Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

IMPLEMENTACE ALGORITMŮ VÍCESMĚRNÉHO ODTOKU DO PROSTŘEDÍ ARCGIS

Bakalářská práce

Roman HITTL

Vedoucí práce: doc. RNDr. Vilém Pechanec, Ph.D.

Olomouc 2016
Geoinformatika a geografie

ANOTACE

Bakalářská práce se zaměřuje na implementaci algoritmů vícesměrného odtoku do prostředí ArcGIS. Výběr vhodných algoritmů byl proveden na základě rešerše teoretických algoritmů a již existujících implementací. K implementování nakonec byly vybrány teoretické algoritmy od Quinna a Freemana. V práci se popisuje implementace algoritmů „Multi flow direction 8“ a „Multi flow acummulation“ ve verzích Freeman91, Quinn 91, Quinn 95 a Quinn 95 s prahem toku. Během implementace se vyskytly problémy s velkou výpočetní složitostí, po následné úvaze mezi multiprocessingem a multithreadingem byl vybrán multiprocessing. Vznikla tak potřeba managementu procesů, což je řešeno sledováním systémových prostředků a inteligentním přidáváním procesů. Pomocí navržených testovacích ploch byly implementace otestovány. Dále proběhlo testování na digitálním modelu reliéfu 4. generace, a to na plochách povodí Všeminka, Dřevnice a Kopaninský potok.

Výsledkem práce je toolbox pro ArcGIS, Python skripty, diagramy a návrhy implementace.

KLÍČOVÁ SLOVA

Vícesměrný odtok, implementace algoritmu, multiprocessing, ArcGIS, MFD8

Počet stran práce: 42

Počet příloh: 10 (z toho 2 volné a 4 elektronické)

ANOTATION

The Bachelor thesis focuses on the implementation of algorithms multi-flow direction into ArcGIS environment. Appropriate algorithm selection is based on a theoretical algorithm and existing implementations. Finally, Quinn's algorithms and Freeman's algorithm had been chosen for implementation. The thesis describes implementation of a "Multi-flow direction 8" algorithm and "Multi-flow accumulation" algorithms in versions of Freeman 1991, Quinn 1991, Quinn 1995 and Quin 1995 with Channel Initiation Threshold. There were some problems with high computational complexity during implementation and some improvements, such as multiprocessing or multithreading, had to be implemented. Multiprocessing was chosen, but it needed some process management. The process management is based on checking of system sources and intelligent adding of processes. The implementation is tested on a prepared test area. Afterwards, tests were performed on a digital terrain model 4 generations on an area of basin Všeminka, Dřevnice and Kopanínský potok.

The result is a toolbox for ArcGIS, Python scripts, diagrams and a design of implementation.

KEYWORDS

Multi-flow direction, Implementation algorithm, multiprocessing, ArcGIS, MFD8

Number of pages 42

Number of appendixes 10

Prohlašuji, že

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

- jsem si vědom, že na moji bakalářskou 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, bakalářskou práci užívat (§ 35 odst. 3),

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

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

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

- použít výsledky a výstupy mé bakalářské 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

Roman Hittl

podpis autora

Děkuji vedoucímu práce doc. RNDr. Vilému Pechancovi, Ph.D., za podněty a připomínky při vypracování práce. Dále děkuji své přítelkyni Denise Rajmonové a rodině za podporu a trpělivost. Také chci poděkovat dobrovolníkům, kteří toolbox ochotně testovali. Dále také katedře geoinformatiky za poskytnutí licence pro software ArcGIS a přírodovědecké fakultě za poskytnutí licence pro software Wolfram Matematika.

UNIVERZITA PALACKÉHO V OLOMOUCI
Přírodovědecká fakulta
Akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	Roman HITTL
Osobní číslo:	R130491
Studijní program:	B1301 Geografie
Studijní obor:	Geoinformatika a geografie
Název tématu:	IMPLEMENTACE ALGORITMŮ VÍCESMĚRNÉHO ODTOKU DO PROSTŘEDÍ ARCGIS
Zadávající katedra:	Katedra geoinformatiky

Zásady pro vypracování:

Cílem bakalářské práce je implementace algoritmů vícesměrného odtoku do prostředí ArcGIS. Na počátku student proveden rešerši algoritmů pro stanovení vícesměrného odtoku (multiple flow) v rastrové i vektorové reprezentaci. Základním bodem bude algoritmus podle Quina a minimálně 3 další řešení. V rámci rešerše bude analyzováno, jaký algoritmus používají klíčové GIS produkty, jež práci s MF zvládají (IDRISI, Atlas DMT-Eroze; GRASS, USLE2D, TANUDEM, Landserf...).
Vlastní práce se zaměří na programování 3 algoritmů, jež budou vybrány na základě rešerše a po dohodě s vedoucím práce. Implementace bude obsahovat vlastní příkaz flowdirection, a dále odpovídající řešení pro flowaccumualtion, výpočet TCI indexu a možnost do výpočtu přidat retenční faktor v rámci modelu LOREP. V řešení bude umožněno uživateli snadno měnit algoritmus, umístění zdrojových a výstupních dat, rozlišení pixelu. Řešení bude v jazyce Python, využívat bude funkce core ArcGIS a Spatial Analyst, bude funkční pro verze 10.2+
Student vyplní údaje o všech datových sadách, které vytvořil nebo získal v rámci práce, do Metainformačního systému katedry geoinformatiky a současně vytvoří zálohu údajů ve formě validovaného XML souboru. Celá práce (text, přílohy, výstupy, zdrojová a vytvořená data, XML soubor) se odevzdá v digitální podobě na CD (DVD) a text práce s vybranými přílohami bude odevzdán ve dvou svázaných výtiscích na sekretariát katedry. O bakalářské práci student vytvoří webovou stránku v souladu s pravidly dostupnými na stránkách katedry. Práce bude zpracována podle zásad dle Voženílek (2002) a závazné šablony pro diplomové práce na KGI.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **max. 50 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

Quinn at al. (1991): The prediction of hillslope flow paths for distributed hydrological modelling using digital terrain models. Hydrological processes, vol.5,59-79(1991)
Tuček. J (1998). Geografické informační systémy. Principy a praxe. Praha, Computer press.
http://www.crwr.utexas.edu - webové stránky Center for Research in Water Resources a jejich dokumenty např. Terrain Analysis.
webové stránky uvedených programů
indexované databáze

Vedoucí bakalářské práce: **doc. RNDr. Vilém Pechanec, Ph.D.**
Katedra geoinformatiky

Datum zadání bakalářské práce: **15. června 2015**
Termín odevzdání bakalářské práce: **10. května 2016**

UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA GEOINFORMATIKY
17. listopadu 50, 771 46 Olomouc

L.S.

prof. RNDr. Ivo Frébort, CSc., Ph.D.
děkan

prof. RNDr. Vít Voženílek, CSc.
vedoucí katedry

V Olomouci dne 15. června 2015

OBSAH

SEZNAM POUŽITÝCH ZKRATEK

Zkratka	Význam
Arc Hydro Tools	Sada nástrojů pro analýzy nad DEM určená pro extrakci hydrologických informací.
ArcGIS	Geografický informační systém společnosti ESRI.
CIT	Práh trvalého vodního toku
CPU	Centrální procesorová jednotka
CUDA	Technologie pro výpočty realizované grafickou kartou společností nVidia
DEMON	Algoritmus směru odtoku učeného za pomocí digitálního výškového síťového modelu.
DMR/DEM	Digitální Model Reliéfu / Digital Elevation Model
FireStream	Technologie pro výpočty realizované grafickou kartou společností AMD
GPU	grafický procesor
GRASS	Geografický informační systém pro prostorové analýzy.
IDE	Vývojové prostředí
MFD	Multi Flow Direction
MFD8	Algoritmus směru odtoku do více směru z osmi směrů.
MFD∞	Algoritmus směru odtoku do více směru z rozsahu 0°-360°.
OpenCL	Průmyslový standard pro paralelní programování heterogenních počítačových systémů, jako jsou například osobní počítače vybavené GPU
Px	Pixel – jedná se o nejmenší obrazovou jednotku
RAM	Vyrovnávací paměť
Rho8	Algoritmus směru odtoku do jednoho z osmi směrů s náhodným prvkem.
SFD	Single Flow Direction
SFD4	Algoritmus směru odtoku do jednoho ze čtyř primárních směrů
SFD8/D8	Algoritmus směru odtoku do jednoho z osmi směrů.
SFD∞	Algoritmus směru odtoku do jednoho směru z rozsahu 0°-360°.
TAPES-G	Program pro terénní analýzy založené na gridu.
TAS	Systém pro terénní analýzy. Známý také jako "Whitebox Geospatial Analysis Tools"
TauDEM	Sada nástrojů pro analýzy nad DEM určená pro extrakci hydrologických informací.
TCI	Topografický vlhkostní index
TOPAZ	Geografický informační systém pro topografické analýzy a analýzy nad povodím.

ÚVOD

Téma této práce Implementace algoritmů vícesměrného odtoku do prostředí ArcGIS bylo zvoleno hlavně vzhledem k zájmu autora o programování, zpracovávání dat a zkoumání toho, co vše se s těmito daty dá udělat.

Text práce čtenáře uvede do problematiky modelování, objasní mu základní pojmy, jako jsou povrchový odtok, směr odtoku, odtoková síť, přispívající plocha a další související pojmy. Následně je v textu popsán návrh, implementace a testování algoritmu.

V pozadí práce je obsaženo mnohem více než jen implementace jednoduchého modelu odtoku vody z povodí. Jedná se o co nejpřesnější zachycení reálného odtoku povrchové vody a její akumulace. Proto bylo snahou navržení robustního a přesného algoritmu pro určení směru odtoku z buňky. Musely být uváženy jednotlivé možnosti odtoku a pro každou možnost vytvořen postup řešení pro určení všech směrů odtoku. Taktéž u implementace algoritmu pro akumulaci vody musejí být uváženy jednotlivé způsoby výpočtu a budou vytvořeny postupy řešení pro výpočet přispívající plochy.

Dále bude provedeno důkladné testování algoritmu na výpočetních serverech nad testovacími plochami a posléze na povodích Dřevnice, Všeminky a Kopaninského potoka. Poté by měl být navržen první prototyp uživatelského prostředí a testovací plochy, nad kterými by byly implementace testovány skupinou uživatelů. Díky tomuto testování bude zjišťována kompatibilita na různých konfiguracích a výkon těchto konfigurací. Po diskuzích s testery a z výsledků testů došlo k finální optimalizaci a bude navrženo finální uživatelského prostředí a přidána nápověda.

1 CÍLE PRÁCE

Cílem bakalářské práce je implementace algoritmů vícesměrného odtoku do prostředí ArcGIS. Na počátku student provede rešerši algoritmů pro stanovení vícesměrného odtoku (multiple flow) v rastrové i vektorové reprezentaci. Základním bodem bude algoritmus podle Quina a minimálně 3 další řešení. V rámci rešerše bude analyzováno, jaký algoritmus používají klíčové GIS produkty, jež práci s MF zvládají (IDRISI, Atlas DMT-Eroze; GRASS, USLE2D, TANUDEM, Landserf, …).

Vlastní práce se zaměří na programování 3 algoritmů, jež budou vybrány na základě rešerše a po dohodě s vedoucím práce. Implementace bude obsahovat vlastní příkaz flowdirection a dále odpovídající řešení pro flowaccumualtion, výpočet TCI indexu a možnost do výpočtu přidat retenční faktor v rámci modelu LOREP. V řešení bude umožněno uživateli snadno měnit algoritmus, umístění zdrojových a výstupních dat, rozlišení pixelu. Řešení bude v jazyce Python, využívat bude funkce core ArcGIS a Spatial Analyst, bude funkční pro verze 10.2+

Student vyplní údaje o všech datových sadách, které vytvořil nebo získal v rámci práce, do Metainformačního systému katedry geoinformatiky a současně vytvoří zálohu údajů ve formě validovaného XML souboru. Celá práce (text, přílohy, výstupy, zdrojová a vytvořená data, XML soubor) se odevzdá v digitální podobě na CD (DVD) a text práce s vybranými přílohami bude odevzdán ve dvou svázaných výtiscích na sekretariát katedry. O diplomové práci student vytvoří webovou stránku v souladu s pravidly dostupnými na stránkách katedry. Práce bude zpracována podle zásad dle Voženílka (2002) a závazné šablony pro diplomové práce na KGI. Práce bude rozdělena do několika dílčích cílů, které budou popsány v kapitokách:

1. Cíle práce, tato kapitola bude seznámením s cíli práce a obsahy jednotlivých kapitol.
2. Metody a postupy zpracování, tato kapitola bude seznámením s použitými metodami, daty, programy a postupy.
3. Současný stav řešené problematiky, tato kapitola bude seznámením s běžně používanou terminologií, typy odtoku, teoretickými algoritmy a existujícími implementacemi algoritmů.
4. Implementace, kapitola bude popisem podrobností implementace, bude se zabývat návrhem a vlastním řešením algoritmů.
5. Testování, kapitola bude popisem optimalizace a testování algoritmů.
6. Uživatelské prostředí a parametry, kapitola bude popisem uživatelského prostředí skriptu v prostředí ArcGIS a parametrů jednotlivých scriptů.

Práce si klade za cíl popsat postup implementace, optimalizace a testování skriptu určeného pro ArcGIS. V rámci práce se popíše tento postup u tří algoritmů, a to u flowdirection, flowaccumulation a TCI. U algoritmu flowaccumulation práce rozvede použití parametrů skriptu a tím změnu běhu algoritmu. V rámci dokumentace tohoto algoritmu práce poukáže na způsob ošetření parametrů, nastavení parametrů v ArcGIS, nastavení skriptu v ArcGIS a konfiguraci nápovědy pro skript.

2 METODY A POSTUPY ZPRACOVÁNÍ

Použité metody

Práce byla zahájena provedením rešerše, která měla za cíl zjistit současný stav řešené problematiky, návrhy algoritmů a existující implementace algoritmů. Z těchto algoritmů bylo nutné vybrat algoritmy vhodné pro implementaci do prostředí ArcGIS. Tento výběr byl prováděn srovnáním chování a výsledků jednotlivých typů algoritmů, dle odborné literatury a internetových zdrojů. Po několika konzultacích s vedoucím práce byly vybrány pro implementaci algoritmy od Freemana a Quinna. Dalším krokem byl návrh implementace, pro který bylo nutné zvolit software vhodný k tvorbě diagramů. Z dostupných softwarů byl nakonec vybrán draw.io, a to hlavně kvůli rychlosti použití a snadné modifikovatelnosti diagramů bez nutnosti instalace. Dále byly vybrány programy pro vývoj a testování, z dostupných možností bylo použito více softwarových řešení, z nichž nejdůležitější byl PyCharm, ve kterém následně probíhal vývoj a debugování celé implementace. Diagram byl rozdělen na menší části, které byly implementovány a testovány samostatně. Následně pak byly spojovány do hlavního skriptu. Tento skript byl poté testován a po testech se provedl finální refaktoring kódu. Nakonec bylo v závislosti na finální podobě kódu navrženo uživatelské prostředí a nápověda.

Použitá data

DMR 4G (image server poskytovaný CUZK)

http://ags.cuzk.cz/ArcGIS/rest/services/dmr4g/ImageServer

Použité programy

Použité programové vybavení by se dalo rozdělit do několika kategorií:

1. **Editory a vývojářské nástroje**
    - **PyCharm 5** je IDE pro Python vyvíjený JetBrains. Jedná se o jeden z nejkvalitnějších vývojářských prostředí pro Python. Obsahuje spoustu nástrojů, jako jsou backtrack, debug, refactor, watches, locals atd., což umožnuje komfortní vývoj aplikací. Výhodou tohoto prostředí je také bezplatná komunitní verze nebo bezplatná profesionální verze pro vzdělávací účely. V bakalářské práci byl využit pro vývoj, odladění a refaktoring kódu Python aplikace.
    - **PSPad** je volně šiřitelný univerzální editor vyvíjený Janem Fialou. Jedná se o nástroj vhodný pro práci s velkým množstvím textu, zvýraznění syntaxí, práci s více dokumenty zároveň, práci s různými programovacími jazyky, který poskytuje variabilní IDE a hlavně obsahuje inteligentní porovnávání verzí.
    - **PhpStorm 10** je IDE pro PHP vyvíjený JetBrains. Jedná se o jeden z nejkvalitnějších vývojářských nástrojů pro PHP. Užitečné funkce byly náhled na proměnné, nápověda chyb v syntaxi. V bakalářské práci byl využit pro odladění kódu webových stránek.

2. **Interpretační nástroje a knihovny**
    - **Python** je moderní, robustní, výkonný, interpretovaný jazyk, který vzniká s důrazem na jednoduchost a přehlednost. Python je vyvíjen jako open source, což mimo jiné znamená, že je k dispozici zcela zdarma.
    - **Modul numpy** je základní modul Pythonu pro práci s numerickými daty, konkrétně s n-rozměrnými maticemi. Implementace numpy je z velké části napsána v jazycích C a Fortran a využívá BLAS knihovny. Numpy tak umožňuje pracovat s numerickými daty stejně jako s Python kontejnery a přitom zachovat rychlost kompilovaných jazyků. V bakalářské práci byla knihovna použita na rychlejší práci s daty rastrů.
    - **Modul shutil** je základní modul Pythonu pro práci se soubory na úrovni nejvyššího oprávnění. V bakalářské práci je použit v aplikaci pro mazání zamčených či chybných souborů a složek.
    - **Modul time** je základní modul Pythonu pro práci s časem. V bakalářské práci je využit pro statistiky spuštění, aplikační výpočty zbývajícího a již uplynulého času a čekání manažera procesů.
    - **Modul process** je základní modul Pythonu pro práci s procesy a jejich správu. V bakalářské práci je použit pro vytváření a správu procesů.
    - **Modul math** je základní modul Pythonu pro práci s matematickými operacemi. V bakalářské práci je využit při výpočtu odmocnin, logaritmů a geometrických funkcí.
    - **Modul sys** je modul Pythonu pro práci se systémovými parametry a funkcemi. V bakalářské práci je využit pro výpis a backtrack chyby.
    - **Modul arcpy** je specifický modul Pythonu od Esri pro práci s analýzami a funkcemi produktu ArcGIS. V bakalářské práci byl využit pro otevření a uložení rastru, přebrání parametrů od aplikace ArcGIS. Dále jeho pod modul Spatial Analyst pro prostorové analýzy.

