UNIVERZITA KARLOVA V PRAZE

Přírodovědecká fakulta

Katedra aplikované geoinformatiky a kartografie

HODNOCENÍ METOD ANALÝZY DOSTUPNOSTI V RÁMCI SOCIOGEOGRAFICKÉ REGIONALIZACE ČESKA

Diplomová práce

Jan Navrátil

srpen 2010

Vedoucí práce: RNDr. Tomáš Hudeček, Ph.D.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a že jsem všechny použité prameny řádně citoval.

Jsem si vědom toho, že případné použití výsledků, získaných v této práci, mimo Univerzitu Karlovu v Praze je možné pouze po písemném souhlasu této univerzity.

Svoluji k zapůjčení této práce pro studijní účely a souhlasím s tím, aby byla řádně vedena v evidenci vypůjčovatelů.

V Praze dne 30. srpna 2010 ……

Jan Navrátil

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce RNDr. Tomáši Hudečkovi, Ph.D. za věnovaný čas, cenné rady a připomínky. Dále bych chtěl poděkovat všem, kteří mi věnovali čas při řešení problémů, převážně Mgr. Michalu Schneiderovi, Mgr. Stanislavu Grillovi a Petru Babičkovi. Dík patří i všem, co mě podporovali při psaní práce a vytvořili mi vhodné podmínky k jejímu dokončení.

Hodnocení metod analýzy dostupnosti v rámci sociogeografické regionalizace Česka

Abstrakt

Cílem diplomové práce je provést srovnání metod, kterými lze analyzovat dostupnost v regionech. Výsledek analýzy pro jeden region musí pokrývat jen jeho území a nesmí přesáhnout za jeho hranice do sousedních regionů. U více regionů musí totéž platit pro každý region zvlášť. První část práce se věnuje metodám, které umožňují analýzy provést. Jsou popsány jejich základní charakteristiky, výhodné a nevýhodné vlastnosti. Nejrozsáhlejší část textu podrobně popisuje funkce a postupy v modelech a skriptech, které vznikly na základě nalezených metod. Jejich využitím v rozsáhlém testování s různými vstupními daty je vytvořeno hodnocení jednotlivých metod a stanovena vhodná použití pro každou z nich. Na základě dosažených výsledků jsou na závěr naznačeny další možnosti zlepšení směřující k ideálnímu řešení zkoumané problematiky.

Klíčová slova: Dostupnost, regiony, ModelBuilder, Python, ArcGIS, hodnocení metod

Assessment of methods of accessibility analysis of sociogeographical regions in Czechia

Abstract

The aim of diploma thesis is to compare methods of accessibility analysis in regions. The result of the analysis must cover only the region itself and not to overlap the neighbouring regions. This condition must be valid for every single region in the multi region system. The first part of the thesis deals with the methods which make the analysis possible. There are described their main characteristics and their advantages and disadvantages. The core part of the thesis consists of the detailed description of the functions and the approaches in the models and scripts, which were based on the methods identified in the former part. The methods are evaluated by the broad use of the models and scripts with various data samples. For every method is found an applicable use. The results are basis for the further inquires and questions eventually leading to the better understanding and the enhancement of the current approaches in that matters.

Keywords: accessibility, regions, ModelBuilder, Python, ArcGIS, assessment of methods

OBSAH

SEZNAM OBRÁZKŮ A TABULEK

KAPITOLA 1

Úvod a cíle práce

V dnešní době patří síťové analýzy k hojně používaným nástrojům GIS. Současně s tím se ovšem jejich aplikace zaměřuje především jen na samotný výpočet dostupnosti. Smyslem GIS a jeho nástrojů však je umožnit uživateli širší pohled na zkoumanou problematiku než je jen strohá analýza dostupnosti z místa A do místa B. Umožňují nám modelovat a zkoumat různé možnosti nastavení modelů, aplikovat speciální případy, ke kterým může dojít, a také výpočet všelijak omezovat.

Zatímco ve většině případů se používají analýzy dostupnosti k výpočtu konkrétních hodnot vzdálenosti, ať už je to vzdálenost v prostoru nebo například čase, tato práce bude zkoumat aplikaci restrikcí při modelaci akcesibility. Restrikcemi jsou v tomto případě míněny bariéry (hranice) prostorově vymezující průběh výpočtu. Jako názorný příklad může posloužit administrativní členění Česka. Pro představu mějme situaci, kdy se má občan z obce A dostavit na úřad v obci B. Stejný úřad se nachází i v obci C. Obec C je vzdálena od obce A necelých 5 km, obec B je od obce A vzdálena 12 km. Zdá se tedy jasné, že občan by raději jel za úřadem do obce C. Je blíže a cesta trvá kratší dobu. Jenže obec A spadá pod správu obce B, a tudíž občan musí tak či tak vyrazit za úřadem do obce B.

Tento názorný příklad jednoduše ukazuje základní myšlenku vedoucí k napsání práce. Cílem práce je najít různé způsoby, jak automaticky provést výpočet dostupnosti do centrálního místa z celého regionu, který pod toto centrum spadá. Nejde tedy o to nalézt nejbližší centrum, ani centrum, do kterého se lze dostat nejrychleji, ale takové centrum, pod které spadá vybrané místo. Výsledkem pro každou lokalitu bude jedna hodnota dostupnosti, která se bude vždy vázat na regionální centrum dané lokality. Kromě nalezení různých způsobů analýzy si práce klade za cíl tyto metody posoudit. Součástí hodnocení metod bude jednak vzájemné porovnání i hodnocení každé metody samostatně. Tedy z pohledu samotného principu metody, kdy se

budou hodnotit omezení a přednosti z něho vyplývající. Na základě zjištěných výsledků se poté u jednotlivých metod určí vhodnost použití. Vhodností se rozumí, pro jaký konkrétní účel nebo vstupní data je daná metoda ze všech nejvhodnější. Naopak se i určí, pro jaký účel a vstupní data je metoda naprosto nevhodná.

Druhotným cílem práce je vytvořit algoritmy, které budou problematiku řešit podle nalezených řešení. Vytvoření modelů, které vypočtou dostupnost podle definovaného postupu, je důležité i pro vzájemné porovnání nalezených řešení. Jedno z nejdůležitějších kritérií hodnocení metod je čas výpočtu. Bez automatizovaných algoritmů by se toto kritérium nedalo uvažovat. Následně budou samozřejmě vytvořené algoritmy sloužit v praxi kolegům z katedry Sociální geografie a regionálního rozvoje k jejich analýzám. Hodnocení metod jim tak může pomoci k lepšímu rozhodování, jaký způsob výpočtu použít.

Práce si neklade za cíl zkoumat samotný výpočet dostupnosti jako takový. Způsob, jakým je zjištěna konkrétní hodnota dostupnosti pro určitou lokalitu, není podstatný pro tuto práci. Důležitější jsou způsoby, kterými lze dosáhnout toho, aby nástroje GIS vypočetly dostupnost podle cíle práce.

KAPITOLA 2

Současný stav zkoumané problematiky

Následující kapitola se věnuje teoretickému úvodu práce. Kapitola je rozdělena do tří podkapitol. První se věnuje sociogeografické regionalizaci, druhá zkoumá současné uplatnění analýz dostupnosti v geoinformatice a poslední je zaměřena na použitý software a postupy. Jelikož se práce převážně zaměřuje na praktickou část, ve které jsou hledány technické možnosti řešení problematiky a následně jsou podrobeny analýze, je hlavní důraz kladen především na třetí podkapitolu. Dostupnost i regionalizace nejsou cílem zkoumání práce, jsou pouze použitými nástroji.

2.1 Sociogeografická regionalizace

Sociogeografická regionalizace stanovuje rozsahu regionální působnosti středisek a vymezuje komplexní sociálněgeografické regiony. Je to způsob hodnocení středisek, jejich hierarchie i celkové organizace osídlení. Na území Česka je již od 70. let minulého století regionalizace zkoumána především prof. Hamplem. Jelikož se společnost dynamicky rozvíjí, mění se i sociogeografická regionalizace. Vzhledem k tomu, že se jedná o komplexní přístup rozdělení území na celky, je nutné mít k dispozici velmi podrobná data. Regionalizace jsou proto prováděny na základě dat ze Sčítání lidu, domů a bytů (dále SLDB). Poslední provedená regionalizace tedy odpovídá stavu z roku 2001 (Hampl, 2005).

Poslední vytvořená regionalizace má celkem čtyři úrovně – mikroregionální 1. stupně, mikroregionální 2. stupně, mezoregionální a makroregionální. Každá vyšší úroveň je složena z regionů na nižší úrovni na základě regionálních procesů. Makroregionální úroveň poté představuje celé Česko s jediným centrem Prahou. Nejnižší mikroregionální úroveň je stanovena na základě dojížďky za prací, která je základním regionálním procesem. Převládající směr pracovní vyjížďky z nestřediskových obcí do střediskových vymezuje regiony

střediskových obcí (přesný metodický postup viz Hampl; Ježek; Künhl, 1978 a Hampl; Gardavský; Künhl 1987). Podmínkou pro vymezení regionu je minimální velikost celého regionu 15 tisíc obyvatel a samotného zázemí 5 tisíc obyvatel. Jelikož velikost obvodů není jev diskrétní, byly stanoveny i dva typy subregionů. Subregion typu A dosahuje velikosti celého regionu 15 tisíc obyvatel, ale samotné zázemí se pohybuje mezi 2 500 až 4 999 obyvateli. Subregion typu B naopak má samotné zázemí větší než 5 tisíc obyvatel, ale velikost regionu se pohybuje mezi 10 000 – 14 999 obyvateli. V zásadě jde o malá města v relativně periferní poloze nebo velká města v exponované poloze (podrobněji o vymezení hraničních hodnot viz Hampl; Ježek; Künhl, 1978). Na základě dat ze SLDB 2001 bylo vytvořeno 144 mikroregionů 1. stupně, 6 subregionů typu A a 15 typu B. Celkem nejnižší úroveň regionalizace tvoří 165 celků (Hampl, 2005).

Mikroregionální úroveň 2. stupně byla vytvořena z nejnižší úrovně na základě nejsilnějšího směru celkové dojížďky z menších středisek do větších. Na této úrovni tedy nehraje roli pouze pracovní dojížďka, ale uplatňuje se i dojížďka do škol. Minimální velikost mikroregionu 2. stupně byla stanovena na 40 tisíc obyvatel. Postupnou agregací nižších celků vzniklo 70 mikroregionů 2. stupně (Hampl, 2005).

Vymezení mezoregionů je poměrně jednoduchá záležitost. Především je zásadní malý počet dostatečně velkých měst, která by mohla plnit tuto funkci. Druhým důležitým faktorem je nápadný skok mezi 11. a 12. městem, pokud se seřadí podle počtu obyvatel. Dvanácté Karlovy Vary ovšem i přes svoji poměrně malou velikost plní funkci mezoregionálního centra, a tak bylo na území Česka vymezeno celkem 12 mezoregionů (Hampl, 2005).

Porovnáním regionalizací za delší časové období lze vypozorovat změny ve společnosti a její dynamiku. Ovšem v této práci slouží výsledky regionalizace pouze jako testovací data pro vytvořené výstupy. Podoba regionalizace na úrovni mikroregionů 1. stupně je jako příloha 1 na konci práce.

2.2 Dostupnost

Dopravní dostupnost vyjadřuje dosažitelnost jednotlivých uzlů při jednom druhu dopravy. Akcesibilita je ovlivněna především geografickou polohou a těsností uzlů. Dopravní uzel je takový dopravní bod na komunikacích, kde se sbíhají nejméně tři komunikace. Dopravním bodem jsou i místa, kde je umožněno započetí nebo ukončení přepravy (Brinke, 1999).

Analýzy dostupnosti patří v posledních letech k hojně užívaným analýzám. Využívají se nejen ke zkoumání současného stavu nejrůznějších jevů, ale také k plánování a hledání vhodných lokalit pro nová zařízení. Analýzy dostupnosti se velmi často vztahují k regionům. V této souvislosti je možné většinu prací rozdělit na dvě základní skupiny. První skupinu tvoří práce, které si vyberou území jednoho nebo několika málo regionů (počet regionů se obvykle pohybuje v řádu jednotek a dohromady tvoří celistvé území) a zkoumají dostupnost za zvolenou

službou nebo zařízením v daných regionech. Hranice regionů přitom nejsou většinou zohledňovány a slouží pouze pro vymezení celkového rozsahu zkoumaného území. Výsledkem je pak stav dostupnosti zvoleného předmětu zájmu na celém území (viz například Lovett … [et. al.], 2002, Müller; Tscharaktschiew; Haase, 2008 nebo Weber, 2003). Druhá skupina prací také zkoumá dostupnost v určitém území, ale až na základě vypočtené dostupnosti jsou vytvářeny regiony (viz práce Handy, 1993 nebo Walsh; Page; Gesler, 1997). Zatímco v prvním případě bývají regiony většinou administrativní, ve druhém vznikají zcela umělá dělení území do menších celků na základě dostupnosti zvoleného předmětu zájmu. Při analýze dostupnosti jiného jevu by byly výsledky regionalizace odlišné. Vztah regionů a dostupnosti je v této práci pojímán ze zcela jiného hlediska.

2.3 Použitý software

Řešení problematiky byla hledána dvěma různými způsoby. Jednak bylo využito softwaru ArcGIS od společnosti ESRI a jednak byl použit programovací jazyk Python a jeho uživatelské prostředí. Přestože Python nenabízí samostatné řešení, ale pouze využívá funkcí obsažených v ArcGISu, bude každý přístup popsán odděleně. Veškeré funkce obsažené v softwaru ArcGIS popisované v dalším textu jsou vyznačeny kurzívou.

2.3.1 ArcGIS

ArcGIS je komerční software od společnosti ESRI. Všechny postupy popsané níže včetně testování a analýz nalezených metod byly prováděny ve verzi 9.2 ArcEditor. V době psaní práce byla k dispozici i verze 9.3, proto jsou veškeré výstupy z práce uzpůsobeny k tomu, aby řádně fungovaly v obou verzích softwaru[1]. ESRI nabízí pro každou verzi tři typy licencí – ArcView, ArcEditor a ArcInfo. Každý typ licence představuje různě velký balíček funkcionality. Zatímco ArcView obsahuje základní balíček funkcí, licence ArcInfo zahrnuje kompletní soubor funkcí a nástrojů (ESRI ArcGIS Desktop, 2010). Modely a skripty pro řešení problematiky byly navrženy tak, aby je bylo možné spustit i pro licenci ArcView, pokud bude obsahem její instalace extenze Network Analyst.

Network Analyst

Jako komplexní nástroj pro práci s prostorovými daty obsahuje ArcGIS i nástroje pro zpracování síťových analýz. Jedná se o specifické analýzy, pro které je vytvořen speciální soubor funkcí – extenze Network Analyst. Mimo nástrojů pro samotnou analýzy po sítí obsahuje extenze Network Analyst další, bez kterých by analýzy nemohly proběhnout. Jedná se

[1] V roce 2010 byla představena verze 10 (ESRI Press Release, 2010). Ovšem ani v době dokončení práce nebyla v Česku k dispozici, takže nelze určit, zda jsou výstupy kompatibilní i s touto verzí softwaru.

například o funkce pro tvorbu network datasetu (viz níže). Pokud není extenze nainstalována a povolena[2], nemůže žádná síťová analýza proběhnout.

Network Analyst potřebuje zvláštní typ sítě tzv. network dataset, aby bylo možné provádět analýzy. Network dataset může být vytvořen z jakékoliv liniové vrstvy v aplikaci ArcCatalog. Od běžné liniové vrstvy se liší tím, že v sobě obsahuje několik subvrstev. Jednak původní liniovou vrstvu, jednak vrstvy související s topologií sítě. Jedná se o liniovou subvrstvu úseků a bodovou subvrstvu uzlů (ESRI, 2007). Všechny úseky obsahují informaci o hodnotě každého atributu, který lze použít k výpočtu dostupnosti. Hodnota určuje, o kolik se dostupnost zvýší, pokud se daný úsek překoná. Pro názornost si lze představit, že je jeden úsek s hodnotu atributu čas 1 000 sekund. To znamená, že na překonání celého úseku je nutné v reálném světě 1 000 sekund (výpočet trvá samozřejmě podstatně kratší dobu). Pokud by tyto úseky byly dva za sebou, byla by hodnota dostupnosti z jednoho okraje k druhému rovna 2 000 sekund. Každý úsek si tak nese informaci o tom, o kolik se vypočtená dostupnost zvýší v případě, že výpočet proběhne po tomto úseku. Při tvorbě network datasetu je z atributové tabulky původní liniové vrstvy vybrán minimálně jeden atribut, který bude s odpovídající hodnotou přiřazen každému úseku a následně používán pro výpočet dostupnosti (ESRI, 2006). Subvrstva uzlů v topologii sítě určuje místa, kde končí jednotlivé úseky. Jde o křižovatky úseků, místa napojení dvou úseků a volné konce úseků. Přehledný návod, jak vytvořit network dataset lze nalézt v Network Analyst Tutorialu (viz seznam zdrojů informací).

Network Analyst umožňuje pomocí network datasetu provádět celkem 4 různé analýzy. Pro účely práce jsou vhodné dvě z nich: Service Area a OD Cost Matrix. Service Area (do češtiny se dá přeložit jako obslužná zóna) slouží k analýze rozložení dostupnosti v území. Na základě nadefinovaných zájmových hodnot jsou vygenerovány zóny (polygony), které jsou do těchto hodnot dostupné ze zvolených míst. Algoritmus vytvoří izolinie[3] dostupnosti odpovídající zadaným hodnotám. Území uzavřené mezi dvě izolinie nebo ohraničené izolinií (v závislosti na nastavení analýzy – viz dále) tvoří jeden polygon. Pokud budou nastaveny například hodnoty 10, 20 a 30 km, budou vygenerovány polygony, které budou označovat území ve vzdálenosti 10, 20 a 30 km od zvoleného počátečního místa (viz obr. 5). Analýza OD Cost Matrix počítá konkrétní hodnotu dostupnosti mezi počátečními a cílovými místy. Oproti Service Area, jejímž výsledkem je rozložení dostupnosti v území, poskytuje OD Cost Matrix přesné hodnoty mezi zadanými místy. Výstupem je vrstva linií, kde vždy jedna přímka spojuje počáteční a cílové místo a jako jeden z atributů linie je hodnota dostupnosti mezi těmito

---

[2] Skripty vytvořené v Pythonu v sobě obsahují kontrolu, zda je extenze nainstalována a povolena. Pokud je nainstalována a není povolena, skript jí povolí. Pokud není nainstalována, skript se zastaví. Modely v ModelBuilderu nic takového neumožňují, a proto se v jakémkoliv jiném případě, než že je extenze nainstalována a povolena, probíhající analýza ukončí.

[3] Izolinie jsou linie spojující místa se stejnou hodnotou nějakého jevu. Podle toho, co hodnota vyjadřuje, se odvozují různé názvy pro izolinie. Linie spojující místa dosažitelná za stejnou jednotku času se nazývají izochrony, spojení míst ve stejné vzdálenosti zase izochory nebo ekvidistanty (Čapek, 1979).

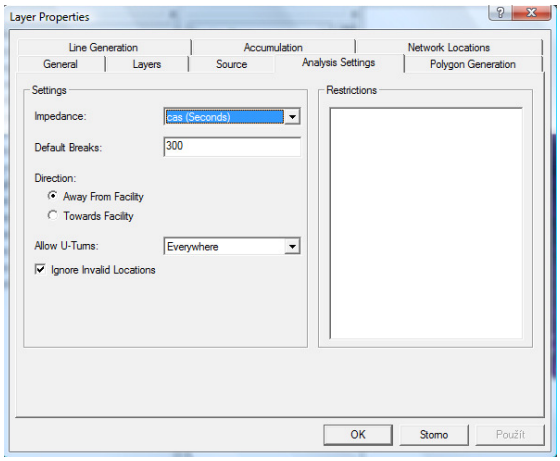
body (viz obr. 13). V dalším textu práce budou blíže popsány jednotlivé možnosti nastavení obou analýz.

Service Area se podobně jako network dataset skládá z několika subvrstev. Jsou to Facilities, Barriers, Polygons a Lines. Subvrstva Facilities představuje počáteční body výpočtu. Bodová vrstva, která bude nahrána do této subvrstvy, bude tvořit místa, ze kterých bude následná analýza vypočítána. Při nahrávání zvolené bodové vrstvy je možné u vlastnosti Name zvolit atribut z atributové tabulky nahrávané vrstvy, který bude obsažen ve výstupní vrstvě polygonů jako jejich název. Barriers mohou obsahovat bodovou vrstvu, která bude značit omezení v analýze. Omezení mohou být jednostranná nebo oboustranná. Jednostranná omezení zamezí v pohybu přes bariéru pouze v jednom směru (analogie jednosměrných ulic v dopravě), oboustranná brání analýze oběma směry. Zatímco subvrstvy Facilities a Barriers tvoří vstupy do analýzy, Polygons a Lines obsahují výstupy analýzy podle jejího nastavení.

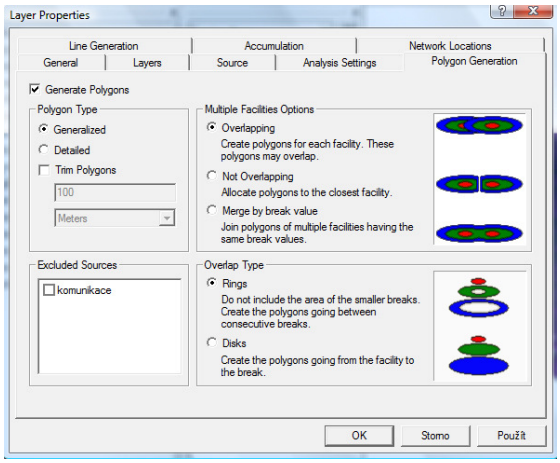
Množství nastavení v analýze je poměrně obsáhlé a není účelem práce se věnovat přímo jednotlivým analýzám, a proto jejich výčet nebude kompletní. Zmíněna budou pouze ta nastavení, která mají dopad na zkoumanou problematiku a byla později použita v modelech a skriptech. Nastavení parametrů analýzy je rozděleno do více záložek. První z nich umožňuje nastavit obecné parametry (viz obr. 1). Předně se musí vybrat *impedance attribute* neboli atribut z network datasetu, který bude použit pro analýzu. Poté se volí hraniční hodnoty polygonů dostupnosti a upravuje se, zda-li má analýza probíhat směrem k a nebo od bodů načtených do subvrstvy Facilities. Defaultně je nastavena možnost od bodů. Možné je povolit při analýze otočky o 180 ° a vynechat takové body ze subvrstev Facilities a Barriers, které se nepodařilo umístit. Obě nastavení jsou defaultně povolena. Atribut a hraniční hodnoty musí vždy zadat uživatel podle účelu analýzy.

Jiná záložka se věnuje generování polygonů (viz obr. 2). Je možné zvolit mezi generalizovanými a detailními polygony a oba typy je možné ještě zahladit. V modelech jsou vytvářeny pouze generalizované polygony. Jejich generování je rychlejší a v případě lokalit na síti nebo v její těsné blízkosti je výsledek identický. Rozdíl mezi detailními a generalizovanými polygony je znatelný až v místech daleko od sítě, kde dochází ke interpolaci hodnot. Detailní polygony lépe vystihují rozložení dostupnosti v území mimo síť network datasetu. Polygony je možné generovat pro každý bod samostatně, nezávisle na ostatních bodech. Druhou možností je spojit polygony generované z různých míst, pokud mají stejné hraniční hodnoty. Poslední možné nastavení zajistí vygenerování polygonů z daného bodu až do míst, která jsou stejně daleko od jiného bodu ze subvrstvy Facilities. To znamená, že každé místo pokryté polygony z určitého bodu má k tomuto bodu blíže než ke všem ostatním bodům, odkud jsou také polygony generovány. Kromě nastavení polygonů ve vztahu k bodům, ze kterých jsou generovány, lze nastavit i polygony ve vztahu k sobě samým. Nabízejí se dvě možnosti nastavení. Buď bude každý polygon kromě prvního ohraničen dvěma hraničními hodnotami dostupnosti, a nebo vždy pouze jednou. Pokud by byly nastaveny hraniční hodnoty

dostupnosti na 10, 20 a 30 km, znamenalo by to, že buď by byly vygenerovány polygony pokrývající území 0 – 10 km, 10 – 20 km a 20 – 30 km od počátku analýzy, anebo by byly vygenerovány polygony 0 – 10 km, 0 – 20 km a 0 – 30 km (ESRI, 2007).



Obr. 1 Okno s obecným nastavením analýzy Service Area v aplikaci ArcMap



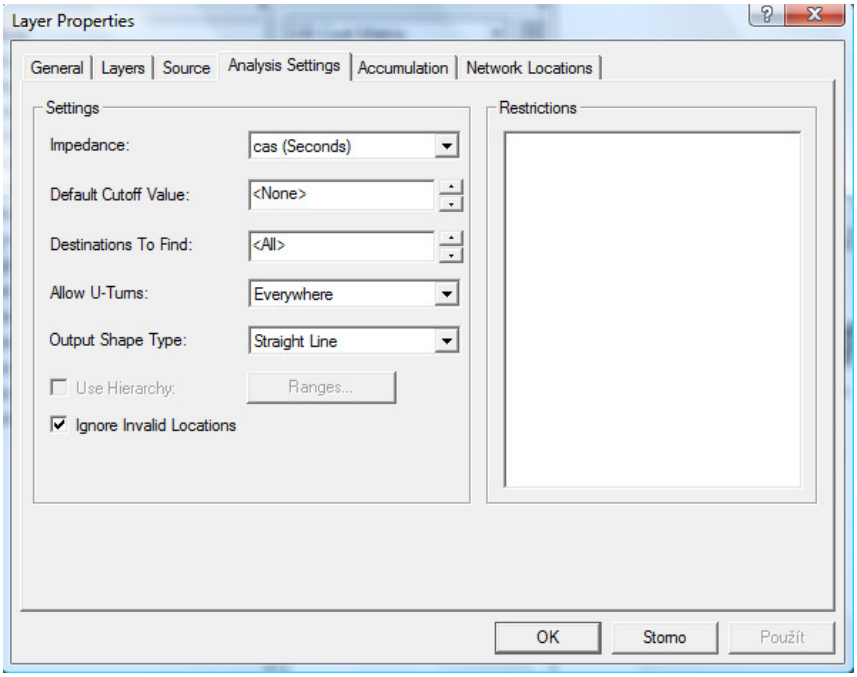
Obr. 2 Okno s nastavení polygonům analýzy Service Area v aplikaci ArcMap

V práci jsou zvoleny samostatné polygony pro každé centrum a tvorba mezikruží (prvně jmenovaný případ). Ostatní nastavení analýzy se ponechávají defaultní. Nemají přímý vliv na podobu výstupu analýzy, a proto již nebudou blíže popsány.

Druhou zkoumanou analýzou je OD Cost Matrix. OD je zkratka sousloví Origin-Destination a celý název analýzy je možné přeložit jako matice nákladů mezi výchozími a cílovými místy. Matice je použito proto, že během analýzy dojde k vypočtení přesné hodnoty dostupnosti (nákladů) mezi všemi výchozími a cílovými místy. Podobně jako Service Area i OD Cost Matrix obsahuje čtyři subvrstvy: Origins, Destinations, Barriers a Lines. Subvrstva Barriers je shodná s tou, která byla u předchozí analýzy. Origins a Destinations jsou obdobnou Facilities. Do subvrstvy Origins je nahrávána bodová vrstva počátečních míst a do subvrstvy Destionations bodová vrstva cílových míst. Oběma lze ve vlastnosti Name vybrat atribut obsahující jejich názvy. Subvrstva Lines slouží pro ukládání výsledků analýzy. Jsou v ní obsaženy linie spojující počáteční a cílová místa.

Nastavení parametrů je poměrně jednodušší než u Service Area. Důležitá je pouze záložka s obecnými parametry (viz obr. 3), u ostatních je použito defaultní nastavení. Na této záložce se volí opět atribut z network datasetu. Namísto hraničních hodnot se může zvolit konečná hodnota dostupnosti. Jakékoliv cílové místo, které se bude nacházet za touto hodnotou, bude ignorováno. Do ostatních míst bude vypočtena konkrétní hodnota dostupnosti. Analýza se dá omezit i tak, že se nastaví, ke kolika nejbližším cílovým místům z každého výchozího místa bude počítána dostupnost. Obě omezení jsou primárně vypnuta a takové nastavení je použito i v modelech. Podobně jako u obecného nastavení Service Area lze povolit otočky o 180 ° a vynechání neumístěních výchozích míst, cílových míst a bariér. Poslední nastavení se týká výstupu. Tím

mohou a nemusí být linie. Pokud se zvolí negenerování linií, je výsledkem liniový shapefile, který sice neobsahuje žádnou linii, ale v atributové tabulce jsou uvedeny všechny záznamy. Pokud jsou generovány linie, jsou ve výstupní vrstvě obsaženy i linie (ESRI, 2007).



Obr. 3 Okno s obecným nastavením analýzy OD Cost Matrix v aplikaci ArcMap

ModelBuilder

ModelBuilder je standardní součástí ArcGISu. Jedná se o rozšíření, které umožňuje vytvářet modely. V tomto případě to znamená automatizovaný sled kroků, které se postupně provádí nad vstupními daty. Jde o formu zautomatizování procesu bez použití programovacího jazyka. ModelBuilder není určen k vytváření nových nástrojů nebo skriptů. Je určen pouze k používání již existujících nástrojů v rámci ArcGISu. Jeho výhodou je, že pokud se musí nějaký proces (sled několika nástrojů) provádět opakovaně, je možné celý proces namodelovat a nemusí se všechny kroky postupu provádět manuálně jeden za druhým (Štych a kol., 2008). Je to také výhodné, pokud mají celý proces provádět lidé bez širšího povědomí o GIS technologiích. Stačí pouze nastavit vstupní hodnoty a model sám vytvoří požadovaný výsledek.

