EÖTVÖS LORÁND TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR

A térképi generalizálás automatizálása a vízrajz példáján

SZAKDOLGOZAT
FÖLDTUDOMÁNYI ALAPSZAK

Készítette:
Balogh Dániel
térképész és geoinformatikus szakirányú hallgató

Témavezető:
Ungvári Zsuzsanna
tanársegéd
ELTE Térképtudományi és Geoinformatikai Tanszék

Budapest, 2015

Tartalomjegyzék

1. Bevezetés

Dolgozatom témája a térképi generalizálás automatizálása és annak gyakorlati alkalmazásai. Korábban is érdekesnek tartottam, hogy két, ugyanazon térkép felhasználásával készült kisebb méretarányú térkép mennyiben különbözhet egymástól az általuk közölt információ tekintetében. Mindkét mű beteljesíti feladatát: tájékoztatja olvasóját, mégis a két befogadó személy máshogy ítéli meg az ábrázolt terület bizonyos elemeinek fontosságát. Ebben rejlik a generalizálás szépsége és egyben nehézsége is.

Tanulmányaim alatt több térinformatikai szoftverrel is megismerkedtem. Ezek használatakor feltűnt, hogy a különböző programok ugyanazt a generalizálási folyamatot más módszerekkel végezik el. Igen érdekesnek találtam a más-más algoritmusok felhasználhatósága közötti eltéréseket.

Dolgozatom célja, hogy ismertessem a generalizálás lényegét, az automatizálásának. rövid történetét, valamint a különböző eljárások hatékonyságát; mindezt a vízrajz példáján.

2. Generalizálás

2.1. A generalizálás definíciója

A térképen maximálisan megjeleníthető információ mennyiségét a térkép befogadóképességének nevezzük; generalizálásnak pedig azt a folyamatot hívjuk, amely során csökkentjük, kiválogatjuk a térképre kerülő információ mennyiségét. Generalizálásnál számos tényezőt kell figyelembe venni, ezek közül a legfontosabbak pedig: a térkép méretaránya, annak célja, az olvasóközönség, akiknek a térkép készül, valamint a technológiai korlátok.

A klasszikus kartográfiában megkülönböztetünk alaptérképeket, valamint levezetett térképeket. Alaptérképek egy adott terület lehető legnagyobb méretarányban elkészített, a lehető legnagyobb befogadóképességgel rendelkező térkép. A levezetett térképek pedig általában kisebb méretarányú, generalizált térképek.

Már a 20. század eleje óta foglalkoztatja a térképészeket a generalizálás problémája. Max Eckert német kartográfus 1908-ban azt írta: „A generalizálásban rejlik a tudományos térképkészítés nehézsége, mivel nem engedi a kartográfusnak, hogy csupán az objektív tényekre hagyatkozzon, hanem szükségessé teszi számára azok szubjektív értelmezését.” John K. Wright két fő komponensét ismerte fel az egyszerűsítést, valamint a kiemelést, vagyis az elszórt információk hangsúlyozását. Az a gondolat, hogy a generalizálást fel lehet bontani e két logikai folyamattá képezte alapját a későbbi kutatásoknak. Raisz Ervin nem csak a domborzatábrázolás továbbfejlesztéséhez járult nagyban hozzá, hanem megállapított három fő generalizálási komponenst, a kombinációt, az elhagyást, valamint az egyszerűsítést, míg Arthur Robinson és kollégái négy összetevőt véltek felfedezni, a kiválasztást, egyszerűsítést, klasszifikálást (osztályozást), és a szimbolizálást. (Slocum, 2004)

2.2. A digitális tárgy-, és térképi modell

Günther Hake kibővítette Eduard Imhof klasszikus definícióját a térképről, miszerint „a térkép a földfelszín, vagy valamely részletének kicsinyített, egyszerűsített, tartalmilag kiegészített és magyarázott alaprajzi képe”. Hake szerint „a térkép a térbeli vonatkozások mértékhez kötött és rendezett modellje”. Ennek a definíciónak az egyik fontos módosítása az előzőhöz képest, hogy a kicsinyített kifejezés helyett a rendezett kifejezést használja, ezáltal utalva a generalizálásra.

A térképi modellek két csoportra oszthatók, az analóg és a digitális modellekre. Az analóg modell közvetlenül rajzi úton, vagy digitális modellből jön létre, szemlélteti az általa közölt információk térbeli összefüggéseit. Analóg modell a hagyományos értelemben vett térkép. Digitális modellben nem csak az objektumok helyzetét meghatározó információkat írjuk le, hanem ezek szomszédsági viszonyait is. Így jön létre a grafika nélküli digitális tárgymodell, valamint a digitális térképi modell.

A digitális tárgymodellben a felmért adatok függetlenek lehetnek a grafikus módszerektől. Ilyen digitális adatmodell például a népszámlálás adatait tartalmazó adattábla is. A digitális térképi modell pedig a tárgyi modellek összessége kiegészítve az adatokhoz tartozó rajzi utasításokkal. (Klinghammer, 2010)

2.3. Generalizálási modellek, alapvető műveletek

Hogy könnyebben megértsük a generalizálás összetett voltát kutatók különböző modelleket hoztak létre.

Arthur Robinson kollégáival együtt 1978-ban kidolgozott modellje volt az egyik első hivatalos vázlat a generalizálás könnyebb érthetősége céljából. A folyamatot két fő lépésre bontották. Az első a kiválasztás, mely valójában egy előkészítő lépés a második szakaszhoz, ami maga a generalizálás folyamata. A kiválasztás lényege, hogy azonosítsa a megtartandó és elhagyandó objektumokat az adatbázisban. A generalizálás lépés további három folyamatot hordoz magában az egyszerűsítést, a klasszifikálást és a szimbolizálást.

A következő modell Tina Kilpeläinen (1997) modellje új megvilágításban vizsgálta a különböző méretarányhoz tartozó adatbázisokat. Elmélete feltételezi egy kartográfiai alapadatbázis az úgynevezett digitális terepmodell vagy DLM (Digital Landscape Model) létezését, amelyből eljárások sorozatával jönnek létre a digitális kartográfiai modellek, a DCM-ek (Digital Cartographic Model). A kiindulási DLM a legnagyobb méretarányú, legrészletesebb adatbázis. Ebből lehetséges további kisebb méretarányú másodlagos DLM-ek generalizálása. Minden méretarányhoz külön másodlagos DLM készül, amik csupán tárgymodellek, és csak a belőlük hoznánk létre generalizálás és kartografálás után a DCM-eket, ezeket nevezhetjük digitális térképi modelleknek. A DCM-ek már térképen megjeleníthető adatok.

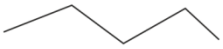
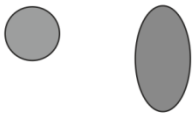
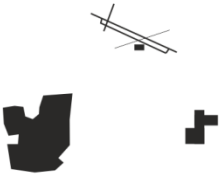
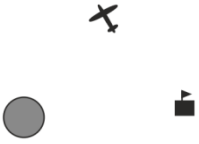
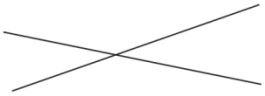
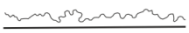
A harmadik modell McMaster és Shea nevéhez fűződik. Az ő törekvésük egy minden részletre kiterjedő generalizálási folyamat kialakítása. Három fő komponenst állapítottak meg amit három kérdés formájában lehet feltenni:

- Miért kell generalizálni? (elvi célok kitűzése)
- Mikor kell generalizálni? (a generalizálás szükségességének felismerése)
- Hogyan kell generalizálni? (a generalizálás alapvető műveleteinek használata)

A generalizálás alapvető céljai közé tartozik a komplexitás csökkentése, a térbeli és tulajdonságbeli pontosság valamint az esztétikai minőség és a logikai hierarchia megtartása, végül a generalizálási szabályoknak következetességének biztosítása. Ezek közül is talán a legfontosabb a komplexitás csökkentése, annak a problémának a kezelése, hogy miként mérsékeljük az információ mennyiségét, egy kisebb méretarányú térkép készítésénél, hogy az megfeleljen a befogadóképességének, és céljának. Hasonlóan fontos a térbeli pontosság megtartása. A térbeli pontosság megtartásának célja, hogy a lehető legkevesebb változtatás mellett a lehető legpontosabban adja vissza a térbeli információkat. Erre példa egy öböl szájának kiszélesítése, ha az a kicsinyítés után zártnak látszana.

Számos körülmény felléphet, amely szükségessé teszi a generalizálást, de ezek közül a hat legfontosabb a zsúfoltság, az egybeolvadás, az ellentmondás, a komplexitás, a következetlenség és az érzékelhetőség nehézsége.

A harmadik kérdés megválaszolásához ismernünk kell a generalizálás alapvető műveleteit. A generalizálási folyamat lebontható logikai eljárások sorozatára, melyek csoportosíthatóak aszerint, hogy milyen típusú térképi objektumra alkalmazhatóak (pontszerű, vonalas vagy felületi). Besorolhatjuk őket aszerint is, hogy raszteres vagy vektoros állományokra használhatók-e fel. A vektor alapú operátorok többnyire jóval összetettebbek, mint a raszter alapú generalizálási műveletek. Ennek oka, hogy míg a raszteres állományokban a szomszédos területek kapcsolatát vizsgáljuk, addig a vektoros rendszereknél x-y koordinátapárok alapján dolgozik az algoritmus. Az alábbi táblázatban az elemi vektoros műveletek láthatóak példákkal, alatta pedig ezek rövid leírása. (Slocum, 2004)

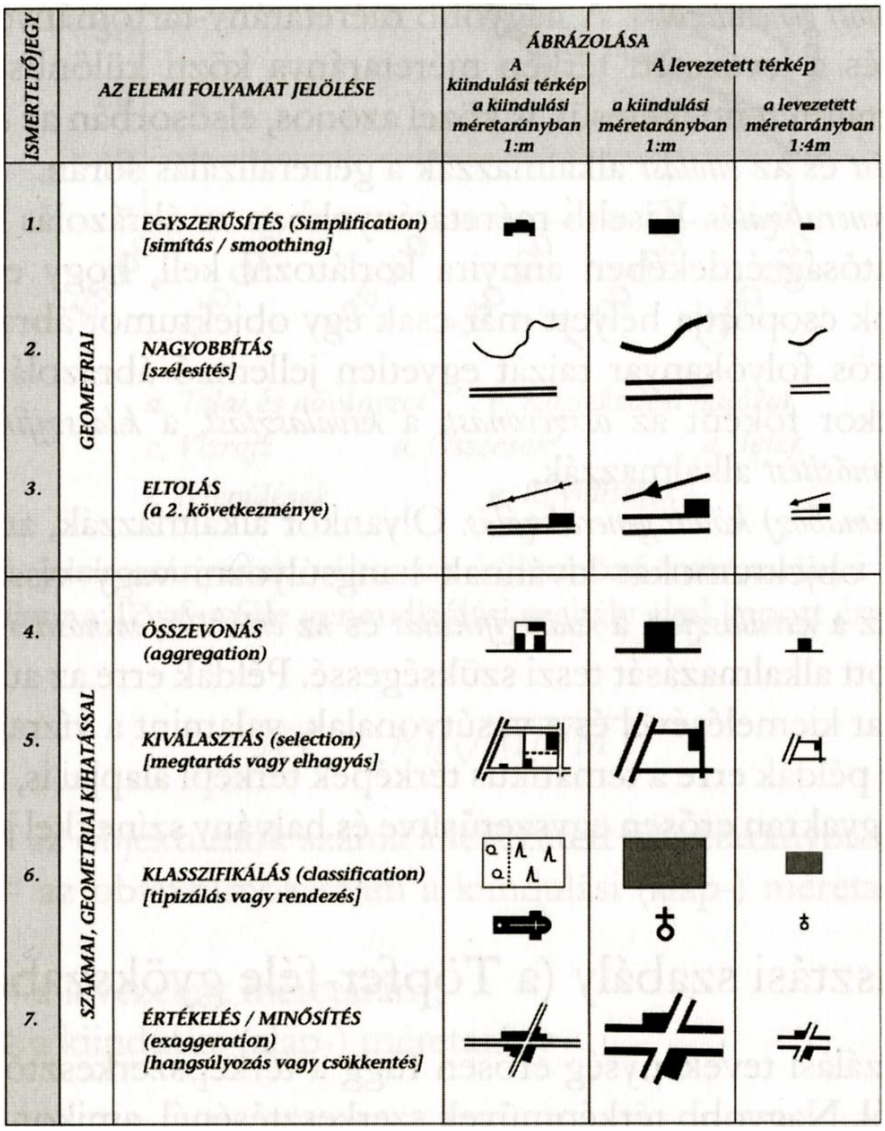
TÉRBELI OPERÁTOR	EREDETI TÉRKÉP	GENERALIZÁLT TÉRKÉP
EGYSZERŰSÍTÉS		
SIMÍTÁS		
HALMAZKÉPZÉS		
EGYESÍTÉS		
OSZTÁLYOZÁS		
ÖSSZEVONÁS		
FINOMÍTÁS		
HANGSÚLYOZÁS		
KIEMELÉS		
ELTOLÁS		

