Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Optimalizace rozmístění rádiových stanic

Diplomová práce

Autor: Bc. Jaromír Homolka
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Barbora Tesařová, Ph.D.

Hradec Králové duben 2017

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 27.4.2017 Jaromír Homolka

Poděkování:

Děkuji vedoucí diplomové práce Ing. Barboře Tesařové, Ph. D. za metodické vedení práce, za pomoc a cenné rady při zpracování této práce. Děkuji také panu Antonínu Heřmánkovi ze společnosti ERA a.s. za poskytnutí tématu, materiálů a odborných rad v této problematice.

Anotace

Diplomová práce je zaměřena na optimalizaci rozmístěné rádiových stanic. Toto rozmístění je velmi důležité pro budoucí přesnost celého multilateračního systému pro určování polohy. Nicméně rozmísťovaní stanic je v mnoha případech řešeno empiricky, což má za následek různou kvalitu v získaných výsledcích. Díky této skutečnosti je v práci nastíněný vlastní algoritmus pro optimalizaci tohoto rozmístění, jehož výhoda spočívá v jednoduchosti a pro dosažení rovnoměrně kvalitních výsledků. Práce je rozdělena do dvou částí. V první části jsou popsány základní pojmy - multilaterační systém, optimalizační metody, kde je blíže popsána větev evolučních algoritmů a popis poskytnutého programu firmy ERA a.s. Ve druhé části byl implementován optimalizační algoritmus a následné otestování výsledného rozmístění proti empiricky zadanému od daného experta.

Annotation

Title: Optimizing the distribution of radio stations

The Diploma Thesis is focused on the optimizing distribution of radio stations. This deployment is very important for the future accuracy of the multilateration surveillance system. The deployment of radio stations is solved empirically. This approach does not have uniform results. For this reason, the thesis designed new algorithm for distribution of radio stations. The algorithm is simple and it reaches uniform results. The thesis is divided into two parts. The first section describes the basic concepts such as multilateration, optimization method, especially evolutionary algorithms and program description, which are provided by ERA a.s. In the second part there has been implemented an optimization algorithm. The result of algorithm was tested against the expert's deployment.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Optimalizace se v posledních letech těší velké pozornosti. Velký důraz je kladen na algoritmy, které prochází velký objem dat, zvláště na jejich rychlost a efektivitu. Zároveň zde máme velkou konkurenci ve všech odvětvích. V současné informační době byly už mnohé procesy optimalizovány pro větší efektivitu a snížení nákladů celého procesu. Nicméně stále existují procesy, které probíhají již dvacet let bez jakékoliv změny. Příkladem může být multilaterační systém a jeho rozmístění rádiových stanic pro lepší přesnost daného systému. Tento proces stále funguje na základě výběru několika míst a následném otestováním, jak dané rozmístění vyhovuje. Hlavním tématem diplomové práce bude podat srozumitelné informace o jednotlivých pojmech, které se v nastavené problematice využívají, informace o celém multilateračním systému a následné optimalizaci rozmístění rádiových stanic.

Cílem diplomové práce bude navržení jednoduchého a intuitivního programu pro optimalizaci rozmístění rádiových stanic. Pro dosažení cíle byly nápomocné informace od společnosti ERA a.s., která se danou problematikou zabývá již několik let.

Práce je členěna do dvou hlavních částí. První část je spíše teoretického rázu. Bude zde vysvětlena základní myšlenka multilateračního systému, který ke svému fungování využívá poznatky z fyzikálních a matematických odvětví, dále pak zde bude vysvětlena základní struktura multilateračních systémů. Jeden z nejdůležitějších pojmů je multilaterace, ze kterého je odvozen i název systémů. V druhé polovině teoretické části bude nahlédnuto na optimalizační algoritmy obecně a blíže popsána větev evolučních algoritmů, které zastřešují např. genetické programování, genetické algoritmy atd.

Ve druhé části diplomové práce bude implementován optimalizační algoritmus pro rozmístění rádiových stanic. Implementace je vytvořena za pomocí programovací jazyku Java. K implementaci optimalizačního algoritmu byl využit princip evolučního algoritmu. Druhá podkapitola popisuje reálný příklad pro otestování výsledného algoritmu s porovnáním s výsledkem od firmy ERA a.s.

V závěru práce budou shrnuty představené informace. Bude poukázáno na klady a zápory výsledného algoritmu. Bude zde také zhodnoceno, jak dobře se algoritmu dařilo, jestli opravdu může zefektivnit proces rozmísťování rádiových stanic a tím pádem, zda má své opodstatnění i v blízké budoucnosti.

2 Teoretická východiska

Je třeba se připravit na praktickou část diplomové práce a vysvětlit všechny důležité pojmy vyskytující se v problematice multilaterace, optimalizací a evolučních algoritmů obecně. Toto téma je velmi rozsáhlé, a proto pro podrobnější rozbor problematiky budou vždy uvedeny odkazy do externích zdrojů.

2.1 Multilaterační systém

Multilaterační systém (MSS – Multilateration Surveillance System) je systém pro určování polohy cíle pomocí multilaterace (MLAT). Systém lze také pojmenovat jako Time Difference of Arrival (TDOA) systém. Tyto systémy poskytují spolehlivou a přesnou polohu v reálném čase. Pomocí signálu lze také identifikovat jednotlivé cíle a tudíž pomocí MSS znát cíl (o co se jedná) a jeho přesnou polohu. Tuto přesnou polohu lze docílit za každého počasí, což oproti ostatním konkurenčním systémům se říci nedá [1]. Také při jejich cenové nenáročnosti a vysokému výkonu jsou tyto systémy velmi často využívány v nejen civilním, ale také ve vojenském sektoru [2].

Multilaterace je v dnešní době běžnou technikou v navigačních systémech. Jsou snadno realizovatelné a nepotřebují společný synchronizovaný čas, protože se měří rozdíl časů příchodu signálu. Dříve bylo měření různých časů příchodů prováděno pomocí osciloskopu. Během druhé světové války se tato technika stala velmi populární [3]. Po vynalezení mikroprocesorů se usnadnila práce a provoz těchto systémů, které vedly k výraznému zvýšení popularity. V dnešní době se MSS využívají především na letištích, jak na objekty letící, tak i na pohyb na letištní ploše. MSS velmi často ke svému fungování využívají dalších upřesňujících informací od palubních systémů, např. barometrickou výšku [1].

V současnosti se často multilaterační systémy využívají s moderním sledovacím systémem ADS-B [4]. Automatic dependent suirveillance - broadcast (ADS-B) používá ke svému fungování informace z GPS, kde si určí vlastní polohu a následně ji spolu s dalšími informacemi cyklicky vysílají do prostoru. Daný signál se šíří všesměrově bez ohledu na to, kdo jej poslouchá. Nicméně v kontextu diplomové práce jsou to pozemní stanice nebo jiné dopravní prostředky. Tento

systém poskytuje přesnou informaci o poloze a relativně často ji aktualizuje, což pomáhá k lepšímu sledování a zvyšuje bezpečnost letového provozu. V moderních dopravních prostředcích je tento systém již standardem [5].

I když jsou systémy multilaterace a ADS-B zcela odlišné pro monitorování leteckého provozu, neměly by být považovány za konkurenční. Multilaterace a ADS-B se mohou mezi s sebou podporovat, verifikovat své výsledky a také zpřesňovat celý letecký provoz [6].

2.1.1 Struktura

Multilaterační systémy se skládají z hardwaru a ze softwaru. Pod pojmem hardware jsou myšleny pozemní stanice, vysílače, dotazovače, ale také CPS (Central proccessing station). Ze softwarové stránky to jsou jednotlivé implementované algoritmy pro výpočet pozice a různé programové struktury pro stanice a CPS. Následující kapitola je čerpána převážně Technického popisu MSS-A [7].

Systém se skládá z různých fyzických zařízení a každé vykonává různou funkci. První touto částí je přijímač (RX). Hlavním úkolem je přijímaní signálu, jeho následné zdigitalizování, změření přesného času příchodu (TOA) a odeslání těchto dat po datové lince do CPS (Central proccessing station). Hlavním hlediskem, jak upřesnit výslednou polohu cíle, je správné rozmístění těchto přijímačů. Správné rozmístění jednotlivých stanic je cílem diplomové práce. Rozdělují se podle toho, kde jsou umístěny, buď na letišti anebo mimo něj. Mezi sebou se liší pouze jinými anténami a jiným spojením s CPS (Central proccessing station).

Dále existují přijímače/vysílače (RXTX). Platí zde stejné informace jako v odstavci předešlém a také se stejně dělí. Tyto stanice nejsou nic jiného než spojení přijímačů a dotazovačů. Systémy jsou od sebe z hlediska funkcionality zcela nezávislé a lze přijímat i dotazovat se současně. Dotazování se provádí za účelem bližších informací pro zajištění lepší přesnosti a integrity.

Dalším důležitým hardwarem je monitorovací transpondér (RMTR – Reference and monitoring transponder). Jeho hlavním úkolem je udržení přesnosti a integrity v průběhu času. Náhodně periodicky odesílá krátké zprávy pro synchronizaci měření přesného času příchodu TOA.

Central proccessing station (CPS) je srdce celého MSS a je velmi důležitou součástí těchto systémů. Poskytuje celou funkcionalitu MSS od výpočtů multilaterace a pozici cíle až po management systému. Nejdůležitější úkoly, které CPS má udělat, je „oštítkování“ všech přijatých signálů z jednotlivých přijímačů, časovou korelaci, synchronizaci všech přijatých dat a výpočet jednotlivých poloh. CPS je složena z několika komponent. CPS obsahuje rychlý switch, redundantní NTP server, komunikační jednotku nebo také management server, který slouží pro řízení celého MSS. Nejdůležitější částí je cílový procesor (TP – target processor).

Target procesor je jednotka zabezpečující inicializaci a sledování cíle. V TP se odehrává výpočet přesné pozice pomocí multilaterace, generují se a posílají se zprávy cílů a také příkazy či data z jednotlivých přijímačů. V mnoha řešení se využívají alespoň dva TP.

Základní myšlenkou těchto systému je výpočet polohy cíle pomocí různých časů příchodu signálu od cíle na jednotlivá přijímací zařízení a jeho rozdílu mezi nimi. Tyto přijímače jsou na odlišných místech s přesnou vzdáleností mezi nimi. Je zřejmé, že je potřeba minimálně třech přijímacích zařízení respektive čtyř. Detailní princip fungování a jednotlivých výpočtů multilaterace bude blíže popsán v následujících kapitolách.

MSS lze rozdělit do dvou skupin – pasivní a aktivní. Pasivní multilaterační systémy fungují na základě odposlouchávání všech okolních signálů. Systém se spoléhá pouze na signál vysílaný z cíle. Výhodou je, že se neplní celkový počet dotazů na jednotlivých frekvencích (dotaz 1030MHz a odpověď 1090MHz). Aktivní systémy fungují téměř stejně jako pasivní. Liší se pouze tím, že si systém dokáže vyžádat bližší informace o cíli.

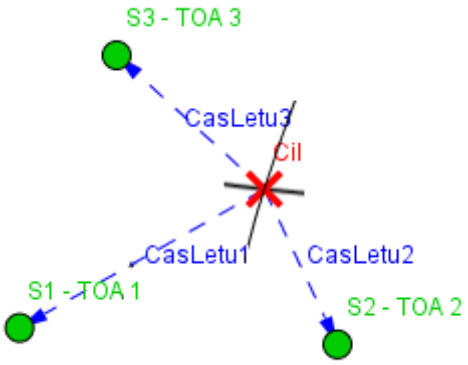
MSS jako takové se velmi často používají pro upřesnění pro další systémy. Příkladem jsou primární či sekundární radary. MSS se hodí do členitých systémů, kde se uplatňují různé výškové rozdíly a lze velmi snadno určit pozici a výšku cíle. Příkladem může být letiště v rakouském Innsbrucku, kde se dosáhlo značného úspěchu, jak z finančního, tak pozičního hlediska [8].

2.1.2 Multilaterace

Multilaterace, také hyperbolické polohování, je navigační technika pro určování polohy cíle pomocí přesného určení časového rozdílu signálu (TDOA) přijímaného na různých přijímačích. Pokud se určování polohy provádí v rovině (dvourozměrný prostor), je potřeba minimálně třech přijímacích zařízení. V případě trojrozměrného prostoru jsou nutné minimálně čtyři přijímače.

Hned na začátku je nutné upozornit, že multilaterace bývá často zaměňována s trilaterací. Je pravdou, že trilaterace funguje téměř na stejném principu, ke své práci využívá absolutní měření času přijímaného signálu (time of arrival, TOA) na různých stanovištích.

Cíl (např. letadlo) odesílá v pravidelných intervalech všesměrový signál v módu A/C nebo v novějším módu S, který poskytuje základní informace o cíli. Signál putuje konstantní rychlostí a rovnoměrně všemi směry. Následně je tento signál detekován na přijímačích. Rozdíl doputování je malý, řádově v nanosekundách. Na základě přesných detekcí času příchodu na jednotlivých přijímacích se určí rozdíl mezi těmito časy. Při konstantní rychlosti signálu a neměnné polohy přijímačů se určí výsledná poloha cíle. Není potřeba znát absolutní čas příchodu, nýbrž jejich rozdíl.

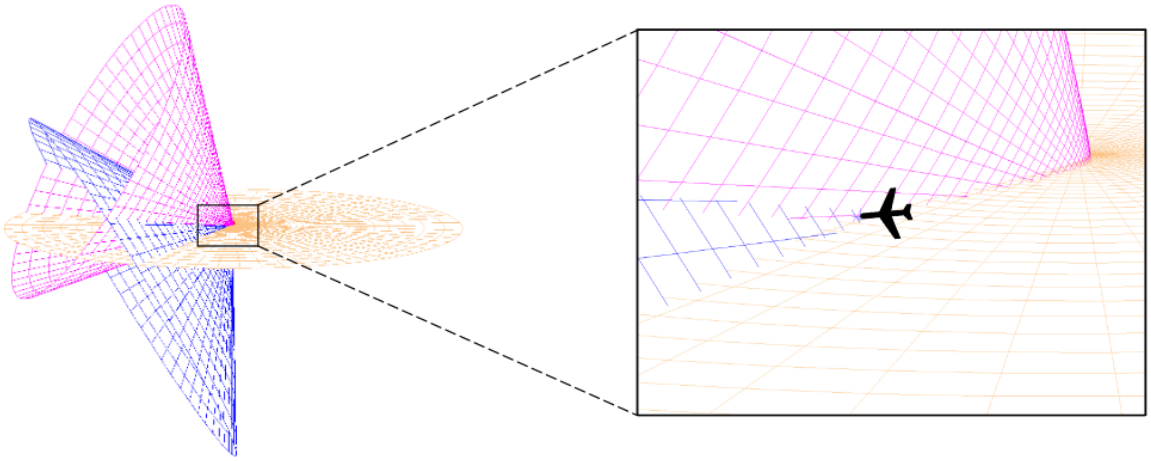


Obr. č. 1 - Ukázka multilaterace ve 2D, zdroj: autor

Na obrázku č. 1 je přehledně ukázáno, jak se principiálně určuje pozice ve dvourozměrném prostoru. Na každém přijímači se měří TOA, čas přijmutí signálu. Díky těmto časům je možné vypočítat vzdálenost od jednotlivých stanovišť a určit hyperboly, na kterých cíl leží. Hyperbola je množina všech bodů o daném rozdílu vzdáleností od dvou pevných ohnisek. Ohniska první hyperboly jsou S1 a S2. Podle toho, na jaký přijímač přijde první signál, lze určit, na jaké větvi hyperboly cíl

leží. Obdobně se určí druhá hyperbola, která musí mít jedno společné (centrální) ohnisko s první (S1 a S3). Hledaný cíl leží na průsečíku těchto hyperbol.

Přidáním dalšího přijímače, lze docílit určení polohy v trojrozměrném prostoru. Cíl se nebude nacházet na průsečíku třech hyperbol, nýbrž ve 3D průsečíku třech hyperboloidů. Průnik hyperboloidů je znázorněn na obrázku č. 2.



Obrázek č. 2 - Detail průsečíku hyperboloidů, zdroj: [39]

2.1.2.1 Výpočet

Předpokladem následujícího výpočtu je, že všechny stanice i cíl se nachází v jedné rovině a tudíž výpočet bude v dvourozměrném prostoru, kde se bude hledat průsečík dvou hyperbol v rovině. Jak již bylo řečeno, je potřeba třech stanic, které reprezentují ohniska výsledných hyperbol. Obě hyperboly musí mít společné centrální ohnisko.

Centrální ohnisko $C = (x_c, y_c)$ a dvě boční ohniska $B_1 = (x_{B1}, y_{B1})$ a $B_2 = (x_{B2}, y_{B2})$ jsou známé souřadnice a naměřené časy t_1,t_2 a t_c jsou také známy. Následně lze určit h_1 a h_2, které definují rozdíl vypočtených vzdálenosti polohy od cíle k bočnímu a centrálnímu přijímači. Cíl leží na souřadnici *T=(x, y).*

$$h_1 = f_1(x,y) = \big||T - B_1|\big| - \big||T - C|\big| \qquad \textit{Rovnice č. 1}$$

$$= \sqrt{(x - x_{B1})^2 + (y - y_{B1})^2} - \sqrt{(x - x_C)^2 + (y - y_C)^2}$$

$$h_1 = f_1(x,y) = \big||T - B_1|\big| - \big||T - C|\big| \qquad \textit{Rovnice č. 2}$$

$$= \sqrt{(x - x_{B1})^2 + (y - y_{B1})^2} - \sqrt{(x - x_C)^2 + (y - y_C)^2}$$

Tento systém dvou rovnic o dvou neznámých lze řešit dvěma způsoby, iteračně či explicitně. Celý postup výpočtu těchto rovnic lze nalézt například v diplomové práce od Vojtěcha Péky [4]. Práce se zabývala sledováním cílů pasivním multilateračním systémem a systémem ADS-B.

Při úspěšném výpočtu těchto rovnic mohou nastat tři situace – žádné, jedno či dvě řešení. Poloha cíle je samozřejmě pouze jedna a je nutno určit, jaká poloha je ta správná. Se správně rozmístěnými stanicemi lze v reálných podmínkách rozlišit několika způsoby polohu daného cíle. První možností je ve dvourozměrném prostoru podle času přijmutí signálu na jednotlivá stanoviště určit, které hyperbola je ta správná. Druhá možnost je, že je známa oblast, která se sleduje, a lze určit výslednou polohu. Poslední možností je existence doplňujících informací od cíle. Ve trojrozměrném prostoru je druhé řešení nereálné, může nastat buď záporná či nelogická výška.

Určení výsledné polohy je více než jasné, tzn. v dvourozměrném prostoru podle příchodu signálu a ve trojrozměrném podle doplňujících informací.

2.1.2.2 Přesnost

Jelikož měření jednotlivých časů příchodů na přijímačích je zatíženo chybou, tak i rozdíl těchto časů je zatížen chybou. Není možné, aby výsledná přesnost byla perfektní. V některých předpisech pro leteckou dopravu se udává, že odchylka může být pro představu v rámci několika desítek metrů [9]. Standardní odchylka v MSS je definována následovně:

$$\sigma_{x,y,z} = \sigma_T * c * PDOP \quad \textit{Rovnice č. 3}$$

kde σ_T značí standardní odchylku měření času příchodu signálu na přijímacích stanicích – většinou je tento parametr pod 10ns a časová diskréta 3,125ns, c je konstanta rychlosti světla a $PDOP$ je Position Dilution of Precision – parametr přesnosti polohy („rozředění přesnosti“ určení polohy). PDOP značí vztah mezi přesností určení polohy cíle a rozmístění (geometrií) stanic. Parametr, jak je zřejmé, závisí pouze na geometrii nastavení celého systému a tudíž jednotlivé stanice a měřící jednotky tento parametr neovlivňují. Podobným způsobem lze definovat i další parametry přesnosti – HDOP (Horizontal DOP) nebo VDOP (Vertical DOP) [1].

Speciálně pro chybu horizontální pozici se využívá ještě jedna hodnota přesnosti – CEPn (Circular Error Probability pro n-percentilovou hladinu spolehlivosti). Využívají se tři základní úrovně spolehlivosti – 50, 90 a 95%, které lze definovat jako

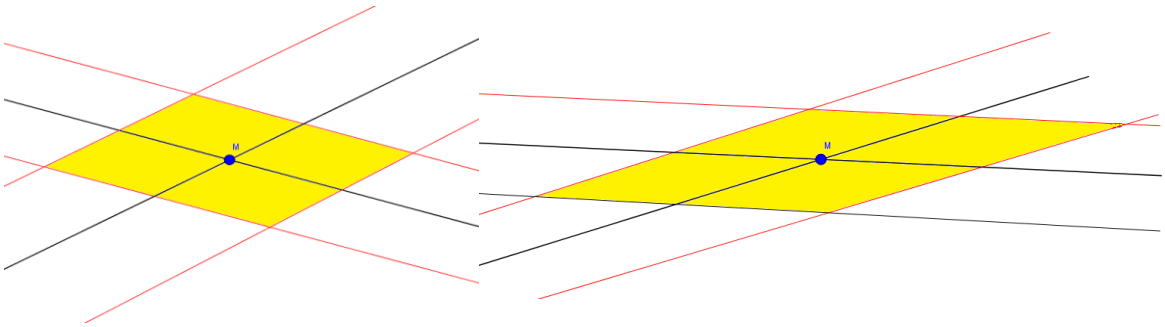
$$CEP_{50} = 0.75 * c * \sigma_{TDOA} * HDOP \qquad \textit{Rovnice č. 4}$$

$$CEP_{90} = 1.60 * c * \sigma_{TDOA} * HDOP \qquad \textit{Rovnice č. 5}$$

$$CEP_{95} = 2.00 * c * \sigma_{TDOA} * HDOP \qquad \textit{Rovnice č. 6}$$

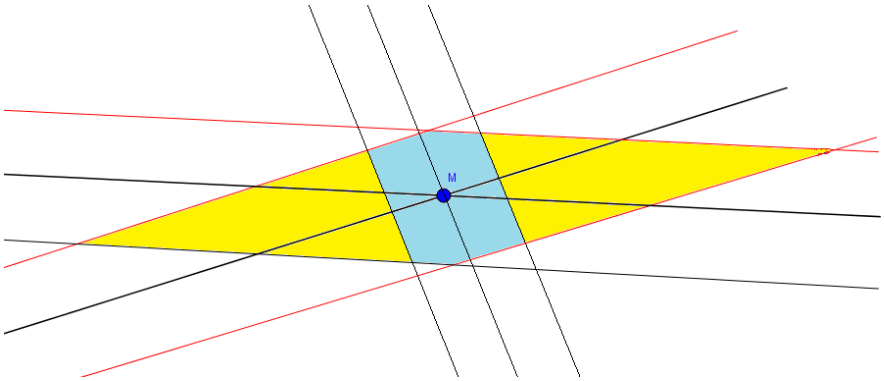
Z předešlých odstavců vyplývá, že se zde objevuje regulární kovarianční matice. Matice je matematickou formulací oblasti nepřesnosti vytvořené průnikem hyperboloidů či hyperbol. Kovarianční matice je základem pro výpočet elipsoidu (elipsy) chyby. V konečném důsledku se projevuje tato nepřesnost jako oblast kolem hyperboly, která značí s cílovou přesností výslednou polohu.

Na následujících obrázcích níže je předvedeno, jak se může lišit přesnost určení polohy cíle na základě polohy letadla či špatného rozmístění stanic.



Obrázek č. 3 - Ukázka dobré HDOP (vlevo) a špatné (vpravo), zdroj: autor

Dosažení menší chyby ve výsledku lze docílit sjednocením další techniky lokalizace pomocí eliptické vzdálenosti. Elipsoid je vytvořen pomocí vysílače a přijímače, které obvykle nejsou na stejném místě a je znám celkový čas mezi dotazem a odpovědí. Tato technika potřebuje znát čas dotazu od vysílače. Tento čas je obvykle dosažen pomocí krátkého dotazovacího signálu na ostatní přijímače. Zlepšení lze vidět na obrázku č. 4 níže.