ModelBuilder byl v práci využit pro vytvoření dvou ze tří řešení zkoumané problematiky. Zpřístupněn je pomocí toolboxů, kdy se v rámci některého z nich může vytvořit model. Ten je poté upravován v ModelBuilderu. Samotný ho spustit nelze.

ToolValidator

ToolValidator je třída programovacího jazyka Python umožňující upravovat chování skriptu před jeho spuštěním v ArcGISu (ESRI, 2009). Jedná se o nástroj na pomezí mezi ArcGISem a programovacím jazykem Python. Součástí ArcGISu je od verze 9.3 a v práci je použit

u skriptu pro tuto verzi. Zpřístupněn je přes vlastnosti skriptu, kde se vyskytuje záložka Validation (viz obr. 14). Zde lze vložit zdrojový kód. Obsah kódu je vykonán, jsou-li vyplněna patřičná pole v okně po otevření skriptu. Lze tak například na základě vložených vrstev do skriptu určit rozmezí proměnné, která se zadává do dalšího pole, případně nějaké pole deaktivovat a podobně. Přesné použití ToolValidatoru ve skriptu vzniklém v rámci práce je podrobněji rozebíráno v podkapitole 4.3.2.

2.3.2 Python

Python je objektově orientovaný programovací jazyk. Vyznačuje se jednoduchou a čistou syntaxí, která z něj činí snadno čitelný jazyk vhodný i pro začátečníky. Přesto se hojně užívá pro vývoj složitých aplikací a neslouží pouze ke psaní krátkých skriptů. Mezi jeho největší klady patří rychlost psaní kódu a jednoduchá syntaxe, velké množství modulů s nepřeberným množstvím funkcí a také jeho nezávislost na platformě. Python lze spustit pod všemi nejpoužívanějšími operačními systémy. Vzhledem k tomu, že se jedná o open source projekt, má Python velkou komunitu uživatelů, kteří se podílejí na jeho dalším vývoji. Python lze kombinovat s jinými programovací jazyky a prostředími, a tudíž se vyhnout nedostatkům obou (Python, 2010).

Nevýhodou může být pomalejší běh aplikací napsaných v Pythonu. Překladače jazyků pro rychlý vývoj aplikací se starají o mnoho aspektů programování, které musí být v tradičních programovacích jazycích explicitně ošetřeny programovým kódem. Jestliže programátor musí přesně popsat všechny aspekty, lze pro danou strukturu alokovat potřebné množství paměti a velmi efektivně pracovat se systémovými zdroji. V Pythonu se o to starají především překladače, takže se množství potřebné paměti řídí aktuální potřebou programu. Jestliže velikost struktury překročí možnou mez, je nutné ji přenést na jiné místo a běh programu se zpomaluje. Případným problémům s výpočetně náročnějšími částmi aplikace je možné předcházet implementací kódu z modulů jazyka C nebo C++ (Harms; McDonald, 2003). Pro někoho by mohlo být i nevýhodou, že neexistuje mnoho literatury věnující se Pythonu. Místo toho je ovšem k dispozici dostatek komunitních serverů (nejznámější český server je http://www.py.cz). Kromě základních informací o Pythonu a uživatelský fór často obsahují i mnohé návody a nebo lekce pro začátečníky (Programujte.com, 2006), díky kterým se může naučit programovat v Pythonu každý.

Mimo jiné aplikace je Python standardní součástí ArcGISu, se kterým je běžně instalován. Pro verzi ArcGISu 9.2 je součástí Python ve verzi 2.4, s verzí 9.3 se instaluje verze Pythonu 2.5. Propojení Pythonu s ArcGISem je zajištěno pomocí modulu arcgisscripting. Importem toho modulu do skriptu je možné zpřístupnit veškeré funkce obsažené v softwaru ArcGIS. Snadno tak lze zkombinovat tyto funkce s funkcemi z jiných modulů a plně využít přednosti programovacího jazyka. Python se tím stává vhodným nástrojem pro automatizaci procesů a efektivnější zpracování.

V následující ukázce programového kódu bude názorně předvedena jednoduchost syntaxe a úspornost jazyka Python včetně jeho propojení s funkcemi obsaženými v softwaru ArcGIS. V ukázce je provedení funkce *Buffer* na zvoleném shapefilu. Jednoduchost celého procesu je v tom, že kromě importování potřebných modulů a vytvoření geoprocessingového objektu pro zpřístupnění funkcí z ArcGISu se další kód týká pouze nastavení samotné funkce. Je nadefinován shapefile, který je vstupem do funkce, je určena velikost bufferu a výstup z funkce. Samotná funkce obsahuje všechny tyto proměnné a další nastavení, která ovlivňují podobu výstupu z funkce. Není potřeba žádný další kód. Text uvedený za znakem # je komentář k jednotlivým řádkům kódu.

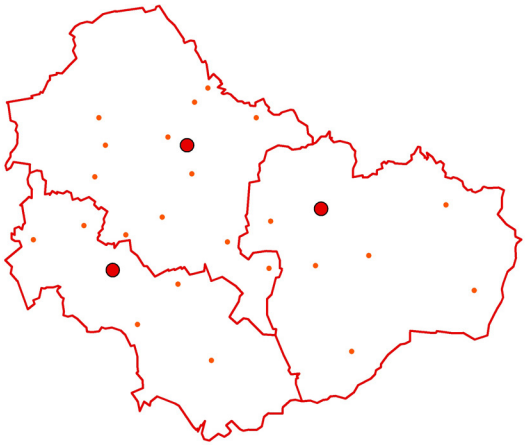
```
# Importovani systemovych modulu.
import sys, string, os, arcgisscripting

# Vytvoreni geoprocessingoveho objektu. Pres tento objekt jsou volany vsechny
# funkce ArcGISu.
gp = arcgisscripting.create()

# Promenne skriptu. Jedna se o vrstvu komunikaci a velikost bufferu. V pripade,
# ze komunikace nebudou uzivatelem zadany, budou pouzity komunikace
# definovane nize. Jde o pouziti podminky if. Veskery kod, ktery se ma provest pri
# splneni podminky je odsazen. Pouhym odsazenim textu se urci, co vse patri
# k podmince.
komunikace = sys.argv[1]
if komunikace == '#':
  komunikace = "C:\\Vrstvy\\komunikace.shp"
velikost = sys.argv[2]

# Definice vystupu z funkce Buffer.
komunikace_Buffer = "C:\\Vrstvy\\komunikace_Buffer.shp"

# Samotna funkce Buffer. Funkce je zpristupnena pres geoprocessingovy objekt.
gp.Buffer_analysis(komunikace, komunikace_Buffer, velikost, "FULL", "ROUND", "NONE", "")
```

KAPITOLA 3

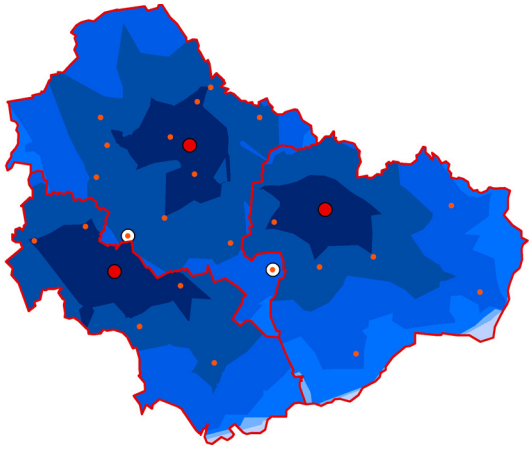
Metody řešení

Na začátku hledání metody, jak dosáhnout správného výsledku, je výsledek samotný. Nejprve je nutné si jasně určit, jak má vypadat výstup, a podle toho navrhnout způsoby, jak k danému výstupu dojít. Cílem práce je vytvořit takový model, který by pro každou lokalitu v regionu vypočetl dostupnost do svého regionálního centra. Důležité je slovo „svého“. Každá lokalita tak má předem již jasně dané, do kterého centra by se měla dostupnost analyzovat. Příslušnost lokalit k jednotlivým centrům je stanovena prostorovým vymezením regionů. Každé centrum má svůj region a všechny lokality v regionu proto spadají k tomuto centru.

Na rozdíl od jiných analýz dostupnosti nevzniká příslušnost lokality k nějakému centru až v průběhu analýzy na základě hodnot dostupnosti. Jedná se o obrácený postup, kdy se analýza dostupnosti nebere jako jeden z faktorů při tvorbě regionalizace, ale kdy slouží regionalizace jako prostorová restrikce analýzy dostupnosti.



Obr. 4 Ukázka podoby vstupních dat



Obr. 5 Ukázka podoby správného výstupu

Obrázek 4 názorně ukazuje, jak vypadají vstupní data. Existuje vrstva regionů, z nichž každý má své centrum (velký červený bod) a své lokality (malé oranžové body). Lokality hrají roli hlavně u analýzy OD Cost Matrix (viz dále), u analýzy Service Area jsou nepodstatné. Vedlejší obrázek 5 zase představuje, jak by měl vypadat výsledek modelů pro analýzu Service Area. Za pozornost stojí lokality ohraničené bílým polem. V rámci svého regionu se nachází na periferiích. Kdyby byla lokalita vždy přiřazena jen ke svému nejbližšímu centru, nebylo by to u těchto dvou lokalit jejich regionální centrum, jak tomu napovídá rozložení dostupnosti v sousedních regionech.

Přesně určený cíl tedy umožňuje hledat řešení. Celkem byla nalezena 3 řešení problematiky. Jedná se o řešení s využitím bariér, řešení využívající VBA funkce[4] pro výběr prvků odpovídající cíli a dávkové zpracování předchozího řešení pomocí programovacího jazyka Python.

První dvě metody řešení jsou rozdílné, využívají zcela odlišného přístupu k dosažení výsledku. Poslední metoda je modifikací druhé. Ovšem nejedná se pouze o jednoduchou úpravu předchozí metody, nýbrž o zcela jiný přístup k řešení. Zatímco první dvě metody jsou zpracovány v ModelBuilderu, tedy ve standardním nástroji softwaru ArcGIS (Štych a kol., 2008), poslední si vyžádala zpracování pomocí programovacího jazyka. Nástroje, které jsou součástí ArcGISu, jsou totiž nedostatečné pro řešení problematiky cestou dávkového zpracování vstupních dat.

Všechny metody mají společnou jen jednu věc: do prostředí ArcGISu jsou implementovány pomocí toolboxu. Nežli je vůbec možné pracovat s ModelBuilderem a nebo spustit skript v rámci ArcGISu, musí se vytvořit vlastní toolbox. Do něj je pak možné vkládat nové (prázdné) modely a nebo vytvořené skripty. Až pak lze začít pracovat s ModelBuilderem a editovat nové modely tak, aby nakonec vytvořily hledaný výsledek.

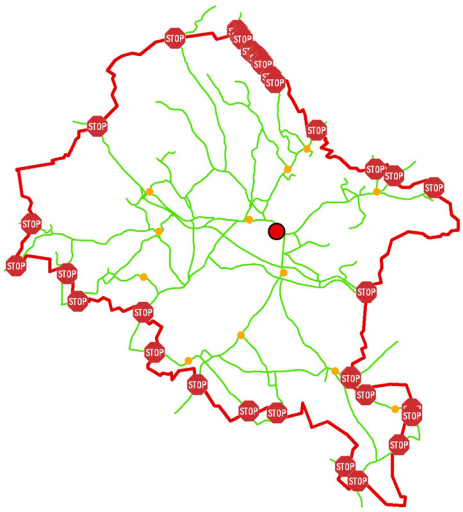
V následující části textu budou blíže popsány jednotlivé metody. Bude nastíněn princip řešení a zdůrazněny klady či zápory z něho vyplývající. Podrobný popis všech použitých funkcí a jejich význam v celém modelu bude pro jednotlivé metody popsán v další kapitole.

3.1 Bariéry

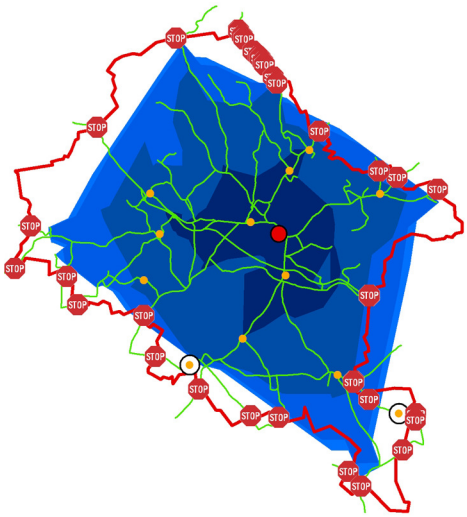
Prvním způsobem, jak daný problém vyřešit, je využití bariér. Do analýzy dostupnosti může vstupovat libovolný počet bodových vrstev, u kterých lze nastavit, že budou sloužit jako tzv. bariéry. Ty jsou pak při samotné analýze uvažovány jako omezení v provozu. Bariéry mohou být propustné v jednom směru a nebo v žádném (ESRI, 2007). Pokud by tedy na každém místě, kde komunikace prochází hranicí regionu, byl bod načtený v analýze dostupnosti jako

[4] VBA je zkratka slov Visual Basic for Applications. Visual Basic pak představuje programovací jazyk vyvinutý v roce 1991 společností Microsoft pro grafické potřeby jejich operačního systému Windows. Od té doby se stal nedílnou součástí mnohých aplikací. (Max Visual Basic, 2010)

oboustranná bariéra, výpočet by se na tomto bodě zastavil. Pokaždé, kdy by výpočet došel až k tomuto bodu, nemohl by pokračovat za něj, tudíž by nemohl pokračovat ani za hranici regionu. To samé by platilo i z druhé strany. Ačkoliv by se na první pohled mohlo zdát, že se jedná o metodu, která kompletně řeší zadaný problém, opak je pravdou.



Obr. 6 Bariéry na hranicích regionu



Obr. 7 Polygony dostupnosti při aplikaci bariér

Obrázek 6 ukazuje rozmístění bariér na každém místě, kde komunikace protíná hranici regionu. Obrázek 7 pak znázorňuje výsledek analýzy Service Area s využitím metody bariér. Za povšimnutí stojí fakt, že ne celé území je pokryté polygony. V místech, kde polygony nejsou, není možné zjistit hodnotu dostupnosti touto metodou. Při porovnání s obrázkem 5 výše, kde byla použita jiná metoda ale stejné hraniční hodnoty dostupnosti, je jasné, že metoda bariér je v okolí hranic regionů nepřesná.

Jednoznačnou, avšak víceméně jedinou výhodou toho způsobu výpočtu, je jeho rychlost. Jedná se o nejrychlejší způsob, jak dojít k cíli. Za výhodu se dá považovat i poměrně jednoduchá implementace tohoto řešení v prostředí ModelBuilderu. Na rozdíl od druhé metody aplikované také v ModelBuilderu není nutné upravovat vstupní data tak, aby mohla být vybrána pouze data odpovídající cíli práce (viz podkapitola 4.2). Ovšem nevýhody metody ve většině případů převáží její výhody.

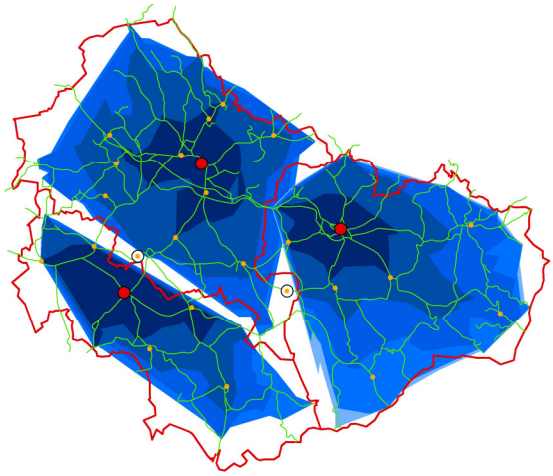
Velkou nevýhodou je, že do výpočtu dostupnosti v rámci regionu jsou uvažovány pouze komunikace uvnitř regionu. Tato nevýhoda přímo vyplývá z principu výpočtu. V případě, že za hranicí regionu vede komunikace, která by byla výhodnější než komunikace uvnitř regionu (například dálnice), nelze ji použít. Výpočet se totiž zastaví na hranici regionu před dosažením této komunikace.

Může nastat i jiná situace, která souvisí s předchozí. Pokud bude nějaká lokalita se svým regionálním centrem spojena pouze komunikací, která nevede celá po území regionu, nebude

možné provést analýzu dostupnosti pro tuto lokalitu. Typicky se může jednat o území, která jsou rozdělena nějakou přírodní bariérou jako jsou řeky nebo pohoří. V takové situaci bývají obě oddělená území spojena jen na některých místech (mosty, horské průsmyky). Je pravděpodobné, že žádné takové přechodové místo se nebude nacházet na území regionu. Aby bylo možné se tedy dostat z lokality do regionálního centra, musí se nejprve opustit území regionu, překonat přirozenou překážku (řeku, pohoří apod.) a posléze se opět vrátit na území svého regionu. Pokud by nastal takový případ, analýza dostupnosti by pro tuto lokalitu nebyla možná. Výpočet probíhající z centra regionu by se zastavil na místě, kde komunikace poprvé opouští území regionu. Samozřejmě stejná situace může nastat i v jiných případech než jsou přirozené překážky (viz obr. 8). Lokalita označená oranžovým bodem v bílém kruhu vpravo na obrázku leží na komunikaci, která v obou směrech opouští území regionu.

V obou popsaných situacích dochází ke zkreslení výsledků. Ve většině případů, ve kterých by mohl model nalézt uplatnění, je to nevýhodná vlastnost (viz příklad s územněsprávními celky v úvodu – nejvýhodnější nebo jediná cesta za úřadem může částečně vést mimo území spravované úřadem). Ovšem jsou případy, ve kterých je tato vlastnost nutností. Pokud by se použil tento model na úrovni států, může být vstupní podmínkou, že se nesmí překročit jejich hranice. Poté by byl ideálním prostředkem, jak dosáhnout správných výsledků.

Další nevýhoda se týká pouze analýzy Service Area a souvisí s tvarem vygenerovaných polygonů. Výpočet dostupnosti jako takový probíhá pouze po komunikacích. V prostoru mezi nimi jsou u analýzy Service Area hodnoty dostupnosti interpolovány a až na základě těchto interpolovaných hodnot se generují polygony (ESRI, 2007). Zastavení výpočtu na hranicích regionů vede k tomu, že vymodelované polygony nekopírují hranici regionů. Namísto toho jsou polygony tvořeny úsečkami spojující místa, kde se výpočet zastavil. (viz obr. 8). Vygenerované polygony náležící jednomu regionu buď zasahují do regionu jiného, a nebo nedoléhají k hranici vlastního regionu a vznikají bílá místa, kde není žádná hodnota dostupnosti.



Obr. 8 Polygony dostupnosti pro více regionů po aplikaci bariér do výpočtu

Pro lokality nacházející se blízko hranic regionu může nastat situace, že nebudou mít žádnou nebo více než jednu hodnotu dostupnosti. Jedna se bude vztahovat k jejich centru a další k sousedním centrům. To může přinejmenším zpomalit analýzu výsledků pro jednotlivé regiony. Nelze ani vyloučit zcela mylnou interpretaci dosažených hodnot.

Nevýhoda týkající se analýzy Service Area by se dala odstranit dalšími úpravami algoritmu. Přesahující části polygonů do jiných regionů mohou být odstraněny a naopak chybějící uvnitř regionu doplněny. Tím by se však zvýšila náročnost výpočtu a vedlo by to k tomu, že by ztratil své výhody – rychlost a jednoduchost. Nemluvě o tom, že úpravy modelu jsou spojeny s velkým množstvím problémů.

Přesahující či nedoléhající polygony se dají upravit dvěma způsoby. Jednodušší na implementaci je využít opět ModelBuilderu, ale to by vyžadovalo od uživatele zadání dalšího parametru do modelu. ModelBuilder slouží pouze k automatizaci namodelovaného procesu. Neobsahuje žádné nástroje, které by umožňovaly vytvářet cykly nebo podmínky, jak je to běžné u programovacího jazyka (Štych a kol., 2008). Uživatel by musel zadat přesně hodnotu poslední hranice vytvořených polygonů, aby mohl model „vyplnit“ prázdná místa polygonem s touto hraniční hodnotou. V dalším kroku by muselo dojít ke spojení posledních vymodelovaných polygonů s polygony vyplňujícími volná místa, aby vznikl pro každý region jen jeden polygon u každého intervalu dostupnosti. S tím souvisí jeden problém. Je nutné zajistit, aby se vždy spojily jen polygony nacházející se ve stejném regionu a aby se nespojovaly polygony mezi regiony. V neposlední řadě je ještě problém při odstraňování přesahujících polygonů. Jak rozpoznat po rozdělení polygonu na dvě části (jedna uvnitř regionu, druhá vně), která část polygonu je vnější a má být smazána.

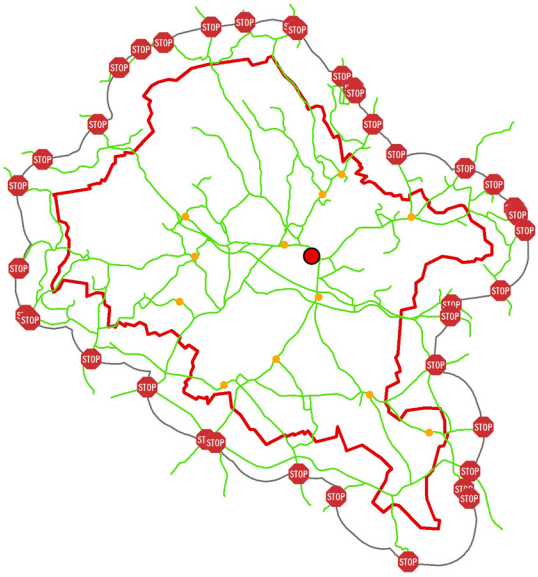
Složitější možností na implementaci je využití programovací jazyka. Tím by odpadla další účast uživatele. Pomocí Pythonu by se všechny potřebné vstupní informace mohly zjistit z již zadaných hodnot. Avšak všechny uvedené problémy by se musely vyřešit, tudíž není řešení přes programovací jazyk velkým přínosem.

Řešení všech problémů se týká pouze situace, kdy uživatel zadal poslední hodnotu dostupnosti blízkou takové, jaká se vyskytuje na hranicích regionu. Ovšem regiony mohou být různě velké (a logicky tedy i hodnoty dostupnosti na hranici každého regionu velmi rozdílné), podobně centrum nemusí být v blízkosti středu regionu (hodnota dostupnosti na hranici regionu blízko centra bude značně odlišná od hodnoty na vzdálenější části hranice). Stejně tak uživatel mohl zadat příliš malé hodnoty a nebo naopak příliš velké hodnoty dostupnosti. Vymodelované polygony se tak nemusí vůbec k hranici regionu přiblížit a nebo naopak by některé polygony zcela obepínaly region (za předpokladu, že by výpočet nebyl zastaven na hranici regionu). Výsledkem všech těchto proměnných je to, že nelze automaticky určit, jakou hodnotu mají mít polygony vyplňující bílá místa.

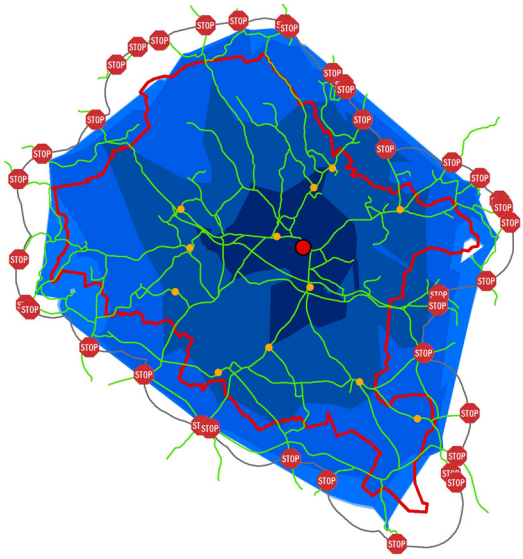
Většina výše nastíněných problémů je součástí metody využívající VBA funkce a jsou v modelu řešeny. Takže pokud by došlo k úpravě algoritmu pro výpočet dostupnosti pomocí

bariér tak, aby nevznikal nevzhledný výstup u analýzy Service Area, vzniklo by řešení na rozmezí metody bariér a metody využívající VBA funkce. Ovšem taková kombinace by spojovala spíše negativa obou metod – zkreslené výsledky z první metody a pomalejší výpočet z druhé metody. Z tohoto důvodu nebyla úprava modelu provedena.

První nevýhodná vlastnost odstraněna být nemůže, jelikož vychází z principu metody. Možnou alternativou je volba bufferu okolo každého regionu a použít bariéry až na hranici bufferu (bariéry by musely být nastaveny jako jednostranné, aby bránily pokračování ve výpočtu jen směrem ven z regionu[5]). Taková situace je nastíněna na obrázcích 9 a 10. Je však otázkou, jak velký buffer volit, což se odvíjí od vstupních dat. Tato modifikace je tedy také nevhodná, protože by primárně závisela na volbě uživatele. V případě, že by se aplikovala, bylo by nutné opět počítat s faktory zmíněnými výše, jelikož by se musely odstranit polygony vygenerované za hranicí regionu. Dosažené výsledky jsou evidentně lepší než u původní metody, jinak ale nevýhody oproti původní metodě přebývají.



Obr. 9 Bariéry na bufferu okolo regionu



Obr. 10 Polygony dostupnosti při použití bufferu

3.2 VBA funkce

Další možnost je navržena tak, aby odstraňovala nedostatky předchozí metody. Do analýzy dostupnosti nevstupují žádné další vrstvy, které by zastavovaly výpočet. Analýza proběhne bez omezení podle vstupních informací zadaných uživatelem. Tímto postupem je zajištěno, že nedojde k žádnému omezování jako tomu bylo u metody bariér. Lze tak využít i komunikace mimo území regionu pro výpočet dostupnosti v daném regionu.

---

[5] Je otázkou, jestli by šlo takto nastavit bariéry automaticky. A v neposlední řadě je dosti možné, že by se vzájemně ovlivňovaly bariéry ze sousedních polygonů a došlo by ke zkreslení výsledků.

Jestliže proběhne analýza z regionálních center bez jakýkoliv restrikcí, je pak nutné vytvořit výstup odpovídající stanovenému cíli. Je třeba odstranit výsledky dosažené pro lokality nacházející se mimo území regionu. To se musí provést pro všechny regiony zahrnuté v analýze. U analýzy Service Area se může u mnoha lokalit překrývat více polygonů vypočtených z různých regionálních center. Analýza OD Cost Matrix zase vypočte přesnou hodnotu dostupnosti z každé lokality do každého regionálního centra (přičemž výsledkem má být pouze jedna hodnota do centra, pod které lokalita spadá). Přesný a podrobný popis postupu, jak je toho dosaženo, je v následující kapitole věnující se jednotlivým metodám blíže. Odstraněním nežádoucích výstupů jsou získána data odpovídající zadání.

Výhodou metody využívající VBA funkce je, že odstraňuje problémy metody s využitím bariér. Během analýzy je počítáno i s komunikacemi mimo region, takže pohyb po síti není limitován žádnými bariérami. Více tak chování modelu odpovídá realitě, jelikož výběr komunikace pro dopravu není obvykle ovlivněn příslušností komunikace k nějakému regionu (pouze s výjimkou států). Dále je zajištěno, že dostupnost bude vypočtena pro každou lokalitu bez výjimky v případě, že uživatel zadá dostatečné vstupní podmínky pro analýzu. Lokality mohou tak být bez hodnoty dostupnosti jen tehdy, jestli uživatel u analýzy Service Area použije malé hraniční hodnoty pro modelaci polygonů. A v neposlední řadě je vzhledem k principu výpočtu nemožné, aby vymodelované polygony u Service Area přesahovaly do jiného regionu. Pro každou lokalitu je tak zajištěno, že bude mít maximálně jednu hodnotu dostupnosti, a to do svého regionálního centra. Analýza výsledků je jednodušší a rychlejší než u metody s využitím bariér.

Přestože jsou výhody metody evidentní, nesou s sebou jednu nevýhodu. Zajištění všech předností znamená značné zvýšení nároků na čas potřebný k průběhu analýz. S tím souvisí i zvýšení nároků na hardware, jelikož během výpočtu vzniká velké množství dat, které je ve výsledku odstraněno. Rostou tedy nároky na výpočetní výkon a místo na disku.

Náročnost výpočtu závisí na dvou faktorech. Zaprvé je to množství dat vstupujících do výpočtu. Jednak záleží na počtu regionu a tudíž i regionálních center, jednak i na počtu lokalit (u analýzy OD Cost Matrix) nebo počtu a krajních hodnotách modelovaných polygonů (u analýzy Service Area). Druhý faktor úzce souvisí s prvním, avšak týká se pouze modelace polygonů. Náročnost výpočtu je ovlivněna velikostí regionů a z ní vyplývající největší možné hodnoty dostupnosti na hranicích regionů. Neboť je vhodné, aby při analýze nevznikala žádná bílá místa ve výstupu, musí být krajní hodnota dostupnosti posledního polygonu minimálně rovna největší možné hodnotě dostupnosti, jaké lze v některém regionu dosáhnout. Samozřejmě nelze předem odhadnout, jaká ta hodnota je, a proto bude obvykle zadána větší než předpokládaná hodnota. Zároveň pokud bude jen několik regionů velkých a zbylé budou malé, bude krajní hodnota dostupnosti nastavena podle velkých regionů. To povede k tomu, že u většiny malých regionů se bude značná část vymodelovaných polygonů nacházet mimo území regionu. To znamená, že většina vypočtených dat bude následně odstraněna. Čím větší podíl dat

má být odstraněn, tím náročnější se stává výpočet. Sociogeografická regionalizace Česka je názorným příkladem takového rozložení. Na jedné straně je obrovský region Prahy, podle kterého musejí být nastaveny parametry výpočtu, a na druhé straně jsou malé regiony například ve východních Čechách.

