Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

METODY VÝPOČTU CENOVÉ FUNKCE NAD PROSTOROVÝMI DATY

Magisterská práce

Bc. Jan PŘÍBORSKÝ

Vedoucí práce Doc. Mgr. Jiří Dvorský, Ph.D.

Olomouc 2016
Geoinformatika

ANOTACE

Cílem diplomové práce je navržení metod výpočtu cenové funkce pro evoluční algoritmy s užitím prostorových dat. Práce je zaměřena na kombinaci restrikce měřící škály geodat pomocí evolučních algoritmů, informační entropie a tvarových metrik. Účelem této práce je nalézt vhodnou metodu, která kombinuje výše zmíněné, vizualizovat výsledky a vyhodnotit je.

V praktické části byl vytvořen nástroj ESMEA (Entropy and Shape Metrics Evolutionary Algorithms) v podobě Python skriptu, který umožňuje testovat různé typy evolučních algoritmů a měnit jejich vstupní parametry. Výsledky byly vizualizovány pomocí dvojice map, posteru a tabelárních výstupů.

KLÍČOVÁ SLOVA

evoluční algoritmus; cenová funkce; informační entropie; tvarové metriky; Python

Počet stran práce: 62

Počet příloh: 4 (z toho 2 vázané a 2 volné)

ANOTATION

The aim of this master thesis is to design a method of calculation of cost function for evolutionary algorithms with use of spatial data. This master thesis is focused on combination of geodata scale restriction using evolutionary algorithms, information entropy and shape metrics. The main objective of this master thesis is to find a suitable method for combination of mentioned methods, visualize and evaluate results.

During work on this thesis, there was created a Python script named ESMEA (Entropy and Shape Metrics Evolutionary Algorithms), which allows to test various types of evolutionary algorithms and change input values. Results were visualised as a pair of maps, poster and tabular outputs.

KEYWORDS

evolutionary algorithm; cost function; information entropy; shape metrics; Python

Number of pages: 62

Number of appendixes: 4

Prohlašuji, že

- diplomovou práci včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu.

- jsem si vědom, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo,

- beru na vědomí, že Univerzita Palackého v Olomouci (dále UP Olomouc) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užívat (§ 35 odst. 3),

- souhlasím, aby jeden výtisk diplomové práce byl uložen v Knihovně UP k prezenčnímu nahlédnutí,

- souhlasím, že údaje o mé diplomové práci budou zveřejněny ve Studijním informačním systému UP,

- v případě zájmu UP Olomouc uzavřu licenční smlouvu s oprávněním užít výsledky a výstupy mé diplomové práce v rozsahu § 12 odst. 4 autorského zákona,

- použít výsledky a výstupy mé diplomové práce nebo poskytnout licenci k jejímu využití mohu jen se souhlasem UP Olomouc, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly UP Olomouc na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Olomouci dne 12. 8. 2016 Bc. Jan Příborský

Děkuji vedoucímu práce doc. Mgr. Jiřímu Dvorskému, Ph.D. za podněty a vedení diplomové práce. Děkuji rodině, která mi byla oporou v průběhu celého studia. Za poskytnutá data děkuji katedře Geoinformatiky UP Olomouc a Ing. Lence Skanderové.

Vevázaný originál **zadání magisterské práce**.

OBSAH

ÚVOD

Primární (naměřená) kvantitativní data jsou v oblasti geoinformatiky mnohdy velmi objemná, tento fakt znesnadňuje jejich zpracování i konečnou vizualizaci. Pro další použití je vhodné objemná data redukovat, k tomuto účelu existuje velké množství již zavedených postupů a návazných algoritmů. Jedná se jak o metody statistické, tak i metody zaměřené na transformací dat do intervalů hodnot (restrikci měřící škály dat) podle určitých specifik redukovaných dat. V této oblasti lze zmínit např. práce Jenks (1967), Kaňok (1992) nebo Robinson (1995). Jistou alternativu představují restrikce měřící škály dat založené na evolučních algoritmech. Metodu používá např. Pászto (2015) při aplikaci na optimální rozdělení primárních dat do intervalů vzhledem k minimalizaci ztráty informace dat.

Pomocí optimalizačních úloh využívajících evoluční algoritmy lze řešit velké množství různých typů problémů, které by běžnými způsoby v reálném čase nebyly řešitelné. Primárním úkolem evolučních algoritmů je nalezení extrémů matematických funkcí, tedy nalezení co nejlepších řešení daného problému. Prakticky lze vyjádřit tuto činnost jako hledání takových parametrů navrhovaného systému (funkce), aby byl výsledek v určitém smyslu co nejlepší. Optimalizační úloha založená na evolučních algoritmech je tedy takovou úlohou, kde se řešitel snaží nalézt minimum / maximum matematické funkce dosazením vhodných proměnných. Tuto funkci lze v obecném případě definovat podle vzorce 1 (Zelinka a kol., 2009).

$$f: D \rightarrow R \tag{1}$$

Optimalizační funkce f se nazývá účelová, nebo také **cenová**. Množina D je parametrickým prostorem. Cenová funkce tedy při procesu hledání nejlepších řešení definuje to, co je myšleno pod pojmem „nejlepší“. Jednotlivé parametry cenové funkce určují vlastnosti daného řešení. Parametry obvykle nabývají hodnot reálných čísel, v tom případě lze množinu D ztotožnit s n-rozměrným reálným prostorem $D = R^n$ (Koška, 2014). Cenová funkce je založena na výpočtu rozličných kritérií. Podle počtu kritérií lze dělit optimalizační úlohy na jednokriteriální a vícekriteriální (Petříková, 2013).

Tato diplomová práce je zaměřena především na konstrukci vícekriteriální cenové funkce, závislé na ztrátě informace v datech a tvarových metrikách. Tato funkce je dále aplikována v různých typech evolučních algoritmů.

1 CÍLE PRÁCE

Cílem diplomové práce bylo navrhnout metody výpočtu cenové funkce pro evoluční algoritmy nad prostorovými daty, dále provést experimenty, výsledky vizualizovat a zhodnotit. Nejdříve bylo nutné nastudovat problematiku výpočtu cenové funkce, tvarových metrik, informační entropie, a také se seznámit s různými typy evolučních algoritmů a specifiky jejich tvorby.

Praktická část diplomové práce je zaměřena na návrh cenové funkce, výběr a navržení jednotlivých evolučních algoritmů a vytvoření nástroje, umožňujícího provádět experimenty na evolučních algoritmech s navrženou cenovou funkcí.

Výsledky byly uloženy do tabelární formy, vizualizovány ve formě map a byl vytvořen poster. Všechny výsledky byly slovně zhodnoceny v textu práce. K prezentaci práce byly zhotoveny internetové stránky a všechna vstupní data a výstupy přiloženy na DVD-ROM.

2 METODY A POSTUPY ZPRACOVÁNÍ

Při tvorbě diplomové práce byl kladen důraz především na nastudování dostupných knižních a internetových zdrojů. Bylo nutné osvojit si jak problematiku informační entropie, tak i tvarových metrik a v neposlední řadě vědomosti týkající se konstrukce evolučních algoritmů. Mezi nejhodnotnější zdroje patřily práce Pászto (2015), Hartmanová (2013) a pro uvedení do problematiky evolučních algoritmů byla autorovi velkým přínosem publikace Zelinka a kol. (2009).

Velmi důležitým aspektem tvorby práce bylo vytvořit návrh cenové funkce a následně jej aplikovat do zvolených typů evolučních algoritmů. Konkrétní realizace řešení a následné testování výkonu bylo velmi časově náročné.

Použité metody

Informační entropie

Pojem informační entropie byl poprvé definován podle Shannon (1948) následovně: „Entropie je střední hodnota míry informace k odstranění neurčitosti, která je dána konečným počtem vzájemně vylučujících se jevů.“ Míra entropie tedy hodnotí množství informace ve zprávě. Podle Voženílek (2005) je možné za zprávu v tomto smyslu pokládat i mapu.

Tvarové metriky

Jedná se o metody používané zejména v krajinné ekologii, ale i v obecně geografických disciplínách. Umožňují kvantitativní hodnocení tvarů ploch reálného světa pomocí výpočtů jejich tvarových charakteristik (Pászto, 2015). Pro hodnocení tvarových charakteristik ploch jsou v práci užity tvarové metriky **Fractal Dimension Index**, **Shape Index**, **Perimeter-Area Ratio**[1] a **Normalized Detour Index**[2].

Evoluční algoritmy

Evoluční algoritmy jsou termínem, který sjednocuje různé přístupy využívající jednoduché modely Darwinovy evoluční teorie vývoje populací pro optimalizaci řešení problému. Jsou charakterizovány využíváním heuristických procesů, které modifikují populaci tak, aby se její vlastnosti zlepšovaly (Volná, 2012). V práci je použito několika variant **genetického algoritmu** a **diferenciální evoluce**.

---

[1] Dostupné z: http://www.umass.edu/landeco/research/fragstats/documents/Metrics/Shape%20Metrics/SHAPE%20METRICS.htm

[2] Dostupné z: http://clear.uconn.edu/tools/Shape_Metrics/graphics/Detour.htm

Programovací jazyk Python

Konečná realizace nástroje ESMEA (Entropy and Shape Metrics Evolutionary Algorithms) umožňujícího provádění pokusů s navrženou cenovou funkcí (spouštění různých typů evolučních algoritmů s různými vstupy), je realizována pomocí programovacího jazyka Python. Python je objektově orientovaný skriptovací jazyk vyvíjený pod Open Source licencí. Zdarma nabízí instalační balíky pro většinu používaných platforem.

Metoda párového srovnávání

Metoda párového srovnávání (Fullerova metoda) je hojně využívaná pro stanovení vah kritérií. Funguje na principu vzájemného porovnání důležitosti mezi dvojicemi kritérií. Výsledné hodnoty představují normované váhy kritérií, které jsou v rozmezí intervalu $\langle 0, 1 \rangle$. Čím vyšší hodnota váhy je, tím je kritérium k důležitější. Počet srovnání N lze vypočítat následovně (Drážná, 2014):

$$N = \binom{k}{2} = \frac{k(k-1)}{2} \tag{2}$$

Následně je sestaven tzv. Fullerův trojúhelník, který představuje uspořádání do k -1 dvojřádků. Uživatel vždy z dané dvojice kritérií vybere důležitější (v případě stejné důležitosti obě). Počet zakroužkování kritéria k_i se rovná hodnotě n_i. Váha je vypočtena podle následujícího vzorce (Drážná, 2014):

$$v_i = \frac{n_i}{N}; \quad i = 1, 2, \ldots, k \tag{3}$$

Kartografické metody

Vizualizace výsledků i poster byly zpracovány základními metodami tematické kartografie (metoda kartogramu).

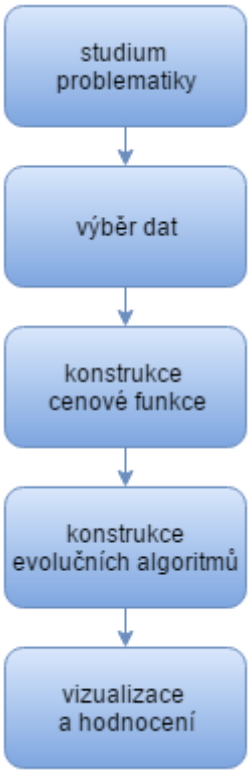
Postup zpracování

Zpracování diplomové práce bylo rozděleno do pěti základních fází (viz obr. 1). Jednalo se v první řadě o studium literatury zabývající se problematikou témat, která jsou primárně v práci řešena (informační entropie, tvarové metriky, evoluční algoritmy). Následoval výběr vhodných datových zdrojů pro budoucí testování funkčnosti a provádění experimentů v nástroji ESMEA.

Stěžejním krokem celého procesu bylo navržení cenové funkce umožňující výpočet informační entropie a tvarových metrik pro hledání optimálních řešení. V dalším kroku

byla navržená cenová funkce implementována do několika evolučních algoritmů a použitím programovacího jazyka Python ve verzi 2.7 byl vytvořen nástroj ESMEA. Pomocí kombinace tohoto nástroje a vstupních dat byly provedeny experimenty (viz kapitola 7) a výsledky těchto experimentů zhodnoceny.

V závěrečné fázi byly výsledky vizualizovány v podobě dvojice tematických map (viz přílohy 1 a 2). Dále byl vytvořen poster obsahující grafickou vizualizaci jednotlivých kroků provedených při řešení diplomové práce, vysvětlení jednotlivých procesů a ukázku výsledků (viz příloha 3). Vstupní i výstupní data, nástroj ESMEA, mapové výstupy a poster byly spolu s textem práce a internetovými stránkami uloženy na DVD-ROM (viz příloha 4).



Obr. 1 Fáze tvorby diplomové práce

(zdroj: vlastní zpracování).

3 POUŽITÁ DATA A PROGRAMY

Použitá data

V rámci testování a experimentů v nástroji ESMEA bylo použito dat průměrné roční teploty vzduchu v České republice za období let 1961 až 2001. Tato data byla na Katedře geoinformatiky UP Olomouc použita již dříve v práci Pászto (2015). Jednalo se o ESRI grid s velikostí buňky 100 x 100 m. Velikost celého rastru představovala 4 865 sloupců a 2 780 řádků. Jednotlivé pixely obsahovaly hodnoty průměrné teploty zaokrouhlené na celé stupně Celsia. Rozsah teplot byl od jednoho do deseti stupňů Celsia (viz tabulka 1). Grid vykazoval střední hodnotu entropie H_0 = 2,135 bitů na jednu buňku gridu.

Tab. 1 Počet pixelů podle hodnoty °C (zdroj: vlastní zpracování)

°C	Počet pixelů
1	1968
2	12680
3	50204
4	160566
5	468285
6	1564203
7	3128512
8	2135303
9	382978
10	950

Použité programy

Prvotní práce spjaté se zjištěním vlastností vstupního rastru byly vykonány v programu ArcMap 10.3.1. V programovém prostředí ArcGIS bylo také zjištěno, které nástroje by bylo možné aplikovat do výpočtů evolučních algoritmů.

Po navržení následnosti jednotlivých procesů, byl nástroj ESMEA vyvíjen v prostředí Python IDLE ve verzi 2.7.10 32-bit. Zde bylo důležitým krokem aplikování jednotlivých knihoven. Využito bylo následujících knihoven:

- **ArcPy** – slouží pro zpracování prostorových dat,
- **Collections** – knihovna implementující vysoce výkonné datové typy,
- **Copy** – podporuje tzv. deep copy operace, tedy činnost, kdy jsou kopírovány složitější struktury,

- **Math** – knihovna obsahující větší množství matematických funkcí (autor zjistil, že některé operace prováděné pomocí této knihovny mohou být vykonávány rychleji než pomocí implicitních),
- **NumPy** – může být využita pro operace s vektorově orientovanými daty,
- **Random** – knihovna umožňující generovat pseudo-náhodná čísla,
- **Time** – užitím knihovny je možné pracovat s časovými údaji.

Knihovna ArcPy byla pro celou diplomovou práci stěžejní. Pomocí volání nástrojů implementovaných knihovnou ArcPy bylo nástroji ESMEA umožněno přistupovat a dále zpracovat prostorová data. Konkrétně byly v ESMEA aplikovány tyto nástroje:

- **Add Field** – přidá nové pole do již existující tabulky třídy prvků,
- **Build Raster Attribute Table** – vytvoří tabulku obsahující jedinečné hodnoty pixelů rastru spolu s jejich počty,
- **Exists** – slouží ke zjištění, zda daný prvek (třída prvků, shapefile, tabulka či dataset) existují,
- **Join Field** – slouží ke spojení záznamů tabulek na základě stejného identifikátoru,
- **Minimum Bounding Geometry** – vytvoří polygonovou třídu prvků reprezentující minimální ohraničující geometrii jednotlivých prvků vstupní třídy prvků,
- **Polygon Neighbors** – vytváří tabulku vyjadřující prostorové vazby a statistické charakteristiky mezi sousedními prvky,
- **Raster to Polygon** – nástroj slouží k transformaci rastrových dat na polygonová,
- **Search Cursor** - umožňuje čtení záznamů v třídě prvků nebo tabulce,
- **TestSchemaLock** – umožňuje zjistit, zda je možné přistupovat k datům,
- **Update Cursor** - umožňuje čtení a zápis záznamů v třídě prvků nebo tabulce.

V průběhu konstrukce nástroje ESMEA bylo nutné vybírat takové knihovny a funkce, které umožňovaly co nejrychlejší provedení požadovaných operací. K testování rychlosti provedení jednotlivých nástrojů užitých v kódu byla používána již zmíněná knihovna Time.

Při samotném testování v již funkčním nástroji ESMEA bylo nutné určit sílu jednotlivých kritérií. Tento proces byl realizován pomocí metody Párového srovnávání v programu Microsoft Excel 2016.

Mapové výstupy byly vytvářeny v programovém prostředí ArcGIS 10.3.1 a grafické výstupy ve formě diagramů byly zpracovány pomocí služby Draw.io[3]. Celkový vzhled diagramů a komponent map byl dotvářen v grafickém editoru PhotoFiltre 7[4].

[3] Dostupné z: https://www.draw.io/

[4] Dostupné z: http://www.photofiltre-studio.com/pf7-en.htm

4 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Metoda restrikce škály dat pomocí evolučních algoritmů je použita například v práci Pászto (2015), kde autor cílí na výpočet ztráty informace pomocí metod informační entropie. Proces zisku nejlepšího řešení je realizován pomocí genetického algoritmu. Zmíněný postup se tato práce snaží obohatit o aplikaci tvarových metrik a následně jej testovat na různých typech evolučních algoritmů. V následujících podkapitolách jsou rozebrány jednotlivé součásti diplomové práce, které si autor musel osvojit.

4.1 Cenová funkce

Optimalizační úlohy využívající k řešení problému evoluční algoritmy nutně potřebují k posouzení kvality řešení určitý ukazatel. Podle Hynek (2008) se k tomuto účelu zavádí tzv. hodnotící funkce, která umožňuje ohodnotit jakékoliv řešení z prostoru všech možných řešení daného problému. Taková funkce se nazývá účelová nebo taktéž **cenová**. Tato funkce je zobrazením z množiny všech řešení X do množiny reálných čísel R (viz vzorec 4).

$$f_{(x)}: X \rightarrow R \tag{4}$$

Zelinka a kol. (2009) uvádí, že účelovou funkcí se rozumí taková funkce, jejíž optimalizace (zjištění maxima či minima) vede k nalezení optimálních hodnot jejích argumentů. V některých případech je tato funkce nazývána funkcí cenovou – z anglického spojení cost function. Funkce f má v bodě x_0 lokální maximum, existuje-li pro všechna x v tomto okolí platnost vztahu (viz vzorec 5).

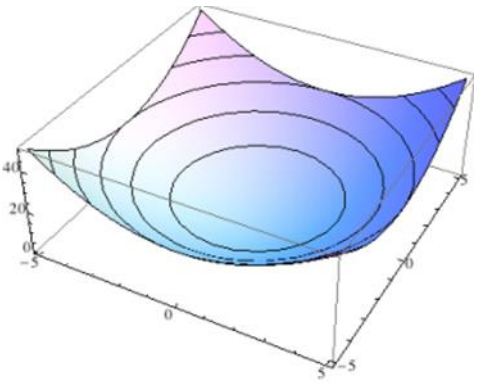
$$f_{(x)} \leq f_{(x_0)} \tag{5}$$

Funkce f má v bodě x_0 ostré lokální maximum, existuje-li pro všechna x v tomto okolí mimo bod $x = x_0$ platnost vztahu (viz vzorec 6).

$$f_{(x)} < f_{(x_0)} \tag{6}$$

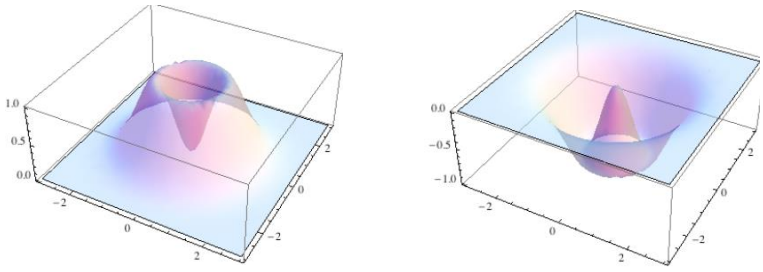
Cenovou funkci je možné reprezentovat v podobě geometrického problému (viz obr. 2). Při řešení tohoto problému je hledána nejnižší (minimální) nebo nejvyšší (maximální) pozice ležící na ploše v $N + 1$ rozměrném prostoru. Plocha v tomto prostoru se nazývá „hyperplocha“ či „prostor možných řešení“. Počet dimenzí N je definován počtem optimalizačních argumentů (kritérií) účelové funkce. Příkladem může být účelová funkce

s pěti argumenty. V tomto případě je hledán extrém na pětirozměrné ploše v šestirozměrném prostoru. Šestá dimenze je reprezentována návratovou hodnotou účelové funkce (Zelinka a kol., 2009).