2.3.1. táblázat Alapvető generalizálási eljárások (Slocum 2004)

- Egyszerűsítés: A leggyakrabban használt módszer, vonalas objektumok generalizálására alkalmas. Törli a felesleges koordinátapárokat, úgy, hogy a vonal jellegzetes geometriája a lehető legkisebb mértékben sérüljön. Az általános egyszerűsítési folyamatnak ötféle megközelítése van.
- Simítás: Ellentétben az egyszerűsítéssel, simításkor nem törlődnek pontok. A simítási operátorok egy része megváltoztatja a pontok helyzetét, míg más algoritmusok görbékkel helyettesítik a tört vonalat, ezáltal finomabb futást eredményezve.
- Halmazképzés: Ennél a módszernél egymáshoz közel álló, sűrűn elhelyezkedő pontokból vagy objektumok kisebb csoportjából képzünk egy nagyobb halmazt. Problémás lehet annak a sűrűségnek a meghatározása, mely felett halmazként kezeljük a pontok sokaságát, valamint ezen halmazok határainak megszabása.
- Egyesítés: Egymáshoz közel álló poligonok egyesítésének folyamata.
- Klasszifikálás vagy osztályozás: Osztályozásnál egyes elemeknél átváltunk egy másik megjelenítési módszerre. Erre példa egy település alaprajzszerű megjelenítése helyett a jelmódszer alkalmazása. Tipikusan felületi objektumoknál alkalmazandó.
- Összevonás: Az összevonás az a művelet, amivel vonalas objektumok csoportját egyesítünk, mint például párhuzamos vasúti vonalak esetében.
- Finomítás: Egy kisebb méretarányban aprólékosan, alaprajzszerűen megrajzolt térképi elemet egyszerűsített formában ábrázoljuk, azoknak bizonyos részeit elhagyva.
- Hangsúlyozás: Az egyik leggyakrabban használt művelet; sokszor szükséges egy adott elem bizonyos részletének kihangsúlyozása a térkép olvashatóságának megtartása érdekében.
- Kiemelés: Némely objektumok ábrázolását megváltoztatva hangsúlyozzuk azokat.
- Eltolás: Adott térképi elemek helyzetének megváltoztatása a könnyebb átláthatóság érdekében.

Ezen elemi folyamatok gyakorlati alkalmazására a kutatók különböző algoritmusokat és eljárásokat dolgoztak ki, melyek fejlesztése és finomítása a mai napig egy igen érdekes és fontos feladat.

Kissé eltérően közelítette meg Klinghammer István a generalizálás elemi folyamatait 2010-es könyvében:



2.3.2. táblázat A kartográfiai generalizálás elemi folyamatai
(Térképészet és Geoinformatika I,. 2010)

Ami mellé az alábbi gondolatokat társította:

„Az ábra alapján egyszerűnek és áttekinthetőnek tűnő folyamatok értelemszerű alkalmazása során célszerű a következő szempontokat figyelembe venni:

- a folyamatok nem függetlenek egymástól, és ezért hatásukban sem különíthetők el teljesen egymástól (lásd nagyobbítás, eltolás),
- a folyamatok alkalmazásánál az objektumok meghatározott sorrendjét kell követni. A topográfiai térképeknél például a vízrajzzal és a közlekedési hálózattal kezdik a generalizálást, ezt követi a településhálózat átdolgozása, a domborzati formákra pedig a síkrajzi generalizálás után kerül sor,

- a szabályokat az új térkép méretarányától és céljától függően különböző súllyal és sorrendben kell alkalmazni.” (Klinghammer, 2010)

A Shea és McMaster által leírt műveletekkel szemben, ahol az automatizálás volt a fő szempont, a Klinghammer-féle táblázatban megjelenő folyamatok kissé hagyományosabb úton közelítik meg a generalizálást. Ez látszik azon is, hogy míg számos párhuzamot és átfedést lehet felfedezni a kettő között, némely esetekben a hasonló elnevezés teljesen már eljárást takar. Ilyen például az egyszerűsítés. Míg az előbbi példában – szem előtt tartva az automatizálást – az egyszerűsítés egy felületi objektumot határoló, vagy egy vonalas objektum töréspontjai számának csökkentését jelentette, addig az utóbbiban egy bármely térképi elemre alkalmazható a szó szoros értelmében vett egyszerűsítésről beszélünk. Ugyanez a művelet megfeleltethető az első táblázatban lévő finomítás eljárással. Figyelemre méltó eltérés még, az eltolás különböző alkalmazása. Klinghammer táblázatából és leírásából kiderül, hogy ő az eltolást a nagyobbítás következményeként kezeli és azt írja: „hatásukban sem különíthetők el teljesen egymástól”; ellentétben Shea és McMaster táblázatával, ahol nemcsak hogy teljesen különálló műveletként értelmezik az egyszerűsítést, nagyobbítás eljárást nem is említenek külön. Érdekes megfigyelni, hogy a bővebb táblázat elkülöníti a felületi és vonalas elemek összevonását, míg a másik nem tesz efféle különbséget. Találunk olyan procedúrákat, melyek csak az egyik táblázatban fordulnak elő, ezek a simítás, halmazképzés, hangsúlyozás; és persze vannak olyan folyamatok is, melyek mindkettő táblázatban megtalálhatóak, ilyenek a klasszifikálás, eltolás valamint a minősítés/kiemelés.

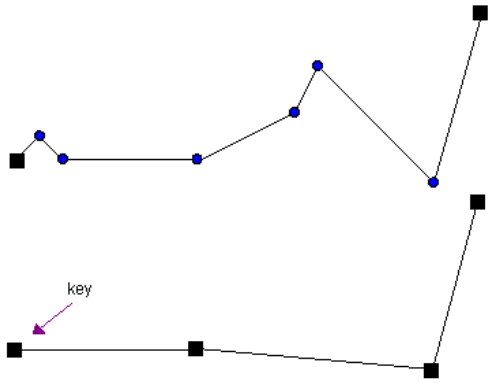
3. Vonalas elemek egyszerűsítése

Számtalan különböző generalizálási segédeszközt találunk az egyes térképészeti szoftverekben. Ezeknek többsége hasonló módszereket használ, dolgozatom következő részében ezek közül fogok bemutatni néhányat, külön vizsgálva a vonalas, és felületi elemekre vonatkozó algoritmusokat.

A vonalas elemekre vonatkozó eszközök közül a két legfontosabbal foglalkoztam, az egyszerűsítéssel és a simítással. Amikor vektoros környezetben vonalas elemről beszélünk, akkor valójában egy több ponton átmenő törött vonalról, úgynevezett polyline-ról van szó. A polyline pontjai egyenes szakaszokkal vannak összekötve, és ezen pontok koordinátái határozzák meg a vonal futását. Egyszerűsítésnél mindegyik operátor lényege, hogy a lehető legtöbb pontot törölje a geometria lehető legjobb megtartása mellett. Számos egyszerűsítési algoritmust használnak a különféle térinformatikai szoftverek; ezeket öt csoportba tudjuk besorolni az alapján, hogy hány pontot használnak és, hogy mennyire veszik figyelembe a pontok közötti kapcsolatokat. Az első csoport a független-pont eljárások. Ezek az algoritmusok a különálló pontokat figyelik, és nincsenek tekintettel azok szomszédjaira. Erre a kategóriára példa az n-edik pont algoritmus, melynek dolga, hogy az első, utolsó és minden n-edik pontot leszámítva törölje a pontokat a vonalból, függetlenül annak fontosságától. Következő kategória a lokális csoport, mely csoportban lévő operátorok közvetlen szomszédsági viszonyait vizsgálja, meghatározva azok jelentőségét. Ilyen eljárás Jenk algoritmusa, mely a szomszédos pontok távolságát és a szögek változását nézi. A harmadik csoport a korlátozottan kiterjesztett lokális módszerű eljárások. Ezek a szomszédos ponton túl is vizsgálják a vonalat, és kiterjesztésük függ a távolságtól, szögtől, valamint a vizsgált pontok számától. Erre példa a Lang-algoritmus, az Opheim-eljárás és a Visvalingam-algoritmus. Az utolsó előtti a nem-korlátozottan kiterjesztett lokális módszerű eljárások csoportja, mely szintén nagyobb terjedelemben veszi figyelembe a vonalat, és csak geomorfológiai komplexitás szab gátat a generalizálásnak. Ilyen algoritmus a Reumann–Witkam-algoritmus, mely a vonal bizonyos egyenes szakaszaival húzott párhuzamosok alapján dolgozik. Az utolsó csoportba a globális eljárások tartoznak, ezek a műveletek az egész vonalat, vagy annak megadott részeit vizsgálják. (NCGIA Core Curriculum, 1994)

3.1. N-edik pont algoritmus

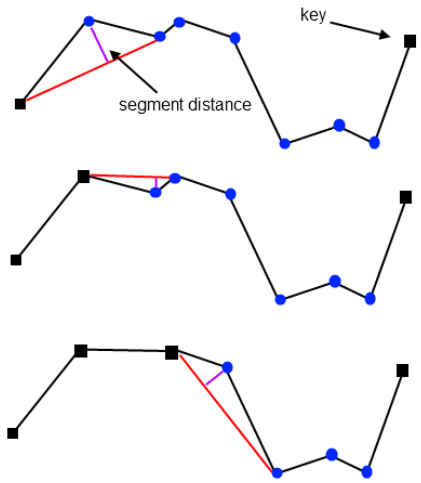
Az n-edik pont algoritmus talán az egyik legegyszerűbb használt eljárás. Ennél az algoritmusnál a vonal első és utolsó pontját mindig meghagyja; a közbülső pontok közül pedig csak minden egy előre megadott n szám többszöröseinek megfelelőek maradnak meg, az összes többi pont törlésre kerül. Ezzel az operátorral igen durva egyszerűsítést érünk el, az eredmény egy bonyolultam vízhálózat esetén meglehetősen torz lehet.



3.1.1. ábra. N-edig pont algoritmus

3.2. Merőleges távolság eljárás

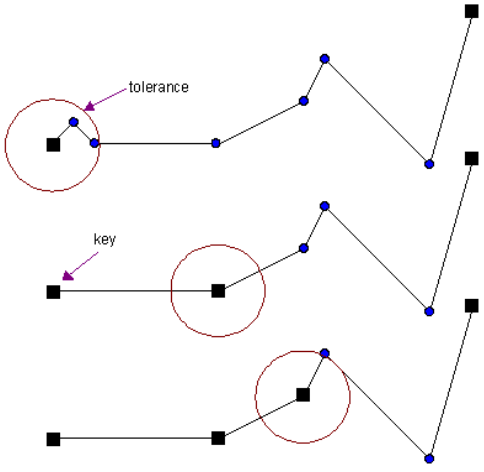
Az egyik legegyszerűbb egyszerűsítési eljárás, a merőleges távolság algoritmus. Egyszerre mindig három pontot vizsgál, mégpedig egy előre megadott határérték alapján. Az eljárás az első ponttól kezdve, minden pontnak vizsgálja az utána következő pont távolságát, az első és a harmadik ponton átmenő egyenestől. Amennyiben ez a távolság kisebb, mint a megadott határérték, úgy törlésre kerül, és az utána lévő ponttól indul újra az eljárás. Ha viszont az egyenestől számított távolság nagyobb, mint a határérték, akkor a pont megmarad, és onnantól kezdődik újra a vizsgálat. Ez az egyszerű algoritmus számos összetettebb eljárásnak szolgál alapul.



