MASARYKOVA UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
INSTITUT BIOSTATISTIKY A ANALÝZ LF A PŘF MU

Bakalářská práce

BRNO 2015 **VENDULA CHUROVÁ**

MASARYKOVA UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
INSTITUT BIOSTATISTIKY A ANALÝZ LF A PŘF MU

Optimalizační algoritmy inspirované živou přírodou

Bakalářská práce

Vendula Churová

Vedoucí práce: Mgr. et Mgr. Petr Dluhoš **Brno 2015**

Bibliografický záznam

Autor:	Vendula Churová Přírodovědecká fakulta, Masarykova univerzita Institut biostatistiky a analýz LF a PřF MU Centrum pro výzkum toxických látek v životním prostředí

Název práce:	Optimalizační algoritmy inspirované živou přírodou
Studijní program:	Experimentální biologie
Studijní obor:	Matematická biologie
Vedoucí práce:	Mgr. et Mgr. Petr Dluhoš
Akademický rok:	2014/2015
Počet stran:	$61 + 12$
Klíčová slova:	Optimalizace, metaheuristické algoritmy, Evoluční strategie, Algoritmus umělých včelích kolonií, registrace medicínských obrazů; afinní registrace

Bibliographic Entry

Author:	Vendula Churová Faculty of Science, Masaryk University Department of Biostatistics and Analyses MU Research Centre for Toxic Compounds in the Environments

Title of Thesis:	Bio-inspired metaheuristic optimization
Degree Programme:	Experimental biology
Field of Study:	Computational biology
Supervisor:	Mgr. et Mgr. Petr Dluhoš
Academic Year:	2014/2015
Number of Pages:	$61 + 12$
Keywords:	Optimization, metaheuristics, Evolution strategies, Artificial bee colony, medical image registration, affine registration

Abstrakt

Bakalářská práce se věnuje přírodou inspirovaným metaheuristickým algoritmům a jejich aplikaci na registraci medicínských obrazů. Práce předkládá rešerši těch nejznámějších metaheuristických algoritmů se zaměřením na algoritmy Evolučních strategií a Umělých včelích kolonií. Tyto dva algoritmy jsou porovnány s jejich biologickou předlohou, implementovány v prostředí MATLAB a testovány na sadě dvourozměrných obrazů mozku z magnetické rezonance. Výsledky algoritmů jsou porovnány a diskutovány. Zdrojové kódy obou algoritmů a vytvořená sada testovacích obrazů jsou součástí přiloženého DVD.

Abstract

This thesis deals with bio-inspired metaheuristic algorithms and their application to the problem of medical image registration. Best known (existing) bio-inspired metaheuristics are reviewed. Main focus is on Evolution strategies and Artificial bee colony. Both algorithms are compared with their biological inspiration, implemented in software MATLAB and tested on two-dimensional image dataset of magnetic resonance images. Results of both algorithms are compared and their efficiency is discussed. The source code and used data set are attached on DVD.

Centrum pro výzkum
toxických látek
v prostředí

MU
IBA
Institut
biostatistiky
a analýz

Kamenice 126/3, 625 00 Brno, fax +420 549 49 2840, www.recetox.cz

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jméno studenta/ky: Vendula Churová **UČO:** 408291

Studijní program/Obor: Experimentální biologie/Matematická biologie

Název práce: Optimalizační algoritmy inspirované živou přírodou

Název práce anglicky: Bio-inspired optimization algorithms

Vedoucí: Mgr. et Mgr. Petr Dluhoš
Konzultant(i):

Datum zadání: 5.10.2014
Datum odevzdání: 25.5.2015
Předpokládaný rozsah: 40 – 60 stran

Postup a zásady pro vypracování:

Propojení světa matematiky a biologie nepřináší užitek jen ve formě aplikace matematických postupů na biologické problémy, ale existují také oblasti matematiky a informatiky, které čerpají svou inspiraci z živé přírody. Jednou z těchto oblastí je řešení optimalizačních úloh.
Vypracujte rešerši existujících optimalizačních algoritmů inspirovaných přírodou s důrazem na algoritmy adaptivní evoluční strategie (Self-Adaptation Evolution Strategy) a umělé včelí kolonie (Artificial Bee Colony Algorithm). Tyto dva algoritmy srovnejte s jejich biologickou předlohou a poté implementujte ve vybraném programovacím prostředí (doporučen MATLAB). Porovnejte jejich efektivitu na problému registrace (lícování) medicínských obrazů.

Centrum pro výzkum
toxických látek
v prostředí

MU
IBA
Institut
biostatistiky
a analýz

Kamenice 126/3, 625 00 Brno, fax: +420 549 49 2840, www.recetox.cz

Doporučená literatura:

- BEYER, H.-G., SCHWEFEL H.-P., 2002: Evolution Strategies – A Comprehensive Introduction. Natural Computing 1 (1), 3–52.
- KARABOGA, D., 2010: Artificial Bee Colony Algorithm. Scholarpedia 5 (3), 6915.

V Brně dne: 5.10.2014

Podpis vedoucího práce: .....

Podpis studenta/ky: .....

Podpis garanta oboru: .....
doc. RNDr. Ladislav Dušek, Ph.D.

Poděkování

Ráda bych poděkovala svému vedoucímu práce, panu Mgr. et Mgr. Petru Dluhošovi, za odbornou pomoc, cenné připomínky a zvláště pohodový a velmi trpělivý přístup. Neméně bych chtěla poděkovat za spoustu času, který mi byl ochoten věnovat.

V druhé řadě děkuji všem čtenářům pracovních verzí za jejich kritické připomínky a postřehy.

Prohlášení

Prohlašuji, že jsem svoji bakalářskou práci vypracovala samostatně s použitím citovaných zdrojů.

Brno 25. května 2015 .....

Vendula Churová

MASARYKOVA UNIVERZITA
INSTITUT BIOSTATISTIKY A ANALÝZ

Institut biostatistiky a analýz Lékařské a Přírodovědecké fakulty Masarykovy univerzity spolupracuje na organizačním zajištění výuky studijního oboru Matematická biologie s Centrem pro výzkum toxických látek v prostředí Přírodovědecké fakulty MU.

Obsah

Přehled použitého značení a zkratek

$\mathbb{R}$	množina všech reálných čísel (set of real numbers)
2D	dvourozměrný prostor (two-dimensional space)
3D	třírozměrný prostor (three-dimensional space)
ABC	Algoritmus umělých včelích kolonií[1] (Artificial Bee Colony)
ACO	Optimalizace mravenčí kolonií (Ant Colony Optimization)
AIS	Umělé imunitní systémy (Artificial Immune Systems)
CLONALG	Algoritmus klonové selekce[1] (Clonal Selection Algorithm)
CMA-ES	Kovarianční evoluční strategie[1] (Covariance Matrix Adaptation Evolution Strategy)
CS	Kukaččí algoritmus[1] (Cuckoo Search)
DE	Diferenciální evoluce (Differential Evolution)
DOF	Stupeň volnosti transformační funkce (Degrees of Freedom)
EA	Třída evolučních algoritmů (Evolutionary Algorithms)
EP	Evoluční programování (Evolution Programming)
ES	Evoluční strategie (Evoltion Strategies)
FA	Algoritmus svatojánských mušek[1] (Firefly Algorithm)
GA	Genetický algoritmus (Genetic Algorithm)
GP	Genetické programování (Genetic Programming)
IR	Registrace obrazů (Image Registration)
MGA	Složitý genetický algoritmus[1] (Messy Genetic Algortihm)
MH	Metaheuristické algoritmy (Metaheuristic algorithms)
MRI	Zobrazování magnetickou rezonancí (Magnetic Resonance Imaging)
MSE	Metoda nejmenších čtverců (Mean Square Error)
PSO	Optimalizace hejnem (Particle Swarm Optimization)
SI	Algoritmy rojové inteligence, populační (Swarm Intelligence)
SAES	Adaptivní verze Evolučních strategií (Self-Adaptation Evolution Strategy)
SOMA	Samoorganizující se migrační algoritmus (Self Organizing Migrating Algorithm)
TSP	Problém obchodního cestujícího (Travelling Salesman Problem)

[1] Volný překlad.

Kapitola 1

Úvod

Příroda se stala zdrojem inspirace již mnoha umělců i bionických inženýrů. Tito lidé většinou z přírody přejímají určité struktury, vzory, tvary a využívají je ke zdokonalení svých vynálezů nebo uměleckých děl. Kromě této míry inspirace, kdy okopírování tvaru křídel ptáků při návrhu tvaru křídla letadla vedlo ke snížení odporu vzduchu, existují i oblasti vědy a lidské činnosti, které využívají potenciálu přírody i v tom, že se neinspirují pouze jejími vlastnostmi, ale celými jejími procesy a životními strategiemi. Příkladem takové vědní oblasti jsou metaheuristické algoritmy na poli optimalizace.

Metaheuristické algoritmy přejímají z přírody koncept a myšlenky v ní probíhajících procesů a využívají je k optimalizaci úloh. V rámci těchto úloh hledají to nejlepší možné řešení z celé množiny všech možných řešení určitého problému. Motivací k inspiraci přírodou je skutečnost, že optimalizační procesy jsou v přírodě poměrně běžné a příroda měla k jejich vývoji (ověření) miliony let. Těchto znalostí se lidé rozhodli využít a aplikovat její postupy a strategie i v některých odvětvích matematiky a informatiky. Metaheuristické algoritmy proto vznikly propojením přírodních věd exaktních a věd poněkud více benevolentnějších, věd o (živé) přírodě. V podobném duchu se bude nést i tato bakalářská práce. Budeme se v ní zabývat především metaforami a jejich zdroji převzatými z živé přírody v kontextu příslušných algoritmů a jejich funkčností při aplikaci na problém lícování medicínských obrazů mozku.

Registrace (lícování) obrazů je běžným procesem obrazového zpracování, jehož úkolem je vzájemně na sebe namapovat dva a více obrazů tak, aby se odpovídající si objekty překrývaly co nejlépe. Hlavní příčinou toho, proč je téma registrace stále aktuální, je skutečnost, že registrační procesy jsou prvním krokem řady úkonů obrazového zpracování a od kvality jejich vyhotovení se odvíjí i kvalita následných obrazových zpracování, jako je segmentace obrazu, statistická analýza, detekce změn apod. Vhodné zpracování a vzájemné propojení více obrazů umožňuje v medicíně mimo jiné přinášet o zobrazovaných objektech nové informace, kombinovat obrazy pořízené různými zobrazovacími technikami, pozorovat vývoj nemoci, obrazy statisticky zpracovávat a lokalizovat v nich určité struktury.

Na poli optimalizace existuje kromě metaheuristických algoritmů řada dalších optimalizačních technik, avšak výkony těchto optimalizačních algoritmů nejsou vždy vyrovnané, protože s rostoucí složitostí a komplexností úloh, jako je registrace obrazů, se exaktní metody ve většině případů stávají neefektivními, nepoužitelnými anebo vůbec nejsou k jejich řešení dostupné. Právě v takových případech přinášejí uspokojivé výsledky heuristické nebo metaheuristické metody, které jsou schopny mezi množinou možných řešení nalézt to globálně nejlepší.

Komplexnost úlohy registrace obrazů mimo jiné spočívá v kombinaci požadavku medicíny na kvalitu se snahou automatizovat všechny metody obrazového zpracování. Dnes jsou již známy efektivní registrační metody pro případy, kdy máme dostatečně dobrý odhad pro započetí registrace. Tyto metody jsou dokonce velmi spolehlivé, ale pracují pouze lokálně a nejsou schopné se vyhnout suboptimálnímu řešení. Pokud řešíme úlohu automatické registrace a hrubou registraci nemáme k dispozici, musíme sáhnout po nějaké globální metodě, která je schopna nalézt kvalitní řešení i bez počátečních znalostí. Úloha tohoto globálního charakteru je značně obtížnější a časově náročnější.

Tato bakalářská práce má za cíl provést rešerši těchto přírodou inspirovaných algoritmů, na dva z nich se podrobněji zaměřit a porovnat je s jejich biologickou předlohou. První kapitola práce poskytuje teoretický úvod do problematiky optimalizace a metaheuristických algoritmů a předkládá rešerši těch nejznámějších a nejpoužívanějších z nich. Tento přehled metaheuristických algoritmů má za cíl nastínit hlavní principy fungování těchto algoritmů, vyzdvihnout jejich podobnosti stejně tak jako rozdíly s ohledem na zdroj inspirace, ze kterého čerpají. Druhá kapitola práce se věnuje teoretickému pozadí registrace medicínských obrazů a jeho souvislosti s optimalizačním procesem. Ve třetí kapitole jsou podrobně popsány dva vybrané metaheuristické algoritmy, jejichž popis je oproti algoritmům z první kapitoly rozšířen o srovnání s jejich biologickou předlohou. Předposlední kapitola práce se věnuje aplikaci těchto dvou algoritmů na registraci dvourozměrných obrazů mozku z magnetické rezonance. Podrobný popis implementace obou algoritmů, prováděné experimenty a dosažené výsledky jsou také součástí předposlední kapitoly. Závěrečná kapitola práce přináší diskuzi dosažených výsledků.

Kapitola 2

Optimalizace

2.1 Optimalizační úloha

S optimalizačními úlohami se setkáváme při řešení praktických úloh, během nichž se snažíme nalézt nejlepší možné řešení problému. O tom, které řešení je nejlepší, rozhoduje kriteriální funkce[1] $f : A \to \mathbb{R}$, která hodnotí kvalitu každého řešení z množiny možných řešení A. Cílem každé optimalizace je najít vektor $\mathbf{x} = (x_1, \ldots, x_d)$, který je optimem této kriteriální funkce f a tedy i nejlepším řešením problému.

Optimalizační úlohu můžeme formálně zapsat jako (Antoniou – Lu, 2007)

$$\min_{\mathbf{x} \in A}{}^{2}\mathbf{f}(\mathbf{x}) \quad \text{vzhledem k} \quad \begin{cases} h_i(\mathbf{x}) = 0 & i = 1,2,\ldots,K \\ g_j(\mathbf{x}) \leq 0 & j = 1,2,\ldots,L \end{cases}, \tag{2.1}$$

kde $\mathbf{f} = [f_1, f_2, \ldots, f_I]^T$ jsou kriteriální funkce a funkce $g_1, g_2, \ldots, g_L$ a $h_1, h_2, \ldots, h_K$ omezení optimalizace, které zužují prohledávací prostor A na prostor přípustných řešení problému $\Omega \subseteq A$. V případě, že $I = 1$, mluvíme o `jednokriteriální` optimalizaci a v případě, že $I \geq 2$, mluvíme o `vícekriteriální` optimalizaci. Pokud se na problém optimalizace podíváme z pohledu počtu pravděpodobných řešení - extrémů funkce f, mluvíme o `unimodální` nebo `multimodální` optimalizaci, tedy o optimalizaci s jedním nebo více extrémy.

Práce se dále zabývá pouze jednokriteriální multimodální optimalizací. Vícekriteriální úlohy se ale často řeší převedením na jednokriteriální úlohy. Dvěma nejčastějšími způsoby jsou: přiřazení váhy každé z kriteriálních funkcí, pak je výsledná kriteriální funkce jejich váhovaným součtem, nebo se určí jedna kriteriální funkce, která se optimalizuje, a ostatní kriteriální funkce se zahrnou do omezujících podmínek (Zelinka et al., 2008). Více o tomto převodu nebo o jiných způsobech vypořádání se s vícekriteriální optimalizací se lze dočíst např. v (Marler – Arora, 2004). Multimodální optimalizace je zase obecnější než unimodální optimalizace.

[1] V některé literatuře se také můžete setkat s označením účelová (objective function) nebo cenová (cost function, fitness) funkce. V některých případech i s pojmy utility function nebo energy function.

[2] Problém minimalizace lze snadno převést na problém maximalizace, a to převedením funkcí $f_i(\mathbf{x})$ na funkce $-f_i(\mathbf{x})$ a úpravou funkcí omezení $g_j(\mathbf{x})$ a $h_i(\mathbf{x})$.

2.2 Optimalizační metody

K hledání optimálního řešení problému se využívá hned několik přístupů. To, jaký přístup k řešení dané úlohy použít, se volí v závislosti na zkoumaném problému. Zjednodušeně se jedná o následující přístupy a metody:

- **enumerativní metody** generují všechna možná řešení daného problému a z nich vybírají to nejlepší řešení
- **analytické metody** použití těchto metod vyžaduje analytický popis kriteriální funkce, protože tyto metody využívají během výpočtu znalosti o derivaci/derivacích kriteriální funkce f
- **další metody** využívají k řešení úloh jiné matematické postupy bez toho, aby generovali všechny možná řešení nebo využívali derivaci kriteriální funkce f

Jak již bylo naznačeno v úvodu, tak ne všechny metody a přístupy jsou stejně efektivní a vhodné k řešení stejné úlohy. V oblasti optimalizace dokonce existuje teorém, který (zjednodušeně) tvrdí, že neexistuje univerzální algoritmus pro řešení libovolného optimalizačního problému, ale spíše platí to, že k řešení různých úloh se hodí různé algoritmy[2]. Použití výše zmíněných tříd optimalizačních metod vychází z povahy optimalizační úlohy a kriteriální funkce, mimo jiné v závislosti na velikosti a struktuře prohledávacího prostoru nebo existenci/neexistenci derivace kriteriální funkce. Enumerativní metody jsou vhodné k řešení problémů, ve kterých mají argumenty kriteriální funkce diskrétní charakter a mají malý prostor přípustných řešení. Důvodem je to, že generování všech možných řešení velkého prohledávacího prostoru by často nemuselo dospět k výsledku v rozumném čase. Analytické metody vyžadují znalost analytického přepisu kriteriální funkce a hodí se k řešení spíše monomodálních problémů, protože mnohdy uváznou v lokálním optimu. Třída dalších metod zahrnuje všechny ostatní optimalizační metody, které nemůžeme zařadit do předchozích dvou kategorií. Tato třída obsahuje i metaheuristické algoritmy, které jsou předmětem této práce.

Jiné dělení optimalizačních úloh může zohledňovat očekávané typy prohledávacího prostoru nebo parametrů kriteriální funkce. Toto dělení budeme využívat při pozdější rešerši, a proto je zde uvedeno. Pak dělíme optimalizační úlohy a optimalizaci na:

- **diskrétní optimalizace** – předpokládá diskrétní hodnoty parametrů kriteriální funkce
- **spojitá optimalizace** – kriteriální funkce i funkce popisující rovnosti a nerovnosti z (2.1) jsou spojitého typu
- **smíšená optimalizace** – parametry kriteriální funkce mohou být jak celá, tak reálná čísla

[2]Tento teorém se odborně nazývá No Free Lunch teorém. Více o něm se lze dočíst např. v Zelinka et al. (2008).

V praxi rozlišujeme mnoho typů optimalizačních úloh a metod klasifikovaných na základě různých kritérií. Dvě předložená dělení jsou jen jedna ze všech možných. Rozsáhlejší přehled dalších dělení lze nalézt například v Zelinka et al. (2008).

Jádrem všech optimalizačních metod jsou používané algoritmy. Pro lepší pochopení následujícího textu si algoritmy rozdělíme na základě dvou kritérii na `lokální` vs. `globální` a `deterministické` vs. `stochastické` algoritmy.

Podle schopnosti prohledávání prostoru přípustných řešení (Ω) můžeme algoritmy rozdělit na globální a lokální. Lokální metody typicky konvergují k lokálnímu extrému, které se nemusí vždy shodovat s globálním řešením. Tyto metody nemají žádný mechanismus-šanci z lokálního extrému uniknout (př. analytické metody) a vypořádat se s multimodálními úlohami. Pro svůj výpočet navíc potřebují nějaký počáteční odhad, ze kterého bude optimalizační proces zahájen. V výsledek optimalizace je na tomto odhadu závislý. Tento problém se řeší kombinací lokálních metod s metodami globálními nebo jejich opakovaným spuštěním. Pro hledání globálních extrémů multimodálních funkcí se spíše hodí globální metody, které mají schopnost z lokálních extrémů uniknout (Yang, 2011), avšak obvykle za cenu delšího výpočetního času.

Deterministické algoritmy jsou takové algoritmy, které podávají ze stejného počátečního odhadu stejný výsledek. Obecné úlohy globální optimalizace ale jimi nejsou řešitelné v polynomiálním čase (Volná, 2012). To je jeden z důvodů, proč vznikly algoritmy, které zefektivňují prohledávání prostoru řešení zavedením prvku náhody – stochastické algoritmy, nebo zavedením jiných postupů založených spíše na zkušenosti, intuici a odhadu (aproximaci) - tzv. `heuristické` algoritmy. Tyto algoritmy neprohledávají celý prostor řešení, ale pouze jeho podmnožinu. Z použití náhodnosti mimo jiné vyplývá, že každé spuštění těchto algoritmů může vracet jiný výsledek. A proto má smysl spustit tyto algoritmy vícekrát a z jejich výsledků vybrat ten nejvýhodnější.

2.3 Heuristické a metaheuristické algoritmy

2.3.1 Heuristické algoritmy

Slovo heuristika je odvozeno z řeckého slova *heuriskein*, které v překladu znamená hledat a objevovat (Vasant, 2012). Heuristiky při optimalizaci hledají a objevují nejvýhodnější kombinace parametrů funkce f (2.1) způsobem, který byl popsán v předchozí části. Tyto parametry budeme dále označovat jako hledané parametry.

Z použití výše popsané náhody vyplývá i několik dalších důsledků. Heuristické algoritmy nezaručují nalezení optimálního řešení, ale nalézají sub-optimální (dostatečně dobré) řešení v přijatelném čase. Formální důkazy správnosti, konvergence, heuristických algoritmů nejsou většinou k dispozici a to, že tyto algoritmy k řešení optimalizačních úloh fungují, je odvozeno (podpořeno) spíše empiricky (Yang, 2010a).

Další skupinu běžně používaných optimalizačních metod tvoří metaheuristické algoritmy. Metaheuristické algoritmy (metaheuristiky, MH) nemají žádnou přesnou definici, ačkoli v literatuře lze nalézt hned několik jejich „přibližných“ definic. Metaheuristiky jsou spíše chápány jako obecný rámec (způsob řešení problému) nasaditelný na jakýkoli optimalizační problém, jehož elementární operace si upravuje samotný uživatel přímo pro potřeby daného problému. To, že žádná přesná definice metaheuristik neexistuje, zdůrazňuje ve své knize také Yang (Yang, 2010a), ve které zároveň upozorňuje na trend poslední doby nazývat metaheuristikou jakýkoli stochastický algoritmus s lokálním prohledáváním. Metaheuristiky přitom ale z principu stochasticitu nutně nevyžadují. Hranice mezi metaheuristickými a heuristickými je často intuitivní - hlavní rozdíl mezi nimi je v tom, že heuristické algoritmy jsou oproti metaheuristickým algoritmům závislé na řešeném problému a velmi často pracují s apriori znalostmi o zkoumaném problému, metaheuristické algoritmy jsou zase více abstraktnější a jejich výkony jsou více jako „černá skříňka“[3]. Nebo ještě jinak, metaheuristické algoritmy poskytují nástroje nezávislé na problému, které mohou být použity ke konstrukci heuristického algoritmu.

2.3.2 Metaheuristické algoritmy

Téma metaheuristik je relativně mladou disciplínou, která se rozvíjela zvláště během posledních dvou dekád a jejíž rozvoj úzce souvisí s vývojem výpočetní techniky. Jako důkaz rostoucí popularity těchto algoritmů může sloužit následující článek (Whitacre, 2011). Přestože formální důkazy správnosti těchto algoritmů opět většinou chybějí, metaheuristiky podávají v řešení složitých optimalizačních problémů velice slibné výsledky a tvoří podstatnou část současných globálních prohledávacích technik. Princip fungování těchto algoritmů je většinou inspirován přírodními procesy nebo fyzikálními zákony.

Výhoda těchto algoritmu spočívá v jejich robustnosti, obecnosti a v tom, že jsou nehledě na počáteční podmínky schopny najít kvalitní řešení globálních úloh. Podle některých zdrojů (Yang, 2012b) se navíc jedná o metody implementačně a konceptuálně jednoduché. Na druhou, stranu správné použití MH i jejich úprava či vylepšení vyžaduje jistou dávku kreativity a invence (Hynek, 2008).

I přes předložený výčet pozitiv metaheuristických algoritmů nejsou tyto algoritmy pro řešení optimalizačních úloh vždy tou pravou volbou - to vyplývá z jejich vlastností. Metaheuristické (i heuristické) algoritmy se používají při řešení optimalizačních úloh až jako poslední možnost v případech, které již byly naznačeny v úvodu. A to, pokud je optimalizovaný problém příliš komplexní, je na něj kladeno příliš mnoho omezení anebo má velký prostor možných řešení, a tím pádem je časově náročný, a my na něj nemůžeme použít nějakou exaktní metodu, která je přímo určená k řešení specifických problémů. Druhou nevýhodou těchto algoritmů je nutné ladění tzv. `řídicích parametrů`, které má velký vliv na výkon algoritmu (více později v 2.5). Z tohoto důvodu se u těchto algoritmech zavádí automatické ladění parametrů nebo `meta verze` algoritmů, při níž se zvolený algoritmus použije k optimalizaci řídicích parametrů podřízeného algoritmu. Ukázkou automatického

[3] Systém, pro který jsou známy vstupy a výstupy, ale už ne jeho postup řešení.

ladění řídicího parametru může být praktická část této práce (viz 4.2.2). Nutno ovšem poznamenat, že i přes výše popsané nevýhody jsou MH v praxi velmi užitečné a soblasti jejich používání se stále rozšiřují.

Forma úpravy, modifikace, daného metaheuristického algoritmu na určitou třídu úloh se označuje jako hybridizace (Hynek, 2008). V hybridizaci jde o kombinace MH s jinými, nejčastěji lokálními metodami, popřípadě jen s některými jejich částmi.

Shrneme-li si celou kapitolu, tak před každou aplikací optimalizačního algoritmu je vhodná analýza řešené úlohy, protože volba optimalizační metody sehrává při řešení optimalizačních úloh svou roli. Bez této analýzy nelze vždy očekávat, že bude samotná aplikace efektivní či bude přinášet uspokojivé výsledky. Neznalosti uživatelů o vlastnostech kriteriální funkce je ovšem často nutí přistupovat k úlohám poněkud intuitivně.

Před klasifikací MH a samotným výčtem těch nejznámějších z nich ještě připomeňme, že tato práce se bude zabývat pouze těmi algoritmy, které v řešení problémů napodobují přírodu (nebo to o sobě alespoň tvrdí).

2.4 Optimalizační a biologická terminologie

Pro lepší pochopení následujícího textu je vhodné poznamenat, že všechny později popisované metaheuristiky patří mezi `populační algoritmy` (Swarm Intelligence based, SI). Populační algoritmy jsou založeny na spolupráci agentů[4]. Síla těchto algoritmů spočívá hlavně v tom, že na rozdíl od algoritmů, které během algoritmu optimalizují pouze jedno řešení, pracují populační algoritmy s množinou řešení, která snižuje pravděpodobnost algoritmu uvíznout v lokálním optimu.

Základem populačních algoritmů je `populace`, množina `jedinců` - kandidátních řešení dané optimalizované úlohy, jejichž kvalitu[5] odráží funkce `vhodnosti` (`fitness`[6]. Reprezentace jedinců v rámci algoritmů se různí podle typu používaného algoritmu a volí se s ohledem na povahu optimalizované úlohy. Jedinci bývají nejčastěji reprezentováni binárním řetězcem, řetězcem reálných čísel či stromovou strukturou (Floreano – Mattiussi, 2008). Zakódovaná varianta jedinců (např. v bitech - viz evoluční algoritmy[7] v 2.5.2 odpovídá jejich `genotypu`, dekódovaná varianta, která jednoznačně určuje hledané parametry optimalizované úlohy, odpovídá jejich `fenotypu`. Úseky `genomu` - reprezentace celého jedince, které kódují jednotlivé parametry, označujeme `geny`. A každou novou iteraci algoritmu označujeme jako `generaci`.

[4] Podle zdroje inspirace algoritmu se jednotlivým členům populace říká jedinec (inspirace evolucí) nebo agent (sociální chování zvířat), kde každý jedinec (agent) představuje jedno řešení optimalizačního problému (viz 2.4).

[5] Vzhledem k určitému kritériu nebo nebo určitým kritériím.

[6] Funkce vhodnosti (fitness funkce) se v některých případech dává do rovnosti s kriteriální funkcí. A proto někteří autoři nazývají kriteriální funkci fitness funkcí 2.1.

[7] Jedná se o konkrétní inspiraci evolucí (Evolutionary Algorithms, EA).

Nad takovými jedinci (u EA), můžeme (nebo spíš musíme být schopni) provádět určité `operace`, jejichž konkrétní podoba závisí na používaném typu algoritmu a formě reprezentace jedinců. Jejich principy zůstávají vždy zachovány. Proto se zde omezíme pouze na jejich myšlenku a v pasážích, kde to bude relevantní, jejich průběh zkonkretizujeme nebo se odkážeme na příslušnou literaturu.

- **selekce (selection)** v optimalizaci představuje formu přirozeného výběru vyskytujícího se v přírodě, která na základě kvality jedinců (jejich vhodnosti) vybírá omezený počet jedinců do nové populace, další iterace algoritmu. Selekcí někdy označujeme i výběr partnera do procesu křížení (viz další operace).
- **křížení, rekombinace (crossing over, crossover, recombination)** výměna (kombinace) genetické informace mezi dvěma nebo více jedinci, která dává vznik novým jedincům. Kombinací genetických informací zvyšujeme variabilitu jedinců v populaci. V závislosti na typu algoritmu a reprezentaci jedinců se používá jednobodové nebo vícebodové křížení, tj. náhrada celého genomu od určitého genu (jednobodové) anebo výměna příslušných úseků genomů jedinců vždy od určitého genu po jiný konkrétní gen (vícebodové).
- **mutace (mutation)** náhodné změny genetického materiálu nejčastěji na úrovní genů (jednotlivých parametrů řešení). Mutační operátor je v populace primárním zdrojem variability, který vytváří nové jedince (nová řešení problému). Podle povahy algoritmu se nastavuje četnost, se kterou k mutacím dochází. Obecně ale platí, že většina mutací má na jedince spíše destruktivní vliv než ten pozitivní, protože snižuje jejich vhodnost.
- **adaptace (adaptation)** označuje operaci, která v průběhu algoritmu upravuje hodnoty řídicích parametrů na základě průběhu algoritmu. Adaptace řídicích parametrů vylepšují konvergenční vlastnosti algoritmu (schopnost dosáhnout optima).

Pro ještě lepší přehlednost a orientaci v textu přikládáme tabulku 2.1, která sjednocuje ekvivalentní pojmy v rámci optimalizace a biologie zvláště EA algoritmů.

2.5 Přehled nejznámějších metaheuristik

Před samotnou rešerší těch nejznámějších MH si ještě definujme několik technických pojmů, které budeme v přehledu a práci používat. V kontextu optimalizačních úloh a optimalizačních algoritmů se mluví o dvou typech parametrů, již zmíněných `řídicích parametrech` (control parameters) a `optimalizovaných parametrech` (solution parameters)[8]. Řídicí parametry představují proměnné, na jejichž hodnotách závisí schopnost algoritmu optimalizovat a které je nutno nastavit ještě před samotným spuštěním algoritmu.

[8]Optimalizované parametry bývají také v některé literatuře označovány jako návrhové parametry (design variables) nebo (optimization parameters) a (decision variables). Pro řídicí parametry se používá i označení (strategy parameters).

Tabulka 2.1: Slovník pojmů – biologie vs. optimalizace

Pojem	**Oblast**	
	Optimalizace	Biologie
`gen` (`gen`)	úsek reprezentace jednoho řešení kódující jeden hledaný parametr	základní jednotka genetiky, dodnes problém s vymezením[a]
`alela` (`allele`)	jeden konkrétní a zakódovaný hledaný parametr	konkrétní forma genu
`chromozom` (`chromosome`)	soubor genů kódující všechny hledané parametry	struktura v jádře eukaryotických buněk, která je nositelem genetické informace jedinců
`genom` (`genome`)	soubor všech chromozomů jedince, většinou odpovídá jednomu chromozomu [b]	soubor všech genů jedince
`populace` (`population`)	množina všech aktuálních kandidátních řešení optimalizačního problému	skupina jedinců stejného druhu obývající stejné území, která je schopná se mezi sebou množit
`vhodnost` (`fitness`)	kvalita řešení	udává relativní genový příspěvek jedince do další generace
`genofond` (`gene pool`)	soubor všech zakódovaných parametrů v rámci všech řešení populace	soubor všech genů v populaci
`lokus` (`locus`)	umístění jednoho zakódovaného parametru řešení v rámci chromozomu	konkrétní umístění jednoho genu na chromozomu
`selekce` (`selection`)	operátor výběru nové populace řešení do nové iterace[c]	přirozený výběr
`mutace` (`mutation`)	operátor náhodné změny řešení	náhodná změna genetické informace
`rekombinace` (`recombination`)	operátor křížení dvou či více řešení	výměna částí chromozomů během 1. meiotického dělení buněk
`adaptace` (`adaptation`)	operátor, který za běhu algoritmu upravuje hodnoty řídicích parametrů	označuje znak, jež je výsledkem adaptabilního procesu

[a] Velmi zjednodušeně bychom mohli říci, že každý gen kóduje jeden protein nebo jeden znak.