Obr. 2 Cenová funkce s globálním extrémem na souřadnicích (0, 0)
(zdroj: http://arg.vsb.cz/data/Vyuka/04%20BIV%20Evoluce%20-%20UcFce.pdf).

Cenovou funkci koncipovanou pro hledání minimálních návratových hodnot je možné snadným způsobem transformovat pro hledání maximálních návratových hodnot, a to vynásobením celé funkce hodnotou *-1* (Volná, 2012). Tímto postupem lze invertovat maximum na minimum bez měny souřadnic extrému (viz obr. 3).



Obr. 3 Konverze extrému vynásobením cenové funkce hodnotou „-1“
(zdroj: http://arg.vsb.cz/data/Vyuka/04%20BIV%20Evoluce%20-%20UcFce.pdf).

Jednoduché optimalizační úlohy, kde se nachází pouze jedno kritérium rozhodování, jsou řešeny jednokriteriálními (angl. single-objective) funkcemi. Oproti tomu řešení komplexních problémů v některých případech vyžaduje optimalizaci dvou a více funkcí. Tento přístup se nazývá vícekriteriální optimalizace (angl. multi-objective) (Savic, 2002).

Problematikou vícekriteriální optimalizace se podrobněji zabývá například To, Korn (1997), Tan, Khor, Lee (2005) nebo Deb (2001) či Zitzler (1999).

4.2 Evoluční algoritmy

Evoluční algoritmy jsou stochastické optimalizační algoritmy, které vycházejí ze základních principů Darwinovy a Mendelovy teorie evoluce. Podle této teorie se organismy v průběhu generací vyvíjejí a přizpůsobují svému okolí. Tento vývoj je zapříčiněn především konkurenčním prostředím a vzájemnou interakcí s okolními organismy. Obdobný přístup používají i evoluční algoritmy při řešení komplexních úloh, které by nebyly klasickými způsoby snadno řešitelné. Evoluční algoritmy jsou tedy takové optimalizační algoritmy, které simulují průběh evoluce pro dosažení kvalitního řešení problému. K nalezení kvalitních řešení využívají prvků heuristiky, tedy postupů, při kterých se využívá prvků náhodně generovaných. Oproti deterministickým algoritmům evoluční algoritmy nezaručují nalezení nejlepšího řešení, ale umožňují v reálném čase nalézt aplikovatelné řešení.

Primární vlastností evolučních algoritmů je prohledávací strategie založena na populacích. V populaci se nachází jednotlivé genotypy. Genotyp se skládá z genů a představuje zakódovaný fenotyp nebo skupinu fenotypů. Fenotypem je myšlen kandidát na řešení úlohy (přípustné řešení). V průběhu chodu optimalizačního procesu nabývají geny většinou numerických hodnot z odpovídajících domén hodnot, a to takových, aby genotyp mohl být v dekódovacím procesu převeden na odpovídající fenotyp. Ohodnocení fenotypu stanovuje návratová hodnota hodnotící funkce aplikované na korespondující genotyp. Evoluční algoritmy preferují genotypy s nejvyšším / nejnižším ohodnocením (podle typu optimalizační úlohy).

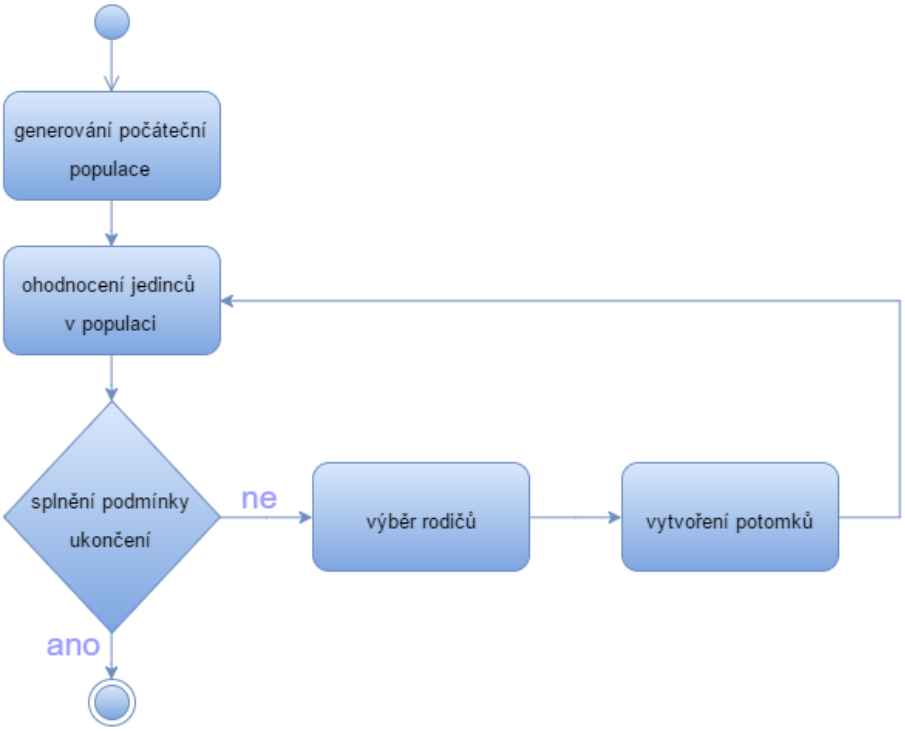
Jednotlivé genotypy jsou vytvářeny ve sledu generací pomocí tzv. **operátorů**. Příkladem operátoru může být selekce, mutace, křížení, inverze aj. Dalšími důležitými vlastnostmi evolučních algoritmů jsou vzájemná soutěživost jedinců v populaci a úprava genetického kódu. Jednotlivé evoluční operátory jsou aplikovány se záměrem, aby se populace postupně vyvíjela směrem ke genotypu, který bude dosahovat co nejvyšší / nejnižší návratové hodnoty ohodnocující funkce (Volná, 2012).

Podle Zelinka (2009) je hlavní ideou evolučních algoritmů předávání rodičovského genetického kódu novým potomkům, kteří podléhají při svém vzniku mutacím. Rodiče a potomci nevhodní pro aktuální životní prostředí vymírají „cyklicky“ po tzv. generacích, čímž uvolňují místo novým lepším potomkům. Ti se následně stávají novými rodiči.

Pojmem evoluční algoritmus je do jisté míry zavádějící, asociuje název jednoho algoritmu. Nicméně, jedná se o zastřešující termín, který představuje soubor celé řady různých typů optimalizačních algoritmů. Tyto algoritmy mají podobnou strukturu, užívají podobných operací, ale mohou se lišit v rozličných aspektech (reprezentace chromozomu, úroveň datové struktury nebo odlišnost při implementaci evolučních operátorů).

Základní schéma evolučních algoritmů (viz obr. 4) je principiálně velmi jednoduché. Podle Volná (2012) se skládá z generování počáteční populace, následného zavedení

cyklu, ohodnocení všech jedinců počáteční populace, aplikování selekce pro výběr rodičů, tvorby potomků pomocí jednotlivých evolučních operátorů a následné opakování cyklu až do stavu, kdy je splněna ukončovací podmínka.



Obr. 4 Obecný princip fungování evolučního algoritmu

(zdroj: vlastní zpracování).

4.2.1 Vybrané typy evolučních algoritmů

V této podkapitole jsou rozebrány některé typy evolučních algoritmů. Autor zařadil jako zástupce velmi jednoduchého příkladu algoritmus **Náhodného prohledávání** (tzv. Slepý algoritmus). Dále jsou zařazeny algoritmy, které Štefan (2011) považuje za nejpoužívanější. Jedná se o **Genetické algoritmy** a **Diferenciální evoluci**.

Náhodné prohledávání

Náhodné prohledávání (někdy také Slepý algoritmus), je typ evolučního algoritmu, který opakovaně generuje náhodná řešení. Tato náhodná řešení jsou omezena prohledávacím prostorem. Řešení je zapamatováno, pakliže je lépe ohodnoceno než řešení předchozí. Náhodné prohledávání je příkladem jednoho z nejjednodušších stochastických algoritmů. Principem hledání globálního minima je užití heuristiky, která lze popsat jako „generuj cokoli přípustného“.

Obecný průběh Slepého algoritmu lze vyjádřit velmi jednoduše podle Tvrdík (2004). Nejdříve jsou načteny následující vstupní parametry:

- f – hodnotící funkce,
- a, b – vektory udávající prohledávací prostor,
- t – počet generací

Po načtení vstupních parametrů jsou definovány proměnné x (reprezentuje nejlepší nalezenou vstupní hodnotu funkce *f*) a fx (udává nejlepší výsledek funkce *f*). Proměnná fx je nastavena na nejvyšší přípustnou hodnotu daného datového typu. Následně je spuštěn cyklus typu FOR, kde proměnná t udává počet iterací. V každé iteraci je generován náhodný vstup funkce f omezený vektory a, b, ten je následně vložen do funkce f pro získání výsledku. Pokud výsledek nabývá nižší hodnoty než aktuální fx, je proměnné fx přiřazena hodnota výsledku, proměnné x hodnota vstupu a algoritmus pokračuje další iterací. Pakliže je hodnota výsledku vyšší než aktuální fx, algoritmus pokračuje další iterací bez změny fx a x. Po ukončení všech iterací cyklu typu FOR jsou uživateli předány aktuální hodnoty proměnných x a fx.

Důležitou vlastností Slepého algoritmu je to, že lze dokázat jeho konvergenci ke globálnímu extrému. Platí, že s rostoucím počtem iterací se aktuálně nalezené řešení x blíží globálnímu minimu x^* (viz vzorec 7). Zde $\| (x - x^*) \|$ představuje vzdálenost nalezeného řešení x od globálního minima x^* (norma vektoru $x - x^*$) a $P(\| (x - x^*) \| < \epsilon \mid t)$ je pravděpodobnost jevu $(\| (x - x^*) \| < \epsilon)$ za podmínky, že bylo provedeno t iterací (Tvrdík, 2004).

$$\lim_{t \to \infty} P(\| (x - x^*) \| < \epsilon \mid t) = 1 \qquad \epsilon > 0 \tag{7}$$

Genetické algoritmy

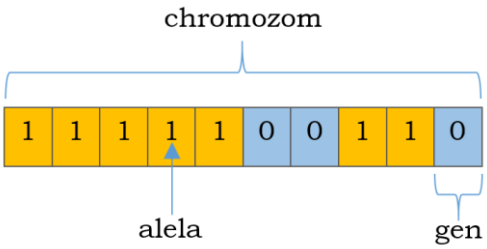
Genetické algoritmy jsou podle Studnička (2010) nejrozšířenějším typem evolučních algoritmů. Principy fungování genetických algoritmů jsou aplikací již zmíněných postupů skupiny evolučních algoritmů. Principiálně se opět jedná o postupnou tvorbu generací různých řešení daného problému. V průběhu řešení algoritmu se zachovává populace jedinců. Každý z těchto jedinců představuje jedno řešení daného problému. Průběhem evoluce se řešení zlepšují.

Velmi často je řešení reprezentováno (zakódováno) binárními čísly, není však neobvyklé použití odlišné reprezentace (celočíselná, reálná, permutační, strom, vektor, pole, matice atd.). Při spuštění genetického algoritmu (v první generaci) je populace vytvořena z náhodných členů. V průběhu přechodu do nové generace je pro každého jedince spočtena tzv. **fitness**, která udává kvalitu jedince, tedy daného řešení. V každé

generaci jsou jedinci modifikováni (pomocí operátorů **mutace** a **křížení**), díky tomuto postupu vzniká nová generace. Zmíněný postup se opakuje a tím se zlepšuje i kvalita řešení v populaci. Genetické algoritmy jsou ukončeny při dosažení dostačující kvality řešení, popřípadě po nastavené době (počtu generací) nebo pokud nedochází v horizontu několika generací k vylepšení kvality řešení.

K tomu aby bylo možné přesněji popsat genetické algoritmy, je nutné se blíže seznámit s jejich terminologií. Mezi základní termíny v oblasti genetických algoritmů patří následující (Studnička, 2010):

- **Jedinec (Individual)** - nositel genetické informace, reprezentace jednoho z přípustných řešení,
- **Chromozom (Genome)** - obecná genetická informace ve tvaru řetězce, jedná se o sekvenci symbolů (čísla, znaky nebo jejich kombinace), která reprezentuje dané řešení, respektive jedince (viz obr. 5),
- **Gen (Gene)** - určitá pozice (umístění) v chromozomu,
- **Alela (Allele)** - konkrétní symbol (kombinace symbolů) v chromozomu, je definována přípustnými hodnotami,
- **Rodič (Parent)** - jedinec vstupující do rekombinace nebo křížení (je vybrán metodou selekce),
- **Potomek (Offspring)** - jedinec vytvořený metodou rekombinace nebo křížením dvou nebo více rodičů,
- **Ohodnocení (Fitness)** - ohodnocení kvality jedince v generaci (kvalitnější jedinci mají větší pravděpodobnost zachování),
- **Křížení (Crossover)** – operátor určený k sestavení nových jedinců (potomků) zkombinováním vybraných jedinců dané generace (rodičů),
- **Mutace (Mutation)** – operátor slouží ke změně hodnoty alely jednoho nebo více genů v chromozomu,
- **Rekombinace (Recombination)** - proces, při kterém je provedeno křížení a následně mutace,
- **Selekce (Selection)** – operátor výběru jedinců, kteří jsou zachováni nebo určeni pro křížení,
- **Populace (Population)** - soubor aktuální skupiny jedinců, na kterou jsou aplikovány operace evolučních algoritmů.



Obr. 5 Struktura binárního chromozomu

(zdroj: vlastní zpracování).

Jak již bylo uvedeno v přehledu termínů z oblasti genetických algoritmů, genetické algoritmy používají několik základních operátorů. Tyto operátory jsou opět odvozeny z evoluční teorie. Jedná se o velmi zjednodušené verze podobných procesů, které probíhají při evoluci živých organizmů.

Prvním z těchto operátorů je **selekce**, ta slouží k výběru jedinců z populace. Vybraní jedinci se dále stávají rodiči pro užití dalších operátorů a následné vytvoření potomků. V některých případech je vybraným jedincům umožněno postoupit do další generace beze změny. Je velmi důležité správné nastavení selekce, tedy to jakým způsobem budou jedinci vybíráni. Podle Němec (2016) je výběr pouze nejsilnějších jedinců populace nevhodnou formou selekce. V tomto případě je totiž velmi pravděpodobné uvíznutí algoritmu v lokálním extrému, tedy řešení kvalitním, ne však nejlepším možném. Vhodné je tedy selekci aplikovat takovým způsobem, aby byli vybíráni jak jedinci s nejlepším ohodnocením, ale i jedinci představující méně kvalitní řešení. Tímto způsobem je zajištěna dostatečná genetická různorodost populace a tím zamezení uvíznutí algoritmu v lokálním extrému.

Dalším hojně užívaným genetickým operátorem je **křížení**. Stejně jako v reálném světě je i v oblasti genetických algoritmů křížení produktem kombinace genetického materiálu rodičů. Specifickou vlastností genetických algoritmů je však fakt, že se nemusí jednat pouze o dva rodiče, může jich být použito více. Kombinací genetického materiálu rodičů vznikají potomci. Není pravidlem, že kombinací velmi kvalitních rodičů vnikne i kvalitní potomek. Po vytvoření potomka je tedy nutné jeho kvalitu ohodnotit hodnotící funkcí. Kvalitní potomci jsou zachováni i v další generaci, naopak nekvalitní řešení jsou zapomenuta.

Operátor **mutace** vytváří nové jedince pouze změnami jednotlivých alel chromozomu. Nejedná se tedy o zásadní změnu chromozomu jako takového. Mutace je velmi důležitá při použití spolu s křížením. Při této kombinaci umožňuje algoritmu překonat uváznutí v lokálních extremitách a tím se přiblížit k extremitě globální – nejlepšímu řešení. Pokud je však křížení použito samostatně, nastává velmi velká pravděpodobnost uváznutí algoritmu v lokálním extrému a následné neschopnosti se z této situace vyprostit.

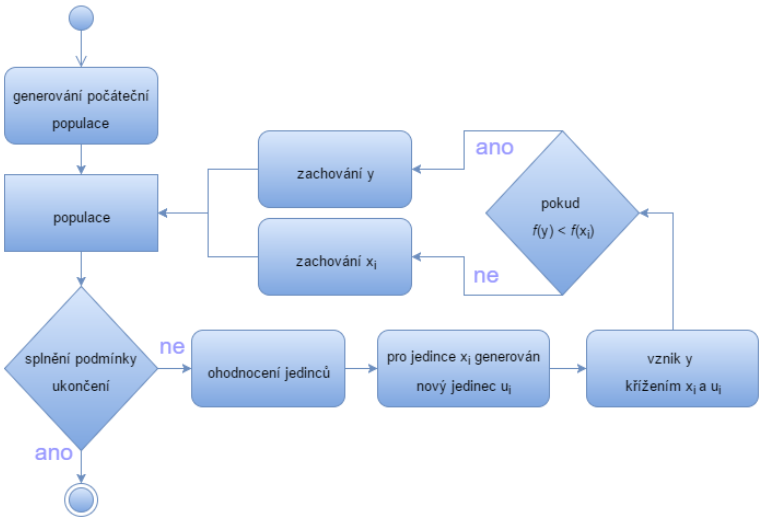
Obecný postup průběhu genetického algoritmu je velmi podobný jako u již zmíněného Náhodného prohledávání. Po inicializaci a načtení vstupních proměnných je generována počáteční populace jedinců. Je zaveden cyklus typu REPEAT s definicí ukončovací podmínky (počet iterací, požadovaná hodnota fitness atd.). Cyklicky jsou prováděny následující činnosti (Zelinka a kol., 2009):

- ohodnocení každého jedince v populaci,
- konverze genotypu na fenotyp,
- konverze ohodnocení na fitness (vhodnost),
- aplikace operátoru selekce,
- aplikace operátoru křížení,
- aplikace operátoru mutace.

Diferenciální evoluce

Diferenciální evoluce je postup využívaný k heuristickému hledání minima multimodálních funkcí. Autory jsou Rainer Storn a Kenneth Price (Storn a Price, 1997). Algoritmus Diferenciální evoluce je značně populární, často aplikován a dále rozvíjen. Experimentální výsledky i zkušenosti z četných aplikací poukazují na to, že velmi často konverguje rychleji než jiné stochastické algoritmy užívané pro globální optimalizaci (Tvrdík, 2004).

V obecné rovině diferenciální evoluce vytváří novou populaci tak, že postupně pro každého jedince x_i ze staré populace vytvoří nového jedince u_i a zkřížením těchto jedinců vznikne pro x_i potenciální konkurent y_i. Do nové populace se z této dvojice zařadí bod s nižší funkční hodnotou (viz obr. 6).



Obr. 6 Obecný průběh algoritmu diferenciální evoluce

(zdroj: vlastní zpracování).

