Universität Osnabrück
Institut für Umweltsystemforschung
GIS-Modell-Integration
Veranstaltungsleiter: Dr. Jürgen Berlekamp
Wintersemester 2021/2022

Protokoll für das Modul GIS-Modell-Integration

Vorgelegt von:
Jana Kombrink-Lübbe

Osnabrück, den 12.03.2022

Inhaltsverzeichnis

1 Vorbereitung und Habitatsanalyse

1.1 Python Notebooks in ArcGIS

2) Die größte zusammenhängende Waldfläche umfasst ein Gebiet, welches östlich von Wallenhorst, westlich von Ostercappeln, nördlich von Espowe und südlich von Kalkriese liegt. Dieses Waldgebiet umfasst unter Berücksichtigung der geforderten Mindestabstände zu Straßen und Bahnlinien eine Fläche von circa 26.026.725,3959 Quadratmetern.

3) Um im Code verschiedene Abstandsmaße ausprobieren zu können, wird eine neue Funktion definiert. Diese Funktion erhält als Übergabeparameter den Abstand in Metern als String, einen Eingabelayer sowie einen Ausgabelayer. Mit den übergebenen Parametern wird innerhalb der definierten Funktion die Buffer Funktion der arcpy Schnittstelle aufgerufen und der. Der Outputlayer, welcher den Buffer beinhaltet, wird als Ergebnis der neu definierten Funktion zurückgegeben.

Code/1_1.py

```
# Importieren des Moduls arcpy
import arcpy

# Variablen initialisieren
laubwald = "CLC_c311"
nadelwald = "CLC_c312"
mischwald = "CLC_c313"
verkehr = "ver021_lk"

# Zeilen fuer den Nahverkehr auswaehlen (OBA = 3101 AND BDU = 1001)
arcpy.SelectLayerByAttribute_management(verkehr, "NEW_SELECTION","OBA = 3101 AND BDU = 1002")
# aus den selektierten Zeilen im Layer Verkehr einen neuen Layer erstellen
nahverkehr = arcpy.CopyFeatures_management(verkehr, "nahverkehr")

# ebenso fuer Fernverkehr und fuer Bahnlinien je einen neuen Layer erstellen
# fernverkehr:
arcpy.SelectLayerByAttribute_management(verkehr, "NEW_SELECTION","OBA = 3101 AND BDU = 1001")
fernverkehr = arcpy.CopyFeatures_management(verkehr, "fernverkehr")
# bahnlinien:
arcpy.SelectLayerByAttribute_management(verkehr, "NEW_SELECTION", "OBA = 3201")
bahn = arcpy.CopyFeatures_management(verkehr, "bahn")

# Funktion schreiben, die als Parameter einen Abstand x, sowie In- und Outputlayer erhaelt
# ein Buffer mit dem uebergebenen Abstand wird um den Inputlayer gelegt
def baueBuffer(abstand, inputlayer, outputlayer):
    return arcpy.analysis.Buffer(inputlayer, outputlayer, abstand, "FULL", "ROUND", "ALL")

# Funktion fuer jede Verkehrsart einmal aufrufen und Output speichern
nahverkehrBuff = baueBuffer("200 Meters", nahverkehr, "nahverkehrBuff")
fernverkehrBuff = baueBuffer("500 Meters", fernverkehr, "fernverkehrBuff")
bahnBuff = baueBuffer("300 Meters", bahn, "bahnBuff")

# die drei Output Layer bezueglich der Buffer zu einem Layer zusammenfassen
# erster Parameter ist eine Liste der Layer, die zusammengefasst werden sollen
gesamtVerkehrBuff = arcpy.Merge_management([nahverkehrBuff, fernverkehrBuff, bahnBuff], "
    gesamtVerkehrBuff")

# anschliessend werden die Grenzen innerhalb des Verkehrspufferlayers entfernt
gesamtVerkehrBuffDissolved = arcpy.Dissolve_management(gesamtVerkehrBuff, "
    gesamtVerkehrBuffDissolved")

