Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Adaptivní genetické algoritmy

Nicholas Zahálka

Bakalářská práce

2023

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Nicholas Zahálka**
Osobní číslo:	**I19158**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Adaptivní genetické algoritmy**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem práce bude porovnat adaptivní metody pro učení genetických algoritmů. V teoretické části student popíše existující (případně navrhne vlastní) adaptivní metody pro učení genetických algoritmů (metaučení), v části praktické provede experimenty a vyhodnotí jejich výsledky. Adaptace genetických algoritmů bude realizována pomocí vybraných přístupů (například hybridizace s PSO, algoritmus Bison Seeker apod.). Pro zhodnocení bude hybridní genetický algoritmus aplikován na vybraný standardní optimalizační problém (benchmarkové funkce, TSP, problém N dam apod).

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

HYNEK, Josef. Genetické algoritmy a genetické programování. Praha: Grada, 2008. Průvodce (Grada). ISBN 978-80-247-2695-3.
MITCHELL, Melanie. An introduction to genetic algorithms. Cambridge, Mass.: MIT Press, c1996. ISBN 978-0262133166.
Harik, G., Lobo, F.: A parameter-less genetic algorithm. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99) I (1999) 258?265
GOLDMAN, Brian W. a William F. PUNCH. Parameter-less population pyramid. In: Proceedings of the 2014 conference on Genetic and evolutionary computation – GECCO '14 [online]. New York, New York, USA: ACM Press, 2014, 2014, s. 785-792 [cit. 2018-10-18]. DOI: 10.1145/2576768.2598350. ISBN 9781450326629. Dostupné z: http://dl.acm.org/citation.cfm?doid=2576768.2598350

Vedoucí bakalářské práce: **Ing. Jan Merta**
Katedra softwarových technologií

Datum zadání bakalářské práce: **17. prosince 2021**
Termín odevzdání bakalářské práce: **13. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2022

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 8. 5. 2023

Nicholas Zahálka

PODĚKOVÁNÍ

Rád bych srdečně poděkoval mému vedoucímu bakalářské práce, panu Ing. Janu Mertovi, Ph.D. za odborné vedení, cenné rady a vhodné připomínky, které vedly k tvorbě a kvalitnějšímu zpracování bakalářské práce. Nakonec bych také chtěl poděkovat své rodině a blízkým přátelům za podporu během studia.

ANOTACE

Cílem práce bude porovnat adaptivní metody pro učení genetických algoritmů. V teoretické části student popíše existující (případně navrhne vlastní) adaptivní metody pro učení genetických algoritmů (metaučení), v části praktické provede experimenty a vyhodnotí jejich výsledky. Adaptace genetických algoritmů bude realizována pomocí vybraných přístupů (například hybridizace s PSO, algoritmus Bison Seeker apod.). Pro zhodnocení bude hybridní genetický algoritmus aplikován na vybraný standardní optimalizační problém (benchmarkové funkce, TSP, problém N dam apod).

KLÍČOVÁ SLOVA

Adaptivní metody, Algoritmus, Optimalizační problém, Genetické učení algoritmů, Optimalizace

TITLE

Adaptive genetic algorithms

ANNOTATION

The aim of the work will be to compare adaptive methods for learning genetic algorithms. In the theoretical part, the student will describe existing (or propose his own) adaptive methods for learning genetic algorithms (meta-learning), in the practical part he will perform experiments and evaluate their results. Adaptation of genetic algorithms will be implemented using selected approaches (for example, hybridization with PSO, Bison Seeker algorithm, etc.). For evaluation, the hybrid genetic algorithm will be applied to a selected standard optimization problem (benchmark functions, TSP, N-queens problem etc.).

KEYWORDS

Adaptive methods, Algorithm, Optimization problem, Genetic learning algorithms, Optimization

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

1B	Jednobodové křížení
2B	Dvojbodové křížení
U	Uniformní křížení
GA	Genetický algoritmus
SA	Simulované žíhání
HC	Horolezecký algoritmus
GA+SA	Použití genetického algoritmu a následná aplikace simulovaným žíháním
GA+HC	Použití genetického algoritmu s následná aplikace horolezeckého algoritmu
GA -> SA	Použití optimalizace simulovaným žíháním před selekcí rodiče v genetickém algoritmus
GA -> HC	Použití optimalizace horolezeckého algoritmus před selekcí rodiče v genetickém algoritmu

1 ÚVOD

Cílem této bakalářské práce je porovnání již existujících adaptivních metod pro učení genetický algoritmů. Tyto algoritmy jsou aplikovány na standardní optimalizační problém. Pro provádění samotných experimentů byla zvolena optimalizační úloha Problém obchodního cestujícího. Pro adaptaci genetických algoritmů se v práci využívají techniky nazývané Simulovaní žíhání a Horolezecký algoritmus.

Toto téma bylo zvoleno, protože optimalizační problémy nejsou pouze teoretickými záležitostmi, ale často se s nimi můžeme setkat i v každodenním životě ve všech různých odvětvích a oborech. Na některé optimalizační problémy, jako je například Problém obchodního cestujícího, kterému se věnuje i tato práce, je možné narazit v různých odvětvích jako je například logistika. Příkladem může být určení minimálního počtu kontejnerů, který se řeší v transportních službách. Tyto a další optimalizační problémy se dají řešit, například pomocí genetických algoritmů. Právě těmi se tato práce zabývá, teoretická část práce je věnována seznámení se právě s genetickými algoritmy a optimalizačními metodami, především pak s adaptivními genetickými algoritmy, které vznikají vylepšením genetických algoritmů pomocí adaptivních mechanismů.

V praktické části práce jsou tyto adaptivní genetické algoritmy porovnávány, protože v dnešní době, kdy jsou technologie na vzestupu a progresivně se rozrůstají, zvyšují se tím pádem také požadavky na výkon a efektivitu. A v tento moment přichází právě adaptivní genetické algoritmy. Tyto algoritmy jsou schopné nacházet ideální řešení ve všemožných oblastech. Od umělé inteligence přes strojové učení až po biomedicínu či ekonomii. Praktická část této práce je tedy věnována realizaci optimalizačních metod, konkrétně jsou využity již zmíněný Horolezecký algoritmus a Simulovaní žíhání. Práce má za cíl aplikovat na konkrétní datový soubor různé konfigurace těchto algoritmů a nalézt tak tu možná nejoptimálnější.

2 GENETICKÉ ALGORITMY

2.1 Názvosloví

Na začátku je důležité uvést názvosloví, které bude používáno napříč celou touto bakalářskou prací. Bakalářská práce se zabývá zpracováním genetických algoritmů. Tyto algoritmy nemají pevně danou definici, jak mají být vytvořeny. Nicméně společné prvky pro většinu genetických algoritmů jsou chromozom, geny, jedinci, populace, selekce, křížení, mutace, fitness. [2]

2.1.1 Genetický algoritmus

První genetické algoritmy byly vytvořeny Johnem Hollandem v roce 1960. Navržený algoritmus byl inspirován přirozeným výběrem v přírodě. Přirozený výběr je v přírodě motor, který umožňuje přežít pouze těm nejsilnějším a nejúspěšnějším jedincům, zatímco slabí jsou odsouzeni k záhubě. Jejich geny, které často nemusí vyhovovat prostředí, ve kterém žijí, jsou eliminovány. Tak jako je tomu v přírodě, jedinci spolu svádí souboje, které určují to, kdo bude moci dále pokračovat ve své reprodukci, a kdo naopak ne. Slabší jedinci dostanou také šanci, ti však kvantitativně zaostanou za silnějšími a množství jejich potomků bude značně nižší. Díky genetickým predispozicím budou noví jedinci, při kombinaci vhodných rodičů, mít lepší vlastnosti než jejich rodiče. [1, 2, 17] Genetické algoritmy vychází z evolučních algoritmů. Tento obecný pojem zaštituje nejen genetické algoritmy, ale dále evoluční strategie, evoluční a genetické programování. [1]

2.1.2 Jedinec

Jedinci jsou základní složkou populace, jakýkoliv dostupný jedinec má možnost být potenciálním řešením pro zkoumaný problém. V případě, že se jedná opravdu o konečné řešení, které splňuje veškeré podmínky, lze ho poté označit za kandidátní řešení. Každý z jedinec je složen z chromozomů, které reprezentují jeho vlastnosti a ze své fitness hodnoty, kterou je hodnocen. Jedinci, kteří jsou vybíráni pro další křížení a tvoření tak nové populace jsou vybíráni náhodně, ovšem jedinci s lepší fitness hodnotou mají vyšší šanci na to být vybráni. [1]

2.1.3 Chromozomy

Základním kamenem genetických algoritmů jsou chromozomy, které jsou složené z genů. Tyto jednotlivé geny je možné si představit jako informaci – vlastnost, kterou jedince lze charakterizovat. Samotný chromozom, který je složený z jednotlivých genů určitým způsobem zakódovává tyto jednotlivé geny a vytváří tak celek. Chromozomy jsou tedy využívány k popisování jednotlivců v populaci, nejčastější reprezentace chromozomů bývá pomocí bitových řetězců, celých čísel anebo reálných čísel. Bitová reprezentace umožňuje všem reprezentovaným chromozomům nastavovat dva dostupné stavy hodnot 0 a 1. Reprezentace pomocí celých a reálných čísel může pro jednotlivé části chromozomu používat hodnot více. [1]
Při řešení *Problému obchodního cestujícího* je cesta reprezentována pomocí jednotkových řetězců, které reprezentují pořadí, v jakém jsou města procházená. [1, 18]

2.1.4 Reprezentace

Reprezentace je jedna z klíčových věcí, kterou je nutné si důkladně promyslet a zvolit při návrhu. Nevhodná reprezentace může přinést znatelné zpomalení při řešení problému a do velké míry ovlivnit získané výsledky. Neexistuje jedna specifická reprezentace pro všechny řešené problémy, pro každý řešený problém je nutné zvolit specifickou reprezentaci individuálně. Existuje více typů reprezentací například binární, jednotkové, stromové či permutační kódování. Nejčastěji používané je binární a permutační kódování. Binárně kódovaný jedinec může ve svých genech nabývat pouze dvou stavů. Permutačně kódovaný jedinec obsahuje permutaci několika čísel, které popisují, v jakém pořadí má řešení poskládané samotné objekty. [20, 2, 1]

2.1.5 Fitness funkce

Fitness funkce je využívána pro získání hodnoty určující kvalitu jedince – kvalitu kandidátního řešení. Zpracování fitness funkce je zásadní pro správné fungování genetického algoritmu. Správná funkce je snadno pochopitelná, rychle prováděná, ovšem je nutné, aby byla také efektivní. Výpočet této funkce tedy slouží k získání účinnosti potenciálně nalezeného řešení. [15, 18]

2.1.6 Populace

Populace obsahuje libovolné množství jedinců, kteří reprezentují jedno z možných řešení problému, kterým se při řešení zabýváme. Pro optimální řešení problémů je nutné zvolit vhodnou velikost populace, i když je možné vybrat libovolný počet jedinců. Málo jedinců nemusí být vůbec schopno vygenerovat potřebné množství kombinací genů pro dobré řešení. Příliš mnoho jedinců zas může způsobit, že se v populaci díky malému tlaku na kvalitu budou držet jedinci se špatnými geny a konvergence algoritmu se zpomalí. Nejsou žádná pevně vytyčená pravidla pro to, jakou velikost má vhodná populace, proto bývá volena vždy specifická populace pro každý řešený problém. [1, 19]

2.1.7 Selekce

Selekce musí co nejvíce napodobovat přirozený výběr v přírodě. Každou další novou populací se snažíme o zlepšení té stávající. Pro samotné zlepšení je vždy nutné vybírat nejsilnější jedince, kteří mají největší pravděpodobnost, že předají své pozitivní vlastnosti na své potomky. Nelze ovšem vybírat do nové generace pouze nejsilnější jedince, je nutné také volit ty slabší pro rozmanitost našich výsledků a lepší autenticitu napodobení přírodního výběru. V případě, že jsou voleni do nové populace pouze nejsilnější jedinci, mohl by algoritmusuvíznout v lokálním optimu. Samozřejmě také v opačném případě, pokud jsou volení převážně slabší jedinci, mohl by nastat problém s příliš pomalým vývojem. Při špatné selekci tedy může dojít k tomu, že výsledky budou příliš rychle konvergovat směrem od globálního optima nebo bude konvergence ke globálnímu optimu značně zpomalená. Známe také velké množství dostupných typů křížení. Nejčastěji používaný typ křížení je například turnajový výběr, ruletové kolo nebo range selekce [1, 2]

Turnajový výběr

Pro možnost aplikování tohoto výběru je nutné, aby bylo dostupné k jedinců, kde $k \geq 2$. Následně se zvolí k jedinců, kteří vstupují do turnaje. Vybraní jedinci následně vstupují do pomyslného turnaje, kde se vzájemně utkají jako v přírodě, kde jsou úspěšní nejsilnější jedinci. Jejich utkání je v podobě porovnání jejich kvality – vyhodnocené fitness funkce, kde je jedinec s nejvyšším skórem zvolen jako výherce, který bude zvolen pro křížení a vytváření nových potomků. [1, 6]