[b] Výjimku tvoří algoritmy, kde nejsou hledané parametry zakódovány v 1, ale více chromozomech (viz MGA, nebo PGA v 2.5.2).

[c] Může označovat i proces výběru jednotlivých řešení k rekombinaci.

Optimalizované parametry jsme si dříve nazvali hledanými parametry a představují hledaný vektor **x** z kapitoly 2.1.

V následujícím přehledu se řídíme klasifikačním kritériem z kapitoly 2.2.

2.5.1 Algoritmy primárně pro řešení diskrétních problémů

· **Optimalizace mravenčí kolonií (Ant Colony Optimization, ACO)**
Algoritmus ACO řadíme do rodiny algoritmů rojové inteligence (SI) (Deneubourg et al., 1990). V oblasti optimalizace je ACO typický zástupce algoritmů pro řešení složitých kombinatorických úloh jako je NP-těžký[9]. problém obchodního cestujícího (Travelling Salesman Problem, TSP (Lawler et al., 1985; Dorigo et al., 1991).

Inspirace tohoto algoritmu pochází ze sociálního chování mravenců při hledání potravy, které se ACO snaží imitovat. Způsob chování mravenců popisuje ve své práci např. Deneubourg et al. (1990). V přírodě si mravenci značí cesty vedoucí k potravě vylučováním feromonů. Tyto feromony jsou pro ostatní mravence detekovatelné a slouží jim jako vodítka ke zdrojům potravy. Tento feromonový způsob komunikace umožňuje mravencům nalézat v prostoru okolo mraveniště nejkratší cestu k potravě, která se po nějakém čase projeví vyšší koncentrací feromonů a tím, že se po této nejkratší cestě k potravě postupně pohybuje většina mravenců. Podobným způsobem funguje i algoritmus ACO. Umělí mravenci se v ACO pohybují po neorientovaném ohodnoceném grafu, který symbolizuje daný optimalizační problém. Možné cesty mravenců v grafu jsou možnými řešeními optimalizované úlohy. Umělí mravenci v každé iteraci paralelně procházejí graf a hledají různá řešení úlohy. Váha každé hrany grafu odpovídá množství feromonu, který na ni byl nanesen, a preferenci, se kterou se po ní mravenci vydají. Do algoritmu je pro zvýšení robustnosti zakomponováno i postupné vypařování feromonů (Zelinka et al., 2008), aby nepoužívané hrany mravence příliš „nelákaly“. Váhy hran se po každé iteraci aktualizují podle počtu průchodů mravenců danou hranou v závislosti na kvalitě řešení (zdroje), ke kterému daná cesta mravence vedla.

Základní verze algoritmu ACO má 3 řídicí parametry[10]. V současnosti existuje mnoho variant mravenčích algoritmů, které se liší hlavně v množství a způsobu ukládání feromonů, podrobný přehled lze nalézt v Cordon et al. (2002). Dnes se ACO využívá především k řešením dvou typů problémů: tím prvním jsou již zmíněné NP-těžké kombinatorické problémy (Garey – Johnson, 1990), tím druhým jsou dynamické problémy nalezení nejkratší cesty. Algoritmus byl také upraven pro řešení spojitých optimalizačních problémů (Liao et al., 2014).

[9] Neexistuje deterministický algoritmus, který by byl schopen řešit tyto úlohy v polynomiálním čase. Dokonce není ani známo, zda lze takový algoritmus vůbec najít

[10] Jedná se o vliv feromonů, vliv délky hrany grafu a sílu vypařování feromonů.

2.5.2 Algoritmy primárně pro řešení spojitých a smíšených problémů

· **Algoritmus umělých včelích kolonií**[11] **(Artificial Bee Colony, ABC)**
Autor algoritmu umělých včelích kolonií se inspiroval sociálním chováním včel a jejich způsobem získávání potravy. Princip fungování algoritmu bude podrobněji popsán v kapitole 4.1.

Značnou výhodou tohoto optimalizačního algoritmu je nízký počet (4) řídicích parametrů[12]. Jako nevýhoda ABC se někdy uvádí pomalá či naopak předčasná konvergence k extrému u komplexnějších problémů (Verma – Kumar, 2013). V literatuře existuje několik variant ABC, jež se snaží tento nedostatek řešit např. (Ampellio – Vassio, 2014).Algoritmus ABC se používá k řešení multimodálních optimalizačních úloh původně bez omezení, s malou úpravou i pro optimalizační problémy s omezením (Karaboga – Basturk, 2007; Karaboga – Akay, 2011).

Chováním včel (a to nejen způsobem, jakým hledají potravu) se inspirovalo i mnoho dalších autorů za vzniku celé řady metaheuristických algoritmů. Podrobný přehled i s jejich aplikacemi lze najít v Karaboga – Akay (2009).

· **Optimalizace hejnem (Particle Swarm Optimization, PSO)**
Optimalizace hejnem nebo také Rojení částic je optimalizační technika, která simuluje choreografii pohybu skupiny živočichů jako je hejno ptáků nebo ryb. Princip PSO spočívá v synchronním prohledávání prostoru přípustných řešení skupinou spolupracujících jedinců (částic). Poloha agenta v prostoru určuje hodnoty hledaných parametrů a rychlost pohybu jedinců souvisí s jejich vhodností a vhodností okolních agentů.

Během jednotlivých iterací PSO si agenti nezávisle na sobě upravují směr a rychlost pohybu, tzv. „rychlostní vektor“ (velocity vector). Rychlostní vektor si agenti upravují podle agenta, který má v populaci dosud nejlepší nalezenou vhodnost, své nejlepší dosažené pozice v prostoru a agenta, který má nejlepší vhodnost v rámci jejich okolí. Okolí jedinců je ve většině případů definováno jako neorientovaný graf (Dorigo et al., 2008) vyjadřující, kteří agenti spolu mohou komunikovat a kteří ne. Formě užívané komunikace (např. v rámci kruhu) se přezdívá topologie sousedství. Sousedství agentů v konečném důsledku znamená to, že agenti nejsou oddělenými jednotkami, ale navzájem propojenou sítí, kde všichni nepřímo ovlivňují všechny, stejně jako je tomu v přírodě. Celá populace agentů se pohybuje prostorem přípustných řešení a je přitahována do slibných míst. Sousedství agentů není sice součástí základní verze PSO, ale zmírňuje tendenci algoritmu předčasně konvergovat (Zelinka et al., 2008), a je dnes považováno za standard. Kvůli nedeterminovanosti pohybu agentů je do úpravy jejich polohy zakomponován i prvek náhody.

Slabinou PSO je relativně vyšší počet řídicích parametrů (7)[13]. PSO se obecně používá

[11] Volný překlad.

[12] Je nízký vzhledem k jiným optimalizačním technikám. Dále v přehledu to již nebudeme zdůrazňovat. Řídicími parametry ABC jsou: počet včel, limit pro lokální prohledávání, počet skautek a maximální počet generací.

[13] Jedná se o počet agentů, vliv setrvačnosti, vliv nejlepšího nalezeného řešení, vliv nejlepšího řešení souseda, maximální rychlost agentů, počet jedinců v sousedství a maximální počet iterací.

pro řešení jak spojitých, tak diskrétních úloh (Dorigo et al., 2008). Na poli optimalizace existuje hodně jejich modifikací, zvláště pro vícekriteriální optimalizaci. Nejznámějšími verzemi jsou „Niching PSO“ a „Speciation PSO“. Ve verzi „Speciation PSO“ se vytváří více skupin hejn (druhů), ve kterých vždy existuje nějaký vedoucí jedinec, podle kterého se všichni agenti řídí. Příslušnost agentů k hejnu i vedoucí jedinci se v každé iteraci mění. V „Niching-PSO“ verzi se pomalu během iterací na základě sousedství oddělují skupinky, a ty prohledávají slibná místa prostoru přípustných řešení individuálně. Jiné verze PSO využívají více topologií najednou nebo v nich jedinci přeskakují v rámci sousedství. Podrobný rozbor algoritmu PSO a jeho verzí lze najít v (Poli et al., 2007).

· **Samoorganizující se migrační algoritmus (Self Organizing Migrating Algorithm, SOMA)**
Filosofie fungování tohoto algoritmu spočívá ve spolupráci jedinců se společných cílem. Společným cílem jedinců u SOMA algoritmu je migrace do nejpříznivějších oblastí prostoru možných řešení, kdy si jedinci mezi sebou sdělují kvality prozkoumaných oblastí. Kvality prozkoumávaných oblastí jsou ohodnocovány hodnotami funkce vhodnosti jednotlivých jedinců. Rozdíl oproti již popsaných SI algoritmům (PSO) leží v tom, že během iterací (migračních kol) nedochází ke generování úplně nových jedinců, ale k migraci jedinců směrem k nejlepšímu nebo nejlepším jedincům za vzniku celé řady potomků (Zelinka et al., 2008). Na začátku každého migračního kola (iterace) se podle nejvyšší vhodnosti zvolí vedoucí jedinec a operací perturbace[14] se vytvoří tzv. perturbační vektor. K jeho vytvoření náhodně generujeme čísla, která porovnáváme s jedním řídicím parametrem algoritmu za vzniku binárního řetězce o délce shodné s počtem hledaných parametrů. Pokud je vygenerované číslo menší než onen řídicí parametr, uložíme 1, naopak uložíme 0. Poté jedinci „skáčou“ po hyperploše (prostoru přípustných řešení) směrem k vedoucímu jedinci a vytváří řadu svých potomků. U SOMA má každý potomek jediného rodiče. Ze všech potomků jediného rodiče vždy přežije pouze ten nejkvalitnější jedinec. Perturbační vektor určuje, jaké hledané parametry se budou během migrace jedinců (jejich skoků) měnit (1) a které zůstanou zafixovány (0).

Autor algoritmu udává jako předlohu algoritmu střídající se fáze kooperace jedinců a jejich soutěžení v prostoru (Zelinka et al., 2008).

Malou nevýhodou tohoto algoritmu může být relativně vyšší počet (5) řídicích parametrů [15]. Jiné verze algoritmu SOMA se liší v počtu vedoucích jedinců, popř. ve způsobu, jak jsou tito jedinci vybíráni (náhodně nebo podle jejich vhodnosti). Kadlec – Raida (2011) publikoval verzi SOMY pro vícekriteriální úlohy. Schwarz et al. (2006) navrhl rozšíření algoritmu SOMA pro dynamické problémy se zavedením omezené životnosti jedinců.

· **Algoritmus svatojánských mušek[16] (Firefly Algorithm, FA)**

[14] Operace mutace je podle zdroje inspirace přejmenována na perturbaci a ctí myšlenku generování nových řešení.

[15] Jedná se o mutační práh, délka skoku, velikost populace, délka migrační cesty a počet iterací

[16] Volný překlad.

Jako předloha algoritmu svatojánských mušek sloužila bioluminiscence světlušek. Světlušky na sebe v přírodě v období páření upozorňují nápadným světélkováním. Cílem světélkování světlušek je přilákat k sobě jedince opačného pohlaví. Podobně fungují i agenti v algoritmu FA. Agenti se pohybují v prostoru přípustných řešení a svým světélkováním se k sobě navzájem přitahují. Je vhodné dodat, že agenti jsou bezpohlavní a všichni přitahují všechny. Míra přitažlivosti agentů závisí na jejich vhodnosti pro řešení optimalizovaného problému a na vzdálenosti[17], v jaké se od potencionálního „partnera" nachází. V algoritmu taktéž figuruje absorpce světla, která snižuje přitažlivost agentů(Yang, 2010b). Během každé iterace si agenti upravují svou polohu vzhledem ke všem agentům v populaci a pohybují se směrem k nejkvalitnějšímu/nejkvalitnějším agentovi. Jejich pohyb je dále ovlivněn náhodným vektorem, pro jehož generování se používají různá pravděpodobnostní rozdělení např. (Yang, 2010c). Pokud se v okolí agenta žádný kvalitnější jedinec nenachází, agent se v prostoru „posune" jen náhodně.

Algoritmus FA má 3 řídicí parametry[18] a původně byl navrhnut pro řešení spojitých jednokriteriálních úloh. Pro řešení vícekriteriálních úloh ho později upravil samotný autor (Yang, 2012a).

Nejpočetnější a odborníky velmi oblíbenou třídu metaheuristických algoritmů tvoří `evoluční algoritmy (EA)`. Tyto algoritmy čerpají svoji inspiraci z paradigmatu Darwinovy teorie evoluce a Mendelových zákonů[19]. Mezi tyto techniky řadíme Genetický algoritmus, Evoluční programování, Evoluční strategie a Genetické programování, Gramatickou evoluci popř. další techniky - to se různí autor od autora. Bavíme-li se o této třídě algoritmů, užíváme terminologii a operátory z kapitoly 2.4.

· **Genetický algoritmus (Genetic Algorithm, GA)**
GA je asi nejpoužívanějším metaheuristickým algoritmem vůbec. Hlavní myšlenka GA[20] spočívá v simulaci přirozeného výběru, kdy vystavujeme jedince selekčnímu tlaku, optimalizačnímu kritériu, který zapříčiňuje vznik lépe přizpůsobených jedinců vůči svému okolí, působícímu tlaku. Je to způsobeno především tím, že jedinci lépe přizpůsobeni svému okolí žijí déle, mají vetší pravděpodobnost se rozmnožit a rozšířit svůj genetický materiál do další generace, což má za následek to, že v příští populaci budou převažovat lépe přizpůsobení jedinci. Podrobnější biologický podklad a mechanismus evoluce je uveden v kapitole 4.2.1. GA simuluje popsanou evolucí opakovanou aplikací operátorů křížení, mutace a selekce 2.4.

V základní verzi GA se používá binární zakódovaní jedinců[21], ale dnes již existují i verze s reálným zakódováním (Herrera et al., 1998). Použití binárního zakódování poskytuje odhad velikosti prostoru přípustných řešení, protože pro danou délku chromozomu umíme spočítat počet možných zakódování jedinců. Nevýhoda binární zakódování jedinců spočívá zase v tom, že před každým oceněním jedince vyžaduje výpočet funkce

[17] Většinou se používá euklidovská metrika, ale volba metriky je na uživateli (Yang, 2009).
[18] Jedná se o absorpční koeficient, sílu přitažlivosti a sílu náhodné složky.
[19] Více o těchto paradigmatech a zdroji inspirace EA se lze dočíst v kapitole 4.2.
[20] To platí i pro další EA algoritmy.
[21] Všechny geny (hledané parametry) jsou v genotypu zakódovány řetězcem 0 a 1 o stejné délce.

vhodnosti zpětný převod binárního řetězce na reálné hodnoty, tedy převod z genotypu na fenotyp. V některé literatuře se i uvádí, že GA má malou schopnost lokálního prohledávání (Verma – Kumar, 2013).

Teoretický důkaz správnosti fungování algoritmu (binární reprezentace) se opírá o teorii schémat (building block hypothesis), která se snaží podpořit funkčnost a smysl používání operátorů křížení a mutace. Hlavní myšlenka této teorie spočívá v tom, že kombinace kvalitních úseků chromozomů rodičů dává vznik ještě kvalitnějším potomkům, ve kterých se kvalitní úseky genomu jedinců zachovávají (Floreano – Mattiussi, 2008; Kvasnička et al., 2000).

Upravenou verzí GA je Složitý GA[22](Messy Genetic Algorithm, MGA), kde už není jedinec vyjádřen jedním binárním řetězcem (chromozomem), ale dvěma řetězci s proměnlivou délkou a s tím rozdílem, že geny (hledané parametry) nejsou v řetězci pozičně závislé. MGA se hodí k řešení klamných problémů (deceptive problems)[22], multimodální optimalizaci a používá se i k řešení dynamických problémů. Od popsané reprezentace jedinců se odvíjí i nutnost přizpůsobení operátorů. Paralelní verze GA (PGA) spočívá ve vytvoření subpopulací, ve kterých dochází k synchronnímu provádění GA. V některých verzích dochází ke komunikaci jedinců mezi sebou a výměně nejlepších jedinců mezi subpopulacemi. Existují také verze centralizované, s jedním nadřazeným GA algoritmem, nebo naopak verze s velmi malými subpopulacemi, tvořenými i jediným jedincem.

- **Evoluční strategie (Evolution Strategy, ES)**
  Algoritmus Evolučních strategií bude podrobněji popsán v kapitole 4.2. Nastiňme si proto tady jen jeho výhody a rozdíly vůči předchozímu algoritmu GA. Algoritmus ES se historicky vyvíjel současně s GA (60. léta 20. století). Prvotním účelem navržení ES bylo řešit úlohy inženýrských návrhů. ES se až později prokázaly jako dobrá (účinná) technika pro numerickou optimalizaci (Beyer – Schwefel, 2002).

  V ES se oproti GA používá reálné kódování návrhových parametrů-genů, které nevyžaduje převod z fenotypu do genotypu a tomu také bývají přizpůsobeny operátory. Dalšími rozdíly mezi GA a ES jsou: četnost mutací a selekční mechanismus, i když to není vždy pravidlem. ES ještě podporují křížení více jedinců, avšak GA to principiálně nezamítají. Rozdíl mezi těmito algoritmy je i v tom, že v GA vznikají křížením dvou jedinců dva potomci, u ES pouze jeden. Původní verze ES navíc křížení vůbec nepoužívaly.

  Hlavní výhoda ES je schopnost upravovat své řídící parametry, především parametr udávající sílu mutace. Základní verze ES má 3 řídicí parametry[23], ale běžně používané verze jich mají mnohem více (viz 4.2). Podobně jako pro většinu optimalizačních algoritmů, byla i pro ES vytvořena verze pro diskrétní úlohy.

[22] Volný překlad.

[22] Pro tento typ problémů platí, že globální řešení problému je obklopeno samými podprůměrnými řešeními. Klasický GA při řešení těchto úloh selhává, protože hledá optimální řešení pomocí genotypu nadprůměrných řešení, která jsou v tomto případě optimálnímu řešení vzdálena (Hynek, 2008).

[23] Řídicími parametry jsou síla mutace a parametr k adaptaci síly mutace a maximální počet generací.

- **Evoluční programování (Evolution Programming, EP)**
  Evoluční programování lze chápat jako rozšířenou verzi ES. Podobnost základních verzí obou algoritmů spočívá v používání pouze operace mutace a v tom, že do další generace vždy vybíráme jen nejlepší jedince rodičů a potomků (dnes se používá jiný typ selekce). Během mutace vznikne z každého rodiče jeden potomek, ačkoli není nutné, aby byl počet jedinců v populaci konstantní. Dalším společným rysem EP a ES je to, že oba tyto algoritmy podřizují reprezentaci jedince řešenému problému (tj. na všechny hledané parametry nemusíme aplikovat stejné operace, o ES více později) (Kvasnička et al., 2000). Rozdíl těchto dvou metod naopak spočívá ve způsobu realizace mutace a pozdějším vývoji.

  Biologickou metaforou algoritmu EP je vývoj vzájemně konkurujících si druhů. EP byl původně navržen k předpovědi nových stavů a událostí u konečných automatů (Kvasnička et al., 2000).

- **Diferenciální evoluce (Differential Evolution, DE)**
  Diferenciální evoluce opět vychází z operací mutace, křížení a selekce. Rozdíl mechanismu DE oproti GA je v pořadí aplikování operací (mutace-křížení-selekce) a v tom, že zdrojem kvalitnějších řešení není u DE operace křížení, ale mutace (Verma – Kumar, 2013).

  K vytvoření nového jedince je u DE zapotřebí 4 dalších jedinců. Ke každému jedinci se vygenerují 3 náhodně zvolení jedinci a z nich se (mutací) vytvoří tzv. `šumový` vektor (noisy vector), kdy spočítám diferenci dvou z těchto jedinců, tu navážím a přičtu k třetímu náhodně zvolenému jedinci. Křížení probíhá nad vybraným jedincem a šumovým vektorem za vytvoření tzv. `pokusného` vektoru (trial vector) (Storn – Price, 1997; Zelinka et al., 2008). Pokusný vektor se vytváří vždy náhodným výběrem mezi odpovídajícími si parametry šumového vektoru a parametry náhodného (čtvrtého) jedince. Vhodnost vzniklého pokusného vektoru se poté porovná s původně vybraným jedince a z těchto dvou jedinců přežije pouze ten kvalitnější - operace selekce. V základní verzi DE je algoritmus ukončen pouze daným počtem generací.

  Výhodou tohoto algoritmu je relativně jednoduchá konstrukce a nízký (4) počet parametrů [24]. Nevýhodou DE je častá stagnace tzn. jev, kdy nedochází ke zlepšování vhodností jedinců, přestože je populace stále diverzibilní (Lampinen – Zelinka, 2000). K tomuto algoritmu neexistuje přímá přírodní metafora.

  Algoritmus má více verzí, které se liší především ve způsobu zvolení jedinců k mutaci, vytvoření pokusného vektoru a mechanismu operátorů křížení. Přehled verzí DE i jejich porovnání lze najít v Mezura-Montes et al. (2006). Původní algoritmus byl navržen pro spojité problémy, dnes již ale existují i verze pro smíšené problémy, popř. adaptabilní nastavení parametrů DE (Lampinen – Zelinka, 1999).

[24] Jedná se o velikost populace, práh křížení, mutační konstantu a počet generací.

2.5.3 Algoritmy pro řešení dalších typů optimalizačních problémů

· **Genetické programování (Genetic Programming, GP)**
Genetické programování tvoří v rámci evolučních algoritmů samostatnou skupinu, protože se používá k řešení úplně jiných typů problémů. Oproti předchozím technikám GP nehledá parametry kriteriální funkce, ale funkci samotnou (nějakou strukturu, program v konkrétním programovacím jazyce). Kostra a princip fungován algoritmu jsou téměř shodné s již zmíněným GA, ve skutečnosti se jedná o jeho rozšíření. GP a GA se liší především v reprezentaci jedinců, v GP jsou jedinci vyjádřeni stromem s proměnlivou délkou chromozómu, a tedy i jinými operacemi - myšlenky GA jsou opět zachovány. Výsledkem GP je program, funkce, nějaká spustitelná struktura.

GP se běžně používá k řešení úloh symbolické regrese[25]. Výhoda této metody spočívá v tom, že vyžaduje velmi malé vstupní znalosti o řešeném problému (Kvasnička et al., 2000). Naopak nevýhodou tohoto algoritmu je časová náročnost, protože pro každé ocenění řešení (vypočítání jeho vhodnosti), se musí jednotlivá řešení vyzkoušet, spustit, na trénovací množině, což bývá časově dosti náročné. Další možnou nevýhodou tohoto algoritmu je určitá chybovost. Chybovostí není myšleno to, že by algoritmus nefungoval (metaheuristické algoritmy to taky nezaručují), ale to, že výsledek algoritmu nezaručuje nalezení řešení, které bude 100 % funkční (výsledek algoritmu - program - nemusí vždy vracet správná řešení). Z tohoto důvodu se ne všechny optimalizační problémy, řešitelné GP, dají pomocí GP počítat, protože v některých případech si chyby nemůžeme dovolit. Různé verze GP se liší v kódování jedince a silně rozlišují mezi genotypem a fenotypem. Vhodnost jedince se počítá z rozdílu mezi chováním programu jedince a požadovaným chováním výsledného programu (Zelinka et al., 2008).

· **Gramatická evoluce (Grammatical Evolution, GE)**
Gramatická evoluce se dá chápat je typ GP, ve které můžeme tvořit programy v libovolném programovacím jazyce. GE se od GP liší i v reprezentaci jedinců, používá tzv. `lineární genomy`, například řetězce celých čísel, které vyžadují převod z genotypu na fenotyp a vychází z gramatiky v O'Neill – Ryan (2003). Princip fungování algoritmu opět vychází z myšlenek GA.

· **Umělé imunitní systémy (Artificial Immune Systems, AIS)**
Třída umělých imunitních systémů je rozsáhlejší oblastí výpočtové inteligence (soft-computing) která našla uplatnění i v optimalizačních úlohách. AIS představují algoritmy, které při řešení optimalizačních úloh využívají zjednodušenou abstrakci imunitního systému živočichů. Přejímají od něj především model učení a vytváření paměti bílých krvinek pro ochranu organismu před cizorodými látkami. AIS pracují se dvěma populačními algoritmy - algoritmy pozitivní a klonální selekce - a imunitními sítěmi[26].

[25] Smyslem symbolické regrese je proložit data, trénovací množinu, vhodnou funkcí, reprezentovanou syntaktickým stromem, pro jejíž/jehož sestrojení máme k dispozici různé matematické operace, nezávislé proměnné a konstanty (Kvasnička et al., 2000).

[26] Imunitní sítě představují jistou podobu neuronových sítí, která se primárně využívá ke klasifikaci dat, extrakci znalostí z dat nebo detekci anomálií. de Castro – Timmis (2002) publikoval rozšíření i na optimali-

Algoritmus pozitivní selekce se typicky používá k detekci chyb a počítačové bezpečnosti z důvodu detekce počítačových virů. Pozitivní selekce probíhá ve dvou fázích. Ve trénovací fázi vytváříme nová řešení problému, ze kterých vybereme jen ty, které svou afinitou vůči množině vzorů (antigenů) překročí zadaný práh[27]. V monitorovací fázi podrobujeme vybraná řešení testování na testovací množině dat, ve které ověřujeme, zda neobsahují chybu (Husakova, 2010).

Algoritmus klonální selekce simuluje tvorbu protilátek proti tělu cizorodým látkám a podobně jako GA vytváří různorodá řešení problému. Tento algoritmus jde o krok dál než pozitivní selekce. Algoritmus klonální selekce vytváří podobně jako algoritmus pozitivní selekce kandidátní řešení problému, které porovnává se vzory (antigeny). Řešení, která překročí svou afinitou stanovený práh se ale začnou klonovat (čím kvalitnější řešení, tím více klonů vytvoří) a podléhají mutaci (čím větší afinita, tím menší míra mutace). Naklonovaná a zmutovaná řešení, které nepřekročily svou afinitou určitý práh, jsou nahrazeny náhodně vygenerovanými řešeními. Na zmutovaná a nově vygenerovaná řešení se aplikuje klasický selekční mechanismus a vybrané protilátky postupují do další generace. Zde popsaný postup odpovídá algoritmu CLONALG (CLONal selection ALGorithm) (Husakova, 2010). Cílem tohoto algoritmu je vytvořit co nejpodobnější řešení ke každému vzoru (antigenu). CLONALG má jen 3 řídicí parametry. Jiný algoritmus, CLONALGOPT, určuje podobnosti mezi vzorem a řešeními na základě funkce vhodnosti a sje už více podobný naším optimalizačním úlohám (Husakova, 2010). Při optimalizaci se AIS využívají pro řešení multimodálních a vícekriteriálních úloh.

V oblasti optimalizace existuje mnoho dalších a používaných MH. Z důvodu omezeného rozsahu práce už zde uvedeme jen jejich výčet s odkazem na příslušnou literaturu. Dalšími hojně používanými MH jsou: Simulované žíhání (Simulated Annealing, (Kvasnička et al., 2000)) , Rozptýlené hledání (Scatter Search, (Zelinka et al., 2008; Martí et al., 2006)), Optimalizace netopýry[28] (Bat Algorithm, (Yang, 2010d)), Prohledávání pomocí harmonií (Harmony Search, (Geem et al., 2001; Yang, 2010a)), Memetické algoritmy (Memetic Algorithm, (Onwubolu – Babu, 2004)), Kukaččí algoritmus[28](Cuckoo Search, (Yang – Deb, 2009; Yang, 2010a)), Optimalizace bakteriemi[28](Bacterial Foraging Optimization, (Passino, 2002)).

Ještě více vyčerpávající seznam, a to jen přírodou inspirovaných metaheuristických algoritmů, lze najít v Fister Jr et al. (2013), popř. v Binitha – Sathya (2012).

zační úlohy.

[27] Afinita odpovídá podobnosti a je počítána na základě různých metrik (Hammingova vzdálenost, Euklidovská vzdálenost ap.). Cílem tohoto porovnání je vytvořit řešení, které je schopné detekovat určitý problém.

[28] Volný překlad.

Kapitola 3

Registrace (lícování) medicínských obrazů

Registrace obrazů (Image Registration, IR) je jednou z dílčích metod předzpracování (analýzy) obrazu. Její výsledky bývají přínosné v mnoha vědních oblastech, což je také důvodem, proč se metodám registrace obrazů a jejich vylepšení věnuje tolik pozornosti (Zitova – Flusser, 2003). Cílem registračního procesu je namapovat na sebe dva obrazy tak, aby se odpovídající si objekty v obrazech překrývaly co nejlépe. Většinou se jedná o obrazy stejné scény získané v jiných časech (pohyb, časová prodleva), snímky zaznamenávající stejnou scénu z různých úhlů nebo obrazy nasnímané různými přístroji[1] (Zitova – Flusser, 2003). V oblasti medicíny umožňuje zaregistrování (dvou či více) obrazů sledovat průběh léčby, porovnávat a identifikovat různé objekty, nalézat v obrazech konkrétní struktury nebo získávat o snímaných objektech komplexnější „obraz“.

Tato práce se bude konkrétně zabývat registrací dvourozměrných (2D) obrazů mozku získaných z magnetické rezonance (Magnetic Resonance Imaging, MRI). Tomuto účelu budou přizpůsobeny i následující kapitoly. Veškeré popisované postupy se ale dají obecně uplatnit i na jiné typy registrací v jiných vědních oborech.

V oblasti zpracování obrazu rozumíme pod pojmem digitální obraz o rozměrech $|M| \times |N|$ jasovou funkci $I : M \times N \to \mathbb{R}$, která každému bodu (x,y) v prostoru (ve 2D případě `pixelu`, ve 3D případě `voxelu`) přiřazuje jeho intenzitu (Burger – Burge, 2009). Registraci potom formálněji můžeme popsat jako proces hledání takové prostorové transformace $T : \mathbb{R}^n \to \mathbb{R}^n$ `plovoucího` obrazu I_S[2], která co nejlépe mapuje odpovídající si body `referenčního` obrazu I_M [3].

[1] Jsou-li registrující se obrazy pořízeny více přístroji-modalitami, registraci nazýváme multimodální. Pro úplnost ještě připomeňme, že v oblasti optimalizace zase modalita označuje počet extrémů kriteriální funkce 2.1.

[2] I_S - S je zkratka pro anglické slovo *scene* a odpovídá plovoucímu (moving) obrazu

[3] I_M - M je zkratka pro *model* a odpovídá fixnímu (fixed) nebo referenčnímu obrazu

3.1 Geometrická transformace obrazu

Klíčovou roli při registraci obrazů hraje volba typu `transformační funkce` T. Ta určuje, jaké druhy operací můžeme pro nalezení příslušné transformace používat, klasicky[4] se jedná o operace posunutí (translation), rotaci (rotation), změnu měřítka (scaling) a zkosení (shearing). Typ transformační funkce dále udává počet parametrů (degrees of freedom, DOF), které danou transformaci popisují, a vlastnosti obrazu, jež se po aplikaci transformace v obraze zachovávají (např. velikost, tvar, orientace, rovnoběžnost přímek atp.). Počet DOF je pro kvalitu registrace a náročnost výpočtu rozhodujícím faktorem (více v 5.4).

Proces geometrické transformace obrazu se skládá ze dvou kroků. Prvním z nich je transformace souřadnic jednotlivých bodů na nové pozice $(x',y') = T(x,y)$, tím druhým je aproximace jasové funkce v souřadnicích, jež se nacházejí mimo pravidelnou mřížku obrazu a jejichž intenzitu neznáme (více v 3.2) (Sonka et al.).

3.1.1 Afinní transformace

Afinní transformace je typ transformace, která zahrnuje všechny výše zmíněné operace – posun, rotaci, změnu měřítka i zkosení. Všechny tyto operace jsou kromě posunu `lineární`[5] a jejich složení můžeme reprezentovat maticí $\mathbf{B} \in \mathbb{R}^n \times \mathbb{R}^n$. Lineární transformace jsou obecně globálního charakteru, protože se aplikují v rámci celého obrazu. Celkovou afinní transformaci můžeme pomocí matice $\mathbf{B}$ a vektoru posunutí $\mathbf{c} \in \mathbb{R}^n$ zapsat jako (Hughes – Foley, 2013)

$$T(\mathbf{x}) = \mathbf{B} \cdot \mathbf{x} + \mathbf{c}. \tag{3.1}$$

3.1.2 Homogenní souřadnice

Aby bylo možné vyjádřit všechny možné afinní transformace (včetně posunu) jedinou maticí $\mathbf{A}$ a složitější transformace skládat „pouhým“ násobením dílčích transformačních matic (3.4)-(3.5), používá se v oblastech zpracování obrazu a grafiky převod do `homogenních souřadnic` (Hughes – Foley, 2013). Ten převádí body z n dimenzionálního prostoru do prostoru o jednu dimenzi vyššího, $n+1$ dimenzionálního prostoru homogenních souřadnic. Ve 2D případě se jedná o převod $\mathbb{R}^2 \to \mathbb{R}^3$ jednotlivých bodů $(x,y) \to (x,y,1)$. Transformaci T poté můžeme v homogenních souřadnicích vyjádřit jako

$$T(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x}, \tag{3.2}$$

kde $T : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$, popř. $\mathbf{x} \in \mathbb{R}^{n+1}$ a $A \in \mathbb{R}^{n+1} \times \mathbb{R}^{n+1}$. Afinní transformaci ve 2D můžeme po zavedení homogenních souřadnic rozepsat jako

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{12} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{3.3}$$

[4] Myšleno ve smyslu rigidní transformace, nerigidní nebo elastická transformace (non-rigid or elastic transformation) zahrnuje i lokální deformace (warping).