Volná (2012) uvádí dva přístupy generování nového jedince u_i. Prvním z nich je postup označený **RAND**. Ten generuje nového jedince u_i, za pomoci tří odlišných jedinců aktuální populace (viz vzorec 8). Proměnné r_1, r_2 a r_3 představují navzájem různé jedince aktuální populace, kteří se nerovnají vybranému jedinci x_i. $F > 0$ je vstupní parametr udávající váhu diference.

$$u_i = r_1 + F(r_2 - r_3) \tag{8}$$

Druhý přístup je nazván **BEST**. Zde je využito nejlepšího jedince populace x_{best} a čtyř dalších náhodně vybraných jedinců aktuální populace (viz vzorec 9). Proměnné r_1, r_2, r_3 a r_4 představují navzájem různé náhodně vybrané jedince populace, kteří jsou také různí od aktuálně zpracovávaného jedince x_i i nejlepšího jedince populace x_{best}. $F > 0$ je opět váhu diference.

$$u_i = x_{best} + F(r_1 + r_2 - r_3 - r_4) \tag{9}$$

Konkurent y_i je generován pomocí operace křížení. Kříží se aktuálně zpracovávaný jedinec x_i s nově vytvořeným jedincem u_i tak, že kterýkoli prvek x_{ij} může být nahrazen prvkem u_{ij}, a to s pravděpodobností C (viz vzorec 10). Parametr I je náhodně vybraným celým číslem z {1, 2, … j}. $R_j \in (0, 1)$ je náhodně generováno pro každé j. Pravděpodobnost křížení C je vstupním parametrem na intervalu [0, 1] (Tvrdík, 2004). Storn a Price (1997) doporučují nejdříve volit hodnoty F = 0,8 a C = 0,5 a následně přizpůsobit podle zkušeností získaných v průběhu testování algoritmu.

$$y_i = \begin{cases} u_{ij} & když\ R_j \leq C \quad nebo\ j = I \\ x_{ij} & když\ R_j > C \quad a\ j \neq I \end{cases} \tag{10}$$

Konečná selekce je v diferenciální evoluci představována výběrem mezi aktuálně zpracovávaným jedincem x_i a jeho vytvořeným konkurentem y_i. Kvalitnější řešení je zachováno do další generace algoritmu.

4.3 Informační entropie

V oblasti geověd existuje velké množství metod pro restrikci škály dat (transformaci primárních dat do intervalů hodnot). Mezi základní z těchto metod patří např. rovnoměrné dělení (intervaly hodnot jsou voleny o stejné délce), dělení pomocí kvantilů, Jenksova

metoda přirozených zlomů (Jenks, 1967) či manuální nastavení intervalů. Žádná ze zmíněných metod však nebere v úvahu ztrátu informace při restrikci škály dat.

Problematikou objemu informace v přenášené zprávě se podrobně zabývá Shannon (1948), který vymezuje pojem **informační entropie**. Informační entropie je zde definována následovně: „Entropie je střední hodnota míry informace k odstranění neurčitosti, která je dána konečným počtem vzájemně vylučujících se jevů." Entropii systému, který nabývá konečného počtu možných stavů $S \in \{s_1, s_2, \ldots, s_n\}$, kde $n \leq \infty$ s pravděpodobností výskytu stavu $P_{(si)}$, lze vypočítat následovně:

$$H_{(s)} = -\sum_{i=1}^{n} P_{(si)} * log_2 * P_{(si)} \quad (11)$$

Podle Shannon (1948), Pezlar (1998) a dalších nastává maximální entropie tehdy, jsou-li vyrovnané šance všech výskytů souboru, tedy pokud je situace nejvíce nejistá (neurčitá). Situaci, kdy se vyskytuje jev s maximální entropií, lze přirovnat k hodu kostkou. Zde existuje stejná pravděpodobnost pro všech šest možných závěrů hodu.

K odstranění této neurčitosti je potřeba získat maximum informace. To znamená, že minimální entropie systému nastane v okamžiku, kdy se jedná o jev jistý (nastane situace, kdy je pravděpodobnost jedné varianty 100% a všechny ostatní varianty mají pravděpodobnost nulovou).

Tématu se podrobněji věnují anglické práce Batty (2010), Batty a kol. (2014) či Wilson (2010). Z českých autorů je nutno zmínit Půlpán (2006), kde se autor věnuje ztrátě informace v důsledku restrikce měřící škály. Pászto (2009) představil koncept aplikace entropie geodat na příkladu klimatologických dat. Z dílčích výsledků práce byl publikován příspěvek Tuček, Pászto, Voženílek (2009), zabývající se použitím entropie na geografická data. Pro tuto diplomovou práci je stěžejní článek autorů Dvorský, Pászto a Skanderová (2013), ve kterém je řešen problém restrikce měřící škály geodat s použitím informační entropie a genetických algoritmů.

4.4 Tvarové metriky

Parent (2009) definuje metriky jako kvantifikační nástroje tvarů. Obecně se jedná o metody používané zejména v krajinné ekologii (zde byly poprvé užity již v 80. letech 20. století), ale i v obecně-geografických vědních disciplínách. Jak již bylo řečeno, představují nástroje pro kvantitativní hodnocení tvarů ploch reálného světa. K hodnocení je používáno od velmi jednoduchých (obsah, obvod plošky) až po složité výpočty.

Parent (2014) dále rozšiřuje teorii metrik o souvislosti s krajinnou ekologií a problematikou městského prostoru. Mimo jiné zmiňuje, že tvarové metriky jsou použitelné pro detekci ploch vhodných k životu některých druhů. S ohledem na eliminaci vlivu závislosti metrik na rozloze plošky Parent (2014) zavádí tzv. Equal Area Circle (kruh

o stejném obsahu jakého nabývá zkoumaný tvar). Jistě zajímavou aplikací je Parent (2008), zde se autor zabývá hodnocením tzv. urban sprawl (sídelní kaše), tedy problém chaotické výstavby předměstí. Dále lze metriky užít např. pro identifikaci gerrymanderingu (účelné manipulování s hranicemi volebních obvodů).

Z česky psaných zdrojů lze uvést Janoška (2011), který poukazuje na to, že v geovědách jsou charakteristiky tvaru důležitou součástí řešení řady úloh. Hartmannová (2013) ve své bakalářské práci představuje metody hodnocení (sub)urbanizace s použitím tvarových metrik, mimo jiné práce obsahuje přehledný seznam různých druhů metrik. Čepová (2015) aplikovala některé tvarové metriky na oblast hydrologie. Tvarovými metrikami se také podrobně zabývá Pászto (2015).

McGarigal (2015) rozděluje metriky podle aspektů vzoru, které jsou hodnoceny, jsou uvedeny následující kategorie: **Area a edge**, **Shape**, **Core area**, **Contrast**, **Aggregation** a **Diversity**. Čepová (2015) dělí tvarové metriky na **jednoduché**, **pokročilejší** a **metriky založené na výpočtu fraktální dimenze**. Toto dělení bylo použito i pro přehled vybraných metrik v této práci.

4.4.1 Jednoduché metriky

- **Patch Area** - Udává rozlohu plochy v hektarech.
- **Patch Perimeter -** Vyjadřuje obvod plochy.
- **Perimeter-Area Ratio -** Udává poměr mezi obvodem *P* a rozlohou plochy *A* (viz vzorec 12). Metrika je schopna měřit tvarovou komplexnost plošky. S rostoucí velikostí plošky index klesá. Perimeter-Area Ratio index má minimální hodnotu 0 a maximální hodnota není omezena. Se zvyšující se tvarovou složitostí (délkou obvodu) se hodnota indexu zvyšuje.

$$\text{Perimeter} - \text{Area Ratio} = \frac{P}{A} \tag{12}$$

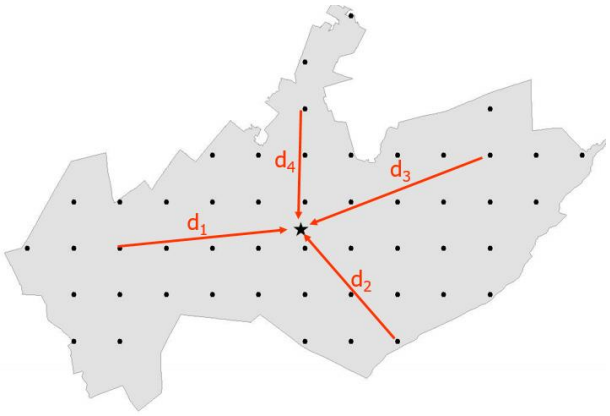
- **Shape index** - Hodnotí složitost plošky vzhledem ke čtverci o stejné rozloze jakou nabývá ploška. Výpočet metriky lze provést viz vzorec 13, kde *P* představuje obvod a *A* rozlohu plošky. Při aplikaci na maximálně kompaktní tvar (v této variantě vzorce čtverec) se hodnoty rovnají jedné, čím se složitost tvaru zvyšuje, tím se zvyšuje i hodnota Shape indexu.

$$\text{Shape index} = \frac{.25\,P}{\sqrt{A}} \tag{13}$$

4.4.2 Pokročilejší metriky

- **Proximity Index** - Vyjadřuje kompaktnost území. Metrika je reprezentována průměrnou vzdáleností vnitřních bodů plošky od jejího těžiště. Všem bodům je přisouzena stejná váha (viz vzorec 14 a obr. 7). Proměnná d udává vzdálenost mezi vnitřním bodem a centrem plošky, n je počet bodů.

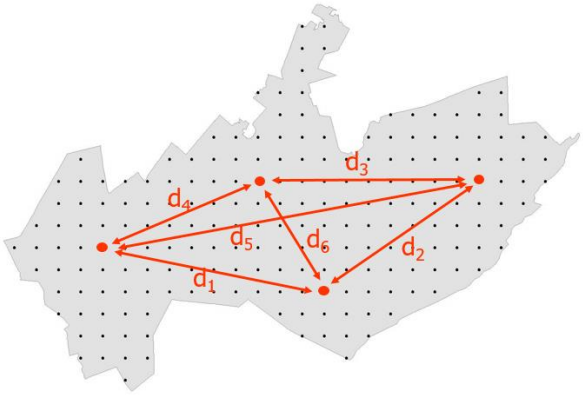
$$\text{Proximity Index} = \frac{d_1+d_2+\cdots+d_n}{n} \tag{14}$$



Obr. 7 Znázornění metriky Proximity Index
(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm).

- **Cohesion Index** - Hodnotí průměrné vzdálenosti mezi vnitřními body plošky. Nad ploškou je vytvořena síť bodů. Vzdálenosti mezi vytvořenými body vstupují do výpočtu indexu (viz vzorec 15 a obr. 8) jako proměnná d, # reprezentuje počet párů bodů.

$$\text{Cohesion Index} = \frac{d_1+d_2+\cdots+d_n}{\#} \tag{15}$$

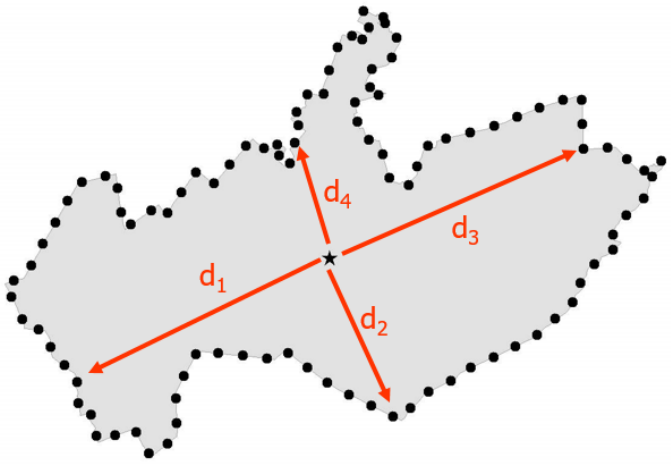


Obr. 8 Znázornění metriky Cohesion Index

(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm).

- **Dispersion Index** – Metrika je dána poměrem sumy vzdáleností od bodů na obvodu plošky k těžišti plošky a počtu těchto bodů. Užitím metriky lze zjistit, zda je jev rovnoměrně rozptýlen do všech směrů. Výpočet (viz vzorec 16 a obr. 9) obsahuje proměnnou d (vzdálenost mezi daným bodem a těžištěm) a n (počet bodů).

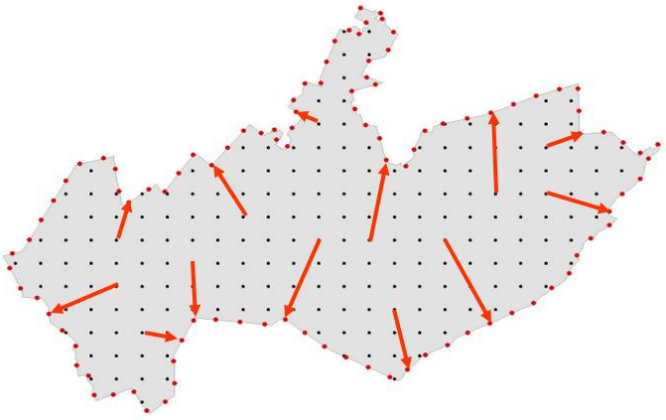
$$\text{Dispersion Index} = \frac{d_1+d_2+\cdots+d_n}{n} \tag{16}$$



Obr. 9 Znázornění metriky Dispersion Index

(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm).

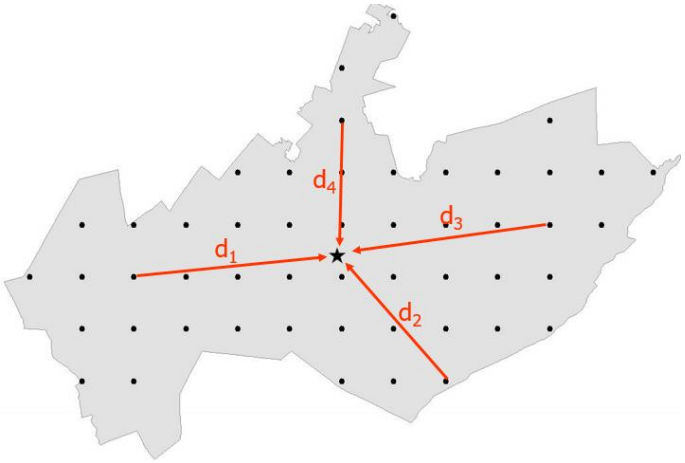
- **Depth Index** - Je představován průměrnou vzdáleností vnitřních bodů vzhledem k nejbližším bodům ležícím na obvodu (viz obr. 10). Hartmanová (2013) uvádí, že s nárůstem délky průměrné vzdálenosti roste i izolace plošky od okolních vlivů. Hlavní využití metriky je v krajinné ekologii (studium vnějších vlivů na výskyt živočišných a rostlinných druhů).



Obr. 10 Znázornění metriky Depth Index

(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm).

- **Spin Index** – Metrika se podobá již zmíněné Proximity Index, na rozdíl od něj je ale vyšší váha nastavena bodům ležícím dále od centra plochy (díky umocnění vzdáleností na druhou). Do výpočtu (viz obr. 11 a vzorec 17) se doplní vzdálenost *d* mezi vnitřními body a centroidem a počet bodů #. Parent, Civco a Angel (2009) uvádí, že tato metrika je vhodná také pro hodnocení tzv. Urban Sprawl.



Obr. 11 Znázornění metriky Spin Index

(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm).

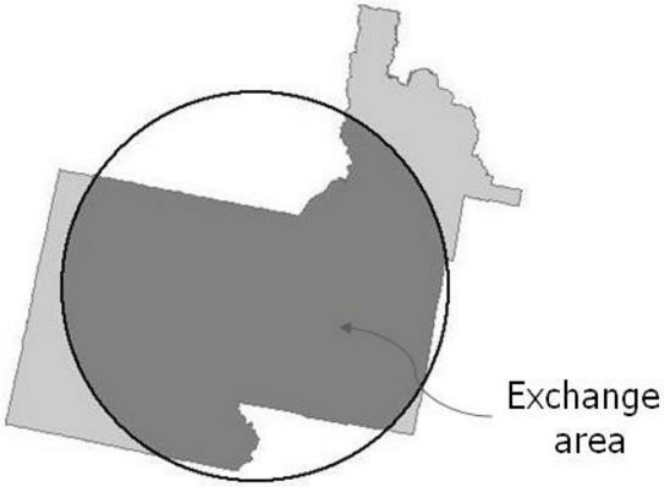
$$\text{Spin Index} = \frac{d_1^2+d_2^2+\cdots+d_n^2}{\#} \tag{17}$$

- **Range Index** – Metrika podává informaci o délce průměru kružnice opsané k dané plošce. Reprezentuje také vzdálenost mezi dvěma nejvzdálenějšími body na plošce (viz obr. 12).

Obr. 12 Znázornění metriky Range Index

(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/graphics/Range.htm).

- **Exchange Index** - Vyjadřuje rozlohu plošky pomocí tzv. Equal Area Circle (kruh o stejné rozloze jaké nabývá ploška). Střed kruhu je umístěn do centroidu plošky (viz obr. 13).

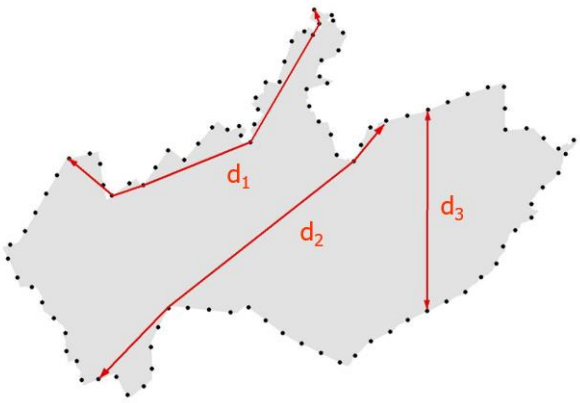


Obr. 13 Znázornění metriky Exchange Index

(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/graphics/Exchange.htm).

- **Traversal Index** – Metrika je dána průměrem nejkratších spojení mezi dvěma náhodně vybranými body ležícími na obvodu plošky. Linie spojení bodů musí vést vnitřkem plošky (viz obr. 14). Výpočet indexu probíhá dle vztahu (viz vzorec 18), kde *d* je nejkratší vzdálenost mezi vybranými body ležícími na obvodu plošky a # představuje počet párů vybraných bodů.

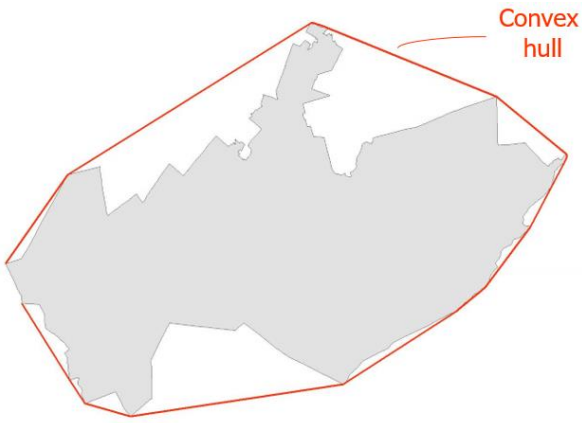
$$\text{Traversal Index} = \frac{d_1+d_2+\cdots+d_n}{\#} \tag{18}$$



Obr. 14 Znázornění metriky Traversal Index

(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm).

- **Detour Index** - Je roven délce obvodu tzv. convex hull (konvexní obálka), která představuje polygon s co nejkratším obvodem, který pojímá celou plošku (viz obr. 15). Pászto (2015) uvádí, že čím více se tvar hodnocené plošky odlišuje od kruhu (zejména pokud ploška obsahuje několik výběžků), tím hodnota Detour Indexu roste. Normalizovaná forma tohoto indexu (viz vzorec 19) představuje podíl obvodu kruhu o stejném obvodu, jaký nabývá ploška (proměnná p_{EAC}) a obvodu konvexní obálky (proměnná $p_{\text{Convex Hull}}$). Zde ploška tvaru kruhu nabývá hodnoty 1, naopak podlouhlé plošky a plošky složité nabývají hodnot blížících se nule.

