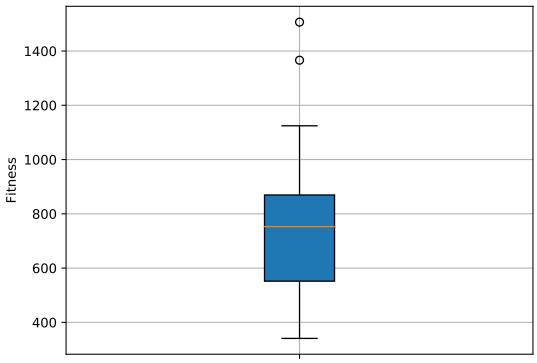
Obrázek A.38: Evoluční průběh GA. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Simulované žíhání

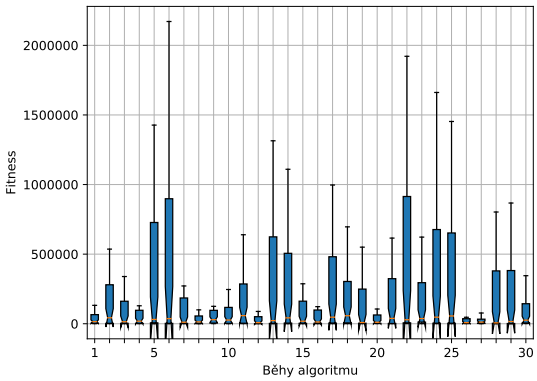
Pro optimalizace řešení byly zvoleny následující parametry:

- Počáteční teplota 2000.
- Chladící krok 1%.

Teplotní rozvrh byl zvolen jako lineární. Perturberance maximálně o pětinu rozsahu.



Obrázek A.39: Boxplot nejlepších výsledků všech 30 běhů SA.



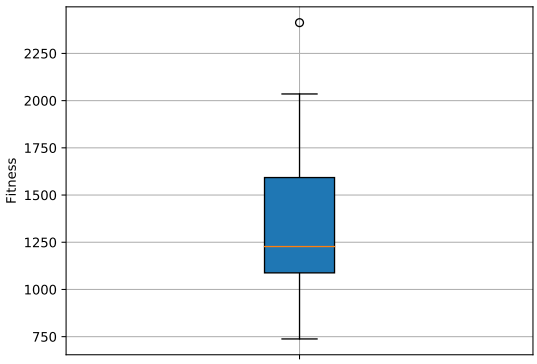
Obrázek A.40: Boxplot stavu poslední generace všech 30 běhů SA.

Tabu prohledávání

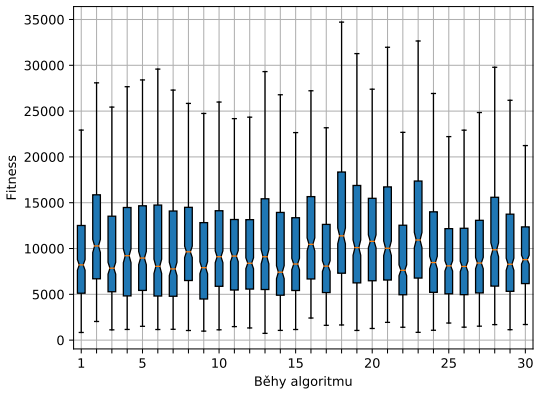
Optimalizace spouštěna s těmito parametry:

- Velikost tabu seznamu - 20.
- Velikost prohledávaného sousedství - 5.

Využita byla pouze dlouhodobá paměť.



Obrázek A.41: Boxplot nejlepších výsledků všech 30 běhů tabu prohledávání.



Obrázek A.42: Boxplot stavu poslední generace všech 30 běhů tabu prohledávání.

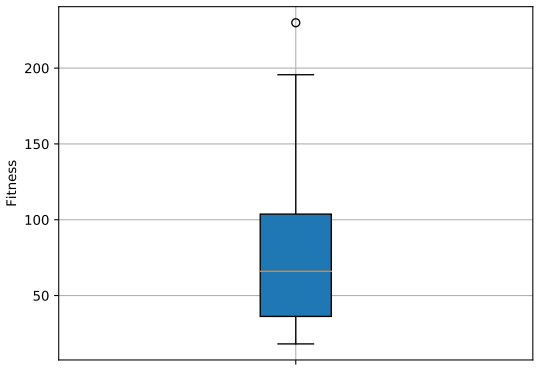
Diferenciální evoluce

Pro optimalizaci byly použity tyto parametry:

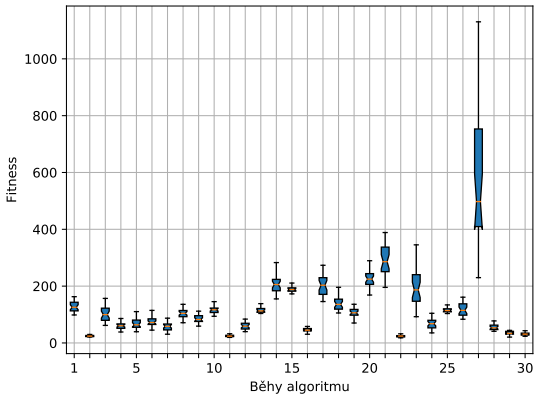
- Velikost populace - 30.

- Rekombinace - 80%.
- Strategie *best/1/bin*.
- Faktor zesílení 0.5 až 0.7.

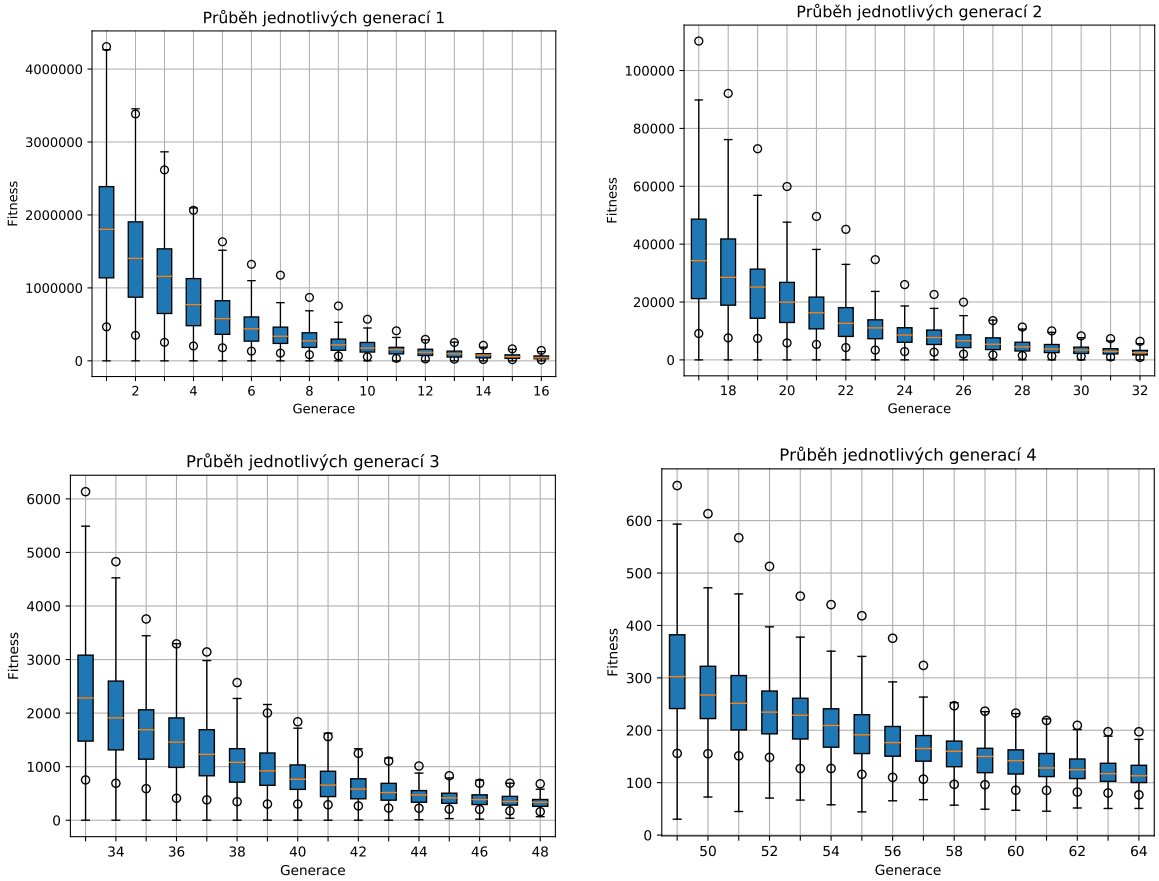
I zde byl počet generací nadhodnocen tak, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness.