```

```
# ebenso werden die drei Waldarten zu einem Layer zusammengefuegt
gesamtWald = arcpy.Merge_management([nadelwald, mischwald, laubwald], "gesamtWald")
# danach werden die inneren Grenzen aufgeloest
gesamtWaldDissolved = arcpy.Dissolve_management(gesamtWald, "gesamtWaldDissolved")

# Ueberschneidung von Wald und Strassen als Polygon erstellen
intersectWaldStrassen = arcpy.analysis.Intersect([gesamtWaldDissolved,
    gesamtVerkehrBuffDissolved], "intersectWaldStrassen")

# von dem Wald die ueberlappenden Gebiete abziehen
waldOhneStrassen = arcpy.analysis.Erase(gesamtWaldDissolved, intersectWaldStrassen,
                                        "waldOhneStrassen")

# das grosse Waldpolygon in mehrere einzelne Polygone aufteilen
waldMultiToSingle = arcpy.MultipartToSinglepart_management(waldOhneStrassen,
            "waldMultiToSingle")

# Feature Class in ein Array kopieren, nur die OBJECTID und Shape_Area als Felder mitnehmen
array = arcpy.da.FeatureClassToNumPyArray(waldMultiToSingle, ["OID@", "SHAPE@AREA"])

maximum = 0      # bisher groesste zusammenhaengende Waldflaeche
oid = 0          # OBJECTID
# array durchlaufen mit Laufvariable i, dabei groesste zusammenhaengende Waldflaeche suchen
for i in array:
    maximum = maximum + 1
    if i[1] > maximum:
        maximum = i[1]
        oid = i[0]

print(oid)       # Polygon mit der OBJECTID 183 hat die groesste Waldflaeche

# Selektieren der groessten Waldflaeche,
groessteWaldflaeche = arcpy.SelectLayerByAttribute_management(waldMultiToSingle,
            "NEW_SELECTION", "OBJECTID = 183")
# Folgendes funktioniert leider nicht:
# whereClause = "OBJECTID = " + str(oid)    # integer zu einem String casten
# arcpy.SelectLayerByAttribute_management(waldMultiToSingle, "NEW_SELECTION", whereClause)
```

1.2 Habitatsanalyse einer Vogelart

1) Die Kriterien der Habitatwahl des Mückenfängers und die anschließende Gesamtbewertung können mit folgenden GIS-Funktionen umgesetzt werden:

   - Geländehöhe und Hangneigung: aus den Layern elevelt250_shp und slopelt40_shp mit der Intersect Funktion nur die Gebiete behalten, welche in beiden Layern vorhanden sind
   - Straßen: in majorrds_shp die kleinen und größeren Straßen getrennt voneinander nach den vorgebenen Kriterien selektieren und je nach dem Feld Distance einen entsprechenden Buffer um die Straßen legen, die Layer mit den Puffern um die Straßen werden mit der Erase-Funktion von den möglichen Habitaten abgezogen
   - Klimazonen: für die zwei unterschiedlichen Klimazonen werden diese getrennt voneinander selektiert und mit den möglichen Habitaten mit Hilfe der Intersect Funktion auf Überlappungen geprüft
   - Vegetation: nachdem die den Klimazonen entsprechenden Mindestgrößen von Habitaten mit den bis dahin vorliegenden Habitaten mit Hilfe der Intersect

Funktion geupdated wurden, muss zuletzt noch die Vegetation berücksichtigt werden, dazu wird die Vegetationsfläche innerhalb der Habitate berechnet und einzelne nicht zusammenhängende Vegetationsflächen innerhalb eines Habitats aufaddiert, je größer die Vegetationsfläche in einem Habitat ist, desto geeigneter ist es als Habitat

Welche Fläche in diesem Modell als mögliches Habitat eingestuft wird, hängt in erster Linie von der Erfüllung der Kriterien bezüglich der Geländehöhe, der Hangneigung, der Distanz zum Verkehr und der Klimazonen ab. Wie geeignet diese möglichen Habitate dann sind, wird durch die Gesamtgröße an Vegetationsfläche innerhalb eines jeden Habitates bestimmt. Je mehr Vegetationsfläche in einem Habitat vorliegt, desto besser.

2) Darstellung der Ergebnisse der Habitatsanalyse für den Mückenfänger:

Abbildung 1: Karte zur Habitatanalyse für den Mückenfänger

Code/1_2.py

```
1 # Importieren des Moduls arcpy
2 import arcpy
3 # aktuellen Workspace dateipfad abspeichern
4 workspace = arcpy.env.workspace
```

```
5
6  # Im Layer elevlt250_shp werden alle Bereiche mit einer Hoehe <= 250 m gespeichert
7  # Im Layer slopelt40_shp werden alle Bereiche mit einer Hangneigung <= 40 % gespeichert
8  # Die ueberschneidung dieser beiden Layer liefert die Gebiete, in welchen eine Hoehe <= 250 m
       und eine Hangneigung <= 40 % vorliegt.
9  arcpy.analysis.Intersect(["elevlt250_shp", "slopelt40_shp"], "elevIntersectSlope")
10
11 # majorrds: In der Spalte Distance ist der zu den Strassen benoetigte Abstand festgelegt
12 # Zuerst die kleineren Strassen selektieren und fuer diese einen Buffer mit 820 Feet anlegen
13 arcpy.SelectLayerByAttribute_management("majorrds_shp", "NEW_SELECTION", "Distance = 820")
14 arcpy.analysis.Buffer("majorrds_shp", "strassenBuffed820", "820 Feet", "FULL", "ROUND", "ALL"
       )
15
16 # Anschliessend die grossen Strassen selektieren und einen Buffer mit 1312 Feet Abstand um
       die groesseren Strassen legen
17 arcpy.SelectLayerByAttribute_management("majorrds_shp", "NEW_SELECTION", "Distance = 1312")
18 arcpy.analysis.Buffer("majorrds_shp", "strassenBuffed1312", "1312 Feet", "FULL", "ROUND", "
       ALL")
19
20 # Die beiden Strassen Layer zusammenfuegen zu einem Layer
21 arcpy.Merge_management(["strassenBuffed1312", "strassenBuffed820"], "strassenBuffedGesamt")
22
23 # Von den moeglichen Gebieten bezueglich Hangneigung und Gelaendehoehe die Strassen inklusive
       Bufferzone abziehen
24 arcpy.analysis.Erase("elevIntersectSlope", "strassenBuffedGesamt", "habitateOhneStrassen")
25
26 # Klimazone 1 und 2 entsprechen der CLIMATE_ID = 2
27 arcpy.SelectLayerByAttribute_management("climate_shp", "NEW_SELECTION", "CLIMATE_ID = 2")
28 # ueberschneidungszone von den Klimazonen 1 und 2 mit den moeglichen Habitaten nach Abzug der
       Strassenabstaende
29 arcpy.analysis.Intersect(["climate_shp", "habitateOhneStrassen"], "klima2IntersectHabitate")
30
31 # Klimazone 1 und 2 entsprechen der CLIMATE_ID = 3
32 arcpy.SelectLayerByAttribute_management("climate_shp", "NEW_SELECTION", "CLIMATE_ID = 3")
33 # ueberschneidungszone von den Klimazonen 3 mit den moeglichen Habitaten nach Abzug der
       Strassenabstaende
34 arcpy.analysis.Intersect(["climate_shp", "habitateOhneStrassen"], "klima3IntersectHabitate")
35
36 # multi to single fuer Klimazone 1 und 2
37 arcpy.MultipartToSinglepart_management("klima2IntersectHabitate", "
       habitateOhneStrassenMultiToSingleKlima2")
38 # multi to single fuer klimazone 3
39 arcpy.MultipartToSinglepart_management("klima3IntersectHabitate", "
       habitateOhneStrassenMultiToSingleKlima3")
40
41 # Gebiete der Klimazone 1 und 2 muessen mindestens 1089000 quadratfeet gross sein
42 arcpy.SelectLayerByAttribute_management("habitateOhneStrassenMultiToSingleKlima2", "
       NEW_SELECTION", "Shape_Area >= 1089000")
43 arcpy.CopyFeatures_management("habitateOhneStrassenMultiToSingleKlima2", "habitate2")
44
45 # Gebiete der Klimazone 3 muessen mindestens 2178000 quadratfeet gross sein
46 arcpy.SelectLayerByAttribute_management("habitateOhneStrassenMultiToSingleKlima3", "
       NEW_SELECTION", "Shape_Area >= 2178000")
47 arcpy.CopyFeatures_management("habitateOhneStrassenMultiToSingleKlima3", "habitate3")
48
49 # moegliche Habitate aus den Klimazonen
50 arcpy.Merge_management(["habitate2", "habitate3"], "habitate")
51
52 # Gebiete mit der Kuestensalbeistaude sind mit HABITAT = 1 gekennzeichnet
53 arcpy.SelectLayerByAttribute_management("vegtype", "NEW_SELECTION", "HABITAT = 1")
54 arcpy.CopyFeatures_management("vegtype", "vegetation")
55
56 # ueberlappung von Vegetation mit Habitatraeumen feststellen, um Vegetationsflaechen, die
       ueber die Habitate hinausgehen, zu entfernen
57 arcpy.analysis.Intersect(["vegetation", "habitate"], "vegetationIntersectHabitate")
58
59 # Falls Vegetationsflaechen durch das Entfernen der nicht mit Habitatgebieten ueberlappenden
       Vegetation getrennt wurden,
```

```
60 # benoetigen diese nun je ein eigenes Polygon
61 arcpy.MultipartToSinglepart_management("vegetationIntersectHabitate", "
       vegetationIntersectHabitateMultiToSingle")
62
63 #  Ueberschneidung innerhalb der Habitate mit der Vegetation bestimmen
64 arcpy.analysis.Identity("habitate", "vegetationIntersectHabitateMultiToSingle", "
       identityVegHab")
65
66 # Die FID_habitate sagt, zu welchem habitat die Vegetationszone gehoert
67 # erstmal Tabelle aufraeumen, klappt noch nicht so ganz, die doppelten Felder sind noch
       drinnen
68 arcpy.management.DeleteField("identityVegHab", ["FID_climate_shp", "CLIMATE_ID", "Zone_", "
       FID_habitateOhneStrassen", "FID_elevIt250_shp",
69                                                "FID_slopeIt40_shp", "ORIG_FID", "
       FID_vegetationIntersectHabitateMultiToSingle", "FID_vegetation",
70                                                "HOLLAND95", "HABITAT", "VEG_TYPE"])
71
72 # Feature Class in ein Array kopieren, dabei nur bestimmte Felder mitnehmen (Objektid und
       Shape_Area hier)
73 array = arcpy.da.FeatureClassToNumPyArray("identityVegHab", ["OID@", "FID_habitate", "
       SHAPE@AREA"])
74 # print(array)
75
76 oid = 0
77 fid = 0
78 fidalt = 1
79 area = 0
80 veg = []
81 sum = 0
82
83 # neues Array erstellen, indem in array die summen der Vegetation fuer jedes Habitat einzeln
       berechnet und gespeichert werden
84 for i in array:
85     oid = i[0]
86     if oid >= 95: # ab Eintrag 95 fangen die kleineren Vegetationsgebiete in den Habitaten an
87         fid = i[1]
88         area = i[2]
89         if fid == fidalt:
90             sum = sum + area
91             #print(str(sum) + " bei der oid " + str(oid))
92         else :
93             veg.append([fidalt, sum])                    # Daten Speichern
94             fidalt = fid                                 # jetzt kommt das naechste Habitat
       dran
95             sum = area                                   # Summe mit neuem Startwert updaten
96
97 # letzte summe auch noch speichern
98 veg.append([fidalt, sum])
99
100 #for j in veg:
101     #print(j[0])
102
103 # aus dem normalen Array ein numpy array machen
104 # numpyveg = numpy.array(veg, dtype=[("oid", int), ("vegarea", float)])
105 numpyveg = numpy.array(veg)
106
107 # anscheinend muessen numpy arrays noch formatiert werden
108 struct_array = numpy.core.records.fromarrays(numpyveg.transpose(), numpy.dtype([("oid", int),
        ("vegarea", float)]))
109 print(struct_array)
110
111
112 # wirft beim ersten mal keinen Fehler, aber ein Read/Write Lock sitzt dann 3 Jahre auf dem
       Layer und
113 # man kann den Layer in der Zeit nicht betrachten, Lock verschwand nach Tabel export...
114 arcpy.da.ExtendTable("habitate", "OBJECTID", struct_array, "oid")
```