3. **Výpočetní a matematické nástroje**
    - **Wolfram Mathematica** je software od Wolfram Research pro matematické výpočty. V bakalářské práci byl využit pro navržení prvních prototypů, následné vytváření podkladů pro testy a automatické testy.
    - **Wolfram Alpha** je webová aplikace od Wolfram Research pro matematické výpočty. V bakalářské práci byla užita pro úpravy vzorců teoretických algoritmu za pomocí nástroje krok-za-krokem a nástroje ekvivalentní zápis.

4. **Nástroje pro testy a vizualizaci**
    - **ArcGIS** je geografický informační systém určený pro práci s prostorovými daty od Esri. Může zobrazovat, vytvářet, vizualizovat, analyzovat a spravovat data. V bakalářské práci byl použit pro zobrazení vstupů, výstupů a testování spuštění v prostředí ArcGIS.
    - **Screaming Frog SEO Spider** je nástroj pro analýzu webu. V bakalářské práci se využívá ke zjištění stavů (4xx, 5xx) a validaci webových stránek.
    - **Mapy API** je aplikační programovací interface od společnosti Seznam. V bakalářské práci je aplikační rozhraní použito pro mapu na web v sekci kontakt.
    - **Draw.io** je nástroj pro návrh diagramů. V bakalářské práci je použit pro návrhy a schémata.

Postup zpracování

1. Vytyčení cílů a požadavků.
2. Zjištění stavu problematiky.
3. Výběr algoritmů pro implementaci.
4. Návrh algoritmu.
5. Rozložení návrhu na dílčí celky.
6. Vyhotovení dílčích celků.
7. Testování dílčích celků.
8. Spojení dílčích celků do finálního skriptu
9. Testování finálního skriptu.
10. Optimalizace a refaktoring skriptu.
11. Vyhotovení uživatelského prostředí a nápovědy.

3 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Algoritmů pro výpočet odtoku a přispívající plochy existuje velké množství. Pro jejich rozlišení bude použito základní členění dle Argeho (2001) a česká obdoba od Konečného (2006), kteří člení algoritmy:

1. Jednosměrný odtok = single flow direction (SFD)
2. Vícesměrný odtok = multiple flow direction (MFD)

Dle Argeho (2001) být flexibilní v modelování odtoku znamená dávat uživateli na výběr mezi použitím SFD, MFD algoritmu nebo jejich kombinace.

Cílem by mělo být implementování vícesměrného odtoku, ale pro uvedení do problematiky by zde měly být uvedeny také jednosměrné algoritmy odtoku, ze kterých vícesměrný odtok vychází.

Pro základ mé bakalářské práce byl použit návrh algoritmu pro vícesměrnný odtok, jehož autorem je Freeman (1991), kterým bylo poprvé přidáno rozdělení odtoku do více okolních buněk. Ale také návrhy Quinna (1991 a 1995), kterými byl dále rozšířen Freemanův algoritmus o vrstevnicovou vzdálenost, případně parametry volitelného prahu stálého vodního toku.

3.1 Používaná terminologie

Povrchový odtok (*overland flow*)

V této práci se povrchovým odtokem rozumí model směru odtoku v rastru. Udává, zda z dané buňky odtéká voda do jedné, více či žádné sousední buňky rastru, a je proveden pro všechny buňky digitálního výškového modelu (DEM). Tribe (1992) shrnuje základní předpoklady jednoduchého povrchového odtoku:

1. odtok je produkován prostorově homogenní srážkou,
2. dopadající na zcela nepropustný povrch bez jakékoliv vegetace.

Martz a Garbrecht (1995) poznamenávají, že tuto představu je možné změnit na prostorově homogenní vegetační pokryv a povrch o konstantní propustnosti v celé ploše. Tudíž se jedná o idealizovaný odtok, při němž hydraulický spád, odpovědný za pohyb vody, je určen pouze sklonem terénu. Za těchto předpokladů je zřejmé, že takto simulovaný povrchový odtok se příliš neshoduje s reálně pozorovatelnými procesy odtoku, ale často slouží k jejich reprezentaci.

Směr odtoku (*flow direction*)

Jedná se o směr nebo směry, které určují směr, kterým odtéká voda z dané buňky rastru. Může být jeden nebo více podle toho, zda je v algoritmu povolen jeden směr odtoku (zpravidla směr odpovídající největšímu spádu), nebo je povoleno více směrů. Pokud je směr odtoku jen jeden, jedná se o jednosměrný odtok (*single flow*), naopak pokud je jich více, jedná se o vícesměrný odtok (*multiple flow*). Případně se může jednat v rámci rastru o jejich kombinaci na základě nějaké proměnné.

Odtok

Odtokem buňky nazýváme sousední buňky, do níž míří směr odtoku. U jednosměrného algoritmu mají všechny buňky nejvýše jeden odtok, u vícesměrných algoritmů pak více odtoků. Tento pojem je nejčastěji používán pro vyjádření odtoků z buňky.

Přítok

Přítokem buňky se nazývá sousední buňka, z které míří směr odtoku do dané buňky. Buňka nezávisle na typu algoritmu nemusí mít žádný nebo může mít jeden až osm přítoků. Tento pojem je nejčastěji používán pro vyjádření počtu přítoků.

Odtoková trasa

Odtokovou trasou nazýváme uspořádaný řetězec buněk, který vzniká postupným sledováním směrů odtoku. Tento řetězec spojuje danou buňku s bezodtokým místem. U vícesměrného odtoku může být buňka spojena s jedním bezodtokým místem více trasami a s více bezodtokými místy za pomoci jiných odtokových tras.

Odtoková síť

Odtokovou sítí nazýváme soustavu všech odtokových tras na určité množině buněk povodí.

Rozptyl toku

Disperzí nazýváme větvení odtokových tras na buňkách s více odtoky. K této dochází pouze u vícesměrných algoritmů odtoku. Někdy je rozptyl toku označován jako divergence nebo disperze toku.

Sjednocení toku

Sjednocení toku nazýváme spojování odtokových tras na buňkách s více přítoky. U tohoto jevu nezáleží na typu algoritmu. Někdy je rozptyl toku označován jako konvergence toku.

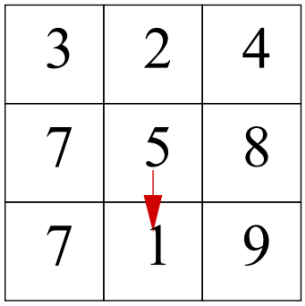
3.2 Jednosměrné algoritmy odtoku

Jednosměrné algoritmy odtoku (Single flow direction) jsou jednodušší algoritmy, avšak méně přesné a méně odpovídající realitě. Jednosměrný odtok je určení maximálně jednoho směru odtoku pro jeden pixel, a to ve směru největšího gradientu (Obrázek 3.1), kde gradientem je poměr rozdílu výšek mezi buňkami a horizontální vzdálenosti mezi nimi (mezi jejich středy).

3.2.1 SFD8 – Jednosměrný algoritmus odtoku

SFD8 je hodně jednoduchý, patří mezi nejstarší a je asi nejpoužívanějším algoritmem pro simulaci povrchového odtoku. Nazývá se také D8. První zmínky jde najít v článcích Mark (1984) a O'Callaghan a Mark (1984).

Princip



Obrázek 3.1: Jednosměrný odtok (Arge, 2001)

Pro každou buňku digitálního výškového modelu se spočítají jednotlivé sklony do všech jejích osmi sousedů podle následujícího vztahu.

$$S_i = \frac{z - z_i}{\sqrt{(y - y_i)^2 + (x - x_i)^2}} \qquad (3.2.1.1)$$

Zde S_i je sklon ve směru souseda i, z je elevace buňky, z_i je elevace souseda i, y je Y souřadnice buňky, y_i je Y souřadnice souseda i, x je X souřadnice buňky a x_i je souřadnice X souseda i. Vztah v čitateli vyjadřuje horizontální vzdálenost a ve jmenovateli rozdíl nadmořských výšek. Následovně jsou vybrány kladné sklony a ve směru toho nejvyššího sklonu je přiřazen směr odtoku. Jestliže není kladný sklon k žádnému ze sousedů, je daná buňka bez odtoku. Vzhledem k pravidelnosti sítě rastru je možné čitatel nahradit v kardinálním směru 1 a v diagonálním směru odmocninou z 2.

Výhody

Jednosměrnost, kdy každé buňce bude přiřazen pouze jeden směr odtoku do jedné jediné buňky (nebo do žádné) ve směru největšího spádu. Z čehož vyplývá snadné pochopení i implementace výpočtů přispívající plochy, kterou lze provést několika způsoby.

Implementace

SFD8 je implementován ve většině GIS, pro příklad jsou uvedeny: Arc Hydro Tools, TauDEM, GRASS, TOPAZ, TAPES-G a TAS.

3.2.2 SFD4 – Jednoduchý algoritmus odtoku

Nejjednodušší a nejstarší, ale méně používaný. Odtok je zvolen ve směru jednoho z primárních směrů. Hlavní nevýhodou je omezený počet směrů.

Princip

Popsán níže u SFD8 s tím rozdílem, že sousední buňky jsou jen v primárních směrech (sever, jih, východ, západ).

Výhody

SFD4 je vhodný pro snadné pochopení principu plošného odtoku.

Nevýhody

SFD4 je omezen v počtu směrů oproti všem ostatním algoritmům, využívá jen primární směry.

Implementace

SFD4 je implementován pro většinu GIS. Například jsou uvedeny GRASS, TOPAZ, TAPES-G a TAS.

3.2.3 Rho8 – Náhodný jednosměrný odtok

Ve snaze o zlepšení reprezentace skutečných směrů odtoku oproti D8 byl navržen algoritmus Rho8, který v roce 1991 přednesli Fairfield a Leymarie. Jejich představa byla jednoduchá: pokud směr odtoku z dané buňky je mezi dvěma povolenými směry, pak se buňka bude náhodně rozhodovat, do které ze sousedních buněk bude směřovat. Zachovává však předpoklad, že odtok z dané buňky je veden vždy nejvýše do jedné sousední buňky. Odtok se nerozptyluje, čímž se zachovává výhoda jednosměrnosti.

Princip

Rho8 je téměř stejný jako u D8, ale v případě, že více sousedních buněk má stejný sklon, přístup D8 je nevýhodný v tom, že si všechny buňky vyberou stejně, oproti tomu přístup Rho8 od Fairfielda a Leymarieho přidává náhodný prvek. Tento náhodný prvek je určen vynásobením sklonu kardinálních sousedů číslem z intervalu $(0,5;1)$ a následně je vybrán nejvyšší tzv. rho-sklon neboli sklon vynásobený náhodným číslem. Výběr už je prováděn jako u algoritmu D8. Přidáním náhodné hodnoty je docíleno toho, že někdy je vybrán sklon, který neodpovídá největšímu sklonu ve vstupním DEM, který by byl zvolen algoritmem D8.

Výhody

Oproti D8 nevznikají paralelní trasy posunuté od skutečného směru k vybranému směru.

Nevýhody

Tento algoritmus je v praxi špatně použitelný. Hlavně proto, že nerespektuje informaci uloženou v DEM a výsledek je při každém spuštění trochu odlišný. Kvůli této vlastnosti se nedá zopakovat se stejným výsledkem. U algoritmu je také nemožnost rozdělit tok.

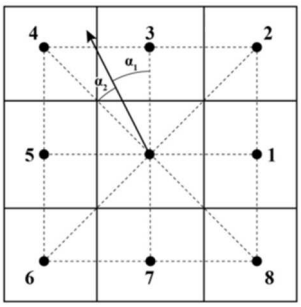
Implementace

Rho8 je implementován v TAPES-G a TAS.

3.2.4 SFD∞ – Jednosměrný odtok

Ve snaze o zlepšení reprezentace skutečných směrů odtoku, ale zachování převažující konvergence toku oproti D8, byl navržen algoritmus SFD∞. Nekonečno symbolizuje výběr z rozsahu 0–360°.

Princip



Obrázek 3.2: Výpočet směru mezi dvěma povolenými směry (Tarboton, 1997)

Středy sousedících buněk jsou propojeny na trojúhelníkové plošky, každou z těchto plošek je proložena rovina a z ní vypočten směr největšího spádu. Následně je vybrán ten směr, který odpovídá největšímu spádu. Pokud je směr odpovídající středu sousední buňky, je této sousední buňce přidělena přispívající plocha, ale ve většině případů nalezený směr největšího spádu směřuje mezi středy sousedních buněk. V takovémto případě je rozdělena přispívající plocha do těchto buněk.(Obrázek 3.2)

Výhody

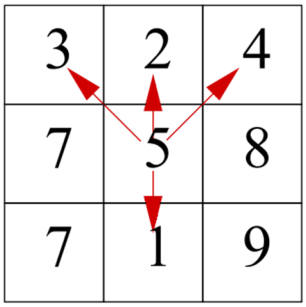
Již dochází k disperzi přispívající plochy. Díky tomu je někdy nazýván jako *bi flow.*

Nevýhody

Stále omezení na pouze dva směry odtoku. Již dochází k disperzi toku.

3.3 Vícesměrné algoritmy odtoku

Jedná se o složitější algoritmy lépe zachycující realitu a přesnější. Vícesměrný odtok určí směr odtoku do všech směrů, ve kterých vypočítá kladný gradient.



Obrázek 3.3: Vícesměrný odtok (Arge et al., 2001)

3.3.1 MFD8 – Vícesměrný odtok

MFD8 je algoritmus navržený Freemanem (1991), ten tvrdí, že voda při simulaci odtoku neteče jen ve směru největšího spádu, ale dělí se mezi směry, u nichž existuje kladný sklon.

Princip

Vybírají se všechny sousední buňky, které mají kladný sklon.

Výhody

Pokrývají se všechny možné směry odtoku. (Obrázek 3.3)[1]

Nevýhody

Dochází k disperzi toku. Narůstá výpočetní složitost. Těžší pochopení a vizualizace výsledků.

3.3.2 MFD∞ – Vícesměrný odtok

MFD∞ je algoritmus navazující na SFD∞, který navrhli Seibert a McGlynn (2007), ti aplikují poznatky Tarbotna z roku 1997, nicméně místo výběru jednoho směru vyberou všechny směry s kladným sklonem.

Princip

Vybírají se všechny směry, které mají kladný sklon. V případě výpočtu odtoku se postupuje obdobně jako v případě MFD8 algoritmu, avšak s tím rozdílem, že tok je většinou nutné rozdělit do dvou sousedních buněk, stejně jako u SFD∞.

Výhody

Pokrývají se všechny možné směry odtoku. Odtok se přerozděluje dle úhlů na základě matematických metod. Na rozdíl od MFD8 se algoritmem více reprezentuje konvergence toku, přičemž divergence toku je připouštěna pouze tam, kde existuje silná divergence v topografii. Například na sedlech, hřebenech, vrcholech atd.

Nevýhody

Výpočetní složitost vyšší než u MFD8. Těžší pochopení a vizualizace výsledků.

[1] Více v kapitole 3.4: Akumulační algoritmy, která se zabývá modely odtoku.

3.3.3 DEMON – Digitální výškový síťový model

DEMON je výsledkem snahy Costa-Cabrala a Burgese (1994) o zachování konvergence toků, ale vyhnutí se nevýhodám D8 algoritmu. Snaží se oproti osmi povoleným směrům v D8 vyjádřit odtok pomocí trubic, jejichž šířka se může průběžně měnit. U algoritmu DEMON je hlavní rozdíl v rozložení hodnot pixelů, které se zde nenacházejí ve středech uvažovaných čtverců, ale v rozích a tím umožnují vytvářet roviny. Vzhledem k tomu, že každá buňka je tvořena čtyřmi hodnotami elevace, je možné pokusit se plochu proložit metodou nejmenších čtverců.

Princip

Costa-Cabralem a Burgesem (1994) byla zvolena pro tento účel rovina, na které se spočítá směr největšího spádu a uloží si jej jako úhel od zvoleného směru, například východní směr. Pomocí úhlu je pak přispívající plocha rozdělena mezi dva kardinální sousedy. Toky rohem buňky zde neuvažují, považují jej za nekonečně malý bod.

Výhody

Řeší některé nevýhody algoritmu D8 i MFD8.

Nevýhody

Ale jak je uváděno Tarbotonem (1997), při aplikaci na reálné topografie bude vznikat problém v tom, že jednotlivé roviny budou mezi buňkami disjunktní. To znamená, že nebudou navazovat. Díky tomuto budou moct vznikat špatné odtokové cesty. Pro příklad vzájemně přispívající buňky (Tarboton, 1997). To znamená spoustu výjimek a při implementaci neúnosnou složitost algoritmu.

3.3.4 Tvarově založený přístup

Pilesjö a kol. (1998) navrhli algoritmus, který volí mezi vícesměrným a jednosměrným odtokem podle křivosti terénu.

Princip

Buňka se sousedy (3 × 3 px) je klasifikována jako jednoduchá, složitá nebo plochá. Na každou z těchto kategorií je uplatněn jiný algoritmus. Posléze je z výsledků složena odtoková síť.

Výhody

Lepší reprezentace terénu při správné kombinaci algoritmů. Kombinace ostatních algoritmů, výhody a nevýhody závisejí na použitých algoritmech.

Nevýhody

Vyšší výpočetní složitost způsobená managementem a určováním typu buňky.

3.4 Akumulační algoritmy

Jedná se o algoritmy pro výpočet přispívající plochy dané buňky podle rastru směrů odtoku, digitálního výškového modelu a dalších parametrů záležících na konkrétním akumulačním algoritmu. Tato práce se bude zabývat pouze vybranými algoritmy, a to konkrétně návrhy algoritmů od Freemana (1991), Quinna (1991), Wolocka a McCabeho ,(1995) a Quinna et al. (1995), u kterého se použijí dva návrhy – statický a s prahem trvalého toku.