[5] Splňují princip superpozice, tj. $T(u + a \cdot v) = T(u) + a \cdot T(v)$ pro jakékoli vektory u, v a $a \in \mathbb{R}$ (Hughes – Foley, 2013).

Z tohoto zápisu je patrné, že k popisu afinní transformace ve 2D potřebujeme nejvýše 6 parametrů ($a_{11}, a_{12}, a_{13}, a_{21}, a_{12}$ a a_{23}) a k popisu ve 3D potřebujeme nejvýše 12 parametrů.

Jednotlivé matice operací, jejichž násobením[6] celkovou transformační matici **A** skládáme, jsou (Burger – Burge, 2009):

- `matice posunu` s parametry d_x a d_y, které postupně značí posunutí obrazu ve směru osy x a y,

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

- `matice rotace` s jediným parametrem ϕ, která způsobuje otočení obrazu o úhel ϕ se středem rotace v bodě $[0,0,1]$

$$\mathbf{R} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

- `matice změny měřítka` s parametry s_x a s_y, které zvětšují či zmenšují obraz ve směru osy x a y,

$$\mathbf{S} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

- a `matice zkosení` se dvěma parametry h_y a h_y, díky nimž se obraz zkosí ve smyslu osy x anebo y

$$\mathbf{H} = \begin{bmatrix} 1 & h_x & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.7}$$

Celková transformační matice **A** potom vypadá například jako[7]

$$T(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} = (\mathbf{R} \cdot \mathbf{S} \cdot \mathbf{D} \cdot \mathbf{H}) \cdot \mathbf{x}. \tag{3.8}$$

3.1.3 Typy transformační funkce

Podle používaných operací rozlišujeme několik typů afinních transformací. Čím je určitá transformace v seznamu níže, tím je obecnější a zachovává méně vlastností původního obrazu (je vůči nim invariantní). Afinní transformace obecně zobrazuje přímky na přímky (případně body) a zachovává poměry mezi zobrazovanými body. Typy afinní transformace jsou (Fitzpatrick et al., 2010):

[6] Tady jen poznamenejme, že pro násobení matic neplatí komutativní zákon a záleží tedy na pořadí, v jakém elementární operace násobíme - typicky zprava doleva.

[7] Pořadí těchto matic je jen jedno z možných. Toto pořadí bude ale použito při implementaci algoritmů, a proto je zde uvedeno.

– **Rigidní transformace (Rigid transformation)** zahrnuje operace posunu a rotace. Ve 2D ji popisují 3 parametry – posun podél os x a y a jeden rotační úhel (d_x, d_y, ϕ). Transformační funkci T poté vyjadřujeme jako $T(\mathbf{x}) = (\mathbf{R} \cdot \mathbf{D}) \cdot \mathbf{x}$.

– **Podobnostní transformace (Similarity transformation)** přidává k rigidní transformaci uniformní změnu měřítka ve směru všech os (ve 2D x, y), a tedy 1 parametr k jejímu popisu $s_x = s_y = s$ z (3.6). Podobnost zachovává úhly a platí, že $T(\mathbf{x}) = (\mathbf{R} \cdot \mathbf{s} \cdot \mathbf{D}) \cdot \mathbf{x}$.

– **Transformace se změnou měřítka**[8] **(Scaling transformation)** využívá operace posunu, rotace a neuniformní změnu měřítka. Pro přesný popis (určení) této transformace potřebujeme ve 2D znát 5 parametrů, $(d_x, d_y, \phi, s_x, s_y)$. Tato transformace jako poslední ze seznamu zachovává vzdálenosti. Platí, že $T(\mathbf{x}) = (\mathbf{R} \cdot \mathbf{S} \cdot \mathbf{D}) \cdot \mathbf{x}$.

– **Obecná afinní transformace (Afinne transformation)** je nejobecnější transformací z předloženého výčtu. Zahrnuje posun, rotaci, změnu měřítka i zkosení a v obraze zachovává pouze rovnoběžnosti přímek, délky úseček, ani úhly mezi nimi už nikoli (Mani et al., 2013). Pro přesné určení funkce T obecné afinní transformace potřebujeme znát všech 7 parametrů elementárních operací (3.4)-(3.5) $(d_x, d_y, \phi, s_x, s_y, h_x, h_y,)$ a platí, že $T(\mathbf{x}) = (\mathbf{R} \cdot \mathbf{S} \cdot \mathbf{D} \cdot \mathbf{H}) \cdot \mathbf{x}$.

Při registraci obrazů se používají i jiné typy transformačních funkcí, ty už ale nejsou afinní (mj. nezachovávají rovnoběžnosti přímek) a pro jejich přesný popis je potřeba mnohem více parametrů (popis nelineárních a lokálních deformací někdy vyžaduje stovky až tisíce parametrů (Valsecchi et al., 2013)).

3.2 Interpolace obrazu

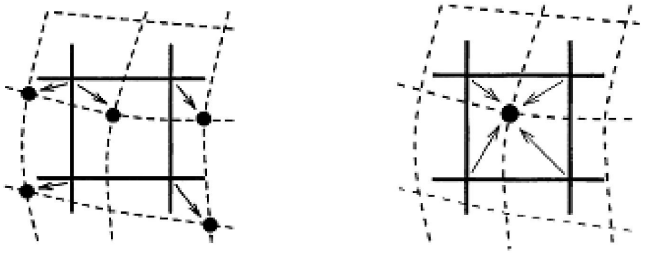
Součástí geometrické transformace obrazu je i odhad hodnot jasové funkce I v nových souřadnicích (x', y'). Nové souřadnice obrazu totiž obecně nepasují přímo na pravidelnou mřížku pixelů původního obrazu a jejich hodnoty musíme dopočítat (intenzity pixelů jsou definovány pouze v diskrétním rastru obrazu). Hodnoty těchto pixelů jsme schopni odhadnout interpolací intenzit pixelů v jejich okolí (Burger – Burge, 2009). Třemi běžně používanými metodami jsou metoda nejbližšího souseda, bilineární interpolace a bikubická interpolace (resp. trilineární a trikubická interpolace pro 3D) (Burger – Burge, 2009). Principy těchto metod jsou následující:

Nejbližší soused (Nearest neighbour interpolation) je výpočetně nejjednodušší metodou. Hodnota intenzity jasové funkce I v nových souřadnicích vznikne okopírováním intenzity nejbližšího souseda (pixelu) v původní mřížce (viz obrázek 6.3a). Nevýhodou této metody je vznik „zubatých“ hran
(Sonka et al.).

[8] Jedná se o volný překlad.

Bilineární interpolace (Bilinear interpolation) intenzita bodu v novým souřadnicích se vypočítá jako vážený průměr 4 pixelů původního obrazu v jeho nejbližším okolí (obrázek 6.3b). Váha každého pixelu z okolí klesá se vzdáleností od počítaného pixelu (Burger – Burge, 2009). Podle stejného principu, ale s 8 okolními body, se počítá i trilineární interpolace ve 3D. Nevýhodou této metody je to, že na rozdíl od metody nejbližšího souseda může do obrazu zanášet intenzity, které se v něm předtím nenacházely, ale na druhou stranu řeší problém „zubatých“ hran (Sonka et al.).

Bikubická interpolace(Bicubic interpolation) používá pro stanovení nových intenzit kubické polynomy z okolí počítaného pixelu o velikosti 4×4 - 16 pixelů. Při použití této interpolace bývá výsledný obraz vyhlazenější než při použití interpolace bilineární, ale také výpočetně náročnější.



(a) nejbližší soused (b) bilineární

Obrázek 3.1: **Schémata běžných interpolací**. Ve schématech je vždy původní obraz znázorněn tučně a transformovaný obraz je vykreslen přerušovanou čarou. Šipky v obrázku naznačují, ze kterých pixelů se nové hodnoty pixelů v transformovaném obrazu interpolují. zdroj: (Sonka et al.)

3.3 Podobnostní míra

Míru podobnosti (překryvu) dvou obrazů, hodnotíme funkcemi, které označujeme jako `podobnostní míry` $f_{similarity}$ (similarity metrics). Po jejich zavedení lze celý proces registrace formulovat v duchu optimalizační úlohy (2.1) jako proces hledání vektoru transformačních parametrů (transformation parameters-based IR approach). Registrační proces můžeme nad množinou různých transformačních funkcí T vyjádřit jako (Valsecchi et al., 2013)

$$\text{argmin}_T \, f_{similarity}(I_M, T(I_S)). \tag{3.9}$$

Cílem registrace je nalézt optimum funkce $f_{similarity}$ (transformační parametry), pro který platí, že v něm lícují registrované obrazy nejlépe.

Při registraci medicínských obrazů se používá široká škála podobnostních metrik. Volba metriky je z důvodu hodnocení kvality registrace různých obrazů[9] rozhodujícím

[9]Co se týká jeho rozložení intenzit.

faktorem. Přehled používaných metrik i s vhodností jejich použití lze najít v (Hill et al., 2001). Pro účely této práce si popišme jen metodu, kterou budeme používat v praktické části, metodu nejmenších čtverců.

Metoda nejmenších čtverců (Mean Squre Error, MSE[10]) je podobnostní metrika, již se snažíme během registračního procesu minimalizovat – čím je hodnota MSE nižší, tím je slícování kvalitnější. Optimální hodnotou této funkce je 0. Předpokladem použití této metody je stejné rozložení intenzit (šedé barvy) v obou obrazech (I_M, I_S) (Mani et al., 2013). MSE se hodí pro registraci monomodálních obrazů, ve kterých se vyskytuje málo šumu[11] a tato metoda je také citlivá na velké rozdíly intenzit v obraze, které mohou být například způsobeny vstřikovanou látkou, která se může používat během vyšetření (Hill et al., 2001). V praxi se většinou kvůli šumu optima (0) nedosahuje (Feng, 2008). Hodnota MSE se vypočítá podle vzorce (Hill et al., 2001)

$$MSE = \frac{1}{N}\sum_{i=0}^{N}(I_M(i) - T(I_S(i)))^2, \quad \forall i \in I_M \cap I_S, \tag{3.10}$$

kde $I_M(i)$ značí hodnoty intenzit referenčního obrazu v souřadnici i a $T(I_S(i))$ hodnoty intenzit transformovaného plovoucího obrazu v souřadnici i. Při počítání hodnoty `MSE` je nutné si ohlídat dostatečný průnik obrazů, aby ji mělo smysl počítat (viz 5.4).

Teď již známe všechny patřičnosti optimalizace a registrace. Popišme si proto, jak registrace obrazů probíhá.

3.4 Průběh registrace

Proces optimální registrace probíhá podle schématu na obrázku 3.2. Registrace má dva vstupy: referenční obraz I_M a plovoucí obraz I_S. Během registračního procesu iterativně zlepšujeme odhad transformace T (její parametry), kterou používáme k zaregistrování vstupních obrazů. Během registrace se snažíme najít takovou transformaci T, která zobrazuje plovoucí obraz I_S do prostoru referenčního obrazu a které zajistí co nejlepší překryv zobrazovaných objektů.

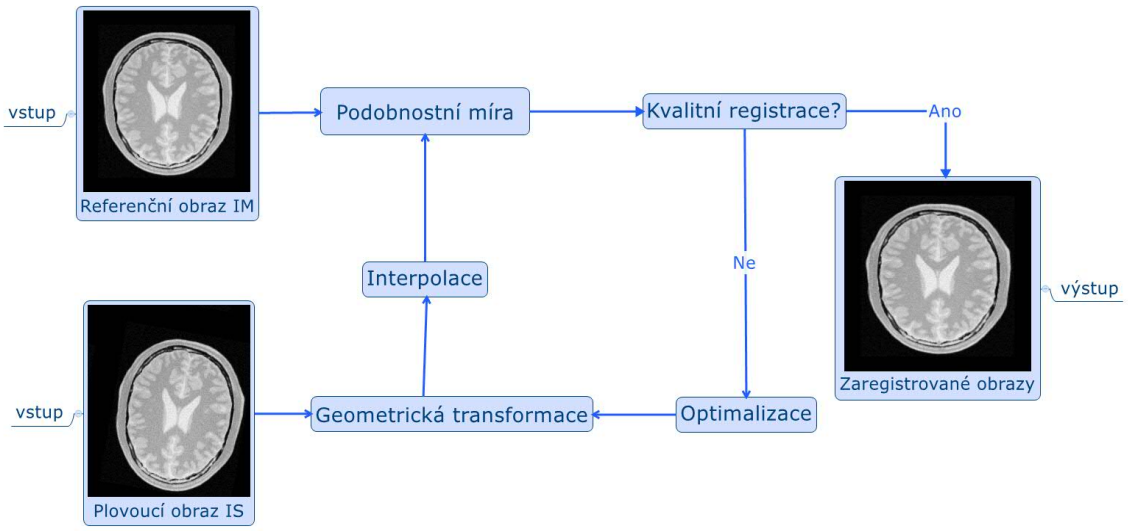
Abychom toho ale byli schopni, musíme si nejprve zobrazit všechny pixely referenčního obrazu do prostoru plovoucího obrazu a jejich intenzity interpolovat pomocí okolních pixelů v plovoucím obraze. Tyto nově spočítané intenzity porovnáváme s hodnotami intenzit odpovídajících si pixelů referenčního obrazu. Důvodem této `zpětné transformace` je to, že při zobrazení plovoucího obrazu přímo do prostoru referenčního obrazu by se některé pixely plovoucího obrazu mohly zobrazit do stejného pixelu nebo naopak by v některých částech obrazu mohly vznikat díry - oblasti, do kterých by se žádné pixely nepromítly a my bychom jejich intenzity neznali, anebo do třetice by se pixely mohly zobrazit mimo

[10] Někdy se také označuje jako SSD (Sum of Squared Differences), ačkoli překlady vyjadřují něco jiného.

[11] Šumem označujeme náhodné změny intenzit pixelů v obrazu. Při pořizování obrazů z MRI bývá šum způsoben samotným přístrojem, vlivem okolních elektronických zařízení nebo fyziologickými projevy organismu (Drastich, 2004).

obraz (Burger – Burge, 2009). Zobrazením referenčního obrazu do prostoru plovoucího obrazu, zpětnou transformací, jsme naopak schopni porovnávat všechny odpovídající si pixely obou obrazů.

Po tomto zobrazení („uměle“ vytvořené transformaci obrazu I_S) spočítáme pomocí zvolené podobnostní míry kvalitu slícování (porovnáme intenzity jednotlivých pixelů I_M a již transformovaného I_S). Pokud je registrace dostatečně kvalitní[12], registrace se nějakou dobu nezlepšuje nebo již proběhl určitý počet iterací, registrační proces se ukončí, pokud ne, parametry transformace T postupují do další iterace. Výstupem celé optimalizační procedury je obraz slícovaných (zaregistrovaných) obrazů a vektor parametrů „registrační“ transformace, který transformuje plovoucí obraz na obraz referenční [13].



Obrázek 3.2: **Schéma optimální registrace medicínských obrazů.**

3.5 Nejpoužívanější metaheuristiky pro registraci obrazů

Hledání minima nebo maxima podobnostní míry je většině případů multimodální optimalizační úloha s mnoha lokálními optimy (Valsecchi et al., 2013). Tradiční numerické techniky založené na gradientu podobnostní míry dnes umožňují spolehlivé zaregistrování obrazů (nalezení globálního optima), ale pro jejich použití je nutný hrubý odhad transformačních parametrů, ze kterého registrace započne. Tento odhad ovšem většinou poskytuje nějaký odborník a pokud my chceme metody zpracování obrazu plně automatizovat, není tento zásah odborníka možný. Metaheuristické algoritmy představují způsob, kterým lze

[12] Jedná se o parametr procedury.

[13] Ještě ale raději upřesněme, že v praxi většinou nehledáme tuto transformaci T, která mapuje plovoucí obraz I_S na referenční obraz I_M, ale inverzní transformaci T^{-1} (odpovídá inverzní matici transformace A^{-1} a $T^{-1}(\mathbf{x}) = A^{-1} \cdot \mathbf{x}$) která namapuje referenční obraz na obraz plovoucí.

dosáhnou globálního řešení problému i bez počátečního odhadu.

Třemi nejpoužívanějšími MH metodami pro IR jsou podle Valsecchi et al. (2013) i dalších Optimalizace hejnem, Simulované žíhání a Genetický algoritmus. Jinými MH algoritmy, které byly na registraci obrazů dále použity, jsou Diferenciální evoluce, Kovarianční evoluční strategie (Covariance Matrix Adaptation Evolution Strategy, CMA-ES) a Rozptýlené hledání (Valsecchi et al., 2013; Damas et al., 2011; Valsecchi et al., 2014).

V případech registrace obrazů, kdy nemáme k dispozici žádné počáteční znalosti o problému, se často používá kombinace metaheuristických algoritmů s jinými běžně používanými lokálními metodami. Globální metaheuristikcé metody slouží k nalezení hrubého odhadu registrace a lokální metody slouží k jeho zpřesnění. Viz výše zmíněné články a jejich rešerše.

Kapitola 4

Vybrané metaheuristické algoritmy

Tato kapitola se věnuje dvěma vybraným metaheuristickým algoritmům (ABC, ES), které budou v praktické části aplikovány na problém lícování medicínských obrazů mozku. Oproti předchozím (již zmíněným) optimalizačním metodám se budeme více věnovat jejich biologické předloze i jejich funkčnímu konceptu a to tak, abychom je ve výsledku mohli porovnat. Zaměříme se hlavně na jejich klíčové myšlenky a vztahy a souvislosti k optimalizaci.

4.1 Algoritmus umělých včelích kolonií

Algoritmus ABC řadíme mezi algoritmy SI a jak již bylo uvedeno v podkapitole 2.5, jeho biologickou předlohou je včelí způsob hledání potravy. Hlavní myšlenky ABC[1] vychází především z vlastností sociálního chování hmyzu, které nacházejí uplatnění (svůj obraz) i v optimalizačním procesu. Těmito vlastnostmi jsou (Karaboga, 2005):

- **samoorganizace (self-organization)** označuje proces spolupráce samostatných a rovnocenných jedinců (včel) bez centrálního vedení, který ve výsledku přináší jakýsi řád celému společenství (včelstvu) (Camazine, 2003). V optimalizaci si pod tím můžeme představit populaci spolupracujících agentů, kteří svých chováním přispívají ke společnému cíli. Tím cílem zde míníme konvergenci algoritmu (včelstva) ke globálnímu extrému daného problému.

- **dělba práce (division of labour)** umožňuje jednotlivým skupinám jedinců specializovat se na určitý úkol. Tato specializace vede k efektivnějším výsledkům celku (včelstva) než při uniformní práci všech jedinců a současně umožňuje jedincům lépe reagovat na změnu podmínek (Oster – Wilson, 1978). Agenti v ABC se specializují na lokální anebo globální prohledávání prostoru přípustných řešení (více v 4.1.2).

Dalším důležitým rysem ABC, ale i většiny ostatních SI algoritmů, je schopnost komunikace jedinců nebo agentů mezi sebou[2]. Podívejme se už ale na algoritmus ABC trochu konkrétněji a ukažme si jednotlivé rysy a důvody inspirace včelami přímo na něm. A popišme si nejdříve jeho biologickou předlohu a až poté funkční princip a jeho průběh.

[1] Ale i jiných metaheuristik inspirovaných sociálním chováním hmyzu, ptáků, ryb atp.

[2] Jen upozorníme, že pojmy komunikace a interakce vyjadřují něco jiného. Interakce je podstatou samoorganizovatelnosti systému, komunikace je přímým sdělováním informací.

4.1.1 Biologické pozadí ABC

Včela medonosná *(Apis mellifera)* je typickým zástupcem eusociálního[3] hmyzu, pro niž je příznačná dělba práce. Práci si včely rozdělují v rámci kastovního systému (matka, dělnice a trubci), ale také v rámci jednotlivých tříd. Nejpočetnější třídou a pracující složkou včelstva jsou dělnice. Ty během života vykonávají mnoho povolání, jejichž průběh není sice pevně daný, ale respektuje nějaký řád [4]. V případě potřeby mohou dělnice zaskakovat na pozicích, kde je to zrovna zapotřebí (Reichholf-Riehmová, 1997; Veselý, 2003).

Trochu zjednodušeně si můžeme dělnice rozdělit na `mladušky` a `létavky`. Mladušky jsou mladé dělnice, starající se o potřeby včel v úlu, a létavky jsou starší včely, které vykonávají práce mimo úl (Veselý, 2003). Hlavním úkolem dělnic ve stádiu létavky je starost o potravu. Ta zahrnuje sběr nektaru, pylu, vody a smůly, propolisu. Dělnice v tomto stádiu života vylétávají z úlu, prozkoumávají okolí, sbírají potravu a pouze o nejvýnosnějších snůškách[5] informují ostatní včely v úlu. To z toho důvodu, aby jim pomohly se sběrem (s jejich vyčerpáním). A to nejlépe co nejdříve, aby je někdo nepředběhl, nepadla tma nebo se např. nezměnilo počasí (Seeley, 2009).

Komunikačními nástroji včel jsou `tanec` (waggle dance) a feromony (Von Frisch, 1967). Pomocí nich předávají dělnice-létavky ostatním dělnicím v úlu informace o slibných snůškách. Tancem létavky sdělují svým družkám vzdálenost snůšky od úlu, směr ke snůšce, bohatost potravy a její vůni. A feromony, látkami, které včely z těla vylučují, si zase létavky nalezené snůšky značkují, což pomáhá ostatním včelám v jejich nalezení (Seeley, 2009). Experimentálně bylo vypozorováno, že počet tančících včel ze všech průzkumnic činí méně než 10 % a hodnota kvality zdroje (s ohledem na další kritéria jako je vzdálenost od úlu, dostupnost nektaru atp.), od které jsou tance prováděny, lineárně závisí na okamžitém množství potravy, které do včelstva proudí (Abou-Shaara et al., 2014; Seeley, 2009).

Abychom mohli této včelí předlohy při optimalizaci využívat, mělo by teoreticky platit, že by včely měly být schopny detekovat a od ostatních možných zdrojů rozlišit ten nejvýnosnější, samozřejmě se zřetelem na vzdálenost, kvalitu, množství potravy atp. A co víc, pokud bychom chtěli brát inspiraci včelami opravdu vážně, mělo by také platit, že čím kvalitnější nebo vydatnější snůška, tím více by měl onen taneček konkrétní včely vzbuzovat pozornosti a nalákat více dělnic[6]. To, že to tak v přírodě opravdu funguje, lze najít např. v knihách Seeley (2010, 2009).

Další důležitou vlastností včel při získávání potravy je schopnost orientovat se v čase

[3] Jedná se důmyslnější formu společenského uspořádání než u sociálního hmyzu, které jedincům přináší jisté výhody. Pravidly eusociálního společenství jsou: jedinci se starají o své potomky, společnost je vícegenerační, jedinci se rozlišují na reprodukční jedince a nerozmnožující se jedince a všichni jedinci se starají o blaho celého společenství.

[4] Tento řád je dán především fyziologickým vývojem včel.

[5] Snůškami označujeme rostliny poskytující včelám hlavní zdroje potravy (men, aktualizace 02.05.2014).

[6] Protože je hodně pravděpodobné, že i oblast kolem kvalitního zdroje je bohatá na potravu. Ta potom vyžaduje intenzivnější průzkum než oblasti kolem méně kvalitních zdrojů.

a prostoru a měřit vzdálenosti mezi objekty. Pro vysvětlení těchto jevů (nebo jejich případných hypotéz) už jen odkazujeme na příslušnou literaturu (Towne – Moscrip, 2008; Abou-Shaara et al., 2014; Menzel et al., 2005) a pro potřeby další části práce jen uvedeme, že prostředí má podle Barron et al. (2005) vliv na rychlost letu včely a tedy i odhad uletěné vzdálenosti. To protože včely měří vzdálenosti na základě zrakového toku (Abou-Shaara et al., 2014).

Jinou pracovní pozicí dělnic, která má v ABC také svou úlohu, je stádium tzv. `skautky`. Úkolem skautek v přírodě je hledat nové a na potravu bohaté snůšky. Skautkami jsou proto nové včely létavky, které právě přešly do stádia, ve kterém začínají mít na starost sběr potravy, anebo jsou jimi létavky, jejichž upřednostňovaná snůška byla vyčerpaná a ony se musí poohlížet po nové (Seeley, 2009)[7].

4.1.2 Samotný algoritmus ABC

Teď trochu obraťme list a podívejme se na to, jak ABC funguje. Pro zasazení do kontextu optimalizace ještě upřesněme, že množině zdrojů potravy odpovídá množina kandidátních řešení optimalizačnímu problému a kvalita zdroje koresponduje s kvalitou řešení daného problému (vhodnost - fitness).

Na tomto místě je ještě vhodné zdůraznit, že ABC se omezuje pouze na způsob získávání potravy včelami a využívá pouze některé principy tohoto procesu. Proto se v něm vyskytují pouze včely létavky.

Celý algoritmus ABC stojí na interakcích a spolupráci tří druhů agentů (umělých včel). Ty rozlišujeme na následující druhy včetně jejich úloh (Karaboga, 2010, 2005; Veselovský – Dungel, 2005):

- **zpravodajky (employed bees)** každá zpravodajka je vždy přiřazena k jednomu řešení daného problému (zdroji). Jejím úkolem je hledat v okolí přiřazeného řešení ještě kvalitnější řešení daného problému a o tom nejvýnosnějším informovat rekrutky během „včelího tance“[8]. V metafoře přírody odpovídají zpravodajkám tančící dělnice, které svým tancem sdělují ostatním dělnicím v hnízdě „lokaci“ výnosného zdroje potravy.
- **rekrutky (onlooker bees)** úkolem rekrutek je následovat jednu vybranou zpravodajku a v okolí jejího řešení také hledat kvalitnější řešení. Počet rekrutek následujících jednu zpravodajku je vždy přímo úměrný kvalitě jejího řešení. V případě nalezení lepšího řešení, než je to aktuální sledované (řešení zpravodajky), dochází

[7] Dovolme si ještě poznamenat, že v některé literatuře (Seeley, 2010) se skauty označují i jedinci, kteří hledají novou lokaci k usazení nového roje v případě narození nové matky (v roji může být přítomna vždy jen jedna). Komunikačním prostředkem pro sdílení jejich lokace je zase taneček, kterým skautky opět verbují skautky-kolegyně. I v případě tohoto tanečku tančí skautky pouze, pokud je nalezené stanoviště dostatečně dobré. Ve výběru potom vyhrává to místo, kam se nahloučí (pro jež se rozhodne) nejvíce skautek (Seeley, 2010).

[8] Včelí tanec má v algoritmu pouze symbolický význam. Takovým jeho ekvivalentem je rozšíření znalostí o známých kandidátních řešeních po celém včelstvu. Slouží tedy jako nějaká zásobárna informací.

k jeho nahrazení a tím i k „přesunu“ zpravodajky k novému zdroji. V řeči optimalizace slouží rekrutky a zpravodajky jako lokální prohledávací techniky slibných oblastí prostoru přípustných řešení.

- **průzkumnice (scout bees)** průzkumnice se v algoritmu nevyskytují pořád. Stávají se jimi zpravodajky, kterým se v určitém počtu iterací[9] nedaří v okolí jejich řešení (zdroje) nalézt kvalitnější řešení, než je to jejich. Pokud se tak stane, průzkumnice si náhodně zvolí bod v prostoru přípustných řešení (hledané parametry) a ten se stane jejím novým řešením (zdrojem). Hned na to se průzkumnice promění zpátky v rekrutku. Účelem existence průzkumnic je hledání optima v nových lokalitách[10]. V biologické předloze odpovídají průzkumnicím dělnice, které hledají nové snůšky nebo vhodné lokality pro případné založení nového roje. Tito agenti byli do algoritmu navrhnuti tak, aby v prostoru řešení vyhledali jakékoli řešení, nejen to dobré. A proto tito agenti ve výsledku nachází lepší řešení pouze zřídkakdy. Důležité ale je, že zvyšují „diverzitu“ známých řešení.

Schématický náčrt úloh jednotlivých druhů agentů-dělnic a fungování algoritmu je znázorněno na obrázku 4.1.

Samotný výpočet algoritmu ABC začíná volbou počáteční populace řešení (zdrojů potravy)[11]. Poté dojde k ohodnocení kvality každého zdroje, co se týká jeho vhodnosti (fitness) jako řešení problému, které následuje cyklické prohledávání prostoru přípustných řešení. Tento cyklus probíhá ve třech fázích. Fáze jsou prováděny jednotlivými druhy agentů - po řadě nejdříve zpravodajkami, rekrutkami a nakonec průzkumnicemi až do ukončení algoritmu. K ukončení algoritmu dojde ve chvíli, pokud je nalezeno dostatečně kvalitní řešení daného problému nebo vyprší časový limit[12] pro výpočet algoritmu. Výstupem algoritmu ABC je nejkvalitnější řešení (nejvydatnější zdroj potravy) nalezený v rámci celého běhu algoritmu.

Počáteční populace zdrojů potravy může být v prostoru přípustných řešení zvolena náhodně nebo může vycházet ze znalostí o řešené úloze, stejně tak způsob implementace prohledávání okolí jednotlivých zdrojů. Implementační detaily ABC budou popsány až v 5.3.2.

4.1.3 Biologie vs. „model“ ABC

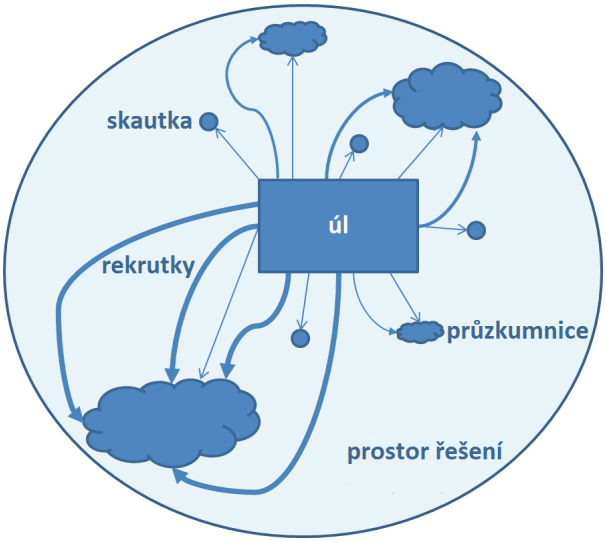
Hlavním účelem předchozím dvou podkapitol bylo propojit biologickou a informatickou podstatu algoritmu ABC. Nyní se pokusme tento vybudovaný základ porovnat s tím, jak to v přírodě skutečně probíhá. Z obou předchozích podkapitol plyne, že autor algoritmu se neinspiroval celým procesem včelího hledání potravy, ale jen některými jeho aspekty.

[9] Jedná se o řídicí parametr algoritmu.

[10] V literatuře týkající se algoritmu ABC se to tento proces někdy označuje jako opuštění zdroje (abandonment) z důvodu jeho vyčerpání (exhaustion) (Karaboga, 2010).

[11] Počet zdrojů je shodný s počtem agentů zpravodajek, protože každá zpravodajka bývá vždy přiřazena k jednomu zdroji.

[12] Obojí jsou opět parametry algoritmu.



Obrázek 4.1: **Základní schéma rolí agentů v algoritmu ABC**. Obsah kruhu na obrázku představuje prostor řešení problému. Obdélník ve středu obrázku je místem, kde se kumulují informace o známých řešeních - úlem a tanečním parketem. Tmavě modré mraky a stejně zbarvené kruhy představují konkrétní řešení daného problému, po řadě průzkumnice a skautky. Velikost mraku koreluje s kvalitou řešení problému a řešení skautek jsou všechna stejně velká, protože se jedná o dosud neohodnocená řešení. Šipky v obrázku označují „pohyb“ jednotlivých agentů k určitému řešení, jehož okolí budou prozkoumávat. Šipky s tenkými čarami symbolizují příslušného agenta, kterému dané řešení přísluší - průzkumnice či skautka - a tloušťka čar zbylých, silnějších šipek, se shoduje s pravděpodobností, s níž je daná průzkumnice pronásledována rekrutkami. Počet silnějších šipek vedoucích k daným řešením (mrakům) odpovídá počtu rekrutek v daném okolí známého řešení a koresponduje s kvalitou řešení (velikostí mraku). zdroj: (nab, aktualizace 20.02.2015), upraveno

V první části porovnávání se proto zaměřme na to, které komponenty přírodního procesu autor v algoritmu zužitkoval, které nevyužil, a pokusme se odhadnout, proč tomu tak je/bylo. V druhé části porovnávání se k hledání paralel mezi biologií a algoritmem ABC postavíme zase z opačného konce a budeme hledat přírodní procesy za (odpovídajícími) částmi algoritmu, ve kterých příroda jako zdroj inspirace nesloužila.