Obrázek č. 4 - Ukázka špatné HDOP a zpřesnění pomocí eliptické vzdálenosti, zdroj: autor

2.2 Optimalizační algoritmy

V předchozí kapitole byly popsány multilaterační systémy a princip multilaterace. Jak již bylo řečeno, určování výsledné polohy závisí na mnoha faktorech. Mezi nejdůležitější faktory ovlivňující přesnost výsledné polohy je rozmístění jednotlivých stanic v prostoru pro správné pokrytí měřeného úseku prostoru. Na dané rozmístění se lze dívat jako na matematický problém, který je potřeba optimalizovat. Na tento problém lze uplatnit různé optimalizační algoritmy. Optimalizační algoritmy se využívají téměř ve všech odvětví např. v chemii, biologii, komunikacích, sítích, fyzice, ekonomii či ve stavebnictví. V následujících odstavcích budou popsány optimalizační algoritmy obecně a také představeny základní optimalizační algoritmy.

Optimalizační algoritmy slouží k nalezení optimálního řešení daného problému. Využívají se tehdy, když daný problém nemá, spíše není dosud znám, matematický popis řešení problému. Existují také problémy, kde je znám popis, ale bohužel při velkém velikosti prostoru je časově neproveditelné projít celý prostor všech řešení. Cílem algoritmů je prohledat prostor a najít optimální řešení v přijatelném čase. Je zřejmé, že toto řešení, a často to tak bývá, není optimální.

Z matematického hlediska optimalizace je minimalizace nebo maximalizace funkce. Mnoho teoretických úloh i úloh z reálného světa vede k řešení úlohy optimalizace. Tato funkce bývá nazývána objektivní funkce, což je funkce, kterou je potřeba minimalizovat (maximalizovat). Rovnice:

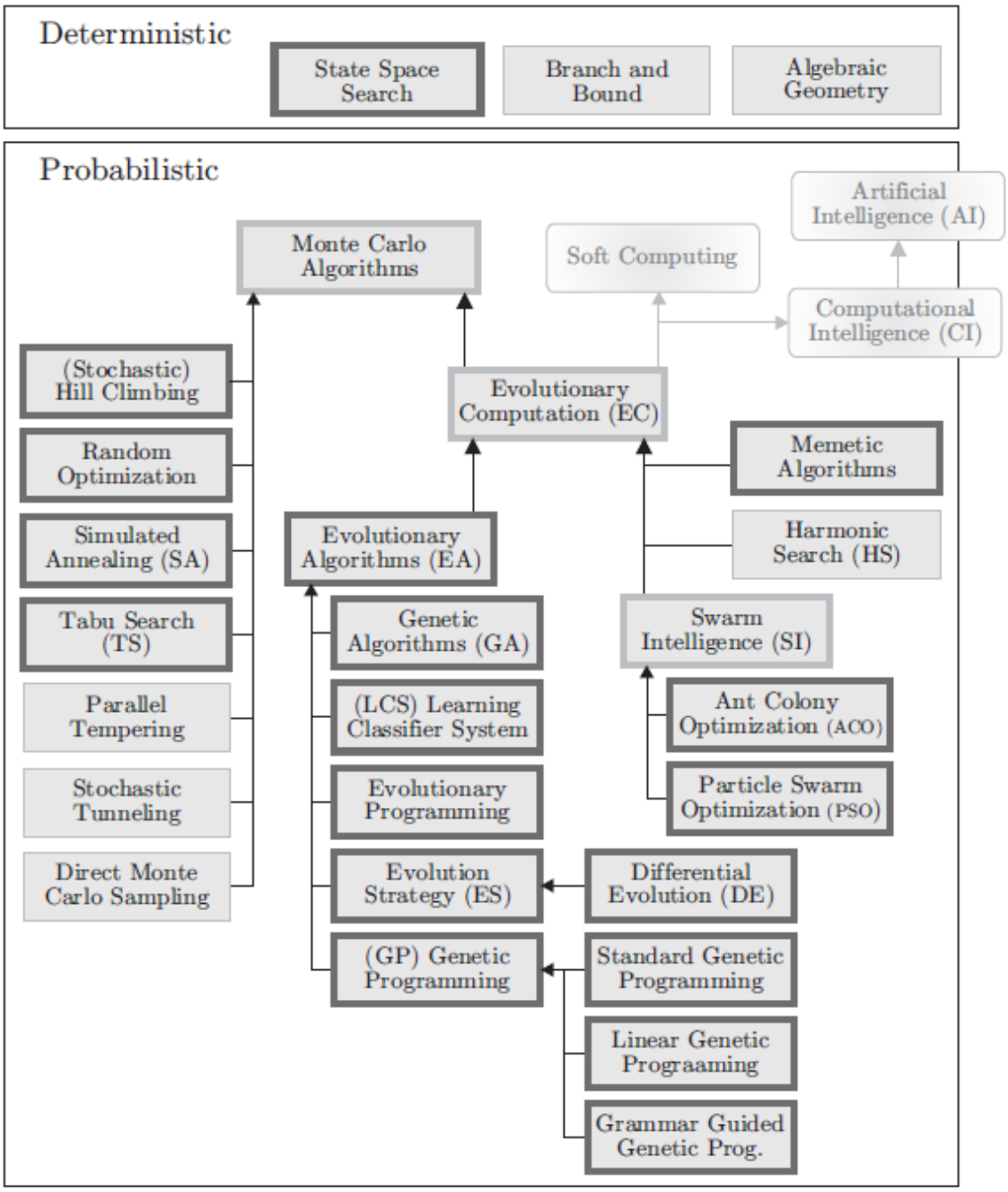
$$f: X \rightarrow Y \; pro \; Y \in R \qquad \textit{Rovnice č. 7}$$

kde Y je z reálných čísel. X zde reprezentuje prostor problému, který může obsahovat nejrůznější typy jako číslo, pole, stavební plán a jiné, podle toho, jaký problém chceme vyřešit [10].

2.2.1 Klasifikace optimalizačních algoritmů

Existuje mnoho různých optimalizačních algoritmů. V této diplomové práce bude nahlédnuto pouze na základní rozdělení a budou představeny pouze základní myšlenky důležitých algoritmů a jejich následné ukázkové využití. Proč vůbec existuje tolik optimalizačních algoritmů? Jednou z odpovědí může být, že existuje

tolik a ještě více různých druhů problémů, kde každý z těchto problému staví různé překážky algoritmu a každý z nich má své vlastní charakteristické problémy [10].



Obrázek č. 5 - Taxonomie optimalizačních algoritmů, zdroj: [10]

Pokud optimalizační algoritmy rozdělíme podle způsobu fungování, lze obecně rozdělit algoritmy na deterministické a pravděpodobnostní. Definice determinismu je vysvětlena v knize Freedom and determinism od autorů Campell, O'Rourke a Shier [11]. Deterministické algoritmy jsou často využívány v případě,

že existuje jednoznačná souvislost mezi vstupy a výstupy pro daný problém. Pro efektivní prohledávání prostoru je možné využít schématu „rozděl a panuj“. Nicméně pokud souvislost mezi vstupy a výstupy nejsou tak zřejmé, jsou velmi komplikované nebo odpovídající prostor je velmi rozsáhlý, tak řešení daného problému deterministicky se stává stále obtížnějším. Při aplikaci deterministických algoritmů na tyto složitější problémy, může to mít za následek dlouhé hledání prostoru i pro relativně malé problémy.

Pro takovéto složité problémy přicházejí na řadu pravděpodobnostní, stochastické algoritmy. Vývojem těchto algoritmů se už započalo před 55 lety. Mnoho matematiků a statistiků prezentovalo své práce [12]. Velmi důležité zde byly algoritmy založené na přístupech Monte Carlo.

Ve zkratce Monte Carlo je náhodný algoritmus, který vždy ukončí svůj proces [13]. Díky této skutečnosti Monte Carlo může vrátit i špatný výsledek. Špatný výsledek v tomto případě neznamená, že algoritmus vrátí nekvalitní či úplně nerelevantní výsledek. Špatný výsledek znamená v daném kontextu, že algoritmus nevrátí globální maximum. Metodu založenou na Monte Carlo využíval ve své práci Petr Hubáček [14], který zdokonaloval měření cíle ve 3D. V jeho práci je dobře popsána celá problematika a ukázka zdrojových kódu v MatLabu. Na druhou stranu, najít o trošku horší výsledek v dostatečně krátkém čase je stále lepší než globálně optimální výsledek, který by trval nalézt 10^{20} let.

Heuristiky používané v optimalizačních algoritmech jsou funkce, které pomáhají rozhodnout, jaké sady z možných řešení budou dále prozkoumávány. U deterministických algoritmů je heuristika taková, že se prohledávají v jasném pořadí jednotlivé řešení, na druhou stranu pravděpodobností algoritmy většinou berou v úvahu pouze některé prvky vybrané podle heuristiky.

Výběr dalších kandidátů se často provádí v náhodném rozložení s využitím statistických údajů získaných ze vzorků z vyhledávacího prostoru na základě různých modelů jako přírodních jevů či fyzikálního procesu. Z příkladu fyzikálního procesu může být optimalizační metoda simulovaného žíhání, kde budoucí kandidát je vybrán za pomocí Boltzmannovy pravděpodobnosti faktoru atomu tuhnutí oceli. Simulované žíhání je vysvětleno v knize Marko Čepina [15]. Z přírodních jevů jsou to Evoluční algoritmy či Mravenčí kolonie. Evoluční algoritmy

kopírují chování přirozeného výběru s tím rozdílem, že kandidáti soutěží ve virtuálním prostředí a generace trvají pouze zlomky vteřin.

Jedny z nejdůležitějších pravděpodobnostních Monte-Carlo metaheuristik jsou Evoluční algoritmy. Tento pojem v sobě zahrnuje všechny algoritmy, které jsou založeny na základě více kandidátů (populaci) a které jsou opakovaně rafinovány (vylepšovány). Tato podtřída je také třídou Soft Computing, většinou známé ohledně umělé inteligence. Evoluční algoritmy budou popsány blíže v následujících kapitolách díky své zajímavosti a dobrým výsledkům. Jak již bylo řečeno, existují metody kopírující fyzikální procesy (simulované žíhání, paralelní temperování či Raindrop metody) a mimo jiné také metody, které nemají přímý model z reálného světa (Tabu Search a Random Optimization) [10].

V předchozích kapitolách i odstavcích byla představena taxonomie optimalizačních metod na základě jejich struktur a základních principů, tudíž z pohledu teorie. Cílového inženýra, uživatele či aplikovatele optimalizačních metod, který chce vyřešit daný problém, zajímá více vlastnost daného algoritmu, jako jsou rychlost a přesnost. Proč jsou zrovna tyto dvě vlastnosti zmiňovány? Přesnost a rychlost jsou ve většině pravděpodobnostních případů protichůdné vlastnosti resp. cíle. Obecnou vlastností či pravidlem je, že pro lepší přesnost výsledku je potřeba nechat algoritmus déle běžet. Vědci se snaží posouvat tuto hranici Paretova optima (viz kapitola 3.2.2.1) pomocí vynalézání nových přístupů či vylepšování stávajících.

2.2.2 Nejlepší výsledek - optimum

Celou dobu jsou vysvětlovány optimalizační metody, které hledají nejlepší možný výsledek. Bylo představeno jednoduché rozdělení těchto metod, ale nikde není vysvětleno, co dělá daný výsledek optimálním.

Optima lze rozdělit podle počtu kritérií na jednokriteriální a vícekriteriální. V případě optimalizace pouze jediného kritéria f je optimální řešení maximum resp. minimum dané funkce [10]. V optimalizaci existuje konvence, která optimalizační problémy nejčastěji definuje jako minimalizaci a pokud je potřeba kritérium f maximalizovat, tak se jednoduše minimalizuje jeho negace ($-f$). Každý si dokáže představit nějakou funkci s lokálními a globálními maximy a minimy. Globální optimum je optimum pro celý prostor a lokální optimum je pouze pro

podmnožinu daného prostoru. Přesné matematické definice lokálních a globálních extrémů jsou blíže popsány knize Handbook of memetic algorithms od autorů Neri, Cotta a Moscata [16].

Jednorozměrné funkce - jedno kritérium - mohou mít více než jedno globální maximum resp. minimum. Příkladem může být funkce kosina, která má své maxima v periodě $2*i*\pi$ a minima $(2*i+1)\pi$ pro celý definiční obor, kde i je z množiny celých čísel. Z předchozího příkladu vyplývá problém, jak daný algoritmus vyhodnotí optimální řešení. Ve správném výsledku optimalizace by se měla objevit množina všech globálních maxim. Bohužel v mnoha řešeních se bude objevovat pouze jedno globální maximum. U jednoho kritéria se obecně hledá maximum resp. minimum dané funkce. Na druhou stranu u vícekriteriálních funkcí je to trochu složitější. Obvykle existuje několik, často nekonečně mnoho optimálních řešení [10].

Nedostatek času a nedostatek paměti v moderních počítačích zapříčiňují, že ve výsledku najdeme pouze konečnou podmnožinu optimálních řešení. Algoritmy mají za úkol najít co nejlepší řešení problému a řešení co nejvíce odlišná od sebe samých [17]. Druhý cíl bude předveden na jednoduchém příkladu, kde $x \in [0,10]$ a existuje nespočetně mnoho řešení. Algoritmus najde řešení $X^*_1 = \{0,0.1,0.11,0.05,0.01\}$ nebo také najde $X^*_2 = \{0,2.5,5,7.5,10\}$. Tyto výsledky jsou pouze malou podmnožinou možných řešení. Nicméně druhý výsledek nám dává širší pohled na optimální sadu řešení a je pro výsledné pochopení problému důležitější.

Bohužel hledání minima a maxima u jednokriteriálních funkcí nestačí na mnoho problému ze skutečného světa. Ve skutečnosti se optimalizační algoritmy aplikují na množiny F skládající se z $n = |F|$ objektivních funkcí f_i a každá z nich reprezentuje jedno kritérium, které má být optimalizováno [18]. Ve výsledku to často znamená hledání kompromisů mezi konfliktními cíli.

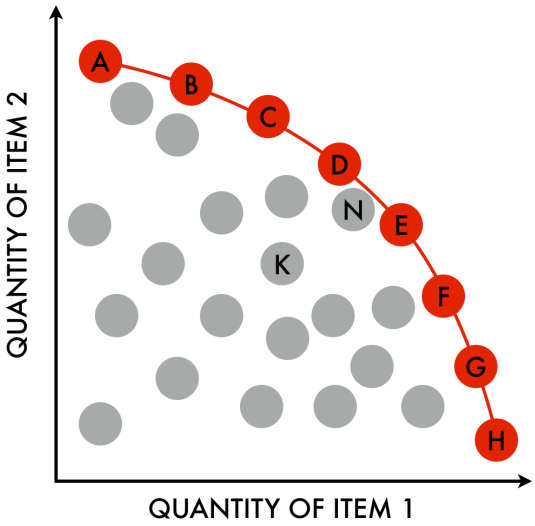
Pro příklad lze uvést optimalizaci výrobního závodu. Existuje několik cílů: minimalizovat čas na výrobu produktu, maximalizovat zisk, minimalizovat náklady, maximalizovat kvalitu výrobků a minimalizovat dopad na životní prostředí. Poslední dva cíle vypadají v rozporu minimalizací nákladů. Tyto protichůdné cíle nejsou vždy zřejmé a tím pádem se také komplikují optimalizační

algoritmy. Příkladů existuje nepřeberné množství a některé z nich jsou z každodenních lidských problémů, aniž bychom si to uvědomovali.

Jedna z nejjednodušších metod, jak určit optimální řešení u multikriteriálních problémů, je přiřadit pro každé kritérium váhu. Touto váhou se následně násobí jednotlivé funkce. Použití vah umožňuje minimalizovat jeden cíl a na druhou stranu maximalizovat cíl druhý. Je zřejmé, že po přiřazení jednotlivých vah lze z multiobjektových problémů zredukovat problém na jednoobjektový [10]. Nevýhodou tohoto přístupu je, že pro různé rychlosti růstu či poklesu jednotlivých funkcí nezvládá objektivně správně určit optimum. Vždy v tomto směru bude jedna funkce zanedbaná vůči druhé. Například na funkcích $f_1(x) = -x^2$ a $f_2(x) = e^{x-2}$ lze vidět, že při řádově nízkých čísel bude f_1 zanedbatelná. Pro takovéto funkce se doporučuje nevyužívat konstantní váhy. Pro jednotlivé funkce by mělo při nejhorším platit stejné O [10]. Velké O je názorně vysvětleno v článku Formalizing O Notation in Isabelle/HOL od autorů Avigad a Donnelly [19]. Často není ani zřejmé, jak se výsledná funkce bude chovat i přes to, že tvar prvotních funkcí je zřejmý. Otázkou ve většině případů zůstává, jak správně nastavit jednotlivé váhy. Optimální řešení vah leží v Paretovu optimu.

2.2.2.1 Paretovo optimum

V předchozích odstavcích byl zmíněn termín Paretovo optimum. Nyní bude problém Paretova optima vysvětlen. Často je tento pojem využíván v oblasti ekonomiky či sociálních věd [20]. Paretovo optimum bylo vymyšleno před 110 lety Wilfredem Paretem [21]. Tato diplomová práce je od těchto oblastí lehce vzdálená a bude tudíž vysvětlena pro kontext právě pročítané diplomové práce. Pareto definoval hranici řešení, která definuje optimální řešení s protichůdnými cíli pomocí kompromisu v této hranici. Funguje na principu nadvlády (dominance), viz obrázek č. 6. Z této hranice (fronty) si algoritmus nebo člověk může vybírat jednotlivá řešení, která jsou nejlépe vyhovující. Redukuje se problém na výběr optimálního řešení pouze z Pareto množiny a nikoli na plný rozsah každého parametru.



Obrázek č. 6 - Paretova množina, kde prvky N a K nepatří do této množiny, zdroj: [22]

Je patrné, že kompletní Paretova množina není často požadovaný výsledek optimalizačních algoritmů. Obvykle středem pozornosti jsou pouze speciálně vybrané oblasti z této množiny. Tato skutečnost je vysvětlena na příkladu mravenčí kolonie [10]. Pouze ve zkratce se v tomto příkladu poukazuje na čistě Paretovu optimalizaci, kde algoritmus najde Paretovu množinu, která obsahuje řešení, která nejsou z logického hlediska výhodná např. mravenec urazí 0 kroků pro 0 jídla atd.

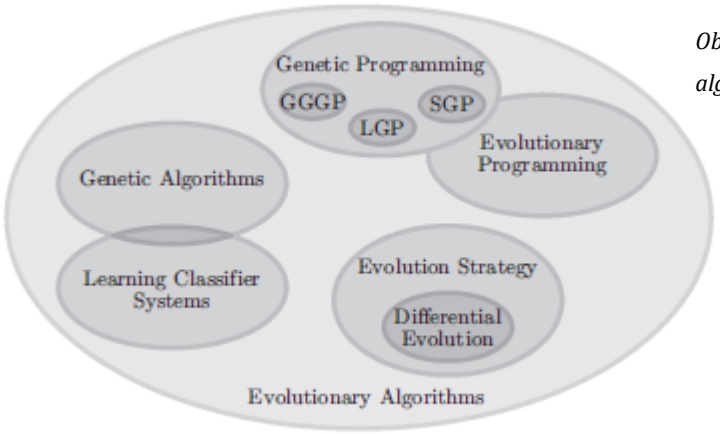
2.2.3 Evoluční algoritmy

Evoluční algoritmy (EA) jsou algoritmy na bázi populace a metaheuristik optimalizačních algoritmů, které používají přírodní mechanismy, jako jsou mutace, křížení, přirozený výběr a přežití nejschopnějších za účelem opakovaného zpřesnění řešení výsledných kandidátů [23].

Charles Darwin (Darwin, Ch. 1859. *The Origin of Species by Means of Natural Selection*. London : John Murray, Albemarle Street, 1859) identifikoval a popisoval principy přirozeného výběru a přežití těch nejschopnějších jedinců. Tento princip a kniha inspirovala mnoho inteligentních lidí a následně celou oblast napodobování těchto přírodních procesů, kterou známe dnes [24]. Nicméně u zrodu evolučních (genetických) algoritmů v algoritmické podobě stál John Henry Holland.

Taxonomie evolučních algoritmů je zahrnuta na obrázku č. 5, kde je zřejmé, že se pod pojmem evoluční algoritmy setkáváme s obsáhlou skupinou optimalizačních metod, které se dále dělí do dalších skupin. Nejčastěji se mluví o

genetických algoritmech, evolučních strategiích, genetickém a evolučním programování. Tyto podskupiny lze vidět na obrázku č. 7. Evoluční algoritmy se jako takové abstrahují z přírodních procesů a zároveň se modifikují k obrazu svému. Ve většině případů jsou více zaměřené na cíl [25].



Obrázek č. 7 - Podmnožiny evolučních algoritmů, zdroj: [10]

2.2.3.1 Jedinec, populace a fitness

Pro budoucí odstavce a diskuze o různých optimalizačních algoritmech je zapotřebí definovat datovou strukturu jedince. V některých algoritmech a metodách, zvláště v evolučních algoritmech, se pracuje s velkým souborem daných jedinců. Tito jedinci dostávají ohodnocení ve formě fitness hodnoty (viz dále), která je často relativní ke všem prvkům z vybrané populace.

V mnoha publikacích se využívá pro jedince označení fenotyp a pro jeho reprezentaci se používá termín genotyp, genom či chromozom [24][26]. Každý jedinec nese s sebou základní informace o sobě samým a často přidané informace v podobě např. hodnoty fitness. Ekvivalentní popis je, že chromozom se dělí na jednotlivé geny, které jsou stejně uspořádány, což znamená, že i-tý gen stejného typu chromozomu reprezentuje stejnou vlastnost. Nejčastější a nejjednodušší reprezentací jedinců v optimalizačních algoritmech je binární řetězec dané délky.

Z kontextu jasně vyplývá, co je populace. Populace je seznam všech jednotlivců použitý při optimalizačním procesu. Populace se postupem času mění a převládají v ní čím dál více lepší jedinci.

Pokud máme dáno, co je optimální řešení problému, je třeba toto optimum také pravděpodobně najít. Jak poznat, že jedno řešení je lepší než ostatní a naopak?

Je potřeba definovat si nějaké srovnávací opatření, které tento problém rozhodne. V mnoha řešení optimalizační algoritmu, zejména v evolučních, je toto opatření reprezentováno reálným číslem [10]. Pro každého kandidáta je spočítáno toto reálné číslo, které reprezentuje způsobilost jedince jako řešení pro danou úlohu. Proces výpočtu fitness hodnoty je reprezentována buď absolutní hodnotou, nebo v mnoha případech hodnotou oproti ostatním jedincům. Díky této skutečnosti se hodnota fitness může během času měnit a má význam pouze uvnitř optimalizačního algoritmu.