Obrázek A.43: Boxplot nejlepších výsledků všech 30 běhů DE.



Obrázek A.44: Boxplot stavu poslední generace všech 30 DE.



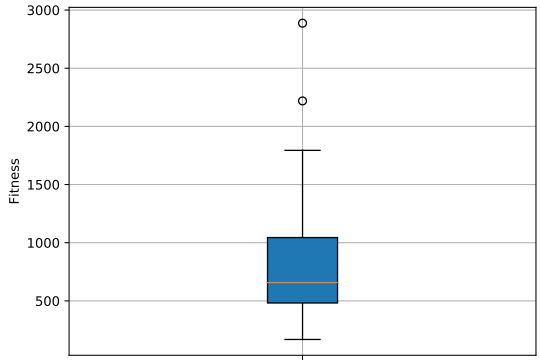
Obrázek A.45: Evoluční průběh DE. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Optimalizace rojem částic

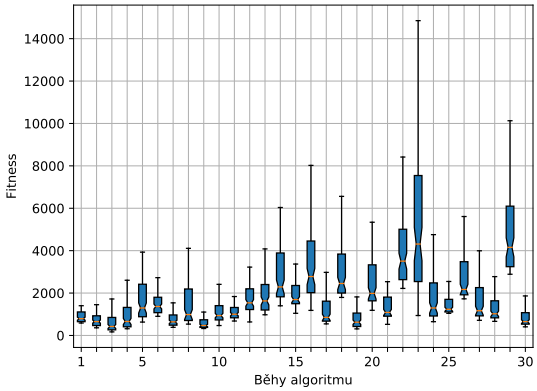
Pro nejoptimálnější řešení byly nalezeny parametry

- 75 částic v roji.
- Akcelerační koeficienty $c1 = c2 = 1$
- Váhový faktor akcelerace - 0.5

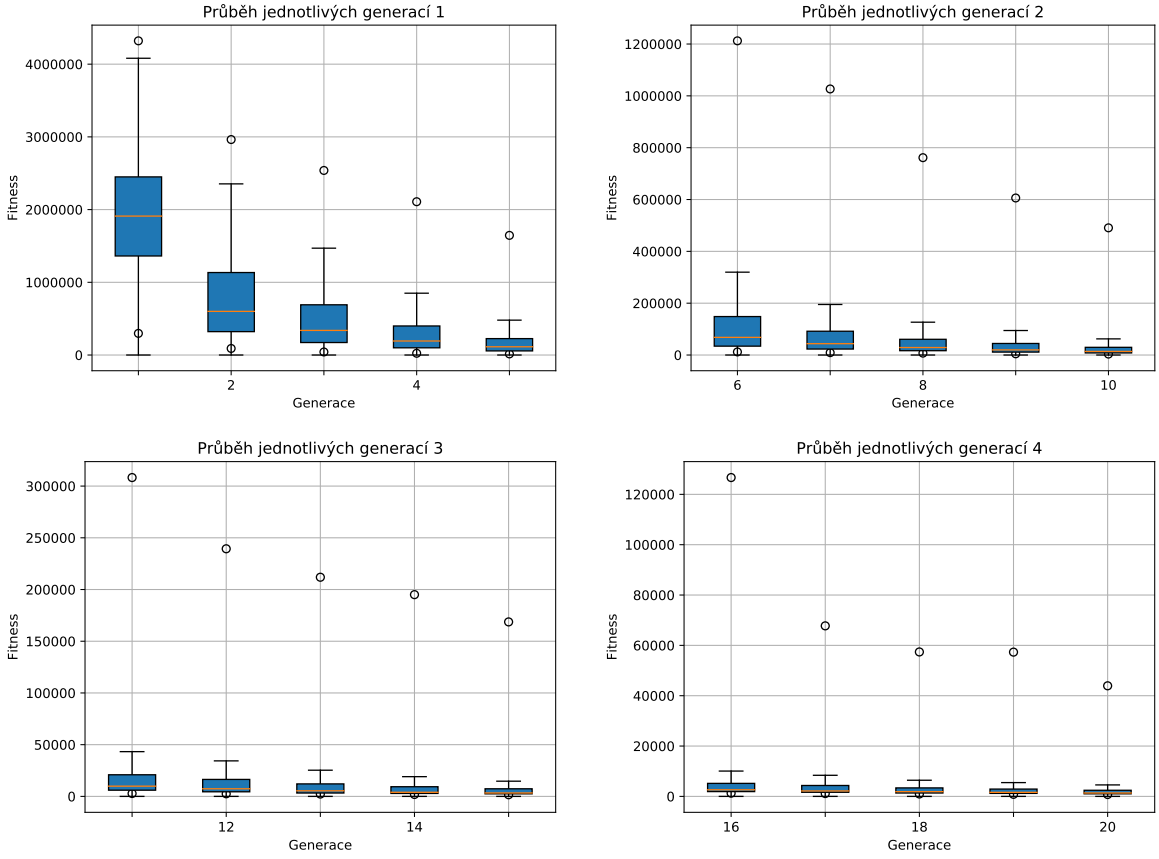
Počet generací nadhodnocen, aby zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness.