3.2.1. ábra. Merőleges távolság eljárás

3.3. Sugaras távolság eljárás

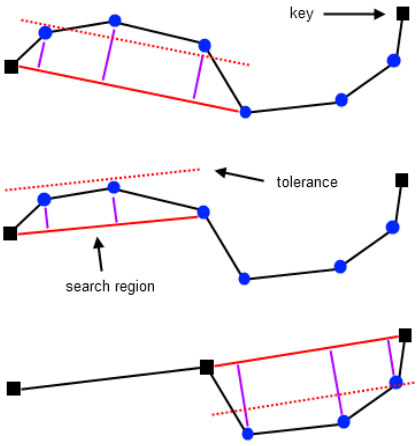
Ez az algoritmus szintén egy toleranciaérték által megadott távolság alapján vizsgálja a pontokat, ezt a távolságot azonban nem egy vonaltól, hanem a különböző pontoktól méri. A vonal első pontjával kezd, és működési elve, hogy egy megadott értéknek megfelelő sugarú körben vizsgálja a vonal többi pontját. Az épp aktuális ponttól kezdve sorban megy végig a vonalat alkotó pontokon, és amelyik a körön belül van, azt eltávolítja a vonalból. Addig vizsgál egy pontot, míg meg nem találja az első olyan soron következő pontot, mely már kívül esik a körön. Ekkor az a pont lesz az úgy tesztpont, azzal folytatja tovább a vonal vizsgálatát egészen addig, amíg el nem éri az utolsó pontot. Ez az algoritmus számos összetettebb eljárásnak szolgál alapul



3.3.1. ábra. Sugaras távolság eljárás

3.4. Lang-algoritmus

A Lang-algoritmus meglehetősen hasonló Douglas–Peucker eljárásához, hiszen ez az operátor is egy egyenesre merőleges irányú határértékkel dolgozik. Ebben az esetben azonban meg kell adnunk még egy értéket, amely érték meghatározza, hogy az aktuális kulcsponttól számolt hányadik pontig vizsgálja az eljárás a vonalat. Első lépésként veszi a vonal első pontját, mint kulcspontot, majd az előre megadott értéknek megfelelő pontig egyenest húz a kulcspont és eme pont között. Ezután ellenőrzi, hogy van-e olyan pont, ami a határértéken kívül esik. Amennyiben talál ilyet, csökkenti a vizsgált pontok terjedelmét egyel, egészen addig, amíg már nem talál a határértéken túli pontot. Ha ezt elérte, akkor minden pontot, ami a határértéken belülre esik töröl, majd beállítja az utolsó vizsgálat végpontját, mint új kulcspontot, és újra vizsgálni kezd, a definiált értéknek megfelelő terjedelemben. Az algoritmusnak akkor van vége, ha már nem talál a határértéken kívül eső pontot. (Slocum, 2004)

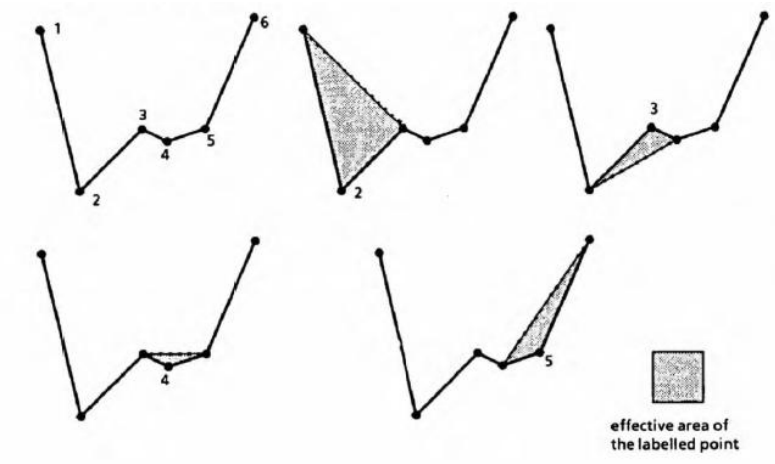


3.4.1. ábra. Lang-algoritmus

3.5. Visvalingam–Whyatt-algoritmus

A Visvalingam–Whyatt-algoritmus egy minden ponthoz hozzárendelt, úgynevezett hatékony terület alapján értékeli és távolítja el, vagy tartja meg a vonal pontjait. Minden pont hatékony területe a vele szomszédos pontokkal együtt alkotott háromszög területe. Ebből adódik, hogy a kezdő- és végpont nem rendelkezik ilyen értékkel, ezek semmiképp sem törlődnek; sőt, ha nem adunk meg toleranciát csak ez a két pont marad végül az egyszerűsítés után. Kezdetben kiszámol minden közbülső ponthoz tartozó hatékony területet, és a nulla

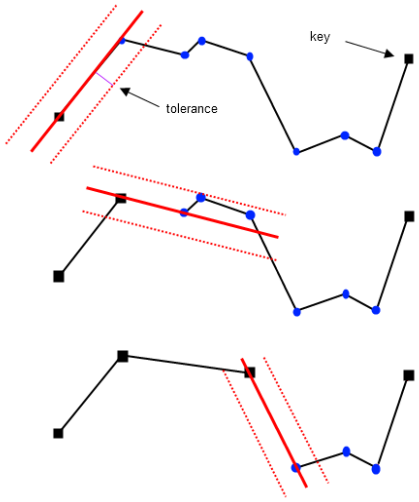
területűeket (egyenest szakaszt alkotó pontok, melyek nem a szakasz végpontjai) eltávolítja. Ezután megkeresi a legkisebb területhez tartozó pontot és törli a vonalból, majd újraszámolja a háromszögek területeit és újból törli a legkisebbet egészen addig amíg a legkisebb hatékony terület nagyobb, mint a meghatározott tolerancia. (Visvalingam, 1993)



3.5.1. ábra. Visvalingam–Whyatt-algoritmus

3.6. Reumann–Witkam-algoritmus

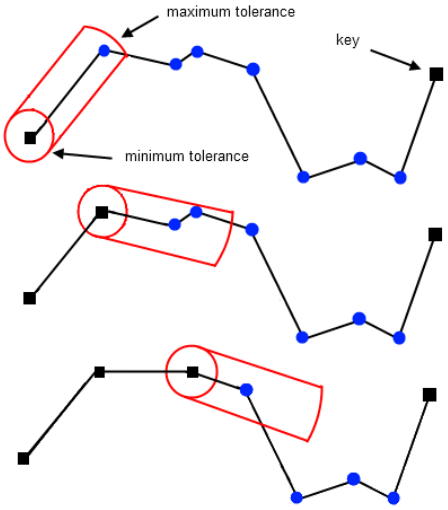
A Reumann–Witkam-algoritmus kicsivel kevésbé összetett, mint a fent leírt műveletek. Működésének alapja, hogy két szomszédos pont közti egyenestől való távolságot figyeli. Az eljárás a vonal első két pontjával kezdődik, meghosszabbítván a két pont közötti egyenes szakaszt. Ezután megfigyeli, hogy a második pont utáni pont, illetve pontok a toleráncián belül helyezkednek-e el. Ha nem, akkor a második pont megtartásra kerül (a vonal első pontja mindig megmarad). Ellenkező esetben sorban törlődnek a pontok, melyek nem haladják meg a toleranciát, egészen addig amíg, találunk olyat, mely a határértéken kívül esik. Ilyenkor egy új pontot hozunk létre ami a meghosszabbított egyenesünk, valamint az első határértéken kívüli ponton, és az előtte lévő ponton átmenő egyenes metszéspontja. A következő vizsgálat két alanya a talált toleráncián kívüli és az előtte lévő, illetve létrejött pont.



3.6.1. ábra. Reumann–Witkam-algoritmus

3.7. Opheim-algoritmus

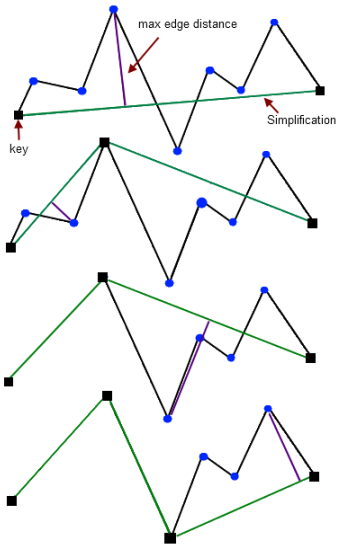
Az Opheim-algoritmus hasonlóan az előbb tárgyalt Reumann–Witkam-algoritmushoz szintén figyeli a két pont közti egyenestől mért távolságot, de ezt kiegészíti még két keresési paraméterrel, egy minimum és egy maximum távolsággal a figyelt ponttól. Az eljárás először is töröl minden pontot, ami a vizsgált ponttól a minimális határértéken belül van. Ezután megkeresi a legutolsó pontot, mely az előbb említett határ, a Reumann–Witkam-algoritmushoz hasonlóan megalkotott párhuzamosok, és a meghatározott maximális tolerancia területén belülre esik és töröli az összes közbülső pontot. Végül beállítja az előbb meghagyott utolsó pontot, mint az új vizsgálati pontot. (Slocum, 2004)



3.7.1. ábra. Opheim-algoritmus

3.8. Ramer–Douglas–Peucker-algoritmus

Talán a legtöbbet használt operátor a Ramer–Douglas–Peucker, vagy egyszerűbb nevén a Douglas–Peucker-algoritmus. Ez egy globális algoritmus, melynek lényege, hogy egyeneseket helyettesít be a vonal megadott részletei helyére és hasonlóan a merőleges távolság eljáráshoz egy előre meghatározott tolerancia alapján vizsgálja az egyes pontok távolságát ezektől az egyenesektől. Először veszi a vonal első pontját ez egy úgynevezett kulcspont lesz, olyan pont, mely fontos a vonal geometriáját illetően. Ezután ezt a kulcspontot a vonal legutolsó pontjával egy egyenessel összeköti. Harmadik lépésként megvizsgálja az összes közbülső pontot, és ha mindegyik a megadott határértéken belül van, akkor törli azokat megtartva csak a kulcs- és a végpontot. Ha talál olyan pontot vagy pontokat, melyek kívül esnek ezen a határértéken, akkor megkeresi az egyenestől legtávolabbi pontot(az adott pontból az egyenesre állított merőleges vonal hosszát figyeli), és felveszi kulcspontnak. Következő lépésként ezt az új kulcspontot is figyelembe véve, összeköti a kulcspontokat a szomszédos másik kulcs- és végpontokkal és kezdi újra a harmadik lépéstől. Az algoritmusnak akkor van vége, ha már nem talál több pontot a határértéken kívül. (Douglas és Peucker, 1973)



3.8.1. ábra. Ramer–Douglas–Peucker-algoritmus

3.9. Wang-algoritmus

A Wang-algoritmus felosztja a vonalat kanyarulatokra, melyeket különböző tulajdonságaik alapján vizsgál, és egy bizonyos referencia félkörrel hasonlítja össze őket, melynek átmérőjét