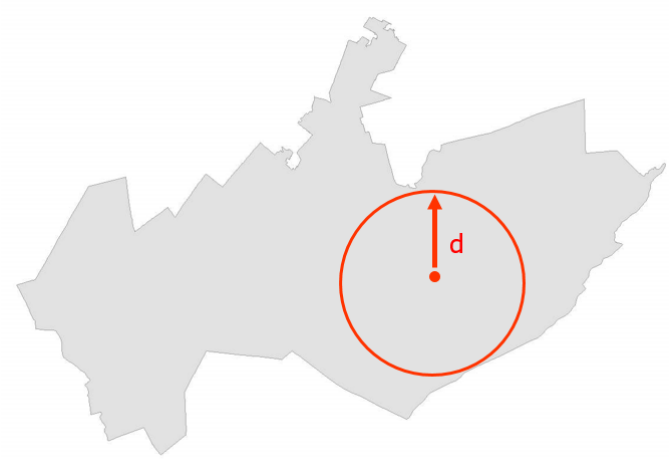


Obr. 15 Znázornění metriky Detour Index

(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm).

$$\text{Normalized Detour Index} = \frac{p_{EAC}}{p_{Convex\ Hull}} \tag{19}$$

- **Girth Index** – Metrika udává poloměr největší kružnice vepsané dané plošky (viz obr. 16). Touto metrikou lze hodnotit, zda ploška obsahuje dostatečně velkou souvislou plochu.



Obr. 16 Znázornění metriky Girth Index
(zdroj: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm).

4.4.3 Fraktální dimenze

- **Fraktální dimenze** dle McGarigal (2015) lze vypočítat (viz vzorec 20), pomocí hodnot P (představuje obvod plošky) a A (rozloha plošky). Nejedná o výpočet fraktální dimenze v pravém smyslu slova, jelikož není koncipován na procházení napříč měřítky, slouží spíše k popisu složitosti tvaru.

$$FD = \frac{2\log(P)}{\log(A)} \tag{20}$$

- **Sinuosity** (vlnitost) - Metrika užívaná pro liniové prvky charakterizuje míru zvlnění. Hodnotu vlnitosti lze vypočítat (viz vzorec 21) s použitím proměnných L_t (představuje kumulativní délku všech úseček) a L_{sf} (vzdálenost mezi počátečním a koncovým bodem linie).

$$\text{Sinuosity} = \frac{L_t}{L_{sf}} \tag{21}$$

5 NÁVRH CENOVÉ FUNKCE

Podle zadání diplomové práce byla vytvořena cenová funkce. Tato funkce byla sestavena s důrazem na možnost výpočtu tvarových metrik a minimalizaci ztráty informace ve vstupních datech. Navržená cenová funkce byla konstruována s ohledem na možnost uživatelsky upravovat váhy vypočtených metrik a entropie. Pro implementaci do cenové funkce byla mimo výpočet entropie vybrána čtveřice tvarových metrik, a to:

- **Fractal Dimension Index** - hodnocení celkové složitosti plošky,
- **Shape Index** - hodnocení složitosti tvaru plošky vzhledem ke čtverci o stejné rozloze,
- **Perimeter-Area Ratio -** hodnocení poměru obvodu a rozlohy plošky,
- **Normalized Detour Index -** hodnocení kompaktnosti tvaru.

5.1 Transformace výstupních hodnot kritérií

Vzhledem k rozdílnému pojetí optimálních hodnot použitých kritérií a také jejich částečné standardizaci, bylo nutné návratové hodnoty kritérií transformovat. Primárně se jednalo o metriky **Fractal Dimension Index**, **Shape Index** a **Normalized Detour Index**, dále také o návratovou hodnotu kritéria **entropie**.

Metrika **Perimeter-Area Ratio**, která produkovala výstupní hodnoty na intervalu $(0, \infty)$ byla ponechána beze změny a při testování upravována pouze pomocí váhy. Tato metrika byla notně ovlivňována velikostí hodnocené plošky, proto byla vnímána spíše jako ukazatel velikosti zkoumaných plošek. Se zmenšujícím se indexem narůstala velikost plošek.

Výstupní hodnota metriky **Fractal Dimension Index** nabývala pro 2D geometrii hodnot od $\langle 1, 2 \rangle$. S narůstající hodnotou indexu byla zvyšována tvarová složitost plošek. Pro normalizaci byly výstupní hodnoty metriky upraveny (viz vzorec 22), kde hodnota proměnné *FD* představovala výstupní hodnotu metriky.

$$X_{FRAC} = FD - 1 \tag{22}$$

Shape Index byl představován metrikou s návratovou hodnotou v intervalu $\langle 1, \infty)$. Tyto hodnoty bylo nutné převést na interval $(0, \infty)$ podle vzorce 23, kde návratová hodnota metriky Shape index byla představována proměnnou *SHAPE*. S narůstající hodnotou indexu byla zvyšována tvarová složitost plošek.

$$X_{SHAPE} = SHAPE - 1 \tag{23}$$

Normalized Detour Index reprezentoval metriku s návratovými hodnotami $(0,1)$. Zde se na rozdíl od předchozích metrik se zvyšujícími se hodnotami tvar plošky stával více kompaktní. Bylo tedy nutné výstupní hodnoty invertovat tak, aby stejně jako u předchozích indexů tvarová složitost se zvyšující se hodnotou rostla (viz vzorec 24).

$$X_{NDETOUR} = 1 - NDETOUR \tag{24}$$

Návratová hodnota výpočtu střední hodnoty entropie aktuálního jedince byla transformována podle postupu Dvorský, Pászto a Skanderová (2013). Získání výstupní hodnoty tohoto kritéria bylo provedeno odečtením střední hodnoty entropie dat (H_0) a střední hodnoty entropie jedince (H_j) (viz vzorec 25).

$$X_{ENTROPY} = H_0 - H_j \tag{25}$$

5.2 Konstrukce cenové funkce

Cenová funkce byla navržena primárně s ohledem na již zmíněné fakty. Tedy cílem diplomové práce bylo aplikovat vytvořenou cenovou funkci na různé typy evolučních algoritmů s užitím různých charakteristik zkoumaných geodat (tvarové metriky a entropie). Dále byla plánována možnost užití vah při testování kombinací metrik a entropie. Z tohoto důvodu bylo nutné návratové hodnoty jednotlivých metrik přizpůsobit potřebám výpočtů (viz předchozí kapitola 5.1). Velmi důležitým krokem bylo také aplikování vzniklé cenové funkce do výkonu procesů nástroje ESMEA. Tímto tématem se práce také zabývá (viz kapitola 6).

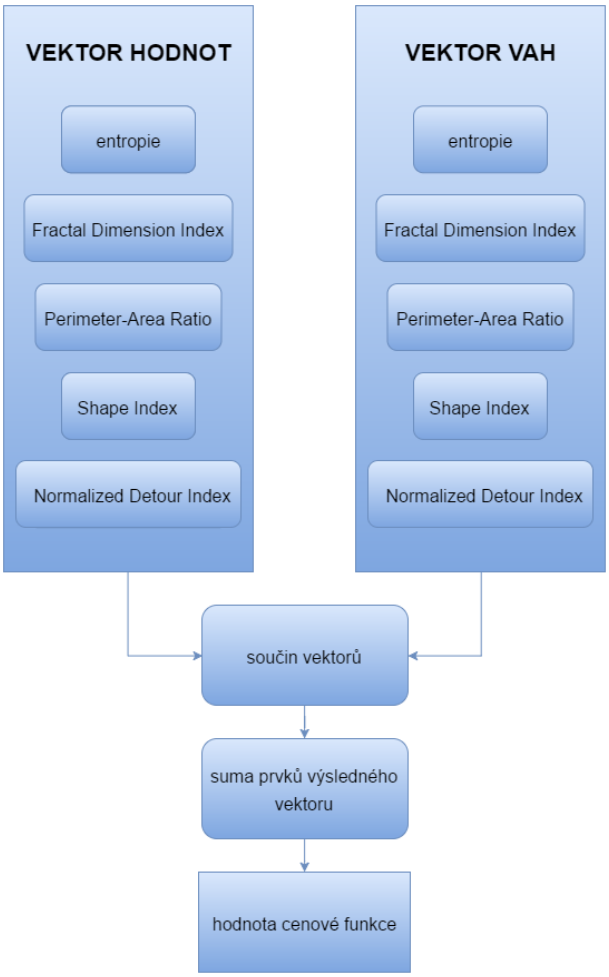
Konečný návrh výpočtu cenové funkce $f_{cost}(x)$ byl proveden pomocí **skalárního součinu** dvojice vektorů s pěti prvky (viz vzorec 26). První z těchto vektorů představoval **vektor návratových hodnot** metrik a entropie (v). Druhý vektor (w) byl naplněn **vahami jednotlivých kritérií**.

$$f_{cost}(x) = \sum_{i=1}^{n} v_i \cdot w_i \tag{26}$$

Prvky vektoru hodnot musely být dále zpracovány vzhledem k následnému násobení vahami. Popis výpočtu vektoru hodnot (viz vzorec 27) obsahoval proměnné *FD* (hodnota Fractal Dimension Index), *PARA* (hodnota Perimeter-Area Ratio), *SHAPE* (hodnota Shape Index), *NDETOUR* (hodnota Normalized Detour Index) a H_0 (střední hodnota entropie vstupních dat) spolu s H_j (střední hodnota entropie jedince).

$$v = \left[FD - 1, PARA, SHAPE - 1, 1 - NDETOUR, H_0 - H_j\right] \quad (27)$$

Pro dosažení návratové hodnoty cenové funkce $f_{cost}(x)$ (hodnoty určující kvalitu jedince), bylo nutné každý prvek vektoru hodnot (v_i) vynásobit korespondujícím prvkem vektoru vah (w_i). Po vynásobení byly **sečteny** jednotlivé prvky vektoru vzniklého součinem vektoru vah a vektoru návratových hodnot (viz obr. 17). Úkolem jednotlivých evolučních algoritmů bylo tuto hodnotu **minimalizovat**.



Obr. 17 Znázornění výpočtu cenové funkce

(zdroj: vlastní zpracování).

6 NÁSTROJ ESMEA

Pro zjednodušení procesu testování jednotlivých evolučních algoritmů a následného aplikování vah kritérií byl vytvořen nástroj ESMEA (Entropy and Shape Metrics Evolutionary Algorithms). Do tohoto nástroje bylo implementováno celkem pět variant evolučních algoritmů. Tři z těchto pěti představovaly celistvá řešení, dvojice použitých algoritmů reprezentovala pouze iterativní provádění operací mutace a křížení.

Nástroj byl navržen takovým způsobem, aby obsahoval zadaná kritéria hodnocení kvality jedinců. Jednalo se o metriky **Fractal Dimension Index, Shape Index, Perimeter-Area Ratio** a **Normalized Detour Index**.

Dále ESMEA podporuje výpočet restrikce škály dat podle návrhu Dvorský, Pászto a Skanderová (2013). Cílem bylo minimalizovat ztrátu informace při dělení geodat do intervalů, a to výpočtem rozdílu středních hodnot **entropie** vstupního souboru dat a jedince. Zde však byly provedeny jisté obměny v podobě již prezentované cenové funkce (viz kapitola 5.2) a reprezentaci jedince.

Popis volby programových prostředků, návrhu, průběhu a konečné implementace ESMEA je uveden v následujících podkapitolách. Podrobně je také vysvětleno, jakým způsobem autor navrhl a sestavil jednotlivé verze evolučních algoritmů.

6.1 Výběr vývojových nástrojů a vstupních dat

Prvním z mnoha kroků fáze tvorby diplomové práce bylo vybrat vhodné programové prostředí pro tvorbu a následné spouštění nástroje. Vzhledem k potřebě implementace prostorových dat do výpočtů některých metrik a použití těchto nástrojů pro zpracování geodat, byl výběr značně zúžen. Nejdříve se autor pokoušel o konstrukci nástroje v prostředí jazyka C#. Nicméně tyto prvotní pokusy nebyly dovedeny do zdárného konce, vzhledem k poměrně složité implementaci nástrojů pro práci s prostorovými daty. Po těchto zkušenostech byl vývoj nástroje ESMEA soustředěn primárně na kombinaci skriptovacího jazyka Python (ve verzi 2.7) a nástrojů pro práci s prostorovými daty, které byly implementovány pomocí knihovny ArcPy.

Vývoj nástroje byl realizován v prostředí programu Python IDLE, stejně tak v tomto programu probíhalo následné spouštění již funkčního nástroje. Ten byl konstruován tak, aby umožňoval i spuštění v prostředí příkazové řádky pomocí dvojkliku na soubor se skriptem ESMEA.py.

Výběr prostorových dat pro realizaci testování funkčnosti nástroje i následného provádění experimentů řešených touto diplomovou prací byl primárně ovlivněn již provedeným výzkumem Dvorský, Pászto a Skanderová (2013). Tato data byla vybrána primárně z důvodu srovnání výsledů, a to jak kontroly správné činnosti ESMEA, tak i rozšíření uvedeného postupu o aplikování tvarových metrik. Vybraná data představovala ESRI grid průměrné roční teploty vzduchu v České republice mezi lety 1961

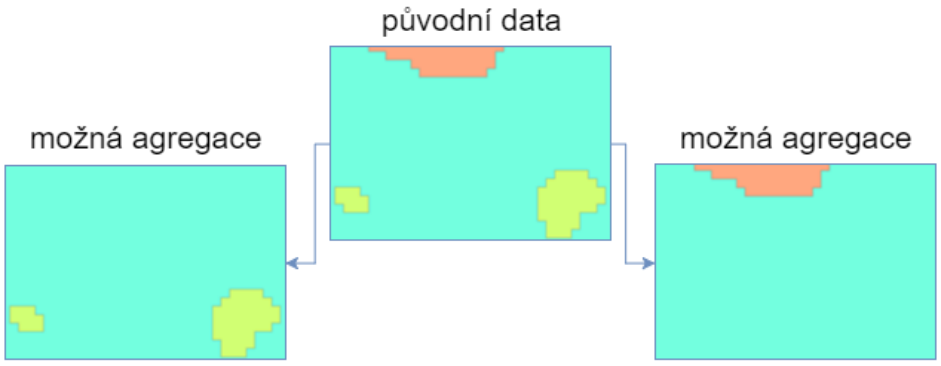
a 2001. Buňka gridu reprezentovala území o velikosti 100 x 100 m. Celý rastr čítal 4 865 sloupců a 2 780 řádků. Grid byl složen z buněk obsahujících hodnoty od 1 do 10 °C, představujících průměr teplot mezi uvedenými lety. Počty buněk pro jednotlivé stupně Celsia byly uvedeny v podkapitole Použitá data (viz tabulka 1). Pro tato data byla stanovena střední hodnota entropie H_0 = 2,135 bitů na jednu buňku gridu (viz vzorec 11).

6.2 Formální popis úlohy

Podle Dvorský, Pászto a Skanderová (2013) lze všechny vstupní hodnoty rastru chápat jako konečnou množinu $V = \{v_1, v_2, \ldots, v_n\}$, kde $v_i \in R$. Dále je předpokládáno, že $v_i < v_{i+1}$ pro $i = 1, \ldots, n - 1$. Restrikcí měřící škály na množině vstupních hodnot V je chápáno rozdělení n vstupních hodnot do N nepřekrývajících se intervalů $V_1, \ldots, V_n$. Každé rozdělení škály dat J_i (možné řešení) je reprezentováno posloupností těchto nepřekrývajících se intervalů $J_i = (V_1, \ldots, V_n)$.

Pro každou vstupní hodnotu v_i byla definována její četnost f_i (počet pixelů rastru nabývajících hodnotu v_i). Pomocí zjištěných informací o počtu jedinečných vstupních hodnot v_i a četností výskytu těchto hodnot f_i ve vstupním rastru byla vypočítána střední hodnota entropie H_0 (viz vzorec 11).

Využitím evolučních algoritmů byla generována přípustná řešení (restrikce měřící škály vstupních geodat). Při změně této restrikce byly ovlivňovány geometrické reprezentace vstupních dat, respektive v závislosti na aktuální restrikci měřící škály byly vytvářeny agregací různé reprezentace plošek (viz obr. 18). Těmito změnami agregace plošek byla ovlivňována jak střední hodnota entropie daných řešení, tak i metriky jednotlivých plošek, respektive průměrné metriky celých řešení.



Obr. 18 Znázornění změny agregace geometrie při změně restrikce měřící škály geodat (zdroj: vlastní zpracování).

Pro každé navrhované řešení byla nejdříve vypočítána střední hodnota entropie (viz vzorec 11), kde tato hodnota byla založena na entropii jednotlivých intervalů V_n

a následně entropie těchto intervalů byla založena na pravděpodobnosti výskytů všech hodnot v_i daného intervalu V_n.

Všem ploškám vytvořeným v závislosti na aktuální restrikci škály dat byly spočteny jednotlivé metriky. Následně, odděleně pro každou metriku byly tyto hodnoty metrik plošek sečteny a výsledek podělen počtem plošek. Tímto způsobem byl získán aritmetický průměr metrik všech plošek daného řešení J_i.

Zjištěné informace o střední hodnotě entropie vstupního rastru a aktuálního řešení spolu s průměrnými hodnotami metrik daného řešení byly dále použity jako vstupy do navržené cenové funkce. Dané řešení bylo ohodnoceno. Evoluční algoritmy byly využity ke generování takových řešení, aby nabývaly co nejmenších hodnot získaného ohodnocení, tedy k minimalizaci tohoto ohodnocení.

6.3 Požadavky na vstupní data a programy

Nástroj ESMEA byl konstruován pro podporu vstupu dat rozličné tématiky, byly však stanoveny určité podmínky, které musely být naplněny pro zajištění správného fungování. Tyto podmínky byly sestaveny tak, aby korespondovaly s výše uvedeným formálním popisem úlohy. ESMEA byl sestaven pro analýzy nad dvojrozměrnými daty, tedy první podmínka byla stanovena tak, že vstupní data musí být 2D. Dalších několik podmínek bylo stanoveno v závislosti na výpočet střední hodnoty entropie dat podle konceptu Dvorský, Pászto a Skanderová (2013).

Jednalo se primárně o požadavek na formát dat (ESRI grid), dále celočíselnost hodnot buněk a posloupnost jednotlivých hodnot gridu v oboru hodnot reálných čísel (tedy množina vstupních hodnot $V = \{v_1, v_2, \ldots, v_i\}$, kde $v_i \in R$, a kde $v_i = v_{i+1} - 1$ pro $i = 1, \ldots, n - 1$).

Mezi požadavky z oblasti programového vybavení je zařazeno programové prostředí ArcGIS verze 10.3.1 a vyšší. Dalším požadavkem pro funkčnost nástroje ESMEA je skriptovací jazyk Python ve verzi 2.7, respektive jeho dostupnost v systému. Ve většině případů je instalace prostředí jazyka Python do systému součástí instalace ArcGIS.

6.4 Návrh reprezentace jedince a evolučního algoritmu

Z hlediska konstrukce nástroje ESMEA a následné aplikace cenové funkce do jednotlivých evolučních algoritmů, bylo nutné vytvořit návrh reprezentace jedince. Kvůli složitosti řešeného problému nebylo vhodné využít binárního kódování. Dvorský, Pászto a Skanderová (2013) uvádí reprezentaci jedince pomocí posloupnosti nepřekrývajících se intervalů $V_1, \ldots, V_n$, kde každý z těchto intervalů obsahuje dvojici vektorů omezujících spodní, respektive horní hranici intervalu.