Není to jen samotná náročnost výpočtu, co klade omezení. Modely využívající VBA funkce musely být upraveny buď pouze pro shapefily, a nebo pro geodatabáze. Jelikož dnes jsou stále ještě běžnější shapefily a jelikož nelze předpokládat, že běžný uživatel bude schopen převodu mezi jednotlivými formáty, byly zvoleny shapefily. Ovšem shapefile má svoje integritní omezení, která limitují jeho maximální velikost. Každý soubor, jenž je součástí shapefilu, může mít maximální velikost 2 GB (ESRI, 2007). Za podmínek popsaných výše může dojít k tomu, že se u některého souboru shapefilu překročí maximální možná velikost a výpočet neproběhne.

3.3 Python

Poslední nalezenou možností, jak dosáhnout požadovaného cíle, je využití programovacího jazyka Python. Je to syntakticky poměrně jednoduchý jazyk umožňující mimo jiné psaní skriptů pro ArcGIS. Výhoda programovacího jazyka je v tom, že není odkázán jen na nástroje obsažené přímo v ArcGISu. Pomocí kódu je možné si vytvořit vlastní nástroje a také využít podmínek či cyklů v běhu skriptu.

Python byl využit především proto, aby došlo k zefektivnění předchozí metody. Základní myšlenkou bylo provést celý proces metody využívající VBA funkce dávkově. Tedy nepočítat dostupnost najednou pro všechny regiony, ale vždy postupně provádět výpočet pro jednotlivé regiony zvlášť. Zjednodušeně si to lze představit tak, že namísto 1 výpočtu pro 100 regionů, proběhne 100 výpočtů dostupnosti pro každý 1 region.

Pokud jde o samotný výpočet jako takový, je samozřejmě časová náročnost obou případů přibližně stejná. Dá se ovšem předpokládat, že 100 výpočtů pro 1 region bude trvat o něco delší dobu než pouze 1 výpočet pro 100 regionů. Je totiž nutné 100 krát načíst vstupní data a informace a 100 krát uložit výsledek. To se v prvním případě děje pouze jednou. Největší rozdíl mezi oběma metodami ovšem nastává ve zpracování dat před a po samotném výpočtu.

Jelikož se analýza Service Area a OD Cost Matix liší, rozdílný je i samotný přístup řešení u obou analýz. Nejprve tedy bude zmíněna analýza Service Area. Před analýzou dostupnosti musí nejprve u dávkového zpracování dojít k rozdělení vstupních dat tak, aby každý region a regionální centrum měl svoji vlastní vrstvu. Vznikne tak tolik vrstev, kolik je regionů a regionálních center dohromady. V případě zpracování všech regionů najednou k žádné podobné úpravě vstupních dat nedochází. Za to je však nutné data upravit tak, aby posléze bylo možné odlišit, jestli se vygenerovaný polygon nachází již mimo území svého regionu či nikoliv. Tato úprava je méně náročná než ta u dávkového zpracování.

Po analýze následuje vyextrahování těch polygonů, které splňují podmínku cíle práce. Jsou tak vymazány takové polygony, které se nacházejí již za hranicí svého regionu. Pokud je dostupnost počítána se všemi regiony, je tato část nejnáročnější na výpočetní čas i hardware z celého běhu modelu. Při dělení polygonů na vnitřní a vnější ve vztahu ke svým regionům mohou vznikat i miliony drobných polygonů, které je potřeba z valné většiny odstranit. U dávkového zpracování je smazání těch částí polygonů, co jsou mimo region, naopak velice rychlé. Vždy dochází ke smazání polygonů jen pro jeden region. Nevznikají tak zbytečné gigabyty dat, které by musely být posléze smazány. Na závěr skriptu sice musí dojít ke spojení výstupů ze všech analýz do jedné vrstvy, ale tato operace není natolik časově náročná. Spojením všech výstupů je zajištěno, že výsledek bude při shodných vstupních datech stejný jako u předchozí metody.

Analýza OD Cost Matrix má několik odlišností. Hlavně musí dojít k další úpravě na straně vstupních dat. Do analýzy vstupuje kromě regionálního centra ještě vrstva cílových míst (lokalit). I ta musí být rozdělena podle regionů. Vznikne tedy tolik vrstev cílových míst, kolik je regionů, přičemž v každé vrstvě budou všechna cílová místa ležící na území jednoho regionu. Analýza provede výpočet přesné hodnoty dostupnosti z každého výchozího místa (regionálního centra) do každého cílového místa. Logicky tedy není nutné po proběhnutí analýzy mazat nějaká vypočtená data. Díky rozdělení výchozích a cílových míst podle regionů je výsledkem to, co je kladeno jako cíl pro každý region. Stačí tedy na závěr pouze spojit výstupní vrstvy za každý region do jedné vrstvy.

Míra rozdílu mezi metodou využívající VBA funkce a metodou pracující s Pythonem je závislá na vstupních datech a požadavcích uživatele. V textu výše bylo zmíněno, za jakých situací je metoda zpracovaná v ModelBuilderu nevhodná. Právě v těchto případech slouží skript v Pythonu jako lepší alternativa (někdy také jako jediná – viz integritní omezení shapefilu výše). Neznamená to však, že použití skriptu v Pythonu je vždy rychlejší než předchozí metoda. Bližší srovnání a hodnocení je uvedeno v kapitole 5.

KAPITOLA 4

Vytvořené modely

Tato kapitola je zaměřena přímo na vytvořené modely. Věnuje se diskusi nad použitými nástroji a popisu jejich funkce v modelech. Zároveň jsou zde blíže popsány obecné postupy řešení a na nich je vysvětleno, proč byl tento postup upřednostněn před jiným možným.

Všechny vytvořené modely měly jako vstupní podmínku, že musí proběhnout opakovaně i beze změny vstupních parametrů. To především znamená, že za běhu modelů nesmí dojít k žádným změnám ve vstupních datech. Jestliže by se změnila vstupní data, jednak by došlo ke změně vstupních parametrů, jednak by stejný postup nemohl proběhnout vícekrát. Tudíž by s každými vstupními daty byl jen jeden pokus na běh jakéhokoliv modelu či skriptu. Tento problém se týká modelů využívající VBA funkce, neboť pro jejich správný průběh se musí do vstupních dat přidat nové atributy.

Jak bylo již naznačeno výše, při řešení problematiky jsou využity dvě odlišné analýzy dostupnosti. První je Service Area a druhá OD Cost Matrix. Pro každou analýzu byl vytvořen zvláštní model či skript, jelikož jsou samozřejmě odlišné. Rozdíl není jen v samotném výpočtu, ale také ve vstupních a výstupních datech. Nejprve bude popsán kompletně princip metody pro analýzu Service Area a poté se bude práce věnovat odlišnostem u analýzy OD Cost Matrix.

Jelikož není žádná literatura, která by se přímo zabývala podobnou problematikou, jsou všechny konkrétní postupy metod vytvořeny autorem. Metody se dají logicky členit do tří částí. Jedná se o předzpracování vstupních dat, samotná analýza dostupnosti a extrahování patřičných výstupů. Aby byla zachována lepší přehlednost celého popisu metod, bude struktura textu odpovídat tomuto členění. Bude tak i snadnější porovnání jednotlivých metod mezi sebou z hlediska náročnosti každé části a lépe vyniknou jejich vzájemné odlišnosti. Ovšem porovnání nelze provádět pouze na základě délky textu, protože to, co bude jednou podrobně vysvětleno, bude podruhé jen zmíněno. Takové porovnání by tedy bylo zavádějící.

Pořadí popisu metod není nikterak náhodné. Obecně se dá říci, že se postupuje od jednoduššího ke složitějšímu, což má hned několik výhod. Zaprvé věci jednou vysvětlené a popsané u dřívější metody nebude nutné popisovat znovu u dalších. Lze snadno i rozšířit nebo navázat na postupy z předchozí metody. Zadruhé budou patrnější rozdíly v postupu a rostoucí náročnost modelů bude názornější. A zatřetí je toto pořadí účelné, jelikož metody na sebe takto logicky navazují. Následující metoda se snaží odstranit nedostatky předchozí a více se tak přiblížit ideálnímu řešení.

Pro lepší pochopení textu, jsou některé pasáže doplněny obrázky. V případě metod, které byly zpracovány v ModelBuilderu, jsou součástí přílohy jejich schémata vyexportována z ModelBuilderu. Schémata názorně zobrazují kompletní postup dosažení výsledku[6].

4.1 Metoda s využitím bariér

Metoda s využitím bariér je nejjednodušší nalezenou metodou. Pojem nejjednodušší se netýká pouze pohledu implementačního (tedy náročnosti tvorby algoritmu řešení), ale také pohledu logického. Je to první logické řešení, které nejspíš většinu napadne při hledání cesty k dosažení chtěného výsledku. Přesný postup, jakým modely pro obě zkoumané analýzy řeší zadanou problematiku, je pospán v následujícím textu.

4.1.1 Model pro Service Area

Prvním popisovaným modelem je model využívající analýzy Service Area.

Vstupní data a jejich úprava

Do modelu vstupují celkem 4 vrstvy: polygonová vrstva regionů, bodová vrstva regionálních center, liniová vrstva komunikací obsažených v network datasetu a samotný network dataset. Prvním krokem je převedení všech shapefilů na Feature Layer (nástroj *Make Feature Layer*). Mít uložená vstupní data v tomto formátu skýtá několik důležitých výhod. Feature Layer je pouze dočasný formát, uložený v paměti počítače v průběhu modelu. Nevznikají tak fyzicky žádná nová data, která by zabírala místo na disku. Dále některé funkce v prostředí ArcGISu neumožňují jako vstup formát Feature Class, kdežto Feature Layer může být vstupem jakékoliv funkce operující s vektorovými daty. Velmi důležité pro účely běhu modelu je to, že v případě opakovaného spuštění modelu se nahradí data uložená jako Feature Layer novými. Nahrazení původních dat nově vytvořenými probíhá na základě stejného jména, což je zaručeno pevně nastavenými názvy výstupů jednotlivých funkcí. Nemůže tedy nastat situace, že by se neustále hromadila nová data při každém spuštění modelu. Po skončení běhu modelu zůstanou pouze data, která musela být uložena na disk. Ostatní budou smazána z paměti počítače po vypnutí softwaru ArcGIS (ESRI, 2007).

---

[6] Ve schématech se vyskytují 2 značky. Funkce jsou ohraničeny obdélníkem a data oválem. Data označená písmenem P jsou vstupní parametry modelu a jsou volena uživatelem. Šipky mezi jednotlivými elementy určují sled funkcí za sebou. Výstup z jedné funkce je zároveň vstupem do další.

Ještě před průběhem samotné analýzy Service Area je nutné vytvořit novou vstupní vrstvu. Tato nová vrstva není zadána uživatelem jako vstup, ale vytváří se až na základě vstupních dat. Jedná se o bodovou vrstvu bariér. Na každém místě, kde jakákoliv linie z vrstvy komunikací protíná hranici kteréhokoliv regionu, musí být vytvořen bod, který bude součástí vrstvy bariér. Jelikož výpočty dostupnosti v prostředí ArcGISu probíhají pouze po komunikacích, lze takto výpočet snadno zastavit na hranicích regionů.

Vytvoření bodové vrstvy, jejíž každý bod by ležel na průsečíku komunikace a hranice regionu a zároveň žádný takový průsečík by nebyl vynechán, je docíleno nástrojem *Intersect*. Výstupem této funkce je v podstatě geometrický průnik dvou a více vrstev. Bude-li nástroj *Intersect* aplikován na vrstvu komunikací a regionů, bude výstupem jejich průnik. Ovšem průnikem liniové a polygonové vrstvy je logicky opět liniová vrstva obsahující všechny části komunikací ležící v regionech. Proto je velmi důležité nastavit jako výstup z nástroje bodovou vrstvu. Při tomto zvláštním nastavení dojde k vytvoření pouze bodové vrstvy, nikoliv liniové[7].

Během práce s bariérami bylo odhaleno, že lze použít pouze bodovou vrstvu, jež má singlepart geometrii (v atributové tabulce vrstvy musí být v sloupci shape hodnota point). Výstupem z funkce *Intersect* je vrstva s multipart geometrií (v sloupci shape je hodnota multipoint[8]). Před přistoupením k analýze dostupnosti je tedy nutné ještě upravit výstup z nástroje *Intersect* tak, aby bodová vrstva měla singlepart geometrii. K tomuto účelu slouží funkce *Multipart to Singlepart*. Převedená vrstva může být dále použita v analýze Service Area jako bariéry.

Jelikož analýza má počítat dostupnost pouze na území regionů, musí být ošetřen i případ, že vstupní vrstva regionálních center bude obsahovat body nacházející se mimo území regionů. Tyto body nesmí do analýzy vstoupit. Podobně jako u bariér se využila funkce *Intersect*, jejíž vstupem byly zadané vrstvy regionů a regionálních center. Výstupem je pak bodová vrstva center, která obsahuje jen body na území regionů. Neboť nelze předvídat původ dat použitých v modelu, je pro jistotu ještě proveden převod vzniklé vrstvy na singlepart geometrii. Jestliže už tuto geometrii měla, bude výstupem identická vrstva.

Analýza Service Area

Analýza Service Area je základem celého modelu. Sice nezabírá výrazně větší podíl z celkového času běhu modelů, ale prakticky všechny funkce jsou na ni navázány. Nejprve dochází k úpravě vstupních dat, aby vyhovovala potřebám analýzy, a po jejím proběhnutí jsou upravována data z ní vzniklá. Data extrahovaná z analýzy jsou upravována tak, aby splňovala podmínky zadání. Jejich úpravy jsou závislé na struktuře extrahovaných dat. Pokud by byla

---

[7] Na polygon lze nahlížet ze dvou stran. Jeden přístup považuje polygon za ohraničenou plochu (Cromley, 1992). Druhý zase chápe polygon jako uspořádanou soustavu minimálně 3 nekolineárních bodů (Kolář, 2003). Spojením těchto bodů vznikne polygon. Podobně k polygonům přistupuje ArcGIS v případě, že má být výstupem bod. Pak je polygon brán jako soustava úseček mezi body, které polygon vymezují.

[8] Multipart geometrie znamená, že jednomu záznamu v atributové tabulce odpovídá více elementů v shapefilu. Převedením na singlepart geometrii bude každý prvek v shapefilu tvořit samostatný záznam (ESRI, 2007).

struktura dat jiná, musely by být použité funkce pro dosažení cíle odlišné. Nelze proto analýzu dostupnosti jen tak opominout jako něco, co jde jen stěží ovlivnit. Přesný princip výpočtu je před uživateli samozřejmě skrytý a lze na analýzu pohlížet jako na černou schránku. Podrobnějším zkoumáním všech vstupních a výstupních parametrů je však možné získat výsledek, který vyhovuje účelům práce ze všeho nejvíc.

Samotná analýza se neskládá pouze z jediné funkce. Nejprve je nutné pomocí jedné funkce nastavit většinu parametrů analýzy a zvolit network dataset, nad kterým bude analýza probíhat. Dalším nástrojem se přidají do analýzy počáteční body výpočtu a až poté je možné třetím nástrojem celou analýzu spustit.

Funkce *Make Service Area Layer* je první ve sledu funkcí tvořících celou analýzu. Tato funkce umožňuje zadat prakticky všechny parametry výpočtu dostupnosti. Mimo zadaných parametrů je jediným vstupem network dataset. Jelikož je parametrů analýzy celá řada, u většiny z nich bylo nastavení ponecháno jako pevně zadané. K tomuto kroku bylo přistoupeno ze dvou zásadních důvodů. Z pohledu uživatele jde především o to, aby celý proces nastavení výpočtu byl co možná nejjednodušší. Ponechání všech parametrů jako volitelných by kladlo na uživatele nároky na rozhodování o celkové podobě výstupu a v důsledku toho k nechtěným výsledkům. Z pohledu technického by ponechaná volnost nastavení také znamenala, že výstupy z analýzy by byly různé pro různá nastavení parametrů. Každý odlišný výstup analýzy by poté musel být zpracováván odlišně, aby se dosáhlo požadovaného cíle. To bohužel v prostředí ModelBuilderu není možné. Modely jsou pouze sledem funkcí, které očekávají určitý vstup a stejně tak poskytují předem definovaný výstup. Pokud vstup bude mít jinou strukturu nebo hodnotu, než jaká je funkcí očekávána, celý proces bude ukončen. Je tedy nutné, aby parametry měnící nějak strukturu nebo hodnoty výstupní dat byly pevně dány. S ohledem na zmiňované důvody byly ponechány volitelné pouze dva parametry funkce *Make Service Area Layer*. Tyto parametry nelze ani pevně stanovit, jelikož vycházejí z charakteru vstupních dat. Není možné odhadnout jejich hodnoty předem a je nutné, aby byly zadány uživatelem ručně.

První volitelný parametr ovlivňuje, s jakým druhem dostupnosti má model počítat. Možnosti, jaké se nabídnou uživateli, závisí na network datasetu, který uživatel zadal jako vstup do modelu. Každý network dataset obsahuje minimálně jeden atribut, podle kterého je možné počítat analýzy po síti (viz podkapitola 2.3.1). Sice jsou nejčastěji uvažovány atributy času a vzdálenosti, ale nemusí to být pravidlem. Navíc název konkrétního atributu nesoucího potřebné informace k výpočtům je samozřejmě volitelný při tvorbě datasetu. A je to právě název atributu, který uživatel volí jako druh dostupnosti. Z toho vyplývá, že není jiná možnost než manuální volba uživatelem.

Druhý volitelný parametr umožňuje volit hraniční hodnoty modelovaných polygonů. Tento parametr se opět nedá nastavit předem na konkrétní hodnoty, jelikož se model může použít pro různá vstupní data a účely. Například pro úroveň ORP budou potřeba jiné hraniční hodnoty

než pro úroveň celého státu. Některé účely mohou vyžadovat konstantní intervaly hodnot, jiné zase proměnlivé.

Ostatní parametry nelze bez zásahu do samotného modelu měnit. Následuje jejich výčet s informací, jaké nastavení bylo u nich zvoleno. Výpočet dostupnosti probíhá směrem od center. Nulová hodnota dostupnosti je tedy vždy ve výchozím místě výpočtu (regionálním centru) a směrem od něho se zvětšuje. Jsou modelovány zjednodušené polygony. Toto je čistě pragmatické nastavení, aby se zkrátila celková doba výpočtu. Polygony pro každé regionální centrum jsou generovány samostatně. Takže pokud se překrývají polygony od různých center se stejnými hraničními hodnotami, nedojde k jejich spojení v jeden polygon, ale každý polygon bude nezávislý na druhém. Posledním parametrem je, že modelované polygony představují prstence okolo regionálních center. Kromě prvního polygonu mají všechny ostatní polygony dvě hraniční hodnoty dostupnosti. Všechna další detailní nastavení jsou ponechána defaultně.

Výstupem funkce *Make Service Area Layer* je vrstva obsahující informace o všech provedených nastaveních. Tato vrstva je zároveň i vstupem do další funkce, která je součástí analýzy Service Area. Funkce *Add Locations* zajišťuje přidání bodových vrstev do analýzy. Bodovou vrstvou nemusí být jen vrstva výchozích míst analýzy, ale také vrstva bariér, jež byla předtím vytvořena. Jakou úlohu budou během analýzy body plnit, záleží na tom, jaký typ subvrstvy (v originálním nastavení je použit anglický výraz „sub layer“) je zvolen. Na výběr je ze dvou možností: Facilities nebo Barriers. První označuje místa (body), která budou ve výpočtu uvažována jako počátky výpočtu dostupnosti – v tomto případě jde o regionální centra. Druhá možnost samozřejmě označuje bariéry. Nástrojem *Add Locations* bohužel lze přidávat vždy jen jednu bodovou vrstvu. Dokonce i v případě, že by se stejný typ subvrstvy skládal z více bodových vrstev. V modelu proto byla nejprve přidána regionální centra jako Facilities. Vstupem do nástroje byla vrstva vytvořená předchozí funkcí a Feature Layer regionálních center. Výstup z nástroje následně sloužil jako vstup do druhé funkce *Add Locations*, pomocí které se do analýzy přidala bodová vrstva bariér jako typ subvrstvy Barriers. V případě typu subvrstvy Barriers je ve vlastnostech uvedena pro parametr CurbAprroach hodnota „Either side of vehicle“. Jedná se o defaultní nastavení, které je ponecháno, jelikož představuje oboustrannou překážku. Právě tímto nastavením je zajištěno, že se výpočet zastaví na obou stranách bariéry.

Na výše popsaném je názorně vidět, že analýza dostupnosti je řetězcem funkcí, kdy výstup jedné z nich je zároveň vstupem do další a nic neprobíhá odděleně. Posledním článkem v tom pomyslném řetězci je nástroj *Solve*. Vstup je jediný, a to vrstva vzniklá z funkce *Add Locations* po přidání bariér. Účelem nástroje *Solve* je provést samotnou analýzu dostupnosti a vytvořit všechny její výstupy.

Tvorba výstupu

Další postup slouží pouze k vizualizaci výsledku analýzy. Výstup z funkce *Solve* je uložen jako subvrstva Polygons. Nástrojem *Select Data* je možné vzniklé polygony z výstupní vrstvy vybrat. Nyní již stačí pouze vybrané polygony uložit. Žádná další úprava není potřeba, jelikož výpočet byl zastaven přesně v místech, kde již neměl dál pokračovat. Uložení vybraných polygonů do podoby shapefilu je zajištěno funkcí *Copy Features*. Výsledek analýzy se tak nebude již jen udržovat v paměti počítače, ale bude fyzicky uložen na disk.

4.1.2 Model pro OD Cost Matrix

Model pro OD Cost Matrix se od modelu pro Service Area liší jen nepatrně. Přesto v každé ze tří fází lze nalézt drobné rozdíly. Ty vycházejí z toho, že jsou v modelech použité jiné analýzy. Výstupem analýzy Service Area jsou polygony, tedy plošné rozložení dostupnosti v určité oblasti, kdežto výstupem analýzy OD Cost Matrix jsou přesné hodnoty dostupnosti mezi konkrétními místy. Hodnoty dostupností jsou obvykle prezentovány spojnicemi (liniemi) mezi těmito místy, jež mají jako jeden z atributů v atributové tabulce hodnotu dostupnosti.

Vstupní data a jejich úprava

Metoda s analýzou OD Cost Matrix vyžaduje shodné vstupy jako v případě analýzy Service Area. Navíc ovšem vstupuje ještě bodová vrstva cílových míst[9] (v analýze označované jako Destinations). Tak jako v předchozím případě je i tato vrstva převedena nástrojem *Make Feature Layer* na Feature Layer. Aby nebyla do analýzy nahrána místa ležící mimo území regionů, byla vrstva cílových míst upravena tak jako regionální centra. Nejprve byla vybrána místa uvnitř regionů funkcí *Intersect* a následně byla převedena na singlepart geometrii.

Analýza OD Cost Matrix

V této fázi je již odlišností více. Podobně jako u analýzy Service Area i zde je tvořena celá analýza sledem funkcí s tím rozdílem, že první funkcí v pořadí je *Make OD Cost Matrix Layer*. Druhým rozdílem je přidání ještě jednoho nástroje *Add Locations* do celého sledu. Na přesném pořadí ovšem nezáleží. Může se vyskytovat prakticky kdekoliv za prvně jmenovanou funkcí.

Nástroj *Make OD Cost Matrix* nabízí taktéž mnoho parametrů ovlivňující výsledek analýzy. Ovšem v tomto případě se ponechávají všechny parametry defaultně nastavené. Především to znamená, že je nastavené generování přímých linií mezi výchozími a cílovými místy. Dále bude analyzována dostupnost z každého výchozího místa do všech cílových míst, což je důležitý aspekt celé analýzy. Bez jakéhokoliv omezení dojde při analýze k takovému počtu výpočtů dostupnosti, který se rovná součinu počtu výchozích a počtu cílových míst. U metody bariér je

---

[9] Pojmenování vrstvy jako vrstva cílových míst je čistě účelové, aby byla jasná spojitost s anglickým označením typu subvrstvy. Tato vrstva představuje lokality v regionech mimo regionální centra. Pro názornost si je lze na příkladu územněsprávních celků představit jako jednotlivá sídla a vrstva regionálních center představuje například obce s rozšířenou působností. Pak regiony jsou oblasti spravované jednotlivými úřady ORP. Blíže viz poznámka 10.

analýza omezena bariérami, takže k takovému počtu výpočtů nedochází. Důsledky tohoto nastavení budou blíže specifikovány u popisu další metody pro analýzu OD Cost Matrix. Společné oběma analýzám zůstává, že vstupem do této funkce je také pouze network dataset a že se uživatelem zadává druh dostupnosti, který bude analyzován. Hraniční hodnoty polygonů nejsou uživatelem definovány, jelikož k žádné modelaci polygonů nedochází.

Přidaný nástroj *Add Locations* slouží k vložení bodové vrstvy cílových míst do analýzy. Analýza OD Cost Matrix oproti předchozí analýze obsahuje jiné typy subvrstev. Subvrstva Barriers samozřejmě zůstává, jinak by nebylo možné analýzu touto metodou provést. Namísto původního typu Facilities se objevuje typ Origins (výchozí místa) a přibyl ještě typ Destinations (cílová místa). Původně vrstvě regionálních center přiřazený typ Facilities byl tak změněn na Origins a nově přibyvší vrstvě cílových míst byl přiřazen typ Destinations[10]. Přiřazení jednotlivých typů subvrstev bylo prováděno během jejich implementace do analýzy pomocí nástroje *Add Locations*.

Jako poslední nástroj analýzy je zachována funkce *Solve* a její účel je stejný jako v případě analýzy Service Area. Důležité je zdůraznit, že ačkoliv je nastaveno, že výpočet má probíhat mezi každou dvojicí výchozích a cílových míst, ve skutečnosti proběhne výpočet od výchozího místa jen k těm cílovým, která se nacházejí ve stejném regionu. Pomocí bariér je totiž výpočet z výchozího místa omezen na ta cílová místa, která se nachází ve stejném regionu jako ono výchozí místo.

Tvorba výstupu

Výstupem nejsou polygony, ale přesné hodnoty zvoleného druhu dostupnosti mezi výchozími a cílovými místy. Ty jsou ve výstupu reprezentovány liniemi. Nástrojem *Select Data* nejsou logicky vybírány z výstupní vrstvy polygony nýbrž linie. Funkcí *Copy Features* jsou vybrané linie uloženy jako shapefile do uživatelem zadaného umístění na disku.

4.2 Metoda využívající VBA funkce

Metoda využívající VBA funkce vychází z odlišného principu než metoda s využitím bariér, takže není zarážející, že se vzájemně liší. Největší rozdíly mezi oběma metodami jsou v poslední fázi tvorby výstupu. Zatímco u metody bariér nebyl potřeba žádný speciální postup pro zpracování výsledku analýz, u metody využívající VBA funkce je to již nevyhnutelné.

---

[10] Někoho může napadnout, že logičtější by bylo regionálním centrům přiřadit typ subvrstvy Destinations (cílová místa) a vrstvě označované jako cílová místa typ Origins (výchozí místa). V reálném světě většinou člověk za něčím dojíždí do centra. Výchozím místem je tedy pro něj místo v regionu a cílem je regionální centrum. Ovšem pokud by se přistoupilo na tuto logiku, místa označená jako cílová v analýze OD Cost Matrix (regionální centra) by byla zároveň místy výchozími u analýzy Service Area. U Service Area je výpočet dostupnosti prováděn směrem od center, tudíž jsou centra výchozími místy výpočtu. Proto, aby nedocházelo k záměnám a matení čtenáře, bylo přistoupeno k tomu, že regionální centra jsou výchozími body výpočtu i u analýzy OD Cost Matrix.

4.2.1 Model pro Service Area

Nejdříve bude opět popsán model pro analýzu Service Area.

Vstupní data a jejich úprava

Oproti předchozí metodě nevstupuje do analýzy bodová vrstva bariér. Ta nebyla přímo volena uživatelem, ale vytvářela se ze zadaných vstupních dat. Pokud tedy některá použitá vstupní data u předchozí metody sloužila pouze k tvorbě vrstvy bariér, lze je u této metody vynechat. Bodová vrstva bariér vznikala průnikem vrstvy regionů a komunikací, které jsou obsaženy v network datasetu. Regiony jsou samozřejmě nutné i v dalším průběhu modelu, jelikož určují rozsah území pro jednotlivá regionální centra. Vrstva komunikací žádný další účel v modelu neplní, neboť vrstva komunikací je obsažena rovněž v samotném network datasetu. U vstupních dat lze tedy jednu položku vynechat. Do modelu vstupují tři vrstvy: polygonová vrstva regionů, bodová vrstva regionálních center a network dataset. Podmínkou pro vstupní vrstvy je, že to musí být shapefily. Jak bude dále vysvětleno, model počítá s určitou strukturou vstupních dat a tato struktura odpovídá shapefilům. Pokud budou vrstvy regionů a nebo center součástí geodatabází, model neproběhne.

Před samotnou analýzou dostupnosti je nutné vstupní data upravit. Principem metody je nechat proběhnout analýzu bez omezení, takže vymodelované polygony mohou přesahovat své regiony. Posléze se polygony rozřežou podle regionů a části polygonů, které sahají za hranici svých regionů, budou smazány. To je základní princip metody na teoretické úrovni. Na praktické úrovni je třeba zajistit, aby bylo možné od sebe snadno odlišit části polygonů, co se nachází uvnitř a vně svých regionů.