Obrázek A.46: Boxplot nejlepších výsledků všech 30 běhů PSO.



Obrázek A.47: Boxplot stavu poslední generace všech 30 PSO.

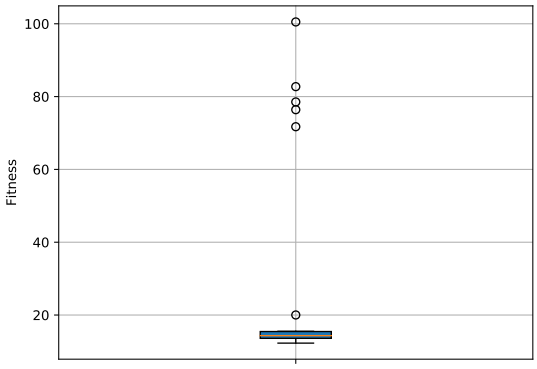


Obrázek A.48: Evoluční průběh PSO. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

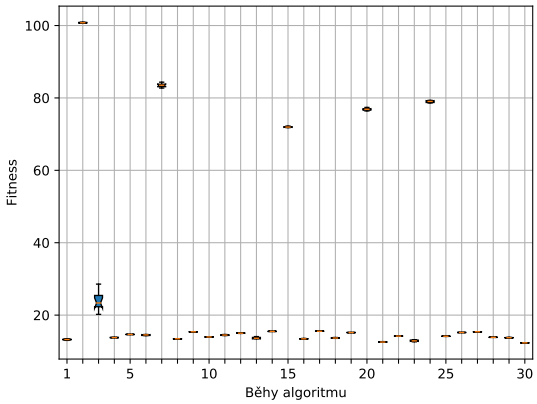
CMA-ES

Výhodou *CMA-ES* je skutečnost, že téměř všechny vstupní parametry jsou závislé pouze na dimenzi problému, který je předem daný. Maximální počet generací nadhodnocen, aby