Autor práce se přiklonil k poněkud obecnějšímu řešení. Jedinec byl navržen jako instance Python třídy (viz obr. 18). Tato třída obsahovala proměnnou *fit* (reprezentující ohodnocení fitness jedince) a slovník *int* (reprezentující restrikci škály vstupních dat). Slovník *int* byl dále rozdělen na N dvojic klíč – hodnota, kde hodnota N byla představována

počtem intervalů. Každý pár klíč – hodnota obsahoval přístupový index intervalu (klíč) a samotný interval hodnot reprezentovaný seznamem posloupnosti jedinečných hodnot nacházejících se v daném intervalu (hodnota) (viz obr. 19).

```
class jedinec(object):
    def __init__(self):
        self.fit = 0        # promenna pro hodnotu fitness
        self.int = {}       # slovnik pro hodnoty jednotlivych intervalu
```

Obr. 18 Reprezentace třídy jedince

(zdroj: vlastní zpracování).

```
{0: [1, 2, 3, 4, 5], 1: [6], 2: [7], 3: [8], 4: [9, 10]}
```

Obr. 19 Způsob uložení restrikce škály dat (možného řešení)

(zdroj: vlastní zpracování).

Reprezentace celého aktuálně prováděného evolučního algoritmu byla realizována opět pomocí instance Python třídy. Tato třída byla pojmenována *g* a byla dále složena z vnořených prvků (viz obr. 20). Instance této třídy byla v průběhu výpočtů nástroje ESMEA volána vždy právě jednou.

```
class g(object):
    def __init__(self):
        self.h0 = 0         # hodnota H0
        self.rCnt = 0       # pocet pixelu vstupniho rastru
        self.rMin = 0       # minimalni hodnota vstupniho rastru
        self.rMax = 0       # maximalni hodnota vstupniho rastru
        self.val = []       # seznam pro hodnoty vstupniho rastru a jejich pocty
        self.pop = []       # seznam pro populaci (sklada se z jednotlivych trid jedincu)
        self.bin = []       # seznam uloziste jiz spoctenych jedincu
```

Obr. 20 Reprezentace aktuálního evolučního algoritmu

(zdroj: vlastní zpracování).

6.5 Návrh operátorů evolučních algoritmů

Před samotnou konstrukcí nástroje ESMEA bylo velmi důležité navrhnout, alespoň rámcově princip fungování jednotlivých evolučních operací. Jednalo se o operace selekce, křížení a mutace.

6.5.1 Selekce

Operace **selekce** byla pro všechny navržené algoritmy aplikována stejným způsobem. Důraz byl kladen především na jednoduchost a přitom výpočetní sílu navrhované varianty tohoto evolučního operátoru.

Základním principem této operace je jednoduché pravidlo. Pro každého jedince populace je generován konkurent. Pokud je konkurent ohodnocen lépe, nahrazuje tento konkurent původního jedince. Tedy, pro každého jedince populace J_i je generován konkurent K_i a následně je lépe ohodnocený jedinec zachován.

6.5.2 Křížení

Návrh operátoru **křížení** je pro nástroj ESMEA realizován následovně. Pro jedince J_i aktuální populace je vybrán jiný jedinec aktuální populace $J_n \neq J_i$. První interval restrikce je získán od náhodně vybraného jedince J_n, zbylé intervaly jsou ponechány z jedince aktuálního.

Pokud je při tomto postupu vytvořen defektní jedinec (maximální hodnota prvního intervalu je vyšší nebo rovna minimální hodnotě obsažené v druhém intervalu), tak je první interval ponechán a zbytek restrikce náhodně generován. Pokud je naopak křížením získán jedinec, obsahující maximální hodnotu prvního intervalu menší než minimální hodnotu druhého intervalu, tak jsou do druhého intervalu zařazeny všechny hodnoty větší než maximální hodnota prvního intervalu a zároveň menší nebo rovny maximální hodnotě druhého intervalu.

6.5.3 Mutace

Operátor **mutace** je realizován náhodným výběrem intervalu V_i restrikce jedince, a to takového že $i < N$. Kde N reprezentuje počet intervalů restrikce. Pozice zvoleného intervalu je tedy maximálně předposlední v pořadí všech intervalů restrikce. Zároveň tento interval musí obsahovat alespoň dvě hodnoty. Samotná mutace je provedena vyjmutím maximální hodnoty z vybraného intervalu V_i a následným vložením této hodnoty do sousedního intervalu V_{i+1}.

6.6 Popis průběhu výpočtů nástroje ESMEA

Průběh výkonu nástroje ESMEA lze rozdělit do dvou celků. První z těchto částí představují operace vykonávané pouze jednou. Jedná se primárně o operace spjaté s načtením vstupních hodnot a následným předzpracováním dat. Druhý celek představují operace, které jsou vykonávány iterativně. Do tohoto bloku jsou primárně zařazeny výpočty jednotlivých evolučních algoritmů.

6.6.1 Operace prováděné pouze jednou

Mimo hlavní tematiku diplomové práce, byl primární problém představován rychlostí výpočtů nástroje ESMEA. Snahou autora bylo tedy především to, aby byl co největší objem výpočetně náročných operací provádět právě v tomto bloku. Zde totiž byly vykonány pouze jednou. Dále bylo nutné, aby nástroj umožňoval zadání parametrů sloužících pro nastavení výpočtů evolučních algoritmů, a to pouze validních. Tato problematika byla řešena právě v bloku operací prováděných pouze jednou.

Nejdříve bylo nutné získat parametry spjaté s chodem evolučních algoritmů. Mezi požadované parametry byly zařazeny: **typ evolučního algoritmu** (Náhodné prohledávání, Mutace, Křížení, Obecný genetický algoritmus a Diferenciální evoluce), dále požadovaný **počet intervalů**, **počet jedinců populace** a **počet generací**. Pokud byl zvolen typ Obecný genetický algoritmus, byly požadovány doplňující informace o pravděpodobnosti náhodného generování jedinců, křížení a mutace. Jestliže byl zvolen typ Diferenciální evoluce, uživatel byl vyzván k zadání parametrů *CR* (pravděpodobnost křížení) a *F* (váha diference).

Po načtení parametrů evolučního algoritmu bylo nutné vložit váhy jednotlivých metrik a entropie. **Součet všech pěti zadaných vah musel být roven hodnotě 1**. To znamenalo, že pro rovnoměrné rozložení vah byla hodnota pro všechna kritéria rovna 0,2. Načtením vah prvků cenové funkce byla ukončena činnost zadávání uživatelských vstupů.

Pro každý uživatelský vstup byl připojen popis a přednastavena výchozí hodnota (uvedená v závorce), které bylo možné využít pomocí stisku klávesy „Enter“ bez zadání vstupu. Uživatelské vstupy, které byly omezeny pouze na hodnoty s desetinnou čárkou, byly koncipovány tak, aby uživatel mohl zadat jak desetinnou čárku, tak i tečku (která je implicitně používána v jazyce Python).

V následujícím kroku byly importovány knihovny **ArcPy**, **NumPy**, **Random**, **Math**, **Copy**, **Time** a **Collections**. Tento proces byl záměrně přemístěn až do tohoto bodu výkonu nástroje ESMEA, a to z důvodu posloupnosti procesů spuštění, zadání parametrů a výpočet.

Proces byl dále směřován k zavedení pracovní geodatabáze *ESMEA.gdb* a nastavení práva přepisování výsledků. V následujících krocích byla definována cesta ke vstupnímu ESRI gridu (pevně nastavena) a cesty k pomocným vrstvám užívaným při dalších výpočtech. Mimo vstupního rastru byla všechna pomocná data (třídy prvků a tabulky) umístěny do tzv. „in_memory workspace“, tedy pracovního prostoru umístěného v paměti. Tato možnost byla zvolena z důvodu rapidního zvýšení výkonu.

V návaznosti na předchozí operace byl proveden přepočet tabulky hodnot gridu (nástroj **Build Raster Attribute Table**), grid převeden do polygonové formy s názvem *rasToPoly* (nástroj **Raster to Polygon**) a zpracována tabulka sousedství vytvořené polygonové vrstvy (nástroj **Polygon Neighbors**). Ta byla pojmenována *tabNeighbors*.

Tento postup nebyl zvolen náhodně. Primárním problémem, který musel autor v průběhu návrhu a konstrukce nástroje ESMEA vyřešit bylo vymyslet, jakým způsobem by bylo možné velmi rychle (v řádech desetin sekund, v ideálním případě méně) zjistit, jaké jednotlivé plošky v jednotlivých intervalech byly vytvořeny při změně restrikce škály dat.

Jako součást programového prostředí ArcGIS 10.3.1 je dodáván i nástroj **Dissolve**. Použitím tohoto nástroje lze podle atributových informací geodat tato data agregovat

i po stránce geometrie. Pomocí nástroje Dissolve je možné vytvořit z primárních (původních) dat novou vrstvu, v níž jsou obsaženy geometrie (plošky) vytvořené s ohledem na zvolené intervaly hodnot (viz obr. 18). Nástroje Dissolve bylo využito při prvních pokusech o sestavení nástroje ESMEA. Bohužel výkon operace byl oproti potřebám implementace značně nedostatečný (jedno provedení celé operace bylo v současných výpočetních podmínkách vykonáno v řádu sekund). Z tohoto důvodu bylo nutné navrhnout řešení, které by bylo schopno vykonat stejný proces s vyšší rychlostí. K vyřešení tohoto problému bylo použito již zmíněného nástroje Polygon Neighbors (užitého v části nástroje, která byla prováděna pouze jednou) a následné aplikace programového kódu v jazyce Python (který byl aplikován při výpočtu každého jedince – uveden viz kapitola 6.6.2 a blíže popsán viz příloha 4).

Vzniklá tabulka sousedství nazvaná *tabNeighbors* (viz obr. 21), obsahovala informace o identifikátorech sousedících plošek (*src_Id* a *nbr_Id*) a délce hranice těchto plošek (atribut *LENGTH*).

OBJECTID *	src_Id	nbr_Id	LENGTH
1	1	9	200
2	1	12	0
3	2	3	0
4	2	577	200
5	3	2	0
6	3	577	200
7	4	12	8200
8	5	577	300

Obr. 21 Tabulka sousedství plošek

(zdroj: vlastní zpracování).

Obsažené informace by však nebyly dostatečné pro další průběh výpočtů, proto bylo využito nástroje **Join Field** a díky identifikátoru src_Id bylo realizováno doplnění údajů z polygonové vrstvy *rasToPoly* (jednalo se o údaje o obvodu, rozloze a kódu pixelů původního gridu nacházející se na pozici plošky). Dále bylo nutné znehodnotit ta sousedství uvedená v tabulce, která nabývala v poli LENGTH hodnoty 0. Tento krok byl realizován pomocí nástroje **Update Cursor** (ve výkonnější verzi arcpy.da.UpdateCursor).

Po znehodnocení těchto záznamů byla provedena transformace prostorových dat vrstvy *rasToPoly* a tabulky *tabNeighbors* do pomocných seznamů užívaných v části nástroje ESMEA prováděné iterativně. Tato transformace byla realizována pomocí nástroje **Search Cursor** (ve verzi arcpy.da.SearchCursor v kombinaci s generátorovou notací seznamů pro zvýšení výkonu).

Posledním krokem celého procesu předzpracování dat bylo přidání nového pole s názvem *hull_code* do polygonové vrstvy *rasToPoly*, pro budoucí užití v iterativní části

nástroje ESMEA. Proces přidání nového pole byl realizován prostřednictvím nástroje **Add Field**.

Následovalo definování tříd *g* (reprezentace sktruktury evolučního algoritmu) a *jedinec* (reprezentace struktury jedince) (viz obr. 18 a 20). V návaznosti na předchozí krok byla volána instance třídy *g* pro zavedení sktruktury evolučního algoritmu.

Pomocí nástroje **Search Cursor** (ve verzi arcpy.da.SearchCursor) byly získány z tabulky hodnot gridu jedinečné vstupní hodnoty a korespondující počty těchto hodnot. Zjištěné údaje v podobě seznamu byly seřazeny od minimální jedinečné hodnoty po maximální, zjištěn celkový počet pixelů gridu a minimální respektive maximální jedinečná hodnota. Důležitým krokem byl také výpočet střední hodnoty entropie vstupního gridu H_0 za pomoci již zjištěných údajů a vzorce pro výpočet střední hodnoty entropie (viz vzorec 11). Zmíněné údaje byly uloženy do proměnných a seznamů struktury instance třídy *g* (viz obr. 20).

Zakončení celého sledu akcí vykonávaných v nástroji ESMEA pouze jednou bylo realizováno pomocí rozhodovacího aparátu funkce IF. Podle zvoleného typu algoritmu byla volána řídící funkce daného algoritmu a prostřednictvím argumentů předána této funkci potřebná nastavení.

6.6.2 Operace vykonávané iterativně

Při výkonu volání jednotlivých typů evolučních algoritmů byly operace prováděny opakovaně. Hlavním cílem bylo tyto operace co nejvíce zjednodušit a provést co nejrychleji, jelikož každá generace algoritmu představovala jeden výkon těchto operací.

První z řady funkcí, které patřily do této skupiny, byla funkce *best*. Ta byla konstruována pro procházení všech jedinců aktuální populace a nalezení nejlépe ohodnoceného. Návratovou hodnotou této funkce bylo ohodnocení nejlepšího jedince aktuální populace a korespondující restrikce škály vstupních dat.

Velmi důležitou funkcí celého nástroje byla funkce *getP*. Ta byla volána každým typem evolučního algoritmu pro každého nově vytvářeného jedince. Úkolem funkce bylo zjistit, jaké jednotlivé tvary v jednotlivých intervalech daného jedince vzniknou při aktuální restrikci škály dat. Funkce *getP* byla konstruována pro využití seznamů, které byly vytvořeny pomocí nástroje **Polygon Neighbors** a následně transformovány (v bloku činností prováděných pouze jednou). Přesnější popis činnosti zjištění tvarů byl uveden pomocí poznámek přímo v programovém kódu (viz příloha 4). Jednotlivé zjištěné tvary byly reprezentovány skupinami identifikátorů polygonové vrstvy *rasToPoly*.

Úkolem funkce *getP* bylo také volání dalších návazných funkcí spojených s výpočty jednotlivých tvarů (plošek). Jednalo se o funkce *cHull*, *cEntr*, *cMetr* a *cFit*. Výstupní hodnota ohodnocení řešení byla uložena do proměnné *fit* aktuálně zpracovávané instance třídy *jedinec*.

Funkce *cHull* využívala nástroje **UpdateCursor** (ve verzi arcpy.da.UpdateCursor) k zápisu identifikátorů plošek (korespondujících s aktuální restrikcí) do polygonové vrstvy *rasToPoly*. Následně s užitím těchto identifikátorů a nástroje **Minimum Bounding Geometry** vytvořila vrstvu *rasToPolyHull* obsahující geometrii i atributové informace o konvexních obálkách plošek. Součástí těchto atributových informací byla i délka konvexní obálky, tedy údaj nutný pro výpočet metriky **Normalized Detour Index**. Tato informace byla spolu s identifikátorem plošky návratovou hodnotou funkce *cHull*. Funkce *cHull* byla jedinou operací celého bloku vykonávaného iterativně, kde bylo užito složitějších nástrojů knihovny ArcPy, bohužel se autorovi nepodařilo nalézt způsob, který by byl rychlejší, než uvedené řešení.

Funkce *cEntr* byla reprezentována výpočtem střední hodnoty entropie pro aktuální restrikci hodnot vstupních dat. Výpočet byl aplikován podle postupu Dvorský, Pászto a Skanderová (2013).

Výpočetní funkce *cMetr* byla zkonstruována k výpočtu tvarových metrik plošek. Použité tvarové metriky byly velmi dobře definovány svými vzorci (viz vzorce 12, 13, 19 a 20). Podrobný popis aplikování výpočtů metrik byl uveden přímo v programovém kódu nástroje ESMEA (viz příloha 4). Velmi zjednodušeně lze proces výpočtu metrik vyjádřit jako kombinaci identifikátorů plošek polygonové vrstvy *rasToPoly*, jejich obvodů, rozloh (získaných z již transformovaných seznamů vytvořených v bloku operací prováděných pouze jednou), délek konvexních obálek plošek korespondujících s aktuální restrikcí měřící škály dat a výpočetních vzorců metrik. Po výpočtu metrik jednotlivých plošek byly hodnoty těchto metrik sečteny a poděleny počtem plošek vzniklých aktuální restrikcí. Proces zisku rozlohy a obvodu plošek korespondujících s aktuální restrikcí byl zajištěn následovně:

- **Obvod** byl získán součtem všech obvodů plošek polygonové vrstvy *rasToPoly*, náležících do plošky korespondující s aktuální restrikcí měřící škály dat a následným odečtením součtů všech délek hranic mezi těmito ploškami.
- **Rozloha** byla získána součtem rozloh všech plošek polygonové vrstvy *rasToPoly*, náležících do plošky korespondující s aktuální restrikcí měřící škály dat.

Funkce ***cFit*** byla volána na závěr výkonu funkce *getP*. Představovala samotný **výpočet cenové funkce**, tedy zisk ohodnocení daného řešení. Její reprezentace v programovém kódu byla vizualizována níže (viz obr. 22).

```
def cFit(v):
    v = [v[0] - 1, v[1], v[2] - 1, 1 - v[3], ga.ho - v[4]]
    f = v * w
    return sum(f)
```

Obr. 22 Výpočet cenové funkce

(zdroj: vlastní zpracování).

6.7 Implementace evolučních Algoritmů

Pro konstrukci nástroje ESMEA bylo navrženo celkem pět variant evolučních algoritmů. Tři z těchto pěti byly představovány komplexními algoritmy. Jednalo se o varianty algoritmů Náhodného prohledávání, Obecného genetického algoritmu a Diferenciální evoluce. Dva zbylé algoritmy byly pouze variantami opakovaných operací křížení a mutace. Z hlediska komplexnosti řešeného problému byly jednotlivé algoritmy upraveny dle aktuálních potřeb.

6.7.1 Algoritmus náhodného prohledávání

Navrhovaná varianta algoritmu Náhodného prohledávání byla složena z funkcí *fRnd* a *rnd*. První z těchto funkcí (viz obr. 23) byla konstruována k iterativnímu volání funkce druhé. Toto iterativní volání představovalo proces vytváření jednotlivých generací algoritmu. Dále byla funkce *fRnd* využívána k záznamu jednotlivých nejlepších řešení dané generace algoritmu (ukládání probíhalo do slovníku *hist*). Na závěr byl vypsán výsledek, tedy nejlepší nalezené řešení algoritmu, reprezentované hodnotou fitness a restrikcí měřící škály geodat.

```
def fRnd(gen):
    hist = {}
    gen += 1
    for i in xrange(1, gen):
        rnd()
        hist[i] = best()
    print "Historie:"
    for x in hist:
        print x, " - ", hist[x]
    print "----------------------------"
    print "Vysledek:"
    print best()
```

Obr. 23 Funkce *fRnd*

(zdroj: vlastní zpracování).

Funkce *rnd* byla navržena tak, aby reprezentovala jednu generaci algoritmu. Mimo to, že byla použita jako primární součást algoritmu Náhodného prohledávání, byla využívána i jako funkce generování prvotní populace jedinců v ostatních typech evolučních algoritmů. Veškeré použité postupy funkce *rnd* byly opakovány i ve výpočetních funkcích algoritmů Křížení, Mutace a Diferenciální evoluce.

Nejdříve byl vytvořen seznam *popRnd* pro budoucí populaci. Tedy populaci generace *gen + 1*, kde proměnná *gen* reprezentovala aktuální generaci. Dále byl zaveden cyklus typu FOR. Tento cyklus byl použit z důvodu procházení všech jedinců populace v aktuální generaci (počet průchodů byl roven proměnné *nInd* – počtu jedinců). V tomto cyklu bylo nejdříve testováno, zda aktuální populace obsahuje jedince. Jestliže existoval jedinec s daným identifikátorem, byla získána hodnota jeho fitness a uložena do proměnné *actual*. Pakliže byla populace shledána za prázdnou (nebyl nalezen jedinec

s daným identifikátorem), bylo proměnné *actual* přiřazeno velmi vysoké číslo, kterého fitness jedince nemohla dosáhnout. V následujícím kroku byla volána instance třídy *jedinec* a přiřazena proměnné *newcre*. Tímto způsobem bylo realizováno vytváření všech nových, prozatím prázdných jedinců. Další postup algoritmu byl zajištěn přiřazením proměnným *maxBeforeIndex* (index vstupní hodnoty v seznamu vstupních hodnot), *counter* (čítač intervalů) a *maxIndex* (počet všech vstupních hodnot) potřebných hodnot.