Pro pochopení toho, proč a jak se vstupní data upravila, je potřeba trochu předběhnout k vytváření výstupu. Rozdělení polygonů na vnitřní a vnější (vztaženo vždy k jednomu konkrétnímu regionu) je provedeno funkcí *Intersect*. Výstupem z funkce je polygonová vrstva, která si nese atributy z obou vrstev, které do funkce vstupovaly. Takže na výstupu je jedna vrstva, která obsahuje polygony z analýzy rozdělené podle regionů a zároveň v atributové tabulce má všechny informace z vrstvy polygonů a vrstvy regionů. Představa je taková, že pokud by každý region ve vrstvě regionů měl v atributové tabulce jednoznačný identifikátor a pokud by se tento identifikátor mohl přenést i na každý vymodelovaný polygon z analýzy Service Area, mohou se oba identifikátory porovnat. Identifikátor u polygonů by odpovídal regionu, ze kterého by byly polygony modelovány. Vrstva polygonů vzniklá funkcí *Intersect* by tak obsahovala oba identifikátory. Každý záznam v atributové tabulce (jeden záznam odpovídá jednomu polygonu) by měl uloženy informace o tom, na území kterého regionu se nachází a zároveň z jakého regionu pochází. Ty záznamy, u kterých by se identifikátory rovnaly, by odpovídaly požadavkům cíle práce. Znamenalo by to totiž, že daný polygon se nachází ve stejném regionu, ze kterého byl modelován.

Nyní je potřeba zajistit, aby to bylo proveditelné. Přidat jednoznačný identifikátor regionům není problém. Vlastně každý shapefile obsahuje v atributové tabulce jedinečný identifikátor,

aniž by docházelo k jakýmkoliv úpravám. Je to sloupec FID, který je součástí každé atributové tabulky shapefilů. Ovšem během operací jako *Intersect* dochází ve výstupní vrstvě k přejmenování původního sloupce FID. Atributová tabulka výstupní vrstvy má vlastní sloupec FID, a aby se odlišily původní atributy FID vstupních vrstev, jsou přejmenovány podle názvu vstupních vrstev. Ve výsledku tak může mít atribut FID z vrstvy regionů ve výstupní vrstvě z funkce *Intersect* název například FID_kraje nebo FID_region. Jelikož je vedlejším cílem práce, aby modely fungovaly pro jakákoliv vstupní data, nelze samotný atribut FID z regionů použít. Vrstva regionů může totiž mít jakýkoliv název a podle toho by se odvíjel i odvozený název ve výstupní vrstvě funkce *Intersect*. Jednoduchým řešením je vytvoření nového atributu ve vrstvě regionů. Tento atribut musí mít takový název, který se z největší pravděpodobností nebude ve vrstvě regionů vyskytovat[11]. Hodnoty v novém atributu by se v každém záznamu tabulky rovnaly hodnotám FID záznamu[12, 13]. Tím je zajištěno, že v novém atributu bude mít každý záznam jednoznačný identifikátor. Jako název nového atributu bylo zvoleno region_ID.

Model má být opakovatelný i pro stejná vstupní data, a proto není možné přidat nový atribut rovnou do vstupních dat. Nejprve je vrstva regionů zkopírována pomocí nástroje *Copy Features*. Kopie je následně převedena funkcí *Make Feature Layer* na Feature Layer. Až poté je přidán atribut, aniž by došlo ke změně ve vstupních datech. Nástroj *Add Field* umožňuje do atributové tabulky přidat nový sloupec. Název sloupce byl nastaven na region_ID a jako datový typ byl vybrán text. Odůvodnění, proč byl zvolen právě tento konkrétní datový typ, bude uvedeno dále. Posledním krokem je přiřazení hodnoty FID do sloupce region_ID u každého záznamu v atributové tabulce za použití nástroje *Calculate Field*. První část podmínky pro úspěšné odlišení polygonů uvnitř a vně regionů je splněna.

Pro splnění i druhé části podmínky je nutné znát podobu atributové tabulky modelovaných polygonů. V atributové tabulce je celkem 5 atributů: FID_SAPoly, FacilityID, Name, FromBreak a ToBreak. První atribut představuje identifikátor každého polygonu a je tedy závislý na jejich počtu. Je generován automaticky a nelze ho předem nijak ovlivnit. FacilityID obsahuje číselné hodnoty od 1 do x, kde x představuje počet center, ze kterých je analýza Service Area počítána. Číselná hodnota jednotlivým centrům je přiřazována na základě neznámého vnitřního algoritmu analýzy, a proto ani tento atribut nelze použít. Atributy FromBreak a ToBreak obsahují hraniční hodnoty každého polygonu. Určují dvě hraniční hodnoty dostupnosti, které polygon vymezují. Tyto hodnoty jsou zadávány uživatelem a jsou

---

[11] Pokud by ve vstupních datech byl sloupec se shodným názvem jako nově přidávaný, nešlo by nový sloupec k datům přidat a celá analýza by se zastavila.

[12] Původně se uvažovalo o tom, že by uživatel sám volil sloupec, podle kterého by se jednotlivým záznamům přiřadil identifikátor. Dá se předpokládat, že by to byl sloupec s názvem regionu. Ovšem za situace, kdy dva regiony budou mít stejný název, by mohlo dojít k chybným výsledkům. Proto bylo přistoupeno k použití atributu FID.

[13] Nutnost, aby regiony byly shapefily, je důsledkem tohoto kroku, jelikož data z geodatabází sloupec FID v atributové tabulce nemají. Ty naopak obsahují sloupec OBJECTID. Splnění podmínky jednoznačného identifikátoru si tedy vynutilo volbu mezi shapefily a nebo geodatabází a nakonec byl vybrán shapefile.

tudíž také nepoužitelné. Zbývá pouze atribut Name. Ten obsahuje několik informací najednou. Jednak jsou v něm opět uloženy hraniční hodnoty dostupnosti pro daný polygon, jednak je v něm i název centra, ze kterého je polygon počítán. Tento jediný údaj je volitelný a zároveň není zadáván uživatelem při spouštění analýzy.

Vrstva regionálních center musí mít atribut, jehož hodnoty budou odpovídat hodnotám atributu region_ID u regionů. Každé centrum bude mít v tomto atributu stejnou hodnotu, jakou má region, ve kterém se dané centrum nachází. Pak je možné v analýze atribut označit jako název centra, a tudíž bude obsažen i v atributové tabulce modelovaných polygonů ve sloupci Name. Aby toto bylo možné, musí se zajistit dvě věci. Zaprvé musí být předem znám název nového atributu ve vrstvě center, jinak by nemohl být v modelu přiřazen jako název centra. Atribut navíc musí být ve vrstvě center vždy, nehledě na použitá vstupní data. Zadruhé je potřeba zajistit, aby hodnoty atributu odpovídaly hodnotám atributu region_ID u regionů. Bez těchto dvou podmínek by model nefungoval správně.

Namísto složitého přidávání nového atributu do vrstvy center a následného určování nové hodnoty pro každé centrum zvlášť na základě polohy a příslušnosti k nějakému regionu byl zvolen jiný postup. Vytvoření nové vrstvy center, která už bude mít potřebný atribut obsažený v atributové tabulce. V prvním kroku proběhne již známé převedení na Feature Layer funkcí *Make Feature Layer*. Pak přijde vytvoření nové vrstvy center nástrojem *Intersect*. Jak již bylo zmíněno, nástroj *Intersect* vytváří průnik vrstev a přitom si může nově vzniklá vrstva nést všechny atributy ze spojovaných vrstev[14]. Průnikem polygonové a bodové vrstvy je samozřejmě opět bodová vrstva, která bude geometricky odpovídat původní bodové vrstvě. Dá se tedy říct, že použitím nástroje *Intersect* v tomto případě došlo pouze k doplnění všech atributů z vrstvy regionů do vrstvy center. Jednoduchým postupem byly splněny obě podmínky najednou, jelikož nyní vrstva center obsahuje atribut region_ID. Ten bude ve vrstvě center vždy obsažen, ať budou do modelu vstupovat jakákoliv data, a navíc bude odpovídat hodnotě region_ID u regionů.

Stejně jako u metody s využitím bariér je ještě převedena nově vytvořená vrstva center na singlepart geometrii nástrojem *Multipart to Singlepart*. Jde pouze o preventivní opatření pro případ, že by původní vstupní vrstva regionálních center měla multipart geometrii.

Analýza Service Area

Velká část dalšího postupu již byla naznačena v předešlé fázi. Nicméně je důležité celý postup popsat v kontextu dané fáze. Samotná analýza Service Area se moc neliší od předchozí metody. Opět začíná nástrojem *Make Service Area Layer*, jehož nastavení je identické s předchozí metodou. Vstupem je network dataset a podle atributů, jaké obsahuje, volí uživatel druh dostupnosti. Stejně tak je na uživateli, jaké nastaví hraniční hodnoty dostupnosti. Ostatní

[14] Jaké všechny atributy bude nová vrstva obsahovat, záleží na nastavení. Zvolit lze všechny atributy, všechny atributy kromě FID vrstev nebo pouze FID spojovaných vrstev. Pro účely modelu vyhovují první dvě možnosti a na konkrétní volbě nezáleží.

parametry jsou ponechány tak, jak bylo popsáno výše. Kromě již zmíněných důvodů je to také proto, aby byly mezi sebou obě metody alespoň částečně srovnatelné. Pokud se použije stejné nastavení parametrů u obou metod a použije se shodný network dataset, lze provést porovnání dosažených výsledků obou metod pro jednotná vstupní data (viz kapitola 5).

Dalším krokem je přidání výchozích míst výpočtu. Regionální centra se opět přidávají nástrojem *Add Locations*. Vstupem do nástroje je výstupní vrstva z předchozí funkce a vrstva nově vzniklých center. Vrstvě center se přiřadí jako typ subvrstvy Facilities a je velmi důležité, že jako název center (vlastnost Name) se nastaví atribut region_ID z atributové tabulky center. Žádné bariéry nejsou v této metodě přidávány, takže se další nástroj *Add Locations* nepoužije.

Následuje funkce *Solve*, která uzavírá analýzu. Vstupem do funkce je opět jen výstupní vrstva z předchozí funkce a výstupem je analýza dostupnosti uložená do subvrstvy polygonů.

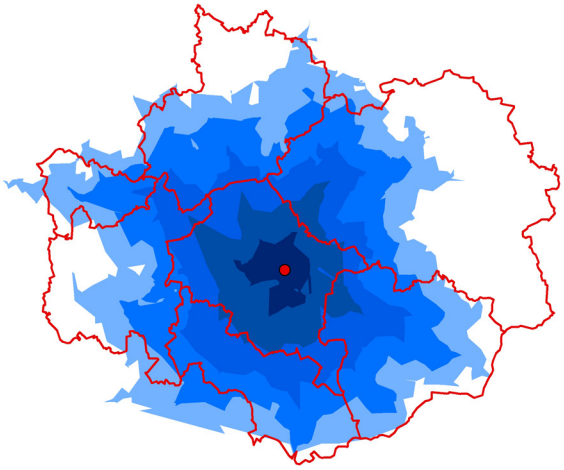
Tvorba výstupu

Poslední fáze tvorby výstupu je časově a hardwarově nejnáročnější částí celého modelu. Podobně jako u předchozí metody jsou nejprve vybrány vytvořené polygony z výstupní vrstvy funkce *Solve*. Subvrstva polygonů je vybrána nástrojem *Select Data*.

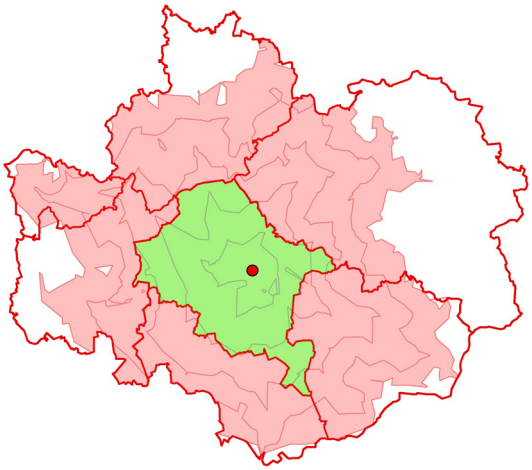
Další krok je příčinou toho, proč je tato fáze nejnáročnější. Musí dojít k rozdělení polygonů na vnější a vnitřní ve vztahu k regionům. Jak bylo již zmíněno, je požadovaného výsledku dosaženo funkcí *Intersect*. Vstupem jsou vybrané polygony a vrstva regionů. Regiony musí obsahovat atribut region_ID, takže vstupuje do nástroje upravená kopie původní vrstvy regionů. Je zachováno defaultní nastavení nástroje, takže ve výstupní vrstvě se vyskytují všechny atributy ze vstupních vrstev. Výstupní vrstva může obsahovat obrovské množství polygonů[15].

Pro lepší představu, jak celý proces rozdělování polygonů probíhá, poslouží následující obrázky. Na prvním je vidět, jak by vypadala situace, kdyby bylo pouze jedno centrum a více regionů. Polygony vygenerované z takového centra přesahují do několika sousedních regionů. Druhý obrázek znázorňuje situaci po aplikaci nástroje *Intersect*. Červeně označené polygony jsou takové, které ve své atributové tabulce mají rozdílný identifikátor v atributu region_ID (původem z regionů) a atributu Name (původem z polygonů, kam byl přejat z regionálního centra). Naopak zeleně označené polygony jsou takové, u kterých se identifikátory shodují. To znamená, že tyto polygony se nachází ve stejném regionu, ze kterého byly původně modelovány.

[15] Celkový počet se nedá předem odhadnout, ale pokud je vstupní situace nevýhodná, může celkový počet polygonů na výstupu přesáhnout i počet několika milionů.



Obr. 11 Polygony analýzy Service Area před použitím funkce Intersect



Obr. 12 Oddělení vnitřních a vnějších polygonů analýzy Service Area u jednoho regionu

Nastíněná situace je do značné míry zjednodušením celého procesu. Ve skutečnosti mohou počty regionů dosahovat i stovek a počet polygonů je vždy násobek počtu regionů[16]. Modelované polygony se často překrývají a mohou zasahovat do velkého počtu sousedních regionů. Z obrovského množství nově vzniklých polygonů je poté potřeba vybrat ty správné.

Hledané jsou polygony, u kterých se rovnají identifikátory v atributech region_ID a Name. V atributu region_ID je identifikátor samotný (příklad: 12). Není u něho žádná další nepotřebná informace. Avšak atribut Name obsahuje kromě identifikátoru ještě hraniční hodnoty polygonu (v atributu Name jsou ještě zapsány hodnoty z atributů FromBreak a ToBreak). Hodnota v atributu Name se skládá z názvu výchozího bodu (region_ID), mezery, dvojtečky, mezery a hraničních hodnot polygonu oddělených mezerami a pomlčkou (příklad: 12 : 0 - 600).

Aby bylo možné porovnat obě hodnoty, je nutné u všech záznamů separovat identifikátor region_ID z atributu Name. Jeho hodnota se musí zkopírovat do nového atributu a pak následně tento atribut porovnat s atributem region_ID. Ještě jedna podmínka je nutná proto, aby se srovnání obou atributů mohlo provést. Oba atributy musí být stejného datové typu. Nelze mezi sebou srovnávat například textový a číselný datový typ. Atribut region_ID je datového typu text, a proto nově vytvořený atribut musí mít stejný datový typ. Nový atribut je vytvořen nástrojem *Add Field* a jako jeho název bylo zvoleno centrum_ID. Vstupem do nástroje je výstupní vrstva všech polygonů z funkce *Intersect*.

Oddělení identifikátoru od zbylých informací v atributu Name je stěžejní pro celý model. Ačkoliv se to může jevit jako drobná úprava, nebylo by možné bez ní provést analýzu touto metodou. K oddělení identifikátoru jsou použity VBA funkce (odtud pramení název metody), které jsou standardní součástí softwaru ArcGIS. V nástroji *Calculate Field* je možné zvolit

[16] V závislosti na počtu hraničních hodnot se modeluje pro každý regionu daný počet polygonů. Takže pokud uživatel zadá 5 hraničních hodnot, pro každý region se vymodeluje 5 polygonů a celkový počet polygonů bude pětinásobek počtu regionů.

výraz, podle kterého budou hodnoty ve zvoleném sloupci vypočteny (centrum_ID). Hodnota atributu centrum_ID se rovná:

Left ([Name], ((InStr ([Name], " ")) – 1))

Je jistě namístě použitý výraz vysvětlit. Základem celého výrazu je VBA funkce Left. Ta slouží k oříznutí jakéhokoliv řetězce zleva. Pod pojmem oříznout je třeba si v tomto případě představit, že se vezme určitá část řetězce z levé strany a tato část se zkopíruje. Funkce je určena dvěma parametry. První určuje řetězec, který má být oříznut, a druhý určuje, kolik znaků se má oříznout. Oba parametry jsou uvedeny v závorce za názvem funkce a vzájemně odděleny čárkou. První parametr je atribut Name a druhý parametr je ((InStr ([Name], " ")) – 1). Funkce InStr určuje, na jaké pozici se v daném řetězci poprvé vyskytuje zadaný znak[17]. Je opět určen dvěma parametry. První definuje, v jakém řetězci má být znak hledán, a druhý, o jaký znak se jedná. Druhý parametr je uváděn v uvozovkách. Oba parametry jsou opět v závorce za funkcí a vzájemně odděleny čárkou. Funkce InStr tedy hledá pozici první mezery v atributu Name. Pozice první mezery v atributu Name je ovšem o jednu větší než kolik je potřeba oddělit znaků zleva. Proto musí být výsledek funkce InStr zmenšen o jedničku[18]. Výrazem pro výpočet hodnoty centrum_ID se volně řečeno říká: Vezmi z atributu Name zleva tolik znaků, kolik odpovídá pozici první mezery v tomto atributu zmenšené o jedničku (ESRI ArcGIS Resource Center, 2010).

Výběr polygonů, u kterých se rovná hodnota atributů centrum_ID a regiony_ID, je proveden nástrojem *Make Feature Layer*. Tento nástroj totiž neslouží pouze k převodu dat na Feature Layer, ale zároveň umožňuje pomocí SQL zadat podmínku výběru. Takže se nejprve provede výběr záznamů, které vyhovují zadané podmínce, a poté se vytvoří nová vrstva Feature Layer obsahující pouze vybrané záznamy. Podmínkou výběru záznamů bylo, že se hodnota centrum_ID musí rovnat hodnotě region_ID.

Poslední použitou funkcí v celém modelu je funkce *Dissolve*. Během použití nástroje *Intersect* při rozdělování polygonů na vnější a vnitřní došlo k rozpadu polygonů se stejnými hraničními hodnotami i uvnitř regionů. To je dáno tím, že se navzájem rozdělují i překrývající se polygony dostupnosti. Vysvětluje to, proč vzniká tak obrovské množství polygonů v průběhu funkce *Intersect*. Namísto jednoho polygonu pro jeden rozsah hraničních hodnot v rámci jednoho regionu je těchto polygonů více. Nástroj *Dissolve* slouží ke spojení polygonů, které mají shodné atributy Name, FromBreak a ToBreak, do jednoho. Vznikne tak pro každý region

---

[17] Někoho možná napadne, že není potřeba používat funkci InStr a že místo toho lze pevně nastavit počet znaků k ořezu. Ovšem počet znaků tvořících atribut region_ID je závislý na počtu regionů. Region_ID svým způsobem představuje pořadí regionu v atributové tabulce. Pokud bude regionů více než 100 a zároveň méně než 1000, bude region_ID tvořen jedním, dvěma nebo třemi znaky. Proto nelze zadat pevný počet znaků, co mají být oříznuty, ale musí se každý záznam řešit odděleně.

[18] Na použitém příkladu struktury atributu Name (12 : 0 - 600) je první mezera na 3. pozici. Výsledkem funkce InStr by bylo číslo 3. Ovšem je potřeba oříznout pouze první 2 znaky zleva, a proto je od výsledku funkce odečtena jednička.

jeden polygon s danými hraničními hodnotami namísto několika. Navíc jsou nástrojem vynechány všechny ostatní atributy z atributové tabulky, takže výsledek modelu není zatěžován množstvím zbytečných informací. Výstup z funkce *Dissolve* je uložen na uživatelem definované umístění na disku.

4.2.2 Model pro OD Cost Matrix

Model pro OD Cost Matrix vychází z modelu pro Service Area. Rozdíly mezi oběma modely jsou opět důsledkem použití odlišné analýzy.

Vstupní data a jejich úprava

Podobně jako u metody s využitím bariér, i zde přichází do analýzy oproti Service Area jedna bodová vrstva navíc. Jde opět o vrstvu cílových míst. Na vstupu do modelu tak jsou network dataset a shapefily regionů, výchozích a cílových míst.

Vstupní shapefily je také nutné upravit tak, aby později ve fázi tvorby výstupu mohly být vybrány takové linie, které odpovídají řešení problematiky. Ovšem analýza OD Cost Matrix se od Service Area na výstupu liší. Subvrstva linií vzniklá analýzou si nese s sebou v atributové tabulce celkem 5 atributů: Name, OriginID, DestinationID, DestinationRank a Total_atribut[19]. Hodnoty OriginID a DestinationID jsou přidělovány výchozím a cílovým místům podle vnitřního algoritmu analýzy (podobně jako FacilityID u Service Area) a není možné je použít pro identifikaci hledaných linií. DestinationRank představuje pořadí cílového místa mezi všemi cílovými místy přiřazenými k jednomu výchozímu místu. Cílové místo s nejnižší hodnotou dostupnosti má DestinationRank roven jedné (ESRI, 2006). Hodnotu DestinationRank také nelze ovlivnit. Total_atribut představuje hlavní výsledek analýzy, jelikož se jedná o konkrétní hodnotu dostupnosti mezi výchozím a cílovým místem. Opět jediným ovlivnitelným atributem je Name. Jeho struktura se ovšem liší oproti jeho ekvivalentu u analýzy Service Area. Hodnoty v atributu Name mají podobu: název výchozího místa, mezera, pomlčka, mezera a název cílového místa. Pokud by názvy výchozích a cílových míst odpovídaly názvům regionů, stačilo by pouze vybrat takové linie, které mají oba názvy shodné. Shodné názvy by totiž znamenaly, že linie spojuje výchozí a cílové místo, které leží ve stejném regionu. Příslušné vrstvy musí mít v atributové tabulce sloupec, jehož hodnoty budou jednoznačně určovat region, ve kterém se nachází. A tento sloupec pak musí být přiřazen do analýzy jako název výchozích a cílových míst.

Patřičná úprava vstupních dat je dosti podobná té u analýzy Service Area. Nejprve se k vrstvě regionů opět přidá atribut region_ID datového typu text, jehož hodnota v každém záznamu bude rovna FID záznamu. Postup je identický z předchozím modelem. Vstupní vrstvy výchozích a cílových míst se upraví tak, že se vytvoří nové vrstvy pomocí nástroje *Intersect*.

[19] Název posledního atributu je variabilní a závisí na názvu atributu z network datasetu, který byl použit pro analýzu dostupnosti. Pokud například v network datasetu je atribut s názvem cas, název posledního atributu v subvrstvě linií by byl Total_cas.

Oproti původním vrstvám tak budou obě obsahovat ještě všechny atributy z upravené vrstvy regionů včetně nového atributu region_ID. Posledním krokem je převedení obou vrstev na singlepart geometrii.

Analýza OD Cost Matrix

Nastavení parametrů analýzy přes funkci *Make OD Cost Matrix Layer* je identické jako u předchozí metody. Jediným vstupem je network dataset a na jeho základě uživatel volí druh dostupnosti, jaký bude analýzou počítán. Všechna ostatní nastavení jsou nechána defaultní. Důležité je nastavení, že analýza bude počítána z každého výchozího místa do všech cílových míst. Jelikož není analýza nijak omezena, dojde k modelaci dostupnosti mezi každou dvojicí cílových a výchozích míst. V praxi to může znamenat i statisíce výpočtů[20].

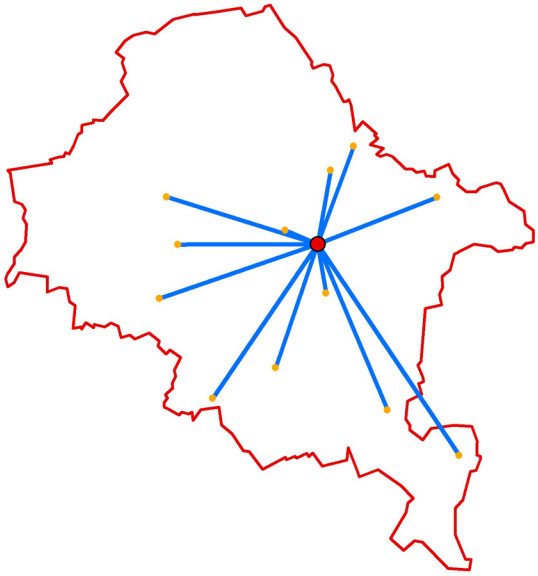
Rozdíly mezi metodou bariér a metodou využívající VBA funkce nastávají při přidávání výchozích a cílových míst do analýzy. V nástroji *Add Locations* se u obou subvrstev musí nastavit ve vlastnostech jako parametr Name atribut region_ID. Tak bude zajištěno, že se budou moci od sebe odlišit linie spojující místa v rámci jednoho regionu od linií spojující místa z různých regionů. Samozřejmě není přidávána žádná subvrstva bariér. Analýza je zakončena nástrojem Solve.

Tvorba výstupu

Základní princip tvorby výstupu byl nastíněn ve fázi úpravy vstupních dat. Nástrojem *Select Data* jsou vybrány linie vzniklé analýzou. Podoba atributu Name je pro každou linii následující: název výchozího místa, mezera, pomlčka, mezera a název cílového místa. Jelikož byl jako název obou míst určen atribut region_ID, příkladem podoby atributu Name může být: 12 – 34. Cílem je nalézt všechny záznamy, které mají hodnoty v atributu Name shodné (např. 12 – 12). Porovnání obou hodnot a tedy zjištění, jestli se rovnají nebo ne, je možné jedině, jestli budou obě hodnoty oddělené a stejného datového typu. Podobně jako u analýzy Service Area je potřeba zkopírovat hodnoty do nových atributů.

Zcela tak odpadá nástroj *Intersect*, který u předchozí analýzy hrál velmi důležitou roli. Jeho využití by bylo nesprávné. Nelze totiž předpokládat, že přímá linie spojující dvě místa v jednom regionu, se bude celá vyskytovat v daném regionu (viz obr. 13). Pokud by se použila podobná logika uvažování jako u analýzy Service Area, musely by být smazány všechny linie, které by po aplikaci nástroje *Intersect* byly rozděleny. Tímto pravidlem by ale došlo k vymazání i hledaných linií. Jak ukazuje zmíněný obrázek, vede jedna linie přes hranici regionu, a přesto spojuje dvě místa ve stejném regionu.

[20] Pokud by byla počítána dostupnost ze všech obcí s rozšířenou působností (205) do všech obcí (6 249), muselo by být provedeno 205 * 6 249 = 1 281 045 výpočtů (Český statistický úřad, 2009).



Obr. 13 Ukázka výstupu analýzy OD Cost Matrix

Přidání atributů do atributové tabulky linií a jejich výpočet byl řešen postupně. Nejprve se nástrojem *Add Field* přidal první sloupec. Jako název bylo zvoleno start_ID a jako datový typ byl nastaven text. Do sloupce start_ID jsou zkopírovány levé hodnoty z atributu Name. Levá hodnota odpovídá názvu regionálního centra, proto také označení start_ID. Zkopírování je provedeno funkcí *Calculate Field* a výraz pro výběr hodnoty identifikátoru vlevo je definován VBA funkcemi. Podoba výrazu je téměř identická tomu, co byla použita u analýzy Service Area:

Left ([Name], ((InStr ([Name], "-")) - 2))

Jediný rozdíl je v tom, že funkce InStr určuje pozici pomlčky místo první mezery a v důsledku toho se musí od hodnoty funkce InStr odečíst dvojka namísto jedničky. Pomlčka místo mezery byla zvolena proto, že se jedná o centrální znak atributu Name. Jiný důvod není, jelikož by původní výraz bez úprav fungoval naprosto stejně.

Když je zkopírována hodnota vlevo, přidá se druhý sloupec pro hodnotu vpravo. Ve vlastnostech funkce *Add Field* byl určen název sloupce jako cil_ID a jeho datový typ nastaven na text. Zkopírování pravé hodnoty z atributu Name je principielně provedeno shodně jako u levé hodnoty. Postup se liší pouze v použitém výrazu pro výběr u funkce *Calculate Field*. Hodnota cílových míst je vybrána takto:

Right ([Name], (Len ([Name]) - (InStr ([Name], "-")) - 1))

Jelikož se kopíruje hodnota napravo, je logické, že místo funkce Left je použita VBA funkce Right. Pro větší přehlednost bude smysl výrazu ještě jednou stručně popsán. Stejně jako Left je i funkce Right určena dvěma parametry. První určuje řetězec, který má být ořezán zprava, a druhý určuje, kolik znaků má být ořezáno. Zatímco první parametr zůstává stejný, druhý se od předchozích liší. Funkce Len v druhém parametru slouží k získání celkové počtu znaků

v zadaném řetězci. Tímto řetězcem je hodnota atributu Name. Takže od celkového počtu znaků v atributu Name se odečte pozice pomlčky a poté se ještě od celého výsledku odečte jednička (ESRI ArcGIS Resource Center, 2010). Celkový výsledek pak určuje, kolik znaků z atributu Name se má zprava zkopírovat.