3.4.1 Freemanův akumulační algoritmus z roku 1991

Poměr, který byl navržen Freemanem (1991) a používá se pro rozdělení přispívající plochy, je dán přispívající plochou buňky, velikostí sklonu mezi zdrojovou a sousední buňkou. Vzorec pro výpočet je následující.

$$\Delta A_i = \mathrm{A}\frac{\tan\beta_i^p}{\sum_{i=1}^{n}\tan\beta_i^p} \qquad (3.4.1.1)$$

Zde ΔA_i je nárůst přispívající plochy ve směru i (pokud v tomto směru existuje odtok), a je přispívající plochou dané buňky, $\tan\beta_i$ je sklon ve směru i (3.2.1.1), n je počet odtoků z dané buňky a p je parametr upravující rozptyl toku, jedná se o volitelný parametr. Čím vyšší parametr p je zvolen, tím více je upřednostněn vyšší sklon nad menším, díky tomu klesá míra rozptylu toku, a naopak čím nižší parametr p je zvolen, tím se zvyšuje míra rozptylu toku.

3.4.2 Quinnův akumulační algoritmus z roku 1991

Dalším zajímavým algoritmem je algoritmus, který byl navržen Quinem et al. (1991), který vychází z poznatku, že voda teče kolmo na vrstevnice a množství vody ve směru odtoku závisí na délce vrstevnice. Vzorec pro výpočet se závislostí na délce vrstevnice je následující.

$$\Delta A_i = \mathrm{A}\frac{L_i\tan\beta_i}{\sum_{i=1}^{n}(L_i\tan\beta_i)} \qquad (3.4.2.1)$$

Oproti Freemanovu návrhu zde není parametr upravující rozptyl toku, ale přibývá parametr L_i, pomocí kterého je reprezentována vrstevnicová délka, ale při čtvercové diskretizaci terénu zde vzniká problém, jak vyjádřit vrstevnici u diagonálních směrů. Což Quinn vyřešil rozdělením 0,5 * hrana (dále jen hr) pro kardinální směr a 0.354 * hr pro směr diagonální. Tyto hodnoty mírně upravují Wolock a McCabe (1995). Těmi je doporučováno pro kardinální sousedy 0,6 * hr a 0,4 * hr pro diagonální. Tuto úpravu odůvodňují tím, že by vrstevnice buňky v 8 směrech měly být 4 * hrany buňky. Dále bylo potvrzeno srovnávací studií, že rozdíl variant je zanedbatelný (Pan et al. 2004).

3.4.3 Quinnův akumulační algoritmus z roku 1995

Tento algoritmus propojuje algoritmy Freemana (1991) a Quinna (1991). Vznikl tak vzorec, ve kterém jsou zohledněny vrstevnicové délky i parametr pro úpravu rozptylu toku:

$$\Delta A_i = \mathrm{A}\frac{L_i\tan\beta_i^p}{\sum_{i=1}^{n}(L_i\tan\beta_i^p)} \qquad (3.4.3.1)$$

Stále však dochází k nadměrnému rozptylu toku, a to zvláště v blízkosti vodního toku, kde vytváří příliš široké proudy s vysokou hodnotou přispívající plochy.

3.4.4 Quinnův akumulační algoritmus z roku 1995 s CIT

Ovšem ve výše uvedených algoritmech bude hodnota parametru p prostorově konstantní. Takže budou zachycovat pouze částečné lokální poměry v krajině, což může způsobovat chyby, s příliš širokými proudy v blízkosti vodních toků. Tato tvrzení jsou podporována Quinnem et al. (1995) a Wolockem et McCabem (1995). Tyto problémy vedly k návrhům algoritmů s prostorově rozlišitelnou mírou divergence, kterou by se co nejlépe reprezentovaly skutečné podmínky.

Algoritmus, který by mohl vyřešit tyto problémy, byl navržen Quinnem et al. (1995). Opírá se o znalost hodnoty prahové přispívající plochy (CIT, *Channel Initiation Treshold*). Využívá přispívající plochy narůstající se zmenšující se vzdáleností k vodnímu toku. Na základě této skutečnosti mění parametr p ve vzorci (3.4.3.1), kde p je spočítáno z následujícího vztahu.

$$p = (1 + \frac{A}{CIT})^h \qquad (3.4.4.1)$$

Zde a je přispívající plocha do dané buňky, CIT je zvolený práh přispívající plochy a h je volitelný parametr. Ve vztahu je připočtena 1 z důvodů, aby koeficient p vycházel větší než 1.

3.5 Algoritmus topografického vlhkostního indexu

Je označován jako sekundární neboli složená charakteristika, vzniká kombinací primárních charakteristik. Jedná se o jeden z nejznámějších a nejdůležitějších indexů pro hydrologické modelovaní. Tento index je definován následujícím obecným vzorcem.

$$w = \ln\frac{a}{S} \qquad (3.4.4.1)$$

Zde w je výsledná hodnota vlhkostního indexu, a je specifická přispívající plocha a S je lokální sklon. Tento index je založen na předpokladu, že plošší terén (menší sklon) ležící pod dlouhým svahem je nasycen přednostně. Topografický vlhkostní index (dále TCI) je tedy přímou úměrou závislý na přispívající ploše a nepřímou úměrou na sklonu. V praxi jsou pak pomocí TCI identifikovány zdrojové plochy, a to na základě překročení prahové hodnoty TCI. TCI u vícesměrného odtoku spočítaného pomocí algoritmu MFD8 od Quinna (1991) je možné vypočíst za pomocí následujícího vzorce.

$$TCI = \frac{(\mathrm{fa} + 1) * \mathrm{s}}{\sum_{i=1}^{n} (L_i \tan\beta_i)} \qquad (3.4.4.2)$$

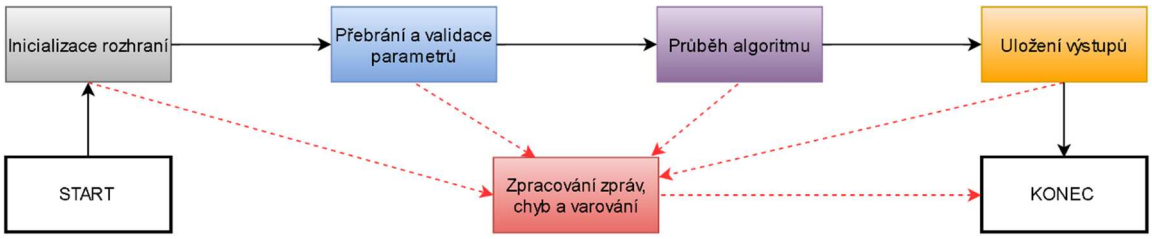
Zde fa je hodnota akumulace přispívající plochy pro danou buňku, s je velikost buňky v metrech čtverečních, L_i je vrstevnicová délka a β_i je sklon ve směru souseda i v radiánech.

4 IMPLEMENTACE ALGORITMŮ

Obsahem této kapitoly je popis praktické části pro algoritmus vícesměrného odtoku, a to zejména jeho návrh, ukázky implementace.

4.1 Návrh algoritmu

Jako první byla navržena kostra průběhu algoritmů pro ArcGIS(Obrázek 4.1).



Obrázek 4.1: Blokový diagram návrhu obecného průběhu algoritmu.

V rámci inicializace rozhraní se inicializuje interpret Pythonu, importují se jednotlivé knihovny a jednotlivé konkrétní externí metody a definují se třídy, metody a globální konstanty, v přebrání a validaci parametrů se přebírají parametry z rozhraní ArcGIS, zpracovávají se, ověřuje se jejich validita a z nich se sestavují odvozené globální konstanty, průběh algoritmu je výpočet sklonů a samotné určení směru odtoku a následné vytvoření reprezentace směrů odtoku jedním číslem, uložení výstupů, zde dochází k finálnímu dokončení rastru a jeho uložení. V případě, že se stane někde chyba, je zde blok zabývající se jeho zachycením a zpracováním chyb a varování.

Jednotlivé části se dále větvily a rozšiřoval se tak tento obecný průběh. Díky tomuto se z obecného návrhu průběhu staly návrhy konkrétních algoritmů.

4.1.1 Inicializace rozhraní

Inicializaci interpreta Pythonu v případě toolboxu plně obstarává ArcGIS a Python 2.7. Následně se importují jednotlivé moduly, jejich třídy nebo metody za pomocí funkcí import a from. Více v blokovém diagramu (Obrázek 4.2) a následujícím příkladu z implementace:

```
try:
 from shutil import rmtree
 from numpy import tile, array as narray,concatenate,asarray
 from ctypes import Structure, c_ulong, c_ulonglong, sizeof, windll,
 byref
 from time import sleep,time
 from arcpy import env, Raster, Describe, Point, RasterToNumpyArray,
 NumpyArrayToRaster
 from arcpy import GetParameterAsText, Delete_management, Exists,
 AddError
 from os import makedirs,path,popen
 from sys import exc_info
 from multiprocessing import Process
 from math import floor, ceil
except Exception,e:
 print "Chyba importu metody některé knihovny."
 print e.message
```

V ukázce je několik funkcí *from* a *import*, ty se používají pro importování funkčnosti, kde funkce *from* umožnuje nahrání částí modulů (konstanty, funkce nebo třídy).

```
from arcpy import GetParameterAsText
```

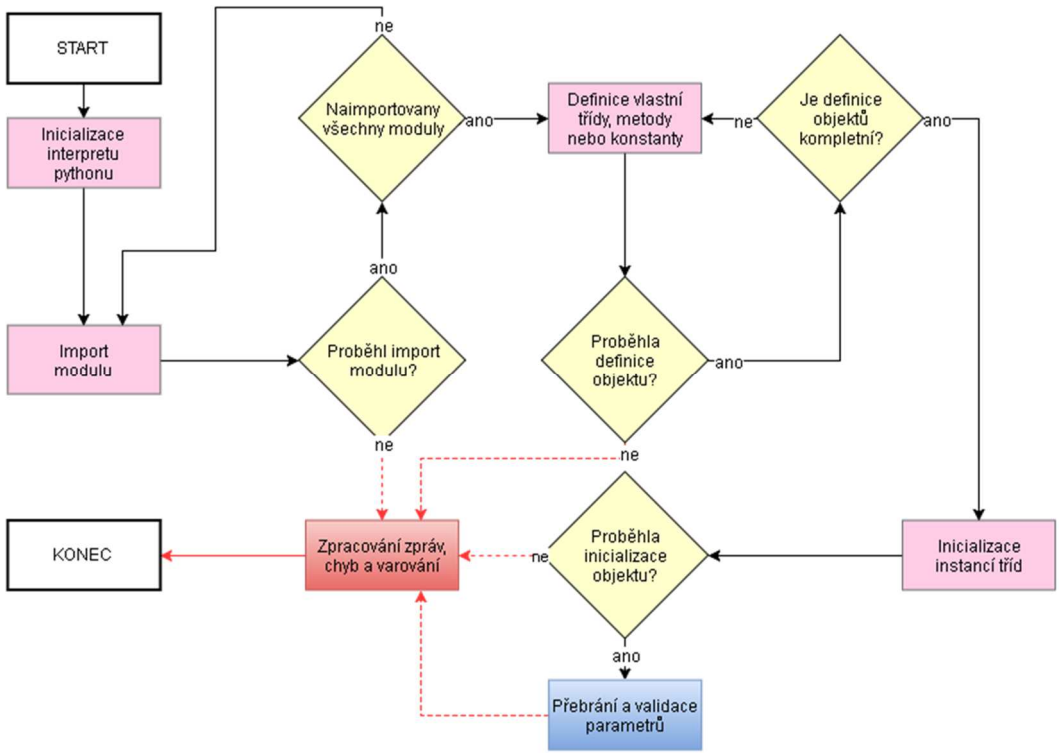
Pro nahrání celého modulu se používá *import*.

```
import arcpy
```

Dále v ukázce jsou funkce *try* a *except* pro řízení cyklu, které způsobují to, že pokud je v bloku *try* něco špatně, přeskočí do bloku *except*, který chybu zpracuje. Poslední neméně důležitou je funkce *print*, která vypisuje hlášky do konzoly.

```
print "Toto je chyba vypisující se do konzoly"
```

Jak je vidět na Obrázek 4.2, využití zpracování chyb následuje, pokud se něco nepodaří. Například není k dispozici některá z knihoven nebo je syntaktická chyba v definici objektu.



Obrázek 4.2: Blokový diagram návrhu inicializace rozhraní

Pokud je načtení modulů úspěšně dokončeno, pokračuje se ve skriptu dále definicí objektů, v této části jsou používány hlavně funkce *class* pro definici třídy:

```
class Trida (…rodičovské objekty třídy oddělené čárkou…):
 …obsah třídy…
```

Přiřazení pro definici globálních konstant.

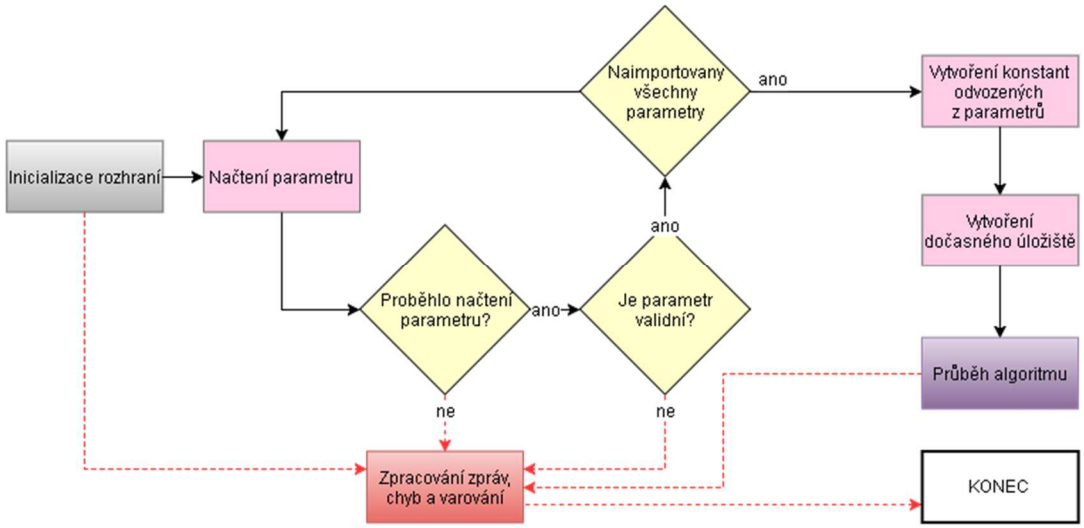
```
MAXRECURSIONLIMIT = 10
```

A funkce *def* pro definici metody či funkce, kde *rarray* je povinný parametr a limit je volitelným parametrem s výchozí hodnotou.

```
def getflowsubcords (rarray,limit = 10):
 …obsah funkce nebo metody…
```

4.2 Přebrání a validace parametrů

Přebíráním parametrů se rozumí import některých konstant z jiného místa, než je zdrojový kód. Nejčastěji se jedná o parametry příkazové řádky, v našem případě se budou přebírat parametry z prostředí ArcGIS pomocí rozhraní skriptu, které lze nastavit dle potřeb (viz kapitola 6). Načtení parametrů a jejich další zpracování je znázorněno následujícím diagramem.



Obrázek 4.3: Blokový diagram návrhu přebrání a validace parametrů.

Samotné načtení parametru z prostředí ArcGIS probíhá pomocí funkce z modulu *arcpy* s názvem *GetParameterAsText*, jejímž argumentem je pořadí parametru v ArcGIS skriptu. Následně se pak provede validace tohoto parametru. Ověřuje se, zda parametr vůbec existuje a je zadán.

```
self.DEM_in = GetParameterAsText (0)
```

```
if self.DEM_in == '#' or not self.DEM_in:
 raise (Exception ("Nebyl vložen digitální model terénu"))
```

Pokud toto nesplňuje, posune o úroveň výše výjimku, která je zpracována za pomoci funkce *exception*, která provede zachycení a zpracování chyby. Dále se ověřují již konkrétní vlastnosti parametrů u cest vstupních souborů, zda soubor existuje.

```
if not Exists (self.DEM_in):
 raise (Exception ("Digitální model terénu nebyl nalezen na disku."))
```

U čísel, zda jsou ve stanoveném rozsahu.

```
if not self.CPU_usage < 0 or self.CPU_usage > 100:
 raise (Exception ("Využití vašeho procesoru není nastaveno správně.
 Uveďte jej prosím v procentech, to je v rozmezí 0-100"))
```

Pokud jsou všechny povinné parametry v pořádku, následuje jejich zpracování. Vytvářejí se z nich konstanty, které jsou dále využívány v rámci běhu hlavního programu. V našem případě tyto konstanty budou ukládány jako atributy třídy u algoritmu pro směr odtoku do třídy *WaterFlow* a třídy *WaterAccumulation* pro výpočet přispívající plochy. Přístup k atributům třídy lze realizovat několika způsoby, v této práci je nejčastěji využívána tečková notace. Ta může být užita interně v rámci třídy, kde se využívá klíčového slova *self* nebo externě, kde se využívá objektu instance třídy.

```
self.DEM_in
```

```
waterflow.DEM_in ()
```

Nakonec je vytvořeno dočasné úložiště v podobě dočasné složky v místě výstupního adresáře. Pokud vše proběhlo v pořádku, tak běh skriptu pokračuje k samotnému algoritmu pro výpočet konkrétní charakteristiky.

4.3 Průběh algoritmu

Při výpočtu konkrétních charakteristik se ve většině případů užívá matematických vyjádření, kde se u algoritmu pro určení směru využívá porovnávání sklonu mezi buňkami rastru a u algoritmu pro výpočet přispívající plochy je užíváno matematických vztahů[2] k přerozdělení přispívající plochy sousedním buňkám s kladným sklonem.

4.3.1 Načtení rastrů

U obou algoritmů jsou vstupem do algoritmu rastry, ať už digitální výškový model, nebo digitální odtokový model u algoritmu akumulace přispívající plochy. Z rastru je nutné, kromě dat rastru samotného, dostat také podrobnosti o rastru, které jsou nutné pro výpočet a vytvoření výstupního rastru. Tuto činnost vykonává funkce *loadraster* za pomoci několika funkcí z modulu *arcpy*:

```
def loadraster (self):
 inraster = Raster (self.DEM_in)
 self.dsc = Describe (inraster)
 self.rwidth = inraster.width
 self.rheight = inraster.height
 self.nodata = inraster.noDataValue
 self.ext = self.dsc.Extent
 self.referencepoint = Point (self.ext.XMin, self.ext.YMin)
 del inraster
```

Zde se za pomocí funkce Raster načte rastr do proměnné *inraster*, ze které se následně pomocí funkce *Describe* a atributů objektu získávají informace o rastru. Nakonec se odstraní již zbytečná proměnná *inraster* z paměti. Získané informace jsou následně použity pro inicializační proměnné, jako například výpočet počtu řádků rastru na jeden proces, počet všech pixelů nebo výchozí hodnoty pro rozdíly využití zdrojů.