Na základě výše napsaného bychom mohli říci, že v průběhu miliónů let existence si včely osvojily opravdu schopný aparát pro řešení řady optimalizačních úloh. Pro zopakování, jednotlivými úlohami jsou (viz 4.1.1):

- Včely jsou schopny si z mnoha dostupných zdrojů potravy vybírat ty výhodné, stejně jako z potenciálních umístění nového roje to nejlepší (Seeley, 2010).
- Jsou schopny přizpůsobovat počty dělnic jednotlivých pracovních skupin v závislosti na změnách okolních podmínek.

V řešení první úlohy včely volí heuristický přístup[13]. Pro upřesnění heuristický proto,

[13] Alternativní verze této (těchto) skutečnosti je zase zobecnitelná na většinu druhů sociálních hmyzu a je také důvodem, proč se chováním právě sociálního hmyzu nechalo inspirovat nespočet autorů optimalizačních algoritmů.

protože z množiny všech řešení, všech potencionálních zdrojů potravy (nebo hnízdišť), prozkoumávají průzkumnice a skautky jen část, a jak jsme si dříve popsali, jsou schopny hodnotit jejich výnosnost. Důsledkem zpracování těchto informací a jejich sdílením dochází ve včelstvu k výběru toho nejvýhodnějšího známého řešení. Experimentálně podaný důkaz, že se opravdu jedná o nejvýhodnější volbu (alespoň pokud jde o volbu hnízdiště a svým způsobem i potravy), podává ve své práci Seeley (2010, 2009). Tímto bychom mohli logicky zdůvodnit rozhodnutí autora inspirovat se při návrhu algoritmu ABC zrovna tímto dějem.

- V algoritmu vidíme výběr nejlepšího známého zdroje potravy ve volbě rekrutky pronásledovat konkrétní průzkumnici a ve schopnosti průzkumnic informovat včelstvo jen o dostatečně kvalitních řešeních. Tato druhá schopnost již ale není využit u umělých včel typu skautka, které informují ostatní kolegyně o prvním nalezeném řešení, nehledě na jeho kvalitu.

- Přizpůsobení počtů agentů v rámci jednotlivých pracovních skupin se v algoritmu promítá do počtu „nově“ vzniklých skautek. Ke vzniku skautek dochází v případech, kdy to vypadá, že zpravodajky uvízly v lokálním nebo globálním optimu funkce vhodnosti (v jejich okolí nedochází k nalezení lepších řešení). Zde se opět jedná o situaci, která není v základní verzi algoritmu ABC plně využita, protože povoluje jen jednu přeměnu zpravodajka-skautka-zpravodajka za cyklus. Samotný autor záhy po zveřejnění algoritmu ABC tuto okolnost upravil, což vedlo ke zlepšení konvergenčních vlastností algoritmu, protože tato úprava pomáhá v algoritmu uniknout z lokálních extrémů (Kiran – Gündüz, 2014).

Dalším důležitým rysem života včel, který v algoritmu sehrává zásadní roli, je schopnost včel kvalitativně hodnotit výnosnost jednotlivých snůšek, popř. nových umístění pro včelstvo (viz 4.1.1).

- Schopnost včel hodnotit výnosnost potravy se v algoritmu odráží v tom, že jsme schopni počítat vhodnost všech přípustných řešení dané optimalizační úlohy a vůbec porovnávat jednotlivá řešení z hlediska jejich kvalit.

Autor v návrhu optimalizačního algoritmu ABC naopak nevyužil následující rysy:

- Feromony. Jak již bylo popsáno v 4.1.1, feromony jsou druhým způsobem komunikace včel (prvním jsou tanečky). Tato komunikační dráha nebyla v algoritmu použita pravděpodobně z toho důvodu, protože u včel mají feromony úplně jiné využití, než u dříve popsaných mravenců[14]. Feromony u včel nefungují jako záznam o průběhu prohledávání prostoru, ale jako rozpoznávající znamení, popř. mechanismus ovlivňující chování včel (Veselý, 2003). Pokud bychom přesto feromony v algoritmu použili a mylně předpokládali, že ve vzduchu zanechávají stopy, je velmi pravděpodobné, že by se tato komunikační dráha vylučovala se současným použitím včelího tance.

[14] Na samostatném způsobu dorozumívání se feromony stojí algoritmus představený v kapitole 2.5, ACO.

Některé verze algoritmu ABC, Fast Mutation ABC (Bi – Wang, 2011), o používání metody feromonů spekulují. V této verzi ABC se přepočtem vhodnosti známých zdrojů na hodnoty (hladiny) feromonů jednotlivých zdrojů upravuje volba rekrutek pronásledovat průzkumnice. Rekrutky pronásledují průzkumnice na základě náhodně vygenerovaného čísla, které porovnávají s hodnotou feromonů jednotlivých známých zdrojů. Podle výsledků porovnání se zdroj rozhodnou prozkoumat nebo ne. Cílem této metody je omezit restrikci výběru méně kvalitních zdrojů rekrutkami. A ačkoli dosahuje tato verze podle prováděných experimentů lepších výsledků než klasický ABC (se současnou úpravou chování skautek), používání „feromonů" omezuje účel (význam) tanečků jen na zásobárnu informací. Mechanismus feromonů nemá ani s přírodními feromony moc společného. O feromonech se zmiňuje i algoritmus Virtual bee algorithm (Yang, 2005), ten ale také nemá s feromony nic společného.

- Druhy tanečku. Ve skutečnosti včely v přírodě netančí jen jedním typem tanečku. Ekologové tanečky rozlišují podle vzdálenosti sdílené (vybrané) potravy od úlu, která se projevuje v rychlosti tance (Veselý, 2003). Toto zjednodušení ABC na jediný tanec je zcela na místě, protože v algoritmu se nic takového, jako je úl, nevyskytuje a jeho vzdálenosti od konkrétních řešení problému by v algoritmu nehrály žádnou roli.

A naopak vlastnosti včelího hledání potravy, které by teoreticky mohly být v optimalizačním procesu dále využity:

- Jedním z dějů života včel, který by se v algoritmu ABC mohl využít, je přelétávání včel z květu na květ. My totiž v přírodě zřídka vidíme včely čerpat nektar jen z jednoho zdroje, ale naopak střídat několik zdrojů v rámci určitého okolí. Toho by se mohlo využít při prohledávání okolí řešení zpravodajkami, rekrutkami, a jak bylo popsáno výše, i svým způsobem skautkami (skautky by nepodávaly v úlu informace o každém nalezeném řešení). Pro algoritmus by to ale znamenalo přidání minimálně jednoho řídicího parametru, který by bylo nutné před spuštěním algoritmu optimálně nastavit, a vícenásobný výpočet funkce vhodnosti, což je časově velmi náročná operace. Záleželo by potom na tom, jestli by konvergenční rychlost a vylepšování nalezených řešení převýšilo tyto nevýhody nebo ne. Více později.

- Další schopnosti včel, které by v algoritmu mohly být využity, jsou jejich schopnosti orientovat se v čase a prostoru. Inspiraci touto vlastností bychom ale nemohli brát příliš doslova, protože my optimalizuje neustále (čas) a orientace v prostoru vyžaduje znalosti o okolí. Odrazem těchto dějů do algoritmu by mohlo například být zakomponování derivace kriteriální funkce do jeho průběhu, protože derivace je jeden ze způsobů, kterým lze směrovat prohledávání do určitých lokalit. Při použití derivace by ovšem metaheuristický algoritmus už nebyl nezávislý na problému (k jeho použití by úloha musela splňovat jisté požadavky), a proto zde tuto možnost nebudeme rozvíjet.

V druhé části porovnávání se budeme věnovat jednotlivým rozšířením algoritmu ABC a jejich původně nebiologickému podkladu a naopak algoritmům, které byly podobně jako

ABC inspirované sháněním potravy včel. V odborné literatuře je dohledatelných spousta modifikovaných verzí ABC, jejich přehled lze najít např. v (Verma – Kumar, 2013) nebo i s hybridizacemi[15] v (Szeto et al., 2011). Kvůli omezenému rozsahu práce se zaměřme pouze na 2 verze[16], u kterých bylo nalezeno vylepšení s obrazem v přírodě, a na 2 algoritmy inspirované stejným zdrojem. Konkrétně půjde o Algoritmus umělých supervčel (Artificial super-Bee enhanced Colony, AsBeC), Opoziční ABC (Opposition-based artificial bee colony algorithm), Včelí algoritmus (Bees Algorithm) a Opravdovou včelí kolonii (Virtual Bee Algorithm)[17].

Algoritmus umělých supervčel se od představené základní verze ABC liší ve více ohledech. Jedním z nich je modifikace tzv. opožděného tance (postponed hive dance), která upravuje fáze rekrutek a průzkumnic a umožňuje tak zlepšit lokální prohledávání algoritmu (Ampellio – Vassio, 2014). Průzkumnice a rekrutky v něm neprovádí prohledávání okolí známého řešení (zdroje) pouze jednou za iteraci, ale vícekrát[18]. Obrazem této úpravy v přírodě by mohlo být výše popsané přelétávání včel z květu na květ. Experimenty s touto úpravou (v kombinaci s jinými úpravami) ve výsledku přinesly zrychlení a zlepšení konvergenčních vlastností algoritmu (Ampellio – Vassio, 2014).

Jinou alternativou přelétávání z květu na květ je využití opozičního učení (opposition-based learning) ve formě opozičního vektoru (opposition vector, princip), který se běžně používá ke generování nových kandidátních řešeních i v jiných metaheuristikách (Xu et al., 2014). U ABC se tato technika využívá různě - ke generování počáteční populace i ve formě skoků, které jsou s určitou pravděpodobností prováděny zpravodajkami (El-Abd, 2011). Ve fázi průzkumnic by se to realizovalo podle Ampellio – Vassio (2014) tak, že bychom opozičním vektorem prohledávali s každým náhodně zvoleným řešením (průzkumnice, skautka) v okolí známého řešení (zdroj, zpravodajka) i řešení, které je s náhodně vygenerovaným řešením vůči známému řešení v opozici.

Pham et al. (2006) je autorem Včelího algoritmu. Tento algoritmus je také inspirovaný chováním včel hledajících potravu. Rozdíl oproti zde představenému algoritmu ABC je v tom, že propouští do dalšího iteračního cyklu (jakási forma selekce) pouze řešení (zdroje potravy), která jsou dostatečně kvalitní. Zbytek populace dohledává svá řešení (zdroje) v prostoru přípustných řešení náhodně. Toto striktní vyřazování méně kvalitních řešení v algoritmu ABC, ani v přírodě nefiguruje, ale je jím řešen problém, že by včely předávaly v úlu všechny informace (náhodně nalezená nekvalitní řešení skautek jsou velmi rychle opuštěna).

Yang (2005) publikoval jinou verzi algoritmu stejné biologické předlohy, která se od diskutovaného ABC liší především v tom, že jedinci (umělé včely) vždy znají nejlepší nalezené řešení problému. Touto znalostí a svým chováním pohybu jedinců ve směru nej-

[15] Algoritmy jsou kombinovány s jinými optimalizačními technikami, kterými se autoři snaží eliminovat slabiny původních algoritmů.

[16] Jako rozšířená verze algoritmu je zde brána jen samostatné verze algoritmu, ne jeho upravené verze na konkrétní problém ani hybridní algoritmus.

[17] V případech všech těchto algoritmů jde o volné překlady originálních názvů.

[18] Počet těchto prohledávání je určen přidaným řídicím parametrem.

lepšího nalezeného řešení se tento algoritmus více podobá algoritmu PSO 2.5.2, než diskutovanému ABC (Yang, 2010a). Přírodní předloha by se hledala obtížně.

Pokud bychom si shrnuli celou kapitolu srovnání algoritmu ABC a jeho biologického podkladu, tak: Je pravda, že z celého biologického konceptu autor (autoři) extrahoval pouze určité komponenty, které mají v optimalizaci uplatnění, a doplnil o ne tolik přírodě odpovídající části. Ty ale mají v algoritmu svůj účel, což dokazuje i to, že ABC při optimalizaci funguje a v praxi se používá. Pro nás je momentálně podstatné, že funkční jádro a myšlenka přírodní předlohy zůstaly v algoritmu zachovány, protože potom můžeme ABC oprávněně nazývat optimalizačním algoritmem inspirovaným živou přírodou. Dále jsme si na dvou rozšířených verzích ABC a dvou myšlenkově podobných algoritmech ukázali, že včelstvo je pro optimalizační algoritmus/algoritmy bohatým zdrojem inspirace, který ještě nemusí být úplně vyčerpán. Každopádně téma vylepšení výkonu ABC je v oblasti optimalizace stále aktuální, hlavně pokud jde o jde jeho předčasnou konvergenci (Verma – Kumar, 2013).

4.2 Evoluční strategie

Nejpočetnější třídu metaheuristických algoritmů tvoří evoluční algoritmy, které k optimalizaci využívají naše představy o evoluci a evolučních procesech. Hlavní myšlenka těchto algoritmů, stejně jako celé biologie a evoluce, spočívá ve schopnosti přírody pracovat s minulostí. Evoluce je obecně chápána jako historický proces, v jehož průběhu se jako reakce na okolní podmínky mění genetické složení populace (Mayr, 2009). Mezi tyto techniky řadíme i algoritmus ES. Jeho princip obdobně stojí na iterativních změnách složení genofondu populace řešení.

Podívejme se teď na mechanismy Darwinovy evoluce trochu podrobněji a ze začátku.

4.2.1 Biologické pozadí ES

Navažme nejprve na úvod práce a racionální zdůvodnění toho, proč se lidé rozhodli zúročit procesy a děje známé z přírody v oblastech výpočetních technik a informatiky. Prvotním a poněkud fundamentálním předpokladem předloženého zdůvodnění je schopnost přírody se v čase vyvíjet. Ačkoli se to může zdát absurdní, víra lidí ve schopnost přírody se měnit, nebyla vždy tak samozřejmá jak je tom dnes. Velkou zásluhu na změně tohoto způsobu uvažování a svým způsobem i jakési „intelektuální revoluci“, která zpochybnila převládající víru lidstva ve stálost světa, měl Charles Darwin a jeho kniha *O vzniku druhů* z roku 1859 (Mayr, 2009). Darwin v ní předložil důkaz, a co víc, vysvětlení evolučně se měnícího se světa pomocí v přírodě běžně pozorovatelných jevů (Mayr, 2009). Jeho teorie měly jedinou slabinu - Darwin v nich sice správně[19] předpokládal, že jedinci dědí s malými rozdíly vlastnosti svých rodičů, ale nedokázal to vysvětlit (Henderson, 2014;

[19] Je to v souladu s dnešním většinovým povědomím o evoluci.

Zrzavý et al., 2004). Tyto okolnosti - mechanismus dědičnosti a variability skrze rekombinaci (z 2.4) - vysvětlují Mendelovy zákony. Propojení těchto dvou paradigmat, Darwinovy teorie a Mendelových zákonů, tvoří základní stavební kámen evolučních algoritmů[20].

Algoritmus ES opět řadíme mezi algoritmy populační (SI), proto i jeho zdroj inspirace vychází z vývoje konkrétní populace v čase. Takovou populaci v genetice označujeme pojmem lokální populace[21]. Vymezíme-li si oblast evolučního zájmu, můžeme více zkonkretizovat, oč v takto vymezené evoluci jde. V evoluci lokální populace hraje klíčovou roli vnitrodruhová konkurence, pod kterou si můžeme představit soutěž-hru konkurujících si jedinců o potravu, zdroje, prostor a partnera, ve které jde jedincům hlavně o to přežít, rozmnožit se, nikoli o to, jak se mnoho lidí domnívá, přizpůsobit se svému okolí (Zrzavý et al., 2004). V evoluci jedinců nejde „jedincům“[22] ani o to být nejlepší (tj. mít optimální vlastnosti vůči vnějším fyzikálním podmínkám i okolí), ale o to, být lepší než jejich konkurenti (Zrzavý et al., 2004).

Dále je vhodné upozornit na to, že tento text je míněn v myšlenkách populační genetiky a neodarwinismu, která vnímá evoluci jako soutěž alel. Ve skutečnosti se ovšem o žádnou soutěž nejedná a jde jen o to, že ne každá alela má stejnou kvalitu a jedinci, kteří ji vlastní nejsou stejně schopní (Zrzavý et al., 2004).

Hlavními mechanismy evoluce, na kterých spočívají popsané principy a také myšlenka EA, jsou (Kvasnička et al., 2000):

- **reprodukční proces (sexual reproduction)** dává vznik novým jedincům kombinací genetického materiálu dvou rodičů. V konceptu, ve kterém byl algoritmus navrhnut, představuje každý jedinec v populaci diploidního jedince[23], který je schopný pohlavně se rozmnožovat. Pohlavní rozmnožování umožňuje jedincům v přírodě šířit (předávat) své vlastnosti z generace na generaci. Z hlediska evoluce nerozmnožující se jedinci jako by neexistovali, protože své geny dál šířit nemohou. Hlavním předpokladem evoluce je to, že každý jedinec zanechává („produkuje“) jiný počet potomků. „Lepší“ jedinci mimo jiné přežívají déle a „produkují“ více potomků, kteří jsou až na mutace stejní jako jejich rodiče. Důsledkem toho se v populaci začnou vlivem dědičnosti šířit znaky rodičů, kteří mají v populaci více potomků (to jsou ti kvalitnější, lépe přizpůsobení jedinci).

- **přírodní výběr neboli selekce (natural selection)** tento princip vyjadřuje „vztah minulosti s budoucností“. Označuje způsob, jež se uplatňuje při výběru toho, co má tendenci se v průběhu let v populaci zachovávat a co ne, protože ne vše má šanci v populaci přetrvat (Zrzavý et al., 2004). To se na genetické úrovni projevuje

[20] Ačkoli se v případě Darwinovy teorie (ve skutečnosti se jedná o 5 teorií (Mayr, 2009)) a Mendelových zákonů jedná o paradigmata, jsou tyto teorie dodnes považovány vědeckou komunitou spíše za dogmata. A tak k nim budeme v práci přistupovat.

[21] Jindy také dem, mendelovské populace nebo subpopulace (Relichová, 2009).

[22] Jedinci se vědomě nesnaží z hlediska genetiky dosahovat nějakého cíle.

[23] V optimalizačním žargonu se jedinci tvoření jen jedním řetězcem zakódovaných parametrů označují trochu nešťastně jako jedinci haploidní. Jedinci tvořeni dvěma řetězci (viz MGA v 2.5.2) se označují jako jedinci diploidní.

změnou genových frekvencí[24] konkrétních vlastností, alel úspěšných jedinců, které se ukázaly jako užitečné - vyhrály v soutěži alel. Tyto kvalitní vlastnosti se vlivem selekce v populaci rozšiřují, popř. fixují, nebo naopak potlačují. Pro upřesnění ještě dodejme, že selekce je pro neodarwinsty jedinou nenáhodnou složkou evoluce a že o schopnosti jednotlivé alely rozhoduje fenotyp celého jedince, nikoli alela samotná.

- **genetický drift neboli genetický posun (genetic drift)** jedná se o náhodné změny v genových frekvencích, které jsou způsobeny takovými náhodnými událostmi jako je smrt prosperujícího jedince ještě před jeho reprodukčním stádiem nebo náhodnými mutacemi, které jsou zodpovědné za vznik evolučních novinek v populaci (Kvasnička et al., 2000). K těmto samovolným i indukovaným mutacím v přírodě dochází poměrně často a bývají způsobeny nekorektním okopírováním genetické informace během rozmnožování nebo vlivem mutagenů[25] . Podle účinků rozlišujeme mutace na negativní, pozitivní a neutrální. Negativní mutace se projevuje snížením biologické zdatnosti jejího nositele[26] nebo v krajním případě neslučitelností se životem. Pozitivní mutace zvyšuje fitness jedince a neutrální mutace se v rámci fenotypu neprojevuje a jedince tak z hlediska přirozeného výběru nijak zvlášť nezvýhodňuje, ani neznevýhodňuje (Flegr, 2007).

  Efekty náhodného driftu jsou významné hlavně pro malé populace, protože na to, aby jedinci vyčerpali všechny možnosti genů (alel), je v populaci málo genetického materiálu, a proto se nepatrné změny ve výskytu alel budou projevovat více. Čím menší je lokální populace, tím víc je také pravděpodobné, že o osudu mutace, neutrální nebo lehce negativní, rozhodne spíše genetický drift než přírodní výběr (Flegr, 2007). Ten potom zjednodušeně rozhoduje o tom, zda se nějaká vlastnost jedince náhodně zafixuje nebo zmizí. A to může být u malé populace s malou diverzitou genů celkem výrazný zásah do genofondu.

Zjednodušeně řečeno se mechanismem náhodné mutace a nenáhodné selekce daří v přírodě zachovávat v populaci určité vlastnosti organismů, díky nimž se postupně zvětšuje průměrná zdatnost jedinců dané populace vzhledem k okolním podmínkám a konkurujícím jedincům. Adaptace (přizpůsobení se okolním podmínkám) potom není žádný aktivní proces, ale jen důsledek výše zmíněné soutěže.

V evolučním procesu sehrává svou roli i efekt pohlavního výběru. Pod pohlavním výběrem si můžeme opět představit soutěž-hru konkurujících si jedinců. Jedinci v tomto případě „soutěží“ mezi sebou. Reprodukční soutěž se v přírodě projevuje podobně jako dříve popsaný přírodní výběr. Jde v něm o to, že některé vlastnosti jedinců mohou ostatním jedincům naznačovat (indikovat), že daný jedinec je sexuálně nebo zdravotně zdatnější než ostatní jedinci v populaci, a proto může být tento jedinec ostatními preferován při reprodukčním výběru (Flegr, 2007). Ve výsledku jde ale v rámci obou soutěží (přírodní a reprodukční) v populaci o totéž - zachovávat v populaci ty nejkvalitnější vlastnosti.

[24] Platí v případě, že uvažujeme o přírodním výběru na úrovni genomu. Ve skutečnosti ale selekce probíhá na více úrovních (Tkadlec, 2008).

[25] Biologický, fyzikální nebo chemický faktor, který je schopen způsobit mutaci.

[26] Biologická zdatnost jedince se v biologií označuje pojmem fitness nebo vhodnost (viz 2.1).

Je důležité ještě poznamenat, že jedinci v přírodě nedosahují globálních optim (z hlediska přizpůsobení se podmínkám), ale lokálních. K tomu existují minimálně tři důvody. Jedním z nich je nízká schopnost přírody přijímat do nové generace horší jedince, než jsou jedinci stávající. Přestože mohou být tito horší jedinci v určitém směru mnohem úspěšnější než jedinci aktuální, celkově nejsou tito horší jedinci (příliš) schopni vytlačit v rámci reprodukční soutěže ty lepší jedince - a nemohou tak šířit své kvalitní kvalitní alely (Zrzavý et al., 2004). Druhým důvodem je omezený genetický materiál jedinců. A třetím důvodem je to, že organismy reagují na okamžité podmínky prostředí, které se neustále mění a fixace popř. vymizení konkrétních alel (ekvivalent adaptace) je oproti tomu velmi pomalý proces s mnoha postupnými mezikroky. Jedinci se proto nikdy nestačí „rychlým“ změnám dokonale přizpůsobit. Z těchto důvodů dochází v přírodě k uváznutí v lokálních optimech.

4.2.2 Samotný algoritmus ES a jeho varianty

Algoritmus ES probíhá analogicky jako výše popsaný evoluční proces v přírodě. Vystavujeme v něm populaci jedinců evolučnímu mechanismu - selekčnímu tlaku, s cílem vytvořit co nejpřizpůsobenější jedince danému prostředí - optimalizovanému problému. Kvalitu jedince, jeho přizpůsobenost vůči konkrétnímu prostředí, opět vyjadřuje hodnota funkce vhodnosti (fitness).

Celý algoritmus vychází z cyklického opakování operací rekombinace, mutace a selekce z kapitoly 2.4. Připomeňme, že hledané parametry jsou v případě ES stejně jako u ABC zakódovány reálnými čísly, a proto mají geny teoreticky nekonečně mnoho alel[27].

- **mutace** jedná se o náhodné změny hodnot jednoho nebo více parametrů genomu jedince. Náhodná změna parametru se většinou u ES při reálné reprezentaci provádí přičtením náhodně vygenerované hodnoty z předem stanoveného pravděpodobnostního rozdělení (spojitá optimalizace). K mutaci se standardně nejčastěji využívá vícerozměrné normální rozdělení pro problémy bez omezení a geometrické rozdělení pro problémy diskrétní (celá čísla) (Beyer – Schwefel, 2002). Četnost mutací je u ES 100 %. Důležitým parametrem této operace je síla mutace, která je zároveň parametrem pravděpodobnostního rozdělení. V průběhu algoritmu sledujeme z důvodu přizpůsobení síly mutace četnosti pozitivních a součet četností negativních a neutrálních mutací (operace adaptace u pravidla $\frac{1}{5}$).

- **selekce** selekční tlak „hodnotí“ kvalitu jednotlivých členů populace řešení a poté z nich vybírá μ nejkvalitnějších jedinců, kteří přežijí do příští generace. Jedince zvažované k selekci vybíráme z množiny řešení, kterou určuje používaná verze algoritmu (`plus` nebo `čárková` verze viz níže). Selekce podporuje vývoj populace výhradně[28] do nadějných oblastí prostoru přípustných řešení a je jakýmsi protipó-

[27] Teoreticky nekonečno - omezeno výpočetní technikou. U binárního kódování GA měly geny omezený počet alel. Ten byl daný počtem všech možných kombinací bitů v řetězci jednoho genu - 2^k, kde k označuje délku řetězce, který kóduje jeden hledaný parametr.

[28] Nutno poznamenat, že selekce pracuje vždy s množinou řešení, která je k dispozici, a proto nadějné oblasti nemusí nutně znamenat nejnadějnější.

lem mutace a rekombinace, protože snižuje v populaci variabilitu, stejně jako v přírodě (Flegr, 2007). Hlavním předpoklad použití operátoru selekce je to, že z kvalitních rodičů vznikají kvalitní potomci - potomci mají velmi podobné genomy jako jejich rodiče, a jsou v průměru mnohem kvalitnější než potomci, kteří vzniknou z méně kvalitních rodičů. Naopak je zase nutné zachovávat v populaci určitou rozmanitost, abychom zabránili předčasné konvergenci algoritmu k lokálnímu extrému. Selekcí se někdy označuje i proces, kterým vybíráme ρ jedinců do rekombinace.

- **rekombinace** vytváření λ nových potomků v každé generaci. Nový potomci vznikají kombinací alel (jednotlivých parametrů) ρ rodičů. Implementace této operace se různí podle toho, zda během rekombinace vytváříme nové alely anebo zachováváme alely rodičů (rekombinativní vs. diskrétní rekombinace).

$$\mathbf{x}^{t+1} = \begin{cases} \frac{1}{2} \cdot (\mathbf{x}_a^t + \mathbf{x}_b^t) & \text{rekombinativní rekombinace (intermediate)} \\ \mathbf{x}_a^t \text{ nebo } \mathbf{x}_b^t & \text{diskrétní rekombinace (dominant),} \end{cases} \tag{4.1}$$

kde $\mathbf{x}_a^t$ a $\mathbf{x}_b^t$ označuje dva jedince vybrané k reprodukci $(\rho = 2)$ a $\mathbf{x}^{t+1}$ nově vzniklého jedince. U ES většinou dochází k rekombinaci všech alel jedinců, ale je možné použít i jednobodové nebo vícebodové křížení viz 2.4 (to platí hlavně pro diskrétní reprezentaci jedinců).

Je vhodné poznamenat, že všechny výše popsané operace jsou stochastickými procesy. Během výběru ρ řešení (rodičů) k rekombinaci většinou nebývají kvalitnější jedinci vybíráni s větší pravděpodobností a šanci zúčastnit se reprodukčního procesu („přežít“) mají i slabší jedinci stejně vysokou. Pokud se používají, jsou body křížení u rekombinace generovány náhodně, stejně tak jako geny podléhající mutaci.

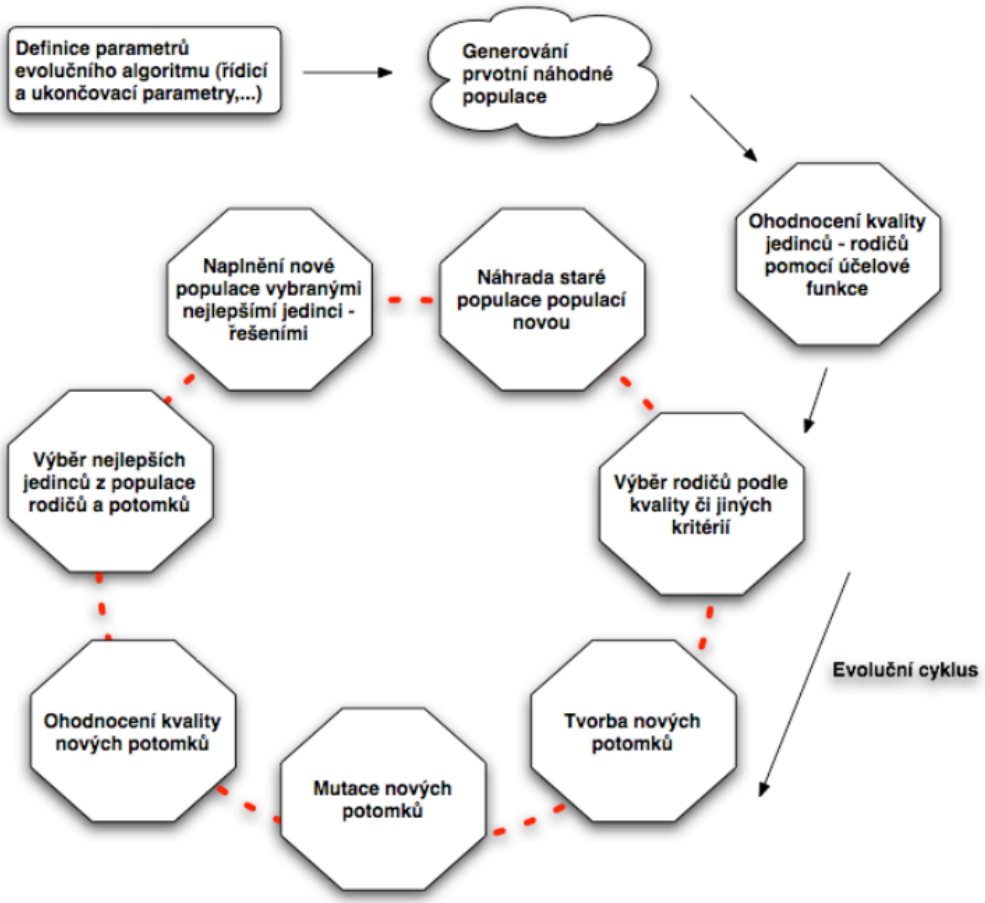
V průběhu algoritmu opět spoléháme na to, že se během generací bude zvětšovat průměrná zdatnost jedinců dané populace vzhledem k optimalizovanému problému.

Průběh samotného algoritmu ES v základní variantě probíhá následovně: na začátku si zvolíme počáteční generaci jedinců, na kterou postupně aplikujeme výše popsané operátory rekombinace, mutace a selekce až do doby, dokud nedojde k ukončení algoritmu. Algoritmus je podobně jako ABC ukončen ve chvíli, kdy dojde k nalezení dostatečně kvalitního řešení, nebo je algoritmus ukončen násilně po uplynutí doby, kterou máme pro jeho výpočet k dispozici, či po určitém počtu iterací[29]. Výstupem algoritmu je nejlepší nalezené řešení v rámci všech iteračních cyklů (generací). Implementační detaily ES budou popsány až v 5.3.1.

Počáteční populaci můžeme volit v prostoru přípustných řešení náhodně nebo s ohledem na primární (a priori) znalosti o optimalizovaném problému. Tyto znalosti mohou být také zakomponovány do jednotlivých operací nebo jim můžeme přizpůsobovat typy používaných operací.

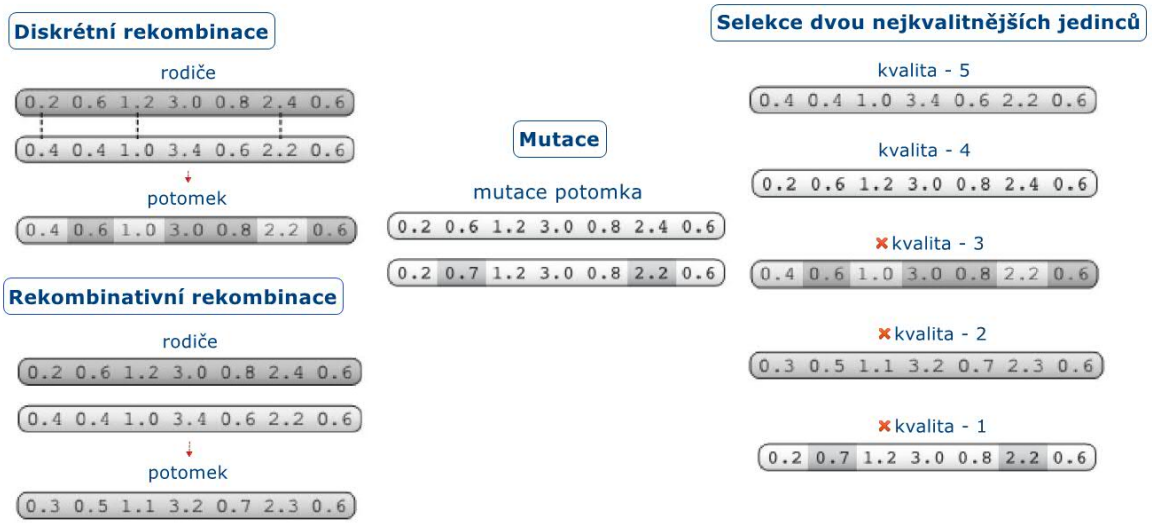
[29] Opět se jedná o řídicí parametry algoritmu.