Ruletové poměrové kolo

Tato metoda byla převzata již z názvu vycházející rulety. Metoda všem jednotlivcům určí poměrový díl z kola, který je určen podle jejich hodnoty fitness funkce. Jedinci s vyšší hodnotou fitness funkce budou mít větší výseč, a tudíž i větší šanci na vybrání než jedinci s nižší hodnotou. Pro aplikování tohoto algoritmu je nutné mít n jedinců, kde $n \geq 2$. [1, 19]

Selekci lze interpretovat více různými způsoby. První je pomocí zatočení pomyslným ruletovým kolem. Na počátku je kolo rozdělené na příslušné díly, které jsou přímo úměrné hodnotám z fitness funkce pro každého jedince. [1] Po zatočení se roztočí kulička a zastaví se u náhodné výseče, která určí jedince pro další křížení. Zjednodušená implementace se skládá z vygenerování náhodného čísla, podle kterého se zjistí příslušná výseč patřící konkrétnímu jedinci a je selekcí vybrán. [19]

2.2 Genetické operátory

Pouhá selekce však nestačí pro vytvoření potomků, je také nutné použít genetické operátory, které jsou dostupné i v naturální evoluci. Při křížení se kombinují nejen pozitivní vlastnosti obou rodičů, ale také negativní. Dále také velkou roli hraje mutace, která náhodně pozměňuje některé vlastnosti jedince, která může nastat. Tyto dva faktory výběr genetických operátorů může značně ovlivnit, zda výslední potomci budou průměrně lepší nebo horší než jeho rodiče. [20]

2.2.1 Křížení

Je dostupné velké množství křížení. Při křížení budou zaměněné jednotlivé geny mezi dvěma zvolenými rodiči do nově vzniklých potomků. Existuje množství různých křížení, přičemž nejjednodušší křížení je jednobodové. U tohoto typu se zvolí náhodně bod *n,* od kterého si zvolení rodiče vzájemně vymění geny. Při vkládání jednotlivých genů do prvního potomka se vloží příslušné hodnoty z prvního rodiče a zbytek hodnot je poté vložen z druhého rodiče. Obdobně tomu je při vkládání genů do druhého potomka, kde se opět nejprve vloží příslušné hodnoty z druhého rodiče a zbytek je doplněn z prvního.

Zobecněním je *k*-bodové křížení, kde je možné si zvolit libovolný počet křížících bodů. [1] To následně určí, jaké skupiny genů budou mezi oběma rodiči vyměněny z příslušného rodiče budou vybráni a aplikováni do následného křížení.

Existuje také uniformní křížení, při kterém se u každého chromozomu rozhoduje, zda bude zvolen z prvního rodiče nebo z druhého. [1, 19]

2.2.2 Mutace

Mutace zvolí náhodný libovolný počet genů v jedinci a ty se následně zamění na jiné náhodné hodnoty. Mutaci lze aplikovat na kompletního jedince, kde se následně zamění jeden člen s určitou pravděpodobností, nebo tuto mutaci lze aplikovat na každý příslušný gen v jedinci. Opět existuje množství různých mutací, každá z nich se více hodí pro specifiky reprezentované prvky. První zmíněnou mutací je swap mutace, kde se zvolí jeden náhodný gen, který bude zaměněný s jiným náhodným genem. Tato mutace je využívaná u permutační reprezentace. Druhým typem je scramble mutace, v tomto případě se bere náhodné množství chromozomů a vytvoří se podseznam genů, které mezi sebou mají být náhodně promíchány a vrátí se zpět do původního jedince, tato mutace je taktéž vhodná pro permutační reprezentaci. Poslední zmíněnou variantou je reverse mutace, kdy se zvolí náhodné množství genů, které se obrátí v daném jedinci, tato mutace je často používaná u binární reprezentace. [16, 2, 8]

2.2.3 Elitismus

Jde o rozšíření selekce. Toto rozšíření přenáší do nové populace vždy nejlepší jedince z té předchozí. Elitismus je používán z důvodu, že tento nejlepší jedinec se nemusí vždy dostat do nové populace, kvůli změně genů při použití genetických operátorů. Velké množství studií prokázalo, že použití elitismu přináší pozitivní výsledky. [2, 17]

3 ADAPTIVNÍ METODY PRO METAUČENÍ

3.1 Adaptivní genetické algoritmy

Na začátku je vhodné si vysvětlit, co samotný termín adaptace znamená. Jedná se o schopnost přizpůsobení organismu vůči okolním jevům a vnějšímu prostředí. [28]

Adaptivní genetické algoritmy jsou variací genetických algoritmů, pokouší se nalézt lepší řešení pomocí adaptování samotných jedinců pomocí různých technik. [21]

Využívají také různé metody pro adaptaci, například Simulované žíhání, Horolezecký algoritmus, Optimalizace roje částic, Optimalizace včelím rojem, Algoritmus světlušek, Gradientní algoritmus, Netopýří algoritmus a mnohé další pro nalezení optimálního řešení. [1, 9]

3.2 Hybridní genetické algoritmy

Hybridní genetické algoritmy vznikají kvůli genetickým algoritmům. Samostatné genetické algoritmy mohou přinášet uspokojivé výsledky, nicméně nemusí být aplikovatelné jako obecný postup pro řešení všech problémů. Při samotné hybridizaci jsou využité nejefektivnější částí z tradičních algoritmů. Samotný návrh hybridního genetického algoritmu je poměrně složitý, protože je důležité znát podrobně princip tradičních algoritmů, ale i konkrétního problému, na který se algoritmus aplikuje. [1, 11]

Existují různé druhy hybridizace. Sekvenční hybridizace je nejčastěji používaný typ hybridizace. Aplikace této hybridizace je snadná, používají se různé metody postupně za sebou. Tedy výsledek předchozí metody je výchozí řešení pro použití další metody. [11] V praktickém zpracování této bakalářské práci je testován i tento sekvenční postup. Na počáteční populaci je aplikován genetický algoritmus, a poté je nejsilnější jedinec použitý jako výchozí řešení pro různé optimalizační metody.

3.2.1 Synchronní paralelní hybridizace

Na rozdíl od zmíněné sekvenční hybridizace, synchronní je komplexnější. Tato hybridizace se snaží kombinovat lokální vyhledávání společně s globálním pro zlepšení výsledků. [11] Používá původní navržený postup a aplikuje vyhledávací algoritmus jako jeden z možných operátorů. Tento vyhledávací algoritmus může být například simulované žíhání, horolezecký algoritmus apod. Algoritmus bude považovat jedince za počáteční řešení a začne vyhledávat lepší možné řešení. Poté počátečního jedince nahradí novým zlepšeným. Tento postup může být brán také jako učení jedince během jeho života. [22]

3.2.2 Asynchronní paralelní hybridizace

Tato hybridizace využívá několik algoritmů, které vyhledávají celek případně část prostoru možných řešení. Pomocí spolupráce zkouší nalézt optimum. Díky používání několika různých algoritmů máme zajištěné nejméně zlepšení alespoň jako jeden samotný algoritmus. V mnoha případech ovšem, ale dojde k většímu zlepšení, protože poskytované informace mezi jednotlivými algoritmy mohou zrychlit hledání. [22]

Dostupné jsou dva základní modely učení během života jedince, a to Lamarckovský a Baldwinovský. Není nutné používat pouze jeden z těchto dvou modelů, lze kombinovat užitečné vlastnosti obou modelů a tím získat nové efektivnější řešení problému. [26]

Příkladem může být například Pyramidový genetický algoritmus bez parametrů. Pyramidový genetický algoritmus bez parametrů patří z rodiny evolučních algoritmů, které byly představeny Brianem W. Goldmanem a Williamem F. Punchem v roce 2014. [4] Běžné pyramidové algoritmy požadují vstupní parametry. Z toho důvodu mají tyto algoritmy vždy velmi specifické nastavení a nelze je tak navrhnout obecně pro daný problém. Bez parametrický pyramidový algoritmus oproti svým předchůdcům disponuje možností nepoužívat žádné vstupní parametry, aniž by díky tomu utrpěl výkonnostní úbytek.

Běžné genetické algoritmy obsahují jednu specifickou populaci s dostupnými řešeními, pyramidové algoritmy mohou ovšem obsahovat i vícero populací. Bez parametrický pyramidový algoritmus ukládá svá data do pyramidové struktury. Nepracuje tedy s jednou konkrétní populací, pro každé patro pyramidy má svoji vlastní. Při postupu skrze jednotlivá patra je možné vidět, že při stoupání do vyšších pater této pyramidy jsou dostupná více optimalizovaná řešení. Veškerá dostupná řešení jsou vždy unikátní, toho je docíleno pomocí tabulky, která uchovává hashe všech řešení. Pro urychlení nalezení optimálního řešení algoritmus může také využívat

různé optimalizační metody nebo genetické operátory pro zlepšení vygenerovaných členů. Nejčastěji bývá kombinován s horolezeckým algoritmem. [25, 30, 3]

Lamarckovské učení

Základní myšlenka je, že jedinec může dědit dále vlastnosti, které se naučil a získal za svého života. Potomci tedy mohou mít již od narození tyto zděděné vlastnosti. Učení používá lokální vyhledávací algoritmus, je zde použitý jako rozšiřující genetický operátor. Po aplikaci lokálního vyhledávacího algoritmu je upravený jedinec navrácen zpět do populace. Rozšířený genetický operátor může vyhledávání optimálního řešení značně zrychlit, nicméně může to také narušit strukturu jedince. Struktura může být narušena natolik, že jedinec se nebude schopný se dále vyvíjet, dosáhne tak svého maxima předčasně a bude genetický algoritmus díky ztratí evoluční charakter a bude předčasně konvergovat. [26]

Baldwinovo učení

Používá lokální vyhledávací algoritmus obdobně jako Lamarckovské učení, ale aplikuje ho jiným přístupem. Zásadní rozdíl je, že nelze předávat nabyté vlastnosti rodičů na své potomky. Lamarckovské učení upravuje samotnou genetickou strukturu Baldwinovo učení pracuje jiným způsobem. Učení neupravuje genetické informace jedince pouze ovlivní prostředí hodnotu fitness funkce a tím i šanci projít selekcí. Pomáhá jedincům, kteří jsou blízko dobré kombinaci genů, aby nevypadli během selekce. [26]

4 OPTIMALIZAČNÍ METODY

4.1 Horolezecký algoritmus

Horolezecký algoritmus patří mezi lokální optimalizační algoritmy. Patři rovněž mezi základní algoritmy, který jsou velmi populární právě díky svému snadnému principu i následné implementaci. Hlavní myšlenka tohoto algoritmu je ze stávajícího řešení prohledat veškerá dostupná okolní řešení tedy lokálně vyhledat prostor možností a poté postupovat do nalezeného řešení, pouze pokud přinese zlepšení. Po přemístění do nového řešení se tento postup provádí znovu. V momentě, kdy algoritmus není schopen nalézt další řešení je ukončen. Předností tohoto algoritmu je jeho rychlost, ale jeho velkou nevýhodou je možnost uvíznutí v lokálním optimu. [23] Tento algoritmus může být nalezený také pod názvem hladový algoritmus. Samotný název popisuje jeho princip, kdy algoritmus bezhlavě pojídá jakákoliv lepší řešení. Prakticky to tedy znamená, že rovnou přijímá jakékoliv lepší řešení, než má nyní bez toho, aby zjistil, zda v tomto řešení nemůže uvíznout a skončit. [24, 7]

Je dostupné velké množství variací tohoto algoritmu, několik základních a velmi výkonných bude zmíněno se zkráceným popisem principu.

Prvním zmíněným je Prostý horolezecký algoritmus. Tento algoritmus se podívá na své okolní sousedy a postupuje do prvního, který má lepší výsledky než on sám. Porovná svoji aktuální hodnotu fitness funkce s nalezeným sousedem a toto opakuje do té doby, dokud nedojde do fáze, kdy už nemá žádné lepší sousedy, do kterých by mohl pokračovat. Tento přístup postupu u tohoto algoritmu může zapříčinit uvíznutí pouze v lokálním optimu, protože algoritmus se nedívá dále než na své okolní sousedy. [29]

Stochastický horolezecký algoritmus vybírá oproti svému předchůdci prostému algoritmu náhodné zlepšující řešení. Pravděpodobnost výběru může být ovlivněna dle strmosti, do které stoupá. Tato variace algoritmu obvykle konverguje pomaleji čímž snižuje riziko uvíznutí v lokálním optimu. [24]

Algoritmus stoupání po nejstrmější cestě je variací prostého horolezeckého algoritmu. Rozdíl oproti prostému je, že neprohledává pouze nejbližší řešení ale veškerá dostupná řešení. Je tedy mírně snížená šance, na uvíznutí v lokálním minimu, protože prohledává větší množství možností. [24]

Poslední zmíněný typ je algoritmus náhodného restartování horolezce. Tento algoritmus provádí řadu vyhledávání z náhodně vygenerovaných pozic, do té doby, dokud není nalezený požadovaný cíl. [24]

4.2 Simulované žíhání

Simulované žíhání je o jedna z nejpoužívanější optimalizačních metod. „*Jde o pravděpodobnostní metodu, navrženou Kirkpatrickem, Gelettem a Vecchim (1983) a Černým (1985) pro nalezení globálního minima nákladní funkce, která může mít několik lokálních minim*“. [13] Samotný princip simulovaného žíhání je takový, že na počátku, když je teplota vyšší je žíhání ochotné více riskovat a nalézá tak i špatná řešení. Díky tomuto riskování předchází možnému nalezení pouze lokálního optima v prohledávaném prostoru možných řešení. Každým krokem se postupně ochlazování zpomaluje až narazí na svojí konečnou teplotu, ve které přestane a zastaví ochlazování. Tato konečná teplota je ve většině případech zvolena na 0 stupňů. Není to však nutnou podmínkou, mohou být zvolené i jiné podmínky, které zastaví ochlazování v jiných případech. Například může být nastavený fixní počet kroků, kolikrát žíhání má být provedeno, nebo může být zastaveno, pokud bude příliš dlouho stagnovat v nalezené hodnotě.