Pokud potřebujeme pracovat s daty z rastru, je rastr převeden na vícerozměrné numerické pole (matici) pomocí funkce *RasterToNumpyArray* z modulu *numpy*.

```
array = RasterToNumpyArray (raster)
```

Matice je pro manipulaci s daty výhodnější.

4.3.2 Implementace „Multi Flow Direction“

Zde se algoritmy již neshodují, proto jsou popsány odděleně. Tato kapitola se bude zabývat implementací MFD8 s využitím modulu *multiprocessing*. Konkrétně se využívá jeho funkce *process*. Díky rozdělení na více procesů umožnuje efektivněji využít systémové prostředky, čímž se urychluje výpočet v závislosti na výkonu konkrétního počítače, ale za cenu toho, že se musí spravovat procesy a kontrolovat využití systémových prostředků.

[2] Více v kapitole 3.4: Akumulační algoritmy.

4.3.2.1. Management procesů

U multiprocesních aplikací by měl být kladen důraz hlavně na precizní a robustní provedení managementu procesů. Důvodů je více, ale hlavní jsou zamezení přetížení počítače a co nejlepší využití systémových prostředků. Manažer procesů začne tím, že si zjistí výchozí stav systémových prostředků, v našem případě volný prostor na procesoru a množství volné vyrovnávací paměti.

Systémové prostředky

V jakém stavu se nachází procesor, zjistíme pomocí funkce *popen* modulu *os*, který nám umožní poslat příkaz příkazové řádce:

```
processor = popen ('wmic cpu get loadpercentage')
result = processor.read ()
processor.close ()
total_cpu = 0
for m in result.split (" \r\n")[1:-1]:
 total_cpu += int (m)
return 100-total_cpu
```

Tento příkaz vrátí jako výstup textový řetězec obsahující aktuální využití procesoru. Následovně je hodnota odečtena od 100 a tím zjistíme využitelnou kapacitu na CPU, kterou potřebujeme pro zajištění optimálního výkonu skriptu. U vyrovnávací paměti je to o něco složitější. V práci bylo použito řešení, které využívá modulu *ctypes*, který vnitřně zavolá funkci *GlobalMemoryStatusEx* () z knihovny windows pro jazyk C.

ctypes.windll.kernel32.GlobalMemoryStatusEx (ctypes.byref (stat))

Touto funkcí se naplní struktura proměnné stat, ve které jsou následovně obsaženy atributy o paměti. Nás zajímá atribut stat.dwMemoryLoad, který obsahuje právě využívanou paměť v procentech. Hodnota je odečtena od 100, aby byla získána nevyužitá vyrovnávací paměť.

Rozdíl systémových prostředků se využívá v případě, že již běží jeden nebo více procesů. Ověřuje se podmínka, zda jsou dostupné systémové zdroje převyšující systémové zdroje, které potřeboval poslední spuštěný proces. Z toho plyne, že rozdíl systémových prostředků se změní až se startem dalšího procesu.

Start nového procesu

Pokud jsou podmínky spuštění splněny, spustí se nový proces a přidá se do listu aktuálně běžících procesů. Spuštění nového procesu vytváří funkce *Process* z modulu *multiprocessing*:

```
self.processList.append (Process (target=line_worker, args=
(…argumenty…)))
```

Zde můžete vidět, že hned po inicializaci je instance objektu typu *Process* přidána na konec listu *self.processList*, který nám dále bude sloužit ke komunikaci s procesem.

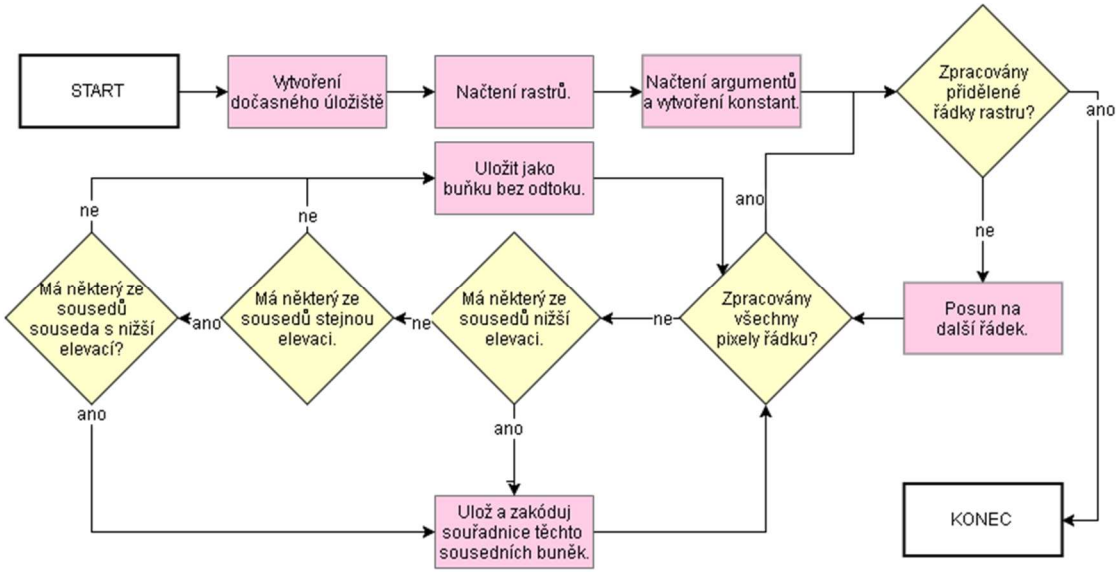
4.3.2.2. Samostatný proces

Nový proces je inicializován s argumenty rozsah řádků rastru, které daný proces má zpracovat, cestu k DEM, počet jeho řádků a sloupců a limit hloubky rekurze. Na začátku je nutné vytvořit procesu jeho individuální pracovní prostředí. V tom je zahrnuto vytvoření individuálního workspace, scratchWorkspace, složky temp a načtení vstupního DEM do pole.

Dále se zjišťuje, zda konec rozsahu pro zpracování není mimo raster, pokud je, tak je nastaven na konec rastru. Proběhne ještě nastavení hodnoty pro *noData* v našem případě na -1 a následně se vygeneruje přes funkci *tile* nové numerické pole:

```
noflow = -1
drarray = tile (noflow, (stop-start, rwidth))
```

V následujícím kroku se prochází raster směrem zleva doprava, řádek po řádku, a v každé buňce s platnou hodnotou výšky se určují sousední buňky, pro které existuje odtok a případně minimální hloubka rekurze, pokud nemá přímý odtok do některé sousední buňky, ale odvozený odtok má.



Obrázek 4.4: Diagram návrhu samostatného procesu.

Souřadnice odtoku

Funkce *getflowsubcords* je uvozena částí, kterou se ošetřují hrany rastru. Ošetřením se odstraní neplatní sousedé buňky. Například u buňky na pozici nejvíce vlevo neexistuje žádný levý soused.

```
if defaultcordX == 0:
 for subcord in subcords:
 if -1 == subcord[0]:
 pass
 else:
 copy.append (subcord)
```

Zde *defaultcordX* je x souřadnice buňky, *subcords* je list uspořádaných dvojic souřadnic možných sousedů a *copy* je dočasná kopie těchto souřadnic. V listu *copy* jsou již vyfiltrováni leví sousedé buňky.

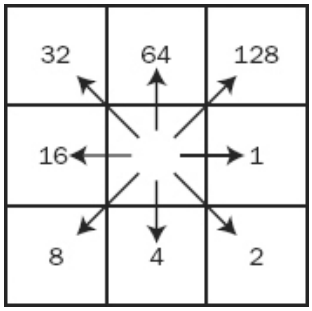
Jak jsou vybráni možní sousedé pro odtok, jsou zkopírováni opět do listu *subcords*. Ten je procházen, nemá-li soused platnou hodnotu výšky, je vyřazen, v opačném případě je vypočten rozdíl výšek. Pokud je rozdíl výšek kladný, je soused přidán do listu *copy*. Pokud je rozdíl roven nule, tak do listu *notflowcopy*.

Jestliže je list copy naplněn alespoň jedním prvkem, vrátí funkce list *copy*. Pokud je list *copy* prázdný a list *notflowcopy* naplněn alespoň jedním prvkem, prochází se list *notflowcopy* a zjišťuje se, zda okolní buňky souseda se stejnou elevací mají odtok. Zjišťuje se tak vzdálenost k nejbližšímu odtoku. Tato rekurze je ukončena v případě

překročení limitu rekurze nebo nalezením směru nejbližšího odtoku. Při rekurzi se ukládají buňky, které již v dané větvi rekurze byly navštíveny, a tím se zamezuje cyklení rekurze.

Kódování souřadnic

Pro zapsání do rastru by měly být souřadnice sousedů, do kterých buňka přispívá, zakódovány ideálně do celého čísla. Proto bylo zvoleno zakódování do 1bajtu, čímž je reprezentován rozsah neznaménkového celočíselného datového typu od 0–255 a je zakódován systémem, kde odtok do všech sousedů je reprezentován součtem hodnot, které mají určené daní sousedi. První soused je reprezentován 1, druhý a každý další dvojnásobkem předchozího souseda a tak dále až po osmého souseda, jenž je reprezentován 128. Díky tomuto vznikne 255 unikátních kombinací pro všechny směry odtoku a 1 hodnota pro bezodtokou buňku.



Obrázek 4.5: Kódování odtoku.

Uložení části rastru

Jakmile je celé numerické pole zakódováno, je pomocí funkce *NumpyArrayToRaster* převedeno na rastr a uloženo do dočasného adresáře. Pokud vše proběhlo v pořádku, proběhne funkce *exit (0)*, pokud se něco nepovedlo, případně nastala chyba, proběhne *exit (3)*.

Ukončení procesu

Pokud proces skončí, čeká ve stavu ukončen, než hlavní proces spustí funkci *processcheck*. Tato se pustí do procházení listu procesů, u kterých se zjišťuje, jestli ještě běží pomocí funkce *proces.is_alive ()*. Pokud ano, je prověřován další proces, a pokud ne, zjišťuje se *exitcode*; pokud je menší nebo roven 1, tak je vše v pořádku. Poté proběhne přidání argumentů procesu do listu *self.tempList*, předání systémových prostředků podprocesu do správy hlavního procesu a odstranění procesu z listu *self.processList*.

Pokud však je *exitcode* vyšší než 1, provede se restart procesu. Případně vypíše konkrétní chybovou hlášku.

Spojování a uložení rastrů

Když je v listu *tempsList* více než 5 položek nebo již zpracování rastrů skončilo, tak seřadí *tempList* dle prvního čísla uspořádané dvojice každého prvku.

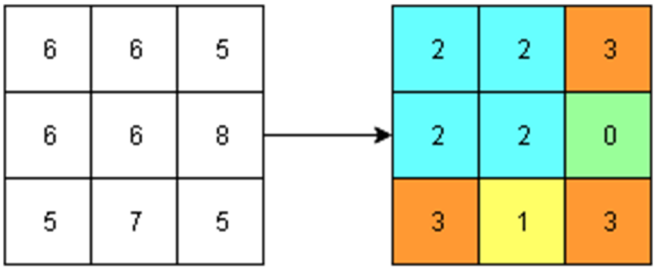
List *tempsList* je procházen a v případě, že některá ze sousedních uspořádaných dvojic navazuje na aktuální uspořádanou dvojici tak jsou spojeny v jednu dvojici. Například pokud *tempList* je [[0,5],[5,10]], tak po dokončení cyklu *tempList* bude [[0,10]]. Při spojení zároveň probíhá spojení dočasných rastrů. Pokud je ukončeno zpracování a v *tempListu* je obsažen již pouze jeden prvek, tak se uloží poslední prvek *tempListu* jako výstupní raster se stejným rozlišením a referencí jako vstupní raster DEM. Následně jsou vyprázdněny a smazány dočasné složky.

4.3.3 Implementace „Multi Flow Accumulation“

Algoritmus přispívající plochy se již nedá tak snadno rozdělit na více souběžně běžících výpočtů, ale naštěstí u tohoto výpočtu není výpočetní složitost tak vysoká. Tak je algoritmus implementován bez využití multiprocesingu. Navíc je tento algoritmus implementován s možností více variant výpočtu přispívající plochy.

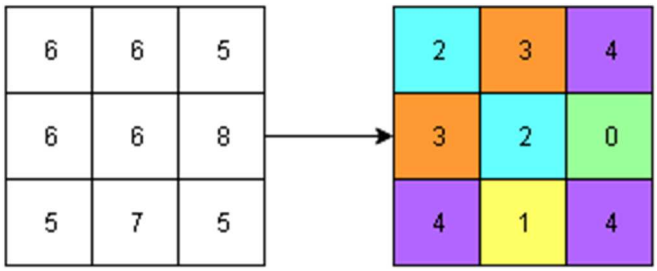
Příprava unikátních výškových záznamů

Jakmile je počítána přispívající plocha nebo akumulace vody, je nutné postupovat od nejvýše položených buněk k těm nejníže položeným, aby nebylo nutno přepočítávat již vypočtené buňky. Takovýto postup se nazývá iterační kaskáda.



Obrázek 4.6: Iterační kaskáda.

U takovéto iterační kaskády je ovšem problém z místy, která se neliší nadmořskou výškou. Tento problém je v rámci implementace řešen přidáním vnitřní interakce navíc, která řeší to, že první se zpracovávají buňky bez přítoku z buněk stejné elevace a následují buňky, které již mají vyřešený přítok, a takto se to opakuje až do doby, kdy všechny buňky budou vyřešeny korektně. Tímto postupem seřazené pole je nazváno jako rozšířená iterační kaskáda.



Obrázek 4.7: Rozšířená iterační kaskáda.

Pro sestavení iterační kaskády je nutné, aby se z pole hodnot vyfiltrovaly unikátní hodnoty elevace. Což umožnuje funkce *unique* z modulu *numpy*.

```
uniqueelevation = numpy.unique (self.rarray)
```

Dále se využívá funkce *sort* z téhož modulu a vestavěné funkce Pythonu *reversed*.

```
self.sorted_elevation = reversed (numpy.sort (numpy.unique
(self.rarray)))
```

Díky tomuto je připraveno pole unikátních hodnot elevace seřazené od nejvyšších výšek po nejnižší. Takto připravené pole je dále procházeno při zpracování jednotlivých pixelů. Pokud je více sousedících buněk stejné nadmořské výšky, je kaskáda rozšířena tak, aby nedocházelo k přítoku do již vyřešené buňky.

Zpracování jednotlivých pixelů

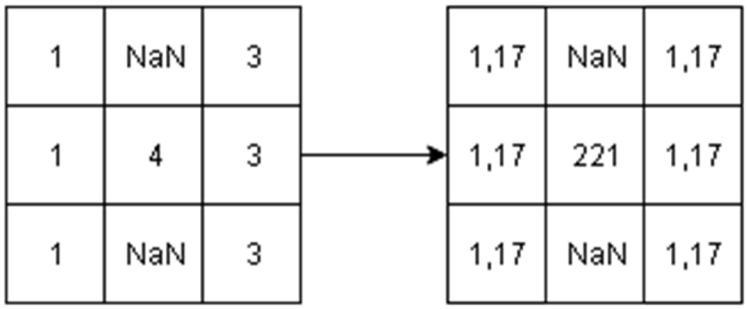
Při zpracování se vyberou buňky nejvyšší nezpracované hodnoty výšky. Z těchto buněk se postupně zpracovávají všechny. Pokud nastane problém s počtem pixelů stejné výšky, tak se pole rozšíří o určení priority zpracování jednotlivých pixelů tak, aby nedocházelo k odtoku do již zpracovaného pixelu. Následně se načtou buňky, do kterých odtéká z buňky voda. Tyto jsou vypočteny prvním algoritmem (4.3.2). Dále jsou mezi zdrojovou buňkou a sousedními buňkami vypočteny rozdíly vzdálenosti a vzdálenost mezi buňkami, z nichž je následně vypočten sklon mezi jednotlivými buňkami. Následně probíhá zpracování dle zvoleného algoritmu.

Implementace „Freeman 91“

V tomto algoritmu je využito pouze parametru *power*, který určuje sílu preference největšího sklonu. Implementován je následovně:

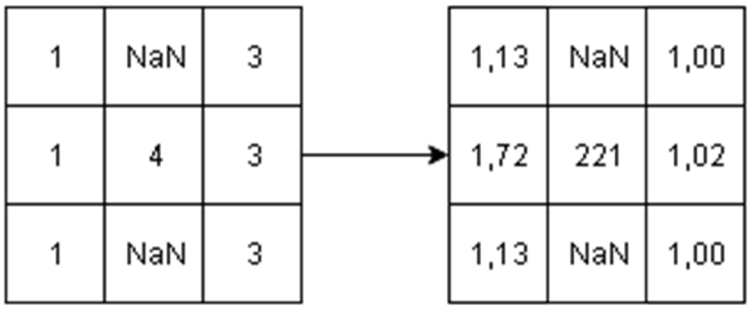
```
tempsolution = pow (subcords[idx][2], power)
```

Vliv tohoto parametru je ovšem rozdílný pro primární a sekundární buňky, a to z důvodu, že při stejné elevaci buňky v primárním i sekundárním směru je sklon různý, protože je rozdílná vzdálenost mezi pixely. V případě, že je parametr p 0, vliv sklonu zaniká úplně.



Obrázek 4.8: Rozdělení přispívající plochy dle Freemana 91 při homogení srážce 1 mm na pixel a parametru p=0.

Čím je parametr p vyšší, tím více zvýhodňuje směr vyššího sklonu. To vede k nižšímu rozptylu toku. Doporučované hodnoty se různí podle lokalit.



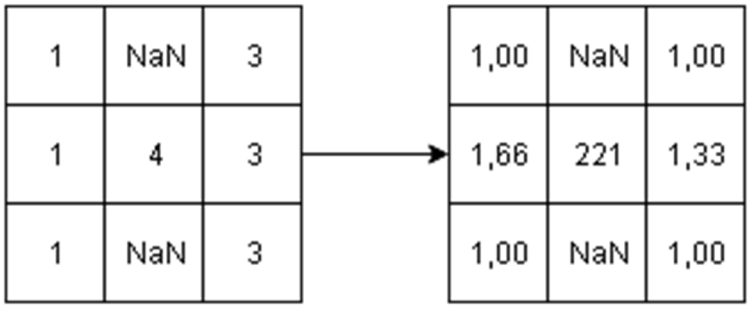
Obrázek 4.9: Rozdělení přispívající plochy dle Freemana 91 při homogení srážce 1 mm na pixel a parametru p=5.