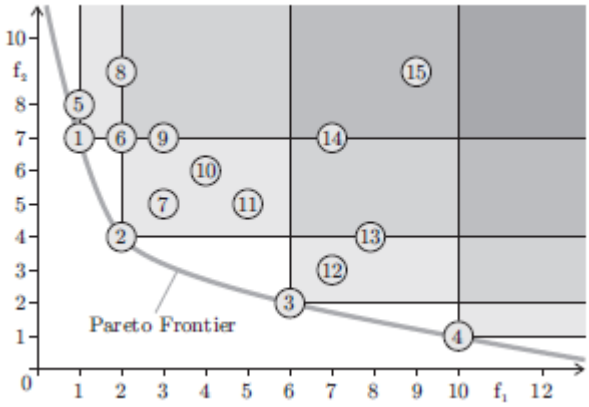
První aplikace genetických algoritmů se zaměřovaly na jednoobjektové optimalizace. Pro tyto aplikace se většinou fitness hodnota rovnala hodnotě funkce. Bohužel tento přístup je v dnešní době zastaralý. V jednoduchých problémech může toto zjednodušení opravdu být použito. U víceobjektových problémů je tato myšlenka nepoužitelná. U multikriteriálních problémů vznikají fitness hodnoty jako vektor objektových hodnot jednotlivých funkcí. Bohužel, jak bylo výše napsáno, mnoho algoritmů pro selekci jednotlivých jedinců nedokáží pracovat s vektorem, nýbrž pouze se skalární hodnotou fitness [10].

Hodnota fitness pro jedince často reflektuje nejen pořadí v populaci, ale také hustotu ve sledované populaci. Tímto způsobem lze zjistit kvalitu daného jedince a následně také celkovou rozmanitost populace. Popsaná skutečnost pomůže zvýšit pravděpodobnost nalezení globálního optima a tím pádem zlepší optimalizační schopnosti algoritmu. V případě mnoha jedinců, kteří mají stejnou hodnotu či nedominují se navzájem (žádný není o poznání lepší), bude informace o hustotě velmi nápomocná a důležitá pro rozmanitost populace. Celá fitness hodnota nezávisí pouze na jedinci samotném, ale také na celé populaci. Z toho vyplývá, že někdy je lepší změnit dobrého jedince za jiného pro rozmanitější populaci a pro lepší prohledávání prostoru.

Určování hodnoty ve vícekriteriálních problémech je obtížné. Bude představeno pouze několik metod, které budou dále detailněji rozebrány. Dalších mnoho metod je sepsáno v knize Thomase Weise [10]. První z nich je nejprimitivnější metoda v daném ohledu - přiřazování vah k jednotlivým objektovým funkcím. Metoda je statická a má stejné nedostatky jako metoda při určování optima (viz kapitola 3.2.2). Je nepoužitelná pro vztahy, kde jedna funkce

dominuje nad druhou. Výsledkem je fitness hodnota ze součtu vah objektových funkcí.

Další v podstatě jednoduchou metodou je počítání hodnoty fitness přímo odrážející Paretovo dominanci. Obrázek č. 6 ilustruje hodnoty objektivních funkci a Paretovu množinu.



Obrázek č. 8 - Ukázka dominujících jedinců s oblastí, nad kterými dominují, zdroj: [10]

Hodnoty fitness jedince se zde přiřazují pomocí množství dalších jedinců, kteří převládají. Názorně je to ukázáno na obrázku č. 8. Jedinci, kteří dominují nad ostatními, budou ve fitness hodnotě lépe ohodnoceni, než ti, kteří jsou dominováni ostatními. Nevýhodou popsaného přístupu je, že metoda podporuje jedince, kteří leží v hustě vyplněné oblasti problému, a diskriminuje jedince v řídce zkoumaných oblastech. Tím, že jsou prohledáváni pouze okrajoví jedinci Paretovy množiny, dochází k tomu, že je získána pouze část nejlepších řešení. V některých případech toto přiřazování vede k předčasné konvergenci na lokální optima. Vylepšený přístup nad stejnou myšlenkou Paretovy dominance je představeno David E. Goldbergem [27]. Ve zkratce myšlenka spočívá v přiřazování nejlepší hodnoty fitness jedincům v Paretově množině. Následně se tito jedinci virtuálně odeberou a vznikne Paretova množina ze zbývajících. V tomto zbytku se znovu určí hodnota fitness, ale o trochu horší hodnotou než v předchozí podmnožině. Tento postup se opakuje, dokud všichni jedinci nemají přiřazenou fitness hodnotu.

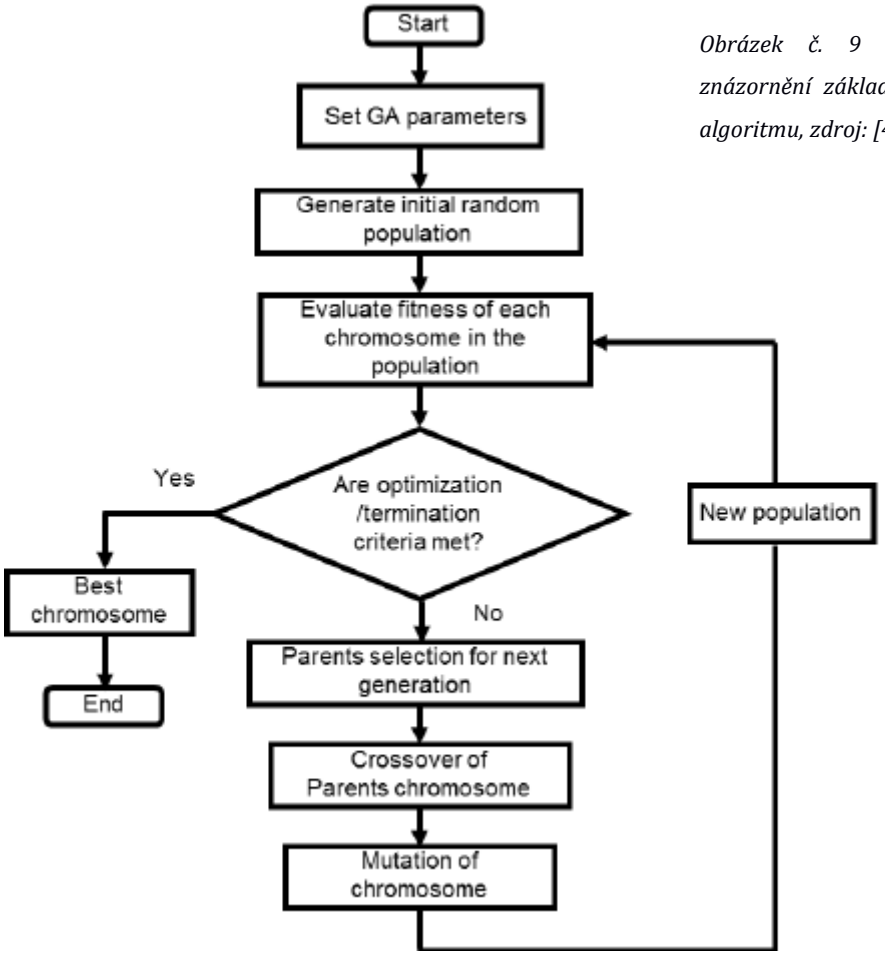
Existuje dále nespočet dalších funkcí pro určení fitness hodnot pro jedince. Využívají se sdílené funkce, které pro určení fitness hodnot používají vzdálenosti od jednotlivých jedinců. Nebo se využívá jednoduché triangulační funkce [28]. Zajímavou metodou je určování pomocí turnaje, kde jedinci mezi sebou soutěží.

Tato metoda se využívá i při selekci jedinců z populace, která je blíže popsána v následujících odstavcích.

2.2.3.2 Fungování evolučních algoritmů

Jak už bylo naznačeno v předchozích kapitolách, lze rozlišovat mezi jednoobjektovými a víceobjektovými evolučními algoritmy. V kontextu diplomové práce budou převážné popisovány pouze evoluční algoritmy optimalizující pouze jedno kritérium.

Základní cyklus fungování všech evolučního algoritmů je v podstatě stejný a je přehledně vyobrazen na obrázku č. 9.



Obrázek č. 9 - Vývojový diagram znázornění základního cyklu evolučního algoritmu, zdroj: [40]

Prvotní krok evolučních algoritmů je vytvoření počáteční populace. Počáteční populace je ve většině případů absolutně náhodná. Druhým krokem je výpočet fitness hodnoty pro každého jedince v populaci. Toto hodnocení může

zahrnovat složité simulace a výpočty a je to stěžejní prvek algoritmů. Následuje proces výběru, který vyfiltruje špatné jedince (mají malou šanci na reprodukci) a umožňuje jedincům s dobrou hodnotou fitness postoupit dále s vyšší pravděpodobností. V dalších odstavcích je tento krok nazýván selekcí. V následné fázi reprodukce vzniká potomstvo, nová populace. Nová populace vzniká změnou nebo kombinací genotypů vybraných jedinců. Tito noví jedinci pak tvoří novou populaci. Kroky, které mění jednotlivé jedince, jsou nazývány křížení a mutace. Tyto dva kroky jsou nejobvyklejší, ale existují i jiné operátory [41]. Posledním krokem je pouze porovnání ukončující podmínky, kde se porovnává výsledek jedinců oproti výsledku, který je dostatečné optimální. V mnoha případech se evoluční algoritmy ukončují po doběhnutí předem stanoveného počtu generací (cyklů).

Při spuštění evolučního algoritmu neexistuje žádná informace o tom, co ve výsledném problému je dobré či špatné. V počáteční populaci jsou prosté náhodné uskupení genů. Autor Thomas Weise [10] to přirovnává ke stejnému problému, jako měli prvotní jednobuněčné organismy před 3,8 miliardami let na Zemi. Dále je zajímavé, že přehledně popisuje problém evoluce a evolučních algoritmů na jednoduchém příkladu - ryba v oceánu. Přežití ryby závisí na její výkonnosti v oceánu. Jedno z kritérií může být velikost ryby. Ačkoliv má velká ryba lepší šanci na přežití, tak velikost nemusí být klíčovým faktorem, pokud je příliš pomalá k lovení a tak dále.

2.2.3.3 Inicializace nové populace

V předchozích odstavcích bylo vysvětleno několik pojmů, které jsou v evolučních algoritmech nutnou součástí - jedinec, populace a fitness funkce (hodnota). Prvním krokem evolučního algoritmu je inicializace počáteční populace. Typicky počáteční populace je zcela náhodná. Tato metoda je nejsnadnější a nejvíce populární metodou inicializace. Nicméně počáteční inicializace populace může přinést nemalý přírůstek úspěchu výsledku a zrychlení celého výběru [29].
Druhou nejjednodušší metodou je vygenerování vice jedinců, než je třeba a vybrat z nich ty nejlepší. Pokud bude potřeba N jedinců, nechá se vygenerovat $5N$ jedinců a vyberou se pouze ti nejlepší.

Další metoda může být využití externích znalostí např. experta v daném oboru, výsledky jiných algoritmů či již publikované výsledky. Tím pádem počáteční populace není náhodnou a je přiměřeně správně nastavena. V knize Evolutionary Optimization Algorithms Dana Simona [29] je demonstrováno, jak moc dokáže počáteční populace změnit a zrychlit evoluční algoritmy.

2.2.3.4 Selekce

Dalším krokem je přiřazení fitness hodnoty, která již byla popsána výše. Následuje výběr nejlepších jedinců z populace - selekce. Výběr se může chovat deterministicky nebo náhodně, záleží na daném problému a implementaci. Obecně platí, že existují dvě třídy výběrových algoritmů: s náhradou a bez náhrady. V algoritmu bez náhrady se každý jedinec z populace bere pouze jednou pro budoucí reprodukci. Podle algoritmu s náhradou se může jedinec brát v úvahu několikrát. Je to podobné jako ve skutečnosti, někteří jedinci mohou mít více potomků. Ve výběrových algoritmech se převážně využívají algoritmu s náhradou.

Výběr správných jedinců k reprodukci má zásadní vliv na výkon evolučních algoritmů. Všechny selekční mechanismy fungují na principu upřednostňování silných a schopnějších jedinců nad slabšími. Na druhou stranu i sebe horší jedinec by měl mít nepatrnou pravděpodobnost výběru. Téměř ve všech algoritmech se přiřazení fitness hodnot jedinců dělá před selekcí a mnoho výběrových algoritmů zakládají svá rozhodnutí pouze na této hodnotě fitness jedince, které mají svá úskalí většinou u víceobjektových problémech [10].

Selekční algoritmy musí brát v potaz i rozmanitost celé populace. Při výběru pouze nejlepších jedinců se tito jedinci budou vyskytovat častěji a velmi brzo převládnou nad ostatními. V některých příkladech to nemusí být na škodu, ale bohužel rozmanitost populace bude nízká a může se stát, že bude konvergovat pouze k sub-optimálnímu řešení [24].

Ve výsledku jsou selekční algoritmy ve většině případů charakterizovány parametry: výběrovým tlakem, výběrovou odchylkou a ztrátou rozmanitosti. Tyto tři parametry nejvíce reprezentují konvergenční vlastnosti algoritmu. Volba parametrů by měla být vyrovnaná, protože při silném tlaku se budou preferovat

silní jedinci, kteří se budou uchylovat k sub optimálnímu řešení, a příliš malý tlak bude mít za následek příliš pomalou konvergenci.

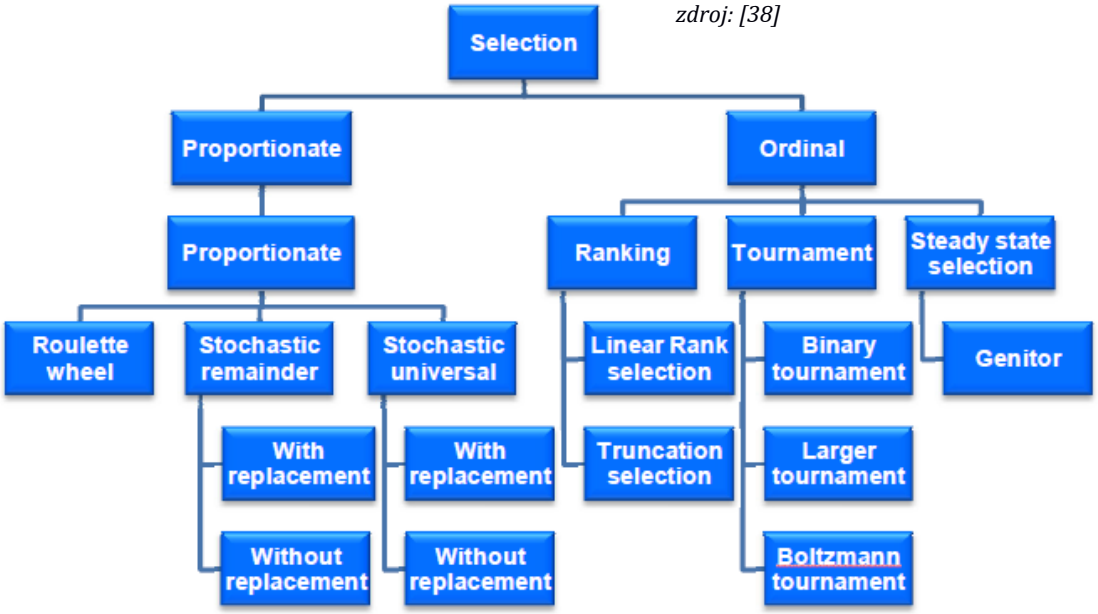
Jedním z nejdůležitějších parametrů výběrových algoritmů je výběrový (selekční)tlak:

$$l = \frac{\overline{F}^{*} - \overline{F}}{\overline{\sigma}} \qquad \textit{Rovnice č. 8}$$

,kde $\overline{F}^{*}$ je průměrná hodnota fitness pro selekci, $\overline{F}$ je průměrná hodnota fitness před selekcí a $\overline{\sigma}$ je rozptyl fitness před selekcí. Parametr značí, do jaké míry se přihlíží na fitness hodnotu jedince, neboli do jaké míry jsou schopnější jedinci zvýhodňováni. Důsledky zvolení selekčního tlaku jsou napsány v předchozím odstavci.

Druhým parametrem je výběrová odchylka, reprezentující očekávanou odchylku fitness hodnoty po použití selekčního algoritmu do normalizovaného Gaussova rozdělení. Ztráta rozmanitosti reprezentuje podíl jedinců, kteří jsou jednou vybráni během algoritmu. Dalším parametrem je výběrová intenzita. Je to očekávaná průměrná hodnota fitness populace po aplikování algoritmu pomocí jednotkového normálního rozdělení. Měří sílu výběrového tlaku algoritmu.

Obrázek č. 10 - Dělení selekčních algoritmů, zdroj: [38]



Nejčastějším výběrovým algoritmem je algoritmus pomocí ruletového mechanismu (roulette-wheel selection). Je jasné, jak funguje ruleta ve skutečnosti.

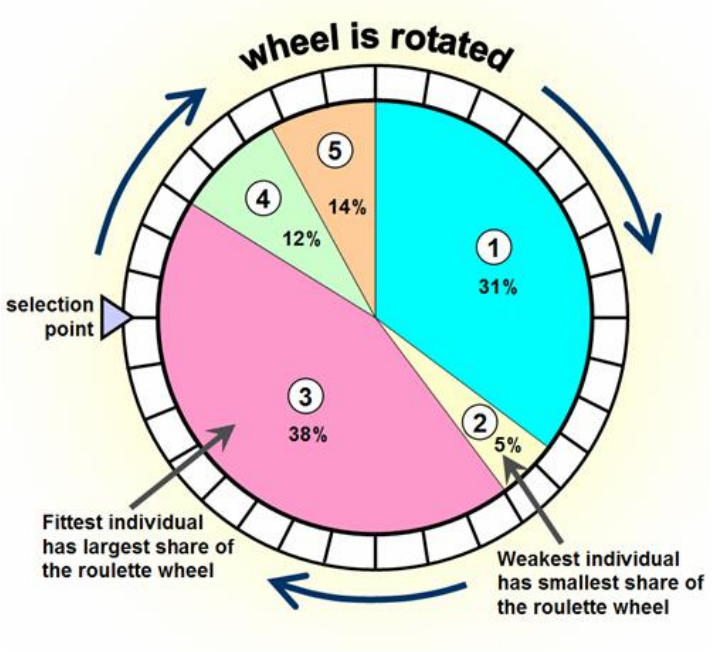
V ruletě má každé číslo stejnou pravděpodobnost na zastavení kuličky. Pokud by přesně tento princip byl použit v evolučních algoritmech, tak by každý jedinec měl stejnou šanci na výběr, což je v kontrastu v předchozích kapitolách a celého Darwinismu, kde schopnější jedinec by měl mít větší pravděpodobnost přežití. Ruletový mechanismus v evolučních algoritmech funguje na principu, že kvalitnější, schopnější jedinci mají v ruletovém kole větší výseč či prostor a tudíž mají větší šanci na výsledný výběr. I ten nejhorší jedinec v populaci má šanci být vybrán k reprodukci, nicméně jeho výseč je nejmenší a tudíž jeho pravděpodobnost na výběr bude také nejmenší.

Existují mnoho výběrových algoritmů a ve většině případů se liší pouze ve způsobu určování velikostí výsečí. Nejklasičtější určování výseče je pomocí fitness hodnoty daného jedince (fitness-proportionate selection) [24]. Metoda byla představena již průkopníkem evolučních algoritmů Johnem H. Hollandem [30]. Metoda funguje na porovnání fitness hodnoty jedince v (p. x) oproti součtu fitness hodnot v celé populaci. Tudíž, pravděpodobnost vybrání jedince P (p1) je:

$$P(p_1) = \frac{v(p_1.x)}{\sum_{\forall p_2 \in Pop} v(p_2.x)} \qquad \textit{Rovnice č. 9}$$

Existuje celá řada metod, které realizují tento mechanismus. Za zmínku stojí stochastická zbytková selekce nebo stochastická univerzální selekce [10]. Nicméně nejčastější klasickou metodou je Monte-Carlo ruletová selekce, která určuje výseč přímo úměrně velikosti fitness hodnoty. Výsledný jedinec je v informačním světě vybrán pomocí generátoru (pseudo)náhodných čísel. Tento postup se opakuje do naplnění celé populace.

Selekční mechanismus pomocí ruletového kol s výsečí podle fitness hodnoty je vyobrazeno na obrázku č. 11. Lze se povšimnou, že největší výseč má jedinec s nejlepší hodnotou fitness funkce. Nejslabší jedinec má nejmenší výseč z rulety. Selection point (*S*) reprezentuje náhodné číslo od nuly do součtu všech fitness hodnot. Z programového hlediska je jedinec následně vybrán pomocí postupného přičítání jednotlivých fitness hodnot jedinců, dokud tato suma není větší než náhodně číslo (*S*).



Obrázek č. 11 - Výběrový mechanismus pomocí ruletového kola, zdroj: [38]

Dalším výběrovým algoritmem je turnajová selekce, která patří mezi populární a efektivní algoritmy. V tomto algoritmu je vybráno k jedinců z populace a jsou porovnávány mezi s sebou v turnaji. Vítěz tohoto turnaje je vybrán k budoucí reprodukci. Je to znovu podobné jako v přírodě, kde se některé druhy musí vzájemně utkat pro budoucí reprodukci [24]. S rostoucím *k* se zvětšuje výběrový tlak, jedinci s dobrou fitness hodnotou vytvoří více a více potomků a jedinci s horší hodnotou budou pořád klesat, čímž klesá rozmanitost celé populace. V mnoha případech si vybírá velikost turnaje dva (*k=2*). Pro každý turnaj jsou jedinci vybráni náhodně pomocí rovnoměrného rozložení a vítězové se posouvají dál k reprodukci. Každý jedinec bude v průměru vybrán do dvou turnajů. Nejlepší v řešení bude ten, který vyhrál oba turnaje a v reprodukci bude v průměru také dvakrát. Průměrný jedinec bude lepší než 50% populace, ale také horší než druhá polovina populace. Nejhorší jedinec v tomto případě ztrácí šanci na postup. Může se stát, že v některých z turnajových selekcí i nejhorší jedinec postoupí k reprodukci a to v případě, kdy v turnaji nastoupí sám proti sobě. V některých případech se přiřazuje pravděpodobnost vybrání slabšího jedince oproti silnějšímu [24], tudíž slabší jedinec porazí silnějšího.

Poslední z představovaných výběrových algoritmů bude algoritmus podle pořadí jedinců (rank selection). V tomto algoritmu pravděpodobnost reprodukce

jedince je podle své fitness a jeho výsledném pořadí v populaci. Oproti mechanismu ruletového kola, kde čím schopnější jedinec, tím větší pravděpodobnost k výběru, v tomto algoritmu neplatí a pravděpodobnost nejschopnějšího jedince v populaci je pořád stejná. Výhodou algoritmu je, že algoritmus napomáhá udržovat selektivní tlak i ke konci běhu, kdy v populaci jsou pouze zdatní jedinci a rozdíly hodnot fitness jsou již nepatrné [24]. Na druhou stranu, často se může výhoda stát nevýhodou. Pokud v populaci jsou jen nepatrné rozdíly mezi jednotlivými jedinci, tak tento algoritmus může udělat rozdíl několika procent mezi nejlepším a nejhorším jedincem, kde díky pořadí, bude tento rozdíl N-násobný, kde N je velikost populace [24]. Naneštěstí lze při tomto nedostatku transformovat výslednou množinu.

Zpomalení celého procesu optimalizace díky tomuto algoritmu je jasná. Nejschopnější jedinci se hůře prosazují v algoritmu díky ignorování rozdílu hodnot fitness v poměru k ostatním jedinců. Nicméně pro rozmanitost populace po dobu celého procesu optimalizace je to nutná daň.

V předchozích odstavcích bylo představeno několik výběrových algoritmů, které lze využít v evolučních algoritmech. Bohužel nelze určit, který mechanismus výběru je nejlepší. Goldberg a Deb [24] provedli důkladnou analýzu jednotlivých algoritmů a dospěli k názoru, že vhodnou volbou parametrů lze docílit srovnatelných výsledků.

2.2.3.5 Reprodukce

V předchozí kapitole bylo vysvětleno, jak se vybírají nejschopnější jedinci z populace a často bylo zmíněno, jak se postupuje dále k reprodukci. Evoluční algoritmus vychází z informací získaných v kroku t (selekce) pro tvoru nových jedinců v kroku $t + 1$. Existují různé metody, jak docílit tohoto výsledku a mnoho z nich mají přímý ekvivalent v přírodě. Existují čtyři základní operace:

1. Vytvoření, které nemá přírodní ekvivalent. Metoda jednoduše vytváří nového jedince bez žádné vazby na svého předka či rodiče. Není žádné porovnání z předešlé generace a dalo by přirovnat k prvoplození na světě (abiogeneze).