Prakticky optimalizační metoda má na počátku svojí počáteční teplotu zahřátou na velmi vysokou. Vysoká počáteční teplota má zde zásadní funkci, díky ní může samotný algoritmus riskovat a přijímat i dočasně horší řešení, než je jeho stávající. Přijmutí horších řešení je jen dočasné a algoritmus se díky tomu dokáže úspěšně vyhnout konvergenci k lokálnímu optimu. Tato optimalizační metoda tedy vystihuje nedostatky, které přináší zmíněný předchozí horolezecký algoritmus.

U samotného algoritmu je poměrně obtížné najít vhodnou počáteční teplotu, ze který by žíhání mělo začínat. Nejčastěji tedy bývá aplikováno začínat na velmi vysoké teplotě, která se bude následně rapidně ochlazovat až do té doby, než budou nejméně vhodná řešení nalezena a vyřazena. Ochlazování poté zpomalí a má díky tomu vyšší potenciál nalézt vhodné řešení či kandidátní řešení. S tímto konceptem přišel Rayward-Smith v roce 1996. [12]

V počátečním kroku je zvoleno náhodné řešení, které se v každém kroku před ochlazením Zduplikuje a je na něj použitý genetický operátor – mutace. Následně proběhne porovnání stávajícího řešení a nově modifikovaného, které automaticky nahradí stávající řešení za nové, pokud je lepší, v opačném případě dojde k výpočtu pravděpodobnosti přijetí kritéria. [12]

$$P(\Delta E) = \exp\left(\frac{(\Delta E_i - \Delta E_j)}{k_B T}\right)$$

Uvedený vzorec popisuje pravděpodobnost na přijetí kritéria. Samotný vzorec slouží pro výpočet pravděpodobnosti pro přijetí horšího řešení. v tomto případě ΔE_i je současná energie (stávající řešení) ΔE_j je nová energie (nové řešení), T je aktuální teplota a k_B je Boltzmannova konstanta. [23]

5 PRAKTICKÁ REALIZACE OPTIMALIZAČNÍCH METOD

5.1 Horolezeckého algoritmu

Praktická implementace byla provedena na prostém horolezeckém algoritmu. Na počátku implementace byl aplikován genetický algoritmus, který po svém dokončení předal nejlepšího jedince napříč celou populací. Následně toto řešení bylo duplikováno a byl na něj aplikován genetický operátor – mutace. Tato použitá mutace slouží jako simulace kroku, který pomyslně provádí horolezec. Po použití genetického operátoru je nové řešení porovnáno s původním. v případě, že nalezené řešení je lepší, než stávající tak nové řešení nahradí původní v opačném případě zůstává původní a nic se nemění. Tato smyčka běží do té doby, dokud není naplněný počet zvolených opakování. Vhodný počet opakování k nalezení nejlepší cesty je vyzkoušení každé možné cesty. Ten lze odhadnout pomocí faktoriálu z počtu jednotlivých měst, které mají být navštívené. Tento způsobem je vhodný pro malý počet měst ovšem při řešení rozsáhlého počtu měst není možné v polynomiálně omezeném čase toto vykonat. Je tedy nutné zvolit takový počet kroků, který bude co nejvíce vyhovovat požadovanému řešení nebo se mu bude co nejvíc přibližovat.

5.2 Simulovaného žíhání

Praktická implementace samotného algoritmu v práci je prováděna v následujících krocích. Metoda na počátku považuje své aktuální řešení za kandidátní, kov v našem případě řešení je pomyslně zahřáté na předem definovanou maximální teplotu. Algoritmus následně toto řešení duplikuje a použije na něj genetický operátor mutace. Po úpravě duplikovaného řešení je poté zjištěno pomocí výpočtu pravděpodobnosti zjištěno, zda modifikované řešení nahradí stávající nebo ne.

Pravděpodobnost je vypočítána porovnáním fitness hodnot původního řešení a modifikovaného. Pokud má modifikované řešení nižší fitness hodnotu, než dosavadní je zvolena 100 % pravděpodobnost, že bude nahrazeno za aktuální řešení. V případě, že modifikované řešení je horší, než stávající je proveden výše zmíněný vzorec pro výpočet pravděpodobnosti. Výsledek se porovná s náhodně vygenerovaným číslem v rozsahu 0 až 1, dle porovnání těchto dvou hodnot je změněné aktuální řešení. Pokračuje se dále k porovnání aktuálního řešení s nalezeným kandidátním. Toto řešení je nahrazeno za aktuální, pokud nové řešení má nižší fitness hodnotu, následně dojde k ochlazení. Tyto kroky jsou opakovány do té doby, dokud nedojde k ochlazení

kovu na předem definovanou teplotu, nebo není splněna jiná specifická podmínka pro zastavení cyklu, v testovaném případu proběhnutí specifického počtu kroků.

6 VYHODNOCENÍ EXPERIMENTŮ

6.1 Popis vybraného optimalizačního problému

Optimalizační problém, který je zkoumán a na kterém jsou prováděny experimenty je *Problém obchodního cestujícího.* Tento problém patři všeobecně mezi neprozkoumanější optimalizační problémy. Samotné první zmínky o tomto problému jsou již dostupné v roce 1759 od švýcarského matematika Leonharda Eulera. Optimalizační problém je běžně dostupný i v dnešním světě hlavně například v logistice. Problém obchodního cestujícího je jedním ze zástupců optimalizačních problému, který spadá do kategorie NP-těžkých problémů. [1, 5]

Z důvodu vysoké matematické náročnosti je však nepravděpodobné, že se podaří najít algoritmus, který bude schopný nalézt řešení v polynomiálně omezeném čase. Kvůli tomu, že se jedná o NP-těžký problém, používají se heuristické algoritmy, to tedy přináší přibližné řešení problému. Nicméně toto řešení není garantované jako globálně optimální, tyto řešení bývají často velmi blízko k optimálnímu řešení. [1]

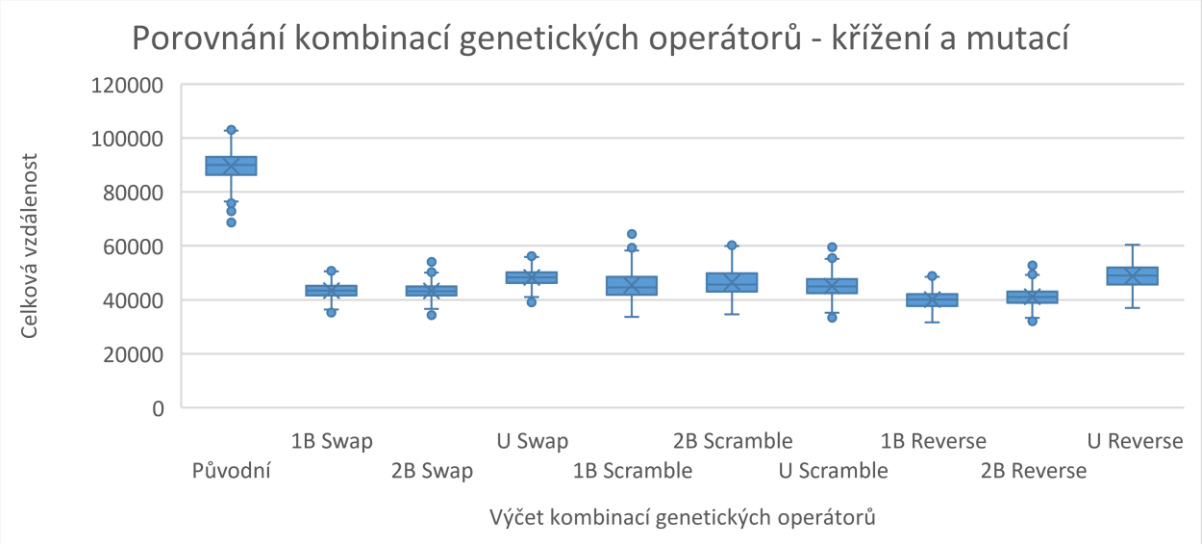
Problém je definován následovně: „*Problém obchodního cestujícího lze snadno slovně definovat tak, že obchodní cestující musí při své cestě navštívit každé město v přidělené oblasti právě jedenkrát a vrátit se do výchozího bodu. Kromě toho je třeba minimalizovat jeho cestovní náklady, které jsou definovány jako celková cena postupného přesunu mezi jednotlivými městy a obvykle závisí na vzdálenosti mezi městy. Cílem je tedy navrhnout takovou okružní cestu, kde součet nákladů nutných pro uskutečnění této cesty bude minimální*.“ [10]

Samotné testovaní probíhalo pomocí datové sady s 29 zastávkami. Datový set je dostupný z webové stránky. [27] Datové sety obsahují souřadnice *x* a *y*.

6.2 Konfigurace algoritmů

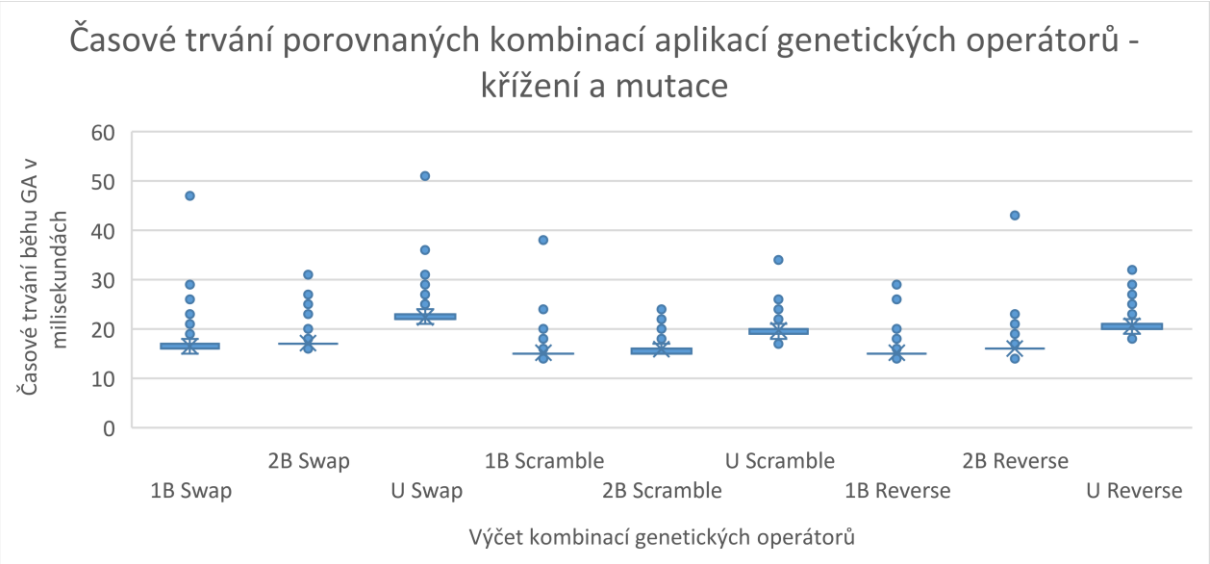
Při vytváření algoritmu bylo nutné zvolit vhodnou kombinaci genetických operátorů, které a jak se budou vzájemně kombinovat. Bylo nutné provést testování pouze samotného genetického algoritmu, poté kombinaci optimalizačních metod tedy simulovaného žíhání a horolezeckého algoritmu s dostupnými genetickými operátory. Získané výsledky následně byly zpracovány pro zjištění, jaká kombinace bude nejvhodnější na provádění experimentů.

Genetický algoritmus prováděl kombinace operátoru křížení s následným aplikováním operátoru mutace. Vzniklo tak několik kombinací. První kombinací bylo jednobodové křížení s použitím swap, scramble a reverse mutace. Druhá kombinace byla dvojbodové a třetí uniformní křížení se stejnými možnostmi mutace jako první jednobodové křížení. Všechny tyto kombinace byly testovány na 200 generacích při 1000 pokusech.