Implementace „Quinn 91“

Tento algoritmus je řešen pouze parametrem contouredsection, který určuje rozdělení přispívající plochy podle vrstevnicové vzdálenosti. Implementace je pak:

```
tempsolution = contouredsection* (subcords[idx][2])
```

V tomto algoritmu se parametry vrstevnicové vzdálenosti dá například omezit nebo eliminovat odtok v sekundárním směru. Jak je vidět na následujícím obrázku, při nastavení vrstevnicové délky na hodnotu 0 je v daném směru odtok ignorován.



Obrázek 4.10: Rozdělení přispívající plochy dle Quinna 91 při homogení srážce 1 mm na pixel délce vrstevnice v primárním směru 1 a v sekundárním 0.

Implementace „Quinn 95“

Tento algoritmus je kombinací výše zmíněných implementací, využívá parametry *power* a *contouredsection*. Implementován je následovně.

```
tempsolution = contouredsection* (pow (subcords[idx][2], power))
```

Implementace „Quinn 95 with CIT“

Tato implementace vychází z ostatních implementací, je však vylepšena o parametr *power*, který je vypočítáván z podílu přispívající plochy buňce a prahu trvale tekoucích vod. Dále je umocněn parametrem h pro případné korekce preference nejvyššího sklonu. Implementován je následovně.

```
power = pow ( ( (self.aflow[row][column]/self.CIT)+1), self.paramh)
tempsolution = contouredsection* (pow (subcords[idx][2], power))
```

Finální výpočet

Jak jsou připraveny všechny mezivýpočty pro rozdělení přispívající plochy, ověří se, zda proměnná reprezentující čitatel ve zlomku vzorci (3.4.1.1) není rovná nule. Pokud by tomu tak bylo, bude nastavena na počet sousedních buněk, do kterých existuje odtok. Následně se procházejí tyto sousední buňky a přerozděluje se jim přispívající plocha zdrojové buňky.

4.4 Implementace „Topographic Wetness Index“

Tato implementace je provázaná na výpočet přispívající plochy, proto je součástí scriptu pro výpočet přispívající plochy. Je implementován:

```
tcitemp = (self.aflow[row][column]+1) / (tcitopsum / tempsolutionsum)
```

kde obsahem listu *self.aflow* je přispívající plocha do dané buňky, *tcitopsum* je suma sklonů mezi buňkou zdrojovou a sousedními buňkami, do kterých zdrojová buňka přispívá, vynásobenou jmenovatelem pro výpočet odtoku do dané sousední buňky, *tempsolutionsum* je čitatelem pro výpočet odtoku do dané sousední buňky. V případě, že buňka nemá odtok, je vzorec zjednodušen:

```
tcitemp = (self.aflow[row][column] + 1)/0.001
```

Nakonec je aplikován přirozený logaritmus a výsledek je uložen do pole:

```
self.tci[row][column] = math.log (tcitemp)
```

4.5 Uložení výstupů

Uložení výstupů se již realizuje podobně u všech implementací. Výstupní matice jsou pomocí *NumpyArrayToRaster*, z modulu *numpy*, převedeny na rastry a uloženy pomocí funkce *save*. Pokud toto selže, je postup opakován. Nastavení těchto rastrů je vždy odvozeno z nastavení vstupního výškového rastru. V případě, že výstupní rastr existuje, je přepsán.

Následně jsou vymazány dočasné soubory a složky. Vypíše se zpráva o zpracování do ArcGIS console a skript se ukončí.

5 TESTOVÁNÍ IMPLEMENTACE

Tato kapitola se zabývá testováním algoritmu, které bylo rozděleno na čtyři části. První částí bylo testování jednotlivých částí implementace, následovně se dělaly testy poloautomatické, automatické a testování různých prostředí.

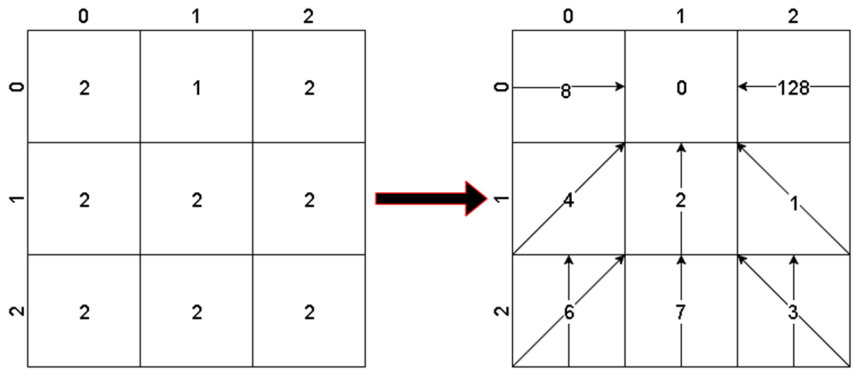
5.1 Testování částí implementace

Nejdříve byla napsána část pro výpočet algoritmu se statickými proměnnými obalená funkcí *try*, díky které se při chybě spustil blok kódu ve funkci *except*, která umožňuje zachycení a vypsání chyby, za pomocí funkce *exc_info* z modulu *sys*, která vrátí tři objekty *exception type*, *exception object* a *exception traceback*. Díky poslednímu z nich může proběhnout dohledání místa chyby, obsahuje řádek chyby a vnořené objekty pro následný backtracking chyby napříč funkcemi.

Jakmile se tento výpočet dostal do stavu, ve kterém po několika spuštěních nenastala výjimka a kontrola správnosti výstupu na testovacích plochách proběhla úspěšně, byl kód optimalizován. Následně byla přidána další část implementace a postup se opakoval, dokud nebyl kód kompletní. Následně probíhala optimalizace a refaktoring jednotlivých částí tak, aby byla co nejmenší časová a paměťová složitost, a to tak, že byl, za pomocí funkce time z modulu time, měřen čas jednotlivých částí kódu před a po optimalizaci. Pokud optimalizace přinesla časovou nebo paměťovou úsporu, aniž se změnil výstup, byla zahrnuta do nové výchozí verze implementace.

5.2 Poloautomatické testy

Při těchto testech bylo zjišťováno chování a stavy proměnných v jednotlivých krocích. Vyhodnocuje se manuálně při pozastavení běhu algoritmu, a to podle přepočítaných hodnot z matematického software Wolfram Matematica nebo přepočítaných testovacích ploch pro daný krok. Tímto způsobem bylo zjišťováno nebo ověřováno, v které části algoritmu dochází k chybě. Po opravě je testování spuštěno znovu, aby se zjistilo, zda je chyba opravena, pokud ano, pokračuje se dále v testu, pokud ne, je nutné kód opět debugovat krok po kroku pro zjištění místa chyby. Toto se opakuje, dokud chyba není opravena. Například tato testovací plocha musí vždy vyjít způsobem, který je přepočítán, pokud tomu tak není, algoritmus není správně a je nutné najít a opravit chybu v algoritmu.



Obrázek 5.1: Testovací plocha a její přepočítaný výsledek

5.3 Automatické testy

Tyto testy se používaly pro vyhodnocení efektivity implementace a rychlosti zpracování dat. Jednalo se o měření časového a paměťového zatížení při rozdílném nastavení. Veškeré tyto testy byly prováděny na notebooku Acer Aspire 5552G-N954G75MNrr s procesorem AMD Phenom II Quad-Core N950, vyrovnávací pamětí 4 GB a operačním systémem Windows 7 Home Premium 64-bit. Testy byly prováděny nad výřezy z DMR 4G o určitém počtu pixelů:

Počet pixelů	Čas s multiprocessingem	Čas bez multiprocessingu
100 px	15.97 s	5.97 s
500 px	19.04 s	8.82 s
10000 px	34.45 s	25.28 s
50000 px	1 m 4.69 s	1 m 5.97 s
1000000 px	32 m 14.93 s	46 m 29.34 s
5000000 px	1 h 22 m 13.81 s	3 h 16 m 35.61 s*
10000000 px	2 h 55 m 14.51 s	16 h+*

Tabulka 1: Porovnání multiprocessingu a běžného jednoprocesového zpracování

Z těchto testů bylo odvozeno, že u DMR 4G je nutná hloubka rekurze pro úplný a správný výpočet pouze 3, ale tato hodnota se mění v závislosti na členitosti terénu a použitém vzorkování.

5.4 Testování prostředí

Testování prostředí probíhalo za pomoci skriptu, s uživatelským prostředím pro ArcGIS, který byl upraven tak, aby posílal informace o průběhu, nastavení skriptu, případných chybách a sestavě počítače, kde byl spuštěn. Reporty o chybách byly opraveny a postupně zapracovány do finálního kódu. Pokud byl proveden bez chyb, zaznamenává se počet zpracovaných pixelů za sekundu na dané sestavě. Ve finále takto proběhlo 1034 spuštění z 634 spuštění bez chyby na verzi takto upraveného skriptu na 363 unikátních počítačích, z toho 161 unikátních sestav. Tyto sestavy byly agregovány dle množství vyrovnávací paměti, počtu jader a frekvence procesoru. V následující tabulce jsou uvedeny sestavy, u nichž proběhlo úspěšné spuštění alespoň 3× a měly defaultní nastavení toolboxu:

Vyrovnávací paměť	Jader procesoru	Frekvence procesoru	Průměrný počet zpracovaných pixelů za sekundu
3072 GB	144*,**	3,3 GHz	29692,3
8 GB	4*	4,2 GHz	831,2
4 GB	8	3,3 GHz	803,1
4 GB	4	4,0 GHz	767,1
4 GB	8	3,2 GHz	709,4
8 GB	6	3,5 GHz	697,3
6 GB	4	4,2 GHz	689,6
2 GB	2*	3,2 GHz	411,3

Tabulka 2 : Přehled testovaných sestav pro algoritmus určení směru odtoku

* HyperThreading

** 8 processorů o 18-ti jádrech (Intel® Xeon® Processor E7-8867 v4)

Z těchto výsledků se dá usuzovat, že výkon výpočtů směru odtoku je ovlivněn hlavně výkonem procesoru a velikost vyrovnávací paměti, kam se ukládají mezivýsledky. Tyto výsledky vedly k další optimalizaci kódu a následkem těchto optimalizací se algoritmus určení směru odtoku z průměrných 761,1 pixelu za sekundu dostal na cca 69390 za sekundu, a to hlavně díky optimalizacím datových typů, redukcí počtu proměnných a přidání společných připočítaných proměnných.

6 UŽIVATELSKÉ PROSTŘEDÍ A PARAMETRY

Uživatelské prostředí se odvíjí od možností softwaru ArcGIS, který umožnuje několik desítek typů vstupů. Tyto vstupy se převádějí na parametry za pomoci rodiny funkcí get parameter z modulu arcpy.

Nastavení a validace parametrů skriptu

V software ArcGIS je umožněno nastavení a validace parametrů pomocí Python scriptu. Pomocí funkce updateParameters je umožněno například vypínání volitelných parametrů.

```
self.params[5].enabled = False
```

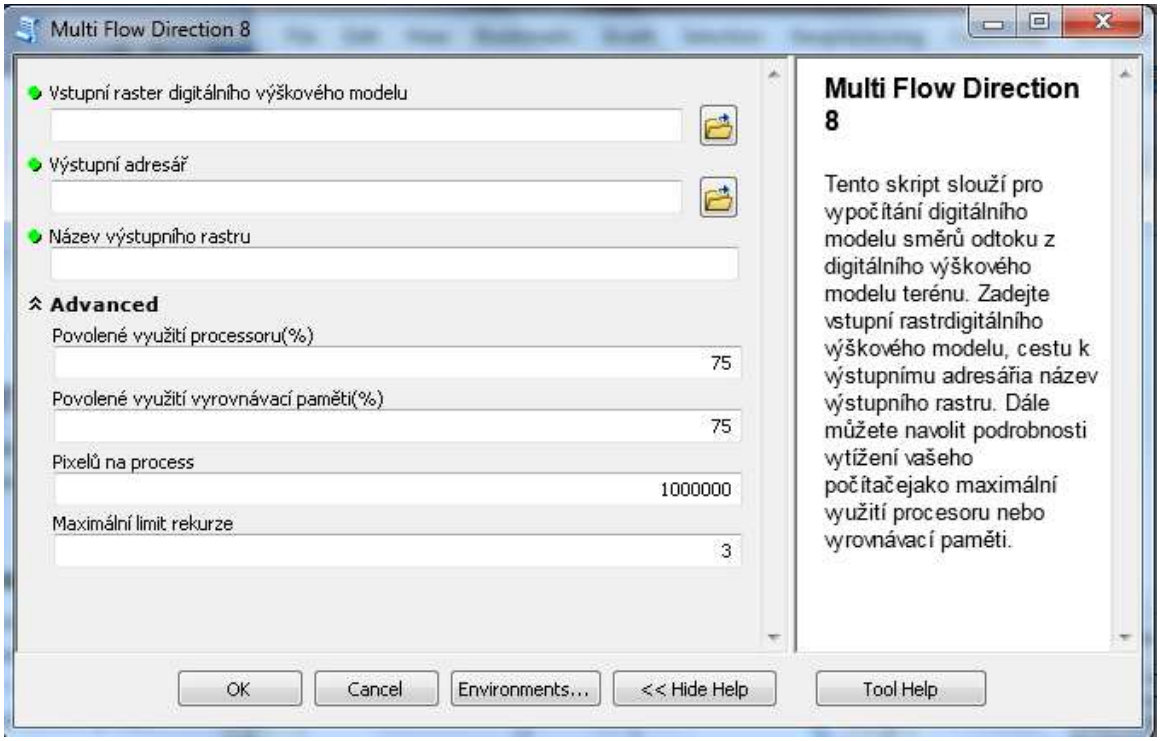
Umožnuje také kategorizaci parametrů:

```
self.params[3].category = "Advanced"
```

Případně jejich validaci umístěnou do funkce updateMessages.

6.1 Prostředí skriptu určení směrů vícesměrného odtoku

Toto prostředí je řešené celkem jednoduše. Důvodem je nízký počet vstupních parametrů, jimiž jsou pouze rastr digitálního výškového modelu, adresa výstupního adresáře a název výstupu. Následně byly rozšířeny o pokročilé možnosti pro management systémových zdrojů. Tyto parametry jsou povolené využití procesoru a vyrovnávací paměti, počet pixelů na jeden proces a maximální hloubka rekurze.

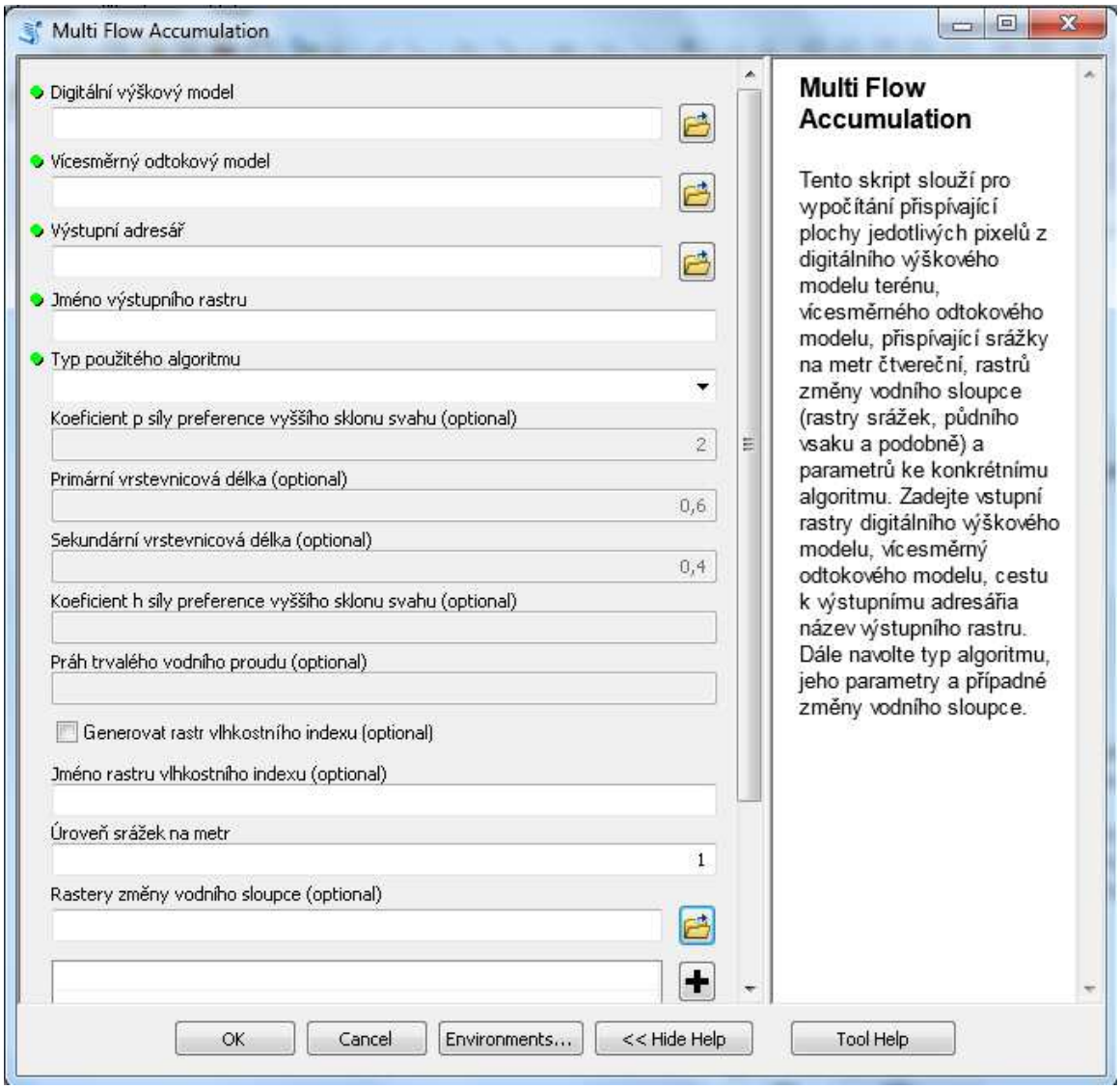


Obrázek 6.1: Prostředí skriptu pro určení vícesměrného odtoku.