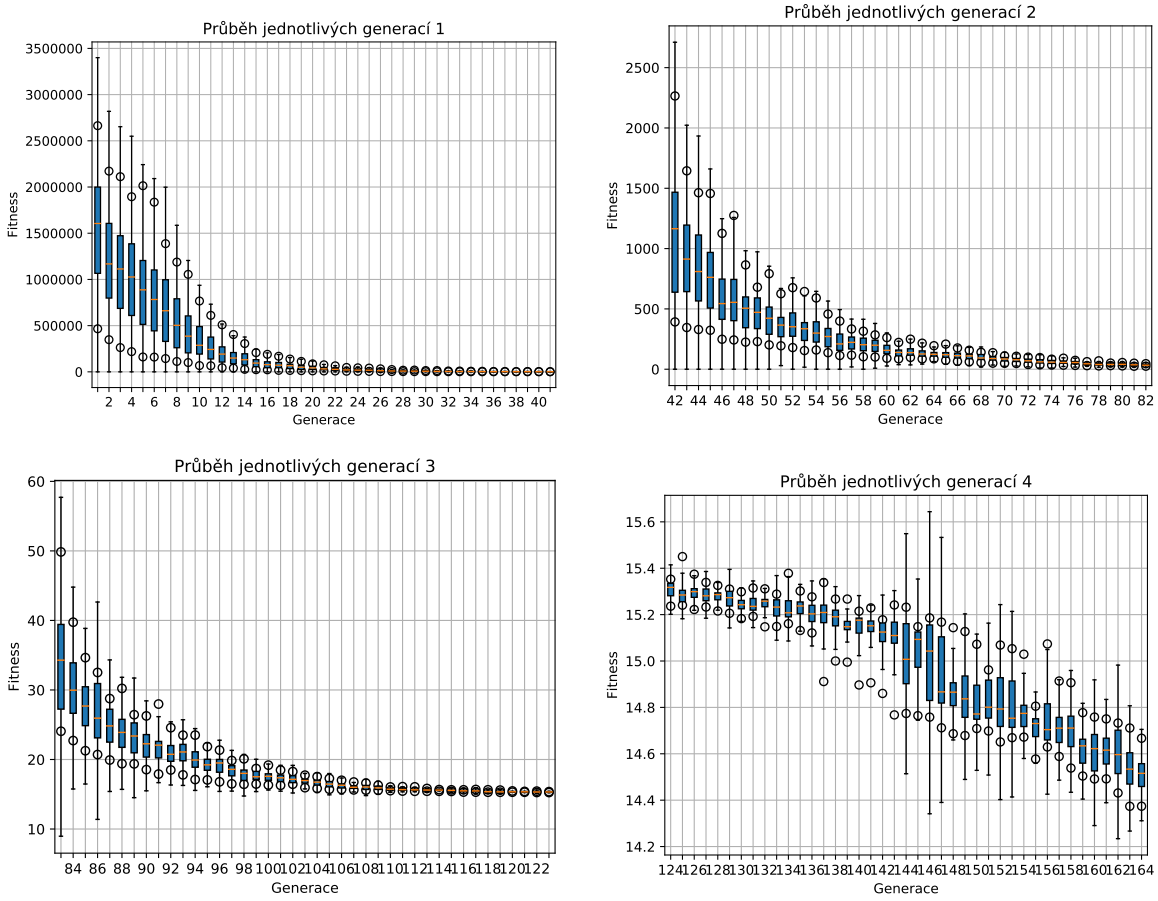
zastavení proběhlo nalezením optima nebo dosažením maximálního množství vyhodnocení fitness. Počáteční průměrná hodnota byla určena jako střední hodnota v mezích proměnných, standardní odchylka jako třetina absolutní hodnoty tohoto rozmezí.



Obrázek A.49: Boxplot nejlepších výsledků všech 30 běhů CMA-ES.



Obrázek A.50: Boxplot stavu poslední generace všech 30 CMA-ES.



Obrázek A.51: Evoluční průběh CMA-ES. Data jsou mediány ze všech běhů v konkrétních generacích. Graf rozdělen na čtvrtiny pro větší detail. Body ukazují medián nejlepších a nejhorších nalezených řešení v dané generaci.