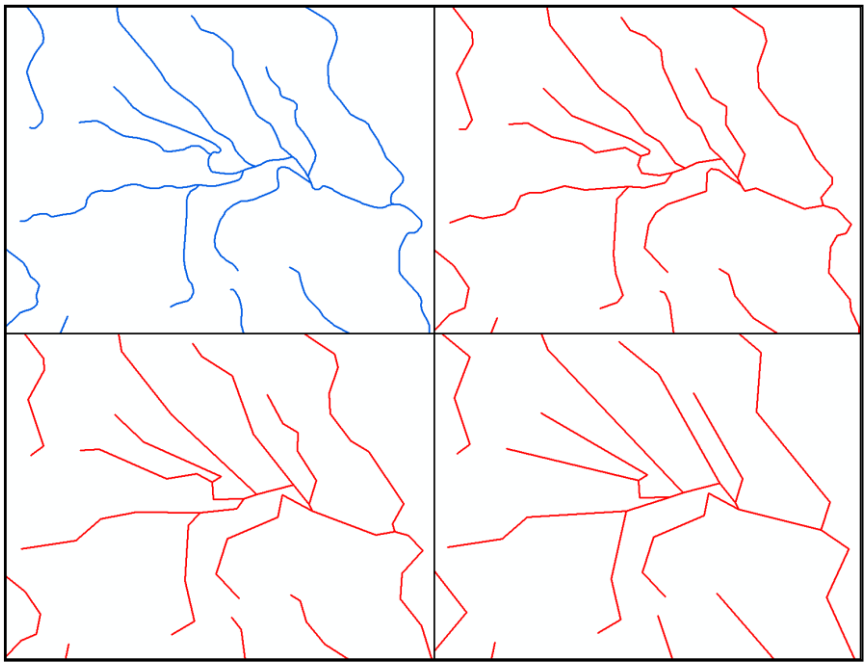
a felhasználó adja meg, mint határérték. Miután felismerte az összes kanyarulatot, a legkisebbel kezdve egymás után eltávolítja azokat, és az alapvonalukkal (a kezdő és a végpontot összekötő vonal) helyettesíti azokat. A legkisebb kanyarulattal kezdi és akkor fejeződik be az algoritmus, ha már nincs olyan kanyarulat, mely a referencia területnél kisebb területű. Az algoritmus működése alapján a globális eljárások közé tartozik. (Wang, 1998)

3.10. Vonalas elemek egyszerűsítése ArcMap programban

Az ArcMap az Esri által fejlesztett térinformatikai programcsoport az ArcGIS része. Kiválóan alkalmas térinformatikai jellegű adatbázisok készítésére, szerkesztésére és megjelenítésére. Elsőként létrehoztam egy saját geoadatbázist „Szakdolgozat" néven, amibe beimportáltam a vizsgálni kívánt „vizek" réteget, mely a Zagyva vízgyűjtő területének adatait ábrázolja 1:500000 méretarányban. Az adattábla meglehetősen részletes, a legkisebb mellékfolyók is ábrázolva vannak és a domborzat jellege miatt sok a kanyarulat és az összefolyás. Ezután megadtam a kívánt vetületet, ami a HD-72, (Hungarian Datum 1972.) vagyis az Egységes Országos Vetület (EOV), és nekiálltam a generalizálásnak. Egyszerűsítésnél első lépésként megnyitjuk az ArcToolbox-ot amit a Geoprocessing → ArcToolbox menüből érhetünk el. Itt kiválasztjuk a „Cartography Tools"-t azon belül is a Generalization opciót, végül pedig a Simplify Line lehetőséget. Itt az „Imput Features"-nél megadjuk, hogy melyik réteget szeretnénk generalizálni, az „Output Feature Class"-nél, hogy hova szeretnénk létrehozni az új réteget és milyen néven, majd kiválasztjuk a használni kívánt algoritmust és megadjuk a tolerancia értékét. A 10.2.2.-es verzióban két féle algoritmust találunk a vonalak egyszerűsítésére. Az első lehetőség a „Point Remove", mely valójában a Douglas–Peucker-algoritmus, a másik pedig egy úgynevezett „Bend Simplify", ami pedig a Wang-algoritmus. Lehetőségünk van még arra, hogy az egyszerűsítés után a program megvizsgálja a topológiát és lehetőség szerint kijavítsa a felmerülő hibákat. Ha használjuk ezt az opciót, akkor az eljárás végeztével megkeresi a topológiai hibákat (egymást keresztező vagy nem folytonos vonalakat), és az általunk beállított tolerancia felével újból egyszerűsíti az érintett vonalrészleteket, egészen addig, míg nem észlel több hibát. Gyorsabb eredményt kapunk, ha nem vizsgáltatjuk meg a topológiát, azonban érdemes élni a lehetőséggel, ha nem vagyunk biztosak az adatok topológiájának pontosságában.

Először „Point Remove" lehetőséget próbáltam. Az egyszerűsítésnél kihasználtam a topológia vizsgálatát és a hibák javítását. Ennek oka, hogy ellenkező esetben az eredmény

csak 100 és 500 méteres toleranciák közötti generalizálásnál volt megfelelő topológiailag. Ennél kisebb vagy nagyobb értékeknél a rövidebb vízfolyások azon pontjai, ahol a becsatlakoztak a folyóba, vagy nagyobb vízfolyásokba megmaradtak, míg a hosszabb vonalakban ezek a pontok esetlegesen törlésre kerültek, ezért a vonalak vagy keresztezték egymást, vagy nem is találkoztak. Mivel a generalizálni kívánt rétegem részletes volt, így emellett az algoritmus mellett a végeredmény csak kis határértékeknél lett megfelelően esztétikus. Az eljárás a merőleges távolságokat figyeli, és ezek alapján egyszerűsít, ezért nagyobb (körülbelül 200-300 méteres) toleranciák felett a vonalak nagy részét egyenesek vették át, csak a hosszabb, kevésbé kanyargós szakaszokon volt felismerhető a vízfolyás eredeti futása. Ennél kisebb határértékeknél pedig nem annyira szembetűnő a változás. További probléma még, hogy az eljárás nem törli azokat a vonalakat, melyek a nagyobb határértékek mellett már túl rövidek lennének, ezért egy kisebb méretarányú térképen már nagyon zsúfolt lenne a terület. Ilyenkor kézi utómunka szükséges. Összehasonlítva a hagyományos, kézzel generalizált térképekkel, az eredmény jóval gyengébb minőségű, viszont érdemes használni, ha viszonylag egyenes vonalakat szeretnénk generalizálni, vagyis önmagában a vízrajz egyszerűsítésére nem megfelelő.



3.10.1. ábra. „Point Remove” algoritmus különböző tolerancia értékekkel eredeti (bal felül), 100 m (jobb felül), 200 m (bal alul), 500 m (jobb alul)

A másik lehetőség a „Bend Simplify” már szebb eredményeket hozott. Mivel ez az algoritmus egy kanyarulat eliminálása után is ívesen hagyja a vonalat, ezért a végeredmény jócskán kifinomultabb. A generalizált vonal jobban követi az eredeti vonal futását, szebben megtartva annak jellegzetességeit még magasabb toleranciaértékeknél is. A határértékek megválasztásánál figyelembe kell venni, hogy a két eljárás alapja nagyon különböző, így a „Bend Simplify” használatánál arányaiban nagyobb (ennél az adatsornál 1000-3000 méter) toleranciát érdemes használni. Az egyetlen szembetűnő probléma itt is a rövidebb mellékfolyóknál, illetve a szintén rövid szakaszoknál lép fel, melyek két egybetorkollás között vannak, de ez is igazából csak erősebb generalizálásnál feltűnő. Ez a megoldás inkább hasonlít a kézzel készült térképeknél alkalmazott generalizálásra, hiszen az egyszerűsített vonal nagy részben ívelt, és jóval inkább igazodik a vízfolyam eredeti futásához.

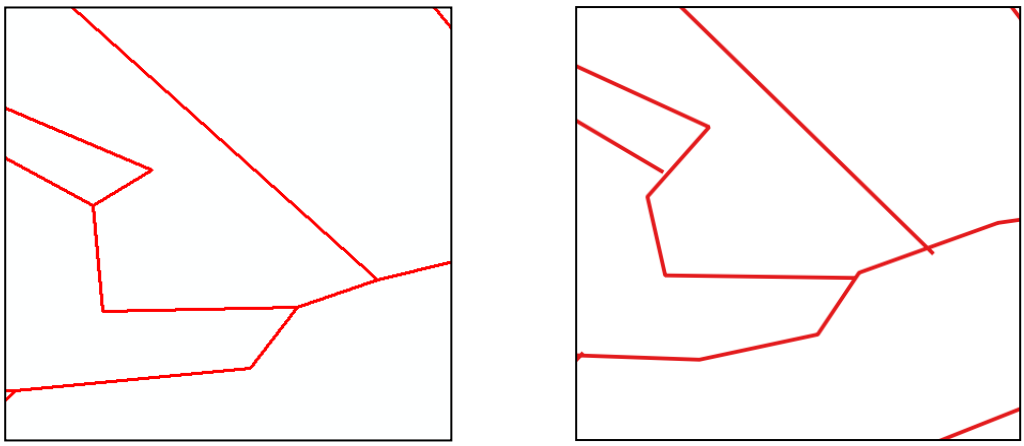
3. 11. Vonalas elemek egyszerűsítése QGIS programban

A QGIS (régebben Quantum GIS) egy nyílt forráskódú térinformatikai szoftver, mely 2002 februárja óta folyamatos fejlesztés alatt áll. Az alap programon kívül a felhasználó számos modullal bővítheti az alkalmazást, melyek között a hivatalos modulok mellett megtalálhatóak ugyanúgy a felhasználók által készítettek is. Elérhető több operációs rendszer alatt, mint például Windows, Linux, Mac OS, valamint fejlesztés alatt áll az Androidos verzió is. Egyszerűsítéshez a „Generalizer” modult használtam, melyet a Modulok menüpont Modul kezelés és telepítés lehetőségnél lehet letölteni. Első lépésként betöltöttem az előzőleg is használt „vizek” rétegemet a bal oldalon található böngészőből, majd itt is beállítottam az EOV vetületet, melyet „HD72 / EOV” néven találtam meg; és nekiálltam a generalizálásnak.

A „Generalizer” egy meglehetősen sok algoritmust tartalmazó generalizálási modul. Lehetőség van benne kisebb elemek eltávolítására, valamint többfajta egyszerűsítési és simítási algoritmus használatára. A modul csak akkor használható, ha már be van töltve egy vonalas elemeket tartalmazó réteg. Miután megnyitottuk a modult az első dolgunk, hogy megadjuk neki, mely réteget szeretnénk generalizálni, majd megadjuk a használni kívánt algoritmust. Ha kiválasztottuk az operátort megjelennek az adott eljárás beállításai, itt tudjuk megadni a toleranciát és egyéb konfigurációkat. Végül lehetőségünk van rá, hogy a végeredményt elmentse egy külön shape-fájlba. A modul lehetőséget ad egy úgynevezett „Batch mode” beállítás alkalmazására is, mellyel alkalmunk nyílik, hogy a kiválasztott réteget, vagy rétegeket automatikusan generalizálja az általunk megadott algoritmusokkal és

beállításokkal. Ez igen hasznos lehet, ha egy vagy több réteget szeretnénk egymás után különböző algoritmusokkal generalizálni, vagy egy algoritmussal, de eltérő határértékekkel.

Először a Douglas–Peucker-algoritmust próbáltam, mely hasonló eredményekkel zárult, mint az ArcMap programban, egy nagyon lényeges eltérést leszámítva. Az előző programmal ellentétben, a „Generalizer”-ben nincs lehetőség a topológia vizsgálatára, ennek eredményeként minden egyes vízfolyamot jelképező vonalat egymagában, a többitől függetlenül generalizált. Míg az ArcMap-ben topológiai vizsgálat nélkül is folyamatos, egymást a közös eredeti pontokban érintő vízhálózatot kaptam, ha 100 és 500 méter között generalizáltam, addig itt ez semmilyen határértékre nem mondható el (3.11.1. ábra). Mindezt összevetve az algoritmus szögletes generalizálásával nagyon alacsony kartográfiai minőségű eredményt kapunk, ha részletes, sűrű vonalas réteget szeretnénk generalizálni. A modulnak ezt az algoritmusát csak olyan rétegeken ajánlanám használni, melyeken viszonylag ritkásan elhelyezkedő, egymást nem érintő, aránylag egyenes vonalas elemek találhatók.