2 Soil Loss USLE

1) Darstellung des Untersuchungsgebiets:

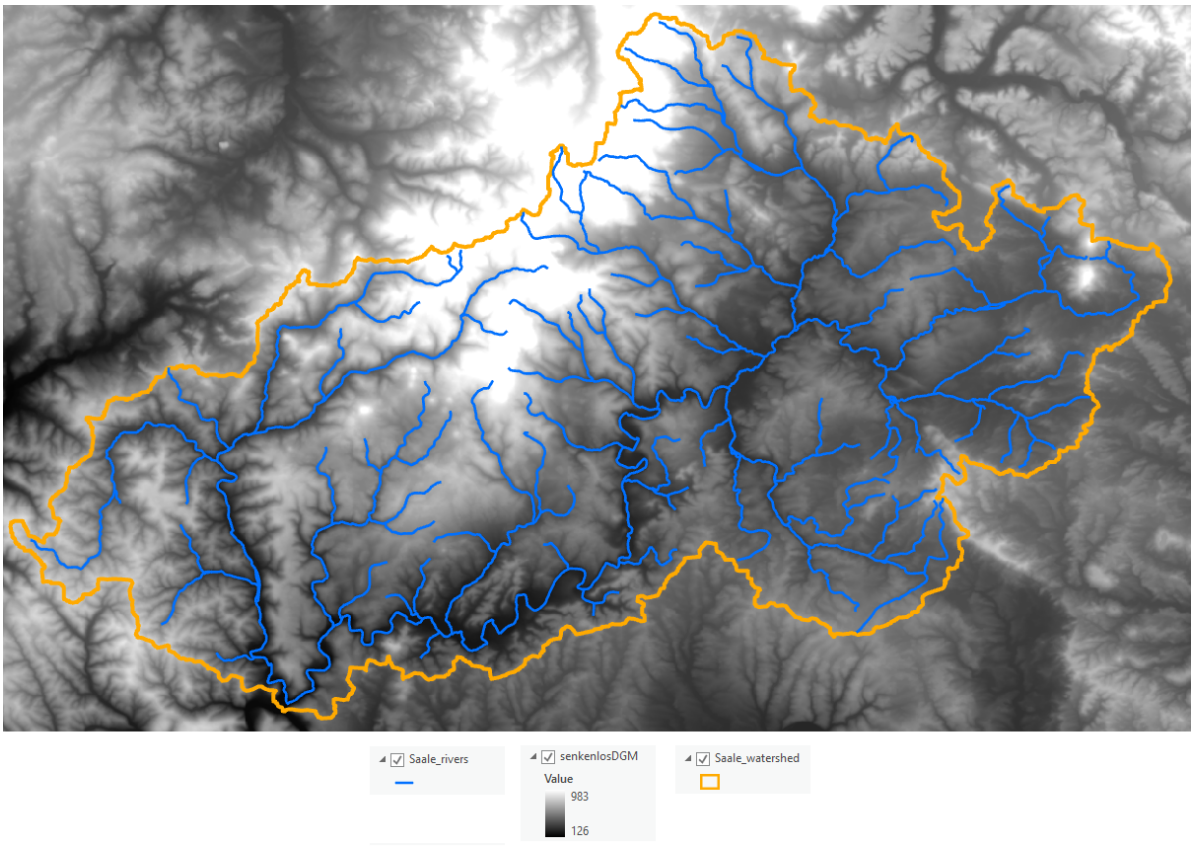

Abbildung 2: Wassereinzugsgebiet der fränkischen Saale

3) Werte aus dem senkenlsen DGM ableiten:
verfülltes Erdvolumen: 782.804.167,6 [m3]
mittlere Füllhöhe: 122.523,7 [m3]
gefüllte Fläche: 2,8 [%]