Příloha B

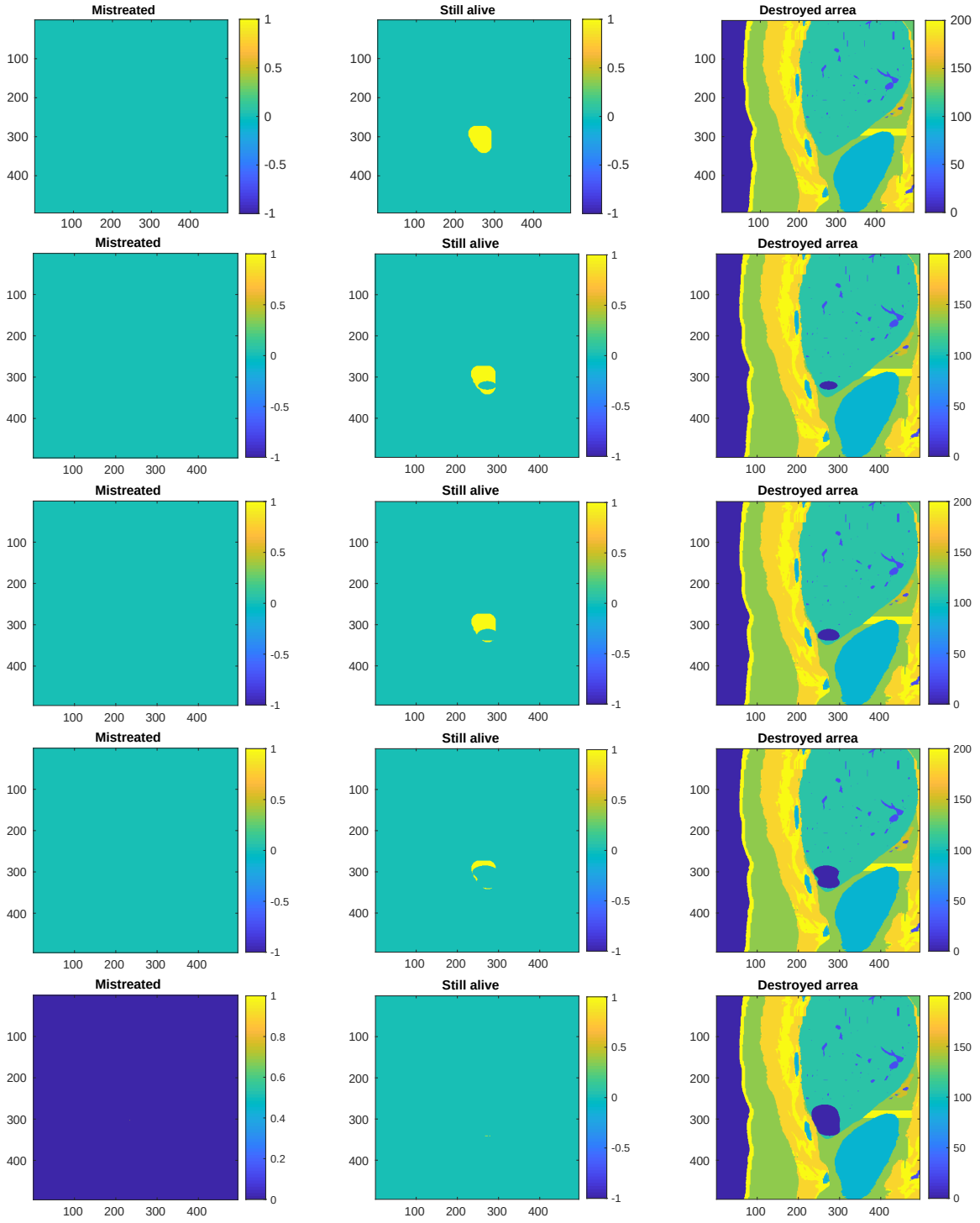
Vizualizace výsledků hledání HIFU trajektorie

Tato kapitola detailně zobrazuje nalezená nejlepší řešení při hledání trajektorie HIFU sonikací.

B.1 Skvrna - malý počet sonikací

Obrázek B.1 vizualizuje průběh simulace nejlepší nalezené trajektorie nad modelem skvrna o 4 sonikacích. Nejlepší nalezená trajektorie I_{best} skončila simulaci s následujícími výsledky:

- 0.30349% cílené tkáně přežilo - 3 body matice
- 0.0071788% necílené tkáně bylo zničeno - 1 bod matice