6.2 Prostředí skriptu pro určení přispívající plochy

Toto prostředí je již o něco složitější, a to hlavně proto, že vnitřně zahrnuje 4 různé postupy výpočtu přispívající plochy a s tím i 4 různé postupy pro výpočet vlhkostního indexu. Hlavními parametry zde jsou výškový model, vícesměrný odtokový model, výstupní adresář, název výstupního rastru a typ algoritmu pro výpočet. Dále jsou zde parametry jednotlivých algoritmů jako například parametr p, který určuje preferenci vyššího svahu, nebo parametry primární a sekundární vrstevnicové vzdálenosti, které určují poměr mezi primárními a sekundárními směry odtoku. Tyto parametry jsou buď použity zvlášť (Freeman 91, Quinn 91) nebo v kombinaci (Quinn 95). Případně může být parametr p dopočítáván z parametru h, který určuje preferenci vyššího svahu, a parametru CIT, který určuje práh trvalého vodního toku.

Následně u každého z těchto algoritmů je možnost zadat homogenní změnu srážek na jeden metr a to pozitivní hodnoty pro nárůst vodního sloupce a negativní pro jeho snížení. Případně jdou přidat rastry obsahující nehomogenní změny vodního sloupce. Například lze přidat rastry srážek, kanalizačních odtoků a podobně.



Obrázek 6.2: Prostředí skriptu pro výpočet přispívající plochy.

7 DISKUSE

Základem pro zjištění aktuálního stavu teorie a implementací algoritmů vícesměrných odtoků bylo nastudování odborné literatury, odborných prací, existujících implementací a odborných článků zabývajících se algoritmy pro výpočet odtoků a terénními analýzami. Na teoretické i praktické úrovni je již poměrně velké zastoupení prací, které poskytují dostatečné množství informací pro konkrétní implementace. Některé z těchto prací byly použity v kapitole 3 zabývající se současným stavem problematiky. Následně byly vybrány pro implementaci konkrétní algoritmy pro výpočet vícesměrného odtoku a přispívající plochy.

V praktické části této práce byl navržen za pomoci diagramů běh skriptů, který byl dále rozdělen na menší části. Ty byly naprogramovány, testovány, a pokud se zdály být optimální, spojovaly se do větších celků a následně do výsledného skriptu. Výsledný skript byl nakonec testován skupinou uživatelů. V tomto testování se zjišťovaly problémové části implementace, ve kterých nastávaly problémy s výkonem, a chování v různých hardwarových prostředích. Díky těmto testům byl kód optimalizován. Zároveň s optimalizací proběhl refaktoring. Nakonec bylo vytvořeno uživatelské prostředí a nápověda pro ArcGIS.

Při implementaci se vyskytlo několik problémů. Jedním z největších problémů bylo to, zda řešit odtok bezodtoké buňky se sousedními buňkami stejné nadmořské výšky. Tento problém byl nakonec vyřešen rekurzivní funkcí, která vyhledává nejbližší možný odtok a přiděluje směr odtoku do sousedních buněk, které mají tento nejkratší možný odtok. Tato část je ovšem volitelná, jelikož původní návrhy autorů s tímto stavem nepočítaly. U implementace je možné nastavit maximální vzdálenost, do které je odtok počítán. U dat DMR 4G je optimální vzdálenost tří buněk, za touto vzdáleností se nadále výsledek téměř nemění. S tímto řešením souvisí i další problémy, jako byl extrémní výpočetní čas, který byl po řadě optimalizací snížen za použití více procesů. Problém u algoritmu výpočtu přispívající plochy je v tom, že do již zpracované buňky odtékala voda. Protože zde není rozdíl ve výšce, iterační kaskáda fungovala nedostatečně, tudíž muselo být přidáno řazení podle vzájemné závislosti buněk a tím se tomuto jevu zamezilo. Tím vzniká rozšířená iterační kaskáda.

Implementace je do budoucna možné využít jako základu pro implementaci nástrojů k extrakci vodních toků, modelování odtoku srážek, analýze a separaci složek odtokového procesu, návrhů protipovodňových nebo protierozních zábran.

Dalším možným krokem u podobných terénních analýz by bylo využití výpočetního výkonu pomocí grafických karet za pomoci technologií OpenCL, FireStream od AMD nebo CUDA výrobce nVidia. Urychlení grafickou kartou by vzhledem k počtu jader, potažmo operací, bylo například 14násobné u grafické karty Nvidia Geforce 9800GT GPU oproti procesoru Intel Core 2 Quad Q8200, i když je takovéto urychlení u každé sestavy individuální.

Vzhledem k tomu, že je v dnešní době již většina osobních počítačů grafickou kartou vybavena. Tato urychlení výpočtu se budou pravděpodobně rozvíjet nejen u terénních analýz, ale v celé škále oborů. Důkazem tohoto je implementace Viewshed 2 od ESRI v ArcGIS 10.3, která již spolupracuje s nástroji CUDA compute capability 2.0 nebo OpenCL 1.2. Do budoucna tato urychlení umožní zpracovávat data okamžitě.

8 ZÁVĚR

V průběhu práce byl úspěšně dosažen hlavní cíl, a to navrhnout, vytvořit, otestovat a na základě testů optimalizovat implementaci algoritmů pro modelování vícesměrného odtoku a akumulace plochy do prostředí ArcGIS. Výsledkem bakalářské práce je tedy toolbox obsahující skripty napsané v Pythonu, které využívají modulu arcpy a jeho podmodulu Spatial Analyst (více v kapitolách 4, 5 a 6). První skript je určen ke zjištění směru odtoku a druhý skript se využívá ke zjištění přispívající plochy. Toto je umožněno 4 různými algoritmy. U každého z těchto algoritmů je možné zapnout vypočet indexu vlhkosti, který je závislý na typu určení přispívající plochy.

Text práce v rámci rešerše uvádí do problematiky, zmiňuje se o dosavadních návrzích a implementacích jednotlivých algoritmů. Popisuje softwarové a hardwarové vybavení využité k dosažení cíle a problematiku rozšíření ArcGIS. Dále je pak popsán postup návrhu algoritmu a samotná implementace algoritmu. Poté je popsáno testování, které bylo rozděleno do několika fází. Nakonec je popsán návrh uživatelského prostředí a s tím spojené volitelné parametry skriptů.

V praktické části bylo třeba navrhnout jednotlivé části skriptu tak, aby byly co nejefektivnější. Následovně se provedla implementace výpočtu se statickými daty, u které byl kladen důraz především na přesnost výpočtů. Tato část zůstala nezměněna až do finálního skriptu. Následně byly přidávány dynamické prvky jako načtení rastrů, jejich převedení do formy matice a následné převedení zpět a uložení výsledků do rastru na konci skriptu. Poté byl skript obalen testovacím rozhraním, bylo přidáno uživatelské rozhraní. Ve finále bylo odstraněno testovací rozhraní a na základě výsledků testů proběhla optimalizace, refaktoring a bylo upraveno uživatelské rozhraní do konečné podoby.

Samotný skript je pak použitelný v hydrologickém modelování, a to buď jako základ komplexnějších analýz, jako je modelování povodní, nebo k jednodušším analýzám, jako je extrakce vodních toků, určení vlhkostních indexů nebo jiných charakteristik. Také lze tyto implementace využít jako základ pro modelování eroze a mělkých sesuvů, případně jako součást nástroje pro jejich analýzu a predikci.

Byl sestaven přehled teoretických pojmů, algoritmů a existujících implementací těchto algoritmů.

Výsledky této práce mohou být použity v dalších diplomových pracích jako základ pro implementace nástrojů hydrologické či terénní analýzy. Případně jako příklad pro implementaci nástrojů k analýze rastrových dat.

POUŽITÁ LITERATURA A INFORMAČNÍ ZDROJE

ARGE, L., J. S. CHASE, P. HALPIN, L. TOMA, J. S. VITTER, D. URBAN a R. WICKREMESINGHE. Efficient Flow Computation on Massive Grid. *GeoInformatica*. 2001, **7**(4), 283-313. DOI: 10.1023/A:1025526421410. ISSN 13846175. Dostupné také z: http://link.springer.com/10.1023/A:1025526421410

COSTA-CABRAL, M.C. a S. J. BURGES. Digital Elevation Model Networks (DEMON): A model of flow over hillslopes for computation of contributing and dispersal areas. *Water Resources Research*. 1994, **30**(6), 1681-1692. DOI: 10.1029/93WR03512. ISSN 00431397. Dostupné také z: http://doi.wiley.com/10.1029/93WR03512

FAIRFIELD, J. a P. LEYMARIE. Drainage network from grid digital elevation models. *Water Resources Research*. 1991, **27**(5), 709-717.

FREEMAN, T. GRAHAM. Calculating catchment area with divergent flow based on a regular grid. *Computers & Geosciences*. Perpmon Press, 1991, **1991**(17), 413-422.

KONEČNÝ, D. *Srovnání SFD/MFD algoritmů a jejich využití při modelování geomorfologických procesů*[online]. Olomouc, 2006 [cit. 2016-08-08]. Univerzity Palackého.

MARK, D.M. Automated detection of drainage networks from digital elevation models. *Cartographica*. 1984, **21**(2-3), 168-178.

MARK, D.M. a J.F. O'CALLAGHAN. The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics, and Image Processing*. 1984, **28**(3), 323-344.

MARTZ, L. W. a J. GARBRECHT. Automated recognition of valley lines and drainage networks from grid digital elevation models: a review and a new method – Comment. *Journal of Hydrology*. 1995,**167**(1-4), 393-396.

PAN, F., Ch. D. PETERS-LIDARD, M. J. SALE a A. W. KING. A comparison of geographical information systems-based algorithms for computing the TOPMODEL topographic index. *Water Resources Research*. 2004, **40**(6). DOI: 10.1029/2004WR003069. ISSN 00431397. Dostupné také z: http://doi.wiley.com/10.1029/2004WR003069

PILESJÖ, P., Q. ZHOU a L. HARRIE. Estimating flow distribution over digital elevation models using a form-based algorithm. *Annals of GIS*. 1998, **4**(1-2), 44-51. DOI: 10.1080/10824009809480502. ISSN 1947-5683. Dostupné také z: http://www.tandfonline.com/doi/abs/10.1080/10824009809480502

QUINN, P., K. BEVEN, P. CHEVALLIER a O. PLANCHON. The prediction of hillslope flow paths for distributed hydrological modelling using digital terrain models. *HYDROLOGICAL PROCESSES*. 1991, **5**(1), 59-79.

QUINN, P. F., K. J. BEVEN a R. LAMB. The ln(a/tanβ) index: how to calculate it and how to use it within the topmodel framework. *HYDROLOGICAL PROCESSES*. 1995, **9**(1), 161-182.

SEIBERT, J. a B. L. MCGLYNN. A new triangular multiple flow direction algorithm for computing upslope areas from gridded digital elevation models. *Water Resources Research*. 2007, **43**(4). DOI: 10.1029/2006WR005128. ISSN 00431397. Dostupné také z: http://doi.wiley.com/10.1029/2006WR005128

TARBOTON, D.G. A new method for determination of flow directions and upslope areas in grid digital elevation models.*Water Resources Research*. 1997, **33**(2), 309-317.

TRIBE, A. Automated recognition of valley lines and drainage networks from grid digital elevation models: a review and a new method. *Journal of Hydrology*. 1992, **139**(1-4), 263-293.

WOLOCK, D. M. a G. J. MCCABE. Comparison of Single and Multiple Flow Direction Algorithms for Computing Topographic Parameters in TOPMODEL. *Water Resources Research*. 1995, **31**(5), 1315-1324. DOI: 10.1029/95WR00471. ISSN 00431397. Dostupné také z: http://doi.wiley.com/10.1029/95WR00471

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

PŘÍLOHY

SEZNAM PŘÍLOH

Vázané přílohy

Příloha 1a Programový kód algoritmu pro výpočet směru odtoku

Příloha 1b Programový kód validace pro algoritmus výpočtu směru odtoku

Příloha 2a Programový kód algoritmu pro výpočet přispívající plochy

Příloha 2b Programový kód validace pro algoritmus výpočtu přispívající plochy

Volné přílohy

Příloha 1 DVD s elektronickými přílohami

Příloha 2 Poster

Elektronické přílohy

Příloha 1 Adresář obsahující programové kódy

a) Programový kód algoritmu pro výpočet směru odtoku
b) Programový kód validace pro algoritmus výpočtu směru odtoku
c) Programový kód algoritmu pro výpočet přispívající plochy
d) Programový kód validace pro algoritmus výpočtu přispívající plochy