Schématický náčrt průběhu evolučního cyklu ES je znázorněn na obrázku 4.2 a schémata tří základních operací ES jsou na obrázku 4.3.



Obrázek 4.2: **Základní schéma evolučního cyklu ES**. Toto schéma je platné pro většinu evolučních algoritmů. Skládá se z několika hlavních částí: vytvoření nové populace jedinců, tvorby nových potomků, mutace potomků náhodnými změnami, ohodnocení jedinců a selekce těch nejlepších jedinců populace. Tito vybraní jedinci jsou vstupem dalšího cyklu algoritmu. zdroj: (Zelinka et al., 2008)

U algoritmu ES hrají podstatnou roli znaménka `plus` a `čárka`. Ty označují množinu jedinců, ze kterých se vybírají jedinci do příští generace (operace selekce). Píšeme-li `čárku` (μ, λ), do nové generace vybíráme nejlepší jedince z populace potomků, rodiče z definice umírají (a potom $\mu \leq \lambda$). Symbol `plus` $(\mu + \lambda)$ zase značí, že do nové populace vybíráme μ nejlepších jedinců z populací rodičů i potomků. `Čárková` verze se obecně doporučuje k řešení spojitých problémů, podmínkami neomezených, `plus` verze k řešení úloh diskrétních (Beyer – Schwefel, 2002). Není to ovšem nutným pravidlem (viz 5.4). `Plus` verze obecně rychleji konverguje ke globálnímu extrému než verze `čárková`, ale naopak u ní více hrozí předčasná konvergence k lokálnímu extrému. Rozdíly mezi oběma verzemi se při použití velké populace stírají (Beyer – Schwefel, 2002). V dalším textu budeme použí-



Obrázek 4.3: **Schéma operací rekombinace, mutace a selekce u ES**. U Evolučních strategií jsou jedinci reprezentováni řetězcem reálných čísel o dané délce. zdroj: (Floreano – Mattiussi, 2008), upraveno

vat označení $(\mu \dot{+} \lambda)$, (popř.$(\mu/\rho \dot{+} \lambda)$), které zahrnuje verze obě.

Nejběžnější jsou následující verze algoritmu ES:

- **dvoučlenné (two-membered,** $(1+1)-ES$**)** je první a nejjednodušší verze ES. Používá jen operátor mutace a jednopopulační generace. Do další generace přežívá vždy kvalitnější jedinec, rodič nebo potomek.
- **vícečlenné (multi-membered,** $(\mu \dot{+} \lambda)-ES$**)** již umožňuje volit mezi `plus` a `čárkovou` verzí algoritmu. Oproti předchozí strategii (verzi) zavádí operaci křížení, která urychluje konvergenci algoritmu k výslednému řešení. Rozdíly mezi `čárkovou` a `plusovou` se projevují především u malých populací (viz odstavec výše). Této verzi se někdy přezdívá ES „s elitismem“.
- **rekombinativní (recombinative,** $(\mu/\rho \dot{+} \lambda)-ES$**)** rozšiřuje předchozí verzi o počet rodičů, ze kterých se noví jedinci vytváří. Počet rodičů je daný parametrem $\rho \in [1,\ldots,\mu]$.
- **sebeadaptabilní (self-adaptive,** $(\mu/\rho \dot{+} \lambda)-ES$**)** nově zavádí proces učení. Jelikož se tato práce v praktické části zabývá implementací právě tohoto typu ES, podívejme se na tuto variantu trochu podrobněji.

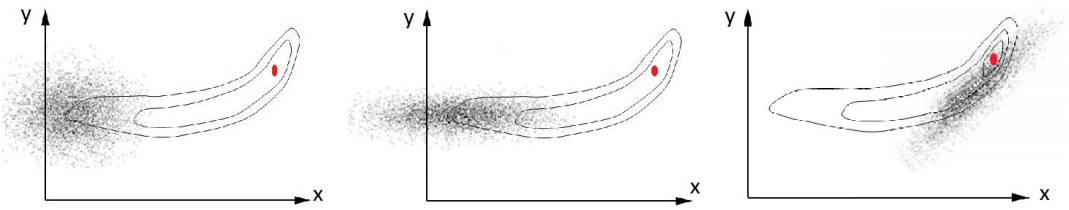
Sebedaptabilní evoluční strategie

Procesem učení přidáváme do základní verze algoritmu operaci adaptace (viz 2.4). Důvodem je to, že při konstantní mutační síle se v pokročilé fázi algoritmu, kdy se přibližujeme k extrému, zastaví vývoj populace – to z toho důvodu, že mutační krok je příliš

dlouhý (mutace příliš silná) a k vylepšení vhodnosti jedinců dochází jen zřídkakdy. Pokud bychom zase od počátku používali extrémně malou mutační sílu, snižovali bychom efektivnost algoritmu a zvyšovali jeho časovou náročnost.

U ES umožňuje operace adaptace (verze $\frac{1}{5}$) reagovat na „tvar“ kriteriální funkce a využívá k tomu informace o úspěších/neúspěších mutací v rámci jednotlivých návrhových parametrů v minulých generacích. Na základě těchto informací dochází k úpravě (tj. adaptaci) velikostí mutací jednotlivých parametrů, které se budou používat k mutaci v příštích generacích. Myšlenka je založena na předpokladech, že vyšší četnost pozitivních mutací u ES objeví, pokud se generace přibližuje ke globálnímu optimu, a nízká četnost pozitivních mutací se v algoritmu vyskytne v případech, pokud jedinci uvázli v lokálním extrému a v jejich okolí se žádná lepší řešení nenachází. U této varianty sebeadaptace, která bude použita v praktické části, se v případě, že je četnost pozitivních mutací vysoká, zvyšuje velikost mutace, protože se pomalu přibližujeme k extrému a zvýšení velikosti mutace jedincům zabraňuje v něm uváznout. Pokud je naopak četnost pozitivních mutací nízká, snížíme velikost mutačního kroku, čímž jedincům umožníme dostat se do slibnějších oblastí prostoru přípustných řešení.

Operace adaptace opět rozlišujeme na více druhů podle toho, zda upravujeme velikosti mutace pro jednotlivé hledané parametry uniformě (isotropic adaptation) nebo pro každý parametr zvlášť (anisotropic adaptation, nonisotropic), tedy s větší či menší odchylkou, popřípadě, zda využíváme kovariancí (korelací) mezi jednotlivými hledanými parametry. Poslední zmiňovaná metoda se hodí především pro úlohy, kde je mezi hledanými parametry lineární vztah. Efekty používání izotropní, anizotropní mutace a mutace s korelací jsou znázorněny na obrázku 4.4.



(a) isotropní mutace (b) anisotropní mutace (c) korelovaná mutace

Obrázek 4.4: **Schémata typů mutací u ES** Na obrázku je zobrazeno, jaký efekt mají jednotlivé typy mutací. Plnými čarami jsou zobrazeny vrstevnice kriteriální funkce a tečkami jsou zobrazeni nově vygenerovaní zmutovaní jedinci podle a) isotropní mutace, b) anisotropní mutace, c) korelované mutace při normálním rozdělení. Optimum je v obrázcích zobrazeno červenou tečkou a na osách jsou hodnoty dvou hledaných parametrů. Je dobré si všimnout, že u anisotropní mutace tvoří vrstevnice pravděpodobnosti umístění zmutovaného jedince do prostoru řešení (ve 2D) elipsu, ne kruh, jak je to tomu u mutace isotropní. A korelovaná verze způsobuje natočení elipsy – její osy už nejsou orientovány v rámci souřadnicových os, ale natáčí hlavní osu elipsy k optimu. zdroj: (Beyer – Schwefel, 2002), upraveno

Druhý způsob, který se pro adaptaci řídicích parametrů optimalizačních algoritmů po-

užívá (kromě výše popsané metody a meta verzí z 2.3.2) je zakomponování adaptovaného parametru do optimalizovaných parametrů. U ES to konkrétně znamená, že i na sílu mutace budeme během iterací aplikovat operace rekombinace a mutace. Podoby operací mutace i rekombinace mohou, ale nemusí, být ekvivalentní s operacemi, které používáme k mutaci a rekombinaci hledaných parametrů. Jediný rozdíl aplikace těchto operací na řídicí parametry je v tom, že blíží-li se populace k optimu, snižuje se vliv mutace. Více informací o jednotlivých přístupech se lze dočíst v Beyer – Schwefel (2002). Podrobný přehled zvláště sebeadaptabilních verzí ES lze zase najít v Bäck et al. (2013).

Propojme nyní předchozí dvě podkapitoly pokusme se kriticky zhodnotit rozdíly mezi biologickým modelem ES a jeho předlohou.

4.2.3 Biologie vs. „model“ ES

U předchozího algoritmu jsme porovnání postavili na vlastnostech biologické předlohy ABC, které byly v algoritmu využity/nevyužity. U evolučních strategií zvolme trochu jiný postup porovnání a jemně porušme počáteční předsevzetí nesoustředit se na neodpovídající si části algoritmu s přírodou. V tomto porovnání naopak z těchto nepřesností vyjdeme a zkusíme si na nich ukázat (odhadnout), proč se od přírodních procesů odlišují.

Už na první pohled bychom mohli říct, že zásadní rozdíl mezi biologickou evolucí a umělou evolucí ES je v tom, že biologická evoluce nemá žádný cíl, je pouze „oportunistická“ (Flegr, 2007), neschopná plánovat dopředu, kdežto evoluce umělá je oproti tomu optimalizační proces se zřejmým cílem - najít řešení optimalizovaného problému. To je sice pravda, ale pokud bychom evoluci chápali tak, jak jsme si ji výše popsali, je optimalizační proces důsledkem evolučního procesu, i přesto, že evoluce, ani jedinci žádný cíl nemají. Okolnosti, že jedinci v přírodě nenachází globální, ale lokální optima, v algoritmech pomůžeme zafixováním okolních podmínek[30] a stanovením jediného selekčního tlaku (jednokriteriální optimalizace), který na jedince působí[31]. Fenotyp jedinců potom udává výsledek interakce genotypu jedinců s prostředím. Za takovýchto podmínek a bez přímé konkurence jedinců[32] očekáváme, že se v populacích řešení budou objevovat i globální řešení řešené úlohy.

Věnujme se teď jednotlivým přírodním jevům, které s v algoritmu vyskytují, a na nich si ukazujme nepřesnosti algoritmu ES vůči přírodě.

- Samotná rekombinace (biologický význam 2.1) v přírodě není primárním původcem nových vlastností (nových genů), ale zdrojem nových kombinací vlastností rodičů (genotypu), které až ve fenotypu dávají vznik novým vlastnostem. Každý jedinec[33]

[30] Toto neplatí pro dynamické problémy, ale i oni se dají řešit evolučními algoritmy. My to zde pro jednoduchost opomíjejme.

[31] Tento zjednodušený model evoluce a přírodních procesů svádí k otázkám, zda by se tomu tak dělo i v přírodě. 1. Zda by se vlastnosti jedinců v přírodě beze změn fyzikálních podmínek a ve velmi dlouhém časovém horizontu ustálily či by to konkurence jedinců nedovolila. 2. Zda by vlastnosti jedinců dosahovaly globálních optim anebo jsou jedinci a jejich vlastnosti a projevy natolik komplexní, že by to nebylo možné.

[32] Jedinci se nijak neomezují a je jich pořád konstantní počet .

[33] Uvažujeme standardního diploidního jedince.

má dvě alely stejného genu, které se ve fenotypu mohou projevit různě a stejné geny mohou mít v kombinaci s jinými geny odlišný vliv na fitness jedince. Diskrétní verze rekombinace tuhle myšlenku respektuje (pouze kopíruje alely rodičů), ale rekombinativní verze ne, ta přímo vytváří nové alely. Pokud si ale uvědomíme, že genotypy jedinců u ES odpovídají jejich fenotypu, tak potom i rekombinativní verze rekombinace reflektuje přírodu. Rekombinace je tedy v přírodě i algoritmu zdrojem nových vlastností a my si jen musíme být vědomi toho, že rekombinací u ES míníme fenotypický odraz jedince po pohlavním rozmnožování. Z důvodu vzniku nových alel je i rekombinativní verze rekombinace v praxi pro spojitou optimalizaci preferovaná.Vliv interakce genů se u ES projevuje zase v tom, že o kvalitě jednotlivých genů (i jedince) rozhoduje až kombinace všech genů jedince.

- K mutacím v přírodě sice dochází relativně často, ale ne každá mutace se projeví ve fenotypu jedince. Algoritmus ES zase na mutace spoléhá v každém iteračním cyklu (četnost mutací je u ES 100 %) a mutace se obvykle vztahuje ke každému hledanému parametru. To je trochu v rozporu s tím, že většina genových mutací má v přírodě pro organismus škodlivý, ne-li letální účinek, a je velmi nepravděpodobné, že by nějakou „supermutací“ (vícečetnou mutací) vznikl prosperující a životaschopný organismus. Autoři ES si jsou toho ale vědomi a mají pro vysokou frekvenci mutací svůj důvod. U ES obdobně jako v přírodě počítáme s tím, že k pozitivním mutací bude docházet jen zřídka, ale sledování frekvencí pozitivních a negativních mutací nám dává vhled o tom, které oblasti prostoru přípustných řešení jsou slibné a které ne. To je důvod, proč je 100% četnost mutací u ES vítaná. Každá generace má navíc konstantní počet jedinců, takže k žádnému vymírání druhu nedochází a byla by škoda této možnosti vhledu do optimalizovaného problému nevyužít.

  Bavíme-li se o letálních účincích mutací, tak k tomu můžeme jen poznamenat, že v algoritmu také dochází v určitém smyslu k vymírání jedinců. V žádné fázi algoritmu se neobjeví řešení, které nenáleží do množiny přípustných řešení (Ω) daného problému. Taková řešení jsou v ES automaticky nahrazována novými jedinci.

- Rozdíl mezi adaptivním procesem ES a biologie je v tom, že příroda nemá žádný mechanismus pro porovnávání toho, co by bylo pro organismy lepší nebo horší, a v tom, že jedinci nemají k dispozici žádnou dlouhodobou paměť, ve které by si zaznamenávali, které vlastnosti jsou v rámci jejich okolí efektivnější a které ne.

  A přestože pro organismy a přírodu existuje jen tady a teď, k adaptacím v přírodě dochází. A to tak, jak jsme si popsali v předchozí kapitole. Adaptace jsou v evoluci důsledkem změn selekčních podmínek, v ES jsou zase nutné pro konvergenci ke globálnímu extrému (lépe řečeno velmi vítané).

  Kvůli absenci dlouhodobé paměti nemohou jedinci v přírodě svůj vývoj nějakým směrem směřovat nebo cílit[34], ale to dělá nepřímo přírodní výběr. A i uživatel v rámci adaptace mutačního parametru.

Teď se alespoň zhruba vraťme ke způsobu porovnání, jež jsme si stanovili u algoritmu ABC. Za prvé se zaměřme na komponenty evolučního procesu, které jsou pro fungování

[34] Otázkou zůstává, jestli by toho byli vůbec schopni.

algoritmu nezbytné a za druhé naznačme, proč by nemělo smysl hledat u ES přírodní mechanismy i v dílčích částech algoritmu, kde příroda předlohou nestála.

Vlastnosti, kterými se autoři při návrhu algoritmu ES inspirovali jsou především mechanismy, které zaručující vznik unikátních řešení (jedinců) v populaci, protože o to nám jde při řešení optimalizačních úloh především - víceméně náhodným způsobem generovat různá řešení daného problému[35]. Integrací se selekčním mechanismem, který v populaci naopak zapříčiňuje restrikci této rozmanitosti jedinců na omezenou množinu těch nejlepších jedinců populace, a s reprodukčním mechanismem (křížením) se v populaci začnou vyskytovat kvalitnější a kvalitnější řešení (jedinci). Výskyt kvalitnějších řešení, je téměř v souladu s teorií schémat z 2.5.2 (reálné kódování) - v populaci se vytváří něco jako stavební bloky, opakující se sekvence chromozomového řetězce (tzn. stejné či velmi podobné hodnoty hledaných parametrů), jejichž vhodnou kombinací se v populaci v průběhu generací vytváří kvalitnější a kvalitnější řešení problému[36].

Mezi vlastnosti evolučního procesu, jimiž se autor (autoři) naopak neinspiroval, patří například genetický draft. Genetický draft zjednodušeně vypovídá o fixaci nebo potlačení mutace, která se na chromosomu nachází v těsné blízkosti jiné selekčně významné mutace - šíření této významné mutace zapříčiňuje souběžné šíření sousedící mutace a naopak. Tento jev nebyl v algoritmech využit, protože (naší) snahou je optimalizovat všechny hledané parametry nezávisle na ostatních. Použití nějaké obdoby genetického draftu by sice mohlo být přínosné, ale zkoumat na chromosomech všechny různé kombinace parametrů a jejich kvality by bylo časově náročné a neefektivní. V malé populaci by také mohlo dojít ke krajnímu případu - konzervaci genů. Ta v přírodě sice také funguje, neboť např. 99,9 % lidského genomu je pro všechny lidi společný (Henderson, 2014), ale ta by u optimalizace přínosná v konečném důsledku nebyla. Navíc metaheuristické algoritmy (a modely) by měli být už z principu jednoduché a univerzální.

Přesto existují i verze ES, kde se konzervace genů využívá (ta ale není důsledkem genetického draftu). Konzervace genů se provádí následovně: při vzniku potomka (rekombinací) se kopírují pouze odpovídající si a stejné části genomů dvou rodičů a zbytek genomu jedince se náhodně doplní (dominantní rekombinace) - používá se to ale hlavně pro binární reprezentaci jedinců a k řešení diskrétních problémů (Beyer – Schwefel, 2002). Nicméně k řešení spojitých problémů se konzervace genů s rekombinativním typem rekombinace již také použila Beyer – Schwefel (2002).

A nakonec důvod, proč by nemělo smysl[37] zkoumat úpravy algoritmu ES a hledat pro ně biologický podklad. Hlavním důvodem je fakt, že „genetika“ (vědní obor i procesy) je natolik komplexní a složitá, že uměle hledat k úpravám algoritmu obrazy v přírodě při zachování různé míry věrohodnosti a abstrakce by bylo velmi „alibistické“, protože vysvětlení některých úprav by se vyloženě dalo vztáhnout téměř na cokoli. Takovéto formě

[35] I tuto náhodnost můžeme doplňovat konkrétními znalostmi o řešené úloze

[36] To je trochu zjednodušeně řečeno. V Evolučních strategiích si jedinci nevybírají partnera ke křížení podle jeho fitness.

[37] Alespoň podle autorky této bakalářské práce.

alibismu se raději vyhneme.

Pokud jde o věrohodnost přírodní inspirace algoritmu, tak z předešlého textu vyplývá, že ES svou biologickou předlohu poměrně věrně reprodukují. Avšak o něco více volněji, než jak je tomu u algoritmu ze stejné třídy, GA. Většina operací ES je vybudována na určitém biologickém základě, ale uměle upravena.

Teď si možná budeme trochu protiřečit s výše popsaným požadavkem metaheuristických algoritmů na jednoduchost a komplexností genetiky, ale další okolností, kterou by možná nebylo úplně od věci prozkoumat, je fakt, že věda a lidské povědomí o dějích v přírodě se neustále prohlubují - konkrétně poznatky molekulární biologie přinesly v posledních letech objasnění mnoha jevů. Otázkou by potom bylo, zda se ve vědecké sféře neobjevilo od 60. let 20. století (publikace algoritmu ES) něco „nového“, co by naopak mělo smysl do algoritmu zakomponovat[38]. A to nejlépe tak, aby to příliš nenarušilo jednoduchost algoritmu. Odpověď na tuto otázku už ale přenechme raději jiným.

Shrňme si opět kapitolu srovnání. I přes výše zmíněné nepřesnosti přírodních procesů bychom mohli říci, že inspirace evolucí je u optimalizačních algoritmů zcela na místě, a to díky tomu, že námi vytvořený model evoluce je optimalizačním procesem. Darwninova evoluce globálních optim sice v přírodě nedosahuje, ale nástroj pro jejich hledání to minimálně je[39]. Zvláště potom v evoluci umělé, kde přímá konkurence nefiguruje. Nepřesnosti algoritmu vůči přírodě mají opět svůj účel, o čemž opět svědčí i to, že tyto algoritmy při optimalizaci opravdu fungují a jsou odborníky z mnoha oblastí hojně využívány.

4.3 Vybrané algoritmy a registrace obrazů

Tato kapitola předkládá přehled prací, které využily k registraci obrazů výše rozebírané algoritmy ABC a ES nebo alespoň využily registraci obrazů pro validaci jejich algoritmu.

Banharnsakun et al. (2011) využil rigidní registraci k otestování funkčnosti vylepšené verze ABC. Rozdíl oproti základní ABC spočívá ve třech bodech: rekrutky se vždy pohybují směrem k nejlepší známé průzkumnici, zavedení lineárního zmenšování prostoru přípustných řešení pro skautky a úprava původně navržené funkce vhodnosti speciální pro ABC. Optimalizovanou funkcí byla míra vzájemné informace[40].

García-Torres et al. (2014) porovnával na rigidní registraci 3D modelů pořízených laserovým skenováním výkony 3 populárních populačních MH s evolučními metodami běžně používanými pro IR. Algoritmus ABC byl jedním z testovaných MH. Jako podobnostní míra byla použita metoda nejmenších čtverců.

[38] Tato možnost nebyla při porovnání ABC zmíněna z důvodu, že ABC vznikl v roce 2005, a tedy pravděpodobnost nového vhledu do způsobu získávání potravy včel je nižší.

[39] A kdo ví, třeba jich někdy dosáhne – nebo už dosáhla?

[40] Další podobnostní metrika, která se při registraci obrazů používá.

Winter et al. (2008) publikovali algoritmus CMA-ES, který byl aplikován na registraci multimodálních obrazů pořízených předoperačně (počítačová tomografie, CT) a během operace páteře (ultrazvuk). CMA-ES optimalizoval kriteriální funkci odvozenou ze zobrazovacích vlastností ultrazvuku (zjednodušeně maximalizace odstínu šedi) na hledání 6 parametrů rigidní transformace (3D). Díky kvalitním testovacím experimentům autoři algoritmus navrhli jako nástroj i pro běžnou registraci medicínských obrazů.

Kapitola 5

Praktická část

Cílem praktické části této bakalářské práce bylo naimplementovat dva metaheuristické přírodou inspirované algoritmy a aplikovat je na problém registrace obrazů mozku z magnetické rezonance. Implementovanými algoritmy jsou ABC a ES z předchozí kapitoly. Obě metody byly implementovány v prostředí MATLAB a jejich zdrojové kódy jsou součástí přiloženého DVD.

Registrační procedura odpovídá schématu 3.2. Samotná registrace probíhá na dvourozměrných obrazech mozku z magnetické rezonance – jedná se o jeden řez celkového třírozměrného obrazu pořízeného magnetickou rezonancí. Ještě konkrétněji se bude jednat o obrazy stejného subjektu pořízené stejným přístrojem (intrasubject, intramodality registration).

5.1 Uměle rozregistrovaná sada obrazů

Uměle rozregistrovaná sada obrazů mozku byla vytvořena geometrickými transformacemi původního obrazu (obraz `original`) s 1, 3, 4 nebo 5 parametry afinní transformace $(d_x, d_y, \phi, s_x, s_y)$ z 3.1.2. Konkrétní kombinace parametrů i jejich hodnoty lze najít v tabulce 6.1.

Do registračního procesu je vždy jako referenční obraz I_M volen obraz původní a jako plovoucí obraz I_S obraz uměle transformovaný.

5.2 Výběr účelové funkce

V oblasti registrace se používají rozličné podobnostní míry, avšak ne všechny jsou vhodné pro řešení všech typů registrací. Dle povahy testovaných dat byla do praktické části vybrána metoda nejmenších čtverců (3.10). Tato metoda byla zvolena především proto, protože je vhodná k registraci monomodálních obrazů, časově méně náročná než jiné metody a běžně se k registraci tohoto typu používá (Hill et al., 2001; Mani et al., 2013).

5.3 Implementace algoritmů

Před samotným popisem implementace obou algoritmů si uveďme ještě pár technických detailů, které jsou nezbytné pro práci se skripty a pochopení následujícího textu. V algoritmech je využívána funkce `affine2d` (z prostředí MATLAB). Je důležité poznamenat, že funkce `affine2d` pracuje s jinou maticí translace (3.4), než jak jsme si ji popsali v 3.1.2. Matice transformace této funkce vypadá takto:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d_x & d_y & 1 \end{bmatrix}, \tag{5.1}$$

kde d_x a d_y opět značí posun ve smyslu osy x nebo y. Na postupu registrace to nic nemění, jen na to musí být brán zřetel. Tato funkce dále umožňuje používat všechny tři zmíněné druhy interpolací. Při registraci je použita `bilineární interpolace`, protože je relativně rychlá, dostatečně kvalitní a běžně se k registraci obrazů používá.

Aby se nám otočené obrazy zobrazovaly do původní mřížky obrazu, je nutné před každou transformací obrazu posunout střed obrazu do bodu, který je vnímán jako střed rotace obrazu (levého horního rohu [0,0,1]). Výsledná transformační matice T se potom z původní matice T' vypočítá jako

$$T = P^{-1} \cdot T' \cdot P, \tag{5.2}$$

kde maticí P provádíme onen posun středu obrazu do středu rotace a maticí P^{-1} provádíme zpětný převod už transformovaného obrazu do původního středu obrazu. Tyto posuny jsou součástí funkce `Aff_transformace5`.

Oba algoritmy byly implementovány podle schémat z kapitol 4.1.2 a 4.2.2. Společný základ obou algoritmů tvoří následující body:

- V obou případech jsou jedinci reprezentováni vektorem reálných čísel o 7 parametrech, které odpovídají parametrům afinní transformace $\mathbf{x} = (d_x, d_y, \phi, s_x, s_y, h_x, h_y)$. Počet hledaných parametrů z celého vektoru (genomu) závisí na volbě typu transformační funkce T (viz 3.1.3), jejich počet je ve funkcích nastavitelný pomocí proměnné `REGTYPE`. Vyhotovené algoritmy lze použít pro `REGTYPE`$\in \{3,4,5,6,7\}$, kde `REGTYPE`$= 7$ odpovídá obecné afinní transformaci ve 2D.

- Jednotlivé parametry byly omezeny (tzv. box boundaries, (2.1)) minimální a maximální hodnotou na intervaly $d_x, d_y \in [-50, 50]$, $\phi \in [-\pi, \pi]$, $s_x, s_y \in [\frac{3}{4}, 1.25]$ a $h_x, h_y \in [-0.4, 0.4]$. Tato omezení lze upravovat v proměnných `dolni_meze` a `horni_meze` příp. `Lb, Ub`.

- Ani u jednoho algoritmu nefiguruje žádné mapování genotypu na fenotyp. Reálné hodnoty hledaných parametrů vektoru $\mathbf{x}$ přímo odpovídají fenotypu genů a hodnotám transformačních parametrů.

A teď už k samotné implementaci - několik postupů je u obou algoritmů ekvivalentních, proto si je zavedeme už teď a při popisu průběhu algoritmů se na ně budeme odkazovat. Standardní algoritmus ABC generuje počáteční populaci v prostotu přípustných řešení zcela náhodně – s rovnoměrným rozdělením, což není nejefektivnější varianta. Jedním z řešení, který se v praxi využívá, je provést před generováním počáteční populace hrubé zalícování obrazů nasazením na sebe těžišť obrazu (vzhledem k intenzitám v obraze) a od něj generování počáteční populace odvíjet. Tento postup byl také zvolen při implementaci algoritmů, protože je v souladu s mechanismem ES generovat počáteční populaci podle normálního rozdělení. Pokud střední hodnotu `means2` vícerozměrného normálního rozdělení nastavíme na hodnoty transformačních parametrů, které způsobují nasazení na sebe těžiště obou obrazů, je to hned dvojí výhra, protože obrazy na sebe budou od počátku zhruba naléhat. Stejný postup předregistrace a použití normálního rozdělení pro generování počáteční populace byly zvoleny i v implementaci ABC. Souřadnice těžiště obrazu vzhledem k intenzitám se počítají podle vzorce

$$T = \frac{\sum_{i=1}^{N} i \cdot O(i)}{\sum_{i=1}^{N} O(i)}, \tag{5.3}$$

kde N je počet pixelů obrazu, $O(i)$ hodnota intenzity pixelu v souřadnici i. Výpočet těžiště provádí funkce `teziste2d`.

Každý jedinec je určen vektorem parametrů afinní transformace $(d_x, d_y, \phi, s_x, s_y, h_x, h_y)$. Při výpočtu kvality jedinců převedeme jejich genetickou informaci (u ABC se toto označení nepoužívá) na transformační matici 5.2 (funkce `parametry_do_matice`) a pomocí této matice zobrazíme pixely transformovaného obrazu do prostoru druhého obrazu, neznámé hodnoty intenzit dopočítáme bilineární interpolací a vypočítáme míru podobnosti mezi odpovídajícími si body obou obrazů. Čím jsou hodnoty `MSE` nižší, tím je zaregistrování kvalitnější a jedinci zdatnější. `Fitness` jedinců odpovídá jejich hodnotě kriteriální funkce. Na tomto místě by také bylo vhodné dodat, že algoritmy počítají inverzní transformaci k té, kterou my hledáme, protože se v nich transformuje referenční obraz na obraz plovoucí. To z důvodů, jež byly popsány v 3.4 a také kvůli kritériu porovnávání kvality registrace (více 5.4). Výstupem obou algoritmů jsou vždy optimalizované transformační parametry i hledané transformační parametry, které transformují plovoucí obraz na obraz referenční a které jsou extrahovány z matice transformace (funkce `parameter_extraction`).

V rámci algoritmu také kontrolujeme, zda nedošlo k vytvoření jedince, který by nebyl přípustný (mimo box boundaries). Takové jedince v algoritmu nepřipouštíme a pokud se podaří takového jedince vygenerovat, je nahrazen novým jedincem.

Oba algoritmy jsou ukončeny, pokud nastane jedna ze tří možností. Všechny možnosti jsou zakomponovány do funkce `ukoncujici_podminky`. Tři možnosti ukončení jsou: u obou algoritmů je omezena maximální hodnota povolených generací (v obou případech parametr `MAX_GEN`, pokud je počítaná generace vyšší, algoritmus se ukončí. Dále byla v algoritmech nastavena hranice kvality. Pokud je `MSE` jedinců nižší než parametr `HRANICE_KVALITY` (minimalizujeme), algoritmus se ukončí, protože již našel dostatečně

kvalitní řešení. A třetí možností, jak může algoritmus skončit, je pokud v 10 iteracích po sobě nedochází ke zlepšení – nalezení kvalitnějšího řešení. V prvním a třetím případě jsou algoritmy ukončeny násilně.

Oproti předchozí kapitole bylo pro lepší pochopení průběhu algoritmů zaměněno jejich pořadí.

5.3.1 Implementace ES

Na poli ES existuje mnoho sebeadaptabilních verzí. Do praktické části této práce byly zvoleny 2 z nich. Prvním z nich je `pravidlo` $\frac{1}{5}$. Toto pravidlo je nejjednodušší verze ES k adaptaci parametru σ, které funguje deterministicky a k úpravám síly mutace zúročuje globální znalosti o provedených mutacích (více později). Pokročilejší metody, jako je druhá verze implementované sebeadaptabilní varianty ES, $(\mu \dot{,} \lambda) - SA - ES$, zúročují k adaptaci parametrů pouze lokální znalosti – znalosti jediného jedince. Ve variantě $(\mu \dot{,} \lambda) - SA - ES$ má každý jedinec své vlastní řídicí parametry, které jsou zahrnuty v jeho genomu a které současně s hledanými parametry podléhají rekombinacím a mutacím. V implementované variantě (anizotropní mutace) se genom každého jedince prodlouží na dvojnásobnou délku, protože každý původní gen jedinců (hledaný parametr) má svou vlastní sílu mutace σ, kterou je třeba měnit. Zařazení parametrů do genomu jedinců umožňuje selekci řídicích parametrů společně s hledanými parametry. Tyto dvě varianty ES byly zvoleny kvůli porovnání globálního a lokálního přístupu úpravy síly mutace a s ohledem na porovnatelnost s dalším implementovaným algoritmem, ABC.