4) C-Faktor und K-Faktor: C-Faktor: landwirtschaftliche Bewirtschaftungsart (Bodenbedeckungs und Managementfaktor), spiegelt Einfluss der landwirtschaftlichen Bewirtschaftung (angebaute Nutzpflanze, Anbausystem) wider: C = 1 für Schwarzbrache, C ¡ 1 für alle anderen Flächen, da Nutzpflanzenbewuchs Bodenabtrag entgegensteht, z.B.: C = 0.5 (Kartoffeln, Karotten), C = 0.02 (Weide, Wiese), C = 0.01 (Wald). Bei dem C-Faktor ist zu erkennen, dass im Osten des Untersuchungsgebietes vermehrt Nutzpflanzenbewuchs vorhanden ist, der sich vermindernd auf die Bodenerosion auswirkt (Werte von 0,5). Im Westen dahingegen sind von der Vegetation eher Waldgebiete, die dem Bodenabtrag weniger stark entgegenstehen (Werte von 0,1).

K-faktor ist Maß für die Erodierbarkeitdes Bodens. Der K-faktor ist abhängig von Bodenbestandteilen: Wasserdurchlässigkeit, Korngrößenverteilung, Anteil organischer Substanz, Aggregatsgröße und Aggregatzustand. Bei dem K-Faktor ist der Großteil der Fläche des Untersuchungsgebietes mit einem Faktor von ca 0,58 bis 0,67 versehen. Lediglich weit im Osten kommen K-Faktor Werte kleiner als 0,58 vor.

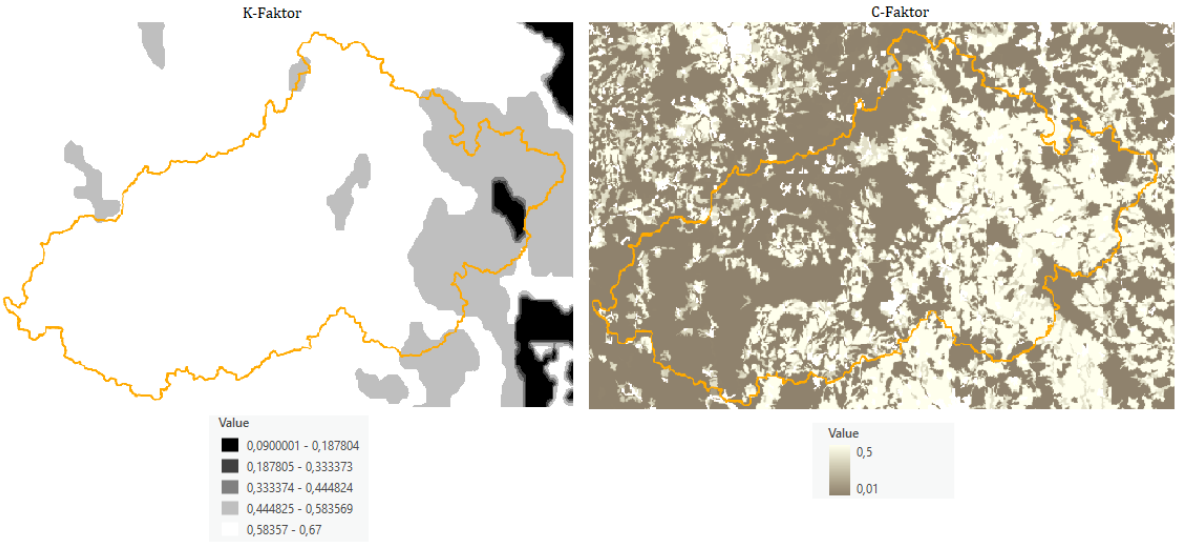

Abbildung 3: K- und C-Faktor

5) Mit der Codezeile 21 wird der LS Faktor berechnet.
Maximum: 7588,03271484375
Mittel: 25,0294874442957
Minimum: 0
Höhere LS Werte befinden sich dem DEM entsprechend ungefähr entlang des Fließgewässer. Im Westen des Einzugsgebietes sind die LS Werte überwiegend höher als im Osten des Gebietes.