Jistě se nabízí otázka, proč vůbec byla použita funkce Len. Vysvětlení je celkem jednoduché. Pozice pomlčky, která slouží jako opěrný bod při určování počtu znaků ke zkopírování, se mění. Ovšem mění se pouze v závislosti na počtu znaků před ní. To znamená, že pokud by před pomlčkou bylo jednociferné číslo, je pomlčka na 3. pozici. U dvojciferného čísla by byla na 4. pozici a tak dále (např. 7 – 13, 12 – 13). Počet znaků za pomlčkou vliv na její pozici nemá. Takže pokud by bylo za pomlčkou jednociferné nebo pěticiferné číslo, byla by pomlčka na stejné pozici. To samozřejmě platí v případě, že v obou příkladech bude před pomlčkou číslo mající stejný počet cifer (např. 12 – 1, 12 - 87354). Nyní, kdyby se použila funkce Right bez přidané funkce Len, zkopírovalo by se zprava tolik znaků, kolik odpovídá pozici pomlčky zmenšené o jedničku. Pomlčka je na 4. pozici, takže by se zkopírovaly poslední 3 znaky. V prvním případě by bylo zkopírováno „– 1“, což je moc znaků, a v druhém případě „354“, což je zase málo znaků. Aby bylo možné zkopírovat jen takový počet znaků, který odpovídá počtu cifer v pravé hodnotě, je třeba vědět, kolik znaků se nachází za pomlčkou. Funkcí Len se zjistí počet znaků v celém řetězci. Odečtením pozice pomlčky od této hodnoty zůstane počet znaků odpovídající délce pravé hodnoty a mezery před ní. Závěrečným odečtením jedničky je odstraněna přebývající mezera a zůstává pouze počet znaků rovnající se počtu cifer pravé hodnoty.

V této fázi, kdy jsou obě hodnoty odděleny a každá je uložena v jiném atributu, je možné je vzájemně porovnat. Nástrojem *Make Feature Layer* se pomocí jednoduchého SQL dotazu (start_ID = cil_ID) vyberou takové záznamy, u kterých se obě hodnoty rovnají. Zároveň s tím se i vytvoří nová Feature Layer obsahující vybrané záznamy. Výsledek je ještě potřeba převést z paměti počítače na shapefile, k čemuž slouží nástroj *Copy Features*. Výstup z tohoto nástroje je uložen na uživatelem definované umístění na disku a model úspěšně ukončen.

4.3 Metoda využívající programovací jazyk Python

Metoda využívající Python je poslední nalezenou metodou. Na rozdíl od předchozích není do softwaru ArcGIS implementována přes ModelBuilder, ale je využit programovací jazyk Python. Ten je standardní součástí softwaru a obsahuje modul umožňující práci s GIS funkcemi.

Metoda je založena na předpokladu, že dávkové zpracování je rychlejší než zpracování všech vstupních dat najednou, jak tomu bylo v předchozí metodě. Zjednodušeně řečeno se analýza dostupnosti provede pro každý region odděleně a poté se výsledky ze všech regionů spojí do jednoho výstupu. U každé analýzy je důvod k tomuto kroku jiný. V případě analýzy Service Area je to především funkce *Intersect*, která se při nevhodné kombinaci vstupních dat

a parametrů stává neúměrně časově a hardwarově náročnou. V extrémních případech může dojít i k tomu, že není možné model dokončit, jelikož by výstupní vrstva z funkce překročila integritní omezení shapefilu. U analýzy OD Cost Matrix je zase důvodem provádění analýzy z každého výchozího místa do všech cílových. Omezování počtu cílových míst, do kolika se má výpočet z výchozího místa provádět, je metodologicky chyba. Pro jakoukoliv konkrétní hodnotu může nastat případ, že v regionu bude cílových míst více. Ideální by tedy bylo, aby výpočet z každého regionálního centra probíhal jen do cílových míst ve stejném regionu.

Nejjednodušší způsob, jak dosáhnout dávkového zpracování dat, je rozdělit vstupní data po regionech a pak na každý soubor vstupních dat aplikovat předchozí postup. Poměrně drobnou úpravou předchozí metody by vznikla metoda jiná. Ovšem, jak bude názorně ukázáno dále, po rozdělení vstupních dat po regionech lze předchozí modely zjednodušit a dosáhnout tak ještě větší efektivity.

V popisech jednotlivých modelů nebudou podrobně rozebírány jednotlivé části kódu, a to ze dvou důvodů. Zaprvé by podrobný popis všeho, co se v kódu vyskytuje, byl příliš obsáhlý a neúměrně by tak narostl objem celé práce. Zadruhé by to nemělo moc velký smysl. Pro laiky, kteří neumějí programovat, by byl takový popis nejspíš nesmyslný a zbytečně zdlouhavý. Programátoři, kteří Python ovládají nebo znají alespoň základní syntaxi, raději nahlédnou do samotného skriptu, který je také součástí práce. V něm odhalí použitý způsob rychleji a navíc je celý skript řádně okomentovaný, aby bylo jasné, k čemu jaká řádka kódu slouží. Popis metody se proto omezí na její samotný princip a důraz bude převážně kladen na použité GIS funkce. Výjimku budou tvořit pasáže s nekonvečními řešeními částí problémů, které budou podrobněji popsány i v samotném textu práce.

Skripty nelze jako to bylo u ModelBuilderu lehce rozdělit do tří fází. ModelBuilder provádí řešení lineárně, bez cyklů a opakování. U popisu skriptů z Pythonu je sice dělení na tři fáze zachováno, ale není to úplně vhodné. V průběhu skriptu totiž dochází k neustálému opakování jednotlivých fází pro každý region. Takže po skončení třetí fáze jednoho regionu se skript vrátí k první fázi dalšího. Celý proces tedy neprobíhá tak, že po skončení jedné části se k ní už nevrací, jak tomu bylo u předchozích metod.

Než je možné přistoupit k samotnému řešení v programovacím jazyce Python, je vhodné zmínit základy společné pro obě analýzy. V první řadě jde o obecná nastavení skriptů. Dále se zvláštní pasáž věnuje specifiku skriptu pro verzi softwaru ArcGIS 9.2[21]. Od poslední dostupné verze v době psaní práce (ArcGIS 9.3) se liší ve způsobu, jakým uživatel může zadat druh dostupnosti použitý v analýzách. Poslední společná část popisuje postup implementace skriptů do prostředí softwaru ArcGIS.

---

[21] Snahou bylo vytvořit skript, který by fungoval pod více verzemi softwaru ArcGIS. Především jde o to, aby jeho použití nebylo vázáno na jednu konkrétní verzi a byl tak přístupný co možná nejširšímu okruhu potenciálních uživatelů.

4.3.1 Obecná nastavení a příprava běhu skriptů

Není toho mnoho, ale bez těchto nastavení by skript nefungoval. V první řadě se musí načíst moduly, které budou v průběhu skriptu použity. Pro oba typy analýz jsou využívány funkce ze čtyř modulů: os, string, sys a arcgisscripting. Posledně vypsaný je z hlediska aplikace funkcí softwaru ArcGIS nejdůležitější. Všechny funkce jsou totiž v Pythonu zprostředkovány přes geoprocesingový objekt, který je vytvořen právě z tohoto modulu. Bez něj by nebylo možné GIS funkce v Pythonu použít. Druhým krokem je logicky vytvoření právě zmíněného geoprocesingového objektu, který je prostředníkem mezi Pythonem a GIS funkcemi.

S vytvořeným objektem se mohou nastavit vnitřní parametry skriptu. Nejprve je povoleno přepisování výstupů. Toto nastavení je důležité v případě, že se skript bude spouštět opakovaně, aniž by byl software ArcGIS vypnut. Vymazání paměti je totiž provedeno až po vypnutí ArcGISu a do té doby jsou vypočtené proměnné uloženy v paměti (ESRI, 2007). Takže při dalším spuštění skriptu musí být přepsány novou hodnotou proměnných, jinak by se skript zastavil.

Ještě před spuštěním první funkce analýzy je vhodné zkontrolovat, jestli je nainstalována a povolena extenze Network Analyst v ArcGISu. Analýza dostupnosti je součástí této extenze a bez ní nemůže skript proběhnout. Kontrola je samozřejmě opatřena podmínkami pro případ, že by extenze nebyla povolena nebo dokonce nainstalována. Hned v úvodu skriptu je tedy možné vidět velkou výhodu programovacího jazyka oproti ModelBuilderu, kterou je stanovování podmínek. Podle výsledku podmínky se pak skript vydává rozdílnými cestami, případně je předčasně ukončen. Samozřejmostí je i informování uživatele o tom, co právě skript počítá, případně proč se výpočet zastavil. Něco takového v ModelBuilderu není možné.

Následuje definice většiny proměnných použitých v skriptu. Každá proměnná musí být před použitím ve funkci definována. V seznamu proměnných jsou jednak ty, které zadává uživatel (cesty ke vstupním datům, hraniční hodnoty polygonů dostupnosti atd.), a jednak proměnné sloužící jako vstup a výstup funkcí. Definice proměnných zadávaných uživatelem se rovnají výrazu: gp.GetParameterAsText(X). Část výrazu „gp“ představuje geoprocesingový objekt, GetParameterAsText je název funkce a v závorce místo X se vyskytují celá čísla od nuly výš. Použitá funkce zajišťuje načtení proměnné uvedené uživatelem při spuštění skriptu jako řetězec. Hodnota v závorce zase určuje pořadí, jaké má ta konkrétní proměnná v úvodním okně spuštěného skriptu. Nula představuje 1. položku, jednička 2. položku atd. Pokud tedy v okně po spuštění skriptu bude jako 1. položka vrstva regionů, musí být v definici proměnné pro načtení regionů v závorce hodnota nula.

Uvnitř skriptu jsou ještě definovány další proměnné, ale ty jsou často definovány na základě jiných proměnných. To znamená, že nová proměnná vznikne úpravou jiné proměnné. Úpravou se většinou rozumí přidání řetězce znaků k původní proměnné. Během vývoje skriptu docházelo často ke změnám v definicích funkcí i proměnných, které do nich vstupovaly, a proto tyto proměnné byly ponechány u funkcí, kterým náleží. Jejich úprava byla pak rychlejší a navíc

spolu tvoří jeden celek. Proměnné definované na začátku skriptu jsou na rozdíl od proměnných uvnitř skriptu používané ve více funkcích a prostupují často velkou částí skriptu. Z toho důvodu bylo lepší je definovat na začátku skriptu.

Posledním společným krokem obou skriptů je vytvoření složky pro ukládání mezivýsledků (složka s názvem tmp). Složka je vytvořena na disku tam, kde se nachází spuštěný skript. Nejprve se jednoduchou funkcí zjistí, v jaké složce se skript nachází, což se uloží do proměnné. Poté je funkcí *Create Folder*, jenž je součástí ArcGISu, vytvořena složka tmp v uloženém umístění. Složka tmp je následně definována jako pracovní prostor skriptu (tzv. workspace). Jestliže si skript potřebuje uložit některá data jako mezivýsledek, použije k tomu právě tuto složku. Poslední funkcí celého skriptu je složka tmp smazána i s celým jejím obsahem. Tímto elegantním řešením je možné mít pod kontrolou veškeré výstupy ze skriptu a nedochází k nechtěnému hromadění dat neznámo kde. Po průběhu skriptu se stávají ta data neužitečná a jen by zbytečně zahlcovala diskový prostor počítače.

Funkce *Create Folder* je v softwaru ArcGIS součástí toolboxu Data Management Tools. Před použitím jakékoliv funkce ve skriptu je nezbytné definovat, ve kterém toolboxu se následující funkce nachází. V opačném případě nebude funkce nalezena a skript se zastaví. Proto před samotným vytvořením nové složky muselo být určeno, že nyní se mají brát funkce z toolboxu Data Management Tools. Pokud je více funkcí z jednoho toolboxu za sebou, postačí toolbox definovat pouze před první z řady funkcí. Dokud nebude definováno jinak, bude skript funkce hledat v definovaném toolboxu.

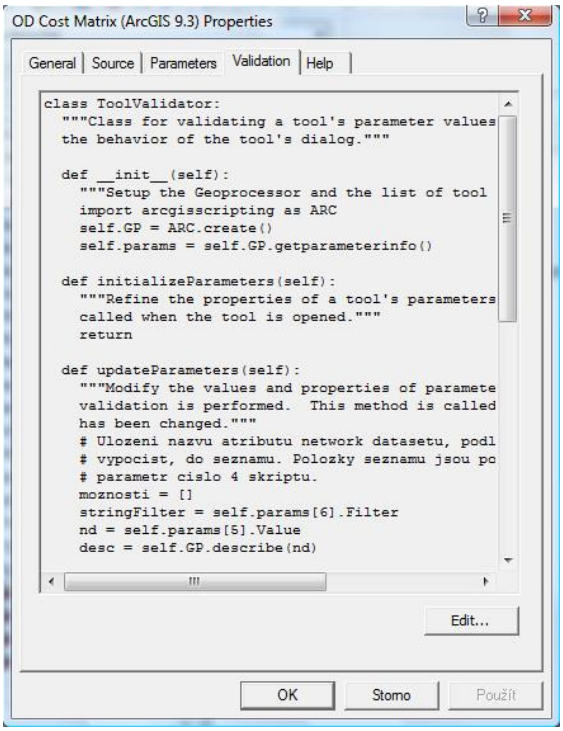
4.3.2 Volba parametru druh dostupnosti

Druh dostupnosti je parametr, který je nastavován uživatelem podle toho, jaký network dataset použije. Je vyloučeno, aby byl nastaven napevno ve skriptu, jelikož v každém network datasetu se identický atribut určený k analýze může jmenovat jinak. Uživatel rozhoduje o tom, jaký atribut bude použit, a nabídnuty mu mohou být jen takové možnosti, které se nachází v načteném network datasetu.

Taková je teorie. V ModelBuilderu je realizace poměrně snadná. Ve funkci *Make Service Area Layer* či *Make OD Cost Matrix Layer* se určí, že parametr druh dostupnosti bude proměnná. Poté, co je u proměnné zaškrtnuta možnost „Model Parameter“, stává se proměnná parametrem modelu. Automaticky se pak při spuštění modelu objeví druh dostupnosti jako možnost volby pro uživatele. Navíc je celý systém přidávání proměnných do modelu navržen tak, že lze druh dostupnosti zvolit až podle přidaného network datasetu.

Pokud se chce podobného výsledku dosáhnout pomocí skriptu, musí být každý krok naprogramován, nic se nevytváří automaticky. Jediná odlišnost mezi verzemi skriptu pro ArcGIS 9.2 a ArcGIS 9.3 je právě ve způsobu, jak zvolit druh dostupnosti k analýze. V případě ArcGISu verze 9.3 je situace poněkud jednodušší. Mezi proměnnými definovanými na počátku skriptu je jedna s názvem atribut. Ta je rovna výrazu gp.GetParameterAtText(4),

takže 5. parametr zadávaný uživatelem je právě druh dostupnosti (toto platí u analýzy Service Area, pro OD Cost Matrix je to 7. parametr). Při importu skriptu do softwaru ArcGIS (blíže viz dále) bude název 5. parametr vstupující do skriptu „Druh dostupnosti“ a bude datového typu String. Po dokončení importu skriptu se zobrazí jeho vlastnosti. Ke všem záložkám, které byly nastaveny během importu, přibyla jedna nová: Validation (viz obr. 14). Ta během importu není dostupná. Vložením správného kódu[22] (v jazyce Python) je možné najít všechny atributy z vloženého network datasetu, které mohou být použity k analýze. Získané atributy jsou uloženy jako nabídka do 5. parametru. Uživatel si pak může zvolit parametr.



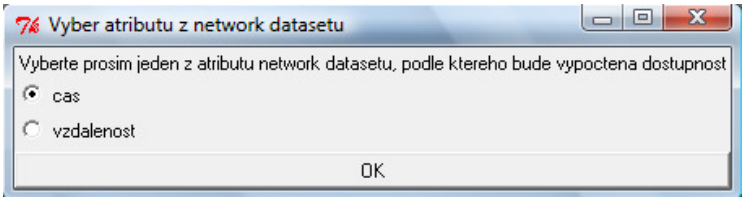
Obr. 14 Záložka Validation u vlastností skriptu implementovaného do toolboxu

Verze ArcGISu 9.2 ovšem ToolValidator neobsahuje, a tak není možné tento postup aplikovat. Řešení spočívá v tom, že se atribut z network datasetu volí až po spuštění analýzy. Zjednodušeně řečeno se po spuštění analýzy jako první prozkoumá zadaný network dataset a naleznou se všechny atributy, které umožňují provést analýzu. Poté se vytvoří okno, ve kterém budou všechny nalezené atributy vypsané a jeden z nich se bude moci zvolit pomocí radiobuttonu (viz obr. 15). Okno je v Pythonu vytvořeno pomocí modulu Tkinter (Programujte.com, 2009)[23]. Zvolený atribut se uloží do proměnné a dosadí do funkce *Make Service Area Layer* či *Make OD Cost Matrix Layer*. Stejný způsob volby atributu by se dal volit i pro ArcGIS 9.3. Ovšem toto řešení má dvě nevýhody. Podmínkou správného fungování je, že

[22] Zmíněný kód je součástí práce jako příloha.

[23] Pro zajímavost je stejný modul použit i pro samotné uživatelské rozhraní Pythonu.

kromě samotného skriptu musí být ve stejné složce i druhý skript, který má za úkol vyvolat okno s radiobuttony. Pokud tomu tak nebude, skript neproběhne. Druhou nevýhodou je, že okno se objeví na obrazovce až po určitém čase od spuštění analýzy[24]. Pokud uživatel okamžitě po spuštění skriptu odejde od počítače, skript se po chvíli zastaví a bude čekat na zadání názvu atributu. Ve verzi 9.3 je přistoupeno k řešení přes ToolValidator, jelikož je to elegantnější a pro uživatele pohodlnější řešení.



Obr. 15 Okno pro výběr druhu dostupnosti v ArcGISu 9.2

Při bližším zkoumání lze dojít k závěru, že obě použitá řešení jsou vlastně stejná. Odlišnost obou přístupů je pouze v tom, že u verze 9.2 proces získání použitelných atributů probíhá až po spuštění analýzy, kdežto u verze 9.3 je stejný postup aplikován už při zadávání vstupů uživatelem.

4.3.3 Implementace skriptu do prostředí softwaru ArcGIS

Implementace skriptu je až poslední krok v celém postupu. Přesto je vhodné jej zmínit už nyní, jelikož je postup společný pro oba modely. Skript může být do prostředí ArcGIS vložen jen prostřednictvím toolboxu. Lze využít vlastní toolbox, který byl vytvořen pro tvorbu modelů.

Než je skript přidán do toolboxu je vhodné ho přemístit do stejné složky, ve které se toolbox nachází. Během importu je totiž hned v prvním okně možnost zaškrtnout relativní cestu ke skriptu. Pak tedy stačí při přenosu do jiného počítače mít toolbox a skripty v jedné složce a vše bude fungovat. V prvním okně importu se volí název skriptu a jeho druhý název, který se bude zobrazovat v toolboxu. Lze připojit i jeho krátký popis.

V dalším okně je volena cesta ke skriptu. Pokud by se nenastavila relativní cesta, bylo by nutné při každém přenášení z jednoho počítače do druhého měnit nastavení této cesty. To proto, že by se s největší pravděpodobností skript nenacházel ve stejné adresářové struktuře jako u předchozího počítače. Poslední okno umožňuje nastavit vstupní parametry skriptu.

Vstupní parametry jsou hodnoty, které volí uživatel po spuštění skriptu. Na jejich základě je ovlivněn výsledek skriptu. Vše, co zadává uživatel, je nastaveno v tomto posledním okně. Jednak se určuje název parametru, který se uživateli zobrazí a podle kterého bude zadávat vstupní hodnoty, a jednak i jeho typ. Pod typem je nutné rozumět cokoliv od obyčejného řetězce přes atribut v atributové tabulce až po shapefily nebo třeba network dataset. Pořadí, v jakém

[24] Okamžitě se objevit nemůže, protože se nejprve musí načíst moduly, vytvořit geoprocesingový objekt a musí se projít network dataset. Doba, po jaké se okno objeví, je odvislá od výkonu počítače.

jsou jednotlivé vstupní parametry ve skriptu nastaveny, je nesmírně důležité. Jak již bylo zmíněno výše, na začátku skriptu jsou proměnné figurující jako vstup do celé analýzy definovány výrazem: gp.GetParameterAsText(X). Číslo X určuje, jaký vstupní parametr dosadit z těch, co jsou navoleny při importu skriptu. Pokud tedy network dataset bude první parametr, co uživatel volí, musí být místo X nula. Pokud by druhým parametrem byly regiony, musí být proměnná pro regiony definována s číslem jedna místo X atd. Pokud se některé pořadí prohodí, skript neproběhne. Proměnné budou obsahovat jiné hodnoty, než jaké jsou očekávány funkcemi. Kdyby došlo k záměně, která umožní proběhnutí skriptu, budou získány zcela mylné výsledky.

Tím je skript přidán v případě verze ArcGIS 9.2. Ve verzi 9.3 je nutné ještě ve vlastnostech skriptu vložit kód do ToolValidatoru a mezi parametry přidat jednu proměnnou s druhem dostupnosti. Samozřejmě na základě pozice v pořadí proměnných se musí upravit pořadí (číslo X) vstupních parametrů i v samotném skriptu. Nyní, když je skript součástí softwaru ArcGIS, ho lze snadno spustit jako jakýkoliv jiný nástroj v toolboxu. Podrobnější průběh analýz je blíže popsán v následujících podkapitolách.

4.3.4 Skript pro Service Area

Opět prvním zpracovaným modelem bude model pro analýzu Service Area.

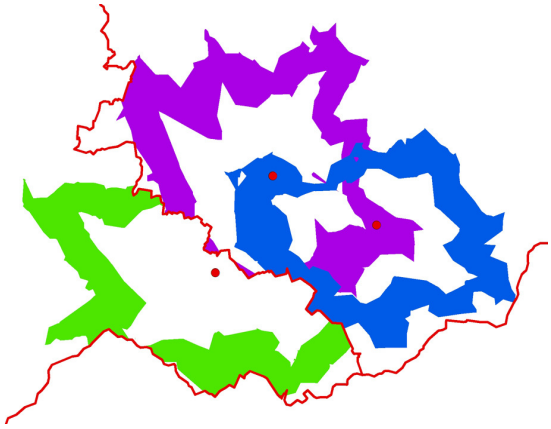
Vstupní data a jejich úprava

Vstupní data se nijak neliší od předchozí metody. Pro analýzu Service Area jsou potřeba network dataset a shapefily regionů a regionálních center. Zásadní rozdíly jsou ve zpracování vstupních dat.

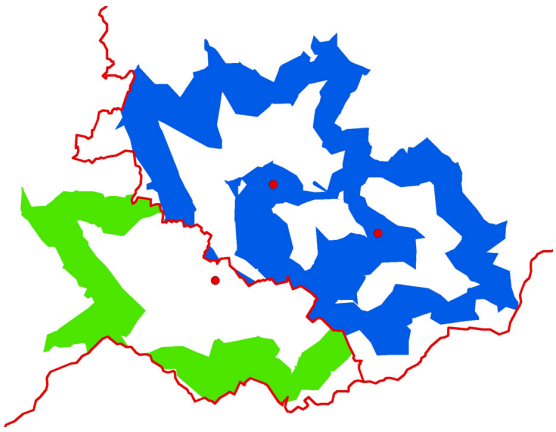
Než bylo možné spustit samotnou analýzu u předchozí metody, musely být obě vstupní vrstvy obohaceny o nový atribut. Hodnotou atributu byl jednoznačný identifikátor, který určoval, která centra a regiony patří k sobě. Za předpokladu, že budou vstupní data nyní rozdělena po regionech a každý soubor dat zpracován odděleně, není již žádný podobný identifikátor třeba. Analýza pokaždé proběhne pro jeden region s jedním centrem[25], takže

[25] Základním předpokladem je, že každý region bude mít pouze jedno centrum. Pokud jich bude více, je výsledek analýzy Service Area závislý na použité metodě. Analýza OD Cost Matrix dává u všech metod s více centry v jednom regionu stejné výsledky. Dostupnost je vždy vypočtena ze všech center do všech lokalit v rámci jednoho regionu. Každá lokalita tak obsahuje informaci o dostupnosti do každého centra v regionu. Výsledek analýzy Service Area u metody bariér a skriptu je stejný. Každé centrum v regionu má vygenerovány polygony podle zadaných hraničních hodnot. Lokality mezi centry tedy mohou mít více hodnot dostupnosti (viz obr. 16). Metoda využívající VBA funkce v této situaci dává pro analýzu Service Area zkreslené výsledky (viz obr. 17). To je dáno tím, že stejný identifikátor má více center, takže při závěrečném použití funkce *Dissolve* dojde ke spojení polygonů se stejnými hraničními hodnotami. Lokality mezi centry budou mít také více hodnot dostupnosti, ale protože všechna centra v regionu mají stejný identifikátor, nelze podle atributové tabulky rozlišit, ke kterému centru se každá hodnota vztahuje. Na obrázcích je znázorněn jeden polygon analýzy Service Area v určitém rozmezí hodnot pro každé centrum. Výstup z metody vytvořené v Pythonu dává každému centru vlastní polygon. Tudíž centra ve stejném regionu mají nenulovou dostupnost, ovšem ve vztahu k jinému centrum. Výstup z metody využívající VBA funkce spojí polygony z center ve stejném regionu, takže při interpretaci výsledků mají centra sama k sobě dostupnost větší než nula. V dalším textu je uvažováno pouze jedno centrum v každém regionu.

po rozdělení polygonů na vnitřní a vnější je jisté, které se mají smazat. Tím pádem odpadají veškeré úpravy vstupních dat prováděné v předchozí metodě. Namísto nich stačí rozdělit vstupní data na jednotlivé skupiny po jednom regionu.



Obr. 16 Jedno rozmezí hraničních hodnot polygonů analýzy Service Area pro případ více center v jednom regionu a metodu využívající Python



Obr. 17 Jedno rozmezí hraničních hodnot polygonů analýzy Service Area pro případ více center v jednom regionu a metodu využívající VBA funkce

V praxi to znamená, že pokud je ve vstupním souboru například 100 regionů, musí být vytvořeno 100 shapefilů, přičemž každý bude obsahovat jeden z regionů. Obdobně musí být rozdělena i centra. Rozdělení regionů po jednom do každého shapefilu je provedeno pomocí funkce *SearchCursor*. Je to funkce, která je dostupná jen přes Python, a proto ji nelze užít v ModelBuilderu (ESRI, 2007). Principem funkce je, že prochází atributovou tabulku jeden záznam po druhém a pro každý záznam je možné provést určitý sled akcí. Vstupem do funkce je Feature Layer regionů. Pro každý záznam se provede následující postup.

Nejprve je vybrán záznam, který se zrovna prochází. Pomocí funkce *Make Feature Layer* je uložen v paměti počítače jako samostatná vrstva. Pak je nutné vybrat centrum, které patří do regionu. Způsobů, jak toho dosáhnout, je několik, takže byl zvolen ten nejjednodušší. Namísto přidávání nových atributů a hledání shody s atributy regionu byl zvolen obyčejný výběr na základě polohy. Z vrstvy center bylo vybráno funkcí *SelectByLocation* to centrum, jež se nachází uvnitř vybraného regionu. Pro případ, že by se v regionu žádné centrum nenacházelo, byla zavedena podmínka, která nastalou situaci řeší. Jestliže nebude vybráno žádné centrum, další kroky se přeskočí a přejde se k dalšímu regionu. Ještě před samotným zpracováním center byla jejich vstupní vrstva upravena funkcí *MultipartToSinglepart*, aby byla zajištěna požadovaná geometrie. Tím jsou upravena vstupní data a je možné přejít k samotné analýze Service Area.

Analýza Service Area

Analýza probíhá stejným způsobem jako u předchozí metody. Nejprve proběhne funkce *Make Service Area Layer*, kde jsou vstupem network dataset zadaný uživatelem, vybraný druh dostupnosti a hraniční hodnoty dostupnosti. Posléze je funkcí *Add Locations* přidáno výchozí místo výpočtu dostupnosti. Jelikož analýza probíhá pro každý region zvlášť, je přidáno pouze centrum vybrané předchozím postupem. Aby výsledek analýzy byl co nejkvalitnější, uživatel volí ještě jednu proměnnou. Tou je atribut z atributové tabulky center, který nese jejich název. Název atributu je uložen jako vlastnost Name při zadávání centra do analýzy. Díky tomu bude každý vymodelovaný polygon obsahovat název centra, ze kterého byl počítán. Samotná analýza není nikterak omezována, takže pak následuje poslední krok analýzy, nástroj *Solve*. Tím je analýza dokončena a je přikročeno k tvorbě výstupu modelu.

Tvorba výstupu

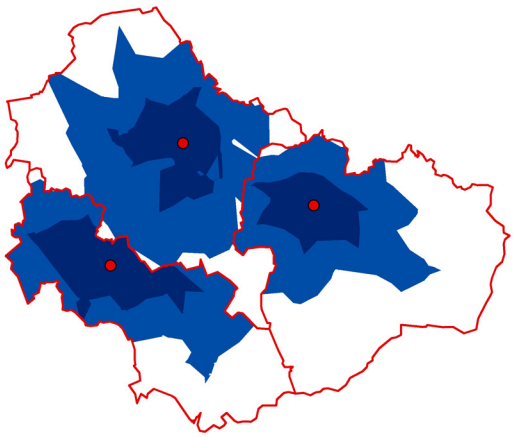
Je třeba neustále mít na paměti, že každý region probíhá odděleně. Takže tvorba výstupu se skládá z vytváření výstupu pro každý region zvlášť a následného spojení všech výsledků do jednoho souboru.