Řídicí parametry

- μ - populace rodičů je tvořena 8 jedinci a je nastavitelná změnou parametru `NY`.
- λ - počet potomků produkovaných populací rodičů je v každém iteračním cyklu 40. Tento parametr byl nastaven tak, aby poměr `NY/LAMBDA` odpovídal doporučenému rozsahu od $\frac{1}{2}$ do $\frac{1}{7}$ z (Beyer, 2007). Empirickým testováním se osvědčila hodnota přibližně $\frac{1}{4}$/ Počet λ byl také volen s ohledem na „přiměřenou“ dobu běhu algoritmu. Zvýšení počtu jedinců v populaci zlepšuje prohledávací schopnost algoritmu (genofond je rozmanitější), ale na úkor jeho rychlosti. V algoritmu ES tento parametr vystupuje jako proměnná `LAMBDA`.
- ρ - Tento parametr specifikuje v algoritmu proměnná `RO` a stanovuje, z kolika jedinců (rodičů) se budou noví jedinci křížit. Parametr je primárně nastaven na hodnotu 2, protože tato hodnota je pro anizotropní mutace doporučována.
- σ - síla mutace má zásadní vliv na výkon algoritmu (Beyer – Schwefel, 2002). V rámci algoritmů se používá anizotropní mutace, a tedy každý hledaný parametr mutuje s jinou silou. Síly mutací byly nastaveny tak, aby vyhovovaly stanoveným omezením parametrů. Pro použití isotropní mutaci lze nastavit všechny mutační síly stejně – změnou parametru `sigma`.

- `MAX_GEN` - maximální počet iterací algoritmu byl nastaven na hodnotu 100. Hodnota byla zvolena s ohledem na to, aby čas výpočtu na běžném počítači nepřekračoval 2 minuty.

Doplňující parametry

- c - řídicí parametr c, slouží k adaptaci velikosti směrodatné odchylky pro pravidlo $\frac{1}{5}$. Jeho hodnota byla nastavena na doporučenou hodnotu 0.87 podle Beyer – Schwefel (2002). Tento parametr lze v algoritmu nastavit změnou proměnné `c`.
- τ - tento parametr je učícím parametrem $(\mu \dot{+} \lambda) - SA - ES$ verze. Používaná hodnota bylo podle (Beyer – Schwefel, 2002) nastavena na $\frac{1}{\sqrt{DIM}}$, ale lze změnit parametrem `tau`.
- τ' - tento řídicí parametr je také učícím parametrem té samé verze algoritmu. Jeho hodnota byla podle stejného zdroje nastavena na $\frac{1}{\sqrt{2 \cdot \sqrt{DIM}}}$ a lze jej změnit `tau_cara`.

Průběh algoritmu a vyhodnocení

– **krok 1: inicializace** Na začátku algoritmu se inicializuje počáteční populace řešení. Ta se generuje v prostoru přípustných řešení podle vzorce

$$\mathbf{x} = \mu + \sigma \cdot \mathbf{N}(\mathbf{0}, \mathbf{E}), \tag{5.4}$$

kde μ představuje střední hodnotu mnohorozměrného[1] normálního rozdělení $\mathbf{N}(\mathbf{0}, \mathbf{E})$ se směrodatnou odchylkou σ. Střední hodnota představuje hodnoty transformačních parametrů vyjadřující posun těžiště obrazu na těžiště druhého obrazu, tak jak bylo popsáno výše. Vektor σ je shodný s mutačními silami. Po vygenerování počáteční populace se určí kvalita každého jedince, která odpovídá tomu, jak dobře obrazy lícují (`MSE`).

– **krok 2: křížení (rekombinace)** V operaci rekombinace se vytvoří nová populace λ potomků. V ES se jedinci párují nezávisle na jejich zdatnosti[2] a jejich alely (hodnoty hledaných parametrů) se rekombinují třemi možnými způsoby. Dva z nich již byly popsány v 4.2.2, ten třetí se od rekombinativní rekombinace liší v tom, že jedinci se neprůměrují se stejnou vahou, ale s ohledem na jejich kvalitu. Typ rekombinace lze volit proměnnou `TYP_REK`. U $(\mu \dot{+} \lambda) - SA - ES$ varianty dochází k současné rekombinaci řídicích parametrů, které jsou obsaženy v genomu jedince. Typ rekombinace vždy souhlasí s používaným typem rekombinace hledaných parametrů. V algoritmu jsou upřednostňovány rekombinativní rekombinace. Po rekombinaci vypočítáme kvalitu každého jedince.

– **krok 4: mutace** Po vytvoření nové populace potomků vystavujeme všechny jedince mutacím. Sílu mutace každého parametru určuje parametr σ, který je buď zakomponovaný v jeho genomu ($(\mu \dot{+} \lambda) - SA - ES$ verze) nebo je pro všechny jedince

[1] Jeho rozměr je dám počtem hledaných parametrů, parametr `DIM`.
[2] V tomto spočívá rozdíl oproti GA. Konvergenci zajišťuje až selekce.

stejně velký. U $(\mu \dot{+} \lambda) - SA - ES$ verze dochází před samotnou mutací hledaných parametrů k mutaci síly mutace podle vzorce

$$\sigma_{i+1} = \sigma_i \cdot e^{\tau' \cdot \mathrm{N}(\mathbf{0},\mathbf{E}) + \tau \cdot \mathrm{N}(\mathbf{0},\mathbf{1})}, \tag{5.5}$$

kde σ_i a σ_{i+1} představují původní a zmutované síly mutací a τ a τ' učící parametry individuálního a globálního charakteru. Nově zmutované síly mutací jsou použity k mutaci hledaných parametrů. Mutace hledaných parametrů jsou realizována normálním rozdělením podle

$$\mathbf{x_{i+1}} = \mathbf{x}_i + \sigma_{i+1} \cdot \mathrm{N}(\mathbf{0},\mathbf{1}), \tag{5.6}$$

kde $\mathbf{x_{i+1}}$ a $\mathbf{x_i}$ představují zmutovaného jedince a nezmutovaného jedince. Pokud používáme verzi s pravidlem $(\mu \dot{+} \lambda) - SA - ES$ mutují se současně i mutační síly v genomech jedinců se směrodatnou odchylkou, která je v algoritmech označena `sigmaG`. Pokud naopak používáme verzi $\frac{1}{5}$, mutujeme řídicí parametry až po samostatných mutacích všech jedinců. K mutacím (adaptaci) síly mutací této verze je potřebné zjistit, jak „vypadá“ okolní prohledávací prostor, proto zaznamenáváme úspěšnosti mutací (zda došlo nebo nedošlo ke zvýšení zdatnosti jedinců). Pokud byla mutace úspěšná, zvýší se hodnota proměnné `success`, v opačném případě se o 1 zvýší proměnná `failure`.

V obou verzích algoritmu jsou původní jedinci nahrazováni zmutovaným jedincem pouze v případě, pokud se jejich kvalita po mutaci zvýšila.

– **krok 5: mutace parametrů (adaptace)** Dalším krokem ($\frac{1}{5}$ verze) je adaptace parametru σ podle pravidla tzv. $\frac{1}{5}$. Toto pravidlo říká, že poměr úspěšných mutací (úspěch odpovídá tomu, když je vhodnost zmutovaného jedince vyšší než vhodnost jedince nezmutovaného) ke všech provedeným mutacím by měl ideálně být $\frac{1}{5}$ (Beyer – Schwefel, 2002)[3]. Pokud tomu tak není a poměr úspěšných mutací je větší než $\frac{1}{5}$, dojde ke zvýšení intenzity mutace, protože prohledávání je moc lokální a algoritmus by mohl uváznout v lokálním extrému, pokud je poměr úspěchů naopak menší než $\frac{1}{5}$, dojde ke snížení intenzity mutace, a tedy k intenzivnějšímu prohledávání prostoru. Pravidlo $\frac{1}{5}$ vyjadřuje vzorec

$$\sigma^{t+1} = \begin{cases} \sigma^t \cdot \frac{1}{c}, & P_s > \frac{1}{5} \\ \sigma^t \cdot c, & P_s < \frac{1}{5} \\ \sigma^t, & P_s = \frac{1}{5}, \end{cases} \tag{5.7}$$

kde P_s je poměr pozitivních mutací ze všech provedených mutací za určitý počet generací (implementováno po 1 generaci), σ^{t+1} a σ^t jsou upravené a neupravené síly mutací a $0 \leq c \leq 1$ je konstanta ovlivňující velikost mutace.

[3] Optimální hodnota úspěšnosti mutací ($\frac{1}{5}$) byla odvozena jako kompromis empirického testování $(1+1) - ES$ varianty na dvou testovacích funkcích. (Beyer – Schwefel, 2002).

– **krok 6: selekce nové populace** Posledním krokem každé iterace je obnova populace, kdy se vybírají noví jedinci (λ) do nové populace. Výběr probíhá v závislosti na volbě selekčního mechanismu ES (+ nebo , verze). Selekční mechanismus lze nastavit volbou parametru `typ`.

– **krok 7: kontrola ukončujících podmínek** Na konci každé iterace algoritmu se testuje, zda již nebylo nalezeno dostatečně kvalitní řešení, zda nebyl překročen limit počtu generací nebo jestli se vývoj populace nezastavil (viz 5.3). Pokud ani jedna uvedená možnost nenastala, algoritmus vstupuje do další iterace opakováním od kroku 2, pokud nastala, algoritmus je ukončen.

Výstupem algoritmu ES je tisk genomu nejlepšího jedince (transformační parametry), jeho `MSE` a transformační matice T na obrazovku. Tato transformační matice odpovídá transformaci, která je použita k registraci obrazů. Grafický výstup - vykreslení zaregistrovaných obrazů pro vizuální kontrolu a graf konvergence algoritmu - je volitelný v proměnné `zobraz`.

5.3.2 Implementace ABC

Algoritmus byl implementován podle schématu z kapitoly 4.1.2. Jediný rozdíl oproti základní verzi je v tom, že základní verze povoluje pouze jednu skautku během jednoho cyklu - tzn. v každé iteraci je nahrazeno pouze to známě řešení, které překročilo `LIMIT` a v jehož prohledávání okolí jsme byli nejvíc neúspěšní. V implementovaném algoritmu bývají nahrazena všechna řešení, jejichž prohledávání okolí překročilo hranici `LIMIT`.

Řídicí parametry

· `N` - počet agentů v populaci je primárně nastaven na hodnotu 25. Změnit ho lze nastavením parametru `SWARMSIZE`. Polovinu populace agentů vždy tvoří průzkumnice a druhou polovinu rekrutky - při zadání lichého čísla se počet zaokrouhlí. Explicitní změnu poměru druhů agentů lze uskutečnit změnou proměnné `pocet_rekrutek` - průzkumnicemi budou zbylí agenti populace. Počet obou druhů agentů je během procedury neměnný. Zvýšením velikosti populace lze dosáhnout větší spolehlivosti algoritmu. Velikost populace se opět projeví na výpočetním čase.

· `limit` - počet pokusů na úpravu v každém iteračním cyklu byl primárně nastaven na hodnotu 40. Hodnota 40 se při empirickém testování vyplatila nejlépe.

· `MAX_GEN` - maximální počet iterací algoritmu byl nastaven na 200. Hodnota 200 je sice mnohem vyšší, než je počet povolených iterací u předchozího algoritmu, ale z toho důvodu, že doba výpočtu algoritmem ABC je i při nynějším nastavení častokrát nižší než poloviční, můžeme si to dovolit. Kratší doba výpočtu jedné iterace je dána především počtem časově náročných operací (výpočet `MSE`), kterých je při nynějším nastavení ABC oproti ES téměř třetina. 200 iterací nebylo ovšem ke kvalitnímu výsledku většinou vůbec potřeba.

Do řídicích parametrů ABC se někdy započítává počet agentů typu skautka. Jejich počet je v algoritmu neměnný.

Průběh algoritmu a vyhodnocení

– **krok 1: inicializace** Registrace pomocí algoritmu ABC začíná volbou počáteční populace řešení. Ta jsou generována v prostoru přípustných řešení náhodně nebo vychází z nějakého počátečního odhadu, který je možno určit. Náhodné generování probíhá podle vzorce

$$\mathbf{x} = \mathbf{h}_{\text{spodní}} + \phi \cdot (\mathbf{h}_{\text{horní}} - \mathbf{h}_{\text{spodní}}), \tag{5.8}$$

kde $\mathbf{x}$ je řešení, $\mathbf{h}_{\text{spodní}}$ a $\mathbf{h}_{\text{horní}}$ vektory spodních a horních hranice omezujících podmínek a ϕ náhodné číslo z $[0,1]$. V algoritmu bylo ale zvoleno totožné generování počáteční populace jako u algoritmu ES (5.4). Střední hodnota opět vyjadřuje nasazení těžišť obrazů na sebe a směrodatné odchylky mnohorozměrného normálního rozdělení mají stejné hodnoty jako u ES.

V první fázi každé iterace se určí kvalita jedinců (`MSE` neboli $fitness(\mathbf{x})$). Po ocenění přechází průzkumnice do další fáze.

– **krok 2: fáze průzkumnic** V této fázi se v okolí každého známého řešení (průzkumnice) generuje nové řešení podle vzorce

$$\mathbf{x}_{\text{nové}} = \mathbf{x} + \phi \cdot (\mathbf{x} - \mathbf{x}_{\text{nahodné}}), \tag{5.9}$$

kde $\mathbf{x}$ označuje známé řešení, $\mathbf{x}_{\text{nahodné}}$ náhodně vygenerované řešení z populace, ϕ náhodné číslo z $[-1,1]$ a $\mathbf{x}_{\text{nové}}$ nově zkoumané řešení. Pokud je nové řešení lepší než původní řešení ($fitness(\mathbf{x}_{\text{nové}}) \leq fitness(\mathbf{x})$) nahradí jej, pokud je nové řešení naopak horší, o 1 se zvýší hodnota `abandonment`, která k danému řešení přísluší. Co je velmi důležité poznamenat, je to, že v průběhu tohoto náhodného generování dochází k úpravě pouze jediného parametru, který je volen náhodně. V tom spočívá velký rozdíl oproti operaci mutace z předchozího algoritmu.

– **krok 3: fáze rekrutek** Během této fáze dochází k výpočtu pravděpodobností, se kterými si průzkumnice vybírají zpravodajky (známá řešení), v jejichž okolí budou opět hledat lepší řešení. Pravděpodobnost výběru jednotlivých zpravodajek se počítá podle

$$p_i = \frac{fitness(\mathbf{x}_i)}{\sum_{j=1}^{N} fitness(\mathbf{x}_j)}, \tag{5.10}$$

pro $i = 1,2 \ldots N$, kde p_i označuje pravděpodobnost výběru řešení $\mathbf{x}_i$, $fitness(\mathbf{x}_i)$ označuje jeho vhodnost (v implementaci ekvivalent `MSE`) a N je počet všech rekrutek v populaci. Prohledávání okolí vybraného řešení v základní verzi ABC opět modifikuje jen jeden náhodně vybraný hledaný parametr (rozsah `DIM`). Realizace prohledávání je podle vzorce (5.9), kde $\mathbf{x}$ značí vybranou zpravodajku (zajímáme se jen o jeden její parametr). Stejně jako v kroku 2, dochází k porovnání vhodnosti

$\mathbf{x}$ a $\mathbf{x}_{\text{nové}}$ a dochází ke stejným důsledkům. Pokud je nové řešení lepší než původní řešení ($fitness(\mathbf{x}_{\text{nové}}) \leq fitness(\mathbf{x})$), dojde k náhradě původního řešení, pokud ne, dojde ke zvýšení příslušné proměnné `abandonment`.

- **krok 4: fáze skautek** Pokud nedošlo ke zlepšení vhodnosti ($fitness(\mathbf{x}_i)$) jednotlivých známých řešení (zpravodajek) v definovaném počtu pokusů (proměnná `LIMIT`), dojde k nahrazení tohoto známého řešení. Nové řešení je generováno v prostoru přípustných řešení podle (5.8).
- **krok 5: úprava nejlepšího řešení** Do proměnné `globMin` si uložíme transformační parametry dosud nalezeného nejlepšího řešení.
- **krok 6: kontrola ukončujících podmínek** Na konci každé iterace algoritmu se testují stejné podmínky jako v předchozím algoritmu.

Výstup algoritmu je stejný jako u ES. Na obrazovku se vytisknou transformační parametry nejlepšího jedince, jeho kvalita (`MSE`), transformační matice a v případě volby se vykreslí zaregistrované obrazy mozků nebo grafy zachycující průběh algoritmu. Při porovnání s algoritmem ES (funkce `run_AffRegComparison_AbcEs`) se výsledky obou algoritmů (parametry i transformační funkce) zapíší do souboru `*.txt` v adresáři, ve kterém je skript uložen.

5.4 Výsledky

Výkony algoritmů ABC a ES byly již mnohokrát testovány i porovnány na řadě testovacích úloh, které se staly standardem pro testování kvality optimalizačních algoritmů (benchmark functions). Výsledky těchto experimentů prokázaly, že výkony těchto dvou algoritmů jsou porovnatelné (Szeto et al., 2011). Praktická část této práce se věnuje konkrétní aplikaci obou algoritmů – registraci medicínských obrazů. Na této úloze budeme všechny tři (ABC a dvě varianty ES) algoritmy porovnávat. Algoritmy byly testovány na umělé sadě rozregistrovaných obrazů, která je součástí přiloženého DVD.

Aby bylo možné oba algoritmy formálně porovnávat, byla do základní verze algoritmu ABC přidána znalost těžiště a upraven způsob, kterým průzkumnice a rekrutky prohledávají okolí známých řešení (viz 5.3.2). Tato úprava byla nutná z toho důvodu, aby způsob mutace jedinců (normální rozdělení) příliš AES oproti ABC nezvýhodňoval.

Kvalita a přesnost registrace se hodnotily na základě dvou kritérií. Použitými kritérii byly hodnoty podobnostní míry `MSE` a skutečné hodnoty parametrů použité k umělému rozregistrování. Důvody použití více kritérií jsou následující: Pokud bychom hodnotili pouze podle hodnoty `MSE`, mohlo by docházet k situacím, kdy by na sebe obrazy pěkně přiléhaly, měly by nízkou hodnotu `MSE`, ale přitom by nemusely být dobře zaregistrovány - např. špatné otočení obrazů. To by se mohlo stávat zvláště v případech, kdyby se obrazy od sebe odlišovaly rotací o 180°. Tento problém uváznutí v lokálním extrému byl dokonce velmi častý, protože počátečním odhadem se na sebe přesunuly těžiště obrazů za nízké hodnoty

MSE a v průběhu algoritmů byl problém jejich registraci „vylepšit“. Proto bylo toto kritérium doplněno o porovnávání se skutečnými hodnotami rozregistrování. Toto kritérium lze ovšem použít jen u rigidní transformace či transformace se změnou měřítka 3.1.3, protože při použití obecnějších typů transformací by se kombinace několika elementárních transformací mohla rovnat nebo dobře aproximovat jinou elementární transformaci s jinými hodnotami transformačních parametrů. Z tohoto důvodu byly pro testování kvality registrace použity pouze transformace se změnou měřítka. Nutno ovšem podotknout, že právě rigidní transformace se v praxi běžně používají, protože u MRI obrazů se zkosení či rozdíly v měřítkách jednotlivých os příliš neobjevují. Algoritmy jsou ale schopny řešit i obecnější typy afinní transformace, avšak s rostoucím počtem optimalizovaných parametrů a nimi definovaným prostorem přípustných řešení klesá i jejich efektivita.

Pro korektní ohodnocování kvality zaregistrování bylo nutné ošetřit dostatečný průnik obou obrazů (3.10). Protože pokud by se jeden obraz transformoval úplně mimo prostor druhého obrazu, tak by se mohlo stát, že průnikem obou obrazů by byla velmi malá oblast, pro kterou by hodnota MSE nešla spočítat nebo by byla velmi malá. Velmi malé hodnoty MSE si u takových nekvalitních řešení nemůžeme dovolit, protože v rámci obou algoritmů pracujeme se všemi možnými řešeními. Tento problém byl ošetřen výpočtem vzdáleností těžišť obou obrazů, která nesmí překračovat určitou hranici. Vzdálenost těžišť obrazů je kontrolována funkcí `kontrola_prunik`.

V hodnocení kvality algoritmů standardně figuruje i časová a výpočetní náročnost. Proto byla i tato okolnost v porovnávání zohledněna. V medicíně mohou také nastávat situace, ve kterých sehrává doba registrace svou roli. Příkladem takové situace může být registrace obrazů během operací, která je prováděna v reálném čase. Výsledky testování jsou shrnuty v kapitole diskuze 6.

Sada testovacích obrazů se skládá ze sedmi obrazů. Jeden z nich byl vytvořen transformací s jediným parametrem (rotace), jeden byl vytvořen kombinací tří parametrů a k transformaci čtyř obrazů byla použita kombinace čtyř nebo pěti parametrů (viz tabulka 6.1). Posledním obrazem sady je obraz původní. Obrazy sady mají velikost 247×247 pixelů, kde jeden pixel odpovídá rozměrům $1mm \times 1mm$. Aby bylo možné hodnotit obrazy na základě obou stanovených kritérií, byla k hledání transformačních parametrů použita afinní transformace zahrnující pět transformačních parametrů d_x, d_y, ϕ, s_x, s_y. Navíc při zobrazování magnetickou rezonancí ke zkosení téměř nedochází. Veškeré experimenty byly prováděny na 50 nezávislých měřeních. Menší průměrná hodnota MSE a rozdílů vypočítaných transformačních parametrů od skutečných hodnot značila kvalitnější výkon algoritmu a menší standardní odchylka zase vyšší stabilitu algoritmu. Souhrnné výsledky testování vizualizují grafy 6.4-6.21, kde menší hodnota MSE značí kvalitnější registraci. Vztah hodnoty MSE a kvality zaregistrování ilustrují obrázky 6.1.

Pro názornost výstupu algoritmů a vizuální kontrolu funkčnosti algoritmů jsou v příloze přiloženy dva obrázky. Jedním z nich je porovnání obrazů před a po registraci při použití implementovaných algoritmů (obrázek 6.2) a druhým je graf, který zobrazuje průběh a konvergenci algoritmu k optimu (obrázek 6.3).

Kapitola 6

Diskuze

Výsledky registrace všech tří algoritmů i jejich variant se od sebe podstatně lišily. Zvláště pokud porovnáme výkon algoritmu ABC oproti oběma verzím ES. ABC byl nejen mnohem rychlejší, ale také podával mnohem stabilnější a kvalitnější výkony než jeho dva konkurenti ve všech prováděných testováních (viz obrázky 6.4-6.21). Detail výkonu ABC zaznamenává obrázek 6.21. Nicméně ani jeho výkon nebyl zcela bezchybný, protože v některých případech nalezl místo řešení optimálního řešení suboptimální. V porovnání se dvěma zbylými algoritmy, jsou jeho výkyvy téměř zanedbatelné a rozregistrování jen ojediněle překračovalo odchylku obou obrazů 1 *mm*, což lze považovat za přijatelný výsledek. Srovnání výkonu ES (s pravidlem $\frac{1}{5}$ - později AES) oproti výkonu $(\mu \stackrel{+}{,} \lambda) - SA - ES$ (později jen SAES)už není tak jednoznačné, protože oba algoritmy podávaly různé výkony pro týž počet transformačních parametrů použitých k rozregistrování obrazů (viz obrázky 6.5-6.6 a 6.8-6.9). Výkony obou algoritmů se značně lišily při registraci obrazu, který byl rotován o 180° 6.19-6.20. Při registraci tohoto obrazu byly již použity pouze `plusové` strategie, protože ty při registraci vykazovaly mnohem lepší výkony. Výsledky registrace a vhodnosti použití těchto strategií jsou dokonce v rozporu s doporučením autorů používat k řešení spojitých optimalizačních úloh `čárkovou` verzi (Beyer – Schwefel, 2002).

Proč se výkon algoritmu ABC oproti oběma verzím ES o tolik lišil můžeme jen odhadovat z konstrukcí algoritmů. Nejjednodušeji lze tento rozdíl vysvětlit nízkým počtem měněných parametrů během jedné iterace u ABC, která se musela při této konkrétní úloze osvědčit. Z tohoto důvodu byl také v průběhu testování změněn počet iterací, během kterých je tolerováno „uváznutí“ v lokálním extrému (z 10 na 20). Vhodná úprava jednoho parametru u ABC stačila k uniknutí z lokálního extrému a přitom se časově vyplatilo ji umožnit, protože algoritmus ABC zlepšoval odhady transformačních parametrů většinou po skocích. Demonstraci skokového chování ABC zobrazuje graf jeho průběhu 6.3. Algoritmy ES oproti tomu měnily kvality dosažených řešení velmi pozvolně, zvláště v pokročilejší fázi jejich běhu (viz obrázek 6.3). Druhým rozdílem algoritmů, který by mohl vysvětlovat lepší výkon algoritmu ABC, je téměř „stabilní intenzita“ prohledávání prostoru přípustných řešení po celou jeho běhu, protože agenti ABC prohledávají prostor přípustných řešení mezi sebou pořád se stejnou „silou“. Volba počáteční populace by měla díky agentům typu skautka více ovlivňovat algoritmus ES, zvláště pro rekombinativní rekombinaci. Nelze také nepoukázat na ekvivalentní počet jedinců přežívajících v každé generaci

u ABC i obou verzích ES (8).

Evoluční strategie velmi často uvízly v lokálním extrému a musely být ukončeny násilně. To platilo pro obě implementované varianty, i když mnohem častěji pro případ SAES (adaptivní verzi ES). Důvodem tohoto chování může být doplňování populace jedinců velmi nekvalitními jedinci s nevhodnými směrodatnými odchylkami, díky nimž se jednoduše zastavuje vývoj populace. Jedinci jsou sice do nové generace selektování, ale i v množině nevhodných řešení jsou některá řešení vhodnější než jiná. Tento problém se více týká `čárkové` strategie, protože nezachovává v populaci ty kvalitnější jedince z předešlých populací. Dalším možným důvodem neefektivního výkonu algoritmu SAES by mohlo být špatné nastavení řídicích parametrů, což je obecně problémem všech metaheuristických algoritmů. Nedeterminovanost chování těchto algoritmů uživatelům totiž nedává do rukou žádný rigorózní matematický aparát pro jejich optimální nastavení. Uživatelé se tak mohou opírat pouze o doporučení autorů, jiných uživatelů či stavět na svých vlastních experimentech. Použití metody $(\mu \overset{+}{,} \lambda) - SA - ES$ pro hledání parametrů afinní transformace o 5 optimalizovaných parametrech přidávalo k základním řídicím parametrům ES 6 dalších řídicích parametrů, které bylo nutné nastavit. Hodnoty zvolené autorkou této práce se potom jednoduše mohly vyrušovat v kombinaci s jinými omezeními algoritmu. Výhoda a současná nevýhoda algoritmu SAES může spočívat v začleňování náhodně zmutovaných řídících parametrů do genomu jedinců před jejich oceněním. Jelikož se tento algoritmus v praxi využívá, bezesporu existují určité úlohy, ve kterých má toto mutování smysl, registrace obrazů ale není podle prováděných experimentů jednou z nich (nebo šlo o již zmiňovaný špatný odhad parametrů). Návrhem na zlepšení by mohlo být zvětšení populace rodičů nebo doplňování populace „ověřenými“ jedinci.

ES s pravidlem $\frac{1}{5}$ vracely o trochu lepší výsledky ve všech pokusech kromě registrace obrazu, který byl schválně rotován o 180°. V tomto případě mnohem častěji uvízl v lokálním optimu. Nevýhoda metody $\frac{1}{5}$ spočívala v tom, že v pokročilé fázi optimalizace příliš rychle zmenšovala mutační sílu, kvůli tomu bylo čím dál obtížnější nacházet lepší řešení a míra pozitivních mutací byla často nižší než $\frac{1}{5}$). Při poklesu směrodatné odchylky na hodnoty v řádu tisícin už prakticky nedochází ke změně optimalizovaných hodnot. Řešením by bylo nastavit pro směrodatnou odchylku nějaký práh, pod který by nemohla klesnout. Konvergenčním vlastnostem algoritmu by to ale zřejmě moc nepomohlo, ani neurychlilo. Další okolností je to, že optimální hodnota poměru úspěšných mutací ke všem mutacím ($\frac{1}{5}$) je optimální nastavení pro verzi $(1+1) - ES$. Proto by možná mělo smysl vyzkoušet jiný poměr (Beyer – Schwefel, 2002).

Výraznější vliv počtu parametrů použitých k umělému rozregistrování nebyl ve výsledcích zaznamenán. K jeho podpoření či vyvrácení by bylo potřeba provést mnohem více měření. Naopak výrazný vliv na úspěšnost algoritmů se projevil ve volbě typu afinní transformace - výsledky tohoto testování nebyly do práce zahrnuty. S komplexnějšími transformacemi se totiž exponenciálně zvyšuje prostor přípustných řešení, což se projevuje ve snižování přesnosti registrace. Stejné chování vykazovaly všechny algoritmy. Navíc pro použití různých transformačních funkcí mohou také být vhodnější jiná nastavení řídicích parametrů.

Schopnost algoritmů uniknout z lokálního extrému byla demonstrována registrací posledního obrazu `obr_px2_py_10_rPi`, který byl oproti originálu posunut a rotován o 180°. Algoritmus ABC byl ve všech případech schopen najít správnou rotaci obrazu, ať už o 180° nebo −180° (ekvivalentní), a „jen“ 21 krát z 50 pokusů se mu nepodařilo splnit kritérium pro maximální odchylku obrazů, kterou jsme před započetím registrace stanovili 0.5 *mm* 6.18. A to i přes to, že předčasná konvergence je algoritmu ABC vyčítána (Verma – Kumar, 2013). U registrace tohoto obrazu se také projevil velký rozdíl ve verzích Evolučních strategií. ES $\frac{1}{5}$ nacházel správnou rotaci obrazu velmi často, ale s nedobrou výslednou kvalitou registrace, kdežto pro SAES bylo nalezení správně rotace pouhou výjimkou (obrázky 6.19-6.20). Z aplikace registrace tohoto i předchozích obrazů a experimentů lze vyvodit, že tyto tři algoritmy jsou vhodné k registraci obrazů podle následujícího pořadí: ABC, ES a nejméně vhodný z nich – SAES. Nakonec jen poznamenejme, že sada testovacích obrazů sice neobsahuje skutečné a zašuměné obrazy, které se v praxi registrují, to ale do jisté míry kompenzuje nereálné rozregistrování obrazů 6.1.

Vraťme se ale zpátky k obecným metaheuristickým algoritmům a jejich zdrojům inspirace. V předchozích kapitolách byli již zmíněno, že zdroje inspirace metaheuristických algoritmů jsou velmi obšírné a že by mělo smysl bádat v přírodních procesech po jejich možném vylepšení. Současné literatura čítá více než 40 přírodou inspirovaných algoritmů (Fister Jr et al., 2013) a jejich počet neustále roste. Na jednu stranu to je dobře, ale na druhou stranu vysoký počet algoritmů ještě neznamená, že je jich hodně kvalitních. Podporovat vývoj algoritmů, které principiálně fungují stejně jako algoritmy již navržené a jen jsou jinak pojmenované (stejné myšlenky jinak zabaleny) není úplně košer. Snaha upozornit na tento fenomén není jen výmyslem autorky této práce, ale i jiných autorů (Fister Jr et al., 2013; Soerensen, 2015). Další věcí je občas vyloženě tristní popis činnosti algoritmů, jako např. (Yang, 2005). A používaní jiných termínů pro podobné děje (náhodné mutace jedinců) nebo používání přílišných přírodních metafor v popisech principů algoritmů k jejich čitelnosti moc nepřidává. Přehled algoritmů v první kapitole práce se snažil balancovat na hranici, kdy jsou používané metafory a konkrétní označení potřebné pro dodatečné studium principů algoritmů a kde by již byly spíše na obtíž. Co by bylo možné ještě označit za slabinu odvětví optimalizačních algoritmů je optimální nastavení řídicích parametrů. Ta jsou v literatuře někdy velmi špatně dohledatelná, přestože nastavení řídicích parametrů má na výkon optimalizačních algoritmů téměř zásadní vliv. Ladění těchto parametrů také někdy není úplně banální záležitost.

Návazností na tuto práci by mohlo být nespočet. Pokud bychom chtěli vhodnosti algoritmů (především algoritmu ABC) k problému registrace medicínských obrazů dále testovat (ověřit), mělo by smysl vyzkoušet jej na obrazech skutečných (ne dvojici uměle vytvořených obrazů), třírozměrných, zašuměných a pořízených různými modalitami. Naimplementované Evoluční strategie se sice pro registraci obrazů příliš neosvědčily, ale jak již bylo napsáno v 4.3, jiná jejich varianta ano. Vyzkoušet proto jinou verzi ES nebo upravit probírané algoritmy, by nemuselo být úplně beze smyslu. Vylepšení algoritmů by mohlo vycházet z biologické inspirace, z kombinace s nějakou lokální metodou, která by registraci upřesnila, nebo bychom také mohli využít jiných postupů, které se k registraci

obrazů využívají - příkladem by mohla být vícefázová registrace, kdy bychom nejdříve hledali hrubou registraci při nízkém rozlišení obrázků (nebo při jejich rozmazání) a jejich postupným zvětšováním (či doostřováním) bychom tento odhad stále vylepšovali. V návaznosti na zkoumání paralel mezi přírodou a optimalizačními algoritmy a jejich výkony by bylo velice zajímavé otestovat, zda jsou principy a strategie přírody pro optimalizaci tou nejlepší volbou.