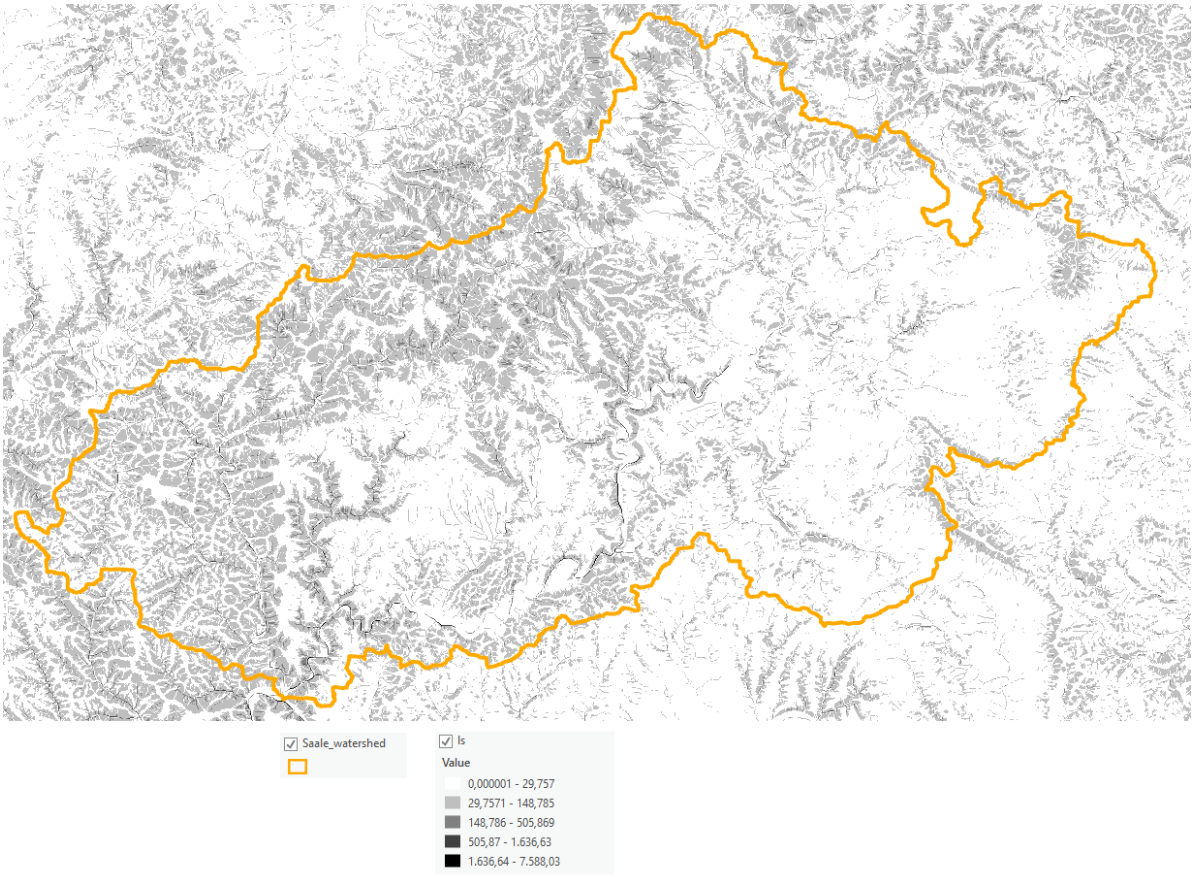

Abbildung 4: LS-Faktor

6) Das Wertespektrum reicht von 0 bis 3053,26978 [t * ha-1 * a-1]. Durchschnittlich liegt der Wert jedoch bei 1,5765 [t * ha-1 * a-1]. Eventuell wäre es bei der Darstellung dann auch schlauer gewesen, die Klassen anders einzuteilen, da nun sehr viele Werte in einer Klasse liegen und es kaum Unterschiede zu geben scheint. Überwiegend liegt in der Saale ein Bodenabtrag zwischen 0 und 100 [t * ha-1 * a-1] vor. Für besonders hohe Bodenabträge sorgen das Geländerelief und Bodenerodierbarkeit.