Tvorba výstupu pro jednotlivé regiony se moc neliší od předchozí metody. Funkcí *Select Data* jsou nejdříve vybrány polygony vzniklé analýzou dostupnosti. Vybrané polygony jsou uloženy do paměti počítače funkcí *Make Feature Layer*. Teď je nutné vybrané polygony opět rozdělit na vnější a vnitřní ve vztahu k regionu, pro který byly modelovány. Vnitřní polygony se dají snadno určit funkcí *Intersect.* Vstupem do funkce je vybraný region a vymodelované polygony. Výstup z funkce je průnik obou vrstev, tedy polygony ležící uvnitř regionu. Výstup je uložen jako samostatný shapefile do dočasné složky tmp.

Pokud by v této fázi došlo ke spojení všech vrstev polygonů do jedné, dosáhlo by se stejného výsledku jako u předchozí metody pomocí ModelBuilderu. Při shodných vstupech by byly výstupy z obou metod totožné. Ovšem díky Pythonu lze výsledek ještě dále upravit a názorně tak ukázat výhody programovacího jazyka oproti ModelBuilderu. V předchozím textu bylo zmíněno, že není možné zaplnit polygonem prázdná místa. Jde o situace, kdy uživatel zadá příliš malé hraniční hodnoty polygonů a vymodelované polygony tak nedosahují hranic regionů (viz obr. 18). Musí se však zdůraznit, že se jednalo o metodu bariér, se kterou byly spojeny další problémy. Pokud není analýza prostorově limitována bariérami, některé problémy se nevyskytnou. U zpracování ModelBuilderem byl zásadní fakt, že nelze zjistit, jakou hraniční hodnotu by měl mít nově vzniklý polygon zaplňující bílá místa. Hodnota je totiž závislá na poslední hraniční hodnotě zadané uživatelem. Ten by ji tak musel zadat ještě jednou, aby ji bylo možné dosadit jako hraniční hodnotu vytvořeného polygonu. Je to řešení, ale není příliš vhodné. Python nabízí funkce, díky kterým lze situaci vyřešit bez dalšího vstupu uživatele.

Nejprve je nutné vytvořit polygon vyplňující mezery mezi hranicemi regionu a polygony vzniklými analýzou Service Area. Možností je více. Avšak vstupní podmínkou bylo, že skript musí být funkční i pro licence ArcView a ArcEditor, a tak zbyla jen jedna. Použitím funkce

Union dojde ke sloučení dvou a více vrstev. Výsledkem je pak jejich sjednocení, takže se vzájemně doplní. Vstupem do funkce je vrstva, která vzešla z funkce *Intersect* a vrstva s regionem. Výstupem je vrstva polygonů z analýzy doplněná o polygony vyplňující mezeru mezi nimi a hranicemi regionu. Funkce *Union* obsahuje také nastavení, jaké atributy se mají ve výstupní vrstvě zachovat. Lze volit mezi všemi atributy, všemi bez FID a nebo pouze atribut FID vstupních vrstev. Ani jedna možnost není ideální. Důležité je zachovat atributy z vrstvy polygonů a zcela nepotřebné pro účely práce jsou atributy z vrstvy regionu. Byla zvolena možnost zachování všech atributů s tím, že atributy z vrstvy regionů budou později smazány.



Obr. 18 Polygony z analýzy Service Area v případě zadání malých hraničních hodnot

Každý vymodelovaný polygon analýzou Service Area obsahuje dva atributy určující, v jakém rozmezí hraničních hodnot definuje dostupnost. Tyto atributy se nazývají FromBreak a ToBreak. Nově vytvořený polygon musí mít hodnotu FromBreak rovnající se hodnotě ToBreak posledního polygonu z analýzy. Hodnota ToBreak u nového polygonu bude pak slovní vyjádření „více než“ a jeho hodnota FromBreak. Pokud tedy poslední vymodelovaný polygon by měl hodnoty FromBreak = 1000 a ToBreak = 1200, tak nově vytvořený polygon musí mít hodnoty FromBreak = 1200 a ToBreak = „více než 1200“.

Jelikož po funkci *Union* obsahuje výstupní vrstva všechny atributy z obou spojovaných vrstev, i nové polygony vyplňující mezeru mezi modelovanými polygony a hranicí regionu obsahují atribut FromBreak a ToBreak. Oba atributy pro tyto polygony obsahují hodnotu NULL. Ostatní polygony mají konkrétní hodnoty. Atributy FromBreak a ToBreak jsou číselného datového typu. Ideální by bylo místo hodnot NULL doplnit u nových polygonů patřičné hodnoty do těchto dvou sloupců. U sloupce FromBreak to lze, jelikož je dán číselnou hodnotou. U atributu ToBreak je ovšem problém v tom, že kromě hodnoty je doplněn i textový řetězec. Proto musí být atribut ToBreak nahrazen jiným, který bude mít datový typ text. Do nového atributu budou zkopírovány hodnoty z atributu ToBreak a zároveň i přiřazena správná hodnota u nových polygonů. Nový atribut je přidán nástrojem *Add Field* do výstupní

vrstvy z funkce *Union*. Název nového atributu byl nastaven na EndBreak, aby byl podobný původnímu názvu[26]. Nástrojem *Calculate Field* byly do nového atributu zkopírovány hodnoty z atributu ToBreak. Dalším krokem je nahradit hodnoty NULL v atributech FromBreak a EndBreak za správné hodnoty. K tomu je potřeba nejprve shapefile, který byl uložen do složky tmp po funkci *Union*, převést na Feature Layer. Je to nutné proto, že vstupem do dalšího nástroje *SelectLayerByAttribute* může být jen tento formát dat. Nástroj je použit k výběrů takových záznamů, které nemají v poli Name žádnou (tedy NULL) hodnotu. Tyto záznamy odpovídají všem nově vytvořeným polygonům. Výběr je prováděn proto, aby následující operace měly efekt právě jen na vybrané záznamy. Pro případ, že by nebyly vybrány žádné záznamy, je podmínkou ve skriptu zajištěno, že dojde k přeskočení dalších kroků. Pouze se zruší výběr a polygony v paměti počítače se uloží nástrojem *Copy Features* do podoby shapefilu. Název shapefilu se skládá ze slova „polygony“ a FID regionu, který je právě zpracováván. Pro každý region tak vznikne jeden polygonový shapefile. Jelikož v názvu vystupuje FID regionu, nemůže dojít k přepsání jednoho souboru jiným a všechny hledané polygony budou uloženy. Poté se posune cyklus na další region.

Pokud funkcí *SelectLayerByAttribute* dojde k výběru alespoň jednoho polygonu, jsou data dále upravována. Nyní je nutné separovat poslední uživatelem zadanou hraniční hodnotu. Tato hodnota představuje hodnotu v atributu ToBreak u posledního modelovaného polygonu a vlastně říká, kam až byla dostupnost modelována. Uživatel zadává hraniční hodnoty za sebou oddělené mezerou. Jednoduše je poslední zadaná hodnota uložena tak, že se ze zadaného řetězce vytvoří pole (seznam), kterému se nadefinuje, že oddělovačem hodnot je mezera. Tudíž je každá hraniční hodnota uložena odděleně. Poté stačí vložit poslední hodnotu pole do zvolené proměnné. Ve vybraných záznamech polygonů je následně za použití nástroje *Calculate Field* vypočten atribut FromBreak, který je u všech vybraných záznamů roven uložené proměnné. Stejným nástrojem je přiřazena hodnota i atributu EndBreak. Jen s tím rozdílem, že před samotnou hodnotou proměnné ještě figuruje text: „více než “. Hraniční hodnoty polygonů jsou uloženy také v atributu Name společně s názvem centra. Hodnota atributu Name nového polygonu je také NULL. Nástrojem *CalculateField* je změněna NULL hodnota na název regionálního centra v regionu, řetězec „ : více než „ a hodnota FromBreak polygonu. Celek tak odpovídá charakteru hodnot modelovaných polygonů (např. Klatovy : více než 1200). Kdyby bylo v regionu více než jedno centrum, je ve skriptu další funkce *SearchCursor*. Ta projde všechna centra vybraná daným regionem a uloží jejich názvy za sebe oddělené čárkami.

[26] Nelze přidat atribut se stejným názvem, který už atributová tabulka obsahuje. Proto byl zvolen název podobný původnímu. Bylo by možné po smazání starého atributu ToBreak nový přejmenovat na název ToBreak. Tím by se ale nezměnil název atributu jako takového, ale pouze jeho alias zobrazující se v tabulce. Samozřejmě lze po smazání atributu ToBreak ho znovu přidat s datovým typem text a zkopírovat do něj hodnoty z atributu EndBreak. Ten by byl následně smazán. Je ovšem otázkou, jestli operace navíc s atributovou tabulkou jsou vhodné při vytváření ideálního řešení problematiky. Přidávání dalších operací vede jen k prodloužení výpočetního času metody a ke snižování její efektivity. Proto byl použit nejnižší nutný počet kroků k dosažení potřebného výsledku a byl ponechán atribut EndBreak tak, jak byl vytvořen.

V takovém případě by výsledná hodnota vypadala například: Klatovy, Plzeň : více než 1200. S poslední úpravou atributové tabulky je dokončen proces doplnění polygonu do bílých míst. Opětovným použitím funkce *SelectLayerByAttribute* je zrušen výběr polygonů. Pokud by nebyl zrušen výběr, byly by uloženy pouze vybrané polygony. Další postup je totiž shodný s tím, jaký probíhal za situace, kdy nebyl vybrán žádný polygon. Polygony se uloží funkcí *Copy Features* podle popsaného klíče do složky tmp.

Po skončení cyklu pro poslední region, je potřeba všechny uložené polygony spojit do jednoho shapefilu. To lze provést funkcí *Merge*, ale nejdříve se musí určit, jaké vrstvy se mají spojit. Vstupem do funkce je řetězec, který obsahuje názvy vrstev oddělené středníkem (např. polygony1.shp;polygony2.shp). Požadovaný řetězec je vytvořen za pomoci funkce *listfeatureclasses*. Je to opět funkce, která je dostupná v ArcGISu jen přes skript v Pythonu. Funguje na podobné principu jako *SearchCursor*, kdy se načtou jednotlivé vrstvy do paměti (vytvoří se jejich seznam) a poté se každá položka ze seznamu upraví. Pomocí funkce *listfeatureclasses* je vytvořen seznam všech polygonových vrstev, jejichž název začíná řetězcem „polygon“. Funkce hledá takové vrstvy v definovaném pracovním prostředí, tedy ve složce tmp vytvořené na počátku skriptu (ESRI, 2007). Poté následuje cyklus, který zpracovává názvy vrstev v seznamu. Cyklus uloží název první položky v seznamu do prázdného řetězce a přidá na konec středník. Název druhé položky je přidán k názvu první a opět se zakončí středníkem. Takto se váží všechny názvy položek za sebe. Díky cyklu je vytvořen řetězec, který je vyžadován jako vstup funkcí *Merge*. Výstup z funkce je uložen do samostatného shapefilu.

Spojením polygonů do jedné vrstvy skript ještě nekončí. Předchozími funkcemi *Intersect* a *Union*, kdy byly zachovávány všechny atributy vstupních vrstev, byla naplněna atributová tabulka zcela nepotřebnými atributy. Důležité jsou pouze atributy FID, Shape, Name, FromBreak a ToBreak. Všechny ostatní lze smazat. Nástrojem *ListField* jsou procházeny všechny atributy v tabulce výstupní vrstvy nástroje *Merge* a pokud není jejich název shodný s jedním z jmenovaných, je přidán do seznamu. Seznam představuje řetězec názvů atributů oddělené středníkem. Tento řetězec slouží jako vstup do funkce *DeleteField*, kterým jsou nepotřebné atributy z tabulky vymazány.

Skript je ještě opatřen jednou podmínkou. Jestliže v průběhu zpracování bylo alespoň v jednom regionu více center, je použita funkce *Dissolve*. Mohlo totiž dojít k překrytu polygonů z různých center. Překrývající polygony se navzájem rozdělí při aplikaci funkce *Intersect*. Nástroj *Dissolve* je opět spojí dohromady tak, aby všechna centra měla pro každé rozmezí hraničních hodnot pouze jeden polygon. Pokud v každém regionu bylo vždy jen jedno centrum, namísto funkce *Dissolve* proběhne funkce *Copy Features*. Výstup z obou funkcí je uložen v podobě shapefilu na uživatelem definované umístění na disku. Na závěr skriptu dojde k vymazání složky tmp a tím je skript ukončen.

4.3.5 Skript pro OD Cost Matrix

Skript pro OD Cost Matrix je dosti podobný tomu pro analýzu Service Area, a proto bude jeho popis poměrně krátký. Hlavní rozdíl, pokud si odmyslíme odlišnosti vzniklé použitím jiné analýzy, je v tom, že není nikterak nutné přidávat další elementy pro vylepšení samotného výstupu. Výsledkem jsou přesné hodnoty dostupnosti mezi dvěma body, takže není žádný způsob, jak výsledek více zpřesnit jako u analýzy Service Area. Aby i tak byla ukázána výhoda použití skriptu oproti ModelBuilderu, bylo přikročeno k jiné úpravě. Opět díky dávkovému zpracování dat nejsou potřeba žádné identifikátory, a proto do vlastnosti Name u subvrstev Origins a Destinations může být zvolen jiný atribut než FID regionů. Pokud uživatel zvolí ve vstupních parametrech, jaké atributy v atributových tabulkách regionálních center a cílových míst představují jejich název, budou jejich názvy obsaženy ve výstupní vrstvě. Uživatel tak snadno určí při pohledu na atributovou tabulku výsledného shapefilu, jakým dvou lokalitám každý záznam odpovídá. U ModelBuilderu takového výsledku nešlo dosáhnout. Oba atributy by musely být přidány zpětně a bylo by to přinejmenším dost složité. Díky dávkovému zpracování je to ovšem jednoduché a elegantní vylepšení celkového výstupu.

Vstupní data a jejich úprava

Vstupní data neprochází opět žádnou úpravou. Naprosto identicky s předchozím skriptem jsou rozdělena podle regionů. Postup je stejný[27], pouze v průběhu skriptu ještě přibyla úprava vrstvy cílových míst. Součástí úpravy vrstvy cílových míst je zajištění singlepart geometrie a následné rozdělení podle regionů. Obdobně jako u vrstvy regionálních center je toho dosaženo nástrojem *SelectLayerByLocation*, kdy jsou vybrána cílová místa ležící ve zpracovávaném regionu a následně nahrána do analýzy.

Analýza OD Cost Matrix

Nastavení analýzy se nikterak neliší od předchozího nastavení v případě metody využívající VBA funkce. Během přidávání regionálních center a cílových míst funkcí *Add Locations* se uvede jako vlastnost Name atribut, který byl zvolen uživatelem. Díky tomu si s sebou ponese každý bod v analýze název, který určil uživatel, a bude pro něj snadnější identifikovat jednotlivé hodnoty dostupnosti mezi výchozími a cílovými místy. Standardně je analýza zakončena nástrojem *Solve*.

Tvorba výstupu

Vytváření výstupu se opět u skriptu dělí na dvě části. První část je prováděna v rámci cyklu pro každý region zvlášť a jsou při ní extrahovány modelované linie dostupnosti a ukládány jako shapefily do složky tmp. Výběr linií z výstupní vrstvy vzniklé po aplikaci funkce *Solve* je proveden nástrojem *Select Data*. K uložení linií slouží nástroj *Copy Features*. Podobně jako

---

[27] Jen krátké připomenutí: Každý region je postupně vybrán funkcí *SearchCursor* a poté proběhne cyklus, ve kterém jsou vybrána vstupní data ležící ve vybraném regionu. Pro případ, že by v regionu nebylo žádné centrum nebo cílové místo, je skript upraven tak, aby byl daný region přeskočen.

u polygonů se název výstupu skládá ze dvou částí, aby byl pro každý region jedinečný a nedošlo k přepsání výstupu z jiného cyklu. Uložený shapefile s liniemi z jednoho regionu má název složený ze slova „linie“ a FID regionu.

Druhá část tvorby výstupu se realizuje po proběhnutí cyklu pro poslední region. Všechny výstupní shapefily s liniemi jsou spojeny v jeden. Postup spojení vychází z předchozího skriptu pro analýzu Service Area. Nejdříve je vytvořen seznam shapefilů, které mají být spojeny. Do seznamu jsou zařazeny funkcí *listfeatureclasses* všechny liniové shapefily, které se nacházejí ve workspace skriptu (složka tmp) a obsahují ve svém názvu řetězec „linie“. Pomocí cyklu je vytvořen potřebný řetězec pro funkci *Merge*, která výstupy spojí v jeden. Zde by opět mohl skript skončit a výsledek by byl identický tomu, co vytváří model z ModelBuilderu. Ovšem na počátku popisu skriptu bylo naznačeno, že i v případě OD Cost Matrix byl výsledek upraven, aby byl o něco lepší než v případě zpracování v ModelBuilderu.

Atribut Name vždy obsahuje název regionálního centra a cílového místa, pro která byla dostupnost modelována. Podobně jako tomu bylo u modelu využívajícího VBA funkce, dojde i zde k rozdělení hodnot tohoto atributu do dvou nových sloupců atributové tabulky[28]. Sloupce jsou přidány nástrojem *Add Field*. Jednomu je nastaven název „centrum“ a druhému „lokalita“. Hodnoty nových atributů jsou vypočteny shodně s předchozí metodou využívající VBA funkce. Název centra se nachází vlevo, a proto je oddělen od názvu cílového místa výrazem:

```
Left ( [Name], ( ( InStr ( [Name], \" - \" )) - 1) )
```

Název cílového místa se nachází v atributu Name napravo. Jeho název je do atributu lokalita zkopírován výrazem:

```
Right ( [Name], ( Len ( [Name] ) - ( InStr ( [Name], \" - \" )) - 1) )
```

Výrazy se od předchozí liší jen tím, že není vyjádřena pozice pomlčky, ale trojice znaků: mezera, pomlčka, mezera. Proto i v případě použití funkce Left není odečítána dvojka, ale pouze jednička. Obrácené lomítko u uvozovek ve funkci InStr je použito proto, aby Python bral uvozovky jako znak a nikoliv jako počátek nebo konec řetězce.

Pak už jsou jen smazány funkcí *DeleteField* pro uživatele nepotřebné atributy vzniklé při analýze: Name, DestinationID, OriginID a DestinationRank. Funkce *CopyFeatures* vytvoří z vrstvy po smazání nepotřebných atributů shapefile. Ten je uložen na místo a pod názvem, které definoval uživatel před spuštěním analýzy. Na úplný závěr je opět vymazána přechodná složka tmp a skript je ukončen.

[28] Samozřejmě by se mohl nechat atribut Name tak, jak je. Avšak pak by vyhledávání případně seřazování záznamů jen podle cílových míst bylo nemožné.

KAPITOLA 5

Srovnání

Porovnání metod řešení problematiky je důležitou součástí práce. Bez něho není možné určit vlastnosti jednotlivých přístupů a rozhodnout o jejich vhodnosti použití. Důkladným testováním byly zkoumány různé aspekty chování modelů a skriptů. Všechny výstupy práce byly posouzeny několika způsoby, aby se zabránilo zkresleným výsledkům v důsledku použité metody srovnání.

5.1 Metody srovnání

Nejprve je nutné zdůraznit, že analýzy Service Area a OD Cost Matrix byly hodnoceny odděleně. Přestože základní principy srovnání jsou pro obě analýzy shodné, musí se od sebe rozlišovat. Každá analýza vyžaduje různé vstupní informace a také poskytuje odlišný výstup. Společné pro obě analýzy jsou podmínky testování a vyhodnocování jednotlivých testů. Rozdíly jsou v množství spuštěných testů a v jejich nastavení.

Podmínky testování byly nastaveny tak, aby došlo k co nejmenšímu vlivu použité výpočetní techniky na výsledky testů. Každý soubor vstupních dat (viz dále) byl testován na dvou počítačích. Prvním byl méně výkonný notebook se softwarem ArcGIS 9.2 ArcEditor a druhým byl výkonnější stolní počítač s verzí ArcGIS 9.3 ArcInfo. Každý test byl na obou počítačích proveden dvakrát. Mezi jednotlivými testy byl vždy proveden restart počítačů, aby všechny testy měly obdobné startovací podmínky.

Během testů byly sledovány kvalitativní i kvantitativní ukazatele, podle kterých bylo možné provést srovnání. Z kvantitativního hlediska byl hodnocen výpočetní čas jednotlivých metod. Výpočetní čas se zkoumal pro různý objem vstupních dat, aby bylo odhaleno chování modelů v různých zátěžových situacích. Z dosažených výsledků jsou poté vyvozeny závěry a určena

vhodnost použití jednotlivých metod. Bližší popis jednotlivých vstupů je uveden v následující podkapitole.

Mezi kvalitativní ukazatele patří jednak atributová tabulka výstupní vrstvy, jednak i samotná výstupní vrstva. V atributové tabulce bylo hodnoceno množství atributů v ní obsažené a hlavně informace, které atributy obsahují. Hodnocení kvality výstupní vrstvy bylo složitější. U obou analýz byl použit výstup z modelů za celé Česko a porovnávaly se hodnoty dostupnosti pro všechny obce v Česku. Porovnání se skládalo ze dvou částí. Nejprve bylo zkoumáno, jestli modely dokázaly vypočítat dostupnost do všech obcí. Výstupy s analýzou Service Area byly hodnoceny tak, že se provedla funkce *Intersect* bodové vrstvy obcí a každého výstupu. Vznikla tak pro každý výstup nová bodová vrstva obcí obsahující atributy z výstupu. Pokud se některá obec nacházela v místech, kde nebyla vypočtena dostupnost, nebyla v nové vrstvě zahrnuta. Naopak pokud byla obec v místě, kde se překrývalo více polygonů dostupnosti, vyskytovala se ve výstupní vrstvě vícekrát. Zvláštním druhem případu jsou situace, kdy k obci nedosáhly polygony z jejího regionu, ale nevhodnou interpolací ji pokrývaly polygony ze sousedního regionu. Obec byla ve výstupní vrstvě sice zahrnuta jen jednou, ale hodnoty dostupnosti patřily sousednímu regionu. Poslední dva jmenované příklady se týkají pouze analýzy s využitím bariér. Ostatní modely veškeré části polygonů přesahující za hranici svého regionu odřezávají. Kontrola byla provedena pomocí nástroje *Summarize* v atributové tabulce vrstvy. Po aplikaci funkce na jeden z atributů z tabulky dojde k vytvoření nové tabulky. Každý záznam v nové tabulce obsahuje jednu unikátní hodnotu ze zvoleného atributu a počet těchto hodnot v celé analyzované tabulce. Aplikací funkce *Summarize* na atribut obsahující jednoznačný identifikátor[29] obcí lze zjistit, jestli se některá obec v záznamech objevuje vícekrát. Pro každý identifikátor obce vznikne v nové tabulce jeden záznam a společně s tím se uloží i informace o tom, kolikrát se daný identifikátor v celé tabulce objevuje. Následným zkoumáním, zda počáteční místo generování polygonů odpovídá regionálnímu centru obcí, byly duplicitní záznamy odhaleny. Pokud obec byla pokryta i polygony ze sousedního regionu, bylo počáteční místo generování polygonů odlišné od regionálního centra obce. Lze tak odhalit i případy, kdy obec nepokryly polygony z vlastního centra, ale pouze ze sousedního regionu. Odhalené záznamy byly odstraněny, aby se mohlo provést další testování.

Druhá část zkoumání výsledků se týkala přímo hodnot dostupnosti. U analýzy Service Area byla prováděna kontrola, zda obce u všech modelů spadají do stejného rozmezí hodnot dostupnosti. Jednoduchým spojením atributových tabulek pomocí jednoznačného identifikátoru obcí lze porovnat záznamy z analýz pro stejné obce.

Hodnocení analýzy OD Cost Matrix je odlišné. Negenerují se polygony, které by přesahovaly do jiných regionů, takže by nebylo nutné provádět analýzu funkcí *Summarize*. Přesto však byla provedena, aby předpoklad, že každá obec má vypočtenou dostupnost pouze

---

[29] Předchozím použitím funkce *Intersect* na vrstvu obcí bylo zajištěno, že výstupní vrstva z funkce obsahuje všechny atributy z původní vrstvy obcí. Mezi těmito atributy je i tento jedinečný identifikátor.

do svého regionálního centra, byl potvrzen v praxi. Jednoduchým zkontrolováním počtu záznamů v atributové tabulce výstupů lze pak zjistit, do kolika obcí byla každá metoda schopna vypočítat dostupnost. Primárně tak byly hodnoceny hodnoty dostupnosti pro jednotlivé obce. Porovnání mezi jednotlivými metodami výpočtu bylo provedeno opět spojením tabulek. Ovšem v tomto případě nemohly být tabulky spojeny na základě společného atributu, proto byly spojeny jednotlivé záznamy na základě polohy. Výstupem z analýzy OD Cost Matrix jsou přímé linie mezi regionálním centrem a obcemi ležícími ve stejném regionu, tudíž jsou linie pro všechny vypočtené obce identické nehledě na použitou metodu. Při spojování tabulek na základě polohy linií dojde k vytvoření nové liniové vrstvy, v jejíž atributové tabulce budou u každé linie atributy z obou původních vrstev. Jestliže se poté vytvoří nový atribut, do kterého bude uložen rozdíl hodnot dostupnosti mezi jednotlivými metodami pro každou obec, výstupy se snadno porovnají.

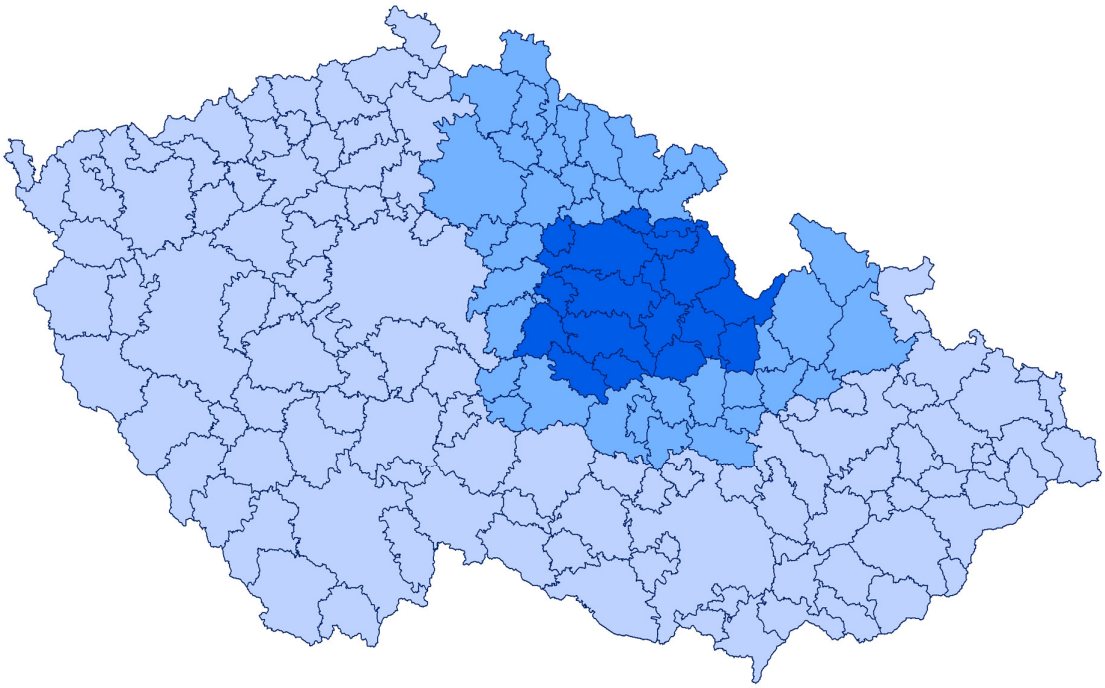
5.2 Testovací data

Modely a skripty jsou navrženy tak, aby dokázaly zpracovat rozličná vstupní data. Je tak možné testovat výpočetní čas algoritmů v závislosti na objemu vstupů. Jako základ pro všechna testování slouží tři vstupní vrstvy. Jedná se o polygonovou vrstvu mikroregionů 1. stupně ze sociogeografické regionalizace Česka včetně všech subregionů (celkem 166 regionů[30]), bodovou vrstvu regionálních center této regionalizace (166 bodů) a bodovou vrstvu obcí Česka (6 248 bodů). Všechna testovací data vycházela z těchto základních vrstev. Pro obě analýzy byly vybrány tři úrovně objemu vstupních dat. Každá úroveň je definována počtem regionů, který byl stanoven tak, aby mezi jednotlivými úrovněmi byl dostatečně velký rozdíl. Vyšší úroveň obsahuje přibližně trojnásobek regionů nižší úrovně. Nejmenší počet regionů byl stanoven na 18, druhá úroveň jich má 55 a poslední 166. Regiony ve všech úrovních tvoří uzavřený celek. Konkrétní úrovně regionů jsou zobrazeny na obrázku 19. Nejtmavší barvou je vyznačeno 18 regionů, střední odstín včetně nejtmavších regionů tvoří úroveň s 55 regiony a celé Česko představuje 166 regionů. Počet regionálních center a obcí vstupujících do analýzy se poté odvíjel od počtu regionů. Do modelů však vstupovaly tyto dvě vrstvy neupraveny, takže k vyčlenění potřebných bodů došlo až v průběhu modelů nebo skriptů.