2. Duplikace, která je podobná k buněčnému dělení. Vznikají dva jedinci podobné, ne-li totožné, jednomu rodiči.
3. Mutace, která má v přírodě přesný ekvivalent se stejným názvem. Mutace je malá, náhodná změna v genotypu jedince.
4. Poslední operací je rekombinace, spíše známá jako křížení. Ze dvou rodičů vzniká nový jedinec s podobnými vlastnosti z obou předků.

Při inicializaci populace v evolučních algoritmech neexistuje žádná informace o předešlých jedincích, protože žádní neexistují. Z tohoto důvodu nelze využít nic jiného než operaci vytvoření. Existují výjimky, kdy se prvotní populace generuje jiným způsobem, viz výše.

Duplikace je prosté zkopírování předka do následníka. Tento stav nastává, pokud ani jedna dále z často využívaných operací (mutace a rekombinace) nejsou použity. Jediný důvod, který duplikace plní, je zvýšení podílu daného jedince v populaci.

Mutace jsou nezbytnou součástí evolučních algoritmů a evoluce samotné. Mutace se také označuje jako nepohlavní způsob. Mutace má v evolučních algoritmech malou pravděpodobnost výskytu. Je to v rozmezí pouze několika procent (1-5%) [24]. Při velké pravděpodobnosti mutace algoritmus ztrácí svoje výhody a mění se do primitivního náhodného hledání (random search) [31]. Mění se ve většině případů jen malá část jedince nebo několik málo hodnot jedince oproti původnímu stavu.

Účelem mutace je zachování rozmanitosti populace. Bez mutace by se často stávalo, že evoluční algoritmus by konvergoval k sub-optimálnímu řešení a rozmanitost celé populace by velmi prudce klesala. Populace by byla čím dál více podobná nejlepším jedinců předchozích populací. Aplikace mutace je závislá přímo na problému. U binárních jedinců se mění bit z nuly na jedničku a obráceně. U jedinců s reálnými hodnotami se může přičítat či odečítat náhodně reálné číslo atd. Existují několik druhů mutací a nejčastější je klasická změna chromozomu na stejné pozici, přehození chromozomu (např. hodnota na třetí pozici prohozena s hodnotou na šesté) nebo proházení chromozomu (vybere se řada tří hodnot a náhodně se zamíchají). V reálné reprezentaci chromozomu reálné hodnoty často

mutují pomocí Creep mutace, kde k dané hodnotě se přičítá náhodná hodnota z normálního rozdělení [32].

Poslední operací je rekombinace. Rekombinace je obecnější pojem a představuje více informací než křížení, ale v kontextu k evolučním algoritmům se nejčastěji využívá pojem křížení. Křížení je základní strategii pro vytvoření nové schopnější populace. Čím lepší a vhodnější metoda křížení, tím rychleji by optimalizační algoritmus mohl skončit. Volba optimálního výpočtu fitness funkce a metody křížení jedinců mají jeden z klíčových dopadů na výkonnost algoritmu.

Křížení probíhá u dvou rodičovských jedinců - je to sexuální způsob reprodukce. Křížení má na rozdíl od mutace velkou pravděpodobnost výskytu v rozmezí od 0,75-1. Křížení funguje na principu výměny části chromozomů mezi dvěma a více jedinci a vznikají dva a více potomků, kteří nesou genetický materiál rodičů.

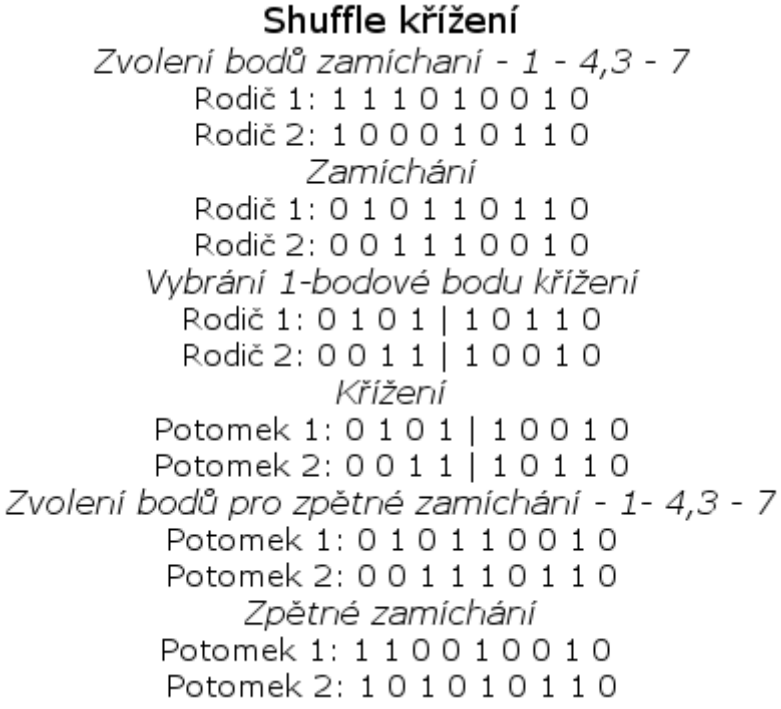
Stejně jako u mutace existuje mnoho odlišných způsobů křížení. Předpokládám, že nejjednodušší způsob křížení je jednobodové křížení. Metoda využívá pouze jediný bod fragmentaci rodičů a následné prohození těchto fragmentů k vytvoření nových jedinců. Po výběru rodičů se náhodně vybere libovolný bod křížení. Noví jedinci tedy vznikají kombinováním rodičů u bodu křížení [33] 1 do *n-1*, protože by se celé křížení stalo zbytečným. Nedošlo by k žádné výměně genetické informace a noví jedinci by byli kopiemi rodičů [24].

Modifikací předchozího způsobu křížení je k bodové křížení. Je podobné předchozímu, ale pro lepší rekombinaci se vybere více bodů křížení (*k*). Vyberou se dva rodiči a následně jsou náhodně určeny body křížení. Noví jedinci jsou vytvářeny kombinací rodičů přes tyto body.

Způsob jakým nejčastěji budou kříženi jedinci v praktické časti je Uniformní křížení. Funguje na principu náhodného výběru genu od jednoho z rodičů. Tímto způsobem se zachovává uniformita v kombinování genů. Pomocí náhody u každého genu se rozhoduje, zda potomek dostane gen od prvního nebo od druhého rodiče.

Posledním způsobem, který bude představen, je Náhodné (shuffle) křížení. Tento způsob pomáhá při tvorbě nové populace, které mají nezávislé body křížení v rodičích. Používá se stejná technika jako u jednobodového křížení a přidá se

zamíchání. Vyberou se dva rodiče pro reprodukci. Nejdříve se zamíchají náhodně geny obou rodičů, ale pozor, stejným způsobem u obou dvou. Pak se aplikuje jednobodové křížení a následně se zkombinují tyto fragmenty. Po provedení jednobodové křížení u potomků jsou zpátky seřazeny stejným způsobem, jak byly zamíchány. Toto křížení se těžko představuje, ale na obrázku č. 12 je přehledně ukázán princip. Výhoda tohoto křížení je, že se odstraní poziční zaujatost, protože pro každé křížení se zde znovu náhodně vybírají jednotlivé pozice.



Obrázek č. 12 - Ukázka shuffle křížení, zdroj: autor

Existuje několik dalších způsobů křížení v evolučních algoritmech. Mnoho dalších způsobů je uvedeno v článku Crossover operators in genetic algorithms Soni a Kumara [33].

Při nebinárním charakteru chromozomu, např. z množiny reálných čísel, je zajímavé sledovat, že tato reprezentace umožňuje relativně snadno definovat smysluplné a konkrétní problémy odpovídající speciálním genetickým operátorům. Mohou tak vznikat potomci pomocí průměru rodičů, odmocninami či nejrůznějšími matematickými operacemi. Reprezentace chromozomů reálnými čísly je skutečně bohatá na různé operátory [24].

2.3 Materiály ERA

ERA a.s. je společnost, která má jednu z předních vývojářských technologií multilaterace a Automatic dependent surveillance – broadcast (ADS-B) pro sledování a řízení letového provozu pro komerční, vojenské a bezpečnostní oblasti. ERA a.s. má více než 100 instalací v řadě letišť v 59 zemích na téměř všech kontinentech. Firma působí v této oblasti již přes padesát let.

Pro dosažení cíle diplomové práce bylo společností ERA a.s. poskytnuto veřejné rozhraní knihovny pro výpočet přesností v Mission Planning System (MIPS) a program s již implementovaným rozhraním pro výpočet přesnosti. V dokumentaci je popsáno, že knihovna byla připravována s ohledem na dvě myšlenky: binární kompatibilita a minimalita rozhraní. Následující odstavce jsou čerpány z dokumentace [34].

Binární kompatibilita znamená to, že běh programu nové verze knihovny pokračuje bez nutnosti překompilování. Knihovna nepotřebuje žádné modifikace či kompilace pro svůj běh na nové verzi. Binarní kompatibilita může být i na úrovni hardwaru a počítače jako takového [34]. Existují i zdrojové kompatibilní knihovny, které nepotřebují žádné modifikace, ale pouze jen rekompilaci. Tento přístup umožňuje snadnější bug-fixing, reagování na nové požadavky či napomáhá snadnější distribuci softwaru. Dále binární kompatibilita umožňuje využití interních knihoven ERA bez nutnosti jejich zveřejňování (typicky jsou includovány až ve zdrojových souborech tříd). Nevýhodou je menší možnost upravování zdrojového kódu např. není možné měnit hierarchii tříd či mazat třídy. Seznam kompletního listu, co lze upravovat, je přehledně ukázáno na webu KDE Community [35].

Minimalitou rozhraní se rozumí, že rozhraní obsahuje pouze to, co je nutné ke splnění požadavků na knihovnu pro výpočet přesnosti. Z popsaného důvodu je v maximální míře uplatněna restriktivní politika v rozhraní, kdy má její uživatel přesně definované konstrukce, které má povoleno využívat. V mém popisovaném případě to znamená absenci veřejného copy konstruktorů či absenci některých get method aj. Cílem je v maximální míře zamezit nezamýšlených použití knihovny a to - pokud je to možné - už na úrovni kompilátoru/linkeru [34].

Knihovna využívá standardní verzi C++11 pro programovací jazyk C++. V dnešní době je aktuální standardní verze C++14 a v tomto roce bude představena nová verze C++17 [35]. Tento standard již podporují v současnosti téměř všechny kompilátory. Autor [34] doporučuje využít nad operačním systémem Microsoft Windows vývojové prostředí Microsoft Visual Studio 2012. Standardem předávání nulových pointerů je literál nullptr.

2.3.1 Struktura knihovny pro výpočet přesností v MIPS

V dalších odstavcích budou nastíněny základní stavební kameny této knihovny. Třídy *CArray<class T>, vector<unsigned int Size, Variance* reprezentují základní stavební prvky, které ukládají data a předávají data do dalších tříd. Všechny splňují binarní kompatibilitu.

Následně skupina tříd, které reprezentují jednotlivé položky reálných multilateračních systémů. Reprezentují softwarovou reprezentaci reálných fyzických komponent. Vlastnosti těchto tříd se nebudou v průběhu výpočtu měnit. To této skupiny patří *Mss::Receiver, Mss:: Interrogator a* Mss::Transponder.

Do další skupiny patří *Mss::Reception, Mss::Interrogation* a *Mss::HeightMeasurement.* Skupina reprezentuje proměnlivé veličiny. Popisují vstupy či parametry, které vstupují do výpočtů a které se se změnou pozice cíle mění.

Poslední podskupinou tříd, které budou popsány, jsou třídy zajišťující výpočetní stránku knihovny. Tuto skupinu reprezentují třídy *Mss::ComputationSettings* a *Mss::MssError.*

Třída CArray<class T>

První třída je *CArray<class T>*, která slouží převážné k předávání instancí *std::vector<T>*. Správné fungování je zaručeno pouze pro inicializaci instance šablony vyskytující se ve veřejném rozhraní knihovny.

Třída Vector<unsigned int Size>

Třída *vector<unsigned int Size>* slouží pro předání vektorů či bodů v Eukleidovském prostoru. Ve skutečnosti se jedná o použití standardní šablony

std::array<double, size>, kde *Size* je parametr šablony a přidané jsou dva konstruktory a dvě metody.

Třída Variance

Poslední třídou skupiny pro práci s daty je Variance, která se stará o předávání rozptylu náhodné veličiny. Motivací programátorů bylo zabalení primitivního typu double do vlastní třídy, podle pravidla „*Make interfaces easy to use correctly and hard to use incorrectly*“ od Scotta Meyerse [34]. Třída poskytuje konstruktor, který vytváří instanci na základě standardní odchylky, což je druhá odmocnina rozptylu.

Třída Mss::Receiver

Třída *Mss::Receiver* reprezentuje přijímač a jeho neměnné vlastnosti. U přijímače se sleduje pouze jeho pozice v geocentrických souřadnicích.

Třída Mss:: Interrogator

Třída *Mss:: Interrogator* reprezentuje dotazovač. Jsou sledovány pouze pozice v geocentrických souřadnicích a hardwarové chyby dotazování.

Třída Mss::Transponder

Poslední třídou v reprezentaci fyzických komponent je *Mss::Transponder*. Třída reprezentuje odpovídač. Vlastnosti, které jsou pozorovány, jsou nejistota systematického zpoždění transpondéru. Ta je vyjádřena rozptylem.

Třída Mss::Reception

První třídou ve skupině pro reprezentaci proměnlivých veličin je *Mss::Reception*, popisující příjem signálu na dané stanici. Pro výpočet se nastavuje na daný přijímač, nejistotu zpoždění signálu způsobeného atmosférou po cestě a náhodnou chybou měření TOA.

Třída Mss::Interrogation

Třída *Mss::Interrogation* reprezentuje dotaz provedeného z daného dotazovače. Vlastnosti, které je potřeba nastavit jsou: odkaz na daný dotazovač, nejistotu zpoždění a náhodná chyba odpovědi.

Třída Mss::HeightMeasurement

Poslední třídou, která bude popsána v této skupině, je *Mss::HeightMeasurement*. Reprezentuje externí měření výšky na základě dekódované zprávy s barometrickou výškou. Nastavuje se rozptyl měření výšky a lokální směr výšky (normalizovaný vektor určující směr, ve kterém výška nejrychleji roste).

Třída Mss::ComputationSettings

První třídou pro výpočet je *Mss::ComputationSettings*, která reprezentuje nastavení všech vstupů pro výpočet přesnosti systému v daném bodě. Nastavuje se pozice cíle, pointer na instanci odpovídače, množina možných příjmů pro danou pozici (viditelné stanice v dosahu odpovídače), množina možných příjmů dotazů na daný cíl a pointer na instanci měření výšky. Pokud měření výšky není k dispozici, předává se nullptr.

Třída Mss::MssError

Poslední třídou představenou v této knihovně je *Mss::MssError*. Tato třída reprezentuje celý výpočetní modul. Pro daný bod rastru předpokládá existenci nastavení instance předchozí třídy (*Mss::ComputationSettings*) a následné předání pointeru na tuto instanci do metody *Precompute*. Následné volání *Compute*-metod poskytuje poziční RMS (3D RMS - root mean square), RMS v rovině dané jednotkovou normálou (2D) a RMS ve směru daném jednotkovým směrovým vektorem (1D). Z důvodů možných změn v implementaci není možné předávat referenci (&) *ComputationSetting* za účelem toho, aby v celém procesu výpočtu byla pořád stejná podoba daného nastavení.

2.3.2 Program Analyzer

V předchozích odstavcích byla popsána část knihovny od firmy ERA a.s. Ale pouze ta část knihovny relevantní k diplomové práci. Víme, že knihovna obsahuje mnohem více funkcionalit pro výpočet přesností v různých systémech. ERA a.s. má ještě jeden program.

Implementovaný program bude pojmenován Analyzer. Analyzer je ve formátu exe. Podle popisu firmy ERA implementuje předešlou knihovnu pro výpočet přesností.

Program vyžaduje konfigurační soubor s příponou pcf. Pcf soubory jsou klasické konfigurační soubory, které lze otevřít v libovolném textovém editoru. Konfigurační soubor v sobě nese základní informace o struktuře multilateračního systému (podobné jako v knihovně ERA a.s.) a je popsána v předešlých odstavcích. V souboru je zmíněn typ výpočetního algoritmu, ve kterém si lze vybrat různá, především stará řešení. Nastavují se zde polohy přijímačů a dotazovačů. V souboru jsou nastavení i pro jednotlivé scénáře, rozsah a přesnost daného MSS. Detailnější popis bude v praktické části diplomové práce, kde budou blíže představeny hodnoty konfiguračního souboru a hlavně hodnoty, na které budou aplikovány optimalizační algoritmy.

Po zapnutí programu přes příkazový řádek s nastaveným konfiguračním souborem se výsledek exportuje do souboru s příponou pcg. V podstatě tento soubor je csv soubor reprezentující mřížku daného definovaného prostoru a každá hodnota reprezentuje přesnost v daném bodě.

3 Praktická část

V předchozích částech diplomové práce byla představena základní charakteristika evolučních algoritmů obecně. Byly popsány teoretická východiska, jednotlivé kroky a princip fungování algoritmů. V následujících kapitolách budou tyto znalosti aplikovány na reálný problém - rozmístění rádiových stanic. Cílem bude najít optimální rozmístění stanic pro nejpřesnější určení polohy cíle.

3.1 Implementace optimalizačního algoritmu

V této části bude popsán současný stav řešené problematiky a bude navrhnuto nové řešení, které by mělo umožnit získávat lepší výsledek za lepší časovou jednotku.

3.1.1 Současný stav

Již dříve bylo zmíněno, že geometrie rozmístění rádiových stanic má velký dopad na celkovou přesnost multilateračního systému. V současné době se rozmístění rádiových stanic dělá empiricky na základě zkušeností daných znalostních expertů.

Tento přístup má několik nevýhod. Jednou z nejzásadnějších je, že jednotliví experti mohou mít, a často to tak bývá, jinou úroveň znalostí a zkušeností. Pokud je dán problém dvěma různým lidem, oba vrátí jiné rozmístění stanic. Tímto je docíleno nerovnoměrně kvalitních výsledků rozmístění. Popsaný fakt zapříčinil, že firma ERA a.s. chce tento postup optimalizovat.

V mnoha případech existují nějaké normy přesnosti, které se musí splnit. Po pečlivém výběru příslušného rozmístění expert vloží dané rozmístění do interního programu pro výpočet přesnosti a zjistí, jestli rozmístění opravdu splňuje danou normu. Pokud nesplňuje, je nutné, celý proces opakovat a vyzkoušet jiné rozmístění.

Tento postup je velice zdlouhavý a nepříliš efektivní. V dnešní informační době je tento přístup k problému zastaralý. Firma ERA a. s. se již pokoušela tento přístup změnit. Nicméně, předešlé pokusy tento proces zautomatizovat nepřinesly uspokojivé výsledky. Bohužel podklady z předchozích pokusů nebyly poskytnuty.

3.1.2 Implementace

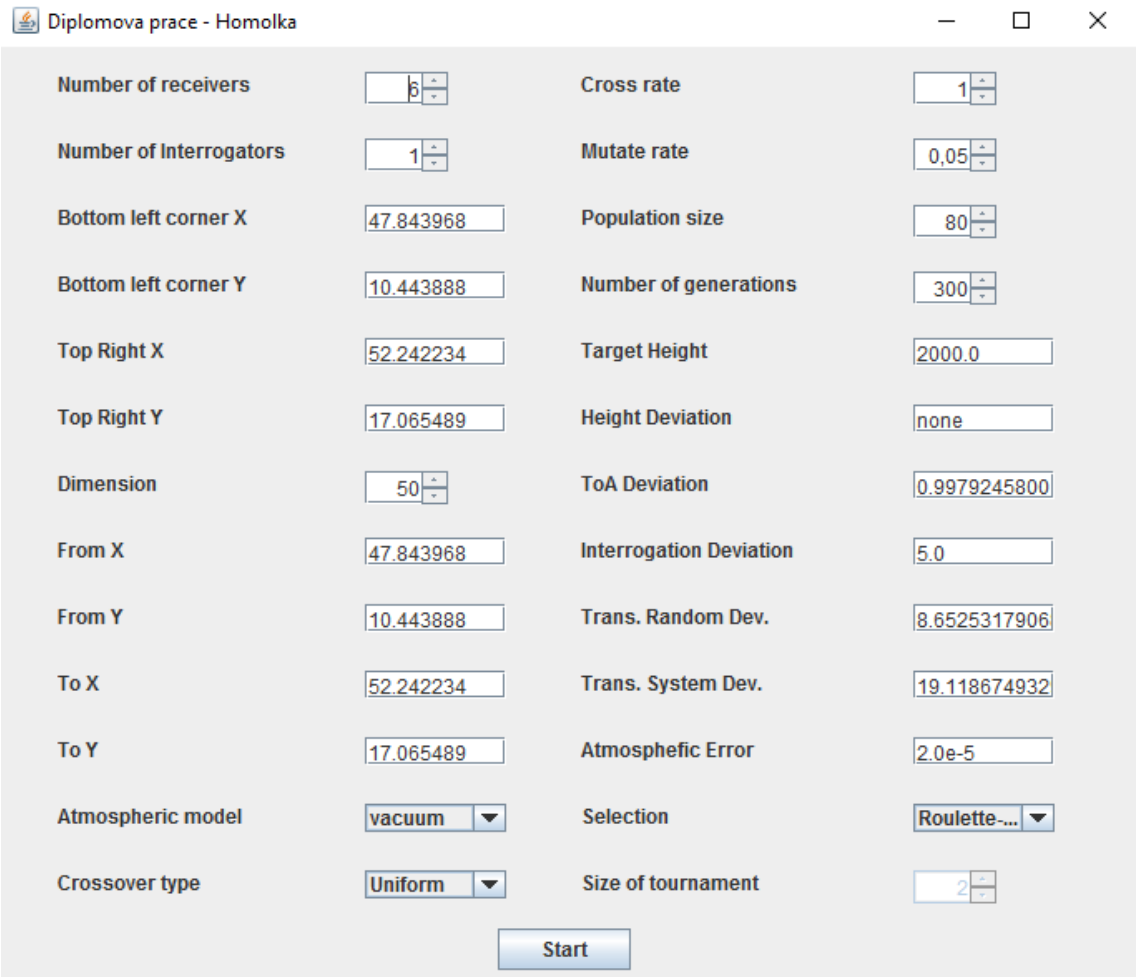
Z předchozí kapitoly je zřejmé, že dosavadní přístup není optimální a cílem diplomové práce je navrhnout optimalizační program. Výsledný program nebude nahrazovat daného experta, nýbrž mu ulehčovat práci. Program bude semi-autonomní, kde některé procesní stavy bude muset stále vykonávat daný expert.

Program jako takový bude pouze optimalizovat rozmístění a navrhne své rozmístění, které považuje za optimální. Na expertovi bude nastavování jednotlivých parametrů (viz dále) a výsledná verifikace daného rozmístění. Program pouze navrhne své řešení a expert s ním bude či nebude spokojen. Pro představu, program navrhne pozici, kde v reálném světě nelze umístit rádiovou stanici například kvůli existenci vodní plochy, skály či močálu. Z toho vyplývá, že program nebere v úvahu kartografické informace či reliéf terénu. Díky této informaci je zřejmé, že algoritmus nebude podávat stoprocentní výsledky. V budoucích příkladech se vždy jednotlivé stanice rozmísťují s předem danou výškou (300 m n. m.). Výška byla doporučena na základě prvotní domluvy s firmou ERA a.s.