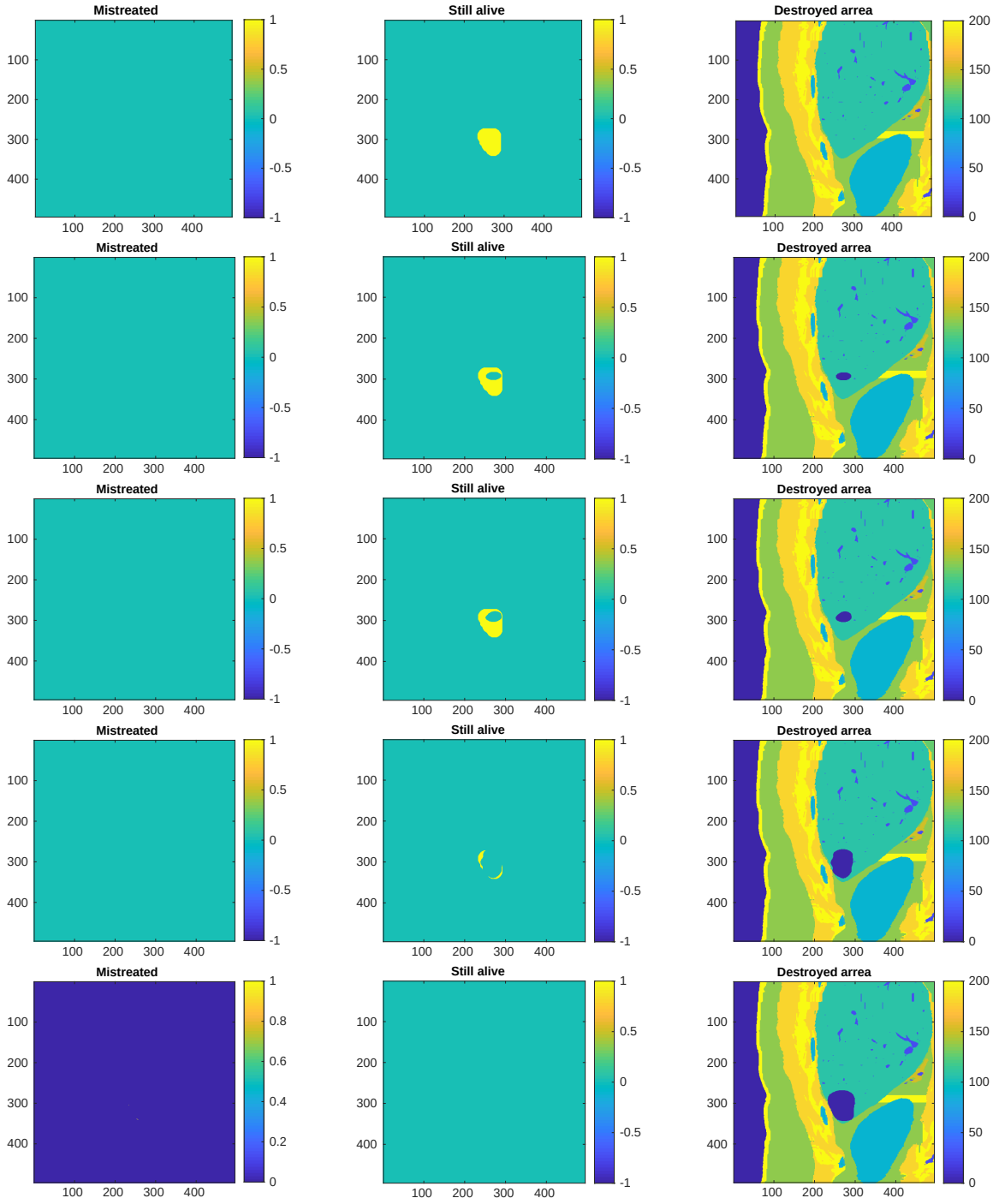


Obrázek B.1: Postupný průběh simulace nejlepšího nalezeného řešení. První řádek ukazuje počáteční stav modelu, každý další řádek poté reprezentuje stav po provedení jedné sonikace. Levý sloupec zobrazuje tkáň, která nebyla cílena a přesto proběhla ablace (při poslední sonikaci byl zasažen jeden bod, proto došlo ke změně kontury bez zřejmého důvodu). Prostřední sloupec ukazuje cílenou tkáň, která je stále naživu. Pravý sloupec ukazuje zničenou oblast na pozadí CT snímku použitého média.

B.2 Skvrna - velký počet sonikací

Obrázek B.2 vizualizuje průběh simulace nejlepší nalezené trajektorie nad modelem skvrna o 20 sonikacích. Nejlepší nalezená trajektorie I_{best} skončila simulaci s následujícími výsledky:

- 0% cílené tkáně přežilo
- 0.035894% necílené tkáně bylo zničeno - 5 bodů matice