Jako vstup do všech analýz je zapotřebí network dataset. Pro testování byl použit již hotový network dataset, který vznikl jako součást bakalářské práce Modelování dostupnosti lékáren a nízkoprahových center (Navrátil, 2008). Tento network dataset obsahuje všechny komunikace od dálnic po silnice třetí třídy a dva atributy, které lze využít k modelaci dostupnosti. Jedná se o čas a vzdálenost. K testování byl použit pouze atribut času. Podrobné informace o tom, jak byl network dataset vytvořen a jednotlivé atributy vypočteny, je možné nalézt ve zmíněné

[30] Sociogeografická regionalizace z roku 2001 má včetně subregionů 165 regionů. Ovšem ve vrstvě dodané katedrou Sociální geografie a regionálního rozvoje byl jeden subregion navíc – Neratovice.

bakalářské práci. Metody s využitím bariér kromě samotného network datasetu ještě vyžadují vrstvu komunikací, ze kterých je network dataset vytvořen.



Obr. 19 Úrovně objemu testovacích dat podle regionů

Analýza Service Area navíc vyžaduje hraniční hodnoty polygonů. V tom se odlišuje od analýzy OD Cost Matrix. Zatímco u ní lze náročnost výpočtu ovlivnit pouze počtem regionů, u analýzy Service Area hrají roli právě ještě hraniční hodnoty polygonů. Proto byly vytvořeny celkem 4 úrovně náročnosti podle hraničních hodnot. Každá úroveň obsahuje 4 hodnoty, které jsou násobkem první hodnoty. Zároveň první hodnoty každé úrovně jsou násobkem první hodnoty první úrovně. První hodnotou první úrovně bylo zvoleno 300 (v souvislosti s atributem času se jedná o sekundy, tudíž první hodnotou je 5 minut). Tabulka níže názorně ukazuje jednotlivé hraniční hodnoty pro všechny úrovně.

Tab. 1 Úrovně hraničních hodnot polygonů pro analýzu Service Area [sekundy]

Úroveň	První hodnota	Druhá hodnota	Třetí hodnota	Čtvrtá hodnota
I	300	600	900	1200
II	600	1200	1800	2400
III	900	1800	2700	3600
IV	1200	2400	3600	4800

U každé metody byla tedy analýza OD Cost Matrix spuštěna se 3 různými nastaveními, analýza Service Area s 12 různými nastaveními. Výsledné časy jednotlivých testů jsou uvedeny v následující podkapitole.

5.3 Výsledky

Jak již bylo zmíněno, hodnocení výsledků srovnání bylo provedeno pro analýzu Service Area a OD Cost Matrix odděleně. U každé analýzy bude nejprve zkoumán výpočetní čas jednotlivých testovacích scénářů a následně kvalita výstupu každé nalezené metody.

5.3.1 Service Area

Jako první byla zkoumána analýza Service Area.

Výpočetní čas

Srovnání jednotlivých metod pro analýzu Service Area bylo náročnější. Je to dáno tím, že kromě různého počtu regionů, co mohou vstupovat do analýzy, bylo počítáno i s různým nastavením hraničních hodnot polygonů. Tabulka 2 ukazuje dosažené časy v sekundách pro jednotlivé metody a testovací scénáře. V sloupci náročnost je uvedeno, jaké vstupy byly v testu použity. Číslo arabskými číslicemi označuje počet regionů, který do analýzy vstupoval, a římská číslice určuje použité hraniční hodnoty polygonů. Sloupce bariéry, VBA funkce a Python označují metody popsané výše. Metoda Python zkrácený je obdobná metodě Python pouze s tím rozdílem, že se nevytváří polygon navíc v případě, že nejsou zadány dostatečné hraniční hodnoty polygonů. Výsledek tak odpovídá výsledku metody VBA funkce a jejich porovnáním se získá lepší přehled o vlastnostech metod.

Důležité je podotknout, že testování probíhalo nepřetržitě několik dní, takže nebylo možné se vyhnout fluktuacím ve výpočetním výkonu počítače. Patrné je to hlavně na starším notebooku s ArcGIS 9.2. Bohužel ani opakované spouštění testů, které trvaly déle, než bylo předpokládáno, nevedlo k jinému výsledku. Při srovnání s výsledky z výkonnějšího počítače je patrné, že výpočetní výkon počítače má vliv na dosažené výpočetní časy. Dosažené výsledky jsou tudíž pouze orientační.

Tab. 2 Výpočetní čas metod pro různá vstupní data a analýzu Service Area v ArcGIS 9.2 [sekundy]

Náročnost	Bariéry	VBA funkce	Python zkrácený	Python
166 IV	384	-	2 304	4 671
166 III	399	1 882	2 301	3 943
166 II	395	299	2 313	4 208
166 I	389	90	2 292	4 328
55 IV	135	910	812	1 024
55 III	128	324	807	716
55 II	131	96	801	689
55 I	132	29	816	727
18 IV	56	125	221	289
18 III	54	66	220	198
18 II	51	28	226	179
18 I	50	11	218	182

Z tabulky lze vypozorovat, že u metod bariér a obou metod založených na Pythonu nedochází k velkému ovlivnění hraničními hodnotami polygonů. V zásadě je výpočetní čas pro každou úroveň srovnatelný. Největší výkyvy jsou u Pythonu, což je dáno různou náročností výpočtu polygonu vyplňujícího bílá místa. Mimoto testování pro čtvrtou úroveň hraničních hodnot proběhlo až později a je zde patrný nárůst výpočetního času oproti předpokladu. Naopak metoda VBA funkcí je silně závislá na hraničních hodnotách. Pro nejnáročnější kombinaci vstupů nebyl dokonce výpočet dopočítán. V průběhu funkce *Intersect*, která rozděluje polygony vygenerované analýzou na vnější a vnitřní ve vztahu k regionu, došlo k překročení integritních omezení u dbf tabulky shapefilu. Ukázalo se tak největší úskalí této metody[31].

Pokud se nyní porovnají výsledky z hlediska náročnosti, nejrychlejší metodou pro větší hraniční hodnoty (třetí a čtvrtá úroveň) jsou bariéry. Pro menší hraniční hodnoty pak VBA funkce. V případě, že hraniční hodnoty nejsou nastaveny tak, že by u většiny regionů přesahovaly jejich území, je využití VBA funkcí nejrychlejší. Nemusí být totiž generovány a nahrávány bariéry nebo naopak regiony rozdělovány po jednom a poté výsledky opět spojovány. Bariéry jsou nejrychlejší pro náročnější vstupy proto, že u nich je výpočet vždy zastaven na území regionu a není tedy zbytečně počítána dostupnost daleko za jeho hranice.

[31] Specielně u této metody byla kontrolována velikost shapefilu, který vznikne po zmiňované funkci *Intersect*. U 166 regionů vzniklo pro první úroveň hraničních hodnot 6 284 polygonů a 14,5 MB dat, pro druhou úroveň 168 482 polygonů a 224 MB dat, pro třetí úroveň 1 346 802 polygonů a 1,53 GB dat. Jasně se tak ukazuje, jak roste náročnost celé metody v závislosti na tom, jak moc vymodelované polygony přesahují hranice regionů.

Samotným porovnáním metod VBA funkcí a Pythonu zkráceného je evidentní, že výhody dávkového zpracování se ukazují až pro větší objem vstupních dat. Časová náročnost rozdělení a opětovného spojení regionů je kompenzována až pro největší hraniční hodnoty polygonů a pro více regionů. Výhodou je pak zpracování i nejnáročnější kombinace, u které metoda VBA funkcí nebyla schopna dojít výsledku. Metoda Pythonu bez úpravy vychází jako nejpomalejší. Sice podle tabulky byla pro méně náročné kombinace vstupů rychlejší než Python zkrácený, ale to principielně musí být chybný výsledek. Metoda Pythonu zkráceného byla testována také později, tudíž je delší výpočetní čas způsobený nižším výkonem notebooku.

Tabulka 3 obsahuje výpočetní časy metod na výkonnějším stolním počítači se softwarem ArcGIS 9.3. Hlavní rozdíl mezi verzemi 9.2 a 9.3 je časová náročnost funkce *Add Locations*. Zatímco ve starší verzi tato funkce probíhala poměrně rychle, v novější verzi trvala asi desetkrát déle. Důvod této změny se nepodařilo odhalit, ale stejný problém se opakoval na více počítačích se softwarem ArcGIS 9.3. Vzhledem k tomu se výrazně prodloužil výpočetní čas pro metodu bariér, která se tak náhle stala nejpomalejší metodou vůbec. Zbylé metody lze mezi sebou porovnat, protože pro každou kombinaci náročnosti dochází k nahrávání stejného počtu lokalit do analýz. Ovšem neúměrný nárůst časové náročnosti jedné funkce smazává případné rozdíly mezi metodami[32].

Tab. 3 Výpočetní čas metod pro různá vstupní data a analýzu Service Area v ArcGIS 9.3 [sekundy]

Náročnost	Bariéry	VBA funkce	Python zkrácený	Python
166 IV	9 573	-	1 076	1 650
166 III	9 613	1 909	958	1 583
166 II	9 611	709	952	1 471
166 I	9 665	549	895	1 460
55 IV	3 101	761	357	450
55 III	3 258	376	304	405
55 II	3 155	225	274	387
55 I	3 136	186	259	386
18 IV	1 186	121	109	137
18 III	1 209	94	103	121
18 II	1 211	70	88	116
18 I	1 224	57	83	119

[32] Pokud by zbylý průběh jedné metody byl dvakrát rychlejší než jiné, ale zároveň více jak 90 % výpočetního času zabírá funkce *Add Locations* společná pro obě metody, výsledný rozdíl výpočetních časů je minimální. Drobné odchylky mezi metodami mohou být způsobeny vícero vlivy, tudíž nelze na jejich základě metody hodnotit.

Pohled na tabulku 3 potvrzuje některá z tvrzení vyvozených z předchozí tabulky. Výpočetní časy metod bariér a obou Pythonů nejsou téměř závislé na hraničních hodnotách polygonů. Na rozdíl od méně výkonného notebooku však lze vysledovat zkracování časů s nižšími hraničními hodnotami. Ovšem tyto rozdíly jsou minimální oproti metodě VBA funkcí a nejsou patrné vždy. Stejně tak si metoda VBA funkcí zachovává závislost na hraničních hodnotách a také nebyla dokončena pro nejnáročnější kombinaci vstupů.

Kromě již zmíněných bariér je vidět odlišnost mezi metodami využívajícími Python. Výkonnější počítač s dostatkem paměti není tolik náchylný ke změnám výkonu, a proto je názorné, že zkrácená verze skriptu je časově méně náročná než úplná verze. Vyšší výkon se projevil i v tom, že dávkové zpracování je rychlejší než metoda VBA funkcí i pro méně náročné kombinace vstupů. Dávkové zpracování pomocí zkrácené verze Pythonu je rychlejší u 166 a 55 regionů pro třetí a čtvrtou úroveň hraničních hodnot a u 18 regionů pro nejvyšší hraniční hodnoty. Úplná verze Pythonu je rychlejší než VBA funkce u 166 regionů pro třetí a čtvrtou úroveň hraničních hodnot a u 55 regionů pro nejvyšší hraniční hodnoty. Jasně se tedy ukazuje, že s výkonnějším počítačem je dávkové zpracování výhodnější.

Kvalita výstupu

Pro zkoumání kvality výstupu byly vybrány z každé metody výstup pro 166 regionů a třetí úroveň hraničních hodnot. Čtvrtá úroveň nemohla být použita, protože pro metodu využívající VBA funkce nebylo možné získat výstup. U zkoumání kvality výstupu byla uvažována pouze úplná verze metody vytvořené v Pythonu. Zkrácená verze má atributovou tabulku stejnou jako úplná verze a co se týče geometrie výstupu je shodná s metodou využívající VBA funkce.

První ukazatelem byl počet obcí, který byl pokryt výstupní vrstvou. Vstupem do funkce *Intersect* byla vrstva obcí Česka (6 248 bodů) a jednotlivé výstupy metod. Metoda bariér pokryla celkem 6 533 obcí, metoda VBA funkcí 6 230 obcí a metoda Pythonu 6 248 obcí. Aplikací funkce *Summarize* bylo u metody bariér odhaleno 547 duplicitních obcí a 2 obce byly dokonce zahrnuty třikrát. Ostatní metody nezahrnovaly žádnou duplicitní obec. Kontrolou, zda obce pokrývají polygony ze stejného regionu, bylo odhaleno dalších 56 obcí u metody bariér, které byly pokryty pouze polygony ze sousedních regionů. Smazáním všech chybných záznamů zůstalo 5 928 obcí. Podle výsledků prvního ukazatele je evidentní, že metoda bariér je nejméně vhodnou metodou z hlediska kvality výstupu. Nejenže dokázala vypočítat dostupnost pro nejmenší počet obcí, navíc ještě byl výsledek zatížen chybami vzniklými interpolací polygonů nerespektující hranice regionů. Metoda VBA funkcí sice tyto nedostatky odstraňuje, ale výsledek je závislý na zadaných vstupních hodnotách. Pro zvolené hodnoty nebyla metoda schopna vypočítat dostupnost pro 18 obcí. Metoda Pythonu naopak dokáže vypočítat dostupnost pro jakékoliv hraniční hodnoty do všech obcí na území regionů.

Druhý ukazatel se zaměřil na hodnoty dostupnosti u obcí. Porovnáním hodnot dostupnosti všech metod u každé obce lze zjistit, jak jednotlivé metody mění výsledky dostupnosti. Srovnání metody VBA funkcí a Pythonu ukázalo 100% shodu pro všechny společné obce.

Rozdíl byl samozřejmě v 18 obcích, které metoda VBA funkcí nepokryla, kdežto Python ano. Vzhledem k tomu, že postup výpočtu samotné dostupnosti je u obou metod identický, byl takový závěr předpokládán. Porovnání metody bariér a VBA funkcí ukázalo, že bariéry díky jiné interpolaci vypočetly dostupnost do jedné obce, do které metoda VBA funkcí nikoliv. Jinak z 5 927 obcí, u kterých mohlo proběhnout porovnání, byla u 5 684 shoda. Dalších 233 obcí metoda bariér přiřadila do polygonu s většími hraničními hodnotami než u metody VBA funkcí a 10 obcí mělo hraniční hodnoty u bariér menší. Všechny obce s menším rozmezím hodnot se nacházely v těsné blízkosti hranic polygonů a byly do nižšího rozmezí zaneseny odlišnou interpolací. Porovnání metody bariér a Pythonu poskytlo identické výsledky jako předchozí. Pouze obcí, které měly menší rozmezí hraničních hodnot u metody bariér, bylo 11. Jedenáctou obcí je zmiňovaná obec, pro kterou metoda bariér vypočetla dostupnost, kdežto VBA funkce nikoliv. Výsledkem srovnání je, že v téměř 91 % všech obcí se metody shodly v rozmezí dostupnosti. Samozřejmě je výsledek ovlivněn nastavenými hodnotami a pro jiné by mohl být odlišný, ale přesto je dosaženo poměrně vysoké shody. Potvrdil se i předpoklad, že u metody bariér budou některé obce ve vyšším rozmezí hraničních hodnot. Využitím jen komunikací uvnitř regionu najde algoritmus často jedinou možnou cestu namísto té nejvýhodnější.

Posledním ukazatelem kvality výstupu je atributová tabulka a informace v ní obsažené. Obsah jednotlivých atributových tabulek je možné vidět na obrázcích níže.

FID	Shape *	FacilityID	Name	FromBreak	ToBreak
150	Polygon	1	Location 1 : 1800 - 2095,8904528194	1800	2095,890453
252	Polygon	1	Location 1 : 900 - 1800	900	1800
499	Polygon	1	Location 1 : 0 - 900	0	900
312	Polygon	2	Location 2 : 900 - 1602,41999046163	900	1602,41999
494	Polygon	2	Location 2 : 0 - 900	0	900

Obr. 20 Podoba atributové tabulky výstupu metody s využitím bariér pro analýzu Service Area

FID	Shape *	Name	FromBreak	ToBreak
0	Polygon	0 : 0 - 900	0	900
1	Polygon	0 : 1800 - 2700	1800	2700
2	Polygon	0 : 2700 - 3600	2700	3600
3	Polygon	0 : 900 - 1800	900	1800
4	Polygon	1 : 0 - 900	0	900
5	Polygon	1 : 1800 - 2700	1800	2700
6	Polygon	1 : 2700 - 3600	2700	3600
7	Polygon	1 : 900 - 1800	900	1800

Obr. 21 Podoba atributové tabulky výstupu metody využívající VBA funkce pro analýzu Service Area

FID	Shape *	Name	FromBreak	EndBreak
0	Polygon	Aš : vice nez 3600	3600	vice nez 3600
1	Polygon	Aš : 2700 - 3600	2700	3600
2	Polygon	Aš : 1800 - 2700	1800	2700
3	Polygon	Aš : 900 - 1800	900	1800
4	Polygon	Aš : 0 - 900	0	900
5	Polygon	Benešov : 2700 - 3600	2700	3600
6	Polygon	Benešov : 1800 - 2700	1800	2700
7	Polygon	Benešov : 900 - 1800	900	1800
8	Polygon	Benešov : 0 - 900	0	900

Obr. 22 Podoba atributové tabulky výstupu metody využívající Python pro analýzu Service Area

Všechny tři atributové tabulky obsahují atributy s názvem a hraničními hodnotami polygonů. Atributy FID a Shape jsou součástí každé atributové tabulky shapefilu. Jediný atribut navíc je FacilityID u výstupu metody bariér. Jednotlivé tabulky se od sebe liší hodnotami v atributu Name. U metody bariér je v atributu Name obsažen název regionálních center, jaký jim byl přiřazen při nahrávání do analýzy. U metody využívající VBA funkce název regionálních center odpovídá FID regionu, ve kterém se centrum nachází. Metoda Pythonu do atributu Name přidává uživatelem volený atribut ze vstupní vrstvy center.

Pro uživatele je nejlepší výstup z metody Python. Sám si může ovlivnit, jaký atribut ze vstupní vrstvy center bude obsažen ve výstupní vrstvě. U metody VBA funkcí se nemůže atribut Name měnit, jelikož slouží k výběru polygonů uvnitř regionů. U metody bariér by se podobně jako u Pythonu mohl volit název z atributů vstupní vrstvy. Ovšem metoda bariér je zpracována v ModelBuilderu a vstupní podmínkou je nezávislost na vstupních datech. Jelikož nejsou předem známa vstupní data, není schopen ModelBuilder nabídnout uživateli atributy obsažené v atributové tabulce vrstvy center. Aby to bylo možné, musela by se předem znát vstupní data a název atributu určit napevno a nebo zpracovat celý algoritmus v Pythonu.

Závěrečné hodnocení

Z kvalitativního hlediska je nejvhodnější metoda využívající Python. Jednak jako jediná obsahuje název regionálních center volený uživatelem, jednak vypočítá dostupnost všech lokalit v regionu nehledě na zadané hraniční hodnoty polygonů. Důležitým aspektem může pro některé uživatele být, že po dokončení analýzy pomocí Pythonu dojde k vymazání všech dat vzniklých v průběhu výpočtu. Počítač tak není zahlcován zbytečnými daty a navíc jejich objem je ve srovnání s metodou využívající VBA funkce minimální. Modely zpracované v ModelBuilderu ukládají dočasná data do tempu počítače a po skončení analýzy tam data zůstávají. Mimoto se musí mezi každými spuštěními modelů z ModelBuilderu restartovat program, ve kterém jsou spouštěny, aby se vymazala data z paměti počítače a mohla být nahrazena jinými. V případě skriptů je povolen přepis dat v paměti, takže lze spouštět skripty opakovaně za sebou bez nutnosti restartu softwaru. Vyšší kvalita výstupů i samotného procesu je ovšem vykoupena větší časovou náročností skriptů.

5.3.2 OD Cost Matrix

Hodnocení analýzy OD Cost Matrix se taktéž dělí na dvě části.

Výpočetní čas

Srovnání výpočetních časů probíhalo pouze pro tři metody a jen pro každou úroveň počtu regionů. Testování proběhlo na stejných počítačích jako u předchozí analýzy. Tabulka 4 obsahuje časy dosažené na notebooku s ArcGISem 9.2.

Tab. 4 Výpočetní čas metod pro různá vstupní data a analýzu OD Cost Matrix v ArcGIS 9.2 [sekundy]

Náročnost	Bariéry	VBA funkce	Python
166	836	1 684	1 793
55	267	268	519
18	104	61	171

Zkoumáním údajů lze dojít k podobným závěrům jako dříve. Metoda bariér je nejrychlejší metodou pro náročnější vstupní data. Pro málo náročná data je nejvhodnější metoda VBA funkcí, jelikož rovnou počítá výsledek a nevytváří vedlejší data nebo postupy. V tomto případě dokázala metoda VBA funkcí vypočítat dostupnost i pro nejnáročnější vstup skládající se ze 166 regionů a 6 248 obcí. Rozdíl mezi dávkovým zpracováním Pythonem a VBA funkcemi se s rostoucím objemem vstupních dat snižuje, ale pro použitá testovací data není nikdy dávkové zpracování rychlejší.

V tabulce 5 jsou uvedeny časy pro stejná vstupní data a zpracování v ArcGISu 9.3. Vzhledem k nárůstu náročnosti funkce *Add Locations*, jsou časy několikanásobně větší než u ArcGIS 9.2. Kromě regionálních center jsou u analýzy OD Cost Matrix nahrávány touto funkcí i obce a v případě metody bariér ještě bariéry. To vede k neúměrnému zvýšení časové náročnosti výpočtu.

Tab. 5 Výpočetní čas metod pro různá vstupní data a analýzu OD Cost Matrix v ArcGIS 9.3 [sekundy]

Náročnost	Bariéry	VBA funkce	Python
166	21 074	15 053	14 887
55	7 104	4 181	4 232
18	2 561	1 404	1 450

Metoda bariér se opět u ArcGIS 9.3 stala nejpomalejší metodou. Rozdíly mezi zbylými dvěma metodami jsou ovšem minimální. Z dosažených výsledků není patrný trend jako u ArcGISu 9.2, kdy s rostoucí náročností se rozdíly mezi oběma metodami zmenšovaly. Sice pro 166 regionů je metoda Pythonu rychlejší, ale u dvou menších náročností je rozdíl mezi metodami takřka stejný. Z výsledků obsažených v tabulce 5 se tak dají těžko usuzovat nějaké závěry.

Kvalita výstupu

Hodnocení kvality výstupu se nejprve zaměřilo na výstupní vrstvu. Srovnání metod bylo provedeno na výstupních vrstvách pro 166 regionů. Jelikož analýza není u žádné z metod omezována, správně by měla být pro každou obec vypočtena jedna hodnota dostupnosti do jejího regionálního centra. Ve výstupní vrstvě by tedy mělo být 6 248 záznamů. Výstupní

vrstva metody bariér obsahovala 5 672 linií, ostatní metody zmíněných 6 248. Nižší počet záznamů u metody bariér je dán tím, že některé obce jsou od svých regionálních center odříznuty, jestliže výpočetní algoritmus může využít jen komunikací uvnitř regionu. Je tedy evidentní, že bariéry nejsou vhodným řešením, pokud je překročení hranic regionů povoleno. Na základě použitých algoritmů by každá obec měla být přiřazena pouze ke svému regionálnímu centru, ale pro jistotu byla provedena kontrola. Funkce *Summarize* neodhalila u žádné metody duplicitní záznamy.

Druhou částí hodnocení výstupu bylo porovnání samotných vypočtených hodnot dostupnosti. U analýzy OD Cost Matrix již nemůže dojít k odchylkám v závislosti na zvolených hraničních hodnotách, jelikož je vypočtena přesná hodnota dostupnosti. Spojením atributových tabulek na základě polohy linií bylo možné porovnat hodnoty dostupnosti do jednotlivých obcí pro všechny výpočetní metody. Srovnání metody VBA funkcí a Pythonu ukázalo naprostou shodu, kdy se vypočtené hodnoty dostupnosti nelišily ani v setinách sekund. Jelikož byly výstupy co do počtu záznamů i hodnot dostupnosti identické, byla metoda bariér porovnána pouze s metodou Pythonu. U 5 342 obcí byla vypočtena metodou bariér shodná hodnota dostupnosti. To představuje více jak 85 % všech obcí. Pro 330 obcí byla vypočtená hodnota odlišná. Ve všech případech byla u metody bariér větší. Nejmenší odchylka byla 0,5 sekundy, největší 3 373 sekund, tedy bezmála hodina. Průměrná hodnota odchylky byla 473 sekund a medián 238 sekund, jestliže jsou uvažovány pouze odlišné záznamy. V 16 případech byla odchylka větší než 30 minut a převážně to jsou obce spadající pod Prahu v okrajové části jejího regionu.

Poslední hodnocení se opět týká atributových tabulek výstupních vrstev metod. Obrázky níže názorně ukazují jejich podobu včetně jejich obsahu.

FID	Shape *	Name	OriginID	Destinatio	Destinat_1	Total_cas
0	Polyline	Location 1 - Location 5	1	5	1	0
1	Polyline	Location 1 - Location 3	1	3	2	775,597362
2	Polyline	Location 1 - Location 1	1	1	3	803,622844
3	Polyline	Location 1 - Location 24	1	24	4	919,209861
4	Polyline	Location 1 - Location 50	1	50	5	990,655913
5	Polyline	Location 1 - Location 15	1	15	6	1000,504856
6	Polyline	Location 1 - Location 2	1	2	7	1038,635254

Obr. 23 Podoba atributové tabulky výstupu metody s využitím bariér pro analýzu OD Cost Matrix

FID	Shape *	Name	OriginID	Destinatio	Destinat_1	Total_cas	start_ID	cil_ID
0	Polyline	0 - 0	1	5680	1	0	0	0
1	Polyline	0 - 0	1	5709	2	410,131864	0	0
2	Polyline	0 - 0	1	5746	3	557,982481	0	0
3	Polyline	0 - 0	1	5472	4	838,625923	0	0
4	Polyline	0 - 0	1	5861	5	1039,747932	0	0
5	Polyline	1 - 1	2	3551	1	0	1	1
6	Polyline	1 - 1	2	3634	2	414,450323	1	1

Obr. 24 Podoba atributové tabulky výstupu metody využívající VBA funkce pro analýzu OD Cost Matrix

	FID	Shape *	Total_cas	centrum	lokalita
	0	Polyline	0	Aš	Aš
	1	Polyline	410,131864	Aš	Krásná
	2	Polyline	557,982481	Aš	Podhradí
	3	Polyline	838,625923	Aš	Hazlov
	4	Polyline	1039,747932	Aš	Hranice
	5	Polyline	0	Benešov	Benešov
	6	Polyline	414,450323	Benešov	Chlístov
	7	Polyline	497,703184	Benešov	Mrač

Obr. 25 Podoba atributové tabulky výstupu metody využívající Python pro analýzu OD Cost Matrix

Nejmenší počet atributů obsahuje atributová tabulka výstupní vrstvy z metody Pythonu. Kromě povinných atributů jsou v ní obsaženy atribut s regionálním centrem, cílovým místem a celkovou hodnotou dostupnosti mezi danými místy. Výstupní vrstva bariér obsahuje všechny atributy, které jsou součástí výstupu z analýzy OD Cost Matrix. Stejné atributy a navíc start_ID a cil_ID obsahuje výstupní vrstva VBA funkcí.

Obsah informací v atributech z analýzy není pro běžného uživatele podstatný, a proto by tyto atributy bylo možné odstranit. Není tak učiněno ze dvou důvodů. Jednak informace navíc běžnému uživateli obvykle nevadí a pro některé účely mohou být užitečné, jednak jejich odstranění by jen vedlo ke zvýšení náročnosti algoritmů. Tyto atributy nejsou součástí pouze výstupu ze skriptu, jelikož tento výstup je co nejvíce zjednodušen pro koncového uživatele a jsou na něm demonstrovány výhody programovacího jazyka oproti ModelBuilderu. Z hodnocení obsahu atributů je nejdůležitější atribut Name. U metody bariér opět obsahuje algoritmem přiřazené názvy regionálním centrům a lokalitám. Metoda VBA funkcí má v atributu Name vyplněné FID regionů regionálních center a lokalit. Tyto hodnoty jsou zkopírovány do atributů start_ID a cil_ID a na základě porovnání jejich hodnot byly vybrány hledané linie do výstupu. Výstupní vrstva z metody Pythonu atribut Name ani neobsahuje. Je nahrazen dvěma jinými atributy – centrum a lokalita. Atributovou tabulku tak lze řadit nejen podle regionálních center, ale také podle cílových lokalit.

Zatímco obsah atributů u metody Python není potřeba nijak dál měnit, u metody VBA funkcí to není možné, jelikož jsou důležité pro identifikaci hledaných linií. U metody bariér by se opět dal zvolit atribut ze vstupních vrstev s názvem center a lokalit, ale bohužel ze stejných důvodů jako u analýzy Service Area je to pro ModelBuilder nemožné.

Závěrečné hodnocení

Pro analýzu OD Cost Matrix v zásadě platí stejné závěry jako u analýzy Service Area. Z kvalitativního pohledu se jako nejlepší jeví metoda Pythonu. Ovšem kvalita výstupu i celého algoritmu je kompenzována větší časovou náročností zpracování.

KAPITOLA 6

Shrnutí

Práce si kladla za cíl nalézt metody řešení problematiky, vzájemně je porovnat a zhodnotit. Celkem byly nalezeny tři metody, přičemž každá má svá pro a proti. Nelze tedy říci, že by některá byla nejlepší ve všech ohledech. Vždy jsou některé výhodné vlastnosti vyváženy těmi zápornými. V následujícím textu budou jednotlivé metody připomenuty a porovnány hypotézy vzniklé na počátku práce s praktickými výsledky testování.