A abychom toto téma alespoň pro tuto práci nějak uzavřeli, tak pro registraci obrazů je zásadní volba tří komponent - optimalizační metody, podobnostní míry a typu transformační funkce. Důležitost volby optimalizační metody může spočívat ve schopnosti prohledávat prostor přípustných řešení (viz výkon ABC vůči ES), důležitost volby podobnostní míry jsme sice v této práci nedemonstrovali, ale již z popisů v předcházejících kapitolách by mohlo plynout, že volba kriteriální funkce sehrává při registraci svou roli. Použití příliš obecného typu transformace je nejen časově náročnějším a komplexnější problém, ale vrací i méně přesné výsledky registrace. Pokud jde o témata dvou diskutovaných algoritmů, tak samoadaptace v optimalizačních algoritmech má určitě své výhody, ale i nevýhody, a ty se v práci spíše projevily. Nevýhodou samoadaptace je to, snižuje divergenci populace, někdy až příliš rychle, díky čemuž algoritmy předčasně konvergují, nebo naopak doplňuje populaci řešeními, která jsou pro ni nesměrodatná. Algoritmus umělých včelách kolonií se pro registraci jednoznačně prokázal jako účinnější (tedy alespoň při tomto nastavení a k této třídě problému).

Závěr

Přírodou inspirované metaheuristické algoritmy dnes patří mezi populární optimalizační techniky, jejichž oblasti používání se neustále rozšiřují. V návrzích těchto algoritmů se propojuje teorie optimalizace s našimi představami o biologických dějích, které slouží jako předloha optimalizačních postupů. Tato bakalářská práce se mimo jiné snažila ukázat, že tato kombinace přírodních schémat s teoretickými znalostmi může vést k velmi schopným optimalizačním nástrojům.

Registrace obrazů je komplexní optimalizační úloha, při jejíž řešení je nutné zohlednit optimalizační metodu i povahu registrovaných obrazů, protože na výslednou kvalitu registrace mají velký vliv. Dnes se výzkum v registraci zaměřuje spíše na globální a nelineární registrační metody, které dnes nejsou tak efektivně vyřešeny jako metody lokální lineární registrace. V práci použité metaheuristické algoritmy jsou jen dvěma zástupci z této třídy (globálních metod) algoritmů. Aplikace obou implementovaných algoritmů na registraci medicínských prokázala, že obě metody jsou robustní vůči počátečního odhadu a jsou schopny celkem spolehlivě nalézt globální optimum. Dosažené výsledky korespondovaly také s tvrzením, že některé algoritmy jsou v kontextu s konkrétním problémem (třídou problémů) mnohem efektivnější než jiné. K nalezení globálních optim ovšem oba algoritmy vyžadovaly relativně hodně času a byly (i obecně jsou) mnohem pomalejší a méně přesné než metody lokální. Tato skutečnost by se mnohem více projevila na celých třírozměrných datech mozku. Slabinou metaheuristických algoritmů je také nestabilní výkon, protože tyto metody někdy podávají (a podávaly) kvalitní výsledky, jindy méně kvalitní a někdy žádné (transformační parametry se nelišily od započetí registrace - nasazení na sebe těžiště). Obecně se ale ukázalo, že tyto algoritmy jsou schopny vracet výsledky, které se od optimálních výsledků liší jen zanedbatelně, a proto je vhodné spustit je vícekrát. V praxi se globální metody obyčejně používají k hrubé registraci obrazů, kterou lokální metody upřesňují, čímž se využívá kvalit obou tříd metod a jejich nedostatky se naopak potlačují. Cesta dalšího možného postupu vylepšení registrace by možná byla ve vyzkoušení i jiných metaheuristických algoritmů, ačkoli mnohem příhodnější cesta by byla pokusit se zefektivnit metody stávající, třeba i v této práci použité, nebo se pokusit je vhodně zkombinovat.

Samostatná třída metahuristických algoritmů je rozsáhlá svým počtem i zdroji inspirací, ze kterých autoři algoritmů při jejich návrzích čerpají. Příroda stejně tak podněcuje kreativitu lidí v mnoha dalších odvětvích lidské činnosti. Proto na závěr práce nezbývá než dodat, že příroda je bohatým zdrojem inspirace a je nasnadě se domnívat, že nám má jako zdroj inspirace (možná i metaheuristických algoritmů) ještě co nabídnout.

Příloha

Součástí této práce je DVD se zdrojovými kódy naprogramovaných skriptů a funkcí (napsáné v prostředí MATLAB), které byly využity k provedení experimentů, a datovou sadou uměle vytvořených obrazů, na níž byly algoritmy a registrace testovány. Popis DVD a jednotlivých funkcí je následující.

Přímé porovnání registrací obrazů pomocí algoritmů ABC a ES lze spustit funkcí `run_AffRegComparison_AbcEs`. Pro její funkčnost není nutné mít v MATLABu doinstalovány žádné rozšiřující knihovny. Výstupem každého algoritmu (AES a ABC) je transformační matice T, transformační parametry $(d_x, d_y, s_x, s_y, \phi, hx, hy)$, které jsou v z matice T extrahovány funkcí `parameter_extraction`[1] a transformační parametry inverzní transformace. Výstupy obou algoritmů se vytisknou na obrazovku a zapíší do souboru `*.txt`. Jméno souboru `(*)` je argumentem funkce `run_AffRegComparison_AbcEs`. Pokud je vyžadován grafický výstup (parametr `zobraz`), zaregistrované obrazy se přes sebe vykreslí. Ukázkou takového výstupu mohou být následující dva obrázky 6.2, 6.3.

Popis samotné funkce je:
`run_AffRegComparison_AbcEs(file, zobraz1, zobraz2)`
`file` – soubor, do kterého se zapisují výsledky (formát `.txt`)
`zobraz1` – boolean proměnná – 0: zaregistrované obrazy se přes sebe nevykreslí
1: zaregistrované obrazy se vykreslí
`zobraz2` – boolean proměnná – 0: žádný grafický výstup samostatných algoritmů
1: samostatné algoritmy budou mít grafický výstup

A stěžejní naimplementované funkce práce s jejich argumenty jsou:
`ABC(swarmsize, MAX_GEN, LIMIT, runtime, HRANICE_KVALITY, typ_reg, fixni, plovouci, Lb, Ub, zobraz)`
`swarmsize` – počet jedinců (včel) v populaci
`MAX_GEN` – maximální počet generací, ukončující podmínka
`LIMIT` – maximální počet prohledávání v okolí každého známého řešení
`runtime` – počet běhů algoritmu
`HRANICE_KVALITY` – určuje vrchní hranici kriteriální funkce, od které jsou už jedinci dostatečné kvalitní (minimalizace)
`typ_reg` – specifikuje typ registrace – 3: rigidní transformace
4: podobnostní transformace

[1] To platí pouze pro jednodušší transformace než je transformace se změnou měřítka (včetně). Pro jiné transformaci vrací funkce `parameter_extraction` nulové hodnoty.

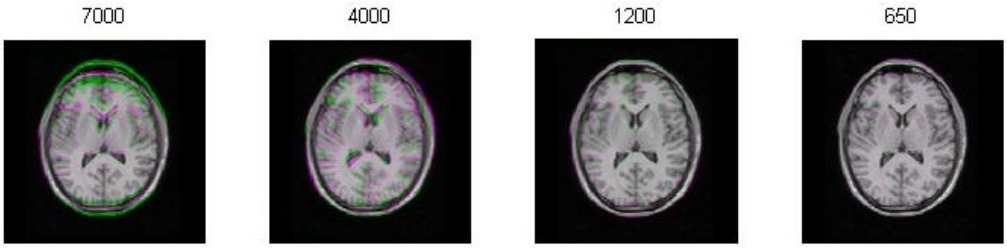
5: transformace se změnou měřítka
7: obecná afinní transformace

`fixni` – fixní obraz I_M
`plovouci` – plovoucí obraz I_S
`Lb` – spodní meze (hranice) hledaných parametrů
`Ub` – horní meze (hranice) hledaných parametrů
`zobraz` – boolean proměnná grafického výstupu – 0: graf a obrazy se nezobrazí
1: vykreslení registrovaných obrazů a vykreslení grafu konvergence

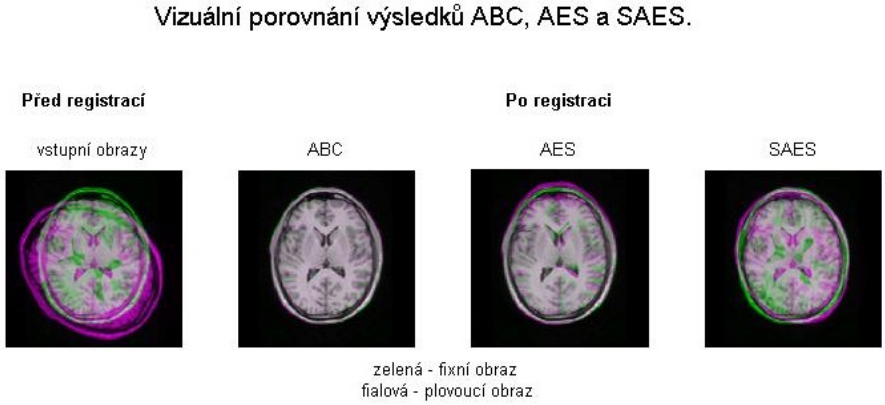
`AESS(ny, ro, lambda, typ, verze, MAX_GEN, TYP_REK, RUNTIME, HRANICE_KVALITY, typ_reg, fixni, plovouci, Lb, Ub, zobraz)`
`ny` – počet rodičů v populaci (μ)
`ro` – počet rodičů, jejichž křížením vznikají noví jedinci (ρ)
`lambda` – počet potomků v populaci (λ)
`typ` – určuje typ algoritmu – 1: plus verze
2: čárková verze
`verze` – určuje typ sebeadaptabilní formy algoritmu – 0: pravidlo $\frac{1}{5}$
2: sebeadaptabilní verze SAES
`TYP_REK` – určuje typ rekombinace – 1: rekombinativní rekombinace
2: diskrétní rekombinace
3: vážená rekombinantní
– zbytek argumentů má stejný význam jako argumenty u předchozí funkce.

Funkce používané v rámci těchto tří uvedených funkcí jsou podrobněji rozepsány až v samotných skriptech.

Datová sada uměle rozregistrovaných obrazů je součástí souboru `Dataset_mozky.mat`. Obsahuje 6 transformovaných obrazů a jeden původní obraz. Původní obraz je v souboru pojmenován `original` a zbylých 6 obrazů je pojmenováno * podle hodnot transformačních parametrů (3.4)-(3.5), které byly použity pro jejich rozregistrování. Místo * je vždy `ParametrHodnota`, podtržítko v názvu mezi číslicemi představuje desetinnou čárku a podtržítko také odděluje jednotlivé parametry. Například pojmenování `dx20_sx0_75` znamená, že obraz byl vytvořen afinní transformací originálního obrazu s transformačními parametry $d_x = 20$ a $s_x = 0.75$. Parametr ϕ (značeno `r` jako rotace) je v radiánech.

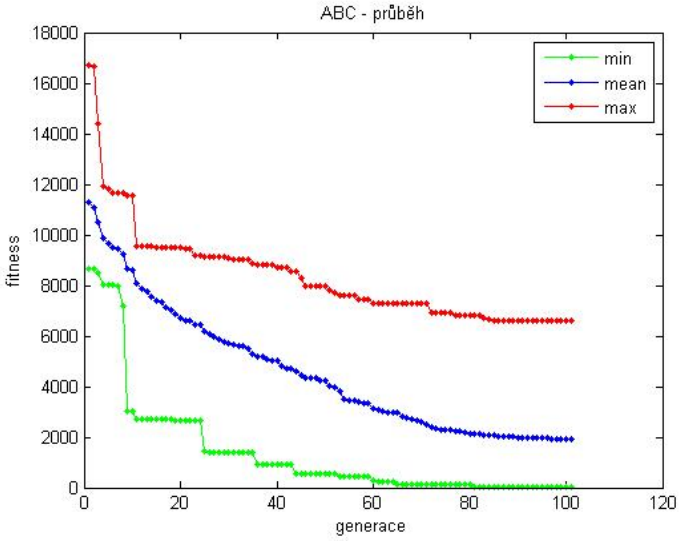


Obrázek 6.1: **Ukázka toho, jak souvisí výsledná hodnota MSE s kvalitou registrace.**.

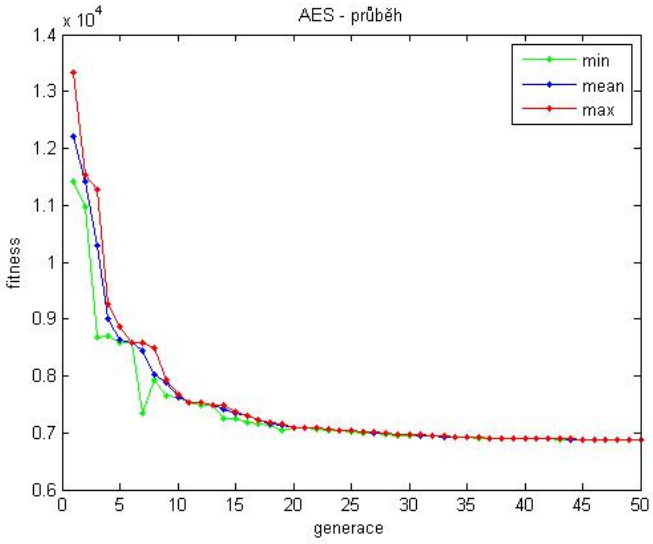


Obrázek 6.2: **Ukázka výstupu algoritmů ES a ABC - obrazy před registrací a po registraci**. V obrázcích jsou přes sebe vykresleny registrované obrazy mozku před registrací a po registraci. V levém obrázku (před registrací) je zelenou barvou zobrazen referenční obraz I_M a fialovou barvou obraz plovoucí I_S. Odstíny šedé v obrázcích vpravo (po registraci) dostaneme, když se intenzity zelené a fialové barvy porovnávaných obrazů přes sebe překryjí[a].

[a] To vychází z RGB barevného modelu - zelená je barva zeleného kanálu (Green) a fialová barva vznikne namícháním červeného (Red) a modrého (Blue) kanálu. Tyto dvě barvy (zelená a fialová) jsou v RGB modelu komplementární, jejich namícháním vzniknou odstíny šedi (0 - černá barva).

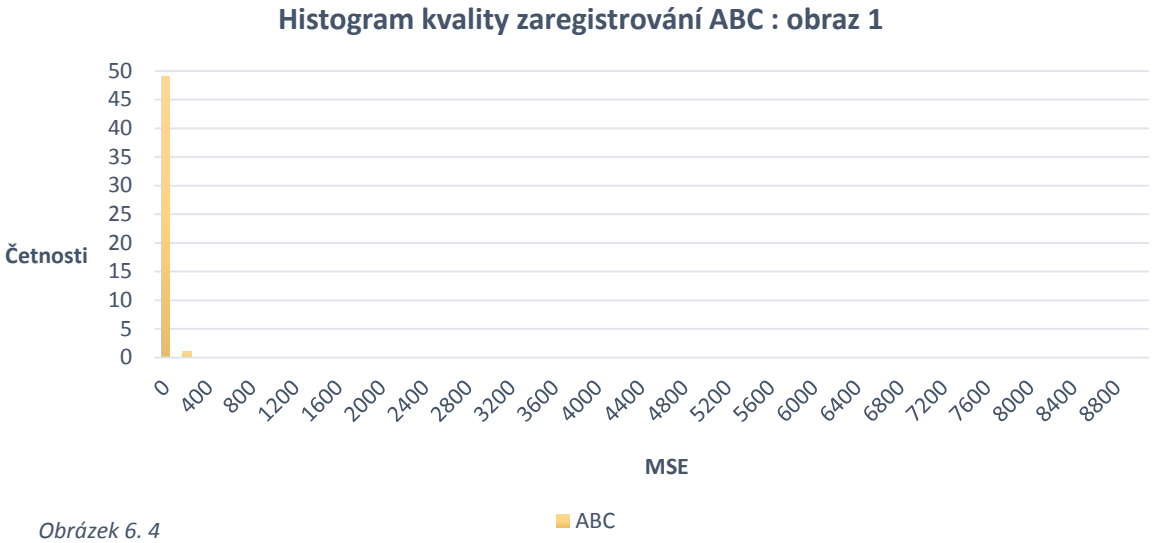


(a) Algoritmus umělých včelích kolonií

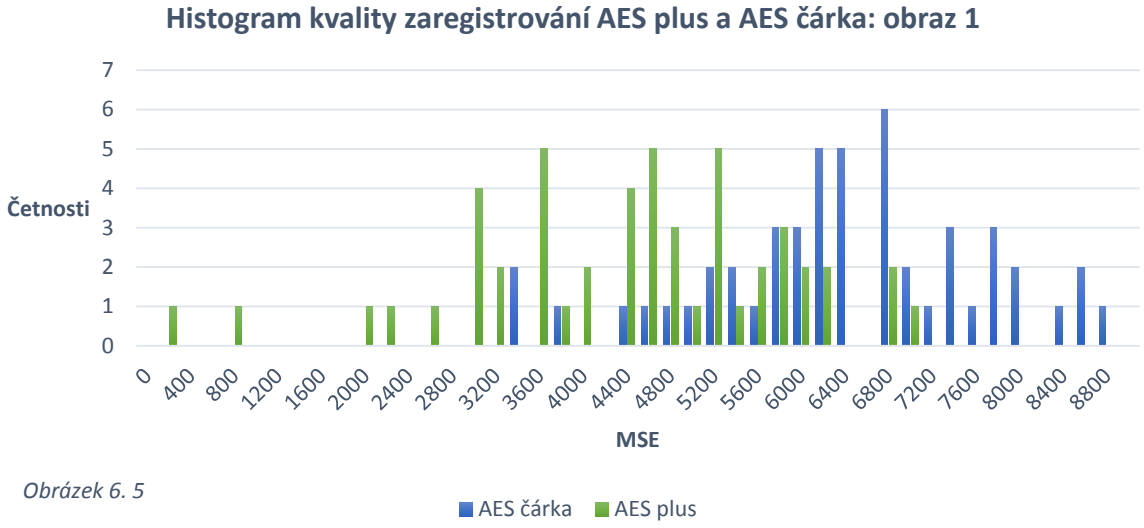


(b) Evoluční strategie - verze $\frac{1}{5}$

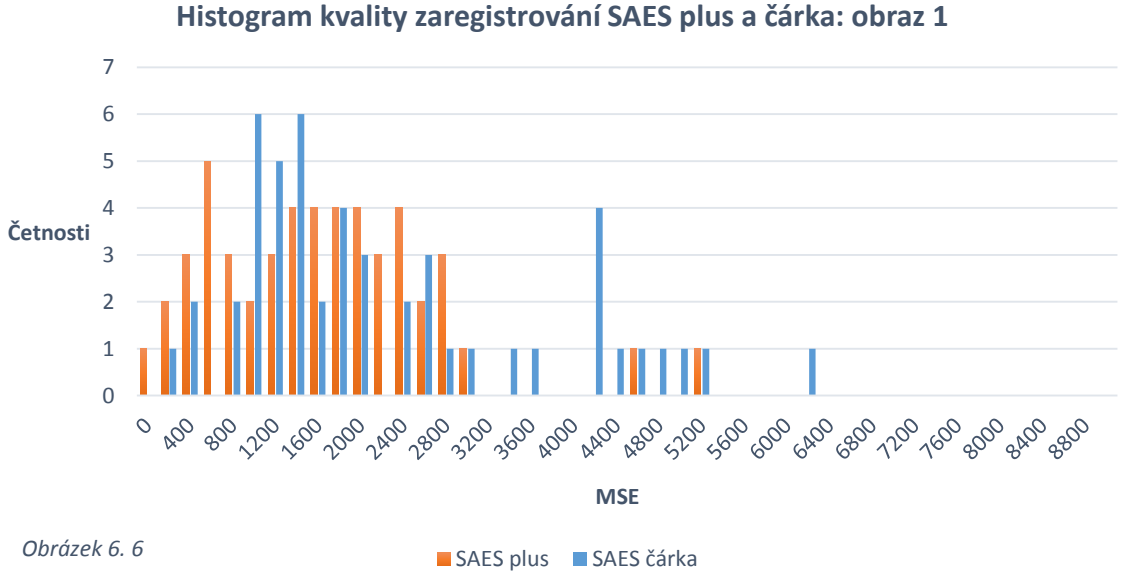
Obrázek 6.3: **Ukázka výstupu algoritmů ABC a ES - grafy průběhu algoritmů**. V grafech jsou zobrazeny typické průběhy optimalizačních algoritmů (při minimalizaci). Barevné křivky znázorňují vývoj maximálních, průměrných a minimálních hodnot `MSE` v populaci po generacích. Červená křivka znázorňuje maximální hodnotu `MSE`, modrá křivka zobrazuje střední hodnotu `MSE` a zelená křivka znázorňuje minimální hodnotu `MSE` v populaci v konkrétních generacích. Z průběhu této registrace lze vyčíst, že algoritmus byl ukončen násilně (`MAX_GEN=100` a `MAX_GEN=50`).



Obrázek 6. 4

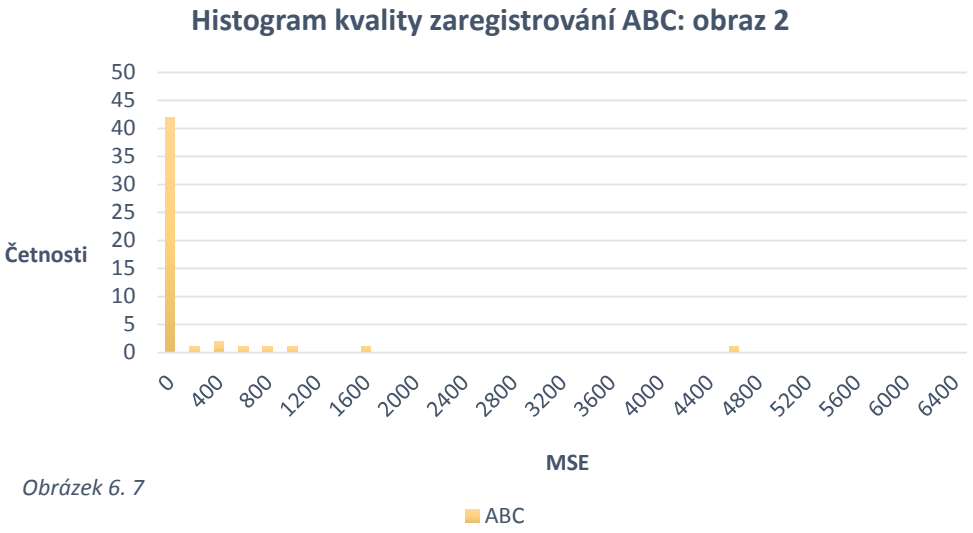


Obrázek 6. 5

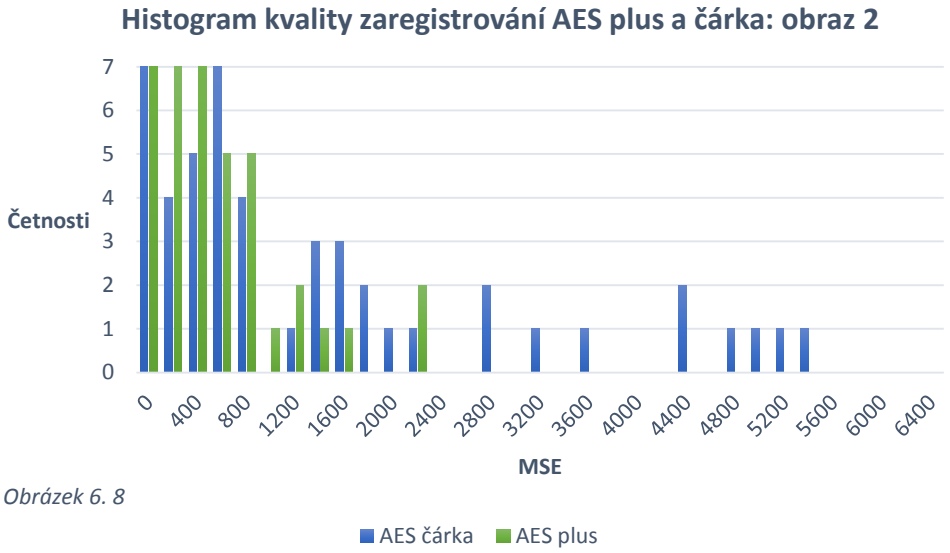


Obrázek 6. 6

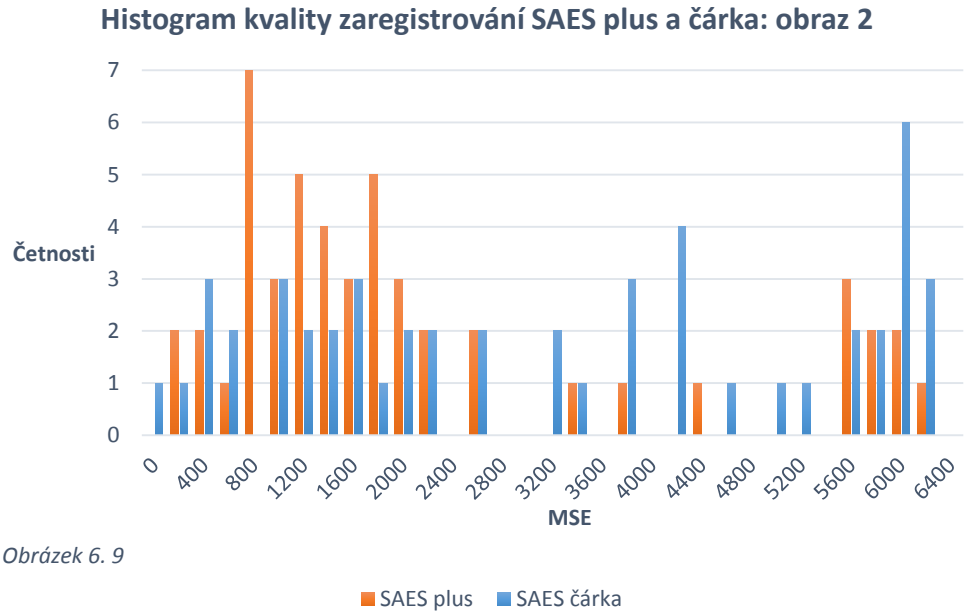
ABC – algoritmus umělých včelích kolonií
AES – Evoluční strategie s pravidlem 1/5
SAES – Adaptivní verze Evolučních strategie



Obrázek 6. 7

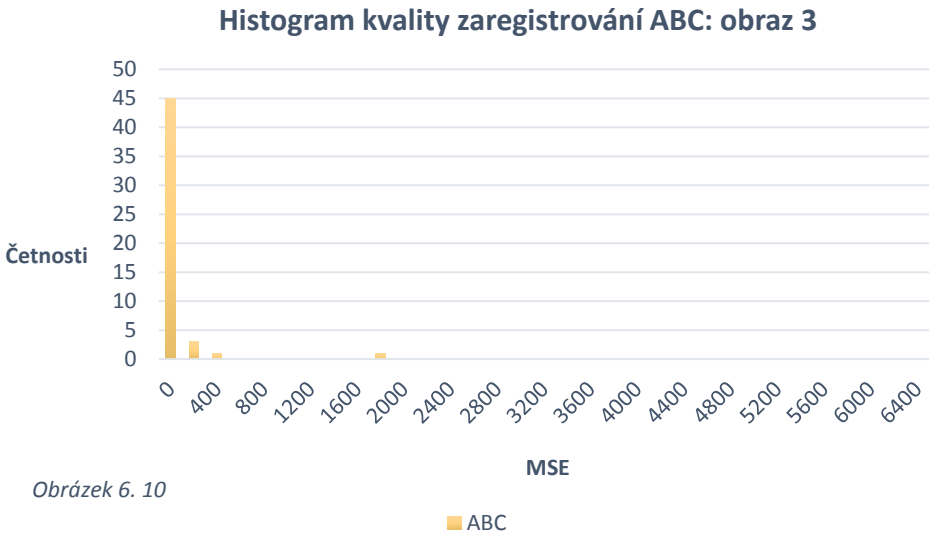


Obrázek 6. 8

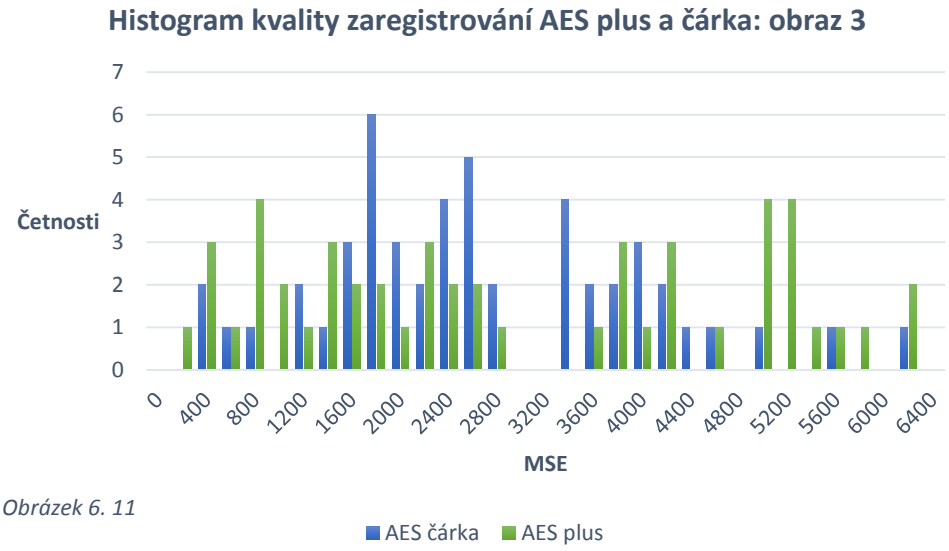


Obrázek 6. 9

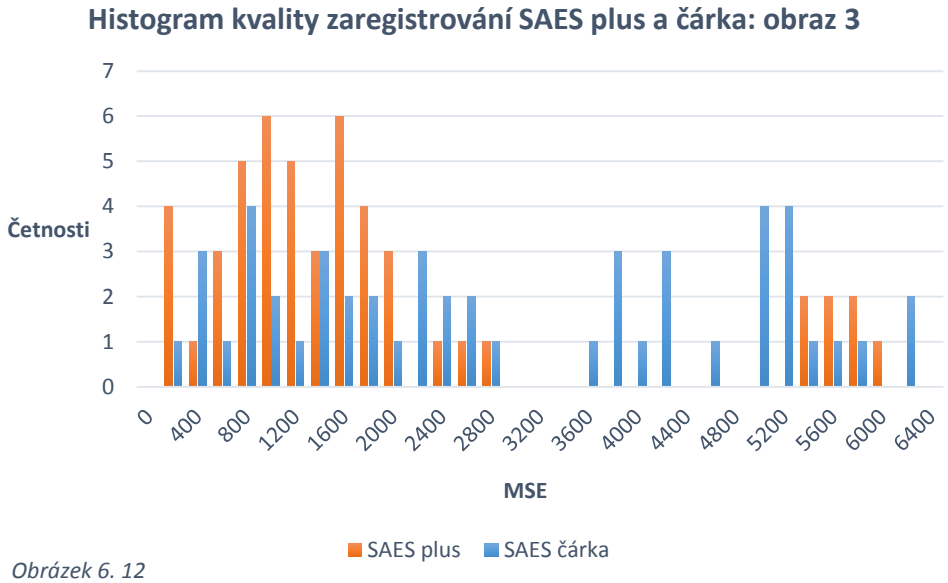
ABC – algoritmus umělých včelích kolonií
AES – Evoluční strategie s pravidlem 1/5
SAES – Adaptivní verze Evolučních strategie