Obrázek 1 – Graf porovnání výsledné délky při testování dostupných kombinací genetických operátorů na genetickém algoritmu

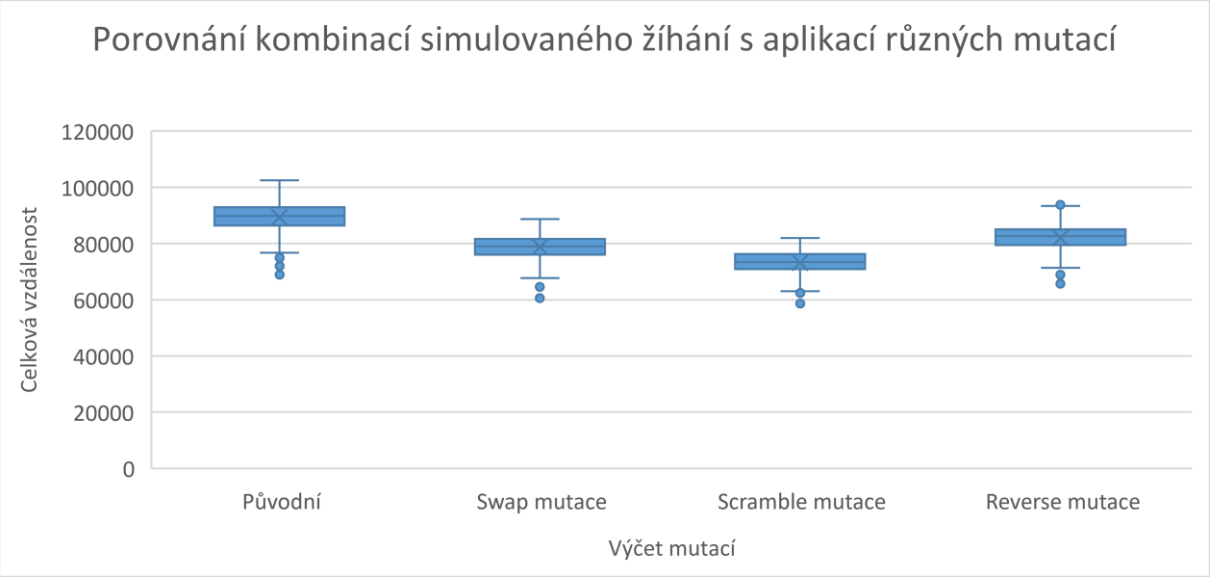
Výsledky je možné vidět na obrázku 1. V tomto případě kombinace vycházely velmi podobně, ale nejlepší při porovnání nalezených řešení bylo jednobodové a dvojbodové křížení při použití reverse mutace. Výsledky byly také porovnány nejen z pohledu nejlepších výsledků, ale také z pohledu časové náročnosti. Porovnání těchto výsledků je možné vidět na obrázku 2, zde patří jednobodové křížení mezi nejrychlejší nehledě na typ testované mutace. Pro další testování byla tedy zvolena kombinace jednobodového křížení a reverse mutace.



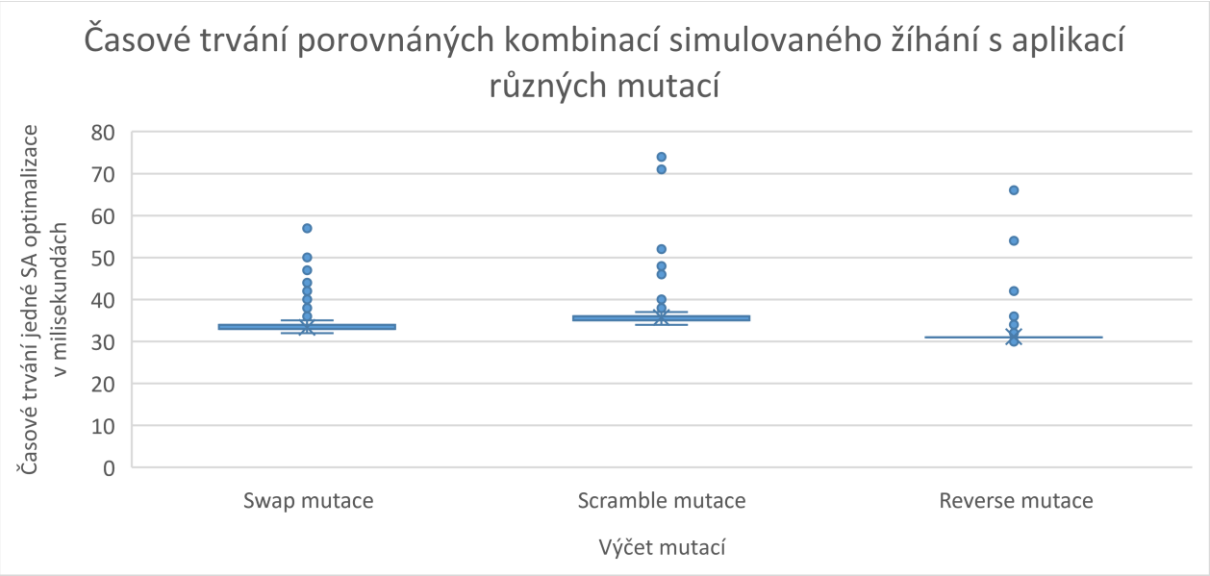
Obrázek 2 – Graf porovnání časového trvání při testování dostupných kombinací genetických operátorů na genetickém algoritmu

Konfigurace simulovaného žíhání

Pro zvolení vhodné mutace, která bude aplikována na simulované žíhání byl proveden experiment, ve kterém byly porovnány výsledky po aplikaci různých mutací. Experiment byl proveden na 1000 krocích. Při porovnání těchto kroků nejlépe optimalizovala z pohledu optimální délky scramble mutace, jak je možné vidět v grafu na obrázku 3. Důležité zde bylo také porovnání časového trvání, v tomto případě vycházely všechny kombinace s rozdíly milisekund, jak je možné vidět v grafu na obrázku 4. V tomto ohledu to tedy nehraje příliš velkou roli. Pro další testování byla tedy zvolena mutace scramble u simulovaného žíhání.



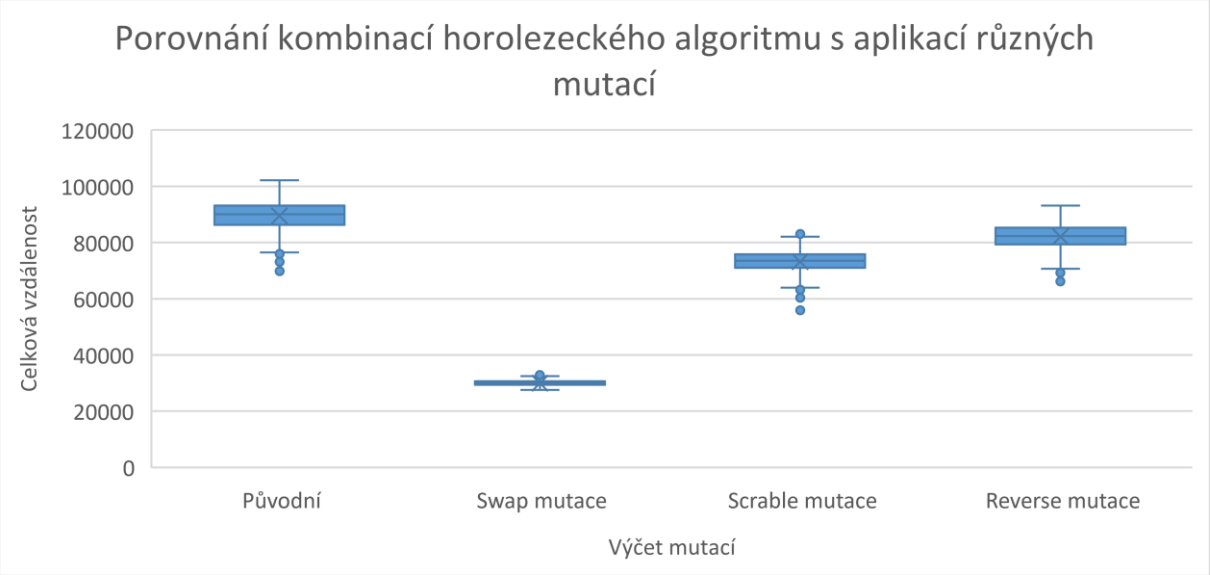
Obrázek 3 – Graf porovnání aplikací mutace na simulované žíhání



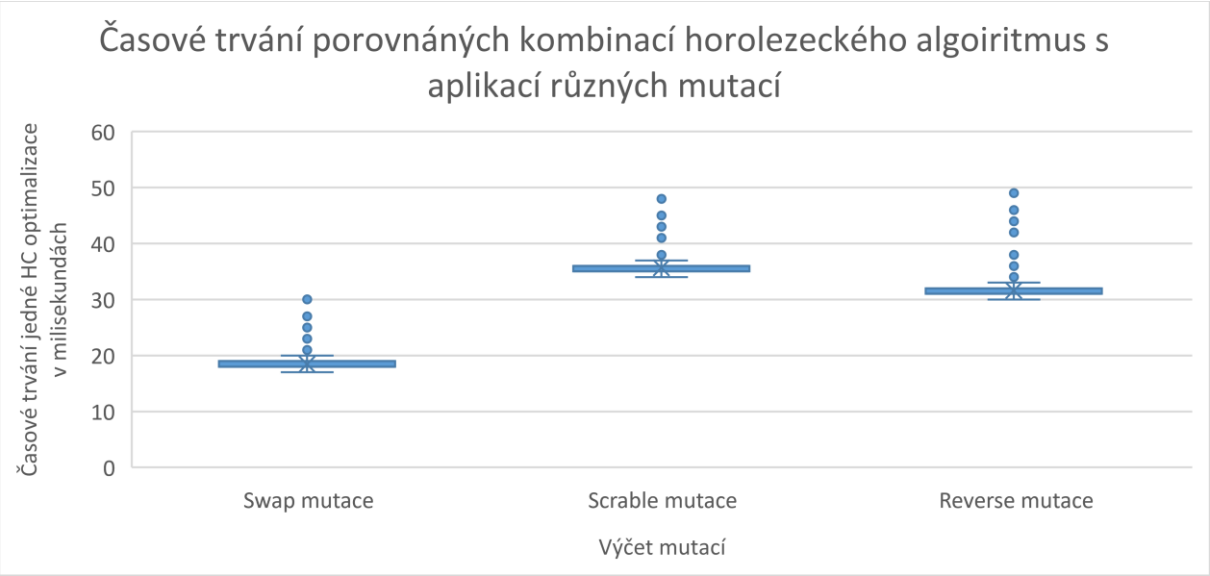
Obrázek 4 – Graf časového porovnání aplikací mutace na simulované žíhání

Konfigurace horolezeckého algoritmu

Pro zvolení vhodné mutace, která bude aplikována na horolezecký algoritmus byl provedený experiment, ve kterém byly porovnány výsledky po aplikaci různých mutací. Experiment byl proveden na 1000 krocích. Při porovnání těchto kroků nejlépe optimalizovala z pohledu minimální délky swap mutace, jak je možné vidět v grafu na obrázku 5. Testováno bylo také porovnání z časového hlediska trvání, zde vycházela nejlépe swap mutace. Porovnané výsledky měly mezi sebou rozdíly v rámci milisekund, jak je možné vidět v grafu na obrázku 6. Pro další testování byla tedy zvolena mutace swap u horolezeckého algoritmu.



Obrázek 5 – Graf porovnáných aplikací mutace na horolezecký algoritmus



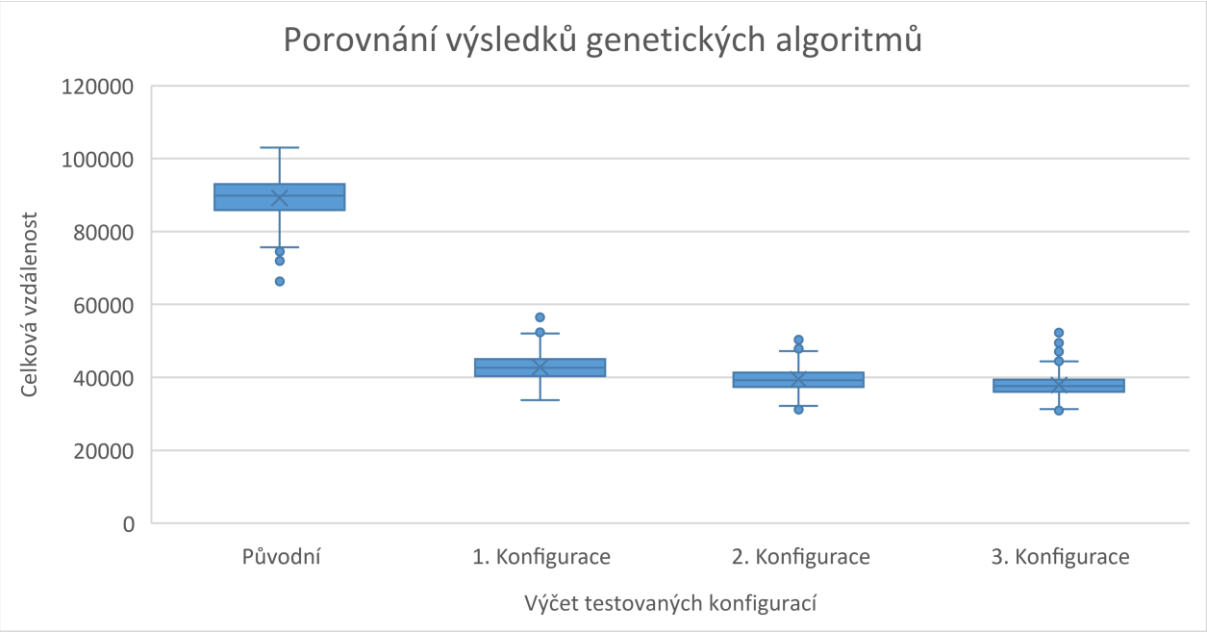
Obrázek 6 – Graf časového porovnání aplikace mutací na horolezecký algoritmus

Krátké zhodnocení nalezených výsledků u jednotlivých konfigurací u testovaných algoritmů je následovné. V případě použití genetického algoritmu je nejvhodnější volba kombinace jednobodového křížení s aplikováním reverse mutace. Tato kombinace vychází nejlépe z pohledu nalezení nejlepšího řešení, z pohledu časového řešení jsou zde rozdíly v milisekundách.
Konfigurace simulovaného žíhání přinesla velmi podobné výsledky, ale největší zlepšení přinesla scramble mutace, při porovnání časové náročnosti jsou porovnávané hodnoty rozdílné opět v rámci milisekund. Konfigurace horolezeckého algoritmu ukázala, že nejefektivnější z pohledu nalezeného řešení, ale i časového je swap mutace, která nalezeným řešením velmi předčila ostatní mutace.