3.11.1. ábra. Helyes topológia az ArcMap (bal), és hibás a QGIS (jobb) programokban

A Lang-algoritmus hasonló elven működik, mint Douglas–Peucker algoritmusa, azzal a kivétellel, hogy ez az eljárás nem feltétlenül egyből az egész vonalat egyszerűsíti, hanem egyszerre csak egy előre megadott értéknek megfelelő számú ponton bizonyos előretekintési toleráncián belül vizsgálja azt. Egy ilyen részletes rétegen, mint az általam vizsgált, ennek a paraméternek a módosításával igen különböző végeredményeket kaphatunk. Minél nagyobbra vesszük az előretekintési toleranciát annál jobban hasonlít a generalizált vonal a Douglas–Peucker-algoritmussal egyszerűsítetthez. Ennek oka, hogy egy vízfolyamot alkotó vonal

véges számú pontokból áll, ha a tolerancia értéka megegyezik egy adott vonalat alkotó pontok számával, akkor gyakorlatilag a Douglas–Peucker-algoritmust használjuk annak egyszerűsítésére. A tolerancia értékét folyamatosan csökkentve (ugyanolyan merőleges határérték mellett) egyre kevésbé töredezett vonalat kapunk végeredményként. Az optimális eredmény elérése érdekében figyelembe kell vennünk a generalizálni kívánt vonal sajátosságait, és azok alapján beállítani a két értéket. Mivel valamelyik érték megváltoztatásával hatunk a másik érték eredményeire, ezért ezek megfelelő beállítása nem egyszerű feladat.

A harmadik egyszerűsítési operátor a Reumann–Witkam-algoritmus. Ez is hasonló a Douglas–Peucker-algoritmushoz, a különbség a kettő között, hogy ez az eljárás két szomszédos pontra fektetett egyenestől való távolságot figyeli. Ennek következményeként a rövid egymást nagy ívben követő vonalelemek generalizálásánál jól követi az eredeti vonal futását, míg a hosszabb egyenesebb szakaszokon jóval több pontot hagy el, ezért azok sokkal szögletesebbekké válnak. Ez a módszer jóval érzékenyebb a tolerancia változtatására. Azoknál az értékeknél, melyek a többi, hasonló elven működő algoritmusnál még közepesnek számítottak (200-500 méter) itt már jóval szembetűnőbb változásokat lehet észrevenni. Ez az eljárás az eddig tárgyaltakhoz képest sokkalta hajlamosabb elhagyni a rövidebb, ívesebb kanyarulatokat.

4. Vonalas elemek simítása

Amikor a térkép készítője kézzel generalizál vonalat, akkor ezek a vonalak általában szépen megrajzolt, folyamatosan futó vonalak. Ezzel ellentétben a digitális adattáblák vonalai sokszor meglehetősen szögletesek, ezáltal nem túl esztétikusak. A simítási algoritmusok feladata, hogy ezeknek a töredezett vonalaknak növelje az esztétikai minőségét azáltal, hogy egy folyamatosabb, a szemnek sokkalta kellemesebb futású vonalat hoz létre az általunk megadott vonalból. Az egyszerűsítési algoritmusokhoz hasonlóan a simítási eljárásokat is be lehet sorolni lokálistól globális csoportokba, azonban a simítási operátorokat rendszerezhetjük működésük alapján is. Ebben a kategorizálásban az első a pont átlagolás csoport. Az ebbe a csoportba tartozó algoritmusok egy átlagolt értékkel dolgoznak, melyet az adott pont, és a pont megadott számú szomszédjainak x és y koordinátájából számol ki. Az így létrejött pontot aztán algoritmustól függően súlyozottan eltolja a vizsgált ponthoz viszonyítva. Ezek az

algoritmusok mindig lokálisak, vagy nem-korlátozottan lokálisak. Ilyen algoritmus McMaster távolsággal súlyozott átlagoló algoritmusa Következő kategória a matematikai közelítések csoportja. Ezek az algoritmusok egy vagy több matematikai függvénnyel írják le a vonalat az általunk megadott toleranciák mellett. Lehetnek lokálisak, nem-korlátozottan lokálisak vagy globálisak. Erre példa a Bézier-görbe. Az utolsó csoportba tartozó eljárások a tolerancia algoritmusok. Ezek bizonyos geometriai kapcsolatokat figyelnek egy megadott pont körüli tolerancián belül. A matematikai közelítésekhez hasonlóan szintén lehetnek lokálisak, nem-korlátozottan lokálisak vagy globálisak. Példa erre a csoportra Boyle „előretekintő” algoritmusa.

4.1. McMaster csúsztatott átlag algoritmusa

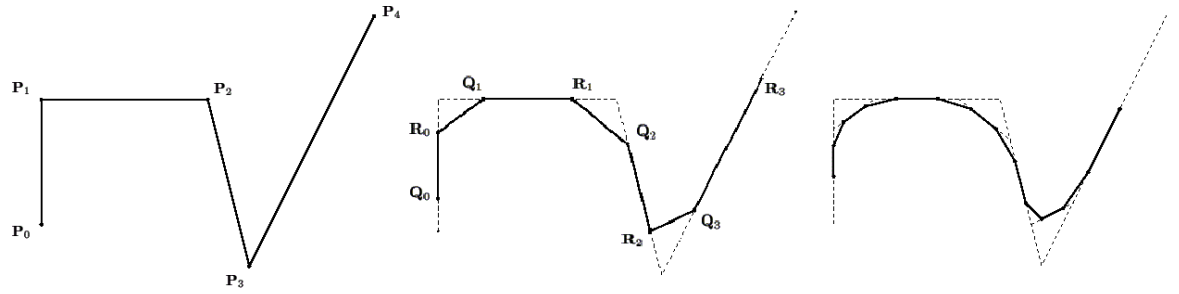
A McMaster csúsztatott átlag algoritmusa, ahogy a nevéből sejteni lehet egy pont átlagolási módszer. Az eljárás kezdetekor meg kell adnunk egy számot. Ez a szám határozza meg, hogy egy adott ponttól számított hányadik szomszédig átlagoljon az eljárás (magát a pontot is beleszámítva). Ha ez az érték például három, akkor az első vizsgált pont a harmadik pont lesz a vonalban. Az eljárás kiszámolja a pont és két-két szomszédja koordinátáinak átlagát és felveszi ezt a pontot. Ezután egy másik előre meghatározott számnak megfelelően az új pontot a vizsgált ponthoz képest eltolja. Ennek következtében minden pont a szomszédjai felé fog eltolódni.

4.2. McMaster távolsággal súlyozott átlag algoritmusa

Ez az algoritmus lényegében azonos ez előbb tárgyalttal. Egyetlen különbség a kettő között, hogy ennél az eljárásnál az átlagolt pont helyzetét súlyozottan befolyásolja, hogy milyen messze helyezkedik el az átlagolásban részvett pontoktól. A súly értéke a vizsgálatban részvett pontoknak az átlagolt ponttól való távolságának reciproka, így minél közelebb van egy pont a simított ponthoz annál nagyobb „erővel húzza” azt. Mindkét algoritmus esetében az első és utolsó n pont (ahol n az előre megadott érték) nem kerül feldolgozásra, mivel nincs elég szomszédos pont. Ezek a pontok megőrzik az eredeti helyzetüket.

4.3. Chaiken-algoritmus

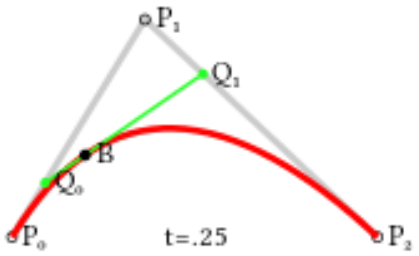
A Chaiken-algoritmus egy geometriai eljárás, melynek célja, hogy lekerekítse a kívánt vonalat. Ezt olyan módon éri el, hogy minden csúcsból levág egy darabot, majd ezt addig ismétli, amíg a vonal eléri a kívánt finomságot. Az algoritmus veszi a vonal minden pontja közti egyeneseket, majd ezeket felosztja olyan arányban, hogy az egyenes mindkét végétől ugyanakkora távolsátra legyenek az új pontok. Ezeket az új pontokat összeköti, majd az ezeken kívül eső részt (mely tartalmazza az eredeti pontot) törli, ezáltal simítva a vonalat. Ezt az eljárást kívánt mennyiségben megismételhetjük egymás után a megfelelő eredmény eléréséért, azonban azt észben kell tartani, hogy minden egyes iteráció után a pontok száma közel duplája lesz az előzőhöz képest, ezért a simítás ideje többszöri ismétlés után jelentősen megnő.



4.3.1. ábra. Chaiken-algoritmus

4.4. Bézier-görbe

A Bézier-görbék igen hasznos és széles körben használt grafikai megoldások képszerkesztő programokban. Mindemellett a Bézier-görbék leírását adó algoritmus nagyszerűen használható törött vonalak simítására is. Ezen görbék összetettségét a fokszámuk adja meg. Egy másodfokú Bézier-görbe létrehozásához legalább három pont kell. Képzeljük el, hogy a két pont közötti két egyenesen azonos idő alatt megy végig két pont. Ha ezt a két pontot összekötjük egy egyenest kapunk, mely a vizsgálat első időpillanatában megegyezik az első két pont közötti egyenessel az utolsó pillanatban pedig a második két pontot összekötő egyenessel. Ha ezen az egyenesen is végigmegy egy pont ugyanannyi idő alatt, miközben mozog az első két pont által megadott pályán, akkor az ez a pont által leírt görbe egy másodfokú Bézier-görbe lesz. Harmadfokú görbéhez már négy pontra, negyedfokúnál már öt pontra van szükségünk és így tovább.



4.4.1. ábra. Másodfokú Bézier-görbe

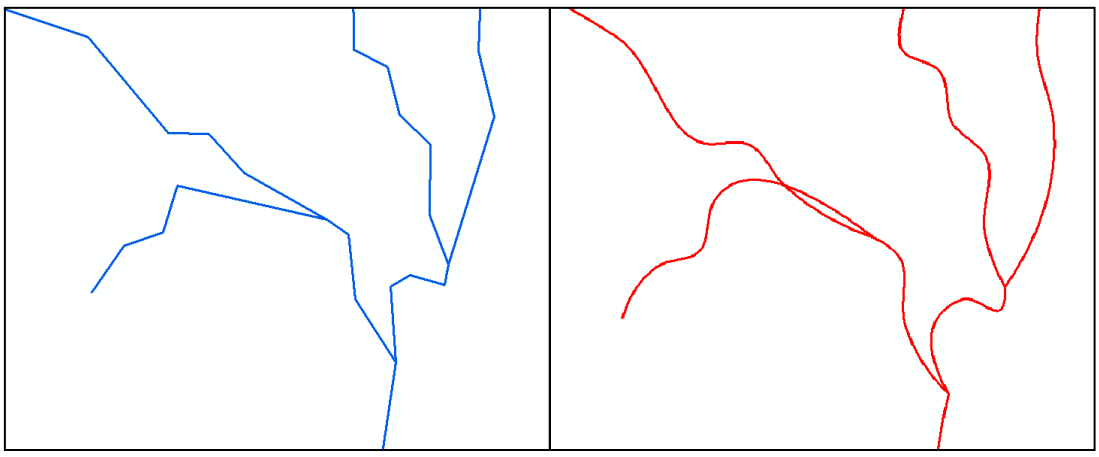
4.5. Vonalas elemek simítása az ArcMap programban

A simítás vizsgálatához az ArcMap-ben egy olyan egyszerűsített réteget használtam, amit a „Point Remove” használata során kaptam 200 méteres tolerancia mellett. Ennek oka, hogy az eredeti réteg igen aprólékos, sok ponttal, míg az egyszerűsített egy jóval töredezettebb, kevésbé esztétikus réteg. Generalizálás során egyébként is az egyszerűsített réteget szokás simítani hasonló okok miatt. Elsőként betöltöttem az előzőleg elkészített réteget, majd a korábbihoz hasonlóan a Geoprocessing → ArcToolbox → Cartography tools-t választottam ezen belül azonban most a „Smooth Line” lehetőséget. Itt is először az „Input Features”-nél megadtam a simítani kívánt réteget, majd az „Output Feature Class”-nál, hogy hova és milyen néven készítse el a simított réteget. Ezután két fajta algoritmus közül választhatunk. Az első egy úgynevezett „PAEK” (Polynomial Approximation with Exponential Kernel, vagy magyarul Polinominális Közelítés Exponenciális Kernellel), a másik pedig egy Bézier-interpoláció. Ezután a toleranciát tudjuk megadni (ez a lehetőség csak a PAEK algoritmus választása esetén elérhető), majd eldönthetjük, hogy szeretnénk-e megtartani a rétegen a különböző zárt vonalak végpontjait. Az egyszerűsítéshez hasonlóan itt is módunkban áll megvizsgálni a topológiát. Az eszköz leírásában az áll, hogy nem javítja ki a topológiát, csak megjelöli azokat, én viszont azt tapasztaltam, hogy ha nincs bekapcsolva, akkor a vonalak végpontja a helyükön maradnak ezzel esetlegesen topológiai hibákat létrehozva, ha viszont a megjelölés opciót választom, akkor a vonalak közös pontjait megtartja, ezáltal biztosítva a vízrajz folytonosságát.