Obrázek 6. 10

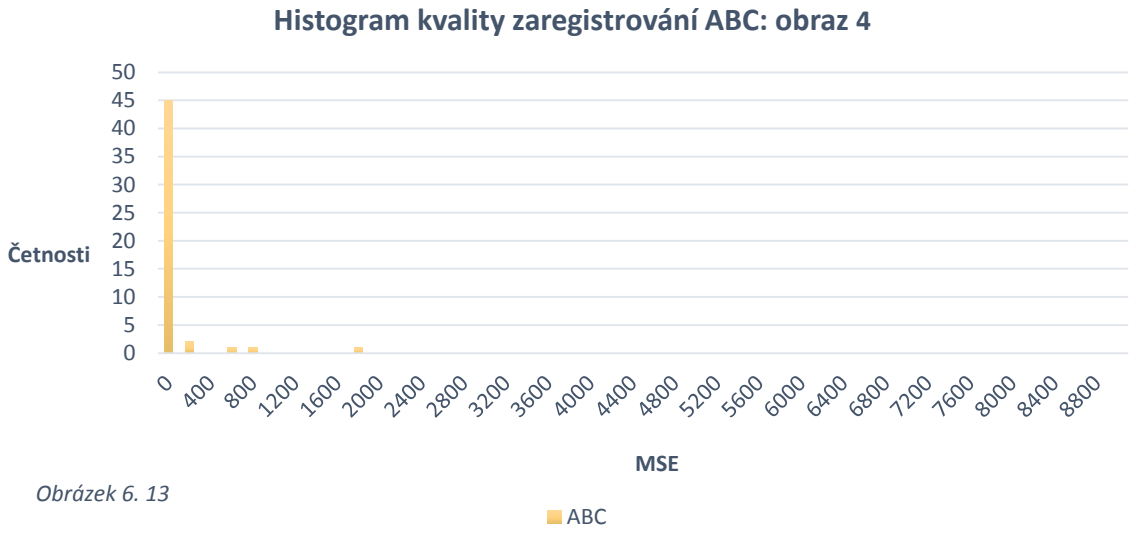


Obrázek 6. 11

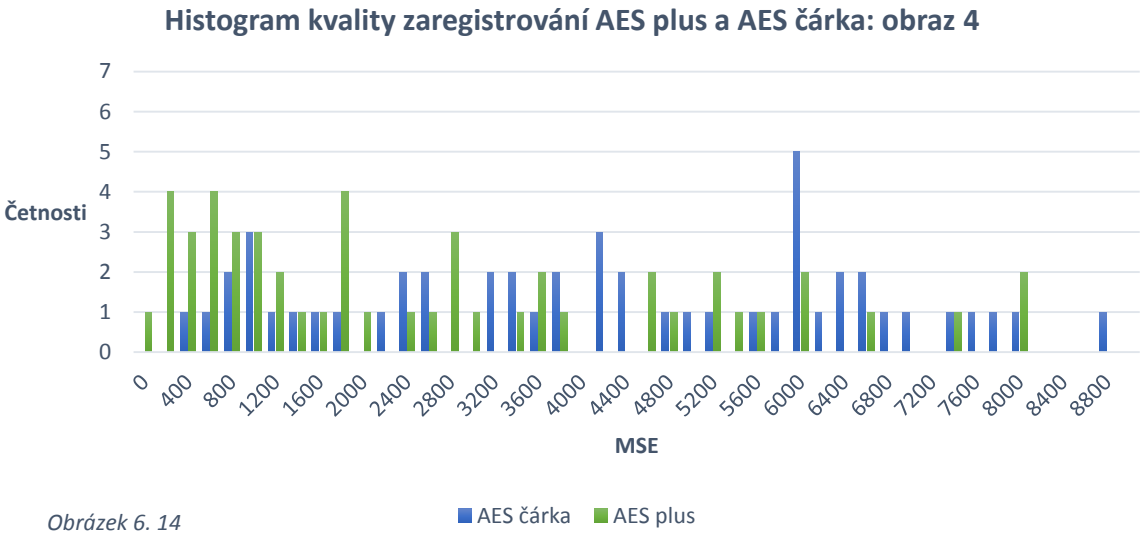


Obrázek 6. 12

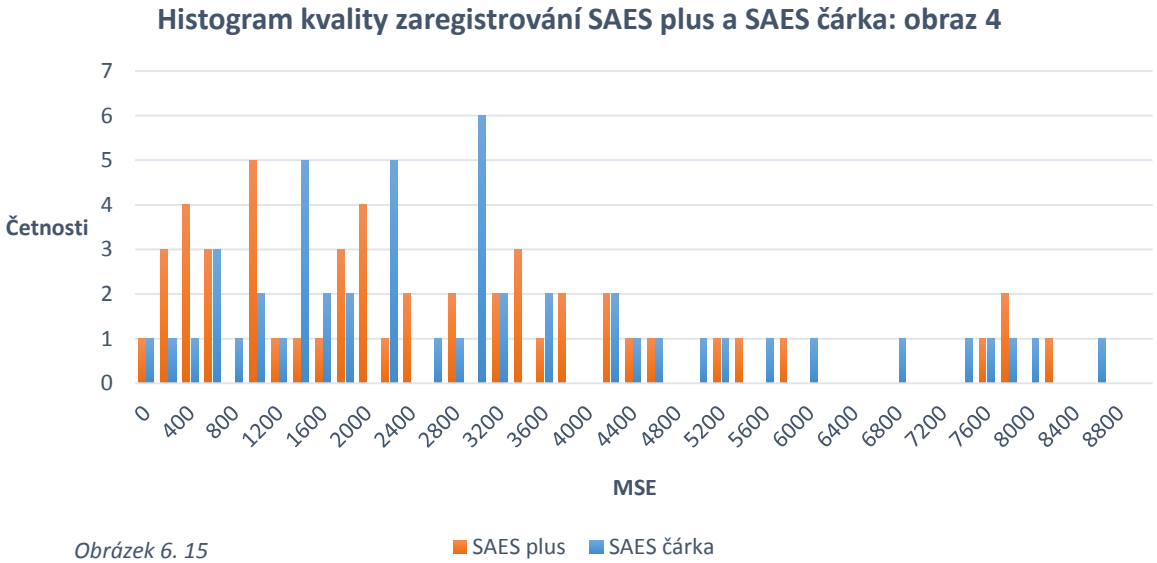
ABC – algoritmus umělých včelích kolonií
AES – Evoluční strategie s pravidlem 1/5
SAES – Adaptivní verze Evolučních strategie



Obrázek 6. 13

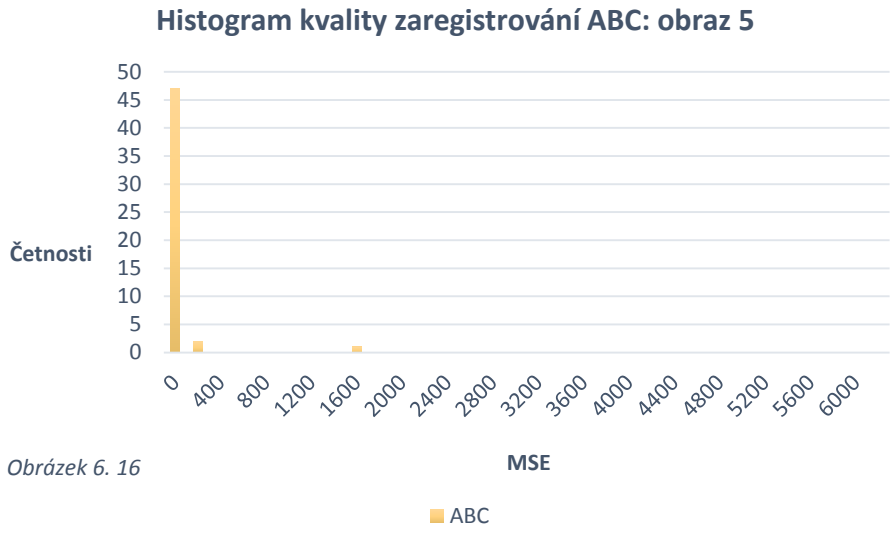


Obrázek 6. 14

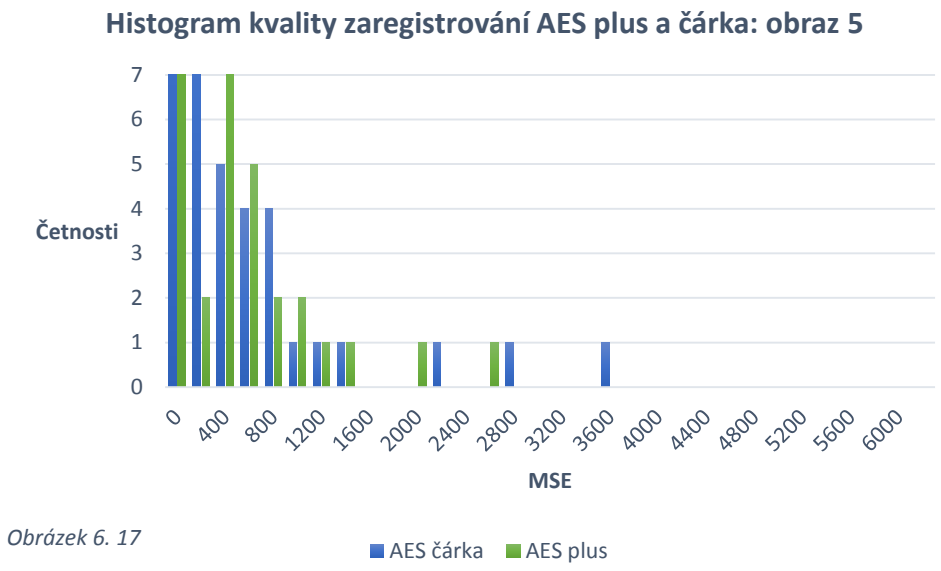


Obrázek 6. 15

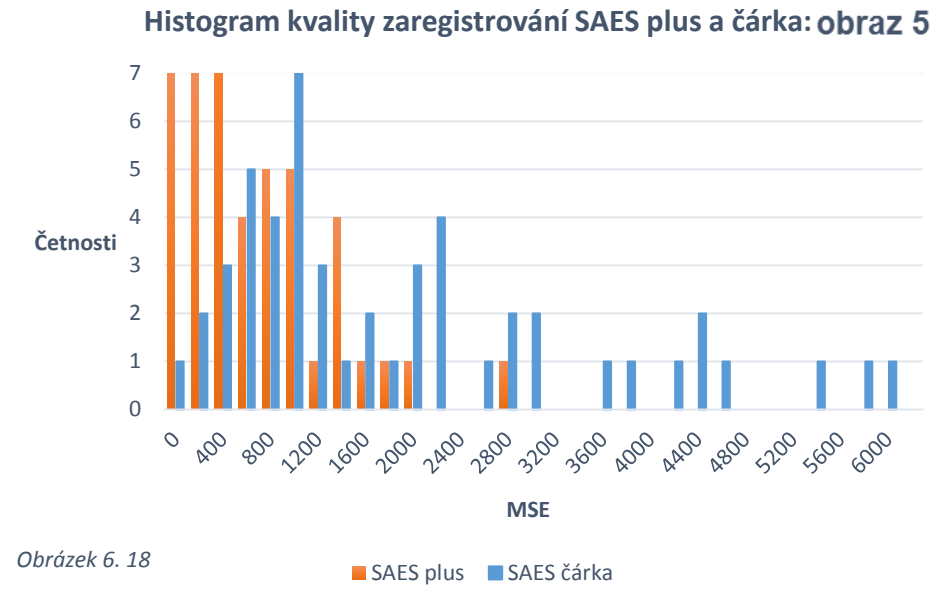
ABC – algoritmus umělých včelích kolonií
AES – Evoluční strategie s pravidlem 1/5
SAES – Adaptivní verze Evolučních strategie



Obrázek 6. 16

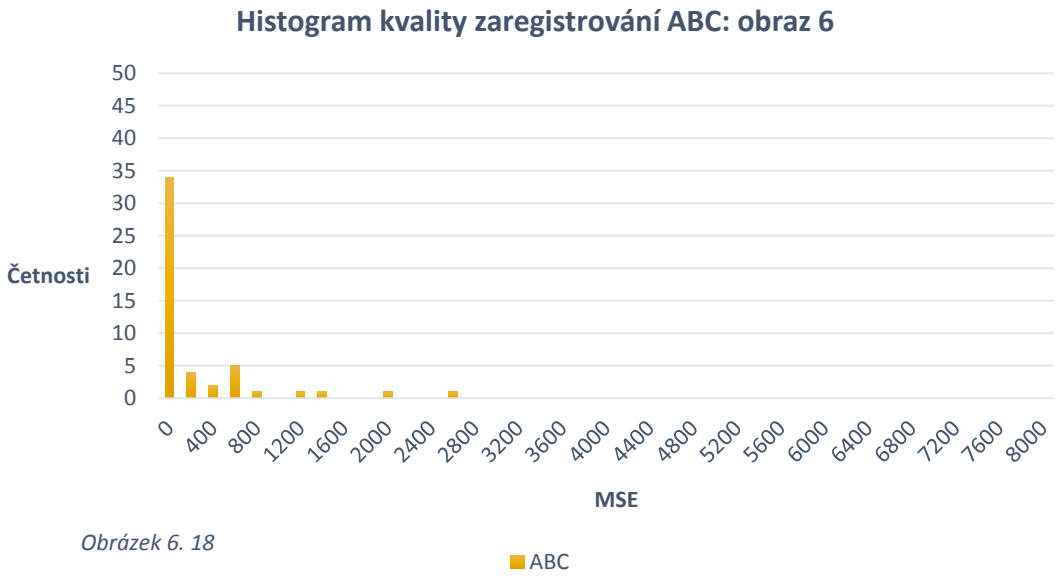


Obrázek 6. 17

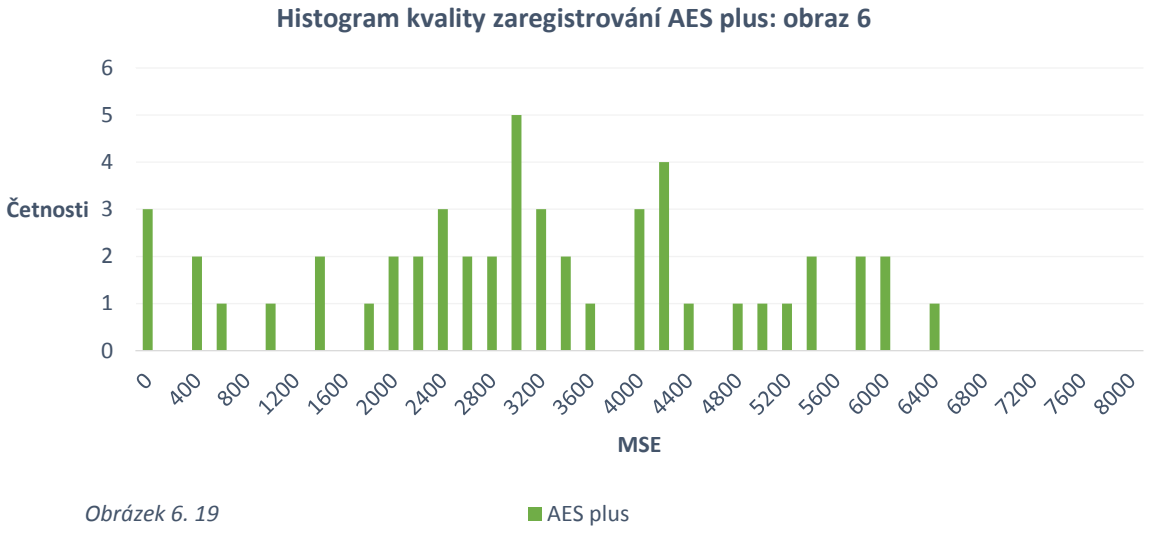


Obrázek 6. 18

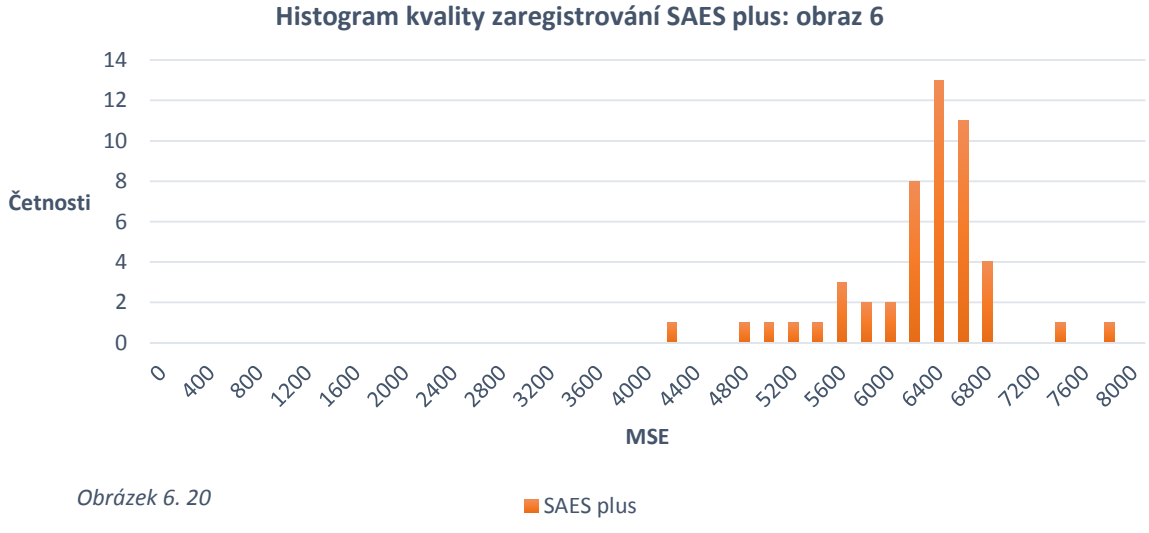
ABC – algoritmus umělých včelích kolonií
AES – Evoluční strategie s pravidlem 1/5
SAES – Adaptivní verze Evolučních strategie



Obrázek 6. 18

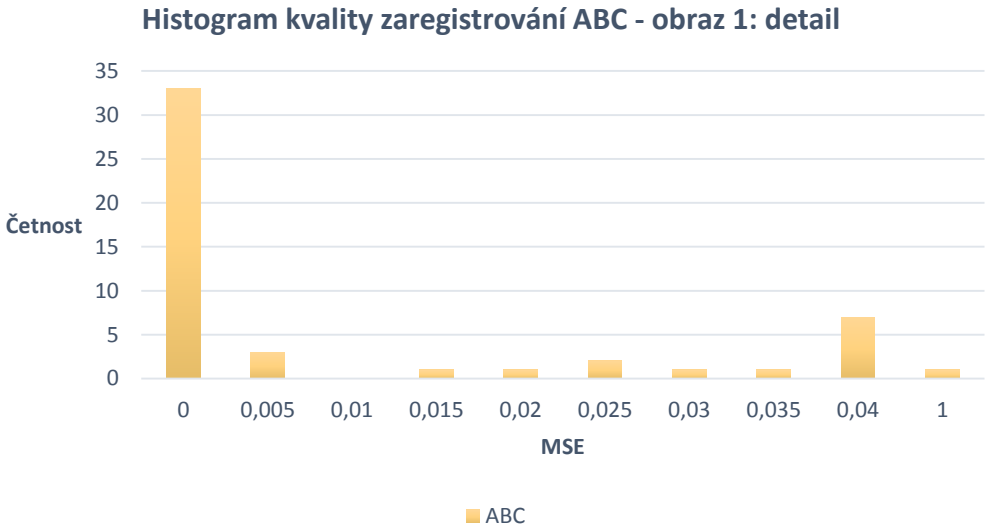


Obrázek 6. 19



Obrázek 6. 20

ABC – algoritmus umělých včelích kolonií
AES – Evoluční strategie s pravidlem 1/5
SAES – Adaptivní verze Evolučních strategie



ABC – algoritmus umělých včelích kolonií

Tabulka 6.1: **Tabulka s hodnotami transformačních parametrů použitými k rozregistrování originálního (referenčního) obrazu**

Název	Označení v sadě	Parametry	Hodnoty parametrů
Obraz 1	obr_dx10_dy15_r0_5_sx0_85_sy1_1	d_x, d_y, ϕ, s_x, s_y	10, 15, 0.5, 0.85, 1.15
Obraz 2	obr_dx1_dy15_r15_sx0_95_sy0_975	d_x, d_y, ϕ, s_x, s_y	1, 1.5, 15, 0.95, 0.975
Obraz 3	obr_dx1_dy15_r5_sx0_9_sy1	d_x, d_y, ϕ, s_x, s_y	1, 15, 5, 0.9, 1
Obraz 4	obr_dy15_r1_sx1_1_sy_1_15	d_y, ϕ, s_x, s_y	15, 1, 1.1, 15
Obraz 5	obr_r1	ϕ	1
Obraz 6	obr_dx2_dy_10_rPi	d_x, d_y, ϕ	2, 10, π
Obraz 7	original	-	-

Seznam literatury

MEN. Včelařství-Včelí snúška během roku. [online], aktualizace 02.05.2014. Dostupné z: <http://web2.mendelu.cz/af_291_projekty2/vseo/stranka.php?kod=2536>. [cit. 29.03.2015].

NAB. NABEECO Network Alignment with Bee Colony Optimization. [online], aktualizace 20.02.2015. Dostupné z: <http://nabeeco.mpi-inf.mpg.de/>. [cit. 28.03.2015].

ABOU-SHAARA, H. – OTHERS. The foraging behaviour of honey bees, Apis mellifera: a review. *Veterinarni Medicina*. 2014, 59, 1, s. 1–10.

AMPELLIO, E. – VASSIO, L. Artificial super-Bee enhanced Colony (AsBeC) algorithm for numerical optimization with limited function evaluations Part 1: technologies and benchmark validation. *ArXiv e-prints*. feb 2014, s. 19.

ANTONIOU, A. – LU, W.-S. *Practical optimization: algorithms and engineering applications*. Springer Science & Business Media, 2007.

BÄCK, T. – FOUSSETTE, C. – KRAUSE, P. Evolution strategies. In *Contemporary Evolution Strategies*. Springer, 2013. s. 7–45.

BANHARNSAKUN, A. – ACHALAKUL, T. – SIRINAOVAKUL, B. The best-so-far selection in Artificial Bee Colony algorithm. *Applied Soft Computing*. Mar 2011, 11, 2, s. 2888–2901. ISSN 1568-4946.

BARRON, A. et al. Influence of flight time and flight environment on distance communication by dancing honey bees. *Insectes sociaux*. 2005, 52, 4, s. 402–407.

BEYER, H. Evolution strategies. 2007, 2, 8, s. 1965. revision #130731.

BEYER, H.-G. – SCHWEFEL, H.-P. Evolution strategies–A comprehensive introduction. *Natural computing*. 2002, 1, 1, s. 3–52.

BI, X. – WANG, Y. *An improved artificial bee colony algorithm*, 2, s. 174–177. Mar 2011.

BINITHA, S. – SATHYA, S. S. A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering*. 2012, 2, 2, s. 137–151.

BURGER, W. – BURGE, M. J. *Digital image processing: an algorithmic introduction using Java*. Springer Science & Business Media, 2009.

CAMAZINE, S. *Self-organization in biological systems*. Princeton University Press, 2003.

CORDON, O. – HERRERA, F. – STÜTZLE, T. A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends. *Mathware and Soft Computing*. 2002, 9, s. 141–175.

DAMAS, S. – CORDÓN, O. – SANTAMARÍA, J. Medical image registration using evolutionary computation: An experimental survey. *Computational Intelligence Magazine, IEEE*. 2011, 6, 4, s. 26–42.

CASTRO, L. N. – TIMMIS, J. An artificial immune network for multimodal function optimization. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, 1, s. 699–704. IEEE, 2002.

DENEUBOURG, J.-L. et al. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*. Mar 1990, 3, 2, s. 159–168. ISSN 0892-7553, 1572-8889.

DORIGO, M. – OCA, M. A. M. d. – ENGELBRECHT, A. Particle swarm optimization. 2008, 3, 11, s. 1486. revision #91633.

DORIGO, M. – MANIEZZO, V. – COLORNI, A. *Positive Feedback as a Search Strategy*. 1991.

DRASTICH, A. *Tomografické zobrazovací systémy*. Vysoké učení technické v Brně, Fakulta elektrotechniky a informatiky, Ústav biomedicínského inženỳrství, 2004.

EL-ABD, M. Opposition-based artificial bee colony algorithm. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, s. 109–116. ACM, 2011.

FENG, D. D. *Biomedical information technology*. Biomedical Engineering. Academic Press, 2008. ISBN 978-0-12-373583-6.

FISTER JR, I. et al. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*. 2013.

FITZPATRICK, J. M. – HILL, D. L. – MAURER JR, C. R. Image registration. *Handbook of Medical Imaging, volume 1: Physics and Psychophysics*. 2010, 2, s. 447–513.

FLEGR, J. *Úvod do evoluční biologie*. Academia, 2007.

FLOREANO, D. – MATTIUSSI, C. *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press, 2008.

GARCÍA-TORRES, J. M. et al. A case study of innovative population-based algorithms in 3D modeling: Artificial Bee Colony, biogeography-based optimization, harmony search. *Expert Systems with Applications*. 2014, 41, 4, s. 1750–1762.

GAREY, M. R. – JOHNSON, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1990. ISBN 0716710455.

GEEM, Z. W. – KIM, J. H. – LOGANATHAN, G. V. A new heuristic optimization algorithm: Harmony search. *Simulation*. Feb 2001, 76, 2, s. 60–68. ISSN 0037-5497. WOS:000168819000001.

HENDERSON, M. *Genetika - 50 myšlenek, které musíte znát*. Slovart, 2014. ISBN 9788073918248.

HERRERA, F. – LOZANO, M. – VERDEGAY, J. L. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial intelligence review*. 1998, 12, 4, s. 265–319.

HILL, D. L. et al. Medical image registration. *Physics in medicine and biology*. 2001, 46, 3, s. R1.

HUGHES, J. F. – FOLEY, J. D. *Computer graphics: principles and practice*. Pearson Education, 2013.

HUSAKOVA, M. Vybrané aplikace algoritmů umělého imunitního systému. 2010.

HYNEK, J. *Genetické algoritmy a genetické programování*. Grada Publishing as, 2008.

KADLEC, P. – RAIDA, Z. A novel multi-objective self-organizing migrating algorithm. *Radioengineering*. 2011, 20, 4, s. 804–816.

KARABOGA, D. Artificial bee colony algorithm. *Scholarpedia*. 2010, 5, 3, s. 6915.

KARABOGA, D. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

KARABOGA, D. – AKAY, B. A modified artificial bee colony (ABC) algorithm for constrained optimization problems. *Applied Soft Computing*. 2011, 11, 3, s. 3021–3031.

KARABOGA, D. – AKAY, B. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*. 2009, 31, 1-4, s. 61–85.

KARABOGA, D. – BASTURK, B. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In *Foundations of Fuzzy Logic and Soft Computing*. Springer, 2007. s. 789–798.

KIRAN, M. S. – GÜNDÜZ, M. The Analysis of Peculiar Control Parameters of Artificial Bee Colony Algorithm on the Numerical Optimization Problems. *Journal of Computer and Communications*. 2014, 2, 04, s. 127.

KVASNIČKA, V. – POSPÍCHAL, J. – TIŇO, P. *Evolučné algoritmy*. Slovenská technická univerzita, 2000.

LAMPINEN, J. – ZELINKA, I. Mixed integer-discrete-continuous optimization by differential evolution. In *Proceedings of the 5th International Conference on Soft Computing*, s. 71–76, 1999.

LAMPINEN, J. – ZELINKA, I. On stagnation of the differential evolution algorithm. 2000.

LAWLER, E. – LENSTRA, J. – RINNOOY-KAN, A. DB Shmoys eds., The Travelling Salesman Problem. 1985.

LIAO, T. et al. A unified ant colony optimization algorithm for continuous optimization. *European journal of operational research*. 2014, 234, 3, s. 597–609.

MANI, V. – OTHERS. Survey of medical image registration. *Journal of Biomedical Engineering and Technology*. 2013, 1, 2, s. 8–25.

MARLER, R. T. – ARORA, J. S. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*. 2004, 26, 6, s. 369–395.

MARTÍ, R. – LAGUNA, M. – GLOVER, F. Principles of scatter search. *European Journal of Operational Research*. 2006, 169, 2, s. 359–372.

MAYR, E. *Co je evoluce: aktuální pohled na evoluční biologii*. Academia, 2009.

MENZEL, R. et al. Honey bees navigate according to a map-like spatial memory. *Proceedings of the National Academy of Sciences of the United States of America*. 2005, 102, 8, s. 3040–3045.

MEZURA-MONTES, E. – VELÁZQUEZ-REYES, J. – COELLO COELLO, C. A. A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, s. 485–492. ACM, 2006.

O'NEILL, M. – RYAN, C. *Grammatical evolution: evolutionary automatic programming in an arbitrary language*. 4. Springer Science & Business Media, 2003.

ONWUBOLU, G. C. – BABU, B. *New optimization techniques in engineering*. 141. Springer Science & Business Media, 2004.

OSTER, G. F. – WILSON, E. O. *Caste and ecology in the social insects*. Princeton University Press, 1978.

PASSINO, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE*. 2002, 22, 3, s. 52–67.

PHAM, D. et al. The bees algorithm-a novel tool for complex optimisation problems. In *Proceedings of the 2nd virtual international conference on intelligent production machines and systems (IPROMS 2006)*, s. 454–459, 2006.

POLI, R. – KENNEDY, J. – BLACKWELL, T. Particle swarm optimization. *Swarm intelligence*. 2007, 1, 1, s. 33–57.

REICHHOLF-RIEHMOVÁ, H. Hmyz a pavoukovci. *Knižní klub, Praha*. 1997.

RELICHOVÁ, J. *Genetika populací*. Masarykova univerzita, 1. edition, 2009. ISBN 978-80-210-4795-2.

SCHWARZ, J. et al. SODOMA:Self-Organizing Migrating Algorithm in Dynamic Environment. In *12th International Conference on Soft Computing*, s. 163–169. Faculty of Mechanical Engineering BUT, 2006. ISBN 80-214-3195-4.

SEELEY, T. D. *Honeybee democracy*. Princeton University Press, 2010.

SEELEY, T. D. *The wisdom of the hive: the social physiology of honey bee colonies*. Harvard University Press, 2009.

SOERENSEN, K. Metaheuristics-the metaphor exposed. *International Transactions in Operational Research*. Jan 2015, 22, 1, s. 3–18. ISSN 0969-6016.

SONKA, M. – HLAVAC, V. – BOYLE, R. Image processing, analysis and machine vision,(2008). *Thomson Corporation, United State of America*.

STORN, R. – PRICE, K. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*. Dec 1997, 11, 4, s. 341–359. ISSN 0925-5001. WOS:A1997YF22800001.

SZETO, W. – WU, Y. – HO, S. C. An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*. 2011, 215, 1, s. 126–135.

TKADLEC, E. *Populační ekologie: struktura, růst a dynamika populací*. Univerzita Palackého v Olomouci, 2008.

TOWNE, W. F. – MOSCRIP, H. The connection between landscapes and the solar ephemeris in honeybees. *Journal of Experimental Biology*. 2008, 211, 23, s. 3729–3736.

VALSECCHI, A. – DAMAS, S. – SANTAMARIA, J. Evolutionary intensity-based medical image registration: a review. *Current Medical Imaging Reviews*. 2013, 9, 4, s. 283–297.

VALSECCHI, A. et al. Intensitybased image registration using scatter search. *Artificial Intelligence in Medicine*. Mar 2014, 60, 3, s. 151–163. ISSN 0933-3657.

VASANT, P. M. *Meta-heuristics optimization algorithms in engineering, business, economics, and finance*. IGI Global, 2012.

VERMA, B. K. – KUMAR, D. A review on Artificial Bee Colony algorithm. *International Journal of Engineering & Technology*. 2013, 2, 3, s. 175–186.

VESELOVSKÝ, Z. – DUNGEL, J. *Etologie: biologie chování zvířat*. Academia, 2005.

VESELÝ, V. *Včelařství. Vyd. 2., upr. a dopl.* Brázda, 2003.

VOLNÁ, E. Evoluční algoritmy a neuronové sítě. [online], 2012. Dostupné z: <http://www1.osu.cz/ volna/Evolucni_algoritmy_a_neuronove_site.pdf>. [cit. 21.05.2015].

VON FRISCH, K. The dance language and orientation of bees. 1967.

WHITACRE, J. Survival of the flexible: explaining the recent popularity of nature-inspired optimization within a rapidly evolving world. *Computing*. Dec 2011, 93, 2-4, s. 135–146. ISSN 0010485X.

WINTER, S. et al. Registration of CT and Intraoperative 3-D Ultrasound Images of the Spine Using Evolutionary and Gradient-Based Methods. *IEEE Transactions on Evolutionary Computation*. Jun 2008, 12, 3, s. 284–296. ISSN 1089-778X.

XU, Q. et al. A review of opposition-based learning from 2005 to 2012. *Engineering Applications of Artificial Intelligence*. 2014, 29, s. 1–12.

YANG, X. Metaheuristic Optimization. 2011, 6, 8, s. 11472. revision #91488.

YANG, X.-S. *Nature-Inspired Metaheuristic Algorithms: Second Edition*. Luniver Press, 2010a. ISBN 1905986289, 9781905986286.

YANG, X.-S. Multiobjective firefly algorithm for continuous optimization. *Engineering with Computers*. Jan 2012a, 29, 2, s. 175–184. ISSN 0177-0667, 1435-5663.

YANG, X.-S. Firefly algorithm, Levy flights and global optimization. In *Research and Development in Intelligent Systems XXVI*. Springer, 2010b. s. 209–218.

YANG, X.-S. Firefly algorithm, Levy flights and global optimization. In *Research and Development in Intelligent Systems XXVI*. Springer, 2010c. s. 209–218.

YANG, X.-S. Engineering optimizations via nature-inspired virtual bee algorithms. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*. Springer, 2005. s. 317–323.

YANG, X.-S. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010d. s. 65–74.

YANG, X.-S. Firefly algorithms for multimodal optimization. In *Stochastic algorithms: foundations and applications*. Springer, 2009. s. 169–178.

YANG, X.-S. – DEB, S. *Cuckoo Search via Levey Flights*, s. 210–214. Ieee, 2009. WOS:000288686500036. ISBN 978-1-4244-5053-4.

YANG, X. Nature-Inspired Mateheuristic Algorithms: Success and New Challenges. *J Comput Eng Inf Technol 1*. 2012b, 1, s. 2.

ZELINKA, I. et al. *Evoluční výpočetní techniky: principy a aplikace*. BEN-technická literatura, 2008.

ZITOVA, B. – FLUSSER, J. Image registration methods: a survey. *Image and vision computing*. 2003, 21, 11, s. 977–1000.

ZRZAVÝ, J. et al. *Jak se dělá evoluce: od sobeckého genu k rozmanitosti života*. Paseka, Praha, 2004.