6.3 Výsledky Genetického algoritmu

Testování genetického algoritmu probíhalo pomocí používání genetického algoritmu pouze s aplikováním genetických operátorů křížení a mutace, které byly zvoleny v předchozí konfiguraci algoritmu. Používaný data set pro testovaní obsahoval 29 měst a každé testování bylo provedeno na 1000 pokusech pro větší škálu hodnot.

První konfigurace testovala 100 evolucí v GA, přinesla průměrné zlepšení o 52 % proti počáteční hodnotě. Druhá konfigurace testovala 250 evolucí v GA, přinesla průměrné zlepšení o 56 % proti počáteční hodnotě. Třetí konfigurace testovala 500 evolucí v GA, přinesla průměrné zlepšení o 57 % proti počáteční hodnotě. Veškeré získané výsledky jsou dostupné v grafu na obrázku 7.



Obrázek 7 – Graf porovnávaných výsledků u genetického algoritmu

Při krátkém zhodnocení celkového porovnání nalezených výsledků z těchto testovaných konfigurací je možné vidět, že samotný genetický algoritmus přináší dobré výsledky proti původnímu řešení. Je možné říct, že vyšší počet evolucí bude úspěšně nacházet lepší řešení do té doby, dokud budou dostupná, ale s vyšším počtem začne být poměrně neefektivní, protože zlepšení bude v minimálních mezích, zatímco počty generací mohou velmi narůstat.

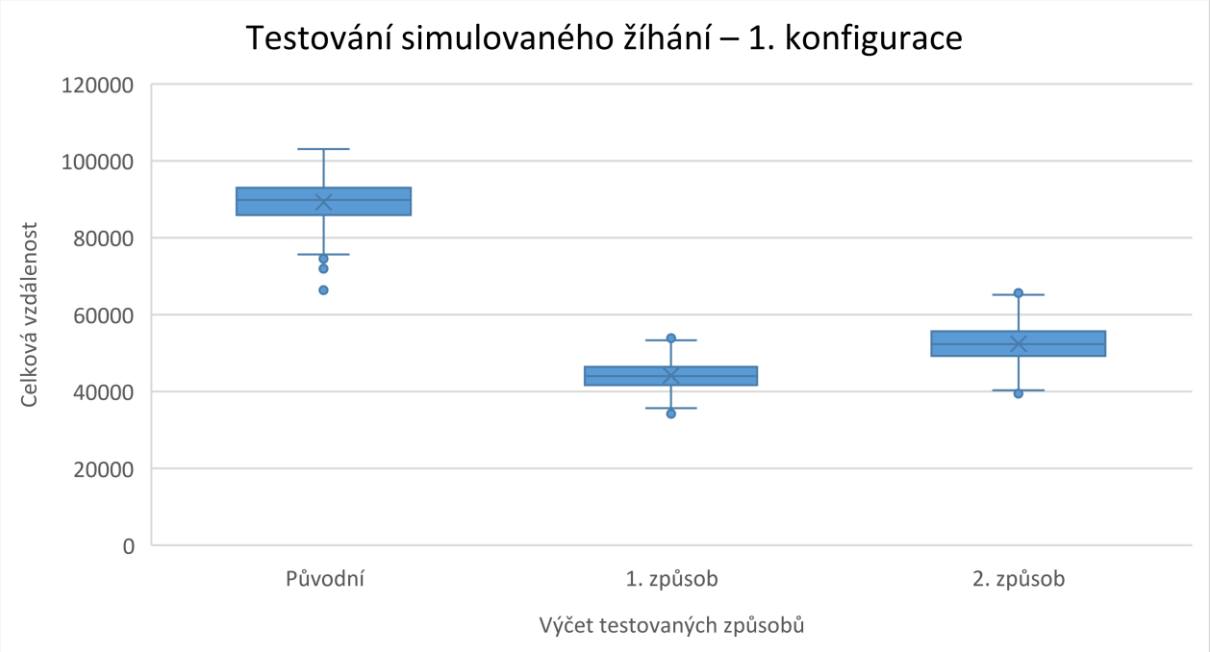
6.4 Výsledky Simulovaného žíhání

Testování simulovaného žíhání probíhalo dvěma způsoby. První testovaný způsob byl pomocí použití genetického algoritmu na počáteční populaci s následným aplikováním simulovaného žíhání na nejlepšího jedince. Druhý způsob probíhal aplikováním simulovaného žíhání před selekcí rodiče v genetickém algoritmu.

Oba tyto testované způsoby byly použity ve třech různých konfigurací. Konfigurace používaly stejný data set obsahující 29 měst, u každého testování bylo provedeno 1000 opakování pro širší škálu hodnot. Každá konfigurace měla nastavené specifické počty provedených evolucí genetického algoritmu a poté specifický počet kroků u simulovaného žíhání.

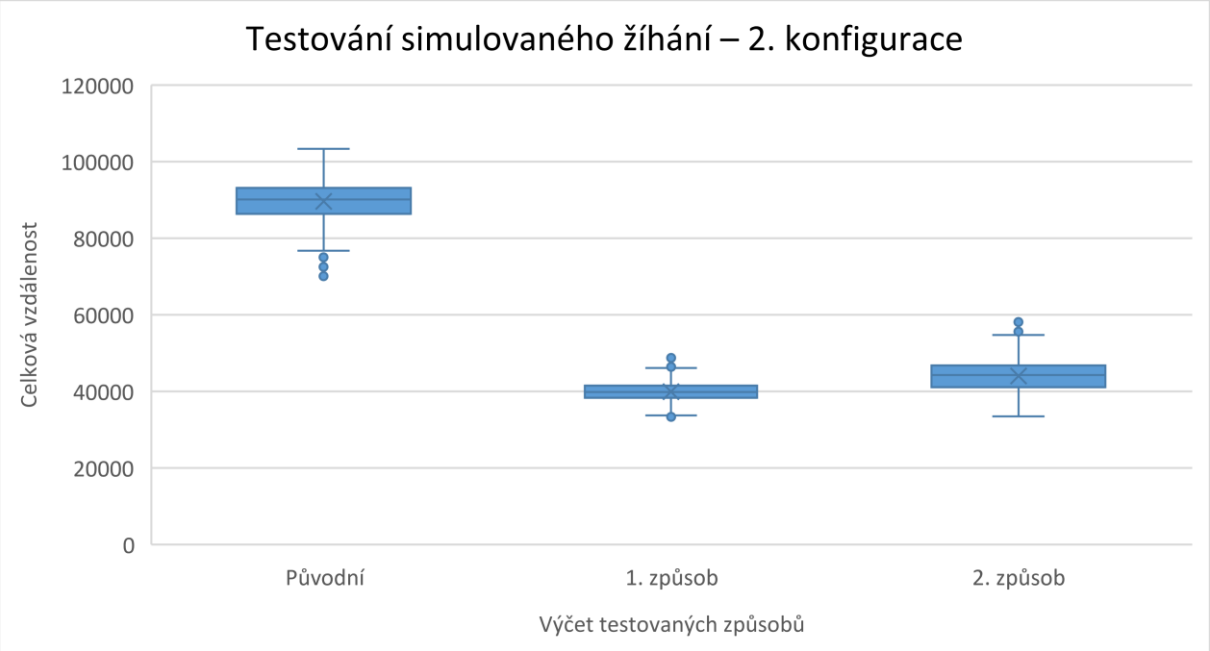
První konfigurace testovala první způsob s 80 generací v genetickém algoritmu a v simulovaném žíhání 580 kroků. Druhý způsob testoval 5 kroků simulovaného žíhání před aplikováním selekce, genetický algoritmus testoval 20 evolucí. Výsledky této konfigurace jsou dostupné

v grafu na obrázku 8. Optimalizace pomocí těchto metod byla u prvního způsobu průměrně zlepšená o přibližně 50 %, u druhého způsobu průměrné o 41 % proti počáteční hodnotě.



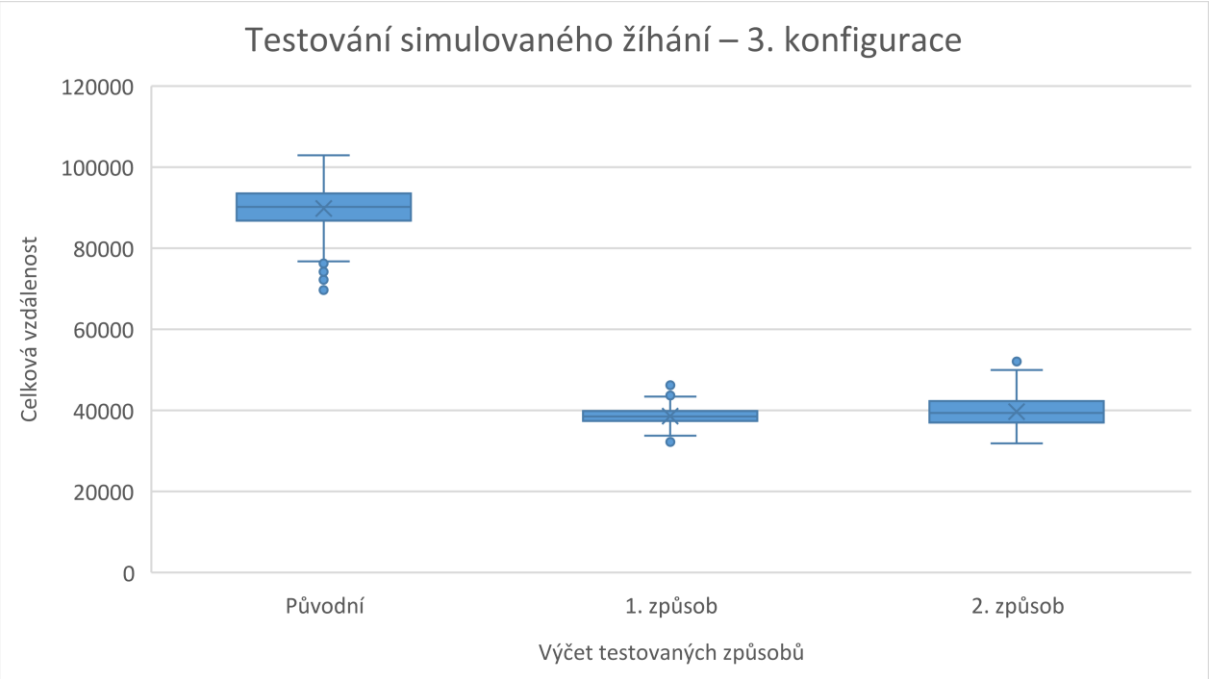
Obrázek 8 – Graf porovnávaných výsledků testování první konfigurace u simulovaného žíhání

Druhá konfigurace testovala u prvního způsobu 200 evolucí v genetickém algoritmu a v simulovaném žíhání 1450 kroků. Druhý způsob testoval 5 kroků simulovaného žíhání před aplikováním selekce, genetický algoritmus testoval 40 evolucí. Druhá konfigurace byla úspěšnější než první. Optimalizace pomocí této konfigurace byla zlepšena u prvního způsobu průměrně o 56 % a u druhého způsobu průměrné o 51 %. Výsledky jsou zobrazené v grafu na obrázku 9.



Obrázek 9 – Graf porovnávaných výsledků testování druhé konfigurace u simulovaného žíhání

Třetí konfigurace testovala u prvního způsobu 350 evolucí v genetickém algoritmu a v simulovaném žíhání 4350 kroků. Druhý způsob testoval 5 kroků simulovaného žíhání před aplikováním selekce, genetický algoritmus testoval 100 evolucí. Výsledky jsou zobrazené na obrázku 10. Optimalizace pomocí této konfigurace byla zlepšena u prvního způsobu průměrně o 57 % a u druhého způsobu průměrně o 56 %.