A „PAEK” egy Chaiken-algoritmushoz hasonló eljárás, azonban a „PAEK” nem csak levág a csúcsokból, ha két azonos irányú törés van a vonalban, akkor a kettő közötti egyenesen kilép a vonal külső oldalára is. Ez azonban alig észrevehető, így végül nagyon hasonlít a Chaiken-algoritmussal elérhető eredményekhez. A topológia javítása lehetőség bekapcsolásával elérjük, hogy a simított vonalak átmenjenek az eredeti vonalak azon pontjain

ahol találkoztak, az eredeti végpontokon azonban nem fognak áthaladni. A tolerancia megadásánál nagyobb fokú simításhoz észrevehetően nagyobb értékeket kell megadni. Figyelembe kell venni a rétegünk sajátosságait. Minél nagyobbak a kanyarulatok, mind elfordulásban, mind hosszúságban annál látványosabb lesz egy nagyobb toleranciájú simítás. Rövid, de éles kanyarok esetén egy bizonyos tolerancia felett már csak nagyon kevéssé változnak a különböző értékkel készített simítások. Az alapanyagtól függetlenül minél kisebb toleranciát adunk meg, annál jobban fog hasonlítani a simított vonal az eredetihez. Nagyobb értékeknél a simított görbék már jóval messzebb lesznek a csúcsoktól.

Bézier-interpoláció esetén nincs lehetőségünk határérték megadására, a végeredmény csakis a vonalas rétegünk jellegétől függ. Az egyetlen beállítás, amit eszközölhetünk az a topológia vizsgálata, mivel azonban az interpoláció figyelembe veszi a közös pontokat, és azokat minden esetben, ezért az általam vizsgált rétegen ez az opció be- vagy kikapcsolása semmiféle különbséggel nem járt. Mivel az egyes görbék végpontjai a közös pontok, ezért az olyan szakaszok ahol közel vannak egymáshoz az összefolyások vagy elágazások közel, vagy teljesen egyenesek is lehetnek. Minél hosszabban fut önállóan egy vonal annál folyamatosabb lesz a simítás után. Az egyetlen észrevehető probléma ezzel az operátorral, hogy az olyan mellékfolyók találkozásánál, melyek más irányból jöttek, az összefolyás előtt viszont közel párhuzamosak voltak, a Bézier-görbék keresztezhetik egymást (4.5.1. ábra). Ettől eltekintve ennek az eljárásnak az eredménye egy igen sima és esztétikus simítás lett.



4.5.1. ábra. Bézier-görbék kereszteződése simítás után

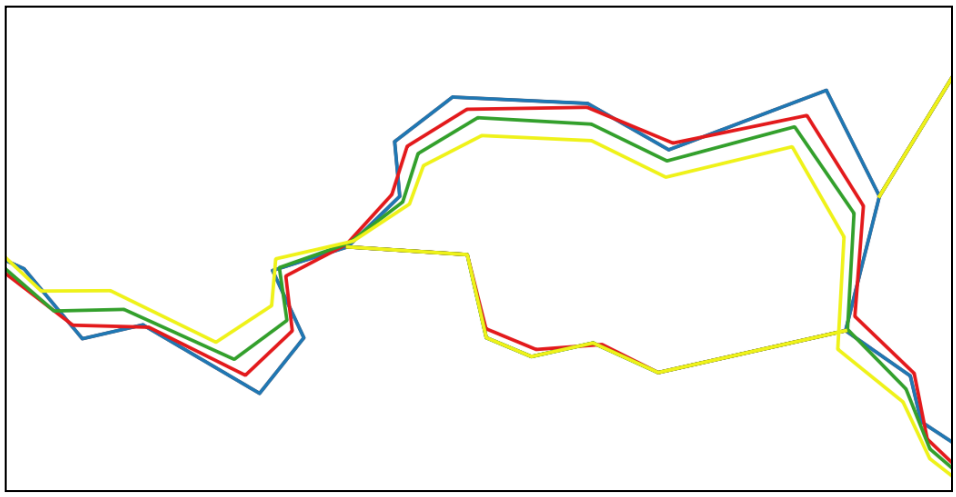
4.6. Vonalas elemek simítása a QGIS programban

QGIS-ben szintén a „Generalizer” modult használtam a simításhoz. Az előzőhöz hasonlóan itt is az ArcMap-ben már előre egyszerűsített réteget használtam. Azért nem a QGIS-ben generalizált réteget vettem alapul, mert ebben a programban végzett egyik egyszerűsítés után sem volt megfelelő a topológia. Itt is betöltöttem a böngészőből az előzőleg készített shape fájlt, mely tartalmazza a kívánt réteget, majd megnyitottam a „Generalizer”-t. A modul ezeknél az algoritmusoknál sem veszi figyelembe a topológiát, ezért a simítások eredménye minden esetben sok különálló, vagy egymást keresztező vonal.

Miután kiválasztottam a réteget az első algoritmus, amit vizsgáltam a Chaiken-algoritmus volt. Ennél az eljárásnál két értéket kell megadnunk: az iterációk számát és az algoritmus „súlyát”. A súlynak megadott szám határozza meg, hogy a vonal egyes egyenes darabjairól milyen arányban kezdje el levágni a sarkokat. Abban az esetben, ha ezt a számot egynek adjuk meg, akkor minden egyenest megfelez, ha kettőnek, akkor harmadol és így tovább; a két szélső részét az egyeneseknek elhagyja és az így keletkezett végpontokat pedig összeköti az első iteráció során. Ha a súly értékét egynek vesszük, akkor függetlenül az ismétlés fokától ugyan olyan vonalakat kapunk. Ezek a vonalak az egyes egyenesek középpontjait összekötő új vonalak. Fontos még megjegyezni, hogy minél többször ismételjük meg az algoritmust annál simábbnak tűnik a generalizált vonalunk, de a negyedik-ötödik ismétlés után már nem igazán van szemmel látható különbség. A rétegemen a legnagyobb mértékű simítást négyes súllyal (vagyis az egyeneseknek mindig a szélső kétötöde került levágásra, és a középső háromötöd maradt meg), és hat fokszámú iterációval értem el.

Következő eljárás a McMaster csúsztatott átlag algoritmus volt. Ennél az operátornál két értéket áll módunkban megadni. Az első érték a csúsztatás mértéke, azaz, hogy az átlagolt pontot mennyire és milyen irányban tolja el a vizsgált ponthoz képest. Ez az érték az alapbeállításnál 0,5, azaz a végső simított pont az átlagolt és az eredeti pont között pont félúton van. Minél kisebb a csúsztatás mértékét megadó szám annál közelebb kerül az eredeti ponthoz, annál jobban követve annak futását. Egynél nagyobb szám megadása esetén már igen torz eredménnyel számolhatunk. A másik érték megadja, hogy egy adott pont mellett mennyi szomszédját vizsgálja az eljárás. A legkisebben megadható érték három, ami azt jelenti, hogy a vizsgált ponton kívül csak a két szomszédos pontot veszi még figyelembe az átlagolásnál. Ennek az értéknek a megadásánál oda kell figyelni a simítani kívánt vonal pontjainak helyzetére, hiszen nagy érték esetén a vonal egy nagyobb szegmensének átlaga

lesz egy pont új helyzete. Ez egy bonyolult futású vonal esetén meglehetősen torz vonalat eredményezhet. A módszer összességében megőrzi az eredeti pontok számát, így az adott rétegen lévő vonalaink még mindig tört vonalak lesznek, viszont az átlagolás eredményeképpen az egyes pontoknál lévő szögek nagyobbá válnak, ezáltal simítva a vonalat.



4.6.1. ábra. McMaster csúsztatott átlag eljárás különböző mennyiségű pontok vizsgálatával eredeti (kék), 5 pont (piros), 9 pont (zöld), 13 pont (sárga)

A McMaster távolsággal súlyozott átlag algoritmussal simított rétegem az eljárások hasonlóságából kifolyólag nagyon hasonlít a McMaster csúsztatott átlag algoritmussal készülthez. Az egyetlen különbség a kettő között, hogy a most használt operátor figyelembe veszi, az átlagolt pontnak a többi vizsgált ponttól való távolságát. Ezt azt eredményezi, hogy a kapott pontot nagyobb súllyal vonzzák a hozzá közelebb levő vizsgált pontok, ezért magasabb számú kontrollpont megadása esetén sem torzul annyira Mind az előző, mind ennél az operátornál érdemes megjegyezni, hogy ha egy adott vonal kevesebb pontból áll, vagy megegyezik a vizsgálatra kijelölt pontok számával, akkor mindkét algoritmus az eredeti vonalat adja vissza, mint simított.

5. Felületi elemek egyszerűsítése

A felületi elemek egyszerűsítése nem sokban különbözik a vonalas elemekétől. Itt a rétegen megjelenő poligonok (sokszögek) körvonalát alkotó önmagukba visszatérő vonalak egyszerűsítéséről van szó, így teljes mértékben alkalmasak az előzőleg használt algoritmusok. Nagy hangsúlyt kap azonban a megfelelő topológia elérésére a generalizálás előtt. Míg a vonalas elemeknél a topológiai vizsgálat feladata az egyes vonalak összefutásainak felmérése volt, addig a felületi elemeknél az egymás mellett fekvő poligonok találkozása áll a figyelem középpontjában. A topológia vizsgálata teszi lehetővé, hogy egy adott területet lefedő poligonok között ne keletkezzenek lyukak vagy átfedések. Ez azonban a vízrajz generalizálásánál nem meghatározó szempont, hiszen nyílt vízfelületeknél ez annyira nem jellemző. Kivételt képez ez alól a nagy méretarányban megjelenített szélesebb folyók, valamint széles folyó és tó találkozása, mivel ilyen esetekben a bő vízfolyásokat szokás kétvonalasan, azaz poligonként ábrázolni.