V průběhu následujících kroků bylo realizováno zavedení cyklu typu FOR pro generování jednotlivých intervalů restrikce měřící škály dat podle zadané proměnné *nInt* (počet intervalů). Byl zaveden slovník k uložení vytvořeného intervalu restrikce. Dále pak následovalo rozdělení postupu do tří možných kroků, které byly realizovány podle aktuální pozice v rámci generování restrikce:

- Pokud byl tvořen **první interval** restrikce, bylo generováno pseudo-náhodné reálné číslo v intervalu *(0, maxIndex - (nInt - counter)*. Nově vytvářenému jedinci byly přiřazeny do prvního intervalu všechny hodnoty ze vstupního seznamu hodnot, které měly menší nebo stejné číslo indexu jako vygenerované pseudo-náhodné číslo. K proměnné *counter* byla přičtena hodnota 1.
- Pokud byl generován **druhý až předposlední interval** restrikce, byla k proměnné *maxBeforeIndex* přičtena hodnota 1 a následně zavedena proměnná *mini*. Té byla nastavena hodnota proměnné *maxBeforeIndex*. Následně realizováno generování pseudo-náhodného čísla naprosto stejným způsobem jako v předchozím případě a toto číslo bylo přiřazeno proměnné *maxBeforeIndex*. Výběr vstupních prvků do dalšího intervalu restrikce jedince spočíval ve výběru hodnot vyšších nebo rovnajících se prvku s indexem stejným jako hodnota proměnné *mini*, a zároveň indexem menším než hodnota proměnné *maxBeforeIndex*. K proměnné *counter* byla přičtena hodnota 1.
- Pokud byl generován **poslední interval** restrikce, byla k proměnné *maxBeforeIndex* přičtena hodnota 1 a následně do posledního intervalu restrikce jedince přiřazeny vstupní hodnoty s indexem větším než hodnota proměnné *maxBeforeIndex*.

Následně bylo zajištěno, aby nebyl znovu prováděn výpočet fitness jedince, pokud již byl vytvořen jedinec se stejnou restrikcí. Tato vlastnost algoritmu byla zajištěna pomocí kontroly vygenerované restrikce vůči všem doposud vygenerovaným restrikcím v úložišti již spočtených restrikcí a korespondujících hodnot fitness (uložiště bylo lokalizováno v instanci třídy *g* a jejím seznamu *bin*). Pakliže se nově vygenerovaná restrikce nenacházela ve zmíněném úložišti, bylo nutné jí podrobit výpočtu pomocí funkce *getP* (viz kapitola 6.6.2).

Po přiřazení hodnoty fitness z úložiště *bin* či novém výpočtu, byla tato hodnota srovnána s hodnotou procházeného jedince aktuální populace. Pokud byla hodnota

fitness nově vytvořeného jedince menší než aktuálního, byl do seznamu *popRnd* zařazen tento nově vygenerovaný jedinec. Naopak pokud byla hodnota fitness nově vytvořeného jedince větší než aktuálního, byl do seznamu *popRnd* zařazen aktuální jedinec.

Tyto zmíněné kroky byly prováděny pro všechny jedince aktuální populace. Jakmile byl tento proces ukončen, seznam *popRnd* přepsal aktuální populaci lokalizovanou v instanci třídy *g* a jejím seznamu *pop*. Tímto krokem bylo ukončeno jedno volání funkce *rnd* představující jednu generaci algoritmu.

6.7.2 Algoritmy založené na iteraci křížení a mutace

Varianty evolučního algoritmu inspirované pouze operátorem křížení respektive mutace (viz kapitoly 6.5.2 a 6.5.3) byly konstruovány následovně. Stejně jako u algoritmu Náhodného prohledávání, i zde bylo použito dvojice funkcí (řídící funkce *fCrs* respektive *fMut* a výpočetní funkce *crs* respektive *mut*). Funkce fCrs a fMut byly sestaveny ve stejné podobě jako již zmíněná funkce *fRnd,* až na obměny v podobě zavedení vstupní populace jedinců pomocí funkce *rnd* a následném opakovaném volání funkcí *crs* respektive *mut* namísto *rnd.*

Samotné funkce crs a *mut* byly implementací návrhu operátoru křížení respektive mutace (viz kapitoly 6.5.2 a 6.5.3) a způsobu reprezentace v programovém kódu užité pro funkci *rnd*. Jedno provedení funkce představovalo jednu generaci algoritmu.

6.7.3 Obecný genetický algoritmus

Navržená variace Obecného genetického algoritmu byla realizována pomocí řídící funkce *fCmb.* Ta zprostředkovala zavedení prvotní populace funkcí *rnd* a následné opakované volání funkcí *rnd* (funkce náhodného generování jedinců), *crs* (funkce operátoru křížení) a *mut* (funkce operátoru mutace). Volání těchto funkcí bylo prováděno po zadaný počet generací *gen* a se zadanou pravděpodobností volání jednotlivých funkcí *cmbRnd* (pravděpodobnost náhodného generování), *cmbCrs* (pravděpodobnost křížení), *cmbMut* (pravděpodobnost mutace). Generování náhodných čísel reprezentujících výběr náhodného generování jedinců, křížení nebo mutace byl zajištěn pomocí knihovny NumPy a její funkce *random.choice*. Přesnější popis funkcí rnd (viz kapitola 6. 7. 1), crs a mut (viz kapitola 6.7.2) byl také uveden v této práci. Stejně jako u algoritmu Náhodného prohledávání byla i zde pomocí řídící funkce vypsána historie nejlepších jedinců jednotlivých generací a celkový výsledek.

6.7.4 Algoritmus diferenciální evoluce

Varianta algoritmu Diferenciální evoluce nástroje ESMEA byla aplikována v programovém kódu principiálně stejně jako v případě variant algoritmu Křížení, Mutace a Obecného genetického algoritmu. Opět byla složena z dvojice funkcí, a to řídící funkce *fDif* (viz obr. 24) a výkonné funkce *dif*. Následně bylo provedeno volání funkce *rnd* pro zavedení vstupní populace. Podle zadaného počtu generací *gen* – 1 byla volána funkce

dif. Jistou obměnu oproti předchozím verzím algoritmů znamenalo poskytnutí uživatelsky zvolených hodnot pravděpodobnosti křížení *difCr* a váhy diference *difF* prostřednictvím parametrů volání funkce.

```
def fDif(gen, difCr, difF):
    hist = {}
    gen += 1
    rnd()
    hist[1] = best()
    for i in xrange(2, gen):
        dif(difCr, difF)
        hist[i] = best()
    print "Historie:"
    for x in hist:
        print x, " - ", hist[x]
    print "---------------------------"
    print "Vysledek:"
    print best()
```

Obr. 24 Funkce *fDif*

(zdroj: vlastní zpracování).

Konečné sestavení programového kódu funkce *dif* bylo realizováno pomocí modifikace postupu *RAND* (viz kapitola 4.2.1). Jednalo se o výběr tří odlišných jedinců populace různých od aktuálního jedince. Proces výběru byl proveden pomocí knihovny NumPy (funkce random.sample).

Nejdříve byl vytvořen nový seznam *popDif* reprezentující nově vytvářenou populaci jedinců pro příští generaci. Pro každého jedince aktuální generace byl proveden následující postup.

Do nově vytvořeného seznamu *k* byla zařazena maximální hodnota prvního intervalu restrikce aktuálně zpracovávaného jedince. Ostatní intervaly byly spočteny již podle zmíněného postupu *RAND*. Zde pro každý interval restrikce bylo rozhodováno, zda bude do seznamu *k* zařazena maximální hodnota daného intervalu restrikce aktuálního jedince, či hodnota spočtená z maximálních hodnot korespondujících intervalů restrikcí trojice dříve vybraných jedinců.

Seznam *k* byl seřazen od minimální hodnoty po maximální a danému jedinci podle získaných horních hranic intervalů (hodnot seznamu *k*) přiřazeny korespondující jedinečné vstupní hodnoty gridu.

Pokud byla vytvořena defektní restrikce měřící škály dat, tedy taková, která obsahovala interval roven prázdnému seznamu, byla danému jedinci přiřazena velmi vysoká hodnota fitness (jedinec byl znehodnocen). Pokud naopak vznikl relevantní jedinec, bylo navázáno na proces kontroly existence dané restrikce v úložišti již vytvořených restrikcí a případný výpočet ohodnocení. V konečném kroku bylo

ohodnocení porovnáno s ohodnocením aktuálně zpracovávaného jedince. Řešení s nižší hodnotou fitness bylo ponecháno a předáno do seznamu *popDif* reprezentujícího populaci příští generace.

Po provedení postupu pro všechny jedince aktuální populace, byl proveden přepis aktuální populace v instanci třídy *g* nově vytvořenou populací.

7 EXPERIMENTY

Po ukončení vývoje nástroje ESMEA bylo navázáno na testování různých vstupů pro jednotlivé typy evolučních algoritmů. Nejdříve byl proveden odhad počtu možných restrikcí, dále bylo opakovaným užitím nástroje zjištěno vhodné nastavení počtu jedinců a generací pro dosažení optimálních výsledků. Po zjištění těchto informací bylo provedeno testování kvality navržených algoritmů. Ze zjištěných skutečností byla vybrána trojice algoritmů pro další experimenty. Následovalo spuštění algoritmu vždy pouze pro jedno kritérium, dvojice kritérií a v konečném stádiu byly zjištěné výsledky testování dvojic aplikovány ke zjištění síly, se kterou jednotlivá kritéria při použití zvolených vstupních dat ovlivňují výsledky.

7.1 Odhad počtu možných restrikcí, jedinců a generací

Podle Dvorský, Pászto a Skanderová (2013) byl stanoven počet možných restrikcí měřící škály dat při počtu deseti unikátních vstupních hodnot. Počet intervalů N s maximálním možným počtem přípustných restrikcí byl stanoven na N = 5 a 6 (viz tabulka 2). Při těchto počtech intervalů bylo možné vytvořit celkem n = 126 odlišných typů restrikce vstupních dat. Pro testování bylo zvoleno ***N* = 5 intervalů**. Vzorec užitý k výpočtu bylo možné rekurzivně vyjádřit následovně:

$$R(n,N) = \begin{cases} n-1 & pro\ N = 2 \\ \sum_{i=1}^{n-N+1} R(n-i, N-1) & pro\ N > 2 \end{cases} \tag{28}$$

Tab. 2 Počet možných restrikcí měřící škály dat (zdroj: vlastní zpracování)

Počet intervalů	**Počet restrikcí**
1	1
2	9
3	36
4	84
5	126
6	126
7	84
8	36
9	9
10	1

Prvním úkolem pro testování nástroje ESMEA bylo zjištění optimálního počtu jedinců populace M a počtu generací G. Opakovaným spouštěním nástroje N = 50 opakování (s nastavením váhy pro kritérium entropie w = 1 a zbylým kritériím na hodnotu 0) spolu

s kontrolou podávaných výsledků, vzhledem k výsledkům prezentovaným v příspěvku Dvorský, Pászto a Skanderová (2013) bylo zjištěno, že kvalitních výsledků bylo dosaženo ve většině případů pro M = 10 jedinců do G = 50 generací. Proto byla stanovena hodnota pro **všechna následná testování *M* = 10 jedinců** a ***G* = 100 generací**.

7.2 Hledání optimálních algoritmů pro experimenty

První z řady experimentů představovalo zjištění kvality navrhovaných algoritmů. K tomu bylo využito opakovaného spouštění nástroje s rovnoměrným nastavením vah (všech pět kritérií s vahou ***w* = 0,2**). Počet opakování testu pro každý algoritmus představoval ***N* = 5**. Pořadové číslo generace dosažení výsledku bylo vždy zaznamenáno. Vyřazeny byly případy nalezené již v první generaci algoritmů (generována náhodně). Takto získané hodnoty byly podrobeny analýze a následně vybrána trojice algoritmů pro provádění dalších experimentů. Experiment byl označen pořadovým číslem 1.

Navrhované varianty Obecného genetického algoritmu (OGA) a Diferenciální evoluce (DE) byly při experimentu č. 1 spouštěny s různými nastaveními. Tato nastavení reprezentovala pravděpodobnost náhodného generování, křížení a mutace u Obecného genetického algoritmu (viz tabulka 3) a pravděpodobnost křížení *CR* spolu s váhou diference *F* u Diferenciální evoluce (viz tabulka 4).

Tab. 3 Užitá nastavení Obecného genetického algoritmu (zdroj: vlastní zpracování)

Název	**Náhodné generování**	**Křížení**	**Mutace**
OGA 060202	0,6	0,2	0,2
OGA 020602	0,2	0,6	0,2
OGA 020206	0,2	0,2	0,6
OGA 030303	0,34	0,33	0,33

Tab. 4 Užitá nastavení algoritmu Diferenciální evoluce (zdroj: vlastní zpracování)

Název	**Pravděpodobnost křížení**	**Váha diference**
DE 0805	0,8	0,5
DE 0508	0,5	0,8
DE 0505	0,5	0,5
DE 0602	0,6	0,2

Provedením experimentu č. 1 bylo zjištěno, že při nastavení rovnoměrných vah všech kritérií generuje nástroj ESMEA restrikci vstupních hodnot uvedenou viz obr. 25. Jednalo se o stejnou restrikci vstupních dat, které bylo dosaženo při experimentech Dvorský, Pászto a Skanderová (2013). Bylo určeno minimální návratové hodnoty cenové funkce $f_{cost}(x)$ = 0,169.

```
Vysledek:
[0.16931567297576122, {0: [1, 2, 3, 4, 5], 1: [6], 2: [7], 3: [8], 4: [9, 10]}]
```

Obr. 25 Výsledek při rovnoměrném nastavení vah

(zdroj: vlastní zpracování).

V průběhu experimentu č. 1 bylo také zjištěno, že při zadaném počtu jedinců $M = 10$ a generací $G = 100$ v některých z $N = 5$ provedení, algoritmy složené pouze z operátoru mutace respektive křížení uvázly v lokálním extrému (viz tabulka 5). Z tohoto důvodu nebylo těchto návrhů algoritmů dále využito při experimentech.

Tab. 5 Výsledky experimentu č. 1 pro Mutaci a Křížení (zdroj: vlastní zpracování)

Typ algoritmu	**1. výkon**	**2. výkon**	**3. výkon**	**4. výkon**	**5. výkon**
Křížení	5	-	63	75	-
Mutace	4	7	-	-	3

Zbylé typy algoritmů vykazovaly již validní výsledky (viz tabulka 6) a byly dále podrobeny analýze kvality (viz tabulka 7). První část této analýzy spočívala v součtu jednotlivých generací nalezení správného řešení (sloupec Suma 1). Těmto sumám bylo určeno pořadí od minimální hodnoty po maximální (sloupec Pořadí 1). Tímto postupem bylo zjištěno, s jakou „rychlostí“ dokáže daný algoritmus nalézt řešení. V druhé části analýzy byl určen průměr absolutních odchylek hodnot od jejich střední hodnoty (sloupec Průměrná odchylka). Pro toto hodnocení variability hodnot bylo opět stanoveno pořadí od minimální hodnoty po maximální (sloupec Pořadí 2). Tímto postupem bylo zjištěno, které algoritmy jsou nejvíce konzistentní v ohledu počtu generací potřebných k nalezení řešení. Na závěr byl proveden součet získaných pořadí (sloupec Suma 2) a určeno konečné pořadí (sloupec Celkové pořadí). Nejnižší hodnota reprezentovala lepší výsledek. Konečné pořadí bylo použito k výběru tří algoritmů (včetně daného nastavení) pro další experimenty. Jednalo se o algoritmy OGA 030303, OGA 020206 a DE 0505 (viz tabulky 3 a 4).

Tab. 6 Validní výsledky experimentu č. 1 (zdroj: vlastní zpracování)

Typ algoritmu	**1. výkon**	**2. výkon**	**3. výkon**	**4. výkon**	**5. výkon**
Náhodné prohledávání	5	9	11	2	60
OGA 060202	2	4	10	5	19
OGA 020602	3	15	2	19	31
OGA 020206	10	10	3	4	5
OGA 030303	11	3	7	3	4
DE 0805	3	10	9	11	31
DE 0508	40	19	16	19	14
DE 0505	12	16	17	7	9
DE 0602	3	2	21	2	34

Tab. 7 Analýza celkové kvality navržených algoritmů (zdroj: vlastní zpracování)

Algoritmus	**Suma 1**	**Pořadí 1**	**Průměrná odchylka**	**Pořadí 2**	**Suma 2**	**Celkové pořadí**
NP[5]	87	8	17,04	9	17	9
OGA 060202	40	3	5,2	4	7	3
OGA 020602	70	7	9,2	7	14	7
OGA 020206	32	2	2,88	2	4	2
OGA 030303	28	1	2,72	1	2	1
DE 0805	64	6	7,28	5	11	5
DE 0508	108	9	7,36	6	15	8
DE 0505	61	4	3,44	3	7	3
DE 0602	62	5	12,08	8	13	6

7.3 Testování jednotlivých kritérií

Experimentem číslo 2 byl označen výpočet jednotlivých kritérií s vahou w daného kritéria i nastavenou na hodnotu $w_i = 1$. Váha ostatních kritérií byla vždy nastavena na hodnotu 0. Důvodem experimentu č. 2 bylo zjištění, jaká restrikce škály dat bude vytvořena optimalizací vždy jednoho kritéria. K provedení experimentu bylo užito typů algoritmů určených pomocí experimentu č. 1, a to vždy s nastavením počtu jedinců $M = 10$ a počtu generací $G = 100$.

Provedením experimentu č. 2 byla získána pro každé kritérium (Fractal Dimension Index – FRA, Shape Index – SHP, Normalized Detour Index – DET a střední hodnotu entropie – ENT) návratová hodnota cenové funkce $f_{cost}(x)$ a daná restrikce. Vizuálním zhodnocením byla zjištěna trojice jedinečných restrikcí měřící škály vstupních dat. Podle těchto jedinečných restrikcí byla jednotlivá kritéria zařazena do tří skupin. Zjištěné výsledky byly vizualizovány pomocí tabulky (viz tabulka 8) a v programu ArcMap 10.3.1 byl užitím nástroje Reclassify ze vstupních dat vytvořen mapový výstup (viz příloha 1).

Tab. 8 Výsledek experimentu č. 2 (zdroj: vlastní zpracování)

Kritérium	$f_{cost}(x)$[6]	**Restrikce**	**Skupina**
FRA	0,0384	[1], [2, 3, 4, 5], [6, 7, 8], [9], [10]	1
PAR	0,0069	[1, 2], [3], [4, 5, 6, 7, 8], [9], [10]	2
SHP	0,3837	[1], [2, 3, 4, 5], [6, 7, 8], [9], [10]	1
DET	0,1756	[1], [2, 3, 4, 5], [6, 7, 8], [9], [10]	1
ENT	0,1131	[1, 2, 3, 4, 5], [6], [7], [8], [9, 10]	3

[5] Náhodné prohledávání

[6] Hodnoty zaokrouhleny na čtyři desetinná místa

Ze zjištěných údajů a následným vizuálním zhodnocením mapového výstupu (viz příloha 1) bylo zjištěno, že použité tvarové metriky vytvořily jednu z jedinečných skupin (Skupina 1 viz tabulka 8). Jedinou metrikou, která nekorespondovala s ostatními, byla Perimeter-Area Ratio. Navržená cenová funkce byla totiž sestavena tak, aby metrika hodnotila spíše velikost plošek než jejich tvarové charakteristiky (se zvyšující se rozlohou hodnota metriky klesá – z pohledu cenové funkce byla nižší hodnota lepší). Z tohoto důvodu bylo dosaženo odlišných výsledků oproti ostatním metrikám. Dále byla vizuálně zjištěna přítomnost velkých souvislých ploch u všech restrikcí vytvořených kritérii metrik.