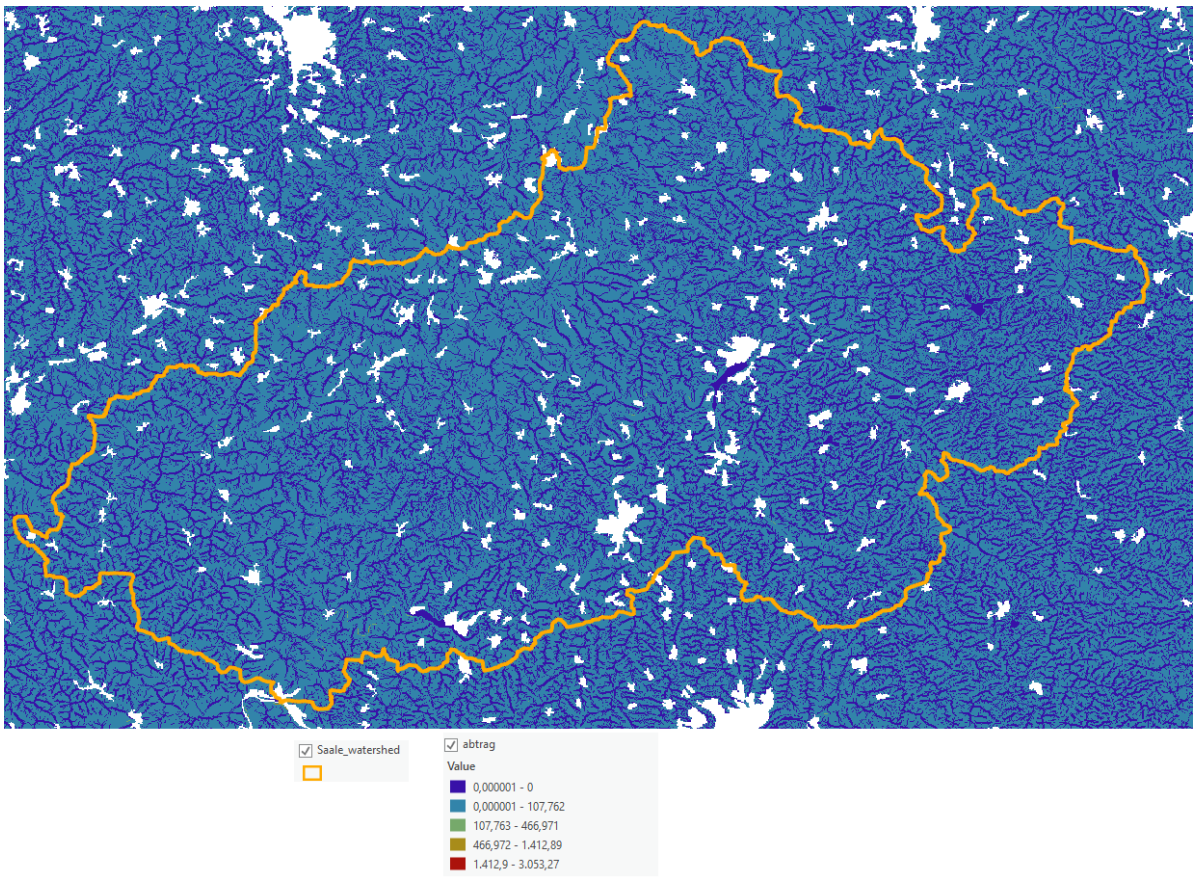

Abbildung 5: Erosionsabtrag

Code/2.py

```python
import arcpy
import math
# importiert alles aus dem Spatial Analyst Tool, ansonsten muesste man ueber arcpy.sa. vor eine Funktion schreiben
from arcpy.sa import*

# 1.3 Raster, welches in der aktuellen Map geladen ist in Python laden, dgm = digitales Gelaendemodell, damit man mit dem Raster Rechnungen durchfuehren kann
dgm = Raster("SRTM_Saale_GK4.tif")

#senkenloses DGM erzeugen
senkenlosDGM = Fill(dgm)

# Fliessrichtung aus dem senkenlosen DGM ableiten
fliessrichtung = FlowDirection(senkenlosDGM)
fliessakkumulation = FlowAccumulation(fliessrichtung)

# Ermitteln Sie die Hangneigung als Gradangaben aus dem senkenlosen
hangneigung = Slope(senkenlosDGM, "DEGREE")

# Berechnung des kombinierten ls Faktors von RUSLE3D
cellSize = 77
ls = ((fliessakkumulation * (cellSize ** 2) * (1 / 22.1)) ** 0.4) * ((Sin(hangneigung * 0.01745) * (1 / 0.0896))** 1.3) * 1.4

# Statistiken vom ls berechnen
lsMax = arcpy.management.GetRasterProperties(ls, "MAXIMUM")
```

```
25 lsMittel = arcpy.management.GetRasterProperties(ls, "MEAN")
26 lsMin = arcpy.management.GetRasterProperties(ls, "MINIMUM")
27
28 # Berechnung des mittleren progostizierten Bodenabtrags: A = R * K * L * S * C * P
29 # R und P werden aufgrund homogener Bedingungen vernachlaessigt: A = K * L * S * C
30 abtrag = Raster("K_factor_Saale.tif") * ls * Raster("C_factor_Saale.tif")
31
32 # Statistiken von abtrag berechnen [t * ha-1 * a-1]
33 abtragMax = arcpy.management.GetRasterProperties(abtrag, "MAXIMUM")
34 abtragMean = arcpy.management.GetRasterProperties(abtrag, "MEAN")
35 abtragMin = arcpy.management.GetRasterProperties(abtrag, "MINIMUM")
```

3 ArcHydro

3.1 Umsetzen der geeigneten Prozessabfolge

1) Zuerst muss entschieden werden, ob das Drainage System dendritic (normal, baumartig), deranged (gestört, unregelmäßig) oder eine Kombination aus deranged und dendritic ist. In dem Fall der Saale ist dendritic zu wählen, da das Flussnetzwerk normale, mäandrierende Verzweigungen aufweist und keine Störungen ersichtlich sind. Da die Streams durch den vorgebenen und somit "benutzerdefinierten"Layer Saale_rivers bekannt sind, ist die Prozessabfolge des Use Case 9 zu wählen (Completely dendritic terrain with known stream locations (using user specified streams)).