5.1. Felületi elemek egyszerűsítése az ArcMap programban

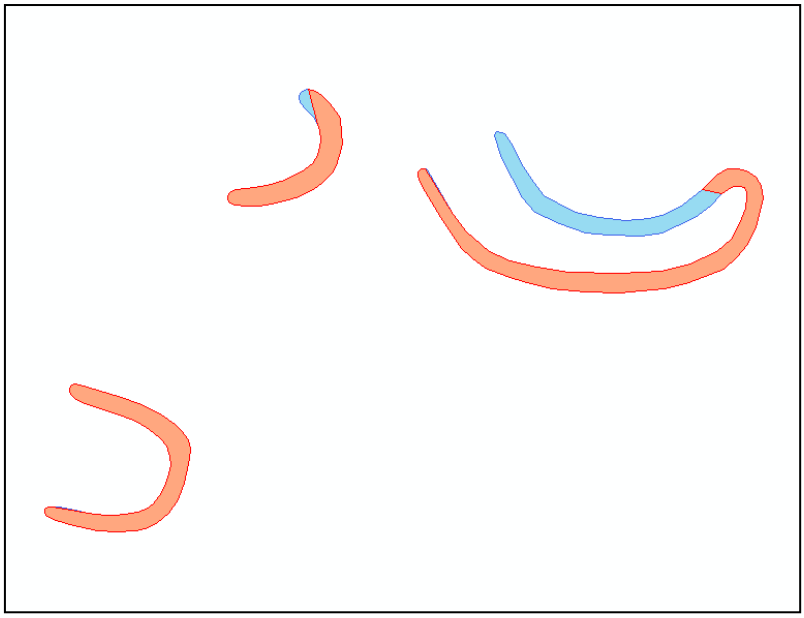
Poligonok egyszerűsítéséhez szintén az ArcToolbox → Cartography Tools → Generalization lehetőségen belül találunk eszközt, ez pedig a „Simplify Polygon”. Elsőként most is betöltöttem a réteget és megadtam a vetületet. Megnyitva a kelléket a simításnál használthoz hasonló panel ugrik fel. Alkalmunk van megadni az „Input Features”-t, az „Output Feature Class”-t, a használni kívánt algoritmust, annak toleranciáját, valamint lehetőség van még egy minimum terület megadására, és a topológia vizsgálatára, illetve javítására. Az eszköz ugyan azokat az eljárásokat használja, mint a „Simplify Line” lehetőség. A megadott minimum területre akkor van szükség, ha a generalizálás mértéke nagy, és nem szeretnénk megtartani az összes egyszerűsített poligont. Ekkor egy megadhatunk egy területi értéket és az egyszerűsítés után minden ennél kisebb területű poligon törlődik az elkészült rétegről. Egymás mellett lévő, közös határokkal rendelkező poligonok csoportjánál ez a határérték a csoport területére vonatkozik. Ha élünk ezzel a lehetőséggel és volt olyan felületi elem, amely törlésre került, akkor létrehoz még egy réteget, ami az eltüntetett poligonok helyét jelző pontokat tartalmazza. A topológiát tekintve három lehetőségünk van. Az elsőnél nem vizsgálja a topológiát, így gyorsabb lesz az egyszerűsítés, azonban ha bármi hiba van a kapott rétegben arról nem lesz tudomásunk. A második lehetőség, hogy megvizsgálja az

egyszerűsített réteget és jelzi, ha valahol hibát észlelt. Végül az utolsó alternatíva választásával kijavítja a talált hibákat. A tolerancia megadása ebben az esetben is a rétegünk jellemzőitől és a generalizálás mértékétől függ. Érdemes a különböző méretű vízfelületeket jelképező poligonokat külön osztályokba sorolni és ezeket eltérő rétegeken tárolni, így ezeket módunkban áll eltérő toleranciával egyszerűsíteni.

Az első eljárás a „Point Remove”. A Douglas–Peucker-algoritmust használó operátor eredményessége nagyban függ attól, hogy helyesen mértük-e fel az egyszerűsíteni kívánt réteg sajátosságait, és ennek megfelelő határértéket választottunk-e. A kiinduló rétegen lévő vízfelületekhez képesti kicsi tolerancia megadása kis mértékű egyszerűsítést eredményez, ebben az esetben a kapott poligonok körvonalai jól követik az eredeti rétegen található sokszögekét. Az határérték növelésével azonban nagyon hamar jelentős mértékben csökkenhet a generalizált réteg minősége. A közel kör alakú vízfelületeket egyre kisebb csúcsszámú sokszögek váltják fel; a tagoltabb partvonalú, vagy konkáv tavak esetében viszont már meglehetősen nagyok a torzulások. Az általában U-alakú morotváknál pedig már teljesen felismerhetetlen, saját körvonalát keresztező poligonokat kaphatunk. Az kereszteződést ugyan ki tudjuk kerülni azzal, hogy bekapcsoljuk a topológiai hibák megoldása opciót, ennek következménye azonban sok esetben ilyenkor sem lesz esztétikus. A „Point Remove” lehetőséget akkor érdemes használni, ha csak kisebb mértékben szeretnénk egyszerűsíteni. Nagyobb mértékű generalizálás alkalmával előfordulhat, hogy a kapott rétegünkön a poligonok erősen szögletesekké válnak és eltérnek a kiindulási felületektől, így simítás után sem lesz a kívánt minőségű az eredmény.

Második lehetőség itt is szintúgy a „Bend Simplify”. Ennek az eljárásnak az alkalmazása ebben az esetben is szebb eredményt hozott. Itt is, mint a vonalas elemek egyszerűsítésénél nagyobb értékű toleranciákat kell megadni az észrevehető változás elérése érdekében. Relatíve kisebb toleranciánál a kapott felületek határvonalai közel megegyeznek a kiindulási poligonokéval. Az érték növelésével a poligonok kerülete egyre simább lesz, a kis ki- és beugrások sorra eltűnnek. Egy közepes tolerancia mellett ez már azt jelentheti, hogy egy kevésbé konkáv felületi elemnél az egyszerűsítés után a konkávságot adó kanyarulat eltűnhet, ezzel növelve annak területét. Még nagyobb határérték megadása viszont már észrevehető problémákat okozhat. Egy bizonyos érték felett az inkább körhöz hasonlító, kisebb területű vízfelületeket egyszerűen törli az algoritmus, még akkor is ha a beállításoknál a minimum területnél nem adtunk meg értéket. Az általam vizsgált rétegen 2000 méteres tolerancia mellett például az egyik keskeny, jelentősen ívelt morotvának az egyszerűsítésénél

elhagyta annak egy részét (5.1.1. ábra). Ettől eltekintve, ha figyelembe vesszük a kiindulási réteg jellegzetességeit és ezeknek megfelelő toleranciát választunk, akkor az eredmény itt is, mint a vonalas elemeknél kiemelkedő minőségű lesz.



5.1.1. ábra. Egyszerűsítés utáni csonkított poligon

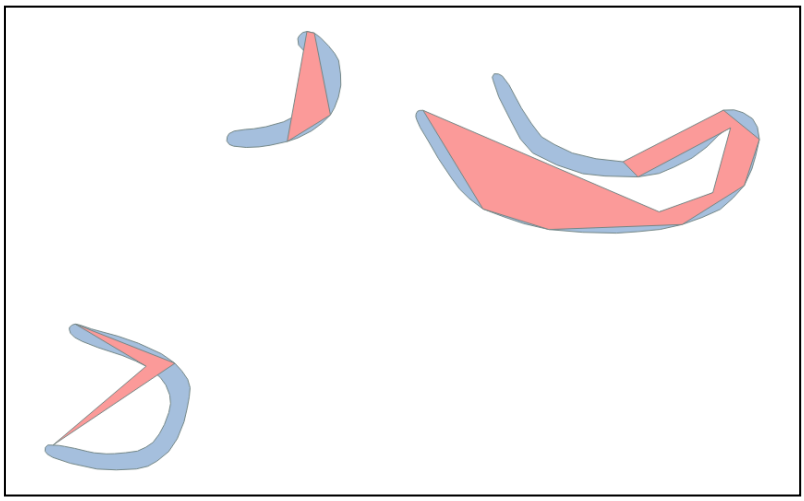
5.2. Felületi elemek egyszerűsítése a QGIS programban

A QGIS program is lehetőséget nyújt poligonok egyszerűsítésére. Az ehhez használatos modult a „Generalizer”-hez hasonlóan a „Modul kezelés és telepítés” menüpont alatt tölthetjük le. Az eszköz neve „SimpliPy” és megannyi lehetőséget ad a generalizálásunk testreszabására. Az „Input” résznél megadjuk, hogy melyik réteget szeretnénk egyszerűsíteni, majd beállíthatjuk, hogy a réteg minden elemét, vagy pedig az előzőleg kiválasztottakat (itt mindkét esetben megmutatja, hogy összesen mennyi objektum kerül feldolgozásra) generalizálja. Ennek ellenére ebben a programban is mind a két algoritmus használatánál külön rétegen kezeltem a nagyságrendileg eltérő méretű poligonokat. Az „Output”-nál azt adhatjuk meg, hogy az elkészült réteg látható legyen-e az algoritmus végeztével vagy maradjon rejtve. Következőnek a használni kívánt algoritmust jelölhetjük ki, erre két lehetőségünk van a Douglas–Peucker, valamint a Visvalingam eljárások. A következő „Parameters” beállításnál módunkban áll megadni az algoritmusokra vonatkozó toleranciákat

és az ezekhez tartozó egyéb beállításokat. Utoljára a „Constraints”-on belül van még alkalmunk további végső beállításokra. Ezen belül az „Expand/Contract”-nál beállíthatjuk, hogy az egyszerűsített poligonunk a kiindulási poligonon belül helyezkedjen-e el vagy fordítva. A „Repair Intersections” lehetőség bekapcsolásával kijavíthatjuk az egyszerűsítés utáni metszéseket. A „Prevent shape removal” beállítás kipipálásával megakadályozzuk, hogy a generalizálás során törlődjenek poligonok, és megadhatjuk, hogy legalább mennyi pont maradjon meg. Abban az esetben, ha egy sokszög törlésre kerülne, akkor az eljárás visszaállítja a legutolsó állapotát amikor még megjeleníthető volt. A „Constraints”-en belül az utolsó opció a „Use topology”, mely a topológiát ellenőrzi és lehetőség szerint ki is javítja azt a megadott precizitással.

Először a Douglas–Peucker-algoritmust próbáltam ki. Alapbeállításokkal egy az egyben ugyanazokat az eredményeket kaptam a különböző toleranciákkal történő egyszerűsítés után, mint az ArcMap „Point Remove” lehetőségével. A „SimpliPy” azonban sokkal több beállítást enged. Az egyik ilyen a paraméterek beállítás alatt található. Alapvetően a Douglas–Peucker-algoritmus vonalak esetén a két végpontot veszi kezdőpontnak, felületi elem esetén azonban zárt vonalakról beszélünk. Így itt a tolerancia értéke mellett lehetőségünk van azt is megadni, hogy a modul hogyan ossza fel az poligont vonalláncokra az egyszerűsítéshez. Háromféle beállítás létezik, az én tapasztalatom szerint azonos tolerancia mellett a három beállítás ugyanazt az eredményt produkálta.

A Visvalingam-algoritmus alkalmazásánál az első szembeötlő különbség az előző operátorhoz képest a toleranciák nagyságbeli különbsége. Mivel a Visvalingam-algoritmus területi határérték alapján dolgozik ezért jóval nagyobb értékeket kell megadni. Az eljárás a konvex vízfelületek esetében a Douglas–Peuckernél szebb eredményt hozott. A konkáv sokszögeknél azonban, mint például a morotvák rendkívül durva és pontatlan egyszerűsítést hajtott végre (5.2.1. ábra). Célszerű ezért a poligonok méretén kívül azok konvexitását is figyelembe venni generalizálás előtt; és vagy külön rétegre átmásolni azokat, vagy kihagyni a kijelölésből. Összehasonlítva az előző eljárással és figyelembe véve az algoritmusnak a konvexitásra való érzékenységét a megfelelő előkészületek után nem sokkal ugyan, de esztétikusabb eredményeket érhetünk el. A generalizált rétegünk tartalma viszont mind a két esetben igen erősen szögletes poligonok, ezért egyiket sem ajánlott simítás nélkül alkalmazni.



5.2.1. ábra. Visvalingam-algoritmus alkalmazása utáni torzulás

Megjegyezném még a modulhoz, hogy bizonytalansága sokszor rendkívül megnehezítette annak használatát. Többször előfordult, hogy valamely egyszerűsítés indításakor a program hibára hivatkozva leállt. Máskor elindult az eljárás, azonban annak futása közben fellépő hiba miatt nem futott le. Ebben az esetben a program nem zárt be, viszont a modul ezek után már semmire nem válaszolt, még annak újbóli megnyitása után sem jött rendbe. Ezt csak a program újraindításával tudtam helyrehozni. További probléma volt még, hogy a sok beállítás jelentősen lassította az algoritmus lefutását, sőt arra is volt példa, hogy a számítások meghaladták az én számítógépem kapacitását.