Přínosem programu bude zjednodušení rutinního procesu s možností nálezu lepších výsledků či alternativních výsledků, které by expert nebral v úvahu. V praktické části bude popsán výsledný program pro optimalizaci rozmístění rádiových stanic. Nejprve bude program představen z hlediska grafického rozhraní. Následně se software rozebere z hlediska vnitřní struktury, z pohledu programátorského.

3.1.2.1 Grafické rozhraní

Celý program je tvořen pouze jedním hlavním oknem respektive dvěma, kde druhé okno reprezentuje výsledek algoritmu. Jedno okno bylo vybráno pro jednoduchý přehled všech vstupujících hodnot do výpočtu. Toto okno slouží pro nastavování jednotlivých parametrů pro následující výpočet. Parametrů pro specifikaci problému je zde poměrně dost. Valná většina parametrů jsou upřesňující hodnoty pro multilaterační systém a pouze několik parametrů reprezentují nastavení optimalizačního algoritmu - hodnota křížení, mutace, velikost populace, počet generací, typy selekčního mechanismu a dva typy křížení.



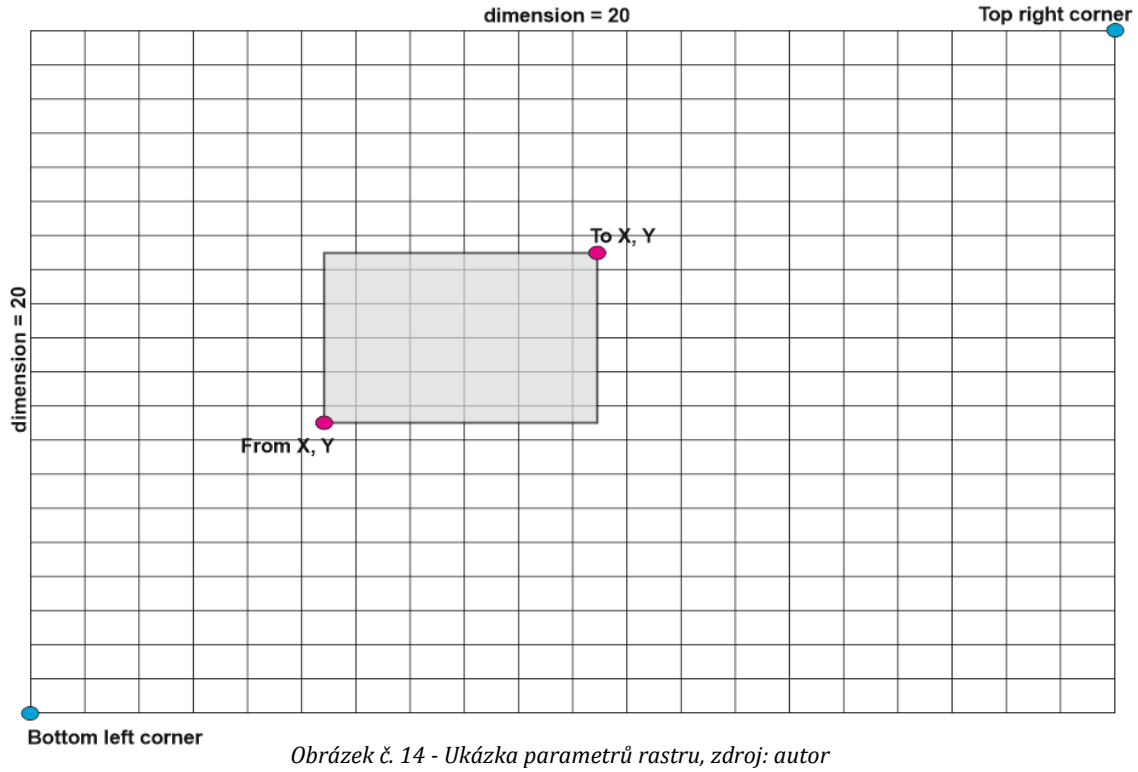
Obrázek č. 13 - Úvodní obrazovka s parametry, zdroj: autor

Parametry

Je nutné vysvětlit jednotlivé parametry pro další postup implementace. Následující popis vychází z okomentovaného ukázkového konfiguračního souboru

poskytnutého firmou ERA a.s. *„Number of receivers“* reprezentuje počet přijímačů vstupujících do výpočtů. Parametr může nabývat pouze celých čísel. Přijímač jako takový má pouze souřadnice v prostoru. Tyto souřadnice jsou ve formátu tří reálných čísel odděleny mezerou. Podobným parametrem je *„Number of Interrogators“* reprezentující počet dotazovačů. Platí pro něj to samé, co platí u přijímačů.

Následující parametry reprezentují mřížku (rastr) daného výpočtu. Souřadnice prostoru pro výpočet přesnosti, souřadnice prostoru pro umístění jednotlivých stanic a také hustotu dané mřížky. Parametry *„Bottom left corner X“* a *„Bottom left corner Y“* značí dolní levý roh a jeho gps souřadnice na mapě. Obdobně parametry *„Top right X“ a „Top right Y“* značí horní pravý roh a jeho gps souřadnice na mapě. Parametry *„From X“* a *„From Y“* reprezentují levý dolní roh prostoru pro rozmístění stanic a parametry *„To X“* a *„To Y“* jsou souřadnice horního pravého rohu. Posledním parametrem mřížky pro výpočet je *„Dimension“.* Parametr reprezentuje hustotu rastru daného výpočtu. Na obrázku č. 14 jsou znázorněné jednotlivé parametry rastru pro lepší představu daného problému.



Obrázek č. 14 - Ukázka parametrů rastru, zdroj: autor

Dále jsou zde čtyři parametry reprezentující genetický algoritmus. První z nich je parametr *„Cross rate“*, který reprezentuje pravděpodobnost křížení jedinců z populace. Obdobným parametrem je *„Mutate rate“* reprezentující pravděpodobnost mutace daného jedince. Velikost jednotlivé populace neboli počet jedinců v každé populaci, je v parametru *„Population size“*. *„Number of generations“* je, jak už název napovídá, počet generací, které proběhnou v genetickém algoritmu. Čím větší jsou hodnoty posledních dvou parametrů, tím větší je pravděpodobnost najít lepší optimum daného rozmístění. Po deseti generacích s počtem jedinců deset v každé populaci vrátí genetický algoritmus velmi pravděpodobně horší výsledek než při větších hodnotách parametrů.

Poslední částí parametrů, které patří do ERA části algoritmu, jsou parametry reprezentující scénář měření. Výpočet probíhá pouze v bodech mřížky dané sekce - viz výše. Prvním je parametr *„Target Height“* reprezentující výšku cíle nad Zemí. Tato výška není klasická kartézská, nýbrž výška elipsoidická. Díky této skutečnosti je mřížka zaoblená. Dále je zde parametr *„Height Deviation“*, který značí přesnost externí informace o výšce. Informace o výšce může nabývat buď slova *none*, reprezentující naprostou přesnost výšky, nicméně také může nabývat čísla (metrech) a následně se tato informace připočítává ke kovarianční matici. Přesnost měření Time of Arrival (TOA) reprezentuje parametr *„ToA Deviation“*. Pracuje se zde s normálním rozdělením. Dalším parametrem je *„Interrogation Deviation“* značící chybu měření dotazu, znovu se uvažuje o normálním rozdělení. Náhodnou prodlevou na odpovídači cíle reprezentuje parametr *„Trans. Random Dev.“* s normálním rozdělením pro jeden cíl. Parametr *„Trans. System Dev.“* reprezentuje systematickou chybu na odpovídače cíle oproti střední hodnotě prodlevy dané normou. Tato norma může při různých aplikacích nabývat různých hodnot. Uvažuje se o normálním rozdělení přes všechny cíle. Poslední parametrem je *„Atmospheric Error“*. Tento parametr reprezentuje chybu atmosféry. S rostoucí vzdálenosti roste vliv atmosférické chyby. Pokud je nastaven na *0.0* ignoruje se chyba, kterou atmosféra vytváří.

Poslední dva resp. tři parametry reprezentují výběr metody selekce a metody křížení. Parametr „*Selection*“ je parametrem určující, jakým způsobem budou jedinci v populaci vybíráni. Parametr v sobě skrývá algoritmus ruletového

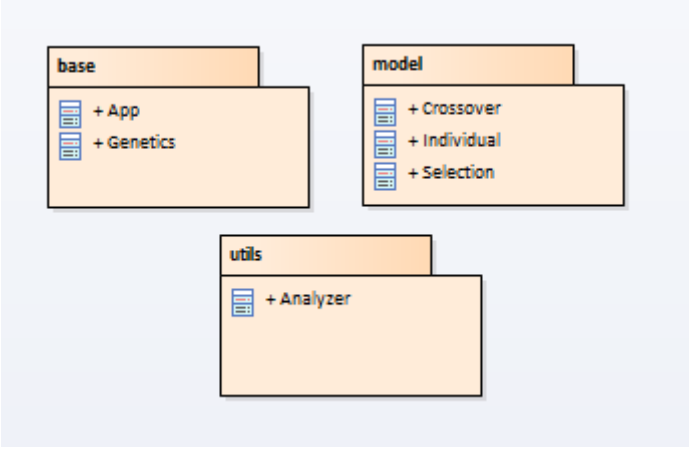
kola s výsečí přímo úměrné hodnoty fitness. Dále turnajovou selekci a selekci pomocí pořadí (rank). Parametr „*Size of Tournament*“ je v defaultním stavu zakázán a je povolen pouze při výběru turnajové selekce v předchozím parametru. Parametr značí velikost turnaje *k* (viz teoretická část). Posledním parametrem je „*Crossover type*“, který, jak název napovídá, reprezentuje typ mechanismu křížení, lze vybrat jednobodové křížení a uniformní.

Výše představené parametry nejsou všechny. Program *Analyzer* od firmy ERA a. s. umí nastavit ještě pár parametrů, které v kontextu diplomové práce nejsou relevantní a převážně reprezentují historický vývoj tohoto programu. Některé nepředstavené parametry zjednodušují a abstrahují velmi důležité vlastnosti, což má za následek méně reálné výsledky.

Předchozí odstavce popisují parametry, které mohou měnit výsledek velmi podstatným způsobem. Jednotlivé parametry by měly být nastavovány s rozvahou, aby se výsledek algoritmu přiblížil, co nejpřesněji, reálnému světu.

3.1.2.2 Implementace genetického algoritmu

Pro optimalizaci rozmístění rádiových stanic v prostoru byl vybrán optimalizační mechanismus evolučních algoritmů. Princip fungování evolučního algoritmu je vysvětlen v teoretické části diplomové práce. Nyní bude představen jednoduchý evoluční algoritmus implementovaný v Javě. Algoritmus bude postupovat klasickým způsobem.

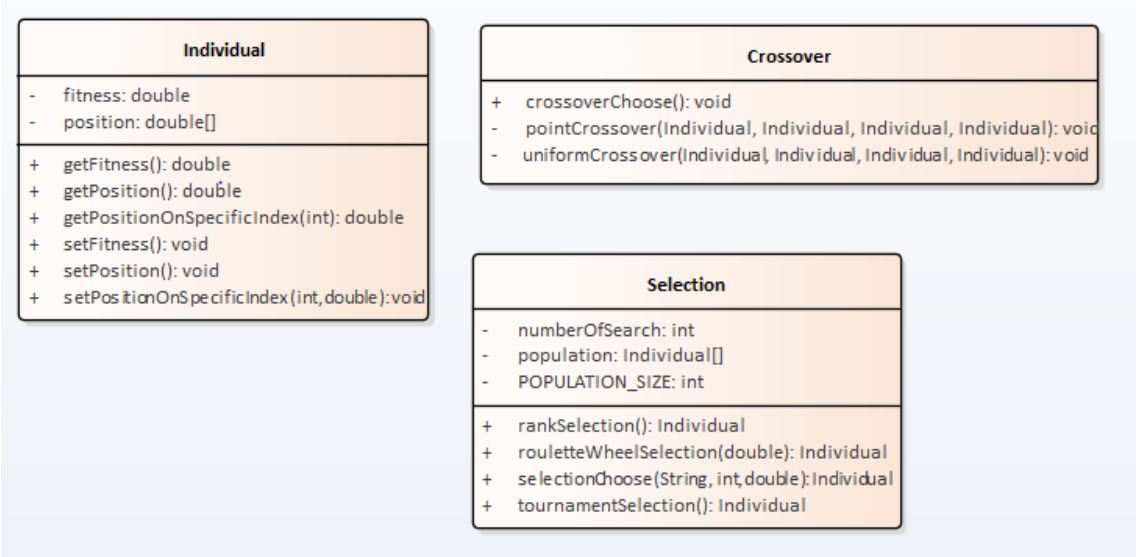


Obrázek č. 15 - Struktura balíčků projektu, zdroj: autor

Velkou výhodou při implementaci algoritmu bylo poskytnutí programu pro výpočet přesností v jednotlivých bodech od firmy ERA a. s. Výpočet fitness hodnot

jedinců zajistil externí program Analyzer.exe. Výpočet hodnot fitness jedinců je jedním ze zásadních kroků pro dobrou konvergenci algoritmu k optimálnímu řešení (viz teoretická část). Díky této skutečnosti bylo snazší navrhnout jednoduchý a efektivní program pro optimalizaci rozmístění rádiových stanic.

3.1.2.3 Balíček model



Obrázek č. 16 - Balíček model, zdroj: autor

Třída Individual

Třída *Individual* je modelová třída pro reprezentaci jedince v populaci. Jedinec v tomto případě má dva atributy. První z nich je double pole *position* reprezentující jednotlivé souřadnice pro přijímače a dotazovače. Double pole bylo vybráno pro svoji rychlost a jednoduchost, protože s hodnotami jednotlivých souřadnic se příliš nepracuje. Je třeba pouze vytahovat hodnoty souřadnic a přidávat hodnoty na přesné místo v poli. Sudé prvky reprezentují souřadnice na ose X a liché prvky pole reprezentují souřadnice na ose Y. Druhým atributem třídy je *fitness*. Tento atribut reprezentuje hodnotu fitness pro daného jedince a byl použit datový typ *double*.

Třída obsahuje tři parametrické konstruktory. První z nich má vstupní parametr *Individual individuum* a nastavuje atributy *position a fitness* ze vstupního parametru do atributů třídy. Do druhého konstruktoru vstupuje parametr

numberOfSearch typu *int*, reprezentující počet hledaných souřadnic, např. při pěti přijímačích a jedním dotazovačem je to parametr dvanáct.

$$numberOfSearch = numberOfRece * 2 + numberOfInter * 2$$

Rovnice č. 10

, kde *numberOfRece* reprezentuje počet přijímačů a *numberIfInter* počet dotazovačů. Obě proměnné se násobí dvěma, protože jsou hledány souřadnice *x* a *y*.

Posledním parametrickým konstruktorem je vstupní parametr *double[] position*, který nastavuje instanci jednotlivých souřadnic.

Individual má několik metod. Jsou to převážně gettery a settery. Metoda *getPosition()* vrací double pole reprezentující pole souřadnic a následně *setPosition(double[] position)* nastavující souřadnice daného jedince. Obdobně metody *getFitness()* a *setFitness(double fitness)* pro zjištění hodnoty fitness a nastavení hodnoty fitness pro jedince. Poslední dvě metody reprezentují souřadnici na specifickém místě v poli. Metoda *setPositionOnSpecificIndex(int index, double position)* nastavující hodnotu souřadnice na daném indexu v poli a *getPositionOnSpecificIndex(int index)* zjištující danou hodnotu souřadnice na určitém indexu v poli.

Třída Selection

Reprezentuje selekční mechanismy genetického algoritmu. Obsahuje tři atributy (*int population_size, Individual[] population, int numberOfSearch*), které se naplňují v parametrickém konstruktoru. Z názvu jednotlivých parametrů je jasná, jejich reprezentace.

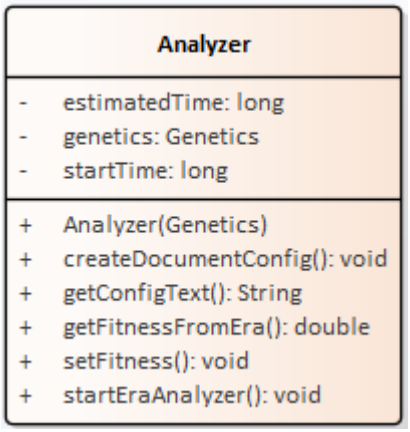
Dále třída obsahuje čtyři metody. První z nich je metoda *selectionChoose (String selection, int sizeOfTournament, double sumFitness)*. Tato metoda volá další metody na základě vstupního parametru *selection*. Jediná metoda v této třídě, která je public, vrací jednoho jedince - návratová hodnota je typu *Individual*. Další tři metody již reprezentují jednotlivé selekce. První metodou je rouletteWheelSelection(double sumFitness). Vstupní parametr reprezentuje upravenou kumulativní fitness všech jedinců v populaci. Díky této hodnotě lze dále určit, jak velkou výseč daný jedinec dostane. Metoda tournamentSelection(int

sizeOfTournament) reprezentuje turnajovou selekci, kde se z populace vyberou jedinci a jejich počet závisí na vstupním parametru. Poslední metodou je rankSelection() reprezentující ruletový mechanismus s pořadím, kde první jedinec má stále stejnou pravděpodobnost výběru.

Třída Crossover

Poslední třída patřící do balíčku model. Reprezentuje výběr a aplikování mechanismu křížení. Neobsahuje žádné vnitřní atributy a žádný parametrický konstruktor. Obsahuje pouze tři metody, kde první z nich je *crossoverChoose(String crossover, Individual origin1, Individual origin2, Individual child1, Individual child2)*, která dále deleguje jednotlivý mechanismy na základě vstupního parametru *crossover*. Parametry reprezentuji dva předky a dva vzniklé potomky. Metoda *pointCrossover(Individual origin1, Individual origin2, Individual child1, Individual child2)* reprezentuje jednobodové křížení, kde pomocí náhodného čísla je určen bod křížení. Poslední metodou je uniformCrossover(Individual origin1, Individual origin2, Individual child1, Individual child2), která náhodně vybírá gen od jednoho z rodičů, který dostane potomek.

3.1.2.4 Balíček utils



Obrázek č. 17 - Balíček utils reprezentuje pouze jedna třída, zdroj: autor

Třída Analyzer

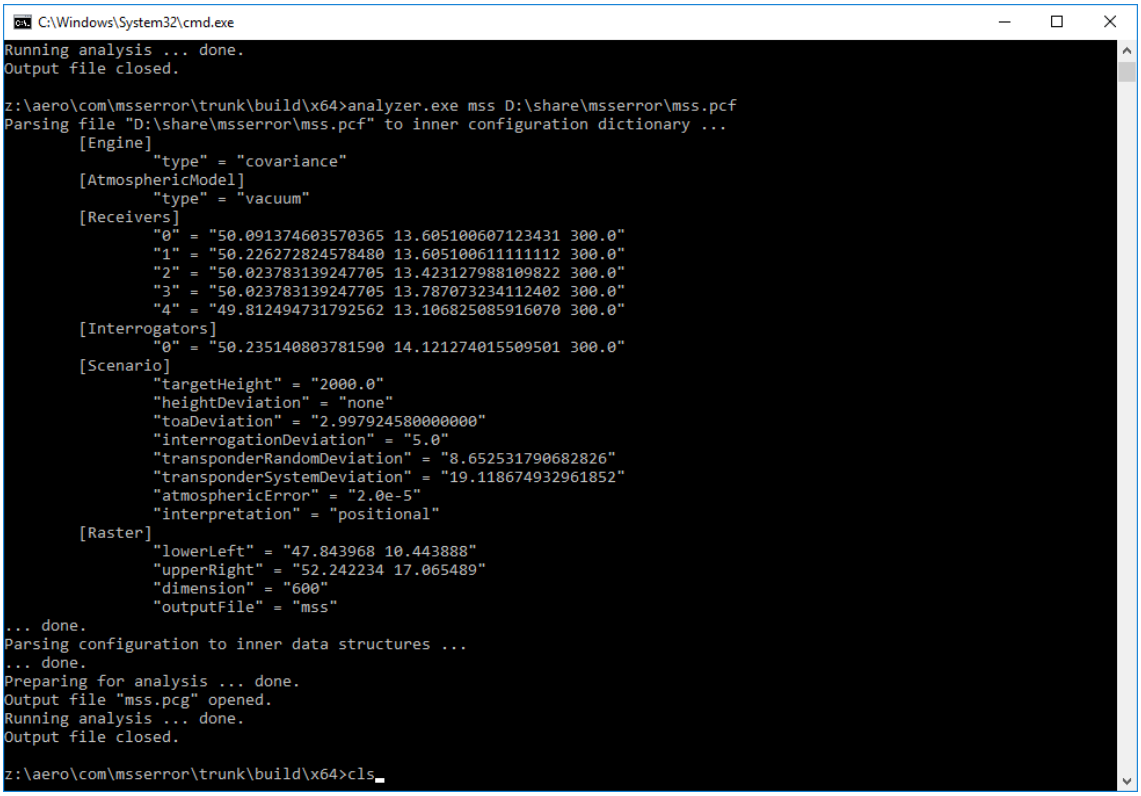
Pomocná třída pro obstarávání všeho ohledně externího programu Analyzer.exe. Třída obsahuje pouze jeden respektive tři atributy. Prvním atributem je *genetics* typu *Genetics* reprezentující instanci třídy *Genetics*. Další dva parametry jsou pouze pomocné - long *startTime* a long *estimatedTime*. Parametry slouží pro účel měření časového úseku jedné generace. Atributy v tomto případě nemají gettery a settery.

Třída obsahuje jeden parametrický konstruktor s parametrem *gen* typu *Genetics*, který nastavuje třídě aktuální instanci *Genetics*.

Třída *Analyzer* obsahuje metodu *getConfigText(double [] position)*, která slouží k vytváření dlouhého textového řetězce reprezentující text konfiguračního souboru. Metoda pracuje s polem souřadnic jednotlivých jedinců populace a následně vrací tento text dále do dalšího postupu.

Předchozí výsledek metody vstupuje jako parametr do další metody *createDocumentConfig(String configText)*. Metoda pouze obstarává vytvoření konfiguračního souboru s koncovkou pcf na disk.

Důležitá metoda je *startEraAnalyzer()*, která reprezentuje zapnutí, setrvání a vrácení výsledku z externího programu Analyzer.exe. Prvním krokem, kterým metoda začíná, je vytvoření *ProccesBuilderu* [37], který obstarává start příkazového řádku se všemi náležitostmi. V analyzer.exe probíhá výpočet přesnosti pro dané rozmístění stanic v prostoru. Tato část programu je nejdéle trvající a je třeba počkat na doběhnutí programu. Pro představu délka tohoto kroku je několikanásobně delší než běh celého zbytku programu. Výsledkem této metody je vygenerování souboru ve formátu pcg. Tento formát lze brát v úvahu jako klasický csv soubor. Jednotlivé hodnoty jsou odděleny středníkem.