Kritérium založené na výpočtu entropie vytvořilo také odlišnou jedinečnou restrikci (Skupina 3 viz tabulka 8). Tím bylo dosaženo zjištění, že pro daná vstupní data existuje rozpor mezi nejoptimálnějšími restrikcemi z pohledu užitých tvarových metrik a minimalizace ztráty informace dat v důsledku těchto restrikcí.

7.4 Testování dvojic kritérií

Experiment číslo 3 byl zaměřen na vytvoření kombinací dvojic kritérií. Nejdříve byl stanoven počet provedení experimentů (počet dvojic kritérií) na p = 10 (viz vzorec 2). Následně byly vytvořeny kombinace těchto kritérií a s rozdělením váhy výpočtů evolučních algoritmů vždy mezi tato dvě kritéria (každé kritérium i váha w_i = 0,5) provedeno testování.

Pro dvojice kritérií byla získána návratová hodnota cenové funkce $f_{cost}(x)$ a daná restrikce. Opět byla vizuálním zhodnocením zjištěna přítomnost trojice skupin jedinečných restrikcí měřící škály vstupních dat. Výsledky experimentu 3 byly vizualizovány pomocí tabulky (viz tabulka 9) a mapového výstupu vytvořeného pomocí nástroje Reclassify v programu ArcMap 10.3.1 (viz příloha 2).

Tab. 9 Výsledek experimentu č. 3 (zdroj: vlastní zpracování)

Kombinace kritérií	$f_{cost}(x)$[7]	**Restrikce**	**Skupina**
FRA, PAR	0, 024	[1], [2, 3, 4, 5, 6], [7, 8], [9], [10]	1
FRA, SHP	0, 211	[1], [2, 3, 4, 5], [6, 7, 8], [9], [10]	2
FRA, DET	0, 107	[1], [2, 3, 4, 5], [6, 7, 8], [9], [10]	2
FRA, ENT	0, 0771	[1, 2, 3, 4, 5], [6], [7], [8], [9, 10]	3
PAR, SHP	0, 197	[1], [2, 3, 4, 5], [6, 7, 8], [9], [10]	2
PAR, DET	0, 0929	[1], [2, 3, 4, 5], [6, 7, 8], [9], [10]	2
PAR, ENT	0, 0613	[1, 2, 3, 4, 5], [6], [7], [8], [9, 10]	3
SHP, DET	0, 2796	[1], [2, 3, 4, 5], [6, 7, 8], [9], [10]	2
SHP, ENT	0, 3061	[1, 2, 3, 4, 5], [6], [7], [8], [9, 10]	3
DET, ENT	0, 1485	[1, 2, 3, 4, 5], [6], [7], [8], [9, 10]	3

[7] Hodnoty zaokrouhleny na čtyři desetinná místa

Jak již bylo zjištěno při experimentu č. 2, některá kritéria z pohledu optimální restrikce měřící škály daných vstupních dat do intervalů vykazovala jistou míru korelace. Jednalo se primárně všechny metriky vyjma Perimeter-Area Ratio. Nebylo tedy překvapující zjištění, že kombinací těchto metrik bylo dosaženo stejných výsledků, jako při experimentu č. 2.

Dále bylo zjištěno, že v případech kombinace kritéria závislého na výpočtu entropie bylo vždy dosaženo výsledku shodného jako při výpočtu tohoto kritéria s vahou 1 (viz experiment 2). Bylo tedy usouzeno, že toto kritérium značně ovlivňuje kritéria ostatní.

Zajímavým výsledkem byla kombinace kritérií metrik Perimeter-Area Ratio a Fractal Dimension Index. Zde totiž došlo k nalezení zatím jedinečné restrikce, tedy k opravdové kombinaci těchto kritérií. Výsledná restrikce (viz tabulka 9) by tedy měla kombinovat co nejnižší tvarovou složitost (Fractal Dimension Index) spolu s co nejvyšší rozlohou plošky (Perimeter-Area Ratio). Toto tvrzení bylo potvrzeno kontrolou vzhledem k mapovému výstupu (viz příloha 2).

Důležitým zjištěním byla také viditelná dominance kritéria Entropie. Ve všech případech, kdy bylo dané kritérium přítomno, byla restrikce ovlivněna na tolik, že se změnila ve prospěch právě tohoto kritéria (výsledná restrikce byla stejná jako v případě testování samotného kritéria Entropie při experimentu číslo 2).

7.5 Stanovení síly kritérií

Výsledky experimentů 2 a 3 naznačovaly, že některá kritéria mají ve výpočtech vyšší schopnost ovlivnit kritéria ostatní. Z tohoto důvodu byla provedena analýza síly jednotlivých kritérií, a to pomocí již zmíněných výsledků experimentu 2 a 3. Ty byly zpracovány prostřednictvím Metody párového srovnávání v prostředí programu Microsoft Excel 2016.

Nejdříve byl konstruován Fullerův trojúhelník. Srovnáním výsledků experimentů 2 a 3 bylo zjištěno, zda jedno kritérium z této dvojice ovlivnilo výsledek kombinace kritérií na tolik, že se rovnal s výsledkem výpočtů pouze pro dané kritérium. V tomto případě bylo „silnější“ kritérium ve Fullerově trojúhelníku barevně zvýrazněno a čítači daného kritéria přičtena hodnota 1.

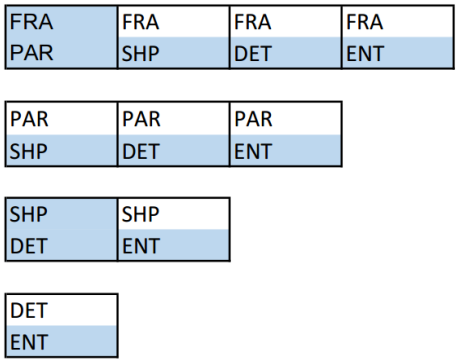
Pokud byly výsledky dvojice kritérií v experimentu 2 shodné, bylo rozhodnuto, že kritéria se vzájemně neovlivňují. Obě kritéria byla v trojúhelníku zvýrazněna a jejich čítačům přičtena hodnota 0,5. Stejně tak byla daným čítačům přičtena hodnota 0,5, pokud kombinací vznikla restrikce odlišná od obou výsledků získaných pro jednotlivá kritéria.

Po dokončení rozhodovacího procesu a přičtení všech hodnot čítačům daného kritéria, bylo užito vzorce pro výpočet vah (v tomto případě spíše síly) kritérií (viz vzorec 3). Ze získaných výsledků (viz tabulka 10 a obr. 26) bylo usouzeno, že pro použitá prostorová data byl výsledek experimentu 1 a experiment 3 značně ovlivněn spočtenými středními

hodnotami entropie (váha 0,4), dále indexy Shape a Normalized Detour (obě metriky nabývaly hodnot váhy 0,25). Poněkud upozaděn zůstal Fractal Dimension Index a metrika Perimeter-Area Ratio (obě s váhou 0,05).

Tab. 10 Výsledek stanovení síly kritérií (zdroj: vlastní zpracování)

Kritérium	**Označení**	**Počet preferencí**	**Váha**
Fractal Dimension Index	FRA	0,5	0,05
Perimeter-Area Ratio	PAR	0,5	0,05
Shape Index	SHP	2,5	0,25
Normalized Detour Index	DET	2,5	0,25
Entropie	ENT	4	0,4



Obr. 26 Výsledek stanovení síly kritérií

(zdroj: vlastní zpracování).

Informace získané v průběhu procesu zjišťování síly jednotlivých kritérií vedly k závěru, že navrhovaná cenová funkce s testovanými daty je uplatnitelná spíše pro experimenty zaměřené na zjištění optimálních restrikcí vstupních vah zaměřené pouze na jedno kritérium. Tedy na experimenty s aplikováním vah, kde jednomu kritériu i je nastavena váha $w_i = 1$ a ostatním kritériím 0. V případě rovnoměrných či jinak nastavených vah lze předpokládat, že výsledky budou ovlivněny podle síly kritérií zjištěné v průběhu procesu zjišťování síly jednotlivých kritérií.

8 VÝSLEDKY

V průběhu vypracování diplomové práce bylo dosaženo několika výsledků. Jednalo se primárně o navrženou **cenovou funkci** (viz kapitola 5) pro výpočet optimálních restrikcí měřící škály geodat s ohledem na tvarové metriky a minimalizaci ztráty informace, a to s užitím genetických algoritmů. Cenová funkce byla konstruována s ohledem na možnost uživatelsky ovlivnit váhu jednotlivých kritérií. Tedy určit, která kritéria mají být upřednostněna na úkor ostatních a realizována pomocí skalárního součinu dvojice vektorů (vektor návratových hodnot a vektor vah kritérií).

V souladu se zadáním práce bylo užito této cenové funkce spolu s několika typy evolučních algoritmů. Nejdříve bylo navrženo **operátorů selekce**, **křížení** a **mutace** (viz kapitola 6.5) vhodných pro zadaný problém (restrikci měřící škály primárních prostorových dat). Tyto operátory byly dále aplikovány do pětice modifikovaných evolučních algoritmů, a to **Náhodné prohledávání**, **Obecný genetický algoritmus**, **Diferenciální evoluce**, **Křížení**, **Mutace**.

Pro zjednodušení experimentů s navrženou cenovou funkcí a implementaci navržených evolučních algoritmů do programového kódu byl vytvořen nástroj **ESMEA** (Entropy and Shape Metrics Evolutionary Algorithms). Ten byl konstruován s ohledem na možnost výběru typu evolučního algoritmu, zadání počtu intervalů restrikce a především vstup uživatelem definovaných vah jednotlivých kritérií hledání optimálních restrikcí měřící škály vstupních prostorových dat (viz kapitola 6.6).

V průběhu experimentu číslo 1 (viz kapitola 7.2) byly pomocí nástroje ESMEA hledány optimální algoritmy pro navazující experimenty. Jednotlivé algoritmy (s různými nastaveními) byly hodnoceny z hlediska počtu generací nutných k nalezení správného řešení, ale také stability tohoto počtu. Výsledkem hledání byla **trojice algoritmů** (OGA 030303, OGA 020206 a DE 0505), pomocí kterých byly prováděny následné experimenty a **tabelární data** popisující průběh experimentu (viz tabulky 5, 6 a 7).

ESMEA byl dále použit k experimentu číslo 2. Zde bylo cílem nalezení optimálních restrikcí vstupních geodat z hlediska jednotlivých kritérií. Pro všech pět kritérií byly s již dříve nalezenou trojicí algoritmů provedeny výpočty optimálních restrikcí měřící škály geodat. Byla získána trojice jedinečných restrikcí měřící škály dat, podle těchto jedinečných restrikcí byla jednotlivá kritéria zařazena do tří skupin. Výsledky experimentu byly vizualizovány pomocí **tematické mapy** (viz příloha 1) a průběh zaznamenán v podobě **tabelárního výstupu** (viz tabulka 8). Během experimentu číslo 2 byla také zjištěna korelace mezi jednotlivými kritérii založenými na výpočtu tvarových metrik (vyjma Perimeter-Area Ratio, která byla užita spíše pro hodnocení velikosti plošek než jejich tvarové složitosti).

V průběhu experimentu číslo 3 bylo cílem zjistit restrikce měřící škály dat pro kombinace vždy dvou kritérií. Výsledky experimentu naznačovaly dominanci kritéria Entropie, jelikož v každém případě, kdy toto kritérium vstoupilo do experimentu, byla

výsledná restrikce shodná s restrikcí zjištěnou v průběhu experimentu číslo 2 právě pro kritérium Entropie. Dále byla zaznamenána nová jedinečná restrikce vzniklá kombinací kritérií Perimeter-Area Ratio a Fractal Dimension index (tedy taková restrikce měřící škály vstupních geodat, která by měla co nejvíce zvyšovat velikost plošek a zároveň se snažit o jejich nízkou tvarovou složitost). Výstupem experimentu číslo 3 byla opět trojice jedinečných restrikcí měřící škály prostorových dat. Jednotlivé kombinace kritérií byly podle příslušnosti k dané restrikci rozděleny do tří skupin. Výsledky experimentu číslo 3 byly zaznamenány pomocí **tabelárního výstupu** (viz tabulka 9) a výsledky vizualizovány pomocí **tematické mapy** (viz příloha 2).

V závěru práce byla zjištěna síla (váha) působení jednotlivých kritérií na průběh experimentů. K tomuto účelu bylo užito kombinace výsledků experimentů číslo 2 a 3. Srovnáním, zda daná kombinace kritérií (experiment číslo 3) byla ovlivněna směrem k výsledku experimentu číslo 2 ve prospěch jednoho kritéria z dané dvojice srovnávaných kritérií, byl vyplněn tzv. Fullerův trojúhelník a metodou Párového srovnávání bylo vypočítáno vah (síly) působení kritérií na experimenty. Zjištěné výsledky byly zaznamenány pomocí **tabelárního výstupu** (viz tabulka 10) a **vizualizace vytvořeného Fullerova trojúhelníku** (viz obrázek 26). Získané informace vedly k závěru, že při užití daných vstupních dat kritérium Entropie dosahuje váhy 0,4 (velmi ovlivňuje ostatní kritéria), Shape Index spolu s Normalized Detour Index 0,25 (dosti ovlivňují kritéria s nižší vahou a zároveň jsou ovlivňovány kritériem Entropie) a Perimeter-Area Ratio spolu s Fractal Dimension Index 0,05 (jsou velmi ovlivňovány ostatními kritérii).

Ze získaných informací je stanoven závěr, že navržená cenová funkce je s testovanými daty prakticky použitelná spíše pro případy, kdy jsou hodnocena **jednotlivá kritéria** (viz experiment číslo 2). Tedy nástroj ESMEA je spíše vhodný pro opakované výpočty jednotlivých kritérií vždy s vahou kritéria i nastavenou na hodnotu $w_i = 1$, zbylé váhy kritérií pak na hodnotu 0.

Na závěr byl vytvořen poster (viz příloha 3) a webové stránky přibližující tuto diplomovou práci. Všechna vstupní data a výstupy byly přiloženy na DVD-ROM (viz příloha 4).

9 DISKUZE

První otázkou, kterou si autor musel klást, byl výběr vhodných tvarových metrik. Existují velmi obsáhlé seznamy tvarových metrik, které bylo možné použít i v této diplomové práci. Použité metriky byly vybrány s ohledem na jednoduchost výpočtu (pro všechny metriky byly dostačující informace o ploše, obvodu a délce konvexní obálky plošky). Dále byl při výběru metrik kladen důraz na zachycení tvarové složitosti plošek (Shape Index), jejich celkové velikosti (zde bylo užito metriky Perimeter-Area Ratio), kompaktnost s ohledem na výstupky a podlouhlost (Normalized Detour Index) a fraktální dimenzi (Fractal Dimension Index).

Dalším důležitým krokem byl výběr vhodných evolučních algoritmů. Autor si kladl za cíl zvolit nejvíce používané typy těchto evolučních výpočetních technik. Konečný výběr byl stanoven na základě hlediska jednoduchosti aplikace (Náhodné prohledávání, Křížení, Mutace) a právě zmíněné popularity algoritmů (Obecný genetický algoritmus a Diferenciální evoluce) viz Štefan (2011).

Otázkou se také stal výběr vhodného prostředí a jazyka pro konečnou realizaci nástroje ESMEA. Skriptovací jazyk Python byl vybrán z důvodu široké podpory prostorových operací (pomocí knihovny ArcPy) a precizní dokumentace nástrojů umožňujících tyto operace provádět.

Velmi důležitou otázkou celé diplomové práce se ukázala normalizace návratových hodnot jednotlivých kritérií. V případech kombinace dvou a více kritérií (experiment číslo 1 a 2) byla pozorována vysoká váha vlivu kritéria Entropie, tedy schopnost ovlivnit ostatní kritéria. Bylo by tedy vhodné toto kritérium normalizovat jiným způsobem než použitým odečtením střední hodnoty entropie jedince H_j od střední hodnoty entropie vstupních dat H_0. Jeden z přístupů, které by podle autora bylo možné užít je spuštění nástroje ESMEA na daných vstupních datech pouze pro kritérium Entropie (váha kritéria Entropie nastavena na hodnotu 1) a zjištění minimální návratové hodnoty tohoto kritéria. Následně by zjištěná návratová hodnota mohla být užita k normalizaci všech návratových hodnot při příštím spuštění nástroje ESMEA se stejnými vstupními daty. Zjištěná hodnota by byla považována za minimální možnou hodnotu tohoto kritéria (nejlepší) a naopak hodnota H_0 za maximální možnou hodnotu tohoto kritéria (nejhorší). Tento postup by však znamenal pouze normalizaci kritéria Entropie a nutnost implementace dalšího rozhodovacího procesu do nástroje ESMEA (konkrétně zvolení, zda uživatel spouští nástroj pro zjištění této minimální návratové hodnoty pro vstupní data či zda již tuto hodnotu z předchozího spuštění vlastní a hodlá ji do nástroje ESMEA zadat pro normalizaci kritéria entropie). Zavedením zmíněného postupu by také bylo nutné vytvořit další cenovou funkci, která by již neodečítala H_j od H_0, ale pouze hodnotila dané hodnoty na intervalu $\langle 0, 1 \rangle$. Z důvodu této složitosti již navrhovaný postup nebyl aplikován, ale představuje jistou možnost pro případné rozšíření funkcionality nástroje ESMEA.

Mezi další témata vhodná k diskuzi bezesporu patří omezení plynoucí ze samotné podstaty řešeného problému. Jedná se o podmínku celočíselnosti vstupního gridu. Tato podmínka je nutná a to z důvodu možnosti výpočtu tabulky hodnot gridu a následné aplikace výpočtu střední hodnoty entropie na buňku gridu. Jistou možnost řešení představuje zaokrouhlení primárně získaných hodnot na celá čísla či vynásobení těchto hodnot určitou konstantou a následné zaokrouhlení. Takto modifikovaná data by již bylo možné v nástroji ESMEA použít. Při užití těchto postupů by však bylo nutné striktně pamatovat na aplikování těchto kroků při následné reprezentaci výsledků.

10ZÁVĚR

Cílem diplomové práce bylo navrhnout metody výpočtu cenové funkce pro evoluční algoritmy nad prostorovými daty, dále provést experimenty, výsledky vizualizovat a zhodnotit. Praktická část diplomové práce byla cílena na návrh cenové funkce, tvorbu nástroje pro zjednodušení provádění experimentů a následné provádění těchto experimentů. Zadání práce bylo aplikováno na problematiku optimální restrikce měřící škály geodat z pohledu tvarových metrik a minimalizace ztráty informace oproti vstupním datům.

Do návrhu cenové funkce byla zařazena čtveřice kritérií představujících tvarové metriky (Fractal Dimension Index, Perimeter-Area Ratio, Shape Index a Normalized Detour Index) spolu s kritériem minimalizace ztráty informace v geodatech způsobené restrikcí měřící škály těchto dat (rozdíl střední hodnoty entropie vstupních dat H_0 a jedince H_j). Vytvořená cenová funkce reprezentovala skalární součin vektoru modifikovaných návratových hodnot jednotlivých kritérií a vektoru vah těchto kritérií.

Cenová funkce byla implementována do pětice modifikovaných evolučních algoritmů (Náhodné prohledávání, Obecný genetický algoritmus, Diferenciální evoluce, Křížení a Mutace). Kombinací těchto algoritmů spolu s možností výběru algoritmu, změny jeho parametrů a váhy jednotlivých kritérií bylo vytvořeno nástroje pojmenovaného ESMEA (Entropy and Shape Metrics Evolutionary Algorithms).

Nástroj ESMEA byl následně užit pro experimentální zjištění optimálního počtu generací a jedinců k nálezu správných řešení a k analýze rychlosti a spolehlivosti nálezu správného řešení jednotlivými typy navržených algoritmů (s různými nastaveními parametrů). Z této analýzy byla vybrána trojice „kvalitních“ algoritmů pro provádění následných experimentů.