6.1 Metoda s využitím bariér

První metoda používá bariér na hranicích regionů, aby zde zastavila výpočet. Základní předpokládanou výhodou byla rychlost výpočtu a relativní jednoduchost na implementaci. Jak je možné vidět v příloze, kde jsou schémata všech modelů vytvořených v ModelBuilderu, je výpočet dostupnosti touto metodou opravdu poměrně snadný. Po nahrání všech bodových vrstev nutných k analýze proběhne jen samotný výpočet dostupnosti a výstup je hotový. Pokud jde o rychlost výpočtu, v ArcGISu 9.2 se potvrdil původní předpoklad a metoda s využitím bariér je nejrychlejší pro větší objemy vstupních dat. Pro vstupy s malým počtem regionů je časově výhodnější metoda využívající VBA funkce. Velmi překvapující je proto časová náročnost metody v softwaru ArcGIS 9.3. Neočekávaný a nezanedbatelný nárůst výpočetní doby u funkce *Add Locations* vedl k degradaci metody na suverénně nejpomalejší způsob řešení. Vzhledem k tomu, že není znám přesný důvod této změny, nelze metodu zcela zavrhnout. Přesto další nevýhody potvrzené testováním značně metodu bariér znevýhodňují oproti dalším metodám.

Zmíněné nevýhody souvisí s kvalitou výstupu. Absence výpočtu dostupnosti v místech, která nejsou propojena s regionálním centrem komunikacemi přes území regionu, je vážným nedostatkem. Výstupy z analýzy Service Area jsou navíc zatíženy chybami způsobenými

interpolací polygonů mezi úseky sítě. Oba problémy mohou vést k mylné interpretaci dosažených výsledků, a to hlavně u analýzy Service Area. Analýza OD Cost Matrix pouze nedokáže vypočíst dostupnost do odříznutých lokalit. Jediným vhodným použitím tak zůstává aplikace metody pro regiony, které jsou uzavřené vůči okolí a není možné překročit jejich hranice (například státy). Jeden z mála nedostatků, který by se dal vylepšit souvisí s obsahem atributu Name. Pokud by se metoda převedla do programovacího jazyka Python, kterým by se nahradilo zpracování v ModelBuilderu, mohl by atribut Name obsahovat názvy regionálních center a případně i lokalit. Nejednalo by se však o nové řešení, pouze o kosmetickou změnu stávajícího, proto nebyla tato možnost realizována.

Celkově lze říci, že metoda bariér splnila očekávání. Nebýt problematické funkce *Add Locations* v ArcGIS 9.3, mohla by být použita pro rychlý přehled o stavu dostupnosti v regionech. Důrazně se doporučuje nepoužívat tuto metodu pro přesné zkoumání dostupnosti, jestliže může být překročena hranice jednotlivých regionů. Jestliže má být vypočtena dostupnost v uzavřených regionech, je to jediná správná metoda. U ostatních nelze zaručit, že by výpočet neprobíhal přes hranice regionů.

6.2 Metoda využívající VBA funkce

Hlavním předpokladem metody bylo odstranění nedostatků předchozí metody s využitím bariér. Výsledky testování to potvrzují. Dostupnost je vypočtena i v místech, která nejsou spojena s regionálním centrem přes území regionu, a ani nedochází k přesahům polygonů dostupnosti do sousedních regionů. Navíc je metoda velmi rychlá pro nízké hraniční hodnoty dostupnosti u analýzy Service Area nebo pro malý počet regionů. Hardwarová a časová náročnost algoritmu se tak ne zcela potvrdila. Problémy nastávají až pro velký objem vstupních dat. V případech, kdy dochází ke vzniku mnoha nadbytečných polygonů nebo linií, je výpočet pomalý a vzniká během něho velké množství dočasných dat. V jednom případě byla porušena integritní omezení shapefilu a model byl předčasně ukončen.

Velkým nedostatkem v kvalitě výstupu je obsah atributu Name. Nejenže neobsahuje názvy regionálních center nebo případně lokalit, navíc nelze jeho obsah změnit. Obsahem atributu musí být FID regionu, ve kterém se nachází regionální centra nebo lokality (u analýzy OD Cost Matrix), aby mohl být vytvořen správný výstup. Úpravou výstupu lze dosáhnout toho, že k jednotlivým polygonům bude přiřazen i název regionů. Ovšem ModelBuilder není vhodným nástrojem k řešení. Vhodnější by bylo převést metodu do programovacího jazyka. Nicméně další úpravy by vedly jen k dalšímu nárůstu časové náročnosti, což není vhodná cesta k dosažení ideálního řešení.

Bez problému s podobou atributové tabulky výstupu by byla metoda využívající VBA funkce dobrým řešením problematiky. Především pro menší objem vstupních dat a nebo pro regiony s přibližně stejnou velikostí by byla nejvhodnějším způsobem řešení. Takto se hodí

pro účely, kdy na atributové tabulce nezáleží a důležitá je pouze podoba výstupu. Pokud jsou regiony různě veliké, je nutné u analýzy Service Area zadávat takové hraniční hodnoty, aby byly pokryté i ty největší regiony. To vede k velkým přesahům polygonů u malých regionů a obrovskému nárůstu náročnosti algoritmu. Pro tyto případy je vhodnější použít metodu další.

6.3 Metoda využívající programovací jazyk Python

Poslední nalezená metoda měla pomoci odstranit nevýhody metody využívající VBA funkce. Předpokládaným nedostatkem byla velká hardwarová a časová náročnost a integritní omezení shapefilu. V průběhu testování metody využívající VBA funkce se sice ukázalo, že integritní omezení mohou být problematické, ale jako největší nedostatek se jeví obsah atributu Name ve výstupní vrstvě. Porovnají-li se atributové tabulky obou výstupů, je u metody zpracované v jazyce Python vidět kvalitativní posun. Atributová tabulka obsahuje názvy center případně i lokalit. Výhodné pro uživatele je, že si sám může určit, který atribut ze vstupních dat bude obsažen ve výstupu. V ostatních ukazatelích hodnocení kvality výstupu si byly při testování obě metody rovnocenné. Jeden rozdíl je patrný u analýzy Service Area, u které je pomocí Pythonu vypočítána dostupnost ve všech částech regionů, i když uživatel zadá příliš malé hraniční hodnoty. Přínos tohoto vylepšení je individuální pro každého uživatele, a proto nebude brán příliš v potaz.

Síla dávkového zpracování dat se ukázala až při velkém objemu vstupních dat. Původně se očekávalo rychlejší zpracování. Dosažené výsledky se značně liší u obou testovacích počítačů, takže je složité usuzovat, na kolik je která metoda rychlejší. Jasné je, že pro menší počet regionů je dávkové zpracování z hlediska času nevhodné.

Nespornou výhodou metody je kvalita jejího výstupu. Díky tomu je analýza výsledků pro uživatele jednodušší a rozhodně rychlejší. Naopak samotný průběh skriptů je dávkovým zpracováním zpomalen oproti původnímu předpokladu. Jestliže má být výstup z analýz dostupnosti dále zpracováván a neslouží pouze k prezentaci výsledků, rozhodně se vyplatí použít tuto metodu.

6.4 Celkové hodnocení

Z výše uvedeného vyplývá, že neexistuje jedna univerzální metoda, která by dosahovala nejlepších výsledků za všech situacích. Nelze proto určit nejlepší metodu, pouze lze jednu doporučit podle konkrétních okolností. Kromě vstupních dat je velmi důležitým aspektem účel, k jakému bude výstup z analýz sloužit.

Během hodnocení jednotlivých metod se nabízely další úpravy algoritmů. Všechny návrhy měly společné, že nejvhodnějším nástrojem pro jejich realizaci je programovací jazyk Python. ModelBuilder je výborný nástroj pro automatizaci postupu, ale pro složitější struktury funkcí

není příliš vhodný. Programovací jazyk nabízí často vícero možností a někdy je jediným možným řešením. Úpravy v programovacím jazyce by mohly metodě VBA funkcí zajistit, aby výstup odpovídal současnému výstupu metody v Pythonu. Pro menší objem vstupních dat by tak vznikla ideální metoda řešení. Velké objemy by se však nadále musely kvůli integritním omezením řešit dávkovým zpracováním. Kvalitnější výstup by úpravou v Pythonu mohl vzniknout i pro metodu s využitím bariér.

Srovnáním ModelBuilderu a Pythonu je tedy jasné, že vhodnějším nástrojem pro složité operace je Python. Stejně jako nalezené metody má svá pro a proti, ale zde pro převyšují proti. Podoba výstupu se snáze přizpůsobí potřebám uživatele, je možné využít cyklů a podmínek a mnoho dalšího. Kombinací programovacího jazyka a nástrojů z ArcGISu vznikl silný nástroj pro práci s prostorovými daty.

KAPITOLA 7

Závěr

Cílem práce bylo nalézt metody, jak vypočítat dostupnost na celém území regionu z regionálního centra, aniž by byl výsledek ovlivněn okolními regiony a centry. Celkem bylo vytvořeno šest řešení problematiky, přičemž tři byly pro analýzu dostupnosti Service Area a tři pro analýzu OD Cost Matrix.

Jak vyplynulo z hodnocení všech metod, žádná nepředstavuje ideální řešení. Definice ideálního řešení je individuální záležitostí. Každý uživatel má jiné požadavky, podle kterých si stanovuje ideální řešení. Ovšem v kontextu hodnocení jednotlivých metod ho lze popsat takto: algoritmus, který by dokázal stejně rychle a kvalitně zpracovat analýzy pro malý i velký počet regionů. Ideál v tomto případě reprezentuje univerzální řešení použitelné bez ohledu na účel výstupu nebo vstupní data. Hledání takového algoritmu je věcí zdlouhavou a není jisté, zda by byl vůbec nalezen. Jestliže se bude na hledání takového řešení pohlížet jako na vývoj, jsou metody obsažené v práci prvními kroky k jeho nalezení. Zatímco jedna metoda poskytuje kvalitně zpracovaný výstup za cenu pomalejšího průběhu analýzy, u dalších je větší rychlost kompenzována méně kvalitní výstupem. Dalším krokem dopředu a přiblížením se k univerzálnímu řešení by byly kombinace nalezených metod. Kombinace, které by dokázaly využít silných stránek metod a slabé by minimalizovaly, jsou otázkou dalšího bádání. Jedno je však jisté. Pro další vývoj je nezbytné opustit zpracování v ModelBuilderu a vydat se pouze cestou programovacího jazyka.

Možnosti nabízenými ModelBuilderem pro zpracování problematiky byly již vyčerpány. Naopak programovací jazyk Python ukázal, že spojení jeho předností s funkcemi v softwaru ArcGIS představuje silný nástroj pro práci s prostorovými daty. Funkce pro úpravu řetězců, cykly, podmínky a další nástroje umožňují efektivněji a lépe využít nástroje z ArcGISu. Poté je možné vytvořit takový výstup, který bude svojí kvalitou vyhovovat uživateli.

Cíle práce byly splněny. Základní metody, jak řešit danou problematiku, byly nalezeny. Hodnocením jejich vlastností se objevily nové možnosti a cesty k jejich vylepšení. Vytvořily tak základ pro další vývoj až k univerzálnímu řešení, které by mohlo být ve všech ohledech označené za ideální.

SEZNAM ZDROJŮ INFORMACÍ

ArcČR 500. 2003. [databáze]. Praha : ArcData Praha, 2003.

BRINKE, J. 1999. *Úvod do geografie dopravy*. 1. vydání. Praha : Karolinum, 1999. 112 s. ISBN 80-7184-923-5

CROMLEY, R. G. 1992. *Digital cartography.* Englewood Cliffs : Prentice-Hall, 1992. 317 s. ISBN 0-13-710930-X.

ČAPEK, R. 1979. Isolinie. *Sborník Čs.spol.zem.* 1979, roč. 84, s.263-271.

Český statistický úřad. 2009. *1419-09, Fakta o České republice* [online]. 2009, aktualizováno 9. 10. 2009 [cit. 2010-08-02]. Dostupné z WWW: <http://www.czso.cz/csu/2009edicniplan.nsf/p/1419-09>.

ESRI. 2006. *ArcGIS 9 : ArcGIS Network analyst tutorial* [online]. [USA] : ESRI. 2006, [cit. 2009-12-18]. Dostupné z WWW: <http://webhelp.esri.com/arcgisdesktop/9.2/pdf/Network_Analyst_Tutorial.pdf>.

ESRI. 2007. *ArcGIS 9.2 Desktop Help* [nápověda online]. 2007, poslední revize 15. 3. 2007 [cit. 2010-02-09]. Dostupné z WWW: <http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=welcome>.

ESRI. 2009. *ArcGIS 9.3 Desktop Help* [nápověda online]. 2009, poslední revize 25. 4. 2009 [cit. 2010-06-13]. Dostupné z WWW: <http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=welcome>.

ESRI ARCGIS DESKTOP. 2010. *Which Desktop GIS Software Is Right for You?* [online]. 2010, poslední revize 12. 7. 2010 [cit. 2010-08-11]. Dostupné z WWW: <http://www.esri.com/software/arcgis/about/gis-for-me.html>.

ESRI ARCGIS RESOURCE CENTER. 2010. *HowTo: Use VBA functions in the field calculator* [online]. 2010, poslední revize 17. 6. 2010 [cit. 2010-06-25]. Dostupné z WWW: <http://resources.arcgis.com/content/kbase?fa=articleShow&d=31807 >.

ESRI PRESS RELEASE. 2010. *ArcGIS 10 Transforms the Way People Use GIS* [online]. Publikováno 2010-06-29 [cit. 2010-08-10]. Dostupné z WWW: < http://www.esri.com/news/releases/10_2qtr/arcgis10-download.html>.

HAMPL, M. ; JEŽEK, J. ; KÜHNL, K. 1978. *Sociálně geografická regionalizace ČSR.* Praha : Vědecký ústav sociálně ekonomických informací, 1978. 301 s.

HAMPL, M. ; GARDAVSKÝ, V. ; KÜHNL, K. 1987. *Regionální struktura a vývoj systému osídlení ČSR.* 1. vydání. Praha : Univerzita Karlova, 1987. 255 s.

HAMPL, M. 2005. *Geografická organizace společnosti v České republice: transformační procesy a jejich obecný kontext.* 1. vydání. Praha : Přírodovědecká fakulta UK, 2005. 145 s. ISBN 80-86746-02-X.

HANDY, S. 1993. Regional versus local accessibility : implications for nonwork travel. *Transportation research record.* 1993, č. 1400. Dostupný z WWW: <http://www.uctc.net/papers/234.pdf>. ISSN 0361-1981.

HARMS, D. ; McDONALD, K. 2003. *Začínáme programovat v jazyce Python.* 1. vydání. Brno : Computer Press, 2003. 455 s. ISBN 80-7226-799-X.

KOLÁŘ, J. 2003. *Geografické informační systémy.* 2. přeprac. vydání. Praha : Vydavatelství ČVUT, 2003. 161 s. ISBN 80-01-02687-6.

LOVETT, A. …[et al.]. 2002. Car travel time and accessibility by bus to general practitioner services : a study using patient registers and GIS. *Social science and medicine.* Červenec 2002, roč. 55, č. 1, s. 97-111. Dostupný z WWW: < http://www.ingentaconnect.com/content/els/02779536/2002/00000055/00000001/art00212>. ISSN 0277-9536.

MAX VISUAL BASIC. 2010. *History of Visual Basic* [online]. 2010, poslední revize 12. 7. 2010 [cit. 2010-08-20]. Dostupné z WWW: < http://www.max-visual-basic.com/history-of-visual-basic.html>.

MÜLLER, S. ; TSCHARAKTSCHIEW, S. ; HAASE, K. 2008. Travel-to-school mode choice modelling and patterns of school choice in urban areas. *Journal of transport geography.* Září 2008, roč. 16, č. 5, s. 342-357. Dostupný z WWW: <http://geography.upol.cz/soubory/lide/hercik/SEDOP/Travel-to-school%20mode%20choice%20modelling%20and%20patterns%20of%20school%20choice%20in%20urban%20areas.pdf>. ISSN 0966-6923.

NAVRÁTIL, J. 2008. *Modelování dostupnosti lékáren a nízkoprahových center v Česku* [rukopis]. Praha, 2008. 51 s. Bakalářská práce na Přírodovědecké fakultě Univerzity Karlovy na katedře aplikované geoinformatiky a kartografie.

PROGRAMUJTE.COM. 2006. *Programování : Python, kurz Python* [online]. 2006, poslední revize 26. 8. 2006 [cit. 2010-03-12]. Dostupné z WWW: <http://programujte.com/?rubrika=26&sekce=105-python>.

PROGRAMUJTE.COM. 2009. *Úvod do Tkinter* [online]. 2009, poslední revize 27. 4. 2009 [cit. 2010-04-26]. Dostupné z WWW: < http://tkinter.programujte.com/>.

PyCZ. 2010. *PyCZ : Programovací jazyk Python* [online]. 2010, poslední revize únor 2010 [cit. 2010-03-13]. Dostupné z WWW: < http://www.py.cz/FrontPage>.

PYTHON. 2010. *About Python* [online]. 2010, poslední revize 13. 8. 2010 [cit. 2010-08-13]. Dostupné z WWW: < http://www.python.org/about/>.

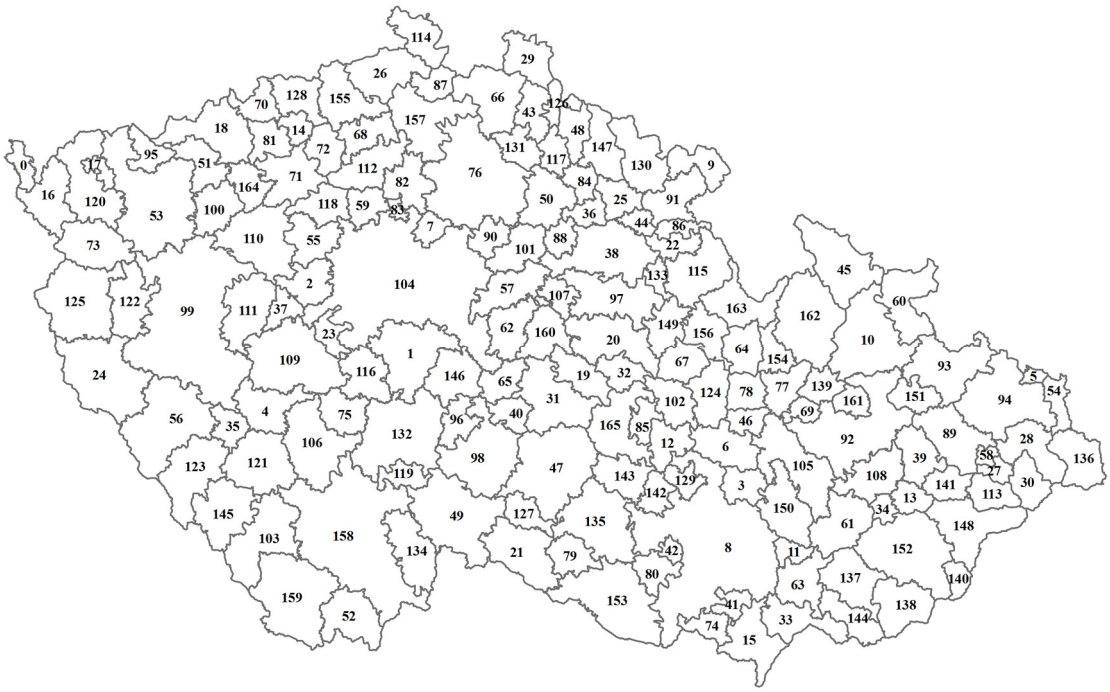
ŠTYCH, P. a kol. 2008. *Vybrané funkce geoinformačních systémů*. Praha : Česká kosmická kancelář, 2008. 177 s.

WALSH, S. J. ; PAGE, P. H. ; GESLER, W. M. 1997. Normative models and healthcare planning : network-based simulations within a geographic information system environment. *Health services research*. Červen 1997, roč. 32, č. 2, s. 243-260. Dostupný z WWW: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1070185>. ISSN 0017-9124.

WEBER, J. 2003. Individual accessibility and distance from major employment centers : An examination using space-time measures. *Journal of geographical systems*. Květen 2003, roč. 5, č. 1, s. 51-70. Dostupný z WWW: <http://www.springerlink.com/content/5g6a8th5j668ar2a/>. ISSN 1435-5949.

SEZNAM PŘÍLOH

Příloha 1 Regiony mikroregionální úrovně 1. stupně sociogeografické regionalizace Česka

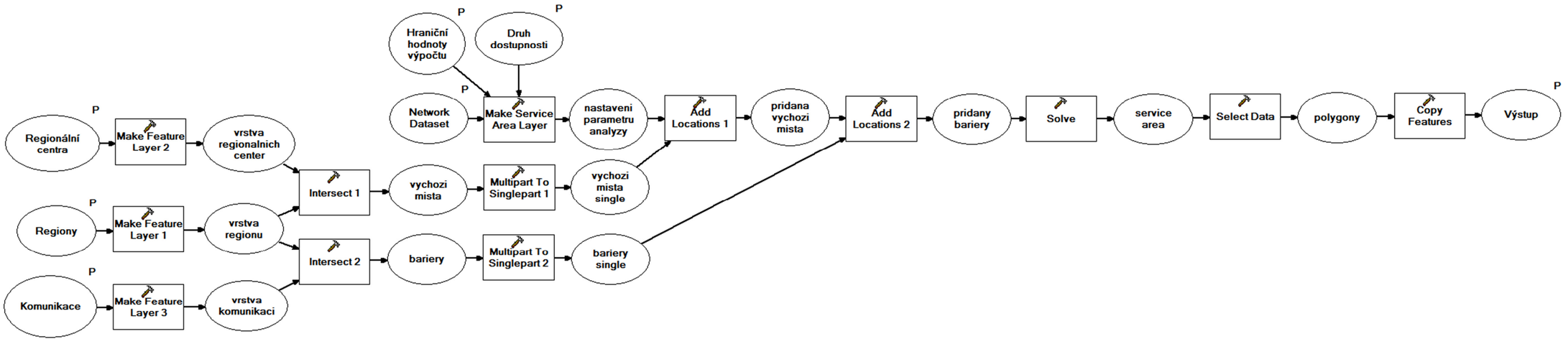


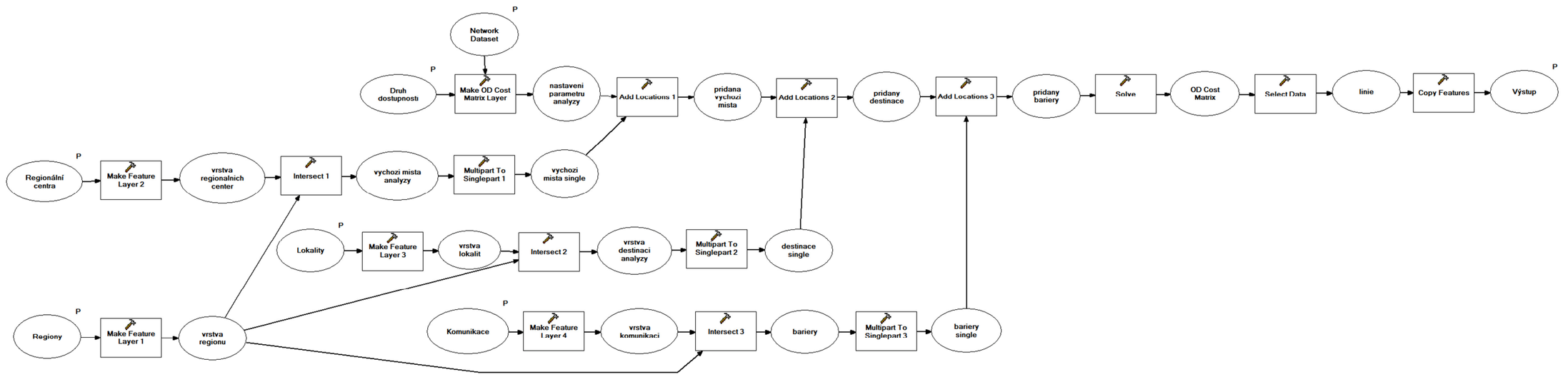
0	Aš	**42**	Ivančice	**84**	Nová Paka	**126**	Tanvald
1	Benešov	**43**	Jablonec nad Nisou	**85**	Nové Město na Moravě	**127**	Telč
2	Beroun	**44**	Jaroměř	**86**	Nové Město nad Metují	**128**	Teplice
3	Blansko	**45**	Jeseník	**87**	Nový Bor	**129**	Tišnov
4	Blatná	**46**	Jevíčko	**88**	Nový Bydžov	**130**	Trutnov
5	Bohumín	**47**	Jihlava	**89**	Nový Jičín	**131**	Turnov
6	Boskovice	**48**	Jilemnice	**90**	Nymburk	**132**	Tábor
7	Brandýs nad Labem-Stará Boleslav	**49**	Jindřichův Hradec	**91**	Náchod	**133**	Týniště nad Orlicí
8	Brno	**50**	Jičín	**92**	Olomouc	**134**	Třeboň
9	Broumov	**51**	Kadaň	**93**	Opava	**135**	Třebíč
10	Bruntál	**52**	Kaplice	**94**	Ostrava	**136**	Třinec
11	Bučovice	**53**	Karlovy Vary	**95**	Ostrov	**137**	Uherské Hradiště
12	Bystřice nad Pernštejnem	**54**	Karviná	**96**	Pacov	**138**	Uherský Brod
13	Bystřice pod Hostýnem	**55**	Kladno	**97**	Pardubice	**139**	Uničov
14	Bílina	**56**	Klatovy	**98**	Pelhřimov	**140**	Valašské Klobouky
15	Břeclav	**57**	Kolín	**99**	Plzeň	**141**	Valašské Meziříčí
16	Cheb	**58**	Kopřivnice	**100**	Podbořany	**142**	Velká Bíteš
17	Chodov	**59**	Kralupy nad Vltavou	**101**	Poděbrady	**143**	Velké Meziříčí
18	Chomutov	**60**	Krnov	**102**	Polička	**144**	Veselí nad Moravou
19	Chotěboř	**61**	Kroměříž	**103**	Prachatice	**145**	Vimperk
20	Chrudim	**62**	Kutná Hora	**104**	Praha	**146**	Vlašim
21	Dačice	**63**	Kyjov	**105**	Prostějov	**147**	Vrchlabí
22	Dobruška	**64**	Lanškroun	**106**	Písek	**148**	Vsetín
23	Dobříš	**65**	Ledeč nad Sázavou	**107**	Přelouč	**149**	Vysoké Mýto
24	Domažlice	**66**	Liberec	**108**	Přerov	**150**	Vyškov
25	Dvůr Králové nad Labem	**67**	Litomyšl	**109**	Příbram	**151**	Vítkov
26	Děčín	**68**	Litoměřice	**110**	Rakovník	**152**	Zlín
27	Frenštát pod Radhoštěm	**69**	Litovel	**111**	Rokycany	**153**	Znojmo
28	Frýdek-Místek	**70**	Litvínov	**112**	Roudnice nad Labem	**154**	Zábřeh
29	Frýdlant	**71**	Louny	**113**	Rožnov pod Radhoštěm	**155**	Ústí nad Labem
30	Frýdlant nad Ostravicí	**72**	Lovosice	**114**	Rumburk - Varnsdorf	**156**	Ústí nad Orlicí
31	Havlíčkův Brod	**73**	Mariánské Lázně	**115**	Rychnov nad Kněžnou	**157**	Česká Lípa
32	Hlinsko	**74**	Mikulov	**116**	Sedlčany	**158**	České Budějovice
33	Hodonín	**75**	Milevsko	**117**	Semily	**159**	Český Krumlov
34	Holešov	**76**	Mladá Boleslav	**118**	Slaný	**160**	Čáslav
35	Horažďovice	**77**	Mohelnice	**119**	Soběslav	**161**	Šternberk
36	Hořice	**78**	Moravská Třebová	**120**	Sokolov	**162**	Šumperk
37	Hořovice	**79**	Moravské Budějovice	**121**	Strakonice	**163**	Žamberk
38	Hradec Králové	**80**	Moravský Krumlov	**122**	Stříbro	**164**	Žatec
39	Hranice	**81**	Most	**123**	Sušice	**165**	Žďár nad Sázavou
40	Humpolec	**82**	Mělník	**124**	Svitavy		
41	Hustopeče	**83**	Neratovice	**125**	Tachov		

Příloha 2 Kód v Pythonu vkládaný do ToolValidatoru v ArcGIS 9.3

```
class ToolValidator:
 """Class for validating a tool's parameter values and controlling
 the behavior of the tool's dialog."""

 def __init__(self):
  """Setup the Geoprocessor and the list of tool parameters."""
  import arcgisscripting as ARC
  self.GP = ARC.create()
  self.params = self.GP.getparameterinfo()

 def initializeParameters(self):
  """Refine the properties of a tool's parameters. This method is
  called when the tool is opened."""
  return

 def updateParameters(self):
  """Modify the values and properties of parameters before internal
  validation is performed. This method is called whenever a parmater
  has been changed."""
  # Ulozeni nazvu atributu network datasetu, podle kterych se muze dostupnost
  # vypocist, do seznamu. Polozky seznamu jsou pote nabidnuty uzivateli jako
  # parametr cislo 5 skriptu.
  moznosti = []
  stringFilter = self.params[4].Filter
  nd = self.params[3].Value
  desc = self.GP.describe(nd)
  attributes = desc.attributes
  attribute = attributes.Next()
  while attribute:
    moznosti = moznosti + [attribute.name]
    attribute = attributes.Next()
  stringFilter.List = moznosti

  return

 def updateMessages(self):
  """Modify the messages created by internal validation for each tool
  parameter. This method is called after internal validation."""
  return
```

Příloha 3 Modely s využitím bariér pro analýzy Service Area (nahoře) a OD Cost Matrix (dole)





Příloha 4 Modely využívající VBA funkce pro analýzy Service Area (nahoře) a OD Cost Matrix (dole)