Obrázek 10 – Graf porovnávaných výsledků testování třetí konfigurace u simulovaného žíhání

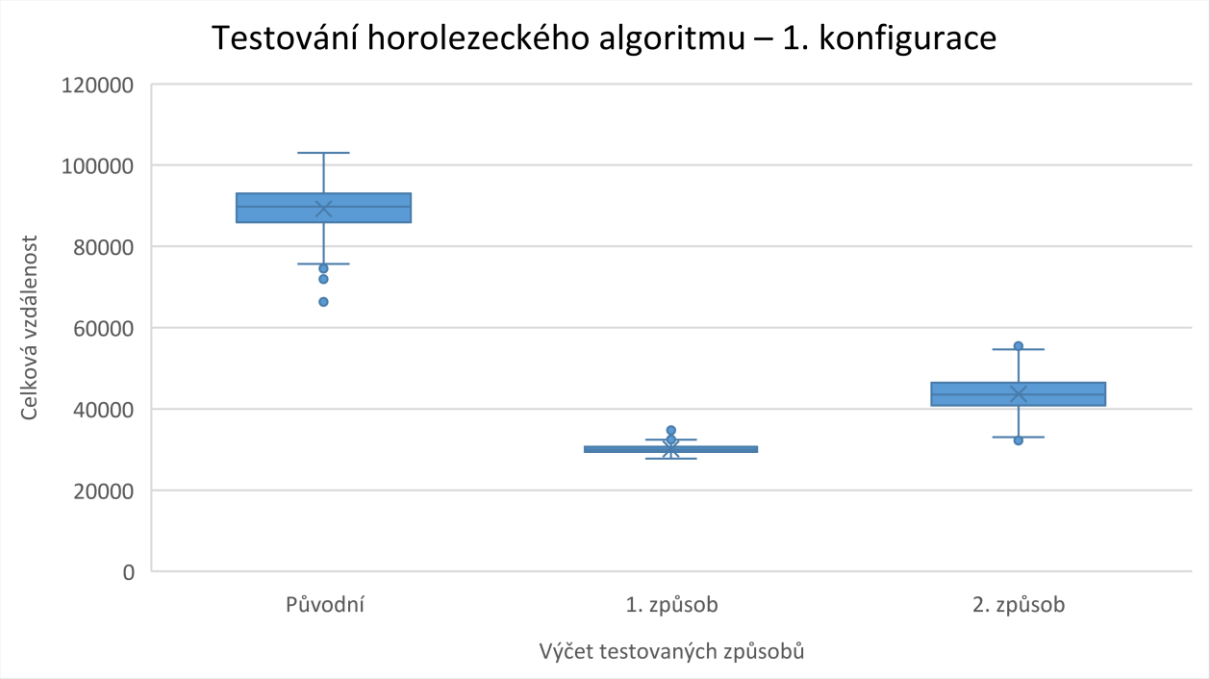
Při krátkém celkovém porovnání výsledků těchto testovaných konfigurací je možné vidět, že tato kombinace přináší zlepšení oproti původnímu řešení v obou kombinacích. Nicméně lze konstatovat, že vyšší počet evolucí a vyšší počet kroků při používání prvního způsobu tedy použití genetického algoritmu s následnou optimalizací simulovaným žíháním nemusí nutně přinášet lepší výsledky. Je možné vidět rozdíl na konfiguraci druhé a třetí, kde bylo vyhodnocení fitness funkce dvojnásobné, avšak výsledky byly zlepšeny průměrně pouze o 1 %. V časovém trvání jsou tyto rozdíly také viditelné. Zvyšující se počet evolucí a vyšší počet kroků také značně prodlužuje dobu trvání samotného běhu testování. Zvýšený počet evolucí a počet kroků přináší také výhodu v tom, že získané hodnoty mají menší rozptyl hodnot. Získané hodnoty jsou více ustálené a přesnější pro možné vytvoření názorů. Při porovnání druhého způsobu tedy aplikování simulovaného žíhání před selekcí v genetickém algoritmu se s vyšším počtem evolucí a kroků se optimalizace příliš nemění, dochází pouze k malým změnám.

6.5 Výsledky Horolezeckého algoritmu

Testování horolezeckého algoritmu probíhalo na specifickém typu, konkrétně prostém horolezeckém algoritmu. Samotný algoritmus měl dva testované způsoby. První testovaný způsob byl pomocí použití genetického algoritmu na počáteční populaci s následným aplikováním horolezeckého algoritmu. Druhý způsob probíhal aplikováním horolezeckého algoritmu před selekcí rodiče v genetickém algoritmu.

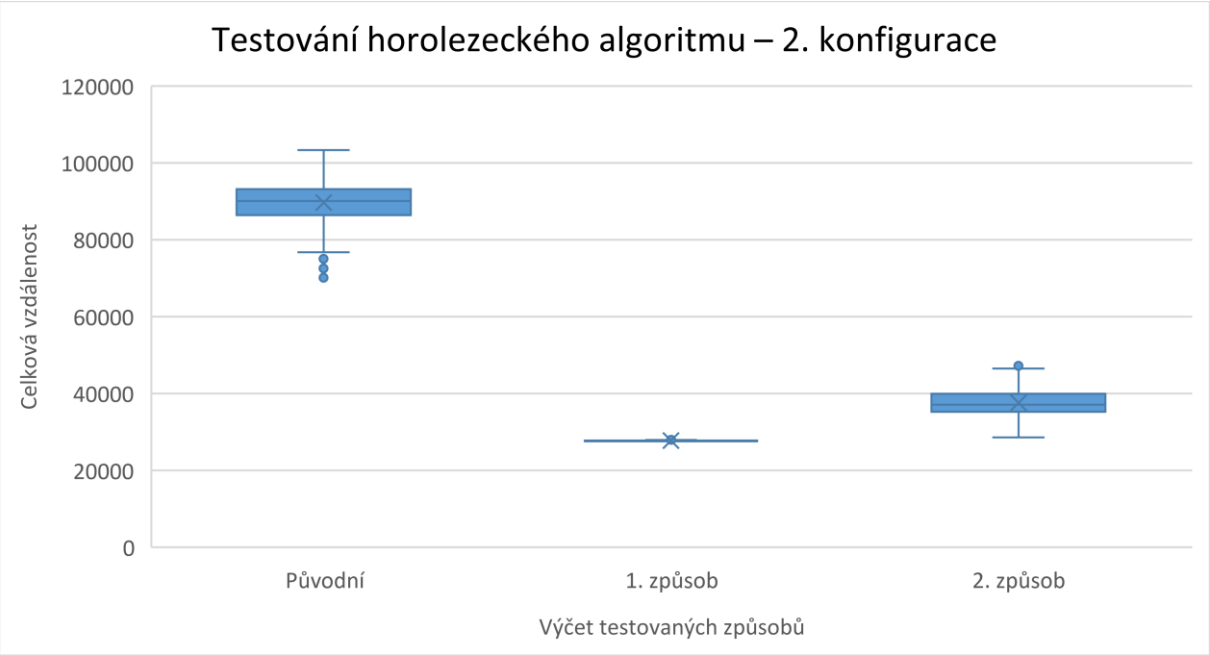
Oba tyto testované způsoby byly použité v třech různých konfiguracích. Konfigurace používaly stejný data set obsahující 29 měst, u každého testování bylo provedeno 1000 pokusů. Jednotlivé konfigurace byly nastavené na specifické počty provedených evolucí genetického algoritmu a také na specifický počet kroků samotného horolezeckého algoritmu.

První konfigurace testovala pro první způsob 80 evolucí v genetickém algoritmu a v horolezeckém algoritmu 580 kroků. Druhý způsob testoval 5 kroků horolezeckého algoritmu před aplikováním selekce, genetický algoritmus poté testoval 20 evolucí. Výsledky získané z této konfigurace jsou zobrazené v grafu na obrázku 11. Optimalizace přinesla průměrné zlepšení u prvního způsobu přibližně o 66 % a u druhého způsobu přibližně o 51 %.



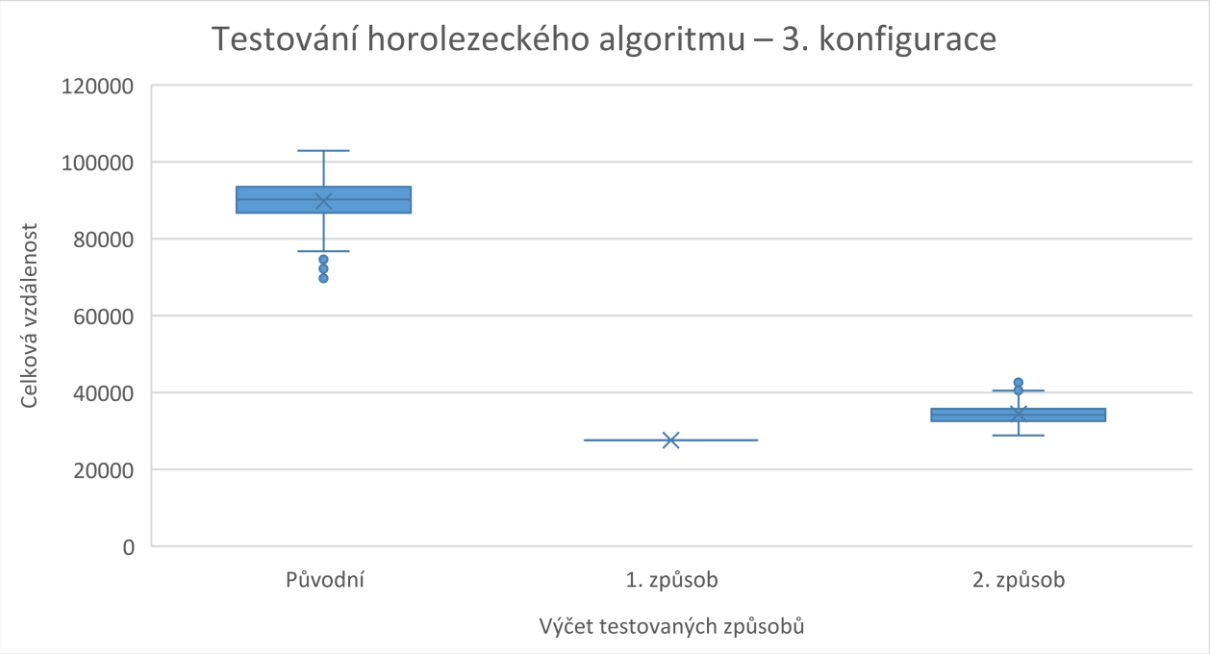
Obrázek 11 – Graf porovnávaných výsledků testování první konfigurace u horolezeckého algoritmu

Druhá konfigurace testovala pro první způsob 200 evolucí v genetickém algoritmu a horolezeckém algoritmu 1450 kroků. Druhý způsob testoval 5 kroků simulovaného žíhání před aplikováním selekce, genetický algoritmus poté testoval 50 evolucí. Výsledky této konfigurace jsou zobrazené v grafu na obrázku 12. Optimalizace přinesla průměrné zlepšení o 69 % u prvního způsobu a u druhého způsobu o 58 % proti původní hodnotě.



Obrázek 12 – Graf porovnávaných výsledků testování druhé konfigurace u horolezeckého algoritmu

Třetí konfigurace testovala pro první způsob 350 evolucí v genetickém algoritmu a poté v následném simulovaném žíhání 4350 kroků. Druhý způsob testoval 5 kroků simulovaného žíhání před aplikováním selekce, genetický algoritmus poté testoval 100 evolucí. Výsledky třetí konfigurace jsou dostupné v grafu na obrázku 13. Optimalizace u třetí konfigurace přinesla průměrné zlepešení o 69 % u prvního způsobu a druhý způsob přinesl průměrné zlepšení o 61 % proti původní hodnotě.