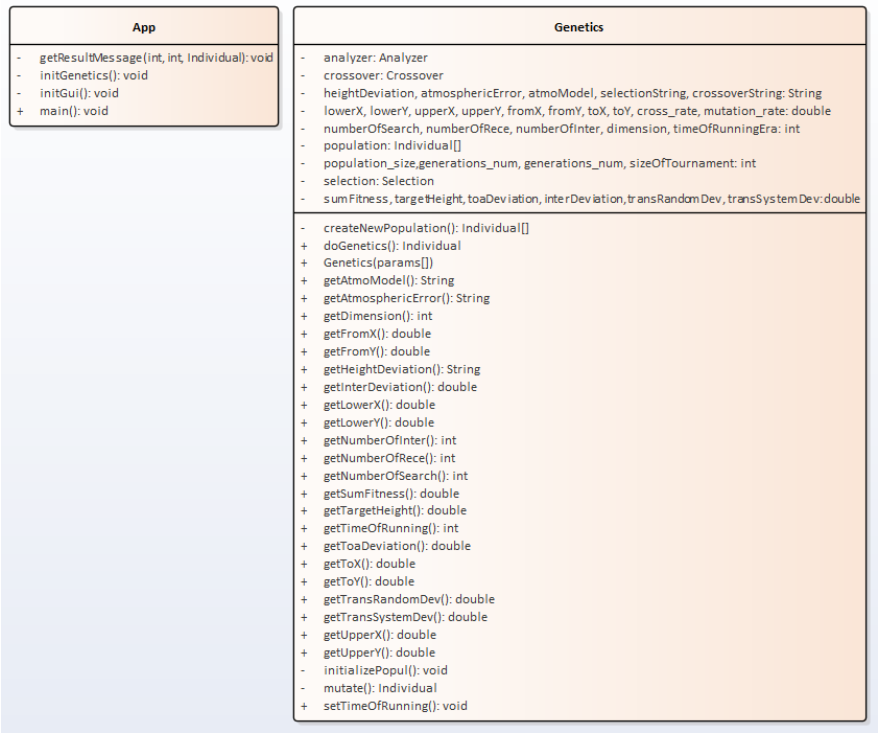


Obrázek č. 18 - Zavolání programu Analyzer v příkazové řádce, zdroj: [34]

Funkcionalitu vytažení výsledné přesnosti daného jedince ze souboru je obstaráno metodou *getFitnessFromEra()* s návratovou hodnotou *double* reprezentující průměrnou hodnotu přesnosti. Metoda využívá *BufferedReader* [37] pro rychlejší čtení výsledného souboru. Tento objekt čte jednotlivé prvky z výsledného souboru a sčítá je. Po přečtení celého souboru se objekt zavře a vrátí se zprůměrovaná hodnota přesnosti v každém bodě.

Všechny předešlé metody jsou součástí metody *setFitness(Individual individuum).* Metoda nastavuje jedinci z populace pomocí předešlých metod hodnotu fitness.

3.1.2.5 Balíček base



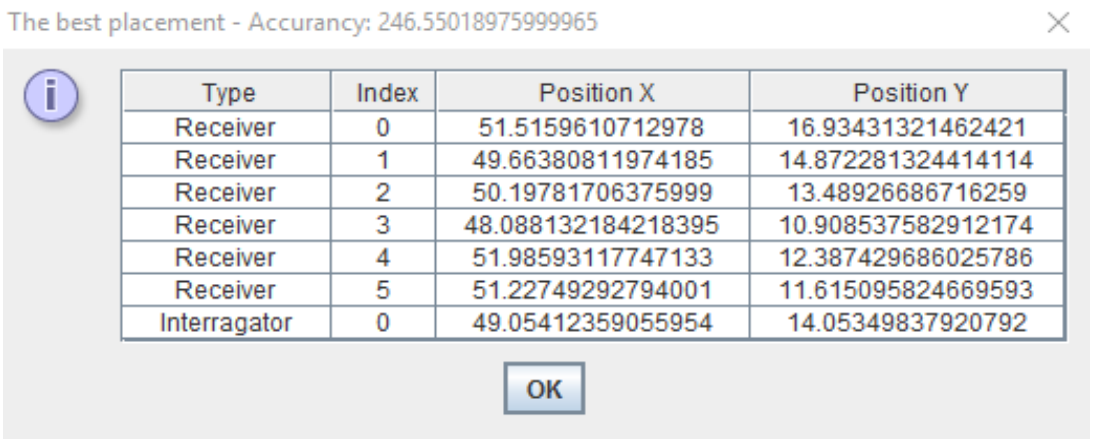
Obrázek č. 19 - Balíček base, zdroj: autor

Třída App

Třída *App* reprezentuje spouštěcí třídu celého programu. Neobsahuje žádné vnitřní atributy, ale má tři vnitřní metody, pokud se nepočítá spouštěcí *main*. První představenou metodou je *initGui()*, reprezentuje inicializaci a formátovaní grafického uživatelského rozhraní (obr. č. 13). Jak lze vidět na obrázku, jsou zde implementovány *JLabel* (text), *JTextField* (textová pole), *JSpinner* (nastavení čísel z intervalu) a *JComboBox* (rolovací nabídka) [37].

Další metodou je *initGenetics(double crossrate, ...)*, vstupní parametry této metody jsou všechny, které lze vidět na obrázku č. 13, a jsou popsány v předchozí kapitole. Metoda zapouzdřující inicializaci instance *Genetics*, která zapíná optimalizační algoritmus a jeho výpočet. Posledním metodou je zde *getResultMessage(int numberOfRece, int numberOfInter, Individual result)*

vytvářející výsledný report algoritmu. Na obrázku č. 20 je výsledek této metody, tvořící jednoduchou naformátovanou tabulku *JTable* s daty o optimálním rozmístění jednotlivých stanic.



Obrázek č. 20 - Okno s výsledným rozmístěním stanic, zdroj: autor

Třída Genetics

Pro účel diplomové práce je tato třída nejdůležitější třídou výsledného programu. Třída zapouzdřuje téměř každou část optimalizačního algoritmu. Obsahuje spoustu atributů, které reprezentují parametry (viz kapitola 3.1.2.1) pro výpočet. Nicméně třída dále obsahuje i atribut *population* typu *Individual[]*, reprezentující populaci jedinců a atribut reprezentující instanci na třídu *Analyzer*. Parametrický konstruktor obsahuje vstupní proměnné reprezentující parametry vyhledávání.

Postup popisování metod bude totožný s obecným fungováním evolučního algoritmu. Prvním krokem algoritmu je vytvoření počáteční populace - *initializePopul()*. Metoda vytváří náhodně jedince se souřadnicemi, které jsou v oblasti nastavené pomocí parametru („*From X*“, „*From Y*“, „*To X*“, „*To Y*“). Dále se přiřazuje hodnota fitness funkce pro daného jedince. Tento krok je řešen ve třídě *Analyzer*.

Pokud již mají jedinci přiřazenou hodnotu fitness, nastává proces selekce. Pro tento účel je zřejmé, že se zavolá konstruktor třídy *Selection*, který si se selekcí poradí.

Po výběru dvou jedinců přichází na řadu reprodukce neboli křížení a mutace jedinců. Tuto funkcionalitu zastřešuje metoda *createNewPopulation()* s návratovou hodnotou pole jedinců. Při křížení se volá třída *Crossover*, která podle vstupního parametru vybere, které křížení bude aplikováno.

Po křížení jedinců může nastat i mutace těchto nově vytvořených jedinců. Metoda *mutate(Individual individuum)* tuto funkcionalitu zapouzdřuje. Pokud nastane mutace, tak se na jednom prvku souřadnice přičte nebo odečte malá náhodná hodnota pro lepší rozmanitost populace. Pro lepší konvergenci algoritmus je na konci vytváření populace přidán předchozí nejlepší jedinec do nové populace pro zachování tohoto jedince. Tento proces je označován jako elitismus. Při jeho použití jsou nejlepší jedinci přeneseny přímo do nově vytvořené populace. Kvalita nejlepšího jedince se díky elitismu bude v průběhu algoritmu pouze zvyšovat. Na konci výpočtu je velmi pravděpodobné, že se už nenachází lepší jedinec. Díky této skutečnosti by bylo nevhodné ztratit nejlepšího jedince z předchozí generace.

Metoda zapouzdřující všechny předchozí metody je *doGenetics()*, která vrací optimální výsledek optimalizačního algoritmu (*Individual*). Metoda průběžně vypisuje do konzole dosavadní postup a ukládá nejlepšího jedince. Je to jediná metoda v této třídě, která je public, pokud se nebudou brát v potaz gettery. Třída neobsahuje settery pro jednotlivé atributy. Všechny atributy jsou nastavovány pouze v parametrickém konstruktoru, protože atributy této třídy jsou při běhu programu neměnné a nebylo třeba generování setterů.

3.2 Testování optimalizačního algoritmu

Optimalizační algoritmus je implementován, funguje a je potřeba výsledný algoritmus vyzkoušet a otestovat. Následující část diplomové práce lze rozdělit do dvou testovacích scénářů. První z nich je testování algoritmu jako takového. Budou otestovány jednotlivé parametry ovlivňující přímo běh algoritmu např. testování různých kombinací pravděpodobností křížení a mutace či typy selekcí a křížení. Evoluční algoritmy patří do stochastických algoritmů (viz teoretická část), a proto bylo pro každou kombinaci provedeno šest měření. Průběhy kombinací jsou v příloze č. 3 Grafy průběhů jednotlivých kombinací. V druhé půlce kapitoly bude algoritmus otestován na reálném problému.

3.2.1 Testování vlastností algoritmu - kombinace selekcí a křížení

První testování, které bylo provedeno, je testování kombinací selekčních mechanismů s různými mechanismy křížení. Byly otestovány všechny kombinace se všemi danými s následným grafickým výstupem v podobě spojnicového grafu. Ze šesti měření bylo vybráno vždy jedno, které mělo nejlepší výsledek. Všechny tyto kombinace byly testovány se stejnými parametry celého procesu jen s jinými selekcemi a kříženími.

Testovací scénář v tomto případě byl takový, že je potřeba rozmístit šest přijímačů a jeden dotazovač. Cílem je, co nejlépe určit toto rozmístění pro oblast zhruba od Mnichova do Vratislavi (Polsko). Jednotlivé stanice lze rozmístit bez restrikcí kamkoliv. Optimalizační algoritmus má v tomto případě hodnotu křížení 1, což znamená, že jedinci se budou křížit pokaždé. Hodnota pravděpodobnosti mutace je 0,05 reprezentující 5% šanci na mutaci nového jedince. Velikost populace je 200 a algoritmus bude běžet 50 generací. Všechny parametry jsou přiloženy v příloze 1 - Tabulka nastavení parametrů - testovací scénář.

Výsledné hodnoty RMS vycházejí metrech z podstaty trojrozměrného prostoru. Nicméně při existenci externí informace o výšce (*Target height*) a s nenastavenou odchylkou výšky (Height *deviation = none*) je upuštěno od jednoho rozměru. Výsledky algoritmu jsou v čtverečních metrech.

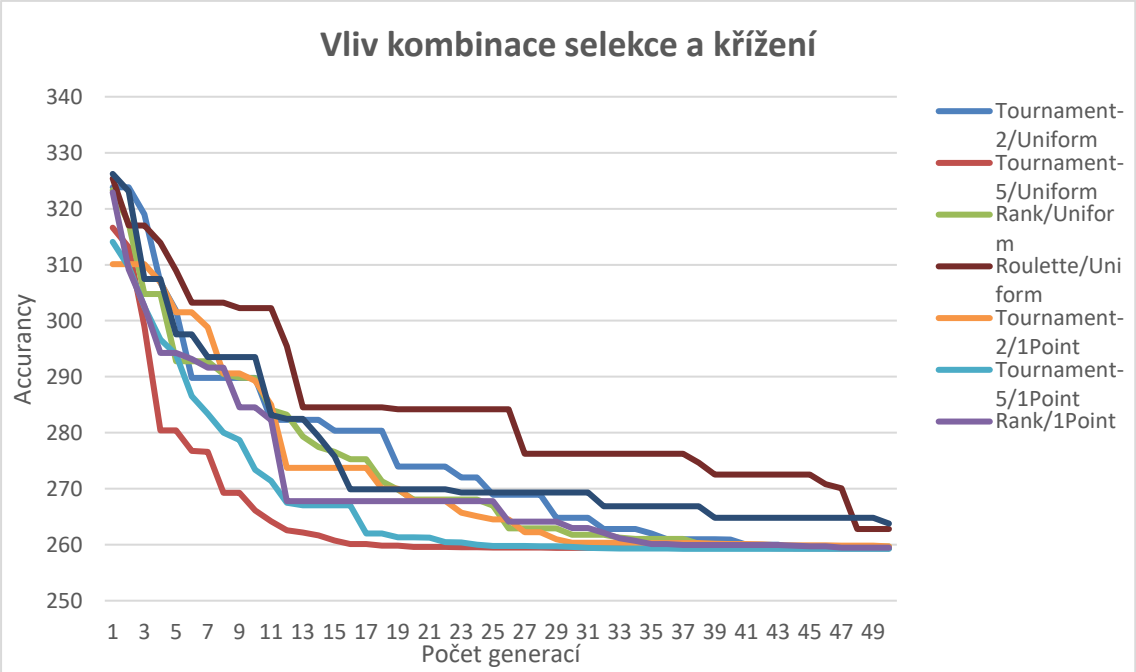
Výsledek jednotlivých kombinací lze vidět na obrázku č. 21. Lze si povšimnout, že průběh jednotlivých kombinací je rozdílný a některé mechanismy,

konvergují rychleji než jiné. Z měření vyplývá, že nejrychleji konvergují ty algoritmy, které mají vysoký výběrový tlak, což v tomto případě mají turnajové selekce o velikosti turnaje pět. Je jedno, jaký typ křížení v tomto případě je, obě křivky jsou si dosti podobné. V tomto případě jedinec, který se dostane do reprodukce, musí porazit další čtyři jedince. Nevýhodou je, že klesá rychleji rozmanitost celé populace oproti ostatním metodám.

Další pomyslnou skupinou lze označit čtyři kombinace, které mají také podobný průběh. Nejhorší kombinaci v této části je Rank/1Point, tedy ruletové kolo podle pořadí v populaci s jednobodovým křížením. V tomto případě algoritmus má velmi skokový průběh, kde se přesnost razantně změní a nejcitelnější změna je z 11 na 12 generaci, kde je skok o více než 16 m².

Poslední skupinou jsou dvě kombinace, které konvergují nejpomaleji. Obě tyto kombinace mají v selekční mechanismus ruletového kola na základe fitness hodnot jedinců. Nicméně obě kombinace skončili na podobných hodnotách v 50. generaci (262.80 m² a 263.78 m²).

Všechny kombinace konvergují v rámci 50 generací velmi dobře a v 50. generaci jsou ve výsledné přesnosti malé rozdíly. Nejhorší hodnotou je 263.78 m² u Rank/1Point a nejlepší hodnotou je 259.26 m² u turnajové selekce velikosti 5 s jednobodovým křížením. Turnajové selekce o velikosti 5 od 20. generace zlepšily svůj výsledek jen minimálně a dalo by se říci, že tyto kombinace konvergovaly k optimálnímu řešení nejrychleji.

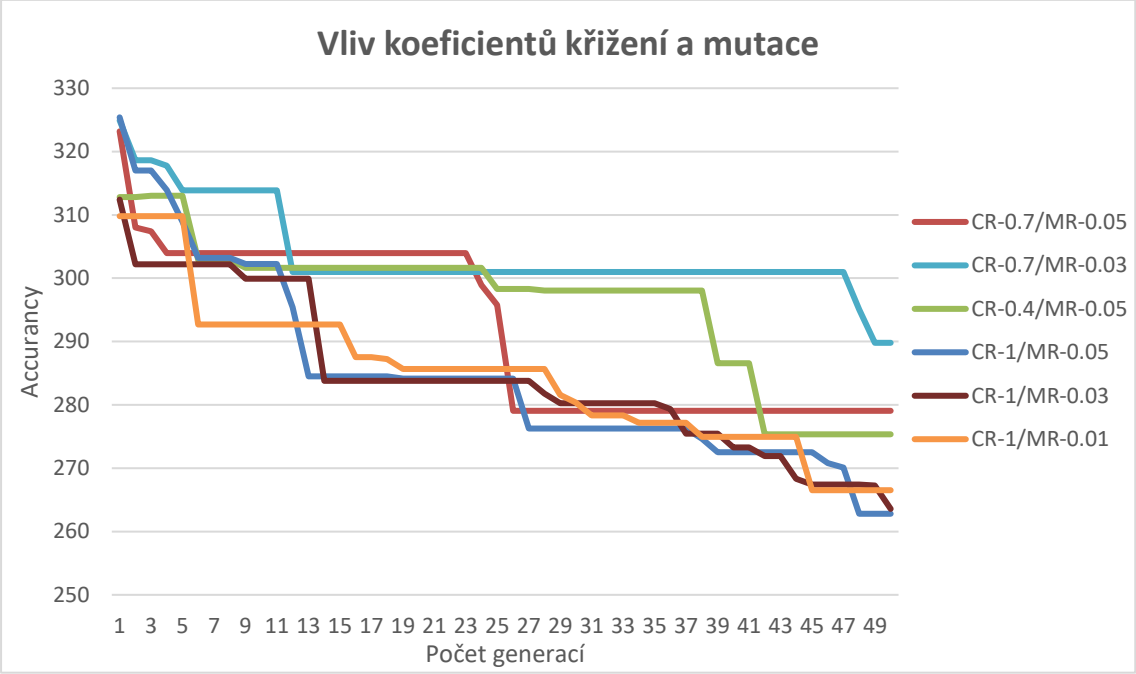


Obrázek č. 21 - Graf vlivu kombinací selekce a křížení na konvergenci algoritmu, zdroj: autor

3.2.2 Testování vlastností algoritmu - pravděpodobnost křížení a mutací

V této kapitole bude nahlédnuto na testování optimalizačního algoritmu, vliv pravděpodobnosti křížení a mutací na výsledek algoritmu. Bude zde stejný testovací scénář jako v předešlém případě, jen s malou odchylkou. Budou se měnit jednotlivé pravděpodobnosti křížení a mutací jedinců. Kombinace selekčních mechanismů a typů křížení zde bude neměnná. Byla vybrána kombinace s menším výběrovým tlakem - ruletové kolo podle hodnot fitness s uniformním křížením. Bylo zde také pro každou kombinaci provedeno šest měření a vždy byl vybrán nejlepší výsledek.

Z obrázku č. 22, je patrné rozdělení do hypotetických dvou skupin. První a zároveň lepší skupinou je skupina s pravděpodobností křížení 1, tudíž každý jedinec se kříží. Kombinace pravděpodobností CR-1 (Crossover rate = 1) s MR-0,05 (Mutation rate = 0,05) a CR-1/MR-0,3 mají podobný průběh a končí víceméně se stejnou hodnotou přesnosti. Kombinace s CR-1/MR-0,01 již má na konci běhu o něco horší výsledek a konverguje nejpomaleji z této skupiny.



Obrázek č. 22 - Graf vlivu pravděpodobnosti křížení a mutace, zdroj: autor

Pokud se změní hodnota pravděpodobnosti křížení, drasticky klesá optimalizační výkon algoritmu. Nejlepším kombinaci z této skupiny je CR-0,4/MR-0,05, která končí v 50. generaci o něco lépe než ostatní dvě. Nicméně tato kombinace má ve většině případě horší průběh, než ostatní dvě. Lze si toho povšimnout v příloze č. 3 Grafy průběhů jednotlivých kombinací.

Celá tato skupina konverguje velmi pomalu. Nejlepší jedinec se v těchto skupinách drží velmi dlouho prvenství např. v kombinaci CR-0,07/MR-0,03 je jeden jedinec objeven ve 12. generaci a do 47. generace byl stále nejlepším.

Z předchozích odstavců je zřejmé, že nastavení jednotlivých parametrů optimalizačního algoritmu mají velmi důležitou hodnotu a měly by být zvoleny rozumně. Konvergenci nejvíce ovlivňují pravděpodobnost křížení a metoda výběru jedince pro další reprodukci. Pravděpodobnost mutace má oproti křížení velmi minoritní dopad na celkovou rychlost algoritmu. Způsob křížení má oproti selekci také malý dopad, nicméně uniformní křížení pomáhalo lepší konvergenci oproti jednobodovému křížení.

3.2.3 Testování algoritmu - reálný případ

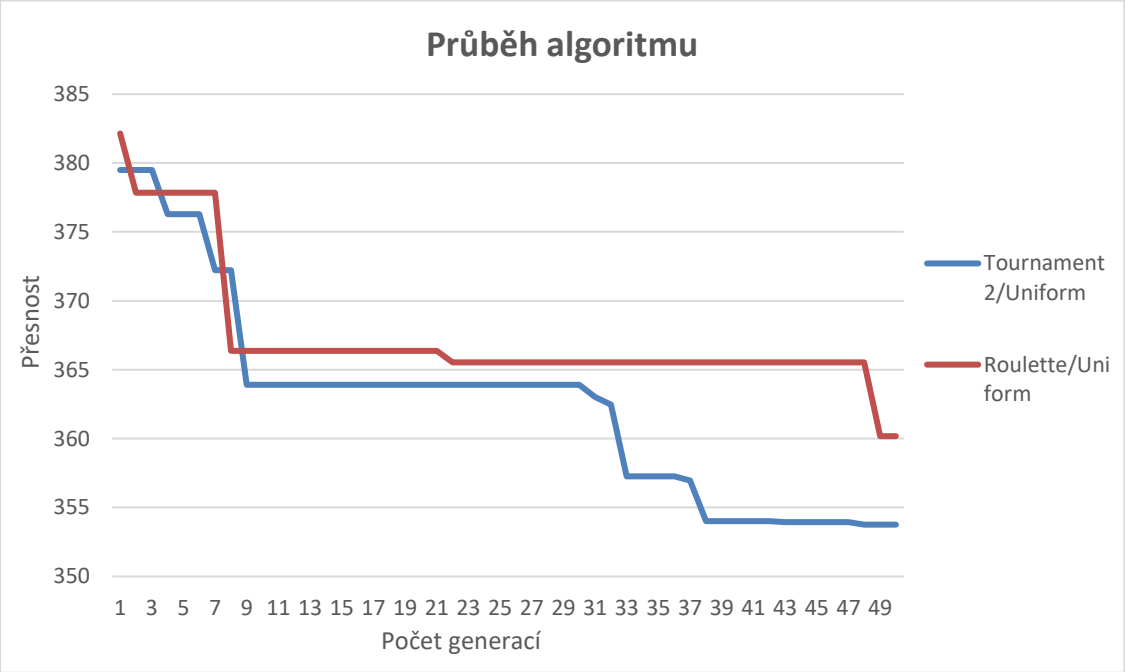
V této kapitole bude otestován optimalizační algoritmus z pohledu reálného využití. V následujících odstavcích bude představen testovací scénář, nastavení jednotlivých parametrů, porovnání výsledků od firmy ERA a.s. a optimalizačního algoritmu. Znovu je podotýkáno, že výsledek algoritmu nemusí být správný kvůli absenci informací o reliéfu - stanice jsou rozmísťovány v konstantní nadmořské výšce 300 m n. m.

Nastavení všech jednotlivých parametrů je přehledně zobrazeno v příloze č. 2 - Tabulka nastaveni parametrů - testovací scénář - reálný příklad. Bude rozmístěno 5 přijímacích stanic a jedna dotazovací stanice. Výška cíle je známa a její hodnota je 600 m. Zkoumaná výsledná oblast je od Zruče nad Sázavou (souřadnice: 49°42'18.257"N 15°9'35.022"E) u vodní nádrže Švihov až po Szczytna (50°23'57.81"N 16°25'43.243"E) kousek za hranicemi u Náchoda. Rozmístění rádiových stanic bude v menším úseku a pouze v České republice. Oblast rozmístění začíná v Červených Janovicích (49°50'06.0"N 15°15'17.1"E) a končí v Bohdašíně (50°20'16.1"N 16°12'29.0"E). Zkoumaná oblast je vyobrazena na obrázku č. 24. Poslední za zmínku stojí hustota mřížky, která je 600, což zajišťuje přesnější výsledky na úkor rychlosti algoritmu (viz obrázek č. 13). Musí se přepočítávat pro každého jedince 360 000 hodnot. Algoritmus bude probíhat 50 generací s počtem jedinců v každé generaci 200.

Z předchozích pokusů ohledně nastavení jednotlivých kombinací je zřejmé, že každým průběhem můžeme dostat o trochu jiné výsledky. Předchozí kombinace měly vždy šest měření. Pro časovou náročnost reálného příkladu bude provedeno pouze jedno měření.