V návaznosti na předchozí experiment byly zjištěny optimální restrikce měřící škály primárních dat do intervalů s ohledem vždy na dané kritérium (dané kritérium váha 1, zbylá kritéria 0). Bylo dosaženo trojice jedinečných restrikcí měřící škály dat a podle těchto restrikcí byla jednotlivá kritéria zařazena do tří skupin. Z vytvořených skupin byla zjištěna jistá míra korelace mezi užitými kritérii tvarových metrik Fractal Dimension Index, Shape Index a Normalized Detour Index. Odlišných výsledků bylo dosaženo v případě metriky Perimeter-Area Ratio (byla užita spíše pro hodnocení rozlohy plošek, proto podávala odlišné výsledky než ostatní metriky) a kritérium Entropie.

Dále byl proveden experiment kombinace vždy dvou kritérií (každé kritérium z této dvojice s vahou 0,5). Z výsledků tohoto experimentu bylo zjištěno, že jednotlivá kritéria mají různou schopnost ovlivňovat ostatní kritéria. Opět bylo získáno trojice jedinečných restrikcí měřící škály geodat (jedna jedinečná i ve srovnání s předchozím experimentem), kombinace kritérií byly podle těchto zjištění zařazeny do tří skupin.

V závěrečné fázi bylo pomocí Metody párového srovnávání a kombinací výsledků předchozích experimentů určeno, s jakou vahou (silou) působí jednotlivá kritéria na celkový výsledek. Bylo zjištěno, že celkový výsledek s použitím vstupních dat této diplomové práce nejvíce ovlivňuje kritérium Entropie, dále dvojice kritérií Shape Index a Normalized Detour Index, nejméně výsledek ovlivňovala kritéria metrik Fractal Dimension Index a Perimeter-Area Ratio. Ze zjištěných skutečností bylo vyvozeno, že pro data zpracovávaná touto prací je navržená cenová funkce vhodná spíše pro experimenty s jednotlivými kritérii (vždy jedno kritérium s vahou 1).

POUŽITÁ LITERATURA A INFORMAČNÍ ZDROJE

BATTY, Michael. Space, Scale, and Scaling in Entropy Maximizing. In: *CASA Working Paper*. London: UCL London, 2010, s. 395–421. ISSN 1467-1298. Dostupné také z: https://www.bartlett.ucl.ac.uk/casa/pdf/paper154.pdf

BATTY, Michael, Robin MORPHET, Paolo MASUCCI a Kiril STANILOV. Entropy, complexity, and spatial information. In: *Journal of Geographical Systems*. 2014, 16(4), s. 363-385. DOI: 10.1007/s10109-014-0202-2. ISSN 1435-5930. Dostupné také z: http://link.springer.com/10.1007/s10109-014-0202-2

ČEPOVÁ, Daniela. *Stanovení vlivu geometrických vlastností povodí na odtok použitím tvarové metriky*. Olomouc, 2015. Dostupné také z: http://www.geoinformatics.upol.cz/dprace/magisterske/cepova15//index.html. Magisterská práce.

DEB, Kalyanmoy. *Multi-objective optimization using evolutionary algorithms*. New York: John Wiley, 2001. ISBN 04-718-7339-X.

DRÁŽNÁ, Aneta. *Stanovení vah a parametrů extenze Arc Urban Planner*. Olomouc, 2014. Dostupné také z: http://www.urbanplanner.cz/publikace/drazna.pdf. Bakalářská práce.

DVORSKÝ, J., PÁSZTO, V., SKANDEROVÁ, L. (2013): Geodata Scale Restriction using Genetic Algorithm, IBICA 2013, Ostrava, Springer AICS, (Thomson ISTP), pp. 215-223, DOI 10.1007/978-3-319-01781-5_20

HARTMANNOVÁ, Sylvie. *Aplikace prostorové a tvarové metriky pro hodnocení (sub)urbanizace*. Olomouc, 2013. Bakalářská práce.

HYNEK, Josef. *Genetické algoritmy a genetické programování*. Praha: Grada, 2008. ISBN 978-80-247-2695-3.

HWANG, C. L. a Abu Syed Md. MASUD. *Multiple objective decision making, methods and applications: a state-of-the-art survey*. New York: Springer-Verlag, 1979. ISBN 03-870-9111-4.

JANOŠKA, Zbyněk. *Hausdorffova dimenze při studiu sídel*. Olomouc, 2011. Dostupné také z: http://www.geoinformatics.upol.cz/dprace/magisterske/janoska11/files/dp_janoska.pdf. Diplomová práce.

JENKS, G. F. (1967): *The Data Model Concept in Statistical Mapping*. International Yearbook of Cartography, 7, s. 186–190.

KAŇOK, Jaromír. *Kvantitativní metody v geografii*. Ostrava: Ethics, 1992. Učební texty Ostravské univerzity. ISBN 80-704-2700-0.

KOŠKA, Pavel. *Hybrid Nelder-Mead a rojové optimalizace*. Praha, 2014. Dostupné také z: http://www.fel.cvut.cz/education/prace/00037.pdf. Diplomová práce.

KORVINY, Petr. *Teoretické základy vicekriteriálnıho rozhodovánı*. 2008. Dostupné také z: http://korviny.cz/mca7/soubory/teorie_mca.pdf

MCGARIGAL, Kevin. UNIVERSITY OF MASSACHUSETTS, Amherst. Fragstats HELP. 2015. Dostupné z: http://www.umass.edu/landeco/research/fragstats/documents/fragstats.help.4.2.pdf

NĚMEC, Jan. *Efektivita evolučních algoritmů*. Brno, 2016. Dostupné také z: https://dspace.vutbr.cz/bitstream/handle/11012/59821/xnemec63_diplomov%C3%A1_pr%C3%A1ce.pdf?sequence=1&isAllowed=y. Diplomová práce.

PÁSZTO, Vít. *Geoinformatické zpracování prostorové entropie klimatických jevů*. Olomouc, 2009. Magisterská práce.

PÁSZTO, Vít. *Geoinformatické zpracování prostorové entropie klimatických jevů*. Olomouc, 2009. Magisterská práce.

PÁSZTO, Vít. *Prostorová informace a vybrané metody geocomputation pro její hodnocení*. Olomouc, 2015. disertační práce (Ph.D.). UNIVERZITA PALACKÉHO V OLOMOUCI. Přírodovědecká fakulta

PARENT, Jason. *Urban Growth Analysis: Calculating Metrics to Quantify Urban Sprawl* [online]. 2008 [cit. 2016-07-30]. Dostupné z: http://events.esri.com/uc/2008/proceedingsCD/papers/papers/pap_1692.pdf

PARENT, Jason, Daniel CIVCO a Shlomo ANGEL. *Shape Metrics*. In: Center for Land Use Education and Research [online]. 2009 [cit. 2015-04-30]. Dostupné z: http://clear.uconn.edu/tools/Shape_Metrics/pubs.htm

PETŘÍKOVÁ, Kamila. *Vícekriteriální programování optimalizační úlohy*. Plzeň, 2013. Dostupné také z: https://otik.uk.zcu.cz/bitstream/handle/11025/8504/bakalarska_prace_kamila_Petrikova_2013.pdf?sequence=1. Bakalářská práce.

PEZLAR, Zdeněk. *Základy teorie informace*. Brno: Konvoj, 1998. Scriptum. ISBN 80-856-1576-2.

PŮLPÁN, Zdeněk. Ztráty informace v důsledku restrikce měřící škály. Olomouc: Univerzita Palackého, 2006. ISBN 80-244-1504-6.

ROBINSON, Arthur Howard. *Elements of cartography*. 6th ed. New York: Wiley, 1995. ISBN 04-715-5579-7.

SAVIC, Dragan. Single-objective vs. Multiobjective Optimisation for Integrated Decision Support. In:*Integrated Assessment and Decision*. 2002. Dostupné také z: http://www.iemss.org/iemss2002/proceedings/pdf/volume%20uno/399_savic.pdf

SHANNON, C. E. (1948): *A Mathematical Theory of Communication*. Bell System Technical Journal, 27, s. 379-423, s. 623–656. Dostupné také z: http://worrydream.com/refs/Shannon%20-%20A%20Mathematical%20Theory%20of%20Communication.pdf

TAN, K. C., E. F. KHOR a T. H. LEE. *Multiobjective evolutionary algorithms and applications*. 1. London: Springer, c2005. ISBN 18-523-3836-9.

STORN, Rainer a Kenneth PRICE. Differential Evolution: A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. In: *Journal of Global Optimization*. Netherlands: Kluwer Academic Publishers, 1997, s. 341–359. DOI: 1008202821328. Dostupné také z: http://jaguar.biologie.hu-berlin.de/~wolfram/pages/seminar_theoretische_biologie_2007/literatur/schaber/Storn1997JGlobOpt11.pdf

STUDNIČKA, Vladimír. *Genetické algoritmy: Multi-core CPU implementace*. Brno, 2010. Dostupné také z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=33586. Diplomová práce.

ŠTEFAN, Martin. *Studium a srovnávání hlavních typů evolučních algoritmů*. Praha, 2011. Dostupné také z: https://is.cuni.cz/webapps/zzp/detail/60592/?lang=cs. Diplomová práce.

TO, Thanh Binh a Ulrich KORN. Scalar Optimization with Linear and Nonlinear Constraints Using Evolution Strategies. In: *Bernd Reusch*. Germany: Springer-Verlag, 1997, s. 381-392.

TUČEK, Pavel, Vít PÁSZTO a Vít VOŽENÍLEK. Použití entropie při studiu nestejnorodosti geografických jevů. In: *Geografie: Sborník České geografické společnosti*. 2009, s. 117-130. Dostupné také z: http://geography.cz/sbornik/wp-content/uploads/2009/08/g09-2-3tucek.pdf

TVRDÍK, Josef. *Evoluční Algoritmy*. Ostrava: Ostravská univerzita v Ostravě, 2004. Dostupné také z: http://prf.osu.cz/doktorske_studium/dokumenty/Evolutionary_Algorithms.pdf

VOLNÁ, Eva. *Evoluční algoritmy a neuronové sítě*. 1. vyd. Ostrava: Ostravská univerzita v Ostravě, 2012, 152 s. Dostupné také z: http://www1.osu.cz/~volna/Evolucni_algoritmy_a_neuronove_site.pdf

VOŽENÍLEK, Vít. *Cartography for GIS :geovisualization and map communication*. 1st ed. Olomouc: Univerzita Palackého v Olomouci, 2005. 142 s. ISBN 8024410478.

WILSON, Alan. *Entropy in Urban and Regional Modelling*: Retrospect and Prospect. In: *Geographical Analysis*. London, 2010, 42(4), s. 364-394. DOI: 10.1111/j.1538-4632.2010.00799.x. ISSN 00167363. Dostupné také z: http://doi.wiley.com/10.1111/j.1538-4632.2010.00799.x

ZELINKA, Ivan, Zuzana KOMÍNKOVÁ OPLATKOVÁ, Miloš ŠEDA, Pavel OŠMERA a František VČELAŘ. *Evoluční výpočetní techniky: principy a aplikace*. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-218-3.

ZITZLER, Eckart. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Switzerland, 1999. PhD thesis. ETH Zurich.

PŘÍLOHY

SEZNAM PŘÍLOH

Vázané přílohy:

Příloha 1 Mapa výsledků experimentu 2

Příloha 2 Mapa výsledků experimentu 3

Volné přílohy

Příloha 3 Poster

Příloha 4 DVD

Popis struktury DVD

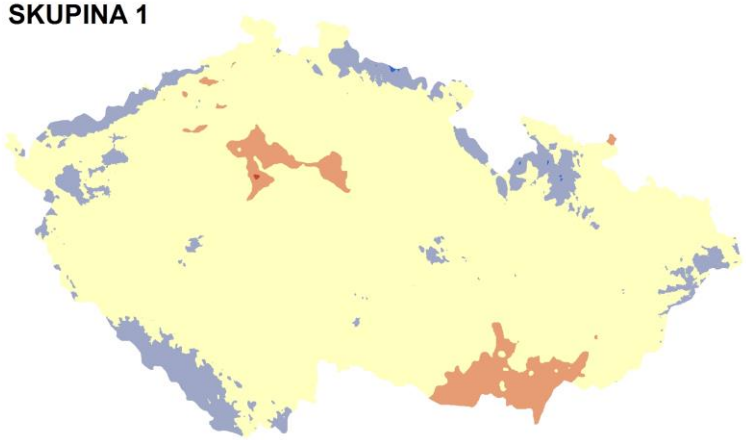
Adresáře:

- ESMEA
- Mapove_vystupy
- Tabelarni_vystupy
- Metadata
- Text_Prace
- WEB

VÝSLEDEK EXPERIMENTU 2

pro průměrné roční teploty v ČR mezi lety 1961 a 2001

SKUPINA 1



Příslušnost kritérií ke skupinám

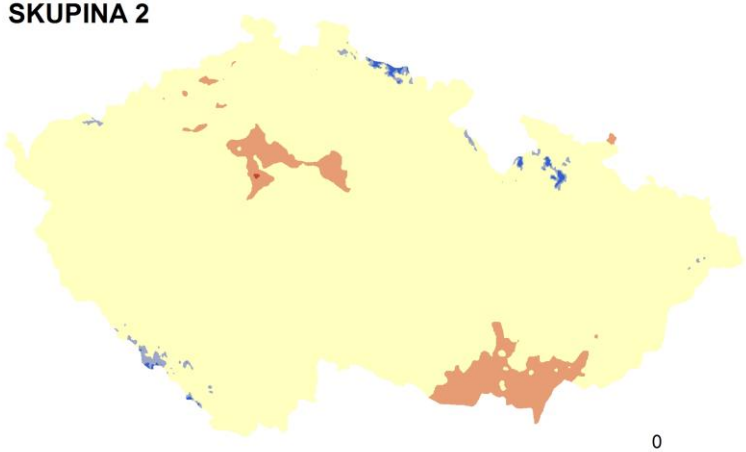
Kritérium	Skupina	Fitness nejlepšího jedince *
Fractal Dimension Index	1	0,0384
Perimeter-Area Ratio	2	0,0069
Shape Index	1	0,3837
Normalized Detour Index	1	0,1756
Entropie	3	0,1131

* hodnoty zaokrouhleny na čtyři desetinná místa

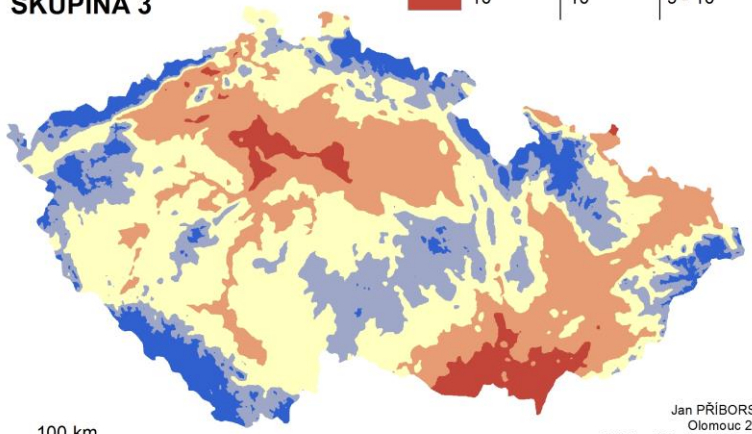
Teplota vzduchu [°C]

	Skupina 1	Skupina 2	Skupina 3
	1	1 - 2	1 - 5
	2 - 5	3	6
	6 - 8	4 - 8	7
	9	9	8
	10	10	9 - 10

SKUPINA 2



SKUPINA 3



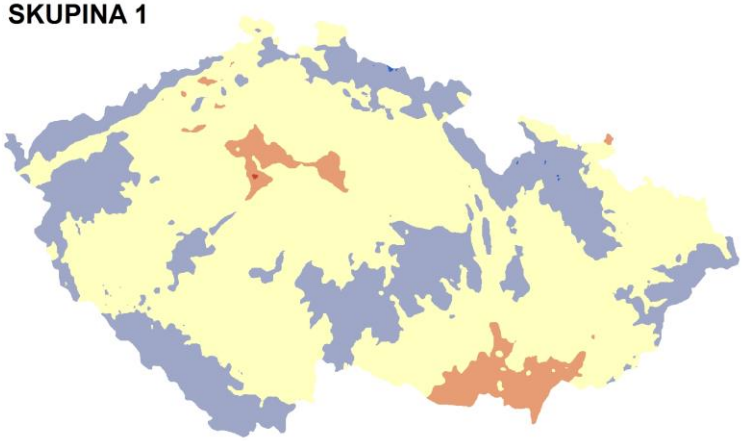
0 100 km

Jan PŘÍBORSKÝ
Olomouc 2016
Příloha 1 Magisterské práce:
Metody výpočtu cenové funkce nad prostorovými daty

VÝSLEDEK EXPERIMENTU 3

pro průměrné roční teploty v ČR mezi lety 1961 a 2001

SKUPINA 1



Příslušnost kritérií ke skupinám

Kombinace kritérií	Skupina	Fitness nejlepšího jedince *
FRA, PAR	1	0.024
FRA, SHP	2	0.211
FRA, DET	2	0.107
FRA, ENT	3	0.0771
PAR, SHP	2	0.197
PAR, DET	2	0.0929
PAR, ENT	3	0.0613
SHP, DET	2	0.2796
SHP, ENT	3	0.3061
DET, ENT	3	0.1485

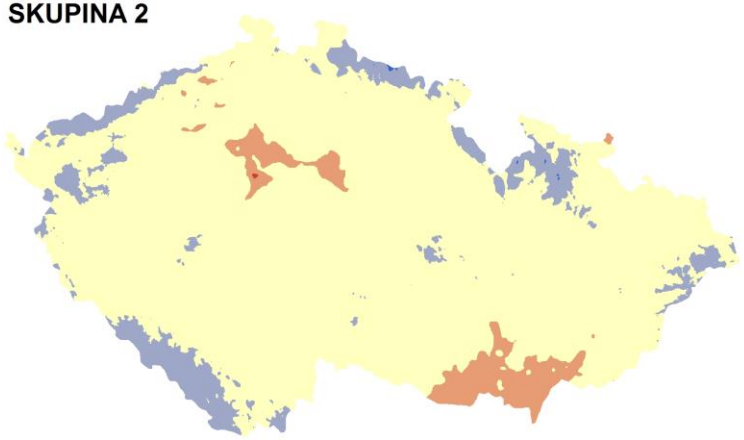
* hodnoty zaokrouhleny na čtyři desetinná místa

Teplota vzduchu [°C]

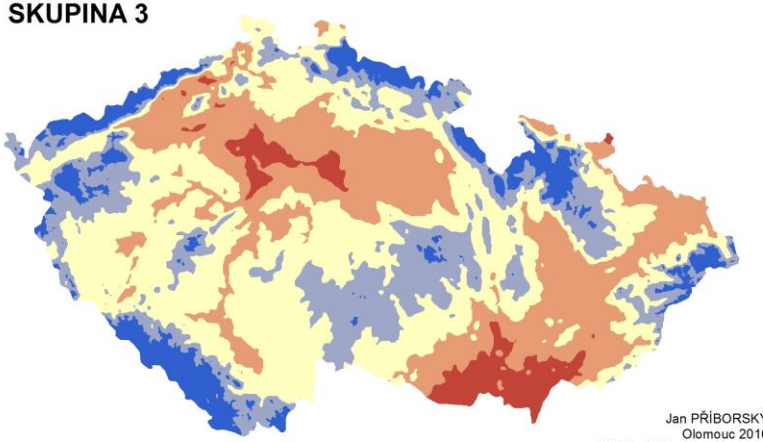
	Skupina 1	Skupina 2	Skupina 3
	1	1	1 - 5
	2 - 6	2 - 5	6
	7 - 8	6 - 8	7
	9	9	8
	10	10	9 - 10

0 100 km

SKUPINA 2



SKUPINA 3



Jan PŘÍBORSKÝ
Olomouc 2016
Příloha 2 Magisterské práce:
Metody výpočtu cenové funkce nad prostorovými daty