Příloha 2 Adresář obsahující web k prezentaci bakalářské práce

Příloha 3 Toolbox obsahující skripty

Příloha 4 Text bakalářské práce ve formátu PDF

Příloha 5 Poster ve formátu PDF

Příloha 1a Programový kód algoritmu pro výpočet směru odtoku

```
try:
    from shutil import rmtree
    from numpy import tile, array as narray, concatenate, asarray,
int32, int16
    from ctypes import Structure, c_ulong, c_ulonglong, sizeof, windll,
byref
    from time import sleep, time
    from arcpy import env, Raster, Describe, Point, RasterToNumPyArray,
NumPyArrayToRaster, AddMessage
    from arcpy import GetParameterAsText, Delete_management, Exists,
AddError
    from os import makedirs, path, popen, environ
    from sys import exc_info
    from multiprocessing import Process
    from math import floor, ceil
except Exception, e:
    print "Chyba importu metody nektere knihovny:"
    print e.message

__author__ = 'Hittl Roman'
SLEEP_TIME = 0.5
MAXRECURSIONLIMIT = 10
PROGRESSPRINT = True
ERRORPRINT = True
SUMMARYPRINT = True


class MEMORYSTATUSEX(Structure):
    _fields_ = [
        ("dwLength", c_ulong),
        ("dwMemoryLoad", c_ulong),
        ("ullTotalPhys", c_ulonglong),
        ("ullAvailPhys", c_ulonglong),
        ("ullTotalPageFile", c_ulonglong),
        ("ullAvailPageFile", c_ulonglong),
        ("ullTotalVirtual", c_ulonglong),
        ("ullAvailVirtual", c_ulonglong),
        ("sullAvailExtendedVirtual", c_ulonglong),
    ]

    def __init__(self):
        self.dwLength = sizeof(self)
        super(MEMORYSTATUSEX, self).__init__()


def get_free_memory():
    stat = MEMORYSTATUSEX()
    windll.kernel32.GlobalMemoryStatusEx(byref(stat))
    return 100 - stat.dwMemoryLoad


def get_free_processor():
    try:
        processor = popen('wmic cpu get loadpercentage')
        result = processor.read()
        processor.close()
        total_cpu = 0
        for m in result.split("  \r\n")[1:-1]:
            total_cpu += int(m)
```

```
    except Exception:
        total_cpu = 1
    return 100 - total_cpu



def delete_folder(dirpath):
    if path.exists(dirpath):
        rmtree(dirpath, ignore_errors=True)



def create_folder(dirpath):
    delete_folder(dirpath)
    if not path.exists(dirpath):
        makedirs(dirpath)



def delete_file(rasterpath):
    while Exists(rasterpath):
        try:
            Delete_management(rasterpath)
        except Exception, e:
            if PROGRESSPRINT:
                print "Soubor '", rasterpath, "' nelze smazat je otevren
jinou aplikaci."



def line_worker(start, stop, rwidth, rheight, TEMP_PATH, defaultraster,
RECURSELIMIT):
    try:
        settemp(start)
        inraster = Raster(defaultraster)
        nodata = inraster.noDataValue
        dsc = Describe(inraster)
        env.outCoordinateSystem = dsc.SpatialReference
        ext = dsc.Extent
        referencepoint = Point(ext.XMin, ext.YMin)
        rarray = RasterToNumPyArray(inraster)

        argstop = stop
        if stop > rheight:
            stop = rheight
        noflow = -1
        drarray = tile(noflow, (stop - start, rwidth))
        for i in range(start, stop):
            for j in range(0, rwidth):
                defaultcordX = i
                defaultcordY = j
                if rarray.item(defaultcordX, defaultcordY) != nodata:
                    subcords, recursedepth = getflowsubcords(rarray,
defaultcordX, defaultcordY, rheight, rwidth,
                                                             nodata,
recurselimit=RECURSELIMIT)
                    data = 0
                    if not subcords:
                        pass
```

```
                    else:
                        for subcord in subcords:
                            if subcord[0] == defaultcordX:
                                if subcord[1] > defaultcordY:
                                    data += 8
                                elif subcord[1] < defaultcordY:
                                    data += 128

                            elif subcord[0] > defaultcordX:
                                if subcord[1] == defaultcordY:
                                    data += 32
                                elif subcord[1] > defaultcordY:
                                    data += 16
                                elif subcord[1] < defaultcordY:
                                    data += 64

                            elif subcord[0] < defaultcordX:
                                if subcord[1] == defaultcordY:
                                    data += 2
                                elif subcord[1] > defaultcordY:
                                    data += 4
                                elif subcord[1] < defaultcordY:
                                    data += 1
                    else:
                        data = noflow
                    drarray.itemset((defaultcordX - start, defaultcordY),
data)
            finish = 0
            while not finish:
                try:
                    new_raster = NumPyArrayToRaster(drarray, referencepoint,
dsc.meanCellWidth, dsc.meanCellHeight,
                                                    value_to_nodata=noflow)
                    partpath = ''.join((TEMP_PATH, str(start), "_",
str(argstop)))
                    delete_file(partpath)
                    new_raster.save(partpath)
                    finish = 1

                except Exception, e:
                    if ERRORPRINT:
                        print "*-*", e.message
                    delete_folder(environ["TEMP"])
                    exit(3)

        except Exception, e:
            exc_type, exc_obj, exc_tb = exc_info()
            if ERRORPRINT:
                print "Chyba zpracovani na radku", exc_tb.line_no, ": ", e
            delete_folder(environ["TEMP"])
            exit(3)

        delete_folder(environ["TEMP"])
        exit(0)


    def settemp(id):
        newtemppath = path.join(environ["TEMP"], str(id))
        environ["TEMP"] = newtemppath
        environ["TMP"] = newtemppath
        env.workspace = newtemppath
```

```
    env.scratchWorkspace = newtemppath
    create_folder(newtemppath)


def getflowsubcords(rarray, defaultcordX, defaultcordY, rheight, rwidth,
noData, recurselevel=0, bannedcords=[],
                    recurselimit=10):
    subcords = [[defaultcordX + 0, defaultcordY - 1], [defaultcordX - 1,
defaultcordY + 0],
                [defaultcordX + 0, defaultcordY + 1], [defaultcordX + 1,
defaultcordY + 0],
                [defaultcordX + 1, defaultcordY + 1], [defaultcordX - 1,
defaultcordY + 1],
                [defaultcordX + 1, defaultcordY - 1], [defaultcordX - 1,
defaultcordY - 1]]
    if defaultcordX != 0 and defaultcordY != 0 and defaultcordX + 1 <=
rheight - 1 and defaultcordY + 1 <= rwidth - 1:
        pass
    else:
        copy = []

        if defaultcordX == 0:
            for subcord in subcords:
                if -1 == subcord[0]:
                    pass
                else:
                    copy.append(subcord)
            subcords = copy
            copy = []

        if defaultcordY == 0:
            for subcord in subcords:
                if -1 == subcord[1]:
                    pass
                else:
                    copy.append(subcord)
            subcords = copy
            copy = []

        if defaultcordX == rheight - 1:
            for subcord in subcords:
                if rheight - 1 < subcord[0]:
                    pass
                else:
                    copy.append(subcord)
            subcords = copy
            copy = []

        if defaultcordY == rwidth - 1:
            for subcord in subcords:
                if rwidth - 1 < subcord[1]:
                    pass
                else:
                    copy.append(subcord)
            subcords = copy
            copy = []
    copy = []
    notflowcopy = []
    for subcord in subcords:
        if rarray.item(subcord[0], subcord[1]) != noData:
            elevationDifference = rarray.item(defaultcordX,
defaultcordY) - rarray.item(subcord[0], subcord[1])
```

```
            if elevationDifference > 0:
                subcord.append(elevationDifference)
                copy.append(subcord)
            elif elevationDifference == 0:
                subcord.append(elevationDifference)
                notflowcopy.append(subcord)

    minrecursedepth = None
    if copy:
        if recurselevel != 0:
            np = narray(copy)
            if (defaultcordX in np[:, 0] or defaultcordY in np[:, 1]):
                minrecursedepth = recurselevel + 1
            else:
                minrecursedepth = recurselevel + 1.4142
        else:
            minrecursedepth = 0
    else:
        copy = []
        if notflowcopy:
            for cords in notflowcopy:
                if (defaultcordX == cords[0] or defaultcordY ==
cords[1]):
                    recursedepthin = recurselevel + 1
                else:
                    recursedepthin = recurselevel + 1.4142
                if cords[0:2] not in bannedcords and recurselevel <
recurselimit:
                    bannedcords.extend([[defaultcordX, defaultcordY]])

                    temp, recursedepth = getflowsubcords(rarray,
cords[0], cords[1], rheight, rwidth, noData,
                                                         recursedepthin,
                                                         bannedcords,
recurselimit=recurselimit)
                    del bannedcords[-1]
                    if temp:
                        if minrecursedepth:
                            if recursedepth:
                                if recursedepth * 10000 ==
minrecursedepth * 10000:
                                    copy.append(cords)
                                elif recursedepth * 10000 <
minrecursedepth * 10000:
                                    minrecursedepth = recursedepth
                                    copy = []
                                    copy.append(cords)
                        else:
                            minrecursedepth = recursedepth
                            copy.append(cords)
                        if minrecursedepth < recurselimit:
                            recurselimit = minrecursedepth

    return (copy, minrecursedepth)
```

```
class WaterFlow():
    def __init__(self):
        env.overwriteOutput = True
        self.start = 0
        self.stop = None
        self.readarguments()
        self.baseinit()
        if PROGRESSPRINT:
            print("Inicializace kommpletni.\nZahajeni zpracovani.")

    def readarguments(self):
        self.DEM_in = GetParameterAsText(0)
        if self.DEM_in == '#' or not self.DEM_in:
            raise (Exception("Nebyl vlozen digitalni model terenu"))
        self.DIRECTORY_out = GetParameterAsText(1)
        if self.DIRECTORY_out == '#' or not self.DIRECTORY_out:
            raise (Exception("Nebyl vlozen vystup"))
        self.FILE_out = GetParameterAsText(2)
        if self.FILE_out == '#' or not self.FILE_out:
            raise (Exception("Nebyl vlozen vystup"))

        self.CPU_usage = GetParameterAsText(3)
        if self.CPU_usage == '#' or not self.CPU_usage:
            raise (Exception("Nebylo vlozeno vyuziti processoru."))

        self.RAM_usage = GetParameterAsText(4)
        if self.RAM_usage == '#' or not self.RAM_usage:
            raise (Exception("Nebylo vlozeno vyuziti vyrovnavaci
pameti."))

        self.PIXEL_LIMIT = GetParameterAsText(5)
        if self.PIXEL_LIMIT == '#' or not self.PIXEL_LIMIT:
            raise (Exception("Nebyl vlozen pocet pixelu pro deleni
procesu."))
        else:
            try:
                self.PIXEL_LIMIT = int(self.PIXEL_LIMIT)
            except:
                raise (Exception("Pocet pixelu neni cele cislo."))

        global MAXRECURSIONLIMIT
        MAXRECURSIONLIMIT = GetParameterAsText(6)
        if MAXRECURSIONLIMIT == '#' or not MAXRECURSIONLIMIT:
            raise (Exception("Nebyl vlozen maximalni limit rekurze."))
        else:
            try:
                MAXRECURSIONLIMIT = int(MAXRECURSIONLIMIT)
            except:
                raise (Exception("Pocet rekurzi neni cele cislo."))

        self.constantsfromparametrs()

    def constantsfromparametrs(self):
        self.CPU_reserve = 100 - float(self.CPU_usage)
        self.RAM_reserve = 100 - float(self.RAM_usage)
        self.PATH_OUT = "".join((self.DIRECTORY_out, "\\",
self.FILE_out))
        self.TEMP_DIRECTORY = "".join((self.DIRECTORY_out, "\\temp"))
        create_folder(self.TEMP_DIRECTORY)
        self.TEMP_PATH = "".join((self.TEMP_DIRECTORY, "\\t"))
```

```
    def baseinit(self):
        self.loadraster()
        self.promptinit()


    def loadraster(self):
        inraster = Raster(self.DEM_in)
        self.dsc = Describe(inraster)
        self.rwidth = inraster.width
        self.rheight = inraster.height
        self.ext = self.dsc.Extent
        if hasattr(self, "stop"):
            self.stop = self.rheight
        elif not self.stop:
            self.stop = self.rheight
        YMin = self.ext.YMax - self.stop * inraster.meanCellHeight
        self.referencepoint = Point(self.ext.XMin, YMin)
        del inraster

    def promptinit(self):
        self.progress = self.start
        self.processorDiff = 100
        self.ramDiff = 100
        self.processList = []
        self.tempsList = []
        self.pixels = 0
        self.limitRow = 1
        self.allpixels = self.rheight * self.rwidth
        self.procenta = 0
        while self.pixels < self.PIXEL_LIMIT and self.pixels <
self.allpixels:
            self.pixels = self.rwidth * self.limitRow
            self.limitRow += 1

    def waitforall(self):
        while self.processList:
            self.processcheck()
            sleep(SLEEP_TIME)

    def processcheck(self):
        for idx, process in enumerate(self.processList):
            if not process.is_alive():
                if process.exitcode <= 1:
                    self.endprocess(idx)
                    if PROGRESSPRINT:
                        print(idx)
                        self.printprogress()
                elif process.exitcode == 3:
                    if ERRORPRINT:
                        print "File lock " +
str(self.processList[idx]._args[0])
                    self.resetprocess(idx=idx)
                else:
                    self.resetprocess(idx=idx)

    def startprocessing(self):
        self.processing()
        if PROGRESSPRINT:
            print("Cekam na dokonceni.")
        self.waitforall()
        if PROGRESSPRINT:
            print("Kompletace vystupniho rastru.")
```

```
        self.joinallrasters()
        if PROGRESSPRINT:
            print("Ulozeni vystupniho rastru.")
        self.saveresults()
        if PROGRESSPRINT:
            print("Uklid docasnych slozek.")
        self.clean()
        if SUMMARYPRINT:
            self.printsummary()

    def processing(self):
        while self.progress < self.stop:
            self.getresourcediff()
            while not self.started:
                if not self.processList:
                    self.startnewprocess()
                else:
                    self.processcheck()

                while self.resourcecheck():
                    self.startnewprocess()
                    sleep(SLEEP_TIME)
                sleep(SLEEP_TIME)
                self.rasterjoinprepare()

    def getresourcediff(self):
        if self.progress - self.start != 0:
            self.processorDiff = self.processorBefore -
get_free_processor()
            if self.processorDiff < 5:
                self.processorDiff = 5
            self.memoryDiff = self.memoryBefore - get_free_memory()
            if self.memoryDiff < 5:
                self.memoryDiff = 5
        self.started = False

    def startnewprocess(self):
        self.processorBefore = get_free_processor()
        self.memoryBefore = get_free_memory()
        if self.progress < self.stop:
            self.processList.append(Process(target=line_worker, args=(
                self.progress, self.progress + self.limitRow,
self.rwidth, self.rheight, self.TEMP_PATH, self.DEM_in,
                MAXRECURSIONLIMIT)))
            self.processList[-1].start()
        self.progress += self.limitRow
        self.started = True

    def resetprocess(self, idx):
        if self.resourcecheck() or len(self.processList) == 1:
            self.processList[idx] = Process(target=line_worker,
args=self.processList[idx]._args)
            self.processList[idx].start()

    def endprocess(self, idx):
        self.tempsList.append([self.processList[idx]._args[0],
self.processList[idx]._args[1]])
        self.processList[idx].join()
        del self.processList[idx]
```

```
    def resourcecheck(self):
        return get_free_processor() - self.CPU_reserve >
self.processorDiff and get_free_memory() - self.RAM_reserve >
self.memoryDiff

    def printprogress(self):
        procenta = (self.progress - self.start) / (float(self.stop -
self.start) / 100)
        if self.progress != 0 and self.progress <= self.rheight and
procenta >= self.procenta + 1:
            cas = time() - timestart
            self.procenta = ceil(procenta)
            uptime = cas
            uhodiny = uptime // 3600
            uminuty = (uptime - (uhodiny * 3600)) // 60
            usekundy = floor((uptime - (uhodiny * 3600) - (uminuty *
60)))
            downtime = (uptime / procenta) * (100 - procenta)
            dhodiny = downtime // 3600
            dminuty = (downtime - (dhodiny * 3600)) // 60
            dsekundy = floor((downtime - (dhodiny * 3600) - (dminuty *
60)))
            print "".join((
                "Zpracovano ", str(int(self.procenta)), " % za ",
str(int(uhodiny)), "h ", str(int(uminuty)),
                "m ", str(int(usekundy)), "s", " jeste zbyva ",
str(int(dhodiny)), "h ", str(int(dminuty)),
                "m ", str(int(dsekundy)), "s", ". Aktualni pocet
procesu: ", str(len(self.processList) + 1)))

    def rasterjoinprepare(self, end=False):
        if len(self.tempsList) > 5 or end:
            self.tempsList = sorted(self.tempsList, reverse=False,
key=lambda pair: pair[0])
        for idx, tempitem in enumerate(self.tempsList):
            if idx < len(self.tempsList) - 1:
                ftempindexstart = tempitem[0]
                ftempindexend = tempitem[1]
                ftemprasterpath = ''.join((self.TEMP_PATH,
str(ftempindexstart), "_", str(ftempindexend)))

                stempindexstart = self.tempsList[idx + 1][0]
                stempindexend = self.tempsList[idx + 1][1]
                stemprasterpath = ''.join((self.TEMP_PATH,
str(stempindexstart), "_", str(stempindexend)))

                if ftempindexend == stempindexstart:
                    newrasterpath = ''.join((self.TEMP_PATH,
str(ftempindexstart), "_", str(stempindexend)))
                    joinstatus = self.rasterjoin(ftemprasterpath,
stemprasterpath, newrasterpath,
                                                 [ftempindexstart,
ftempindexend, stempindexstart, stempindexend])
                    if joinstatus:
                        del self.tempsList[idx + 1], self.tempsList[idx]
                elif stempindexend == ftempindexstart:
                    newrasterpath = ''.join((self.TEMP_PATH,
str(stempindexstart), "_", str(ftempindexend)))
                    joinstatus = self.rasterjoin(stemprasterpath,
ftemprasterpath, newrasterpath,
                                                 [stempindexstart,
```

```
stempindexend, ftempindexstart, ftempindexend])
                    if joinstatus:
                        del self.tempsList[idx + 1], self.tempsList[idx]

    def joinallrasters(self):
        while len(self.tempsList) > 1:
            try:
                self.rasterjoinprepare(True)
            except Exception, e:
                if ERRORPRINT:
                    print e.message

    def rasterjoin(self, frasterpath, srasterpath, outrasterpath,
positionsinraster):
        try:
            fraster = Raster(frasterpath)
            farray = RasterToNumPyArray(fraster, nodata_to_value=-1)
            sraster = Raster(srasterpath)
            sarray = RasterToNumPyArray(sraster, nodata_to_value=-1)
            del fraster, sraster
            if farray.dtype != sarray.dtype:
                farray = farray.astype(int16)
                sarray = sarray.astype(int16)
            outArray = concatenate((farray, sarray))
            del farray, sarray
            outUpdateRaster = NumPyArrayToRaster(outArray,
self.referencepoint, self.dsc.meanCellWidth,

self.dsc.meanCellHeight, value_to_nodata=-1)

            outUpdateRaster.save(outrasterpath)
            self.tempsList.append([positionsinraster[0],
positionsinraster[3]])
            del outUpdateRaster, outArray
            delete_file(frasterpath)
            delete_file(srasterpath)
            return True
        except Exception, e:
            if ERRORPRINT:
                print "Chyba spojovani rastru", e.message
            return False

    def saveresults(self):
        outrasterpath = ''.join((self.TEMP_PATH,
str(self.tempsList[0][0]), "_", str(self.tempsList[0][1])))
        outupdateraster = Raster(outrasterpath)
        fullsave = False
        while not fullsave:
            try:
                delete_file(self.PATH_OUT)
                delete_folder(self.PATH_OUT)
                outupdateraster.save(self.PATH_OUT)
                fullsave = True
            except Exception, e:
                sleep(SLEEP_TIME)

    def clean(self):
        delete_folder(self.TEMP_DIRECTORY)
        delete_folder(environ["TEMP"])
```

```
    def printsummary(self):
        cas = time() - timestart
        uptime = cas
        uhodiny = uptime // 3600
        uminuty = (uptime - (uhodiny * 3600)) // 60
        usekundy = (uptime - (uhodiny * 3600) - (uminuty * 60))

        message = "".join((
            str(self.rwidth * self.stop), "px zpracovano za ",
str(int(uhodiny)), "h ", str(int(uminuty)),
            "m ", str(usekundy), "s a recurzi", str(MAXRECURSIONLIMIT),
" pixel limit",
            str(self.PIXEL_LIMIT)))
        AddMessage(message)
        print message


if __name__ == '__main__':
    try:
        if PROGRESSPRINT:
            print("Start inicializace.")
        timestart = time()
        settemp("MultiFlow")
        waterflow = WaterFlow()
        waterflow.startprocessing()
        AddMessage("Uspesne dokonceno")
    except Exception, e:
        exc_type, exc_obj, exc_tb = exc_info()
        AddError(e.message)
```

Příloha 1b Programový kód validace pro algoritmus výpočtu směru odtoku

```
import arcpy
class ToolValidator(object):

  def __init__(self):
    self.params = arcpy.GetParameterInfo()

  def initializeParameters(self):
    self.params[3].category = "Advanced"
    self.params[4].category = "Advanced"
    self.params[5].category = "Advanced"
    return

  def updateParameters(self):
    return

  def updateMessages(self):
    return
```