Pravděpodobnost křížení jedinců je zde nastaveno na 100%, neboť z předchozích kapitol dosahovala nejlepších výsledků. Typem křížení je uniformní křížení. Pravděpodobnost mutace byla ponechána na 5% hladině a typ selekce je turnajová s velikostí turnaje dva.

I přes snahu optimalizovat rychlosti algoritmu, jedna generace běžela několik minut (10 minut) s tím, že celý optimalizační proces probíhal celou noc. Nicméně zrychlit tento algoritmus není možné z pohledu diplomové práce. Nejdelší čas trval pro výpočet přesnosti (9 min) a byl zajišťován pomocí externího programu Analyzer.exe. Druhým bodem, proč algoritmus probíhal takto dlouho, je existence velmi husté mřížky (600). Při existenci poloviční mřížky by jedna generace algoritmu probíhala kolem 3 minut s tím, že 2,5 minuty by bylo potřeba na externí výpočet. U mřížky velikosti 100 jedna generace by algoritmus probíhal pouze 30 vteřin, pro externí výpočet je třeba 22 vteřiny.



Obrázek č. 23 - Graf průběhu algoritmu na reálném příkladu, zdroj: autor

Na testovacím příkladu byla použita dvě nastavení optimalizačního algoritmu. Měnil se pouze typ selekčního mechanismu. V prvním případě to byla již zmiňovaná turnajová selekce s velikostí turnaje dva. V druhém případě byla použita ruletová selekce s velikostí výseče podle hodnot fitness funkce. Průběh algoritmu lze vidět na obrázku č. 23, kde je patrné, že turnajová selekce má lepší průběh a relativně skončila o dost lépe.

Výsledek algoritmu s nastavenou ruletovou selekcí je přibližně 360,1782 m². Algoritmus tento výsledek našel na konci svého průběhu v 48. generaci. Lepší z dvojice měl výsledek přibližně 353,7599 m², kde tuto hodnotu našel v 42. generaci. Optimalizační algoritmus s nastavenou turnajovou selekcí vylepšil svůj

výsledek od první do poslední generace o více než 25 m² a průběh s nastaveným ruletovým kolem zlepšit téměř o 22 m².

3.2.3.1 Porovnání výsledků

Po zjištění a popsání průběhu optimalizačního algoritmu a jeho výsledku přichází na řadu jedna ze stěžejních částí diplomové práce - porovnání s výsledkem od firmy ERA a.s. Porovnání je předvedeno v následující tabulce č. 1.

Rozmístění stanic poskytované společností ERA a.s. dosáhlo výsledku v představené metodice přibližně 453,5521 m², což na oblast pokrývající téměř celý Pardubický a Královehradecký kraj je dobrý výsledek. Oproti výsledkům optimalizačního algoritmu je výsledek o poznání horší. Nicméně kdyby nyní bylo provedeno celkové shrnutí, že algoritmus je jednoduše lepší, dopustili bychom se nesmírné chyby. Oproti výsledku teoretického algoritmu je toto rozmístění již dané a využívá prověřené oblasti rozmístění. Například přijímač jedna je umístěn v areálu ERA a.s. v Pardubicích a přijímač pět je umístěn na území letiště Skuteč.

Finální rozmístění stanic v algoritmu je v mnoha případech umístěn v poli, bez okolního kontaktu s civilizací, ale i v hustě obydlené oblasti (přijímač dva a pět v algoritmu s ruletovou selekcí). Zároveň nesmí být zapomínáno na třetí rozměr (nadmořská výška) rozmístění, se kterým rozmístění ERA počítá a testovaný algoritmus ne. Pro lepší představu byly všechny stanice přidány do mapy, viz obrázek č. 24, kde si lze povšimnou jednotlivého rozdělení.

Rozmístění jednotlivých výsledků je od sebe rozděleno barvou a tmavším odstínem dané barvy je určen dotazovač. Modrá barva reprezentuje nejlepší výsledek a to výsledek algoritmu s turnajovou selekcí. Zelená barva reprezentuje výsledek algoritmu s ruletovou selekcí a barva oranžová rozmístění ERA a.s.

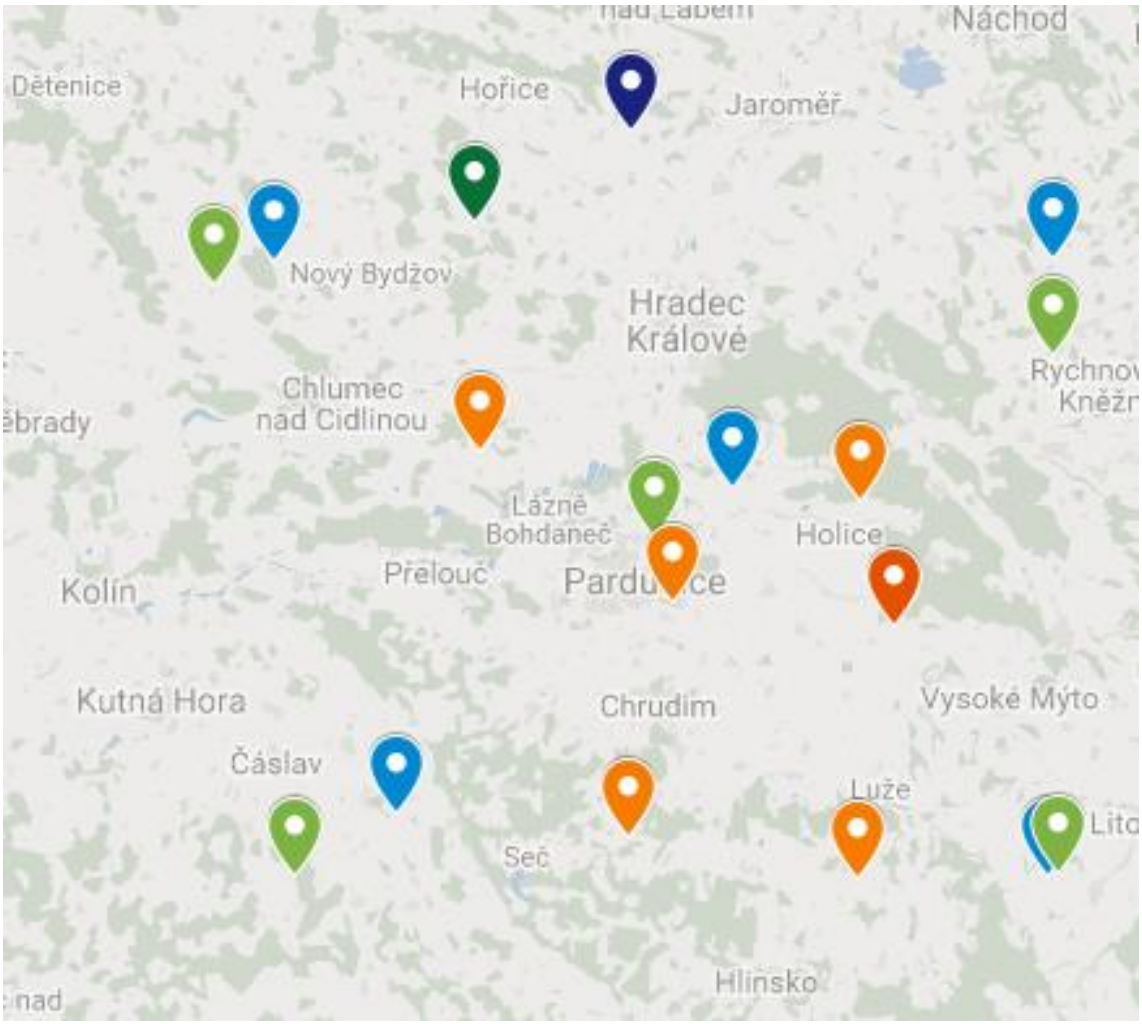
Je zajímavé si všimnout, že představený algoritmus z pěti stanic vytváří útvar podobný *X,* kde čtyři stanice jsou v krajích hledané oblasti a jedna stanice v jejím středu. Toto rozmístění je nejefektivnější pro pět stanic. Dotazovač je vždy přibližně na severní straně mezi dvěma horními stanicemi. Dokonce stanice na jihovýchodě mají pro oba algoritmy totožnou polohu (pro lepšího zobrazení na mapě posunuto). Tento útvar se snažila vytvořit i firma ERA a.s., nicméně je její *X* trochu menší, méně symetrické a otočené o 90 stupňů doprava.

Je jasné, jak postupuje optimalizační algoritmus. Ale bylo by zajímavé, jak postupoval znalostní expert, že došel k podobnému tvaru. Zda má expert základní matematické a geometrické znalosti a ví, které rozmístění při kolika stanicí dělá nejlepší výsledky nebo zda k tomuto rozmístění rádiových stanic došel postupem času.

Tabulka č. 1 - Porovnání výsledných souřadnic a přesností

Typ	Souřadnice - Roulette	Souřadnice - Tournament	Souřadnice - ERA a.s
Přijímač 1	[50.23550529588238, 15.323576240081087]	[50.251719943385666, 16.208062]	[50.02016825111111, 15.807951513611112]
Přijímač 2	[50.186645610241925, 16.208062]	[49.834998, 16.208062]	[50.088056300000005, 16.005454341666667]
Přijímač 3	[49.834998, 16.208062]	[49.87683422178436, 15.516625104223891]	[49.86025952777778, 15.759058555555555]
Přijímač 4	[49.834998, 15.407529585586579]	[50.249927436426674, 15.38654903825054]	[50.12210013333333, 15.603691416666667]
Přijímač 5	[50.06474180348372, 15.787470285763566]	[50.09729579819759, 15.870438188453226]	[49.83272998888889, 16.002081425]
Dotazovač 1	[50.27785509274133, 15.597285014456247]	[50.337797, 15.76246740791887]	[50.00362472222222, 16.039182222222223]
Přesnost	**360.1782275 m²**	**353.7599082 m²**	**453.5521383 m²**

zdroj: autor



Obrázek č. 24 - Znázornění jednotlivých stanic na mapě - výseč mapy = oblast měření - turnajová selekce - modrá, ruletová - zelená, ERA - oranžová (tmavší odstíny = dotazovač), zdroj: autor

4 Shrnutí výsledků

Cílem diplomové práce bylo podat srozumitelné informace ohledně multilateračních systémů. Byl představen princip fungování multilaterace jako stěžejní prvek těchto systémů. Bylo poukázáno na hardwarovou a softwarovou strukturu multilateračních systému a jejich základní použití v oblasti moderní sledování polohy cíle.

V oblasti optimalizačních algoritmů bylo poukázáno na jejich velkou rozmanitost. Blíže byly popsány evoluční algoritmy, které kopírují přírodní jevy z evoluční teorie darwinismu. Detailněji byly popsány jednotlivé principy a kroky algoritmu.

Nejdůležitějším cílem práce bylo navrhnutí a otestování programu pro rozmísťovaní radiových stanic. Testování probíhalo na parametrech optimalizačního algoritmu, kde byla předvedena důležitost typu selekce a pravděpodobnosti křížení na konvergenci algoritmu. Dále byl algoritmus otestován na reálném případu od firmy ERA a.s., kde algoritmus podával lepší výsledky, než poskytnuté rozmístění od ERA a.s. Nicméně tento test nemůže být považován za úspěšný v tom ohledu, že nenahrazuje znalostního experta. Je to zapříčiněno tím, že algoritmus nebere v potaz reliéf terénu. Díky této skutečnosti, výsledky algoritmu nemusí být v reálných podmínkách o tolik lepší a verifikace od firmy ERA a.s. ještě nebyla provedena.

5 Závěry a doporučení

Diplomová práce měla mnoho cílů. Jedním z nich bylo představení multilateračních systému. Bylo předvedeno jeho použití v oblasti určování polohy cíle. Tyto systémy se ve většině případů využívají jako další zdroj určení polohy a verifikují další výsledky. Byla vysvětlena multilaterace jako taková, která se opírá o základní matematické a geometrické znalosti.

Dalším cílem bylo podat srozumitelné informace o optimalizačních algoritmech a vybrat vhodný algoritmus pro optimalizaci rozmístění rádiových stanic. Na konec byly vybrány evoluční algoritmy, kde byla vysvětlena jejich funkcionalita. Jedna z nejdůležitějších funkcionalit je vypočítání fitness hodnot jedinců, která byla vypočítávána pomocí externího programu od firmy ERA a.s.

Hlavním cílem bylo navrhnutí a implementovaní optimalizačního algoritmu na základě informací představených v diplomové práci. Optimalizační algoritmus poskytoval lepší výsledky než empiricky zadané rozmístění od znalostního experta.

Optimalizační program na jednu stranu splnil zadání tím, že poskytoval lepší výsledky. Na druhou stranu, tento program může být stále také optimalizován a mít více možností pro ještě jednodušší a kvalitnější přístup k problému. Program je stále pouze semi-autonomním a je potřeba lidské verifikace.

Pro budoucí vývoj je mnoho dalších dílčích vylepšení. Bylo by vhodné určovat více oblastí pro rozmístění stanic a také určit oblasti, do kterých nelze postavit stanice. Tato funkcionalita by usnadnila hledání optimálního řešení z důvodu menšího prostoru hledaných výsledků. Také představený program má velký nedostatek v podobě neimplementování reliéfu terénu, tedy třetího rozměru rozmísťování.

Také by bylo zajímavé optimalizovat program od firmy ERA a.s. Analyzer, který průběh optimalizačního algoritmu značně zpomaloval. Algoritmus by mohl zatím být aplikován jako podpůrný mechanismus pro práci experta. Program jako takový má určitě své místo v blízké budoucnosti i přes svoji nedokonalost, protože člověk, který určuje rozmístění empiricky, nese velkou zodpovědnost.

6 Seznam použité literatury

[1] HOŘČIČKA, Pavel. ERA A.S. *Rádiové systémy určování polohy - multilaterace*. Pardubice, 2016.

[2] GAVIRIA, Ivan A. Mantilla. *New Strategies to Imporve Multilateration Systems in the Air Traffic Control*. Valencia, Spain, 2013. Doctoral Thesis. Universitat Politecnica de Valencia.

[3] MANTILLA-GAVIRIA, Ivan A., Mauro LEONARDI, Gaspare GALATI a Juan V. BALBASTRE-TEJEDOR. Localization algorithms for multilateration (MLAT) systems in airport surface surveillance. *Signal, Image and Video Processing* [online]. 2015, 9(7), 1549-1558 [cit. 2017-03-18]. DOI: 10.1007/s11760-013-0608-1. ISSN 1863-1703. Dostupné z: http://link.springer.com/10.1007/s11760-013-0608-1

[4] PÉK, Vojtěch. *Využití informace z ADS-B v pasivních multilateračních systémech*. Plzeň, 2008. Diplomová práce. Západočeská univerzita v Plzni.

[5] BURIÁN, Petr. *Identifikace pohybů na letištní ploše*. Brno, 2008. Diplomová práce. Vysoké učení technické v Brně.

[6] UMLAUF, Lukáš. *Aplikace MLAT metody nad sítí low-cost ADS-B přijímačů*. Praha, 2015. Diplomová práce. České vysoké učení technické v Praze.

[7] *Technical Description - Text - MSS-A SOF*. EGO110A00357. Pardubice: ERA a.s., 2014.

[8] DRÁPAL, Stanislav. *Využití SSR módu S pro řízení pohybů letadel a vozidel po ploše letiště* [online]. Vysoké učení technické v Brně. Fakulta strojního inženýrství, 2015 [cit. 2017-03-06]. Dostupné z: http://hdl.handle.net/11012/40108. Diplomová práce. Vysoké učení technické v Brně. Fakulta strojního inženýrství. Letecký ústav. Vedoucí práce Slavomír Vosecký.

[9] *LETECKÝ PŘEDPIS - POSTUPY PRO LETOVÉ NAVIGAČNÍ SLUŽBY*. 439/2011 -220 -SP/1. MINISTERSTVO DOPRAVY ČESKÉ REPUBLIKY, 2011. Dostupné také z: https://lis.rlp.cz/predpisy/predpisy/dokumenty/L/L-4444/data/print/L-4444_cely.pdf

[10] WEISE, Thomas. *Global Optimization Algorithms - Theory and Application*. 2. University of Kassel, Distributed System Group, 2009. Dostupné také z: http://www.it-weise.de/projects/book.pdf

[11] CAMPBELL, Joseph Keim, Michael O'ROURKE a David SHIER. *Freedom and determinism*. Cambridge, Mass.: MIT Press, c2004. ISBN 02-625-3257-3.

[12] ROBBINS, Herbert a Sutton MONRO. A Stochastic Approximation Method. *The Annals of Mathematical Statistics* [online]. 1951, 22(3), 400-407 [cit. 2017-03-18]. DOI: 10.1214/aoms/1177729586. ISSN 0003-4851. Dostupné z: http://projecteuclid.org/euclid.aoms/1177729586

[13] GALBRAITH, Steven D. *Mathematics of public key cryptography*. New York: Cambridge University Press, 2012. ISBN 978-1-107-01392-6.

[14] HUBÁČEK, Petr. *Optimalizace topologie TDOA systému z hlediska přesnosti určení polohy cíle*. Brno, 2010. Disertační práce. Univerzita obrany, Fakulta vojenských technologií.

[15] ČEPIN, Marko. *Assessment of power system reliability: methods and applications*. New York: Springer, c2011. ISBN 978-0-85729-688-7.
[16] NERI, Ferrante, Carlos COTTA a PABLO MOSCATO (EDS.). *Handbook of memetic algorithms*. Berlin: Springer, 2012. ISBN 9783642232473.
[17] DEB, Kalyanmoy, Karthik SINDHYA a Jussi HAKANEN. Multi-Objective Optimization. *Decision Sciences* [online]. Taylor & Francis Group, 6000 Broken Sound Parkway NW, Suite 300, Boca Raton, FL 33487-2742: CRC Press, 2016, s. 145 [cit. 2017-03-19]. DOI: 10.1201/9781315183176-4. ISBN 978-1-4665-6430-5. Dostupné z: http://www.crcnetbase.com/doi/10.1201/9781315183176-4
[18] DEB, Kalyanmoy. *Multi-objective optimization using evolutionary algorithms*. New York: John Wiley, c2001. ISBN 9780471873396.
[19] AVIGAD, Jeremy a Kevin DONNELLY. *Formalizing O Notation in Isabelle/HOL* [online]. s. 357 [cit. 2017-03-19]. DOI: 10.1007/978-3-540-25984-8_27. Dostupné z: http://link.springer.com/10.1007/978-3-540-25984-8_27
[20] PIETRZAK, J. A systematic search for Pareto optimum solutions. *Structural Optimization* [online]. 1999, 17(1), 79-81 [cit. 2017-03-19]. DOI: 10.1007/BF01197716. ISSN 0934-4373. Dostupné z: http://link.springer.com/10.1007/BF01197716
[21] MOORE, H.L. Cours d'Economie Politique. By VILFREDO PARETO, Professeur a l'Universite de Lausanne. Vol. I. Pp. 430. I896. Vol. II. Pp. 426. I897. Lausanne: F. Rouge. *The ANNALS of the American Academy of Political and Social Science* [online]. 1897, 9(3), 128-131 [cit. 2017-03-19]. DOI: 10.1177/000271629700900314. ISSN 0002-7162. Dostupné z: http://ann.sagepub.com/cgi/doi/10.1177/000271629700900314
[22] Pareto efficiency. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-19]. Dostupné z: https://en.wikipedia.org/wiki/Pareto_efficiency
[23] BÄCK, Thomas. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. New York: Oxford University Press, 1996. ISBN 9780195099713.
[24] HYNEK, Josef. *Genetické algoritmy a genetické programování*. Praha: Grada, 2008. Průvodce (Grada). ISBN 9788024726953.
[25] DER HAUW, Koen van. *Evaluating and improving steady state evolutionary algorithms on constraint satisfaction problems* [online]. Computer Science Department of Leiden University, 1996 [cit. 2017-03-19]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.1568&rep=rep1&type=pdf. Master's thesis.
[26] MITCHEL, Melanie. *An Introduction to Genetic Algorithms*. Cambridge: MIT Press, 2002. ISBN 0262631857.
[27] GOLDBERG, David E. *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass.: Addison-Wesley Pub. Co., c1989. ISBN 0201157675.
[28] SARENI, B. a L. KRAHENBUHL. Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation* [online]. 2(3),

97-106 [cit. 2017-03-20]. DOI: 10.1109/4235.735432. ISSN 1089778x. Dostupné z: http://ieeexplore.ieee.org/document/735432/
[29] SIMON, Dan. *Evolutionary Optimization Algorithms*. John Wiley & Sons, 2013, s. 179-183. ISBN 9781118659502.
[30] HOLLAND, John H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor: University of Michigan Press, 1975. ISBN 0472084607.
[31] Mutation (genetic algorithm). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-21]. Dostupné z: https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)
[32] SONI, Nitasha a Tapas KUMAR. Study of Various Mutation Operators in Genetic Algorithms. *International Journal of Computer Science and Information Technologies* [online]. 2014 [cit. 2017-03-21]. ISSN 0975-9646. Dostupné z: http://ijcsit.com/docs/Volume%205/vol5issue03/ijcsit20140503404.pdf
[33] A.J., Umbarkar a Sheth P.D. CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW. *ICTACT Journal on Soft Computing* [online]. 2015, 06(01), 1083-1092 [cit. 2017-03-21]. DOI: 10.21917/ijsc.2015.0150. ISSN 09766561. Dostupné z: http://ictactjournals.in/ArticleDetails.aspx?id=2109
[34] SCHOLLE, Marek. *Veřejné rozhraní knihovny pro výpočet přesností v MIPS – MSS & PCL/MSPSR systémy*. Pardubice: Era a.s.
[35] Policies/Binary Compatibility Issues With C++. *KDE Community Wiki* [online]. [cit. 2017-03-22]. Dostupné z: https://community.kde.org/Policies/Binary_Compatibility_Issues_With_C%2B%2B
[36] Next C++ standard to arrive in 2017. *The H open* [online]. [cit. 2017-03-22]. Dostupné z: http://www.h-online.com/open/news/item/Next-C-standard-to-arrive-in-2017-1743138.html
[37] Java™ Platform, Standard Edition 8 API Specification. *Oracle* [online]. [cit. 2017-04-06]. Dostupné z: https://docs.oracle.com/javase/7/docs/api/overview-summary.html
[38] JYOTISHREE. *Knowledge based operation and problems representation in genetic algorithms* [online]. Kurukshetra University, 2012 [cit. 2017-04-10]. Dostupné z: http://shodhganga.inflibnet.ac.in/bitstream/10603/32680/16/16_chapter%206.pdf
[39] *PASIVNÍ RADIOLOKACE - VYUŽITÍ MULTILATERACE V CIVILNÍCH A VOJENSKÝCH APLIKACÍCH* [online]. NOVOZÁMSKÝ, Adam. Pardubice: ERA, s. 56 [cit. 2017-04-18]. Dostupné z: http://www.urel.feec.vutbr.cz/~sebestaj/MRAR/MRAR_P_CZ09.pdf
[40] GERAGHTY, John a Noraini Mohd RAZALI. *Genetic Algorithm Performance with Different Selection Strategies in Solving TSP* [online]. In: . 2011 [cit. 2017-04-18]. Dostupné z: https://www.researchgate.net/publication/236179245_Genetic_Algorithm_Performance_with_Different_Selection_Strategies_in_Solving_TSP

[41] HARRIES, Kim a Peter SMITH. *Exploring Alternative Operators and Sear ch Strategies in Genetic Pr ogramming* [online]. [cit. 2017-04-18]. Dostupné z: https://pdfs.semanticscholar.org/27bd/246d0a1e957bfa11c9a7c52d10fdfaa43658.pdf