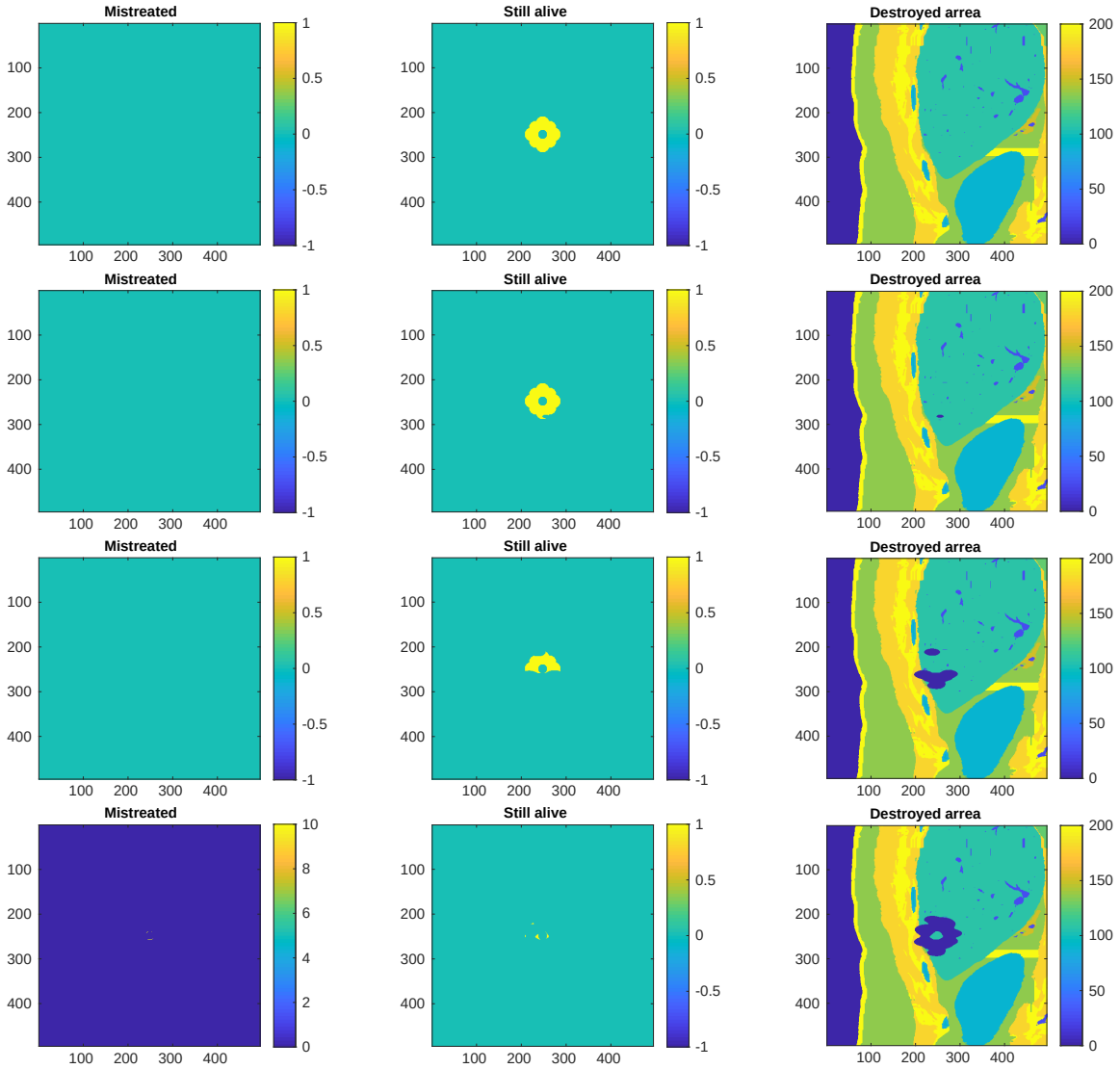


Obrázek B.2: Postupný průběh simulace nejlepšího nalezeného řešení HIFU operace. První řádek ukazuje stav modelu před začátkem operace, každý další řádek poté reprezentuje stav modelu po provedení následujících 5ti sonikací. Levý sloupec zobrazuje tkáň, která nebyla cílena a přesto proběhla ablace (při poslední sonikaci byl zasaženo několik bod, proto došlo ke změně kontury bez zřejmého důvodu). Prostřední sloupec ukazuje cílenou tkáň, která je stále naživu. Pravý sloupec ukazuje zničenou oblast na pozadí CT snímku použitého média.

B.3 Květina

Obrázek B.3 vizualizuje průběh simulace nejlepší nalezené trajektorie nad modelem skvrna o 20 sonikacích. Nejlepší nalezená trajektorie I_{best} skončila simulaci s následujícími výsledky:

- 2.7664% cílené tkáně přežilo - 108 bodů matice
- 0.36112% necílené tkáně bylo zničeno - 160 bodů matice



Obrázek B.3: Postupný průběh simulace nejlepšího nalezeného řešení HIFU operace. První řádek je počáteční stav modelu před začátkem operace, každý další řádek reprezentuje provedení následujících 5ti sonikací. Levý sloupec zobrazuje tkáň, která nebyla cílena a přesto proběhla ablace. Prostřední sloupec ukazuje cílenou tkáň, která je stále naživu. Pravý sloupec ukazuje zničenou oblast na pozadí CT snímku použitého média.