Příloha 2a Programový kód algoritmu pro výpočet přispívající plochy

```
import math
import os
import numpy
import time
import arcpy
import sys
from arcpy.sa import ExtractByRectangle

__author__ = 'Hittl Roman'

MAXTRY = 10
PIXEL_LIMIT = 10000
SLEEP_TIME = 0.5
PROGRESSPRINT = True
ERRORPRINT = True
SUMMARYPRINT = True


class WaterAccumulation:
    def __init__(self):
        arcpy.env.overwriteOutput = True
        self.readarguments()
        self.baseinit()
        if PROGRESSPRINT:
            arcpy.AddMessage("Inicializace kommpletni.\nZahajeni
zpracovani.")

    def hasinflow(self, row, column, elevation):
        flowlist = []
        if 0 <= row and len(self.fdarray) > row:
            if 0 <= column and len(self.fdarray[row]) > column:
                if 0 <= row - 1:
                    if 0 <= column - 1:
                        if self.rarray[row - 1, column - 1] == elevation
and self.fdarray[row - 1][
                                        column - 1] % 128 % 64 % 32 // 16 ==
1:
                            flowlist.append([row - 1, column - 1])
                    if self.rarray[row - 1, column] == elevation and
self.fdarray[row - 1][
                            column] % 128 % 64 // 32 == 1:
                            flowlist.append([row - 1, column])
                    if len(self.fdarray[row]) > column + 1:
                        if self.rarray[row - 1, column + 1] == elevation
and self.fdarray[row - 1][
                                        column + 1] % 128 % 64 // 64 == 1:
                            flowlist.append([row - 1, column + 1])
                if len(self.fdarray) > row + 1:
                    if len(self.fdarray[row]) > column + 1:
                        if self.rarray[row + 1, column + 1] == elevation
and self.fdarray[row + 1][
                                        column + 1] % 128 % 64 % 32 % 8 % 4
% 2 // 1 == 1:
                            flowlist.append([row + 1, column + 1])
                    if self.rarray[row + 1, column] == elevation and
self.fdarray[row + 1][
                            column] % 128 % 64 % 16 % 8 % 4 // 2 == 1:
                            flowlist.append([row + 1, column])
                    if 0 <= column - 1:
                        if self.rarray[row + 1, column - 1] == elevation
```

```
and self.fdarray[row + 1][
                                        column - 1] % 128 % 32 % 16 % 8 // 4
== 1:
                                flowlist.append([row + 1, column - 1])
                    if len(self.fdarray[row]) > column + 1:
                        if self.rarray[row, column + 1] == elevation and
self.fdarray[row][column + 1] // 128 == 1:
                            flowlist.append([row, column + 1])
                    if 0 <= column - 1:
                        if self.rarray[row, column - 1] == elevation and
self.fdarray[row][
                                    column - 1] % 64 % 32 % 16 // 8 == 1:
                            flowlist.append([row, column - 1])
                    if flowlist:
                        return flowlist
                    else:
                        return False
                return False
            return False

    def baseinit(self):
        self.loadrasters()
        self.promptinit()

    def loadrasters(self):
        self.inraster = arcpy.Raster(self.DEM_in)
        self.nodata = self.inraster.noDataValue
        self.dsc = arcpy.Describe(self.inraster)
        self.ext = self.dsc.Extent
        self.referencepoint = arcpy.Point(self.ext.XMin, self.ext.YMin)

        self.fdraster = arcpy.Raster(self.FLOW_in)
        self.noflow = self.fdraster.noDataValue
        self.rarray = arcpy.RasterToNumPyArray(self.inraster)
        self.rwidth = self.inraster.width
        self.rheight = self.inraster.height
        self.fdarray = arcpy.RasterToNumPyArray(self.fdraster)

        self.pixelsizex = self.inraster.meanCellHeight
        self.pixelsizey = self.inraster.meanCellWidth
        self.pixelsurface = self.pixelsizex * self.pixelsizey

        self.waterchangerasterlist = []
        self.waterperpixel = self.waterpersquaremeter *
self.pixelsurface
        rectExtract =
arcpy.NumPyArrayToRaster(numpy.tile(self.waterperpixel,
self.rarray.shape), self.referencepoint,
                                                self.dsc.meanCellWidth,
self.dsc.meanCellHeight)

self.waterchangerasterlist.append(os.path.join(os.environ["TEMP"],
"".join(("retras"))))
        rectExtract.save(self.waterchangerasterlist[-1])
        if len(self.waterchangerasterpathlist) != 0:
            arcpy.AddError(str(len(self.waterchangerasterpathlist)))
            if arcpy.CheckExtension("Spatial") == "Available":
                arcpy.CheckOutExtension("Spatial")
                del rectExtract
                for idx, raster in
enumerate(self.waterchangerasterpathlist):
```

```
                    if arcpy.Exists(raster):
                        rectExtract = ExtractByRectangle(raster,
self.ext, "INSIDE")
                    else:
                        arcpy.AddError("".join((raster, "
neexistuje.")))
                        self.waterchangerasterlist.append(
                            os.path.join(os.environ["TEMP"],
"".join(("retras" + str(idx)))))
                        rectExtract.save(self.waterchangerasterlist[-1])
                arcpy.AddMessage("CellStatistics")
                self.waterchange =
arcpy.sa.CellStatistics(self.waterchangerasterlist, "SUM", "DATA")
            else:
                arcpy.AddError("Licence pro modul spatial neni dostupna.
Ignoruji rastry zmeny vodniho sloupce.")
                self.waterchange = rectExtract
        else:
            self.waterchange = rectExtract
        self.waterchange.save(os.path.join(self.DIRECTORY_out,
"waterchan"))
        self.waterchange = arcpy.RasterToNumPyArray(self.waterchange)

    def promptinit(self):
        self.noacc = -1
        self.aflow = []
        self.tci = []
        for i in range(len(self.fdarray)):
            self.aflow.append([])
            self.tci.append([])
            for j in range(len(self.fdarray[i])):
                self.aflow[i].append(self.noacc)
                self.tci[i].append(self.noacc)

    def readarguments(self):
        arcpy.AddMessage("Start args")
        self.DEM_in = arcpy.GetParameterAsText(0)
        if self.DEM_in == '#' or not self.DEM_in:
            raise (Exception("Parameter digital elevation raster path
miss."))

        self.FLOW_in = arcpy.GetParameterAsText(1)
        if self.FLOW_in == '#' or not self.FLOW_in:
            raise (Exception("Parameter flow direction raster path
miss."))

        self.DIRECTORY_out = arcpy.GetParameterAsText(2)
        if self.DIRECTORY_out == '#' or not self.DIRECTORY_out:
            raise (Exception("Parameter out folder path miss."))

        self.FILE_out = arcpy.GetParameterAsText(3)
        if self.FILE_out == '#' or not self.FILE_out:
            raise (Exception("Parameter out file name miss."))
        self.TCI_ENABLE = arcpy.GetParameterAsText(10)
        if self.TCI_ENABLE == '#' or not self.TCI_ENABLE:
            self.TCI_ENABLE = False
        if self.TCI_ENABLE:
            self.TCI_NAME_out = arcpy.GetParameterAsText(11)
            if self.TCI_NAME_out == '#' or not self.TCI_NAME_out:
                raise (Exception("Parameter tci file name miss."))
            self.TCI_OUT = "".join((self.DIRECTORY_out, "\\",
```

```
self.TCI_NAME_out))
        self.PATH_OUT = "".join((self.DIRECTORY_out, "\\",
self.FILE_out))

        self.ALGORITMTYPE = arcpy.GetParameterAsText(12)
        if self.ALGORITMTYPE == '#' or not self.ALGORITMTYPE:
            raise (Exception("Parameter algoritm type miss."))
        else:
            self.ALGORITMTYPE = int(self.ALGORITMTYPE)

        if self.ALGORITMTYPE == 1 or self.ALGORITMTYPE == 3:
            self.POWER = arcpy.GetParameterAsText(5)
            if self.POWER == '#' or not self.POWER:
                raise (Exception("Parameter power miss."))
            else:
                self.POWER = float(self.POWER.replace(",", "."))

        if self.ALGORITMTYPE == 2 or self.ALGORITMTYPE == 3 or
self.ALGORITMTYPE == 4:
            self.CONTUREDIRECT = arcpy.GetParameterAsText(6)

            self.CONTUREUNDIRECT = arcpy.GetParameterAsText(7)
            if self.CONTUREDIRECT == '#' or not self.CONTUREDIRECT or
self.CONTUREUNDIRECT == '#' or not self.CONTUREUNDIRECT:
                raise (Exception("Parameter conture length miss."))
            else:
                self.CONTUREDIRECT =
float(self.CONTUREDIRECT.replace(",", "."))
                self.CONTUREUNDIRECT =
float(self.CONTUREUNDIRECT.replace(",", "."))
        else:
            self.CONTUREDIRECT = float(1)
            self.CONTUREUNDIRECT = float(1)

        if self.ALGORITMTYPE == 4:
            self.paramh = arcpy.GetParameterAsText(8)
            self.CIT = arcpy.GetParameterAsText(9)
            if self.paramh == '#' or not self.paramh:
                raise (Exception("Parameter h miss."))
            else:
                self.paramh = float(self.paramh.replace(",", "."))

            if self.CIT == '#' or not self.CIT:
                raise (Exception("Parameter h miss."))
            else:
                self.CIT = float(self.CIT.replace(",", "."))

        self.waterpersquaremeter = arcpy.GetParameterAsText(13)
        if self.waterpersquaremeter == '#' or not
self.waterpersquaremeter:
            raise (Exception("Neni zadan parametr srazka na metr."))
        else:
            self.waterpersquaremeter =
float(self.waterpersquaremeter.replace(",", "."))

        self.waterchangerasterpaths = arcpy.GetParameterAsText(14)
        self.waterchangerasterpathlist = []
        if self.waterchangerasterpaths == '#' or not
self.waterchangerasterpaths:
            pass
        else:
```

```
            for path in self.waterchangerasterpaths.split(';'):
                self.waterchangerasterpathlist.append(path)

    def startprocessing(self):
        self.processing()
        if PROGRESSPRINT:
            arcpy.AddMessage("Cekam na dokonceni.")
        if PROGRESSPRINT:
            arcpy.AddMessage("Ulozeni vystupniho rastru.")
        self.saveresults()
        if SUMMARYPRINT:
            self.printsummary()

    def processing(self):
        self.prepareelevationlist()
        for eidx, elevation in enumerate(self.sorted_elevation):
            self.counter += 1
            erow, ecolumn = numpy.where(self.rarray == elevation)
            if len(erow) > 1:
                waitlist = []
                sortedlist = []
                for idx, i in enumerate(erow):
                    j = ecolumn[idx]
                    inflows = self.hasinflow(i, j, elevation)
                    if inflows is False or inflows in sortedlist:
                        sortedlist.append([i, j])
                    else:
                        waitlist.append([i, j, inflows])

                while waitlist:
                    for idx, part in enumerate(waitlist):
                        inflows = part[2]
                        if inflows is False or all(item in sortedlist
for item in inflows):
                            sortedlist.append([part[0], part[1]])
                            x = len(waitlist)
                            if x > 2:
                                print x
                            del waitlist[idx]
                del waitlist
            else:
                sortedlist = [[erow[0], ecolumn[0]]]

            for row, column in sortedlist:

                value = self.fdarray[row][column]
                if value != self.noflow:
                    self.getflowbase(row, column)

                    if value != 0:
                        subcords = self.getsubcords(row, column, value)
                        if subcords:
                            tempsolutionsum = 0
                            for idx, subcord in enumerate(subcords):
                                if self.rwidth > subcord[1] and
self.rheight > subcord[
                                        0] and self.rwidth > column and
self.rheight > row and 0 <= subcord[1] and 0 <= \
                                         subcord[0] and 0 <= column and 0
<= row:
                                    elevationDifference =
```

```
self.rarray[row, column] - self.rarray[subcord[0]][subcord[1]]
                                    lengthbetwenpixels =
self.getlengthbetwenpixels(row, column, subcord)

subcords[idx].append(elevationDifference / lengthbetwenpixels)

                                    if (row == subcord[0] or column ==
subcord[1]):
                                        contouredsection =
self.CONTUREDIRECT
                                    else:
                                        contouredsection =
self.CONTUREUNDIRECT

                                    tempsolution =
self.gettempsolution(row, column, subcords, idx, contouredsection)

                                    subcords[idx].append(tempsolution)
                                    tempsolutionsum += tempsolution

                                else:
                                    subcords[idx].append(0)
                                    subcords[idx].append(0)
                            tcitopsum = 0
                            arcpy.AddMessage(tempsolutionsum)
                            if tempsolutionsum == 0:
                                tempsolutionsum = len(subcords)
                            for idx, subcord in enumerate(subcords):

                                if self.rwidth > subcord[1] and
self.rheight > subcord[
                                    0] and self.rwidth > column and
self.rheight > row and 0 <= subcord[1] and 0 <= \
                                    subcord[0] and 0 <= column and 0
<= row:

                                    if
self.aflow[subcord[0]][subcord[1]] != self.noacc:

self.aflow[subcord[0]][subcord[1]] += self.aflow[row][column] * (
                                        subcord[3] / tempsolutionsum)
                                    else:

self.aflow[subcord[0]][subcord[1]] = self.aflow[row][column] * (
                                        subcord[3] / tempsolutionsum)


arcpy.AddMessage(self.aflow[subcord[0]][subcord[1]])
                                    if wateracc.TCI_ENABLE:
                                        tcitopsum += subcord[2] *
subcord[3]

                            if wateracc.TCI_ENABLE:
                                if tcitopsum != 0 or tempsolutionsum !=
0:
                                    tcitemp = (self.aflow[row][column] +
1) / (tcitopsum / tempsolutionsum)
                                    if tcitemp > 0:
                                        self.tci[row][column] =
math.log(tcitemp)
                                    else:
```

```
                                        self.tci[row][column] =
self.noacc
                                else:
                                    self.tci[row][column] = self.noacc
                    else:
                        if wateracc.TCI_ENABLE:
                            tcitemp = (self.aflow[row][column] + 1) /
0.001
                            if tcitemp > 0:
                                self.tci[row][column] =
math.log(tcitemp)

                elif value == self.noflow:
                    self.aflow[row][column] = self.noacc

            if PROGRESSPRINT:
                if self.counter != 0 and (self.counter %
self.progressonepercent == 0 or self.counter % 1000 == 0):
                    cas = time.time() - timestart
                    procenta = self.counter / (self.progressonepercent)
                    uptime = cas
                    uhodiny = uptime // 3600
                    uminuty = (uptime - (uhodiny * 3600)) // 60
                    usekundy = (uptime - (uhodiny * 3600) - (uminuty *
60))
                    downtime = (uptime / procenta) * (100 - procenta)
                    dhodiny = downtime // 3600
                    dminuty = (downtime - (dhodiny * 3600)) // 60
                    dsekundy = (downtime - (dhodiny * 3600) - (dminuty *
60))
                    arcpy.AddMessage("".join((
                        "Zpracovano ", str(procenta), " % za ",
str(uhodiny), "h ", str(uminuty),
                        "m ", str(usekundy), "s", " jeste zbyva ",
str(dhodiny), "h ", str(int(dminuty) + 1),
                        "m ", str(dsekundy), "s")))

    def getflowbase(self, row, column):
        if self.aflow[row][column] == self.noacc:
            self.aflow[row][column] = self.waterchange[row][column]
        else:
            self.aflow[row][column] += self.waterchange[row][column]

    def getsubcords(self, row, column, value):
        subcords = []
        if value >= 128:
            value -= 128
            subcords.append([row, column - 1])
        if value >= 64:
            value -= 64
            subcords.append([row + 1, column - 1])
        if value >= 32:
            value -= 32
            subcords.append([row + 1, column])
        if value >= 16:
            value -= 16
            subcords.append([row + 1, column + 1])
        if value >= 8:
            value -= 8
            subcords.append([row, column + 1])
        if value >= 4:
```

```
            value -= 4
            subcords.append([row - 1, column + 1])
        if value >= 2:
            value -= 2
            subcords.append([row - 1, column])
        if value >= 1:
            subcords.append([row - 1, column - 1])
        return subcords

    def getlengthbetwenpixels(self, row, column, subcord):
        if not (row == subcord[0] or column == subcord[1]):
            lengthbetwenpixels = math.sqrt(self.pixelsizex *
self.pixelsizex + self.pixelsizey * self.pixelsizey)
        else:
            if subcord[0] != row:
                lengthbetwenpixels = self.pixelsizex
            elif subcord[1] != column:
                lengthbetwenpixels = self.pixelsizey
        return lengthbetwenpixels

    def gettempsolution(self, row, column, subcords, idx,
contouredsection):

        # Freeman 91
        if self.ALGORITMTYPE == 1:
            power = self.POWER
            tempsolution = pow(subcords[idx][2], power)

        # Quinn 91 vrstevnice
        elif self.ALGORITMTYPE == 2:
            tempsolution = contouredsection * (subcords[idx][2])

        # Quinn 95 kombinace
        elif self.ALGORITMTYPE == 3:
            power = self.POWER
            tempsolution = contouredsection * (pow(subcords[idx][2],
power))

        # Quinn 95 Channel Initiation Threshold
        elif self.ALGORITMTYPE == 4:
            power = pow(((self.aflow[row][column] / self.CIT) + 1),
self.paramh)
            tempsolution = contouredsection * (pow(subcords[idx][2],
power))

        return tempsolution

    def saveresults(self):
        outAcc = numpy.array(self.aflow)
        outUpdateRaster = arcpy.NumPyArrayToRaster(outAcc,
self.referencepoint, self.dsc.meanCellWidth,

self.dsc.meanCellHeight, value_to_nodata=self.noacc)
        fullsave = False
        while not fullsave:
            try:
                if arcpy.Exists(self.PATH_OUT):
                    arcpy.Delete_management(self.PATH_OUT)
                outUpdateRaster.save(self.PATH_OUT)
                fullsave = True
            except Exception, e:
```

```
                arcpy.AddMessage(e.message)
        if self.TCI_ENABLE:
            outTci = numpy.array(self.tci)
            outUpdateRaster = arcpy.NumPyArrayToRaster(outTci,
self.referencepoint, self.dsc.meanCellWidth,

self.dsc.meanCellHeight, value_to_nodata=self.noacc)
            fullsave = False
            while not fullsave:
                try:
                    if arcpy.Exists(self.TCI_OUT):
                        arcpy.Delete_management(self.TCI_OUT)
                    outUpdateRaster.save(self.TCI_OUT)
                    fullsave = True
                except Exception, e:
                    arcpy.AddMessage(e.message)

    def printsummary(self):
        cas = time.time() - timestart
        uptime = cas
        uhodiny = uptime // 3600
        uminuty = (uptime - (uhodiny * 3600)) // 60
        usekundy = (uptime - (uhodiny * 3600) - (uminuty * 60))

        arcpy.AddMessage("".join(("Provedeno za ", str(int(uhodiny)), "h
", str(uminuty), "m ", str(usekundy), "s")))

    def prepareelevationlist(self):
        self.sorted_elevation =
reversed(numpy.sort(numpy.unique(self.rarray)))
        self.sorted_elevation_len =
self.sorted_elevation.__length_hint__()
        self.progressonepercent =
math.ceil(float(self.sorted_elevation_len) / 100)
        self.counter = 0


if __name__ == '__main__':
    try:
        timestart = time.time()
        wateracc = WaterAccumulation(1)
        wateracc.startprocessing()
    except Exception, e:
        exc_type, exc_obj, exc_tb = sys.exc_info()
        arcpy.AddError(e.message)
```

Příloha 2b Programový kód validace pro algoritmus výpočtu přispívající plochy

```
import arcpy
class ToolValidator(object):

  def __init__(self):
    self.params = arcpy.GetParameterInfo()

  def initializeParameters(self):
    return

  def updateParameters(self):
    self.params[5].enabled = False
    self.params[6].enabled = False
    self.params[7].enabled = False
    self.params[8].enabled = False
    self.params[9].enabled = False
    self.params[12].value = 0
    if self.params[4].value == "Freeman 91":
        self.params[5].enabled = True
        self.params[12].value = 1
    elif self.params[4].value == "Quinn 91":
        self.params[6].enabled = True
        self.params[7].enabled = True
        self.params[12].value = 2
    elif self.params[4].value == "Quinn 95":
        self.params[5].enabled = True
        self.params[6].enabled = True
        self.params[7].enabled = True
        self.params[12].value = 3
    elif self.params[4].value == "Quinn 95 with CIT(Channel Initiation
Threshold)":
        self.params[6].enabled = True
        self.params[7].enabled = True
        self.params[8].enabled = True
        self.params[9].enabled = True
        self.params[12].value = 4
    return

  def updateMessages(self):
    return
```