7 Přílohy

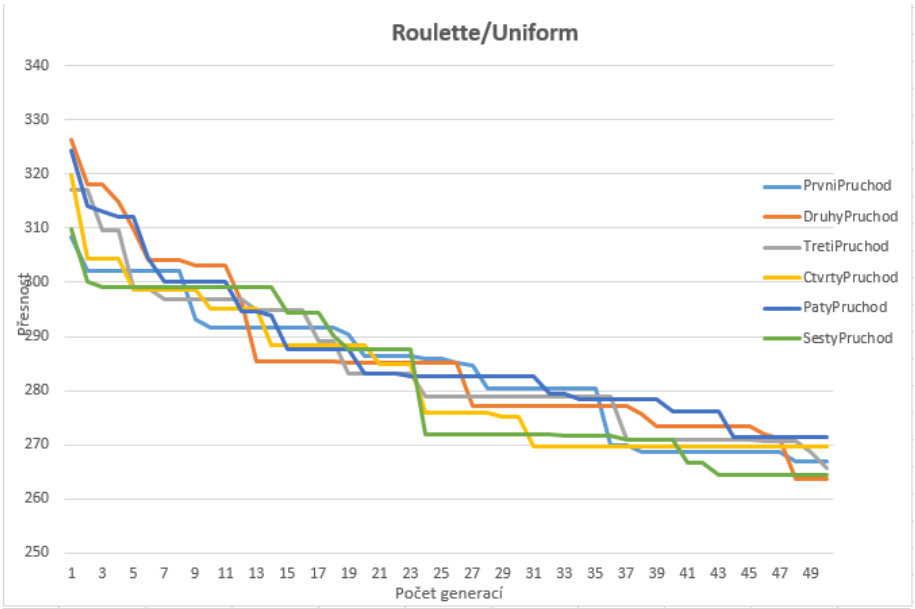
1) Tabulka nastavení parametrů - testovací scénář - nastavení algoritmu
2) Tabulka nastaveni parametrů - testovací scénář - reálný příklad
3) Grafy průběhů jednotlivých kombinací
4) CD - obsahující Java projekt s programem Analyzer.exe

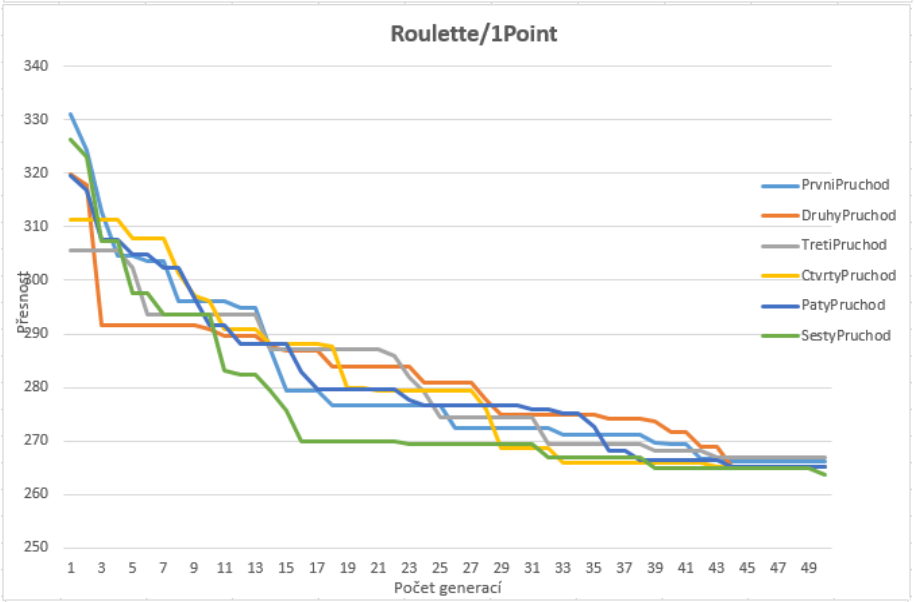
Tabulka nastaveni parametrů - testovací scénář - nastavení algoritmu

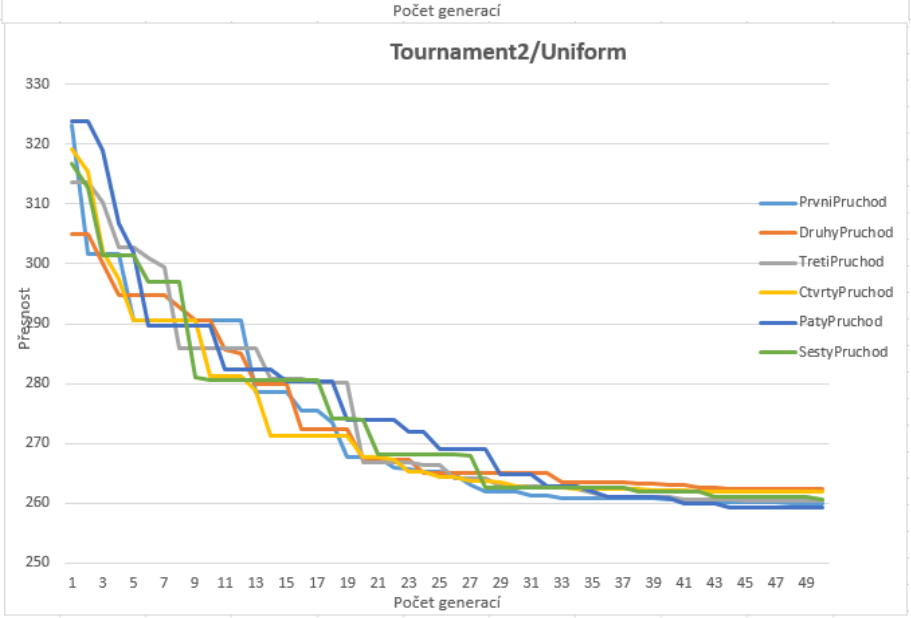
Název parametru	Hodnota parametru	Název parametru	Hodnota parametru
Number of receivers	6	**Cross rate**	1
Number of Interrogators	1	**Mutate rate**	0,05
Bottom left corner X	47,843968	**Population size**	200
Bottom left corner Y	10,443888	**Number of generations**	50
Top right X	52,242234	**Target Height**	2000,0
Top right Y	17,065489	**Height Deviation**	none
Dimension	50	**ToA Deviation**	0,99792458
From X	47,843968	**Interrogation Deviation**	5,0
From Y	10,443888	**Trans. Random Dev.**	8,6525317906
To X	52,242234	**Trans. System Dev.**	19,118674932
To Y	17,065489	**Atmospheric Error**	2.0e-5
Atmospheric model	vacuum		

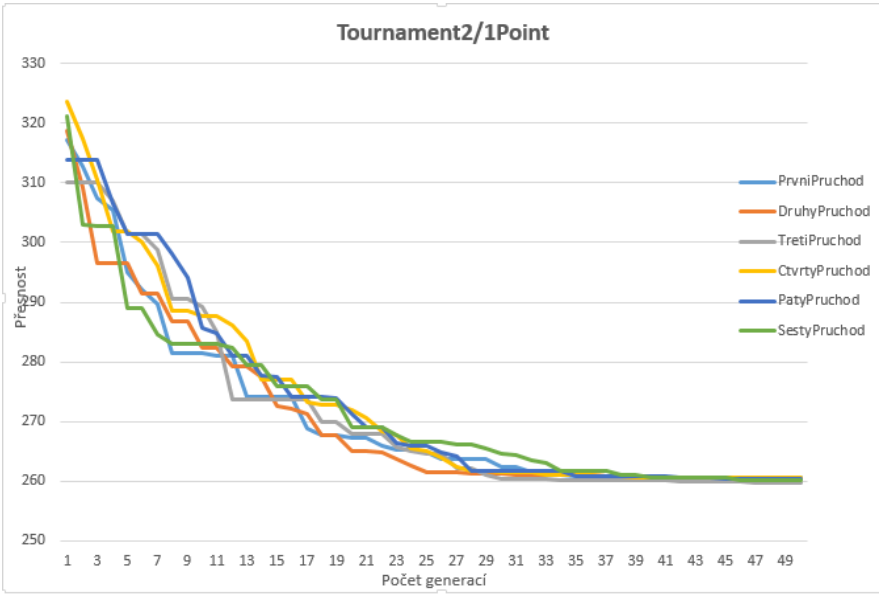
Tabulka nastaveni parametrů - testovací scénář - reálný příklad

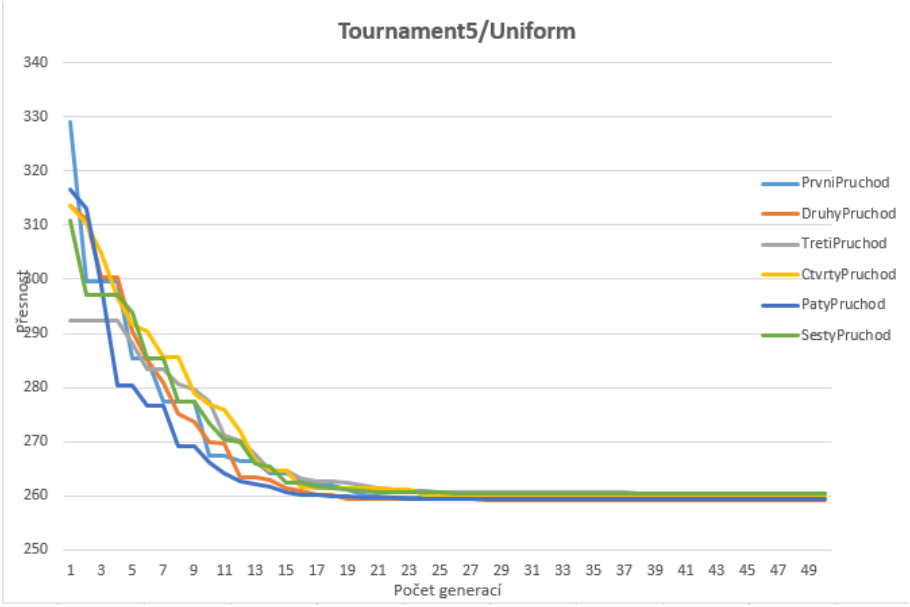
Název parametru	Hodnota parametru	Název parametru	Hodnota parametru
Number of receivers	5	**Cross rate**	1
Number of Interrogators	1	**Mutate rate**	0,05
Bottom left corner X	47,843968	**Population size**	200
Bottom left corner Y	10,443888	**Number of generations**	50
Top right X	52,242234	**Target Height**	2000,0
Top right Y	17,065489	**Height Deviation**	none
Dimension	50	**ToA Deviation**	0,99792458
From X	47,843968	**Interrogation Deviation**	5,0
From Y	10,443888	**Trans. Random Dev.**	8,6525317906
To X	52,242234	**Trans. System Dev.**	19,118674932
To Y	17,065489	**Atmospheric Error**	2.0e-5
Atmospheric model	vacuum	**Selection**	Tournament
Crossover type	Uniform	**Size of tournament**	2

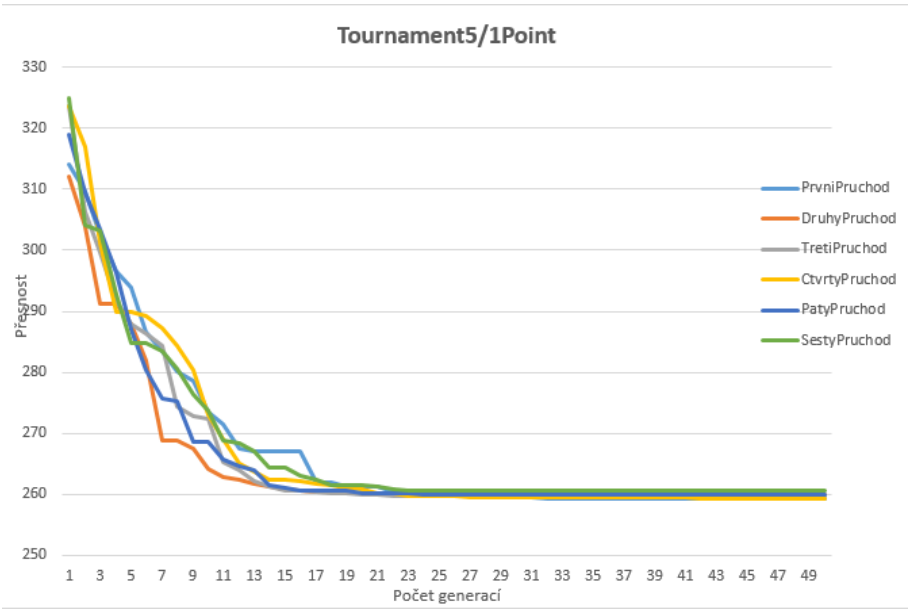

Roulette/Uniform
Přesnost
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod

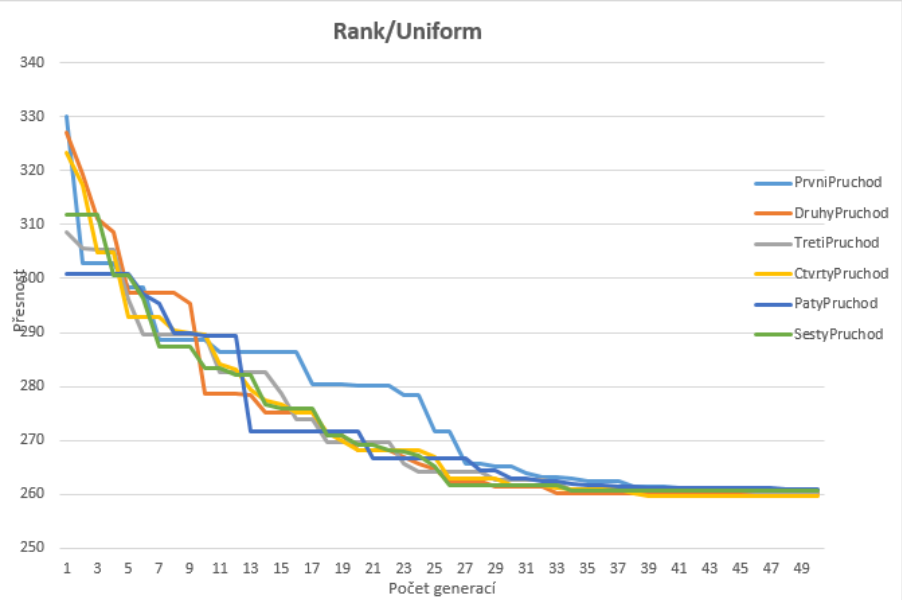

Roulette/1Point
Přesnost
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod

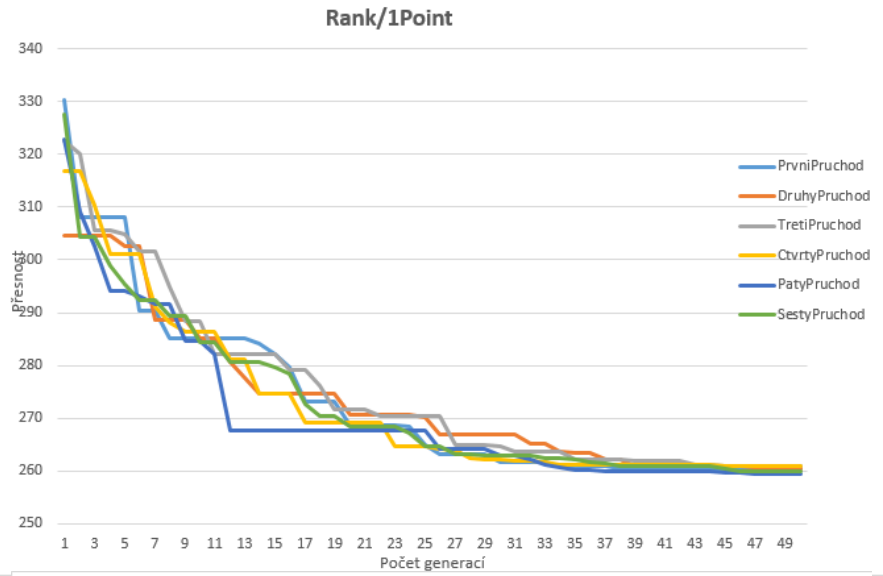

Tournament2/Uniform
Přesnost
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod


Tournament2/1Point
Přesnost
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod

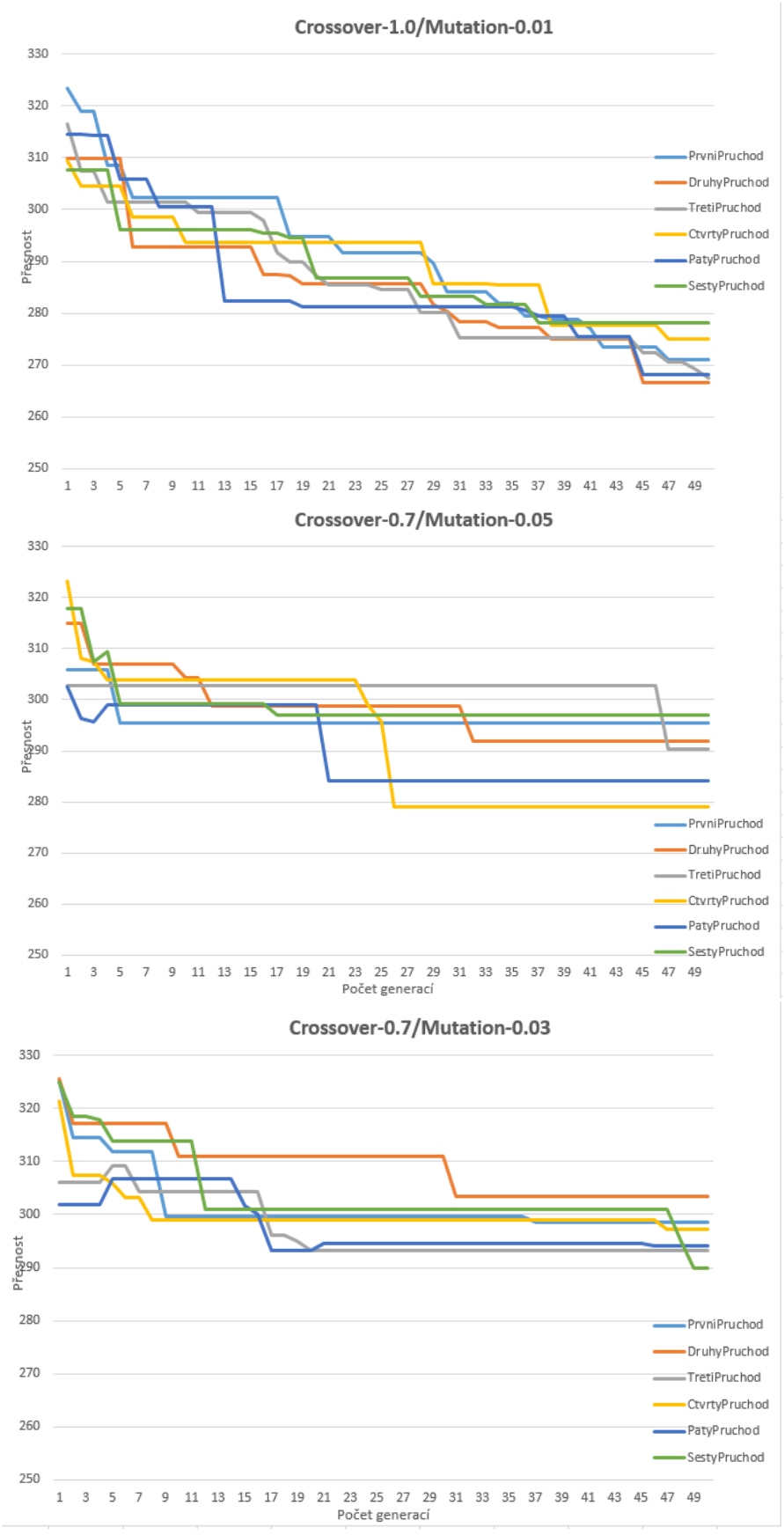

Tournament5/Uniform
Přesnost
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod

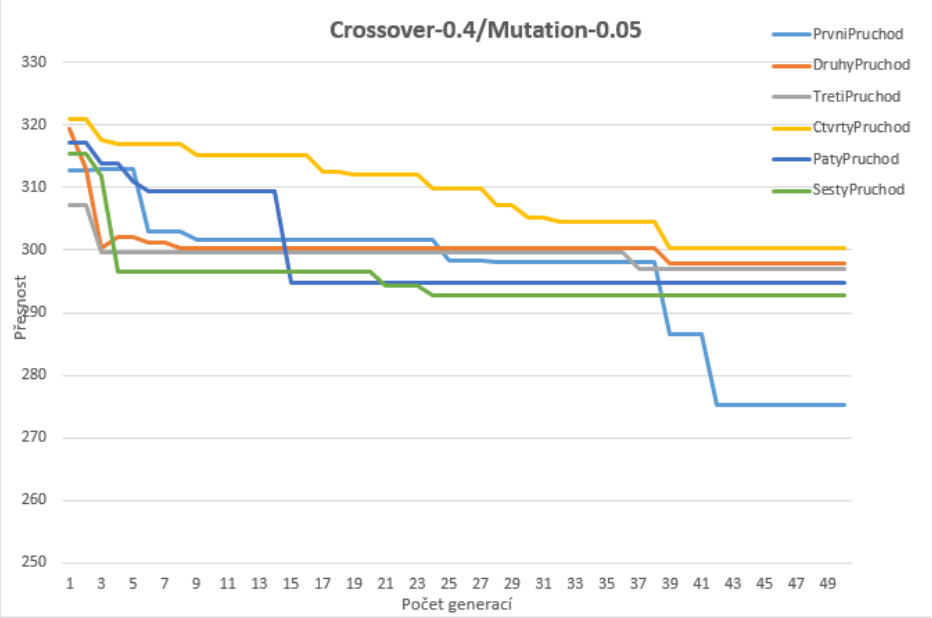

Tournament5/1Point
Přesnost
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod


Rank/Uniform
340
330
320
310
300
290
280
270
260
250
Přesnost
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod


Rank/1Point
340
330
320
310
300
290
280
270
260
250
Přesnost
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod


Crossover-1.0/Mutation-0.03
340
330
320
310
300
290
280
270
260
250
Přesnost
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
Počet generací
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod


Crossover-1.0/Mutation-0.01
Přesnost
330
320
310
300
290
280
270
260
250
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod
Crossover-0.7/Mutation-0.05
Přesnost
Počet generací
Crossover-0.7/Mutation-0.03
Přesnost
Počet generací


Crossover-0.4/Mutation-0.05
PrvniPruchod
DruhyPruchod
TretiPruchod
CtvrtyPruchod
PatyPruchod
SestyPruchod
330
320
310
300
290
280
270
260
250
Přesnost
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
Počet generací

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2016/2017

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bc. Hornolka Jaromír	Spartakiádní 1957, Sokolov	I1500684

TÉMA ČESKY:

Optimalizace rozmístění rádiových stanic.

TÉMA ANGLICKY.

Optimizing the distribution of radio stations.

VEDOUCÍ PRÁCE:

Ing. Barbora Tesařová, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cílem práce je nalézt a pokusit se implementovat automatické či semi-automatické metody optimalizace rozmístění stanic multilaterčního systému firmy ERA na základě daných kritérií pro oblasti krytí systému.
Osnova:

1 - Úvod
2 - Teoretická východiska
  2.1 - Multilaterační systém
  2.2 - Optimalizační algoritmy
  2.3 - Popis knihovny firmy ERA pro výpočty parametrů systému
3 - Praktická část
  3.1 - Implementace optimalizačních algoritmů
  3.2 - Testování optimalizačních algoritmů
4 - Shrnutí
5 - Závěr

SEZNAM DOPORUČENÉ LITERATURY:

Studijní materiály ERA

Podpis studenta: ..... **Datum:** 21.11.2016

Podpis vedoucího práce: ..... **Datum:** 21.11.2016