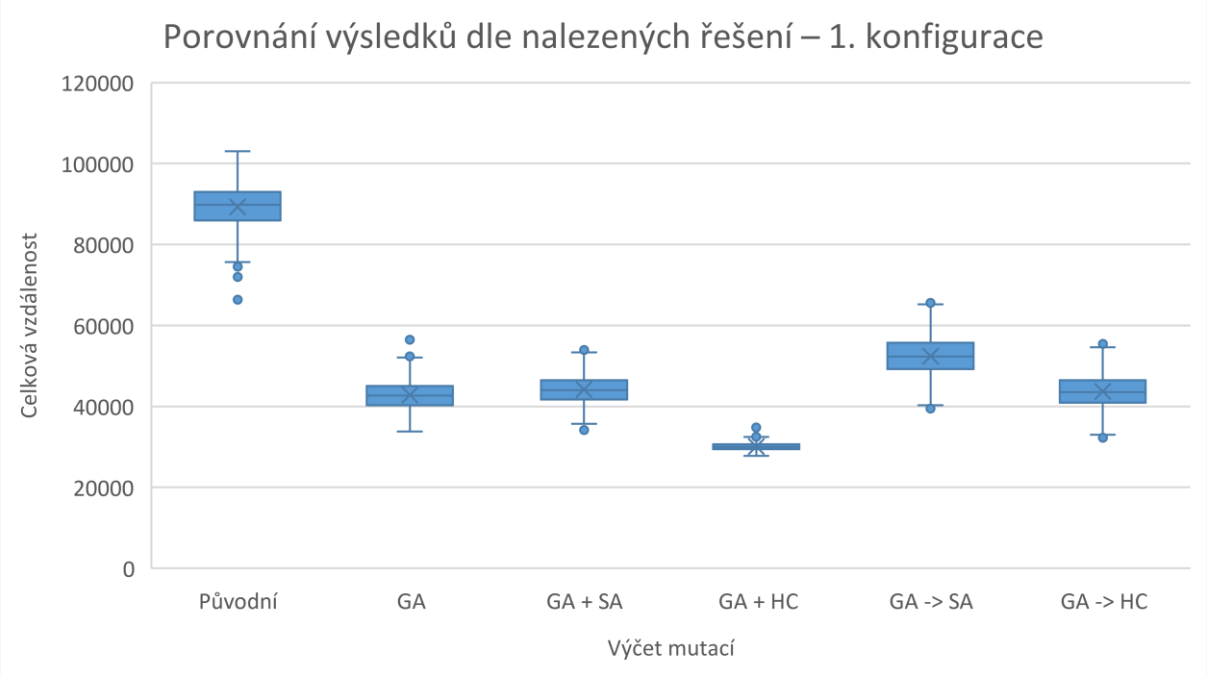
Obrázek 13 – Graf porovnávaných výsledků testování třetí konfigurace u horolezeckého algoritmu

Při krátkém celkovém porovnání výsledků testovaných konfigurací je možné vidět, že tyto kombinace přináší velmi dobré výsledky při používání obou způsobů. Už u první konfigurace jsou hodnoty velmi ustálené, při zvyšování počtu evolucí a počtu optimalizačních kroků horolezeckého algoritmus. U druhého způsobu je viditelné, že při zvýšení evolucí o dvojnásobek, jak je tomu u druhé a třetí konfigurace, zde je dvojnásobný rozdíl ve vyhodnocení fitness funkce, ale rozdíl zlepšení je velmi malý, nicméně je možné vidět že s rostoucím počtem evolucí a kroků horolezeckého algoritmu se hodnoty více ustálují.

6.6 Celkové vyhodnocení výsledků

Při zpracování Problému obchodního cestujícího pro vyhodnocení celého algoritmu byl zvolený datový set obsahující 29 měst, které byly zpracovávány. Při každém zpracování byla použitá jiná konfigurace. Konfigurace byla vždy aplikována na tisíc opakování pro větší rozmanitost získaných dat. Na počátku bylo otestováno, jaká kombinace genetických operátorů bude přinášet nejlepší výsledky a jaký druh mutace je vhodné aplikovat na optimalizační metody. Každá z testovaných konfigurací měla stejný počet vyhodnocení fitness funkcí. Konfigurace byly testovány na genetickém algoritmu s následnou aplikací optimalizační metody tedy simulovaného žíhání nebo horolezeckého algoritmu a poslední aplikací optimalizační metodou před selekcí v genetickém algoritmu.

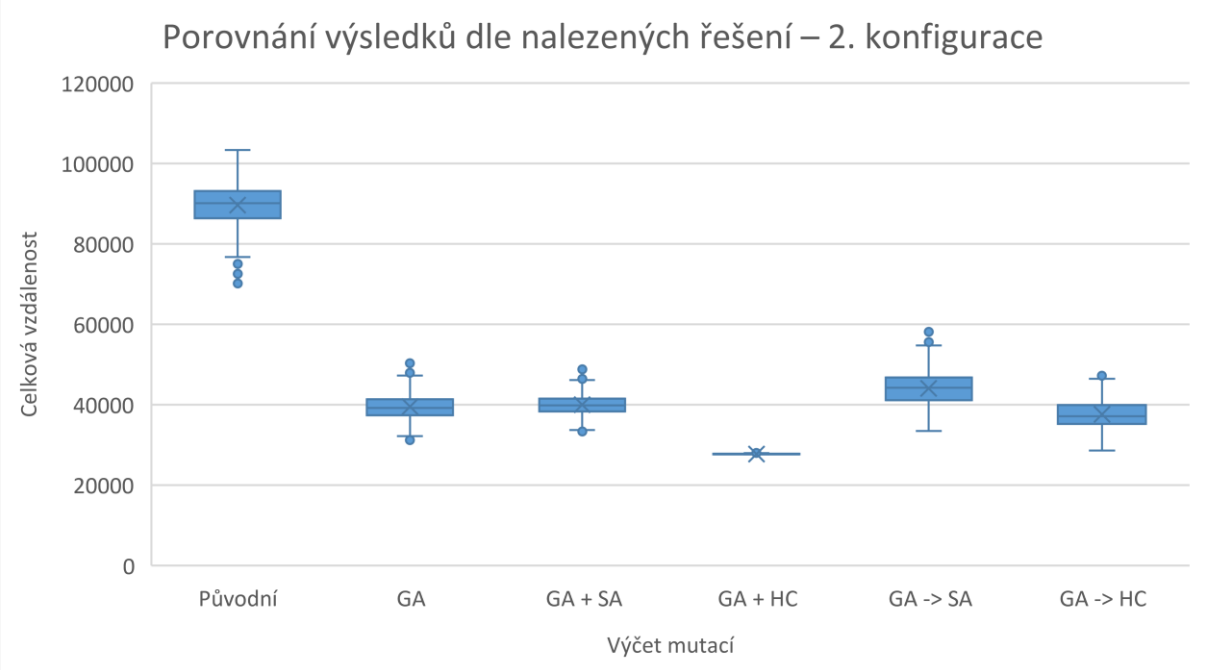
První konfigurace testovala 100 evolucí pro samotný genetický algoritmus, tedy v přepočtu 2900 vyhodnocení fitness funkce. Zbylé algoritmy musely být vhodně upravené pro odpovídají počet vyhodnocení fitness funkcí. Genetický algoritmu s použitím optimalizační metody byl nastavený na 80 evolucí a 580 kroků optimalizační metody. Obě optimalizační metody testovaly vždy stejný počet kroků. V případě optimalizování před samotnou selekcí uvnitř genetického algoritmu bylo zvoleno 5 kroků optimalizační metody a 20 evolucí u genetického algoritmu. Celkové vyhodnocení je možné vidět v grafu na obrázku 14.



Obrázek 14 – Graf porovnávaných kombinací u první testované konfigurace

Při porovnání získaných výsledků v první konfiguraci z grafu na obrázku 14 je na první pohled viditelné, že došlo k rapidnímu zlepšení oproti původnímu řešení. V grafu je viditelné, že samotný genetický algoritmus si vede poměrně dobře, nejlépe vychází genetický algoritmus s následnou aplikací horolezeckého algoritmu. Naopak nejmenší zlepšení přináší aplikování simulovaného žíhání před selekcí genetického algoritmu. To může být zapříčiněno tím, že simulované žíhání nemusí nutně postupovat pouze do zlepšujícího řešení, ale může také do horšího pro vyhnutí se uvíznutí v lokálním minimu, zatímco horolezecký algoritmus jde pouze do lepšího.

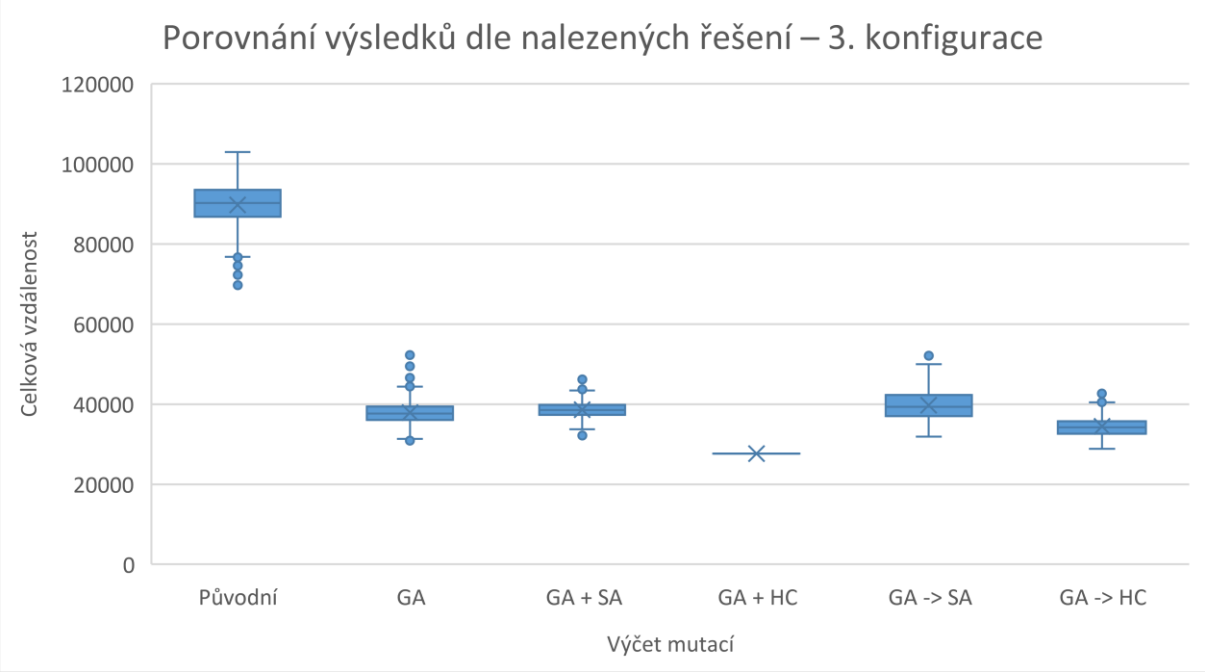
Druhá konfigurace testovala 250 evolucí na samotný genetický algoritmus, to je v přepočtu 7250 vyhodnocení fitness funkce. Zbylé algoritmy byly díky tomu upraveny následovně, pro genetický algoritmus s následným vstupem do optimalizační metody, bylo nastaveno 200 evolucí a samotná optimalizace měla 1450 kroků. V případě optimalizování před samotnou selekcí uvnitř genetického algoritmu bylo zvoleno 5 kroků optimalizační metody a 50 evolucí samotného genetického algoritmu. Celkové zhodnocení je možné vidět v grafu na obrázku 15.



Obrázek 15 – Graf porovnávaných kombinací u druhé testované konfigurace

Následné porovnání získaných výsledků druhé konfigurace je viditelné v grafu na obrázku 15. Při porovnání vůči první konfiguraci dostupné v grafu na obrázku 14 je vidět, že výsledky přinesly úspěšně vyšší zlepšení než předchozí konfigurace. Oproti předchozí konfiguraci lze vidět, že genetický algoritmus s optimalizací před selekcí pomocí horolezeckého algoritmu předčil běžný genetický algoritmus. Jinak výsledky zůstávají ve stejném pořadí v rámci efektivity. Nejefektivnější je genetický algoritmus s následnou optimalizací pomocí horolezeckého algoritmu. Pomyslně nejméně optimalizujícím algoritmem nadále zůstává genetický algoritmus s optimalizací pomocí simulovaného žíhání. Zbylé nalezené hodnoty přinesly taktéž zlepšení, ale spíš se v této konfiguraci zmenšilo spektrum nalezených hodnot, a začalo být více stabilní.

Třetí konfigurace testovala 500 evolucí na samotný genetický algoritmus, to je 14500 vyhodnocení fitness funkce. Použití genetického algoritmu s následným použití optimalizační metody obsahovalo 350 evolucí genetického algoritmu a 4350 kroků optimalizační metody. Stejný počet kroků optimalizační metody bylo pro simulované žíhání i pro horolezecký algoritmus. V případě optimalizování před samotnou selekcí uvnitř genetického algoritmu bylo zvoleno 5 kroků optimalizační metody a 100 evolucí samotného genetického algoritmu.



Obrázek 16 – Graf porovnávaných kombinací u třetí testované konfigurace

Porovnání získaných výsledků třetí konfigurace je viditelné v grafu na obrázku 16. Při porovnání vůči předchozí konfiguraci dostupné v grafu na obrázku 15. je vidět, že výsledky přinesly už minimální zlepšení vůči druhé konfiguraci, ale došlo k většímu ustálení hodnot.
Genetický algoritmus s aplikací optimalizační metody horolezeckého algoritmu zřejmě narazil na svoji nejnižší hodnotu, kterou může nalézt, protože má hodnoty velmi ustálené, tudíž jakékoliv další zvýšení počtu kroků či evolucí již ztrácí smysl. Genetický algoritmus s aplikací horolezeckého algoritmu před selekcí přinesl opětovné zlepšení výsledků. Dosáhne-li alespoň mírného zlepšení, mohl by se případně stát kandidátním řešením.