6. Felületi elemek simítása

Sokszögek simításánál szintén az a cél, hogy egy simább, kellemesebb körvonalú poligont kapjunk. Itt is, mint a polyline simításánál előzőleg egyszerűsített sokszögeket tartalmazó rétegeket használtam. Ezek a rétegek szintén az ArcMap-ben „Point Remove” algoritmussal készültek. A megfelelő eredmények elérése céljából ugyanazokat az algoritmusokat használjuk, mint vonalak simításánál. Egymással határos poligonok esetében ez egy igen könnyű feladat, ha figyelembe vesszük a topológiát. Különálló felületi elemek simításánál azonban felmerülhetnek problémák. Erre lehet példa két egymáshoz közel elhelyezkedő téglalap alakú mesterséges tó generalizálása. Ha a simításhoz mondjuk Bézier-

görbéket használunk, akkor az ilyen téglalap alakú poligon az eljárás után ovális alakú lesz, ezek pedig átfedhetik egymást. Bár az átfedést kijavíthatjuk a topológia vizsgálatával, a kapott eredmény akkor sem lesz hű a kiindulási állapothoz.

6.1. Felületi elemek simítása az ArcMap programban

Az ArcMap-ben az eddigiek példájára szintén megtalálható a simításhoz használt „Smooth Line" lehetőség a „Cartography Tools"-on belül. A vonalas simításhoz hasonlóan itt is két algoritmus közül választhatunk: a „PAEK" és a Bézier-interpoláció között. Mint az előbbiekben itt sincs lehetőségünk túl sok beállításra. Az „Input Features", „Output Feature Class", az algoritmus, és annak toleranciájának értékén kívül a topológiát tudjuk még vizsgálni és javítani. A „PAEK" algoritmus használatánál adott még egy beállítás, mellyel eldönthetjük, hogy az eljárás megtartsa-e a körvonal végpontjait. Nem nagyon érdemes élni a lehetőséggel, hiszen ha bepipálva hagyjuk, az azt jelenti, hogy a simított alakzatunknak egy csúcsánál megmarad az egyszerűsített pont. A kapott poligonunk körvonala egy ponton élesen megtörik.

Az első lehetőség a „PAEK" használatával meglepően szép eredményeket kaptam. Mivel az eljárás hasonló a Chaiken-algoritmushoz, így kisebb tolerancia értéke mellett jól követi az egyszerűsített körvonalat, ennek ellenére mégis egyenletes a simítás. A tolerancia további növelésével is kiegyensúlyozott eredményeket kapunk, még a nehezen simítható, keskeny U-alakú morotváknál is.

A Bézier-interpolációnál a simítás esetén sem tudunk megadni határértéket. A eljárás során a vonalas elemek simításánál nem észlelt probléma került elő. Mivel a simított vonal mindig keresztül megy az poligon eredeti pontjain, ezért míg a hosszanti, kis szögben megtörő részleteket itt is szépen kisimította, addig a nagy törésszögű csúcsok közelében jelentősen eltért az egyszerűsített körvonaltól. Ennek eredményeképpen az egymáshoz közel elhelyezkedő nagyjából téglalap alakú vízfelületek a simítás után jóval nagyobb területű gömbölyded formákká lettek, ezáltal sok esetben átfedésbe kerültek egymással. Ezt a hiba akkor is fennáll, ha élünk a topológiai vizsgálattal. Mindez inkább a kisebb területű elemek simításánál szembetűnő. Egy nagyobb vízfelületnél ezek annyira nem okoznak problémát. Ebben az esetben is feltűnő lehet azonban, ha az egyszerűsített körvonal maradt hirtelen kiugró rész. Az ilyen részek környezetében szintén jelentős lehet a torzulás. A fent leírtak

miatt a Bézier-interpolációval történő simítást inkább csak egyenletesebb, kevésbé tagolt partvonalú vízfelületek generalizálására ajánlanám.

7. Példa a vízrajz generalizálására

A folyóvizek közép- és kisméretarányú térképi ábrázolása középvízhozam alapján

Az 1970-es kiadású Térképi generalizálás című jegyzetben Dr. Rátóti Benő merőben új oldalról közelíti meg a vízfolyások generalizálását. Az ő ötlete az akkoriban elfogadott módszerek: a hossz alapján és a szélesség alapján történő generalizálás helyett az egyes folyók középvízhozamának, vagyis a sokévi vízhozamok számtani középértékének figyelembevételén alapszik. Választását azzal indokolta, hogy míg egy folyó hossza nem sokat mondd annak jelentőségére vonatkozóan, és a meder szélessége sem meghatározó, hiszen több különböző terepi adottságtól is függ, ezért kis távolságokon belül is jelentősen változhat; addig a középvízhozam „olyan komplex adat, amely a folyó összvízmennyiségének, hosszának, szélességén, mélységének, és sebességének függvénye, amely az eredettől a torkolatig folyamatosan növekszik". Rátóti módszerét összevetve a Töpfer-féle gyökszabállyal kiderül, hogy bár mindkét eljárás empirikus alapokon nyugszik, néhány esetet leszámítva jelentős különbségek vannak a vonalvastagságban; különösen a nagysebességű hegyvidéki folyók ábrázolásánál. Az összevetés alapja egy Rátóti által több akkori térkép és atlasz vizsgálata után elkészített grafikon, melyen az egyes középvízhozamokhoz tartozó vonalvastagságokat tünteti fel különböző méretarányokban. A grafikonról leolvasható, hogy mekkora vonalvastagsággal kell ábrázolni egy adott középvízhozamú folyót, valamint, hogy mely folyók nem ábrázolhatóak már bizonyos aránymérték mellett. Kimondja, hogy az ábrázolt folyóvizeket teljes hosszukban kell ábrázolni, továbbá, hogy a földrajzilag vagy gazdaságilag különösen fontos vízfolyásokat akkor is ábrázolni kell, ha azok az alsó határérték alatt vannak. A folyókanyarulatok számának csökkentése terén azt vette figyelembe, hogy míg a térképi területek négyzetesen csökkennek, addig a vonalvastagságok csak kevésbé, így egy úgynevezett „öngenerációs folyamat" jön létre, ami azt jelenti, hogy azok a kanyarulatok módosulnak vagy esnek ki, amelyeket az adott méretarány, nagyságrend és vonalvastagság mellett már nem lehet ábrázolni. Megemlíti azt is, hogy ezzel a módszerrel megfelelően lehet dolgozni olyan alaptérképekből, melyek tartalmilag helytelenek, hiszen a középvízhozam egy, az adott folyóra jellemző, önálló adat, mely hidrológiai évkönyvekből (manapság már különböző adatbázisokból) kiolvasható. (Stegena, 1970)

8. Összefoglalás

Dolgozatomban ismertettem a térképi generalizálás alapjait, automatizálásának rövid történetét, a hagyományos és az automatizált generalizálás közötti szemléletbeli különbségeket, valamint a különböző algoritmusok működését, végül pedig megvizsgáltam némelyiket két ismert program használatával.

A kapcsolódó irodalom áttekintése közben meglepődve tapasztaltam, hogy milyen sok kutatásnak képezi alapját a térképi generalizálás automatizálása. Még a mai napig sem létezik tökéletesen működő, teljesen automatizált generalizálási folyamat, de biztos vagyok benne, hogy a jövőben el fog készülni egy olyan általánosan használható eljárás, mely a kívánt szempontoknak és elvárásoknak megfelelve maradéktalanul képes generalizálni.

Munkám során rengeteg tapasztalatra tettem szert, betekintést nyerhettem az ismerős programok általam eddig nem ismert részleteibe, és azok működésébe. Új ismeretekkel gazdagodtam az automatizálás történetének alakítóinak munkásságáról, a témához való hozzáállásáról, és ezek fontosságáról.

Véleményem szerint ez egy igen érdekes terület, mely sokban hozzájárul a térképtudományok fejlődéséhez.

Köszönetnyilvánítás

Köszönettel tartozom témavezetőmnek, *Ungvári Zsuzsannának* a dolgozattal kapcsolatos meglátásaiért, a szakirodalomhoz való hozzáférésben nyújtott segítségéért, a folyamatos konzultációért, valamint az irántam tanúsított türelméért.

Irodalomjegyzék

- **Klinghammer István (2010):** Térképészet és geoinformatika I., szerk.: Klinghammer István. ELTE Eötvös Kiadó, Budapest
- **Douglas, D. – Peucker, T.: 1973:** "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer 10(2), 112–122 (1973) doi:10.3138/FM57-6770-U75U-7727
- **NCGIA Core Curriculum, 1994:** szerk: Márkus Béla, Márton Mátyás, Paksi Judit (1994) Térinformatikai alapismeretek 48. fejezet: Vonalgeneralizálás, ford.: Márton Mátyás.
- **Terry A. Slocum (2004):** Slocum Thematic Cartography and Geographic Visualization (2nd Edition) (Prentice Hall Series in Geographic (2e). Chapter 6. Scale and Generalization. 103-120. oldal.
- **Stegena, 1970:** Térképi generalizálás. szerk.: Stegena Lajos. ELTE Természettudományi Kar, Kézirat, 1970. 54-77. oldal
- **M. Visvalingam – J. D. Whyatt (1993):** Line generalisation by repeated elimination of points 1993/1, 46-51. oldal
- Zeshen Wang – Jean-Claude Müller (1998): Line generalisation based on analysis of shape characteristics 1998/1, 3-15. oldal

Internetes hivatkozások:

- **ArcGIS Help 10.1** http://resources.arcgis.com/en/help/main/10.1/index.html#/An_overview_of_the_Cartography_toolbox/007000000003000000/ Utolsó elérés: 2015. május 14.
- **Elmar de Koning (2011): Polyline Simplification** http://www.codeproject.com/Articles/114797/Polyline-Simplification Utolsó elérés: 2015. május 14.
- **On-Line Geometric Modeling Notes: Chalkin's algorithm for curves** http://graphics.cs.ucdavis.edu/education/CAGDNotes/Chaikins-Algorithm/Chaikins-Algorithm.html Utolsó elérés: 2015. május 14.
- **Wikipedia: Bezier Curve** http://en.wikipedia.org/wiki/B%C3%A9zier_curve Utolsó elérés: 2015. május 14.

Ábrajegyzék

3.1.1.–3.4.1. ábra: Elmar de Koning (2011): Polyline Simplification http://www.codeproject.com/Articles/114797/Polyline-Simplification Utolsó elérés: 2015. május 14.

3.5.1. ábra: M. Visvalingam – J. D. Whyatt (1993): Line generalisation by repeated elimination of points 1993/1, 46-51. oldal

3.6.1. –3.8.1. ábra: Elmar de Koning (2011): Polyline Simplification http://www.codeproject.com/Articles/114797/Polyline-Simplification Utolsó elérés: 2015. május 14.

4.3.1. ábra: On-Line Geometric Modeling Notes: Chalkin's algorithm for curves http://graphics.cs.ucdavis.edu/education/CAGDNotes/Chaikins-Algorithm/Chaikins-Algorithm.html Utolsó elérés: 2015. május 14.

4.4.1. ábra: Wikipedia: Bezier Curve http://en.wikipedia.org/wiki/B%C3%A9zier_curve Utolsó elérés: 2015. május 14.

A többi saját készítésű ábra.

Nyilatkozat

Alulírott, Balogh Dániel nyilatkozom, hogy jelen szakdolgozatom teljes egészében saját, önálló szellemi termékem. A szakdolgozatot sem részben, sem egészében semmilyen más felsőfokú oktatási vagy egyéb intézménybe nem nyújtottam be. A szakdolgozatomban felhasznált, szerzői joggal védett anyagokra vonatkozó engedély a mellékletben megtalálható.

A témavezető által benyújtásra elfogadott szakdolgozat PDF formátumban való elektronikus publikálásához a tanszéki honlapon

HOZZÁJÁRULOK NEM JÁRULOK HOZZÁ

Budapest, 2015. május 15.

…….
a hallgató aláírása