Při celkovém zhodnocení vše testovaných konfiguraci lze konstatovat, že i samotný genetický algoritmus přináší velmi dobré výsledky. Použitím správné kombinace může prostý genetický algoritmus nalézt optimální řešení pro daný problém. Díky aplikování optimalizační metody může však toto řešení nalézt mnohem rychleji. Nejúčinnější kombinací se dle testovaných konfigurací zdá být genetický algoritmus s následným aplikováním optimalizace pomocí horolezeckého algoritmu. Případně genetický algoritmus s optimalizací před samotnou selekcí. Tento přístup přinesl o něco horší výsledky, které by šly zlepšit zvýšením počtu evolucí nebo případně kroky optimalizace, ale byl by mnohem rychlejší v ohledu časového trvání zpracování. Genetický algoritmu s následným aplikováním optimalizace pomocí simulovaného žíhání zpočátku zaostával, ale s postupným zvyšováním evolucí a kroků optimalizace začala dohánět samotný genetický algoritmus. Pomyslně nejslabším algoritmem ale byl genetický algoritmus, který aplikoval optimalizaci před selekcí pomocí simulovaného žíhání. Pozitivním zjištěním je, že všechny provedené kombinace fungovaly, protože přinesly zlepšení nejméně o 41 % proti původnímu náhodně vygenerovanému řešení.

Porovnání procentuálních zlepšení vůči původnímu nalezenému řešení bylo následovné.
První konfigurace přinesla u genetického algoritmu o přibližně 52 %, genetický algoritmus s aplikací optimalizace v případě použití simulovaného žíhání o přibližně 50 % a v případě horolezeckého algoritmu přibližně o 66 %. Použití optimalizace před selekcí v genetickém algoritmu přineslo zlepšení u simulovaného žíhání o přibližně 41 % a horolezecký algoritmus přibližně o 51 %.
Druhá konfigurace přinesla u genetického algoritmu zlepšení o přibližně 56 %, genetický algoritmus s aplikací optimalizace v případě použití simulovaného žíhání se zlepšil přibližně o 56 %

a v případě horolezeckého algoritmu přibližně o 69 %. Použití optimalizace před selekcí v genetickém algoritmu přineslo zlepšení u simulovaného žíhání o přibližně 51 % a pro horolezecký algoritmus přibližně o 58 %.

Třetí konfigurace přinesla u genetického algoritmu zlepšení o přibližně 57 %, genetický algoritmus s aplikací optimalizace v případě použití simulovaného žíhání se zlepšil přibližně o 57 % a v případě horolezeckého algoritmu přibližně o 69 %. Použití optimalizace před selekcí v genetickém algoritmu přineslo zlepšení u simulovaného žíhání o přibližně 56 % a pro horolezecký algoritmus přibližně o 61 %.

7 ZÁVĚR

Hlavní cílem této bakalářské práce bylo porovnání několika adaptivních genetických algoritmů. Adaptivní genetické algoritmy prováděly testování standardního optimalizačního problému, konkrétně Problém obchodního cestujícího. Při tvorbě počátečního genetického algoritmu bylo provedeno testování, které určilo, jakou vhodnou kombinaci genetických operátorů použít pro nalezení nejlepších výsledků. Nejvhodnější sadou genetických operátorů pro genetický algoritmus bez žádné optimalizační metody byla kombinace operátorů jednobodového křížení s využitím reverse mutace. Dále byly navrženy optimalizační metody, které se pokoušely zlepšit nejlepšího jedince z populace po aplikování genetického algoritmu nebo před selekcí v samotném genetickém algoritmu. Jako optimalizační metody byly zvoleny simulované žíhání a prostý horolezecký algoritmus. Simulované žíhání používalo scramble mutaci a horolezecký algoritmus swap mutaci. Po navržení a implementaci samotného algoritmu a optimalizačních metod byla testována vhodná konfigurace pro testování datového souboru. Veškeré testované konfigurace přinesly zlepšení, některé větší jiné zase o něco menší. Se zvyšujícím počtem generací napříč všemi algoritmy rozptyl nalezených řešení postupně klesalo a ustalovalo se konzistentně kolem hodnot. Toto zvyšování generací však přineslo úskalí v podobě časové prodlevy mezi vyhodnocením.

Při pohledu napříč všemi testovanými konfiguracemi došlo k minimálnímu zlepšení o průměrně 41 % a nejvyšší zlepšení bylo v podobě 69 %. Nejúspěšnější kombinací bylo použití genetického algoritmu s následným aplikováním optimalizační metody horolezeckého algoritmu, které přineslo zmíněných průměrných 69 %. Tato kombinace může být potenciální kandidátní řešení. Pomyslně nejhorší kombinace byla pokus optimalizace pomocí simulovaného žíhání před selekcí v genetickém algoritmu, která i při vyšším počtu fitness vyhodnocení přinesla horší výsledky než její kolegové.

Důležité je zdůraznit, že konfigurace, které byly použité u testovaného datového souboru, který obsahoval dvacet devět měst, jsou uzpůsobené pro tento specifický datový soubory a nemusí tak vyhovovat pro většinu jiných. Pro případný jiný datový soubor, který bude například o polovinu kratší, není nutné provádět tak vysoké množství generací nebo případně u optimalizačních metod takové množství kroků pro nalezení globálního optimálního řešení. Naopak pokud bude použitý datový soubor, který bude obsahovat tisíce až deseti tisíce zastávek, tak pro něj bude může být tato konfigurace nevyhovující.

Na závěr lze konstatovat, že bakalářská práce splnila všechny vytyčené cíle a sloužila jako dobrý úvod do problematiky genetických algoritmů. Samotná práce poskytuje velké možnosti v případě budoucího rozšíření. Rozšíření by mohly být například v podobě implementování dalších optimalizačních metod, po získáních dalších znalostí a zkušeností případně navrhnout svoje vlastní adaptivní metody a provést testování na rozsáhlých data setech. Další možnost rozšíření by byla implementovat další optimalizační problémy a provést případné porovnání jaké algoritmy více vyhovují jakému problému.

POUŽITÁ LITERATURA

[1] HYNEK, Josef. Genetické algoritmy a genetické programování. Praha: Grada, 2008. Průvodce (Grada). ISBN 978-80-247-2695-3.

[2] MITCHELL, Melanie. An introduction to genetic algorithms. Cambridge, Mass.: MIT Press, c1996. ISBN 978-0262133166.

[3] Harik, Georges & Lobo, Fernando. (1999). A parameter-less genetic algorithm. Proceedings of the Genetic and Evolutionary Computation Conference. 258–265.

[4] GOLDMAN, Brian W. a William F. PUNCH. Parameter-less population pyramid. In: Proceedings of the 2014 conference on Genetic and evolutionary computation – GECCO '14 [online]. New York, New York, USA: ACM Press, 2014, 2014, s. 785-792 [cit. 2018-10-18]. DOI: 10.1145/2576768.2598350. ISBN 9781450326629. Dostupné z: http://dl.acm.org/citation.cfm?doid=2576768.2598350

[5] LARRAÑAGA, P., C. M. H. KUIJPERS, R. H. MURGA, I. INZA a S. DIZDAREVIC. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. In: KUIJPERS, I. INZA. The Artificial intelligence review [online]. 2. vyd.: Springer Netherlands, 1999 [cit. 2012-05-22]. ISSN 0269-2821. DOI: 10.1023/A:1006529012972. Dostupné z: http://dx.doi.org/10.1023/A:1006529012972

[6] Fang, Y., Li, J. (2010). A Review of Tournament Selection in Genetic Programming. In: Cai, Z., Hu, C., Kang, Z., Liu, Y. (eds) Advances in Computation and Intelligence. ISICA 2010. Lecture Notes in Computer Science, vol 6382. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16493-4_19

[7] Lones, Michael. (2011). Sean Luke: essentials of metaheuristics. Genetic Programming and Evolvable Machines. 12. 333-334. 10.1007/s10710-011-9139-0.

[8] Katoch, S., Chauhan, S.S. & Kumar, V. A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80, 8091–8126 (2021). https://doi.org/10.1007/s11042-020-10139-6.

[9] Hassanien, A.E., & Emary, E. (2016). Swarm Intelligence: Principles, Advances, and Applications (1st ed.). CRC Press. https://doi.org/10.1201/9781315222455

[10] HYNEK, Josef. Definice problému. In: Genetické algoritmy a genetické programování. Praha: Grada, 2008, s. 112. Průvodce (Grada). ISBN 978-80-247-2695-3.

[11] Gharsalli L. Hybrid Genetic Algorithms [internet]. Optimisation Algorithms and Swarm Intelligence. IntechOpen; 2022. Available from: http://dx.doi.org/10.5772/intechopen.104735

[12] KENDALL, Graham. *Simulated Annealing* [online]. Manchester, 2012, 20/2/2012, 8 [cit. 2023-04-03]. Dostupné z: http://syllabus.cs.manchester.ac.uk/pgt/2017/COMP60342/lab3/Kendall-simulatedannealing.pdf

[13] Dimitris Bertsimas, John Tsitsiklis "Simulated Annealing," Statistical Science, Statist. Sci. 8(1), 10-15, (February, 1993)

[14] VAN LAARHOVEN, Peter J. M. a Emile H. L. AARTS. *Simulated Annealing: Theory and Applications* [online]. Dordrecht: Springer Netherlands, 1987 [cit. 2023-05-07]. ISBN 978-90-481-8438-5. Dostupné z: doi:10.1007/978-94-015-7744-1

[15] Kour, Haneet & Sharma, Parul & Abrol, Pawanesh. (2015). Analysis of Fitness Function in Genetic Algorithms. 1. 87-90.

[16] Soni, Nitasha and Tapas Kumar. “Study of Various Mutation Operators in Genetic Algorithms.” (2014).

[17] RABUÑAL DOPICO, Juan Ramón, Julian DORADO a Alejandro PAZOS, ed. *Encyclopedia of Artificial Intelligence* [online]. IGI Global, 2009 [cit. 2023-05-03]. ISBN 9781599048499. Dostupné z: doi:10.4018/978-1-59904-849-9.

[18] Bottaci, Leonardo. (2001). A Genetic Algorithm Fitness Function for Mutation Testing. SEMINAL: Software engineering using metaheuristic inovative algorithms, workshop.

[19] GOLDBERG, David Edward. *Genetic algorithms in search, optimization, and machine learning*. Boston: Addison-Wesley, 1989. ISBN 978-0201157673.

[20] EIBEN, A. E. a J. E. SMITH. *Introduction to evolutionary computing*. Second edition. Heidelberg: Springer, [2015]. Natural computing series. ISBN 978-3-662-44873-1.

[21] SONG, Xudong a Yunlong XIAO. An Improved Adaptive Genetic Algorithm. In: *Proceedings of the 2013 Conference on Education Technology and Management Science* [online]. Paris, France: Atlantis Press, 2013, 2013, - [cit. 2023-04-05]. ISBN 978-90786-77-72-7. Dostupné z: doi:10.2991/icetms.2013.398

[22] Talbi, El-Ghazali. (2013). Combining metaheuristics with mathematical programming, constraint programming and machine learning. 4OR. 11. 10.1007/s10288-013-0242-3.

[23] BURKE, Edmund K. a Graham KENDALL, ed. *Search Methodologies* [online]. Boston, MA: Springer US, 2014 [cit. 2023-04-06]. ISBN 978-1-4614-6939-1. Dostupné z: doi:10.1007/978-1-4614-6940-7

[24] RUSSELL, Stuart J. a Peter NORVIG. *Artificial intelligence: a modern approach*. 3rd ed. Upper Saddle River: Prentice Hall, 2010. Prentice Hall series in artificial intelligence. ISBN 978-0-13-604259-4.

[25] Brian W. Goldman and William F. Punch. 2014. Parameter-less population pyramid. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14). Association for Computing Machinery, New York, NY, USA, 785–792. https://doi.org/10.1145/2576768.2598350

[26] El-Mihoub, T.A., Hopgood, A.A. & Nolle, L. Self-adaptive learning for hybrid genetic algorithms. Evol. Intel. 14, 1565–1579 (2021). https://doi.org/10.1007/s12065-020-00425-5

[27] University of Waterloo. (n.d.). World TSP problem instances by country. University of Waterloo. Retrieved from https://www.math.uwaterloo.ca/tsp/world/countries.html#WI

[28] REEVE, Hudson Kern a Paul W. SHERMAN. Adaptation and the Goals of Evolutionary Research. *The Quarterly Review of Biology* [online]. 1993, **68**(1), 1-32 [cit. 2023-04-28]. ISSN 0033-5770. Dostupné z: doi:10.1086/417909

[29] ESHELMAN, Larry J. a J. David SCHAFFER. Real-Coded Genetic Algorithms and Interval-Schemata [online]. In:. Elsevier, 1993, 1993, s. 187-202 [cit. 2023-04-28]. Foundations of Genetic Algorithms. ISBN 9780080948324. Dostupné z: doi:10.1016/B978-0-08-094832-4.50018-0

[30] Brian W. Goldman and William F. Punch. 2014. Parameter-less population pyramid. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14). Association for Computing Machinery, New York, NY, USA, 785–792. https://doi.org/10.1145/2576768.2598350