2) Die HydroID dient der eindeutigen Identifikation eines Flussabschnitts. Die NextDownID eines Flusssegments beinhaltet die HydroID des nachfolgenden Unterliegersegments, in welches das Wasser des aktuell betrachteten Segments hineinfließen wird. Die NextDownID ist also ein Verweis auf den nächsten flussabwärts anschließenden Flussabschnitt.

   Bei dem DEM Reconditioning geht es um das Einbrennen von Linienfeatures mit Hilfe des AGREE Algorithmus in das DEM. Dafür werden neben dem DEM und dem Linienfeature noch drei Werte benötigt.
   • Stream buffer: ist die Anzahl an Zellen um das Linienfeature herum, für welches die Glättung durchgeführt werden soll. Der Standardwert beträgt 5 Zellen. Eine Veränderung dieses Wertes würde die Bufferzone entweder vergrößern (bei größer als 5) oder verkleinern (bei kleiner als 5). Das heißt, dass je nach Wert sich die Glättung über mehr oder weniger Zellen (bzw. größere oder kleinere Zone) erstrecken würde.
   • Smooth drop/raise value: gibt an um welchen Betrag das Linienfeature in vertikaler Richtung herabgesenkt (bei positiven Werten) oder erhöht (bei negativen Werten) wird. Der Standardwert liegt bei 10 vertikalen Einheiten des DEM. Eine Veränderung würde bei größeren positiven Werten für eine vertikal tiefere Einbrennung sorgen, während kleinere negative Werte zu einer stärkeren vertikalen Erhöhung führen würden. Dieser Wert wird verwendet, um das DEM in dem gepufferten Bereich zu interpolieren, sodass keine Inselartefakte oder harte Abbruchkanten entstehen sollten. Eine Veränderung dieses Wertes hat also entspre-

chende Auswirkungen auf die Pufferzone um das Linienfeature herum. Je nach der Höhendifferenz zwischen Pufferzentrum und Puffergrenze fürht eine Veränderung dieses Wertes zu einem steileren Gefälle oder einem steileren Anstieg.
• Sharp drop/raise value: ist ein zusätzlicher Betrag, um welchen das Linienfeature in vertikaler Richtung zusätzlich erniedrigt (bei positiven Werten) oder erhöht (bei negativen Werten) wird. Der Standardwert liegt bei 1000 vertikalen Einheiten des DEM. Dieser weitere Wert ist notwendig, da nach der Glättung das eigentliche Linienfeature durch die Interpolation sich nicht mehr stark von dem Geländemodell abhebt. Dies ist quasi die eigentliche Einbrennung des Linienfeatures, da dieser Wert wird nicht für die Interpolation verwendet wird. Eine Veränderung dieses Wertes würde bei größeren positiven Werten zu einer tieferen vertikale Einbrennung führen, während kleinere negative Werte zu einer stärkeren vertikalen Erhöhung führen würden.

Code/3_1.py

```
# Importieren des Moduls arcpy und der Toolbox Arc_Hydro_Tools_Pro
import arcpy
arcpy.ImportToolbox("C:\Program Files\ArcGIS\Pro\Resources\ArcToolBox\Toolboxes\Arc_Hydro_Tools_Pro")

# 1) Create Drainage Line Structures
with arcpy.EnvManager(scratchWorkspace=r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb", workspace=r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb"):
    arcpy.archydropro.CreateDrainageLineStructures("SRTM_Saale_GK4.tif", "Saale_rivers", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Layers\FdrStr", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Layers\StrLnk", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb\Layers\DrainageLine", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb\DrainageLine_FS", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb\Layers\EditPoint", "CLEAR_ANGLES_NO", "USE_RASTER_EXTENT_FALSE", None, 448.747618514969)

# 2) Fill Sinks
arcpy.archydropro.FillSinks("SRTM_Saale_GK4.tif", "SinksFilled", None, None, "ISSINK_NO")

# 3) DEM Reconditioning
arcpy.archydropro.DEMReconditioning("SRTM_Saale_GK4.tif", "StrLnk", 5, 10, 1000, r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Layers\AgreeDEM", "NEGATIVE_NO")

# 4) Fill Sinks, nochmal da beim Einbrennen neue Senken entstanden sein koennen
arcpy.archydropro.FillSinks("AgreeDEM", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Layers\Fil", None, None, "ISSINK_NO")

#5) Flow Direction
arcpy.archydropro.FlowDirection("Fil", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Layers\Fdr", None)

#6) Adjust Flow Direction in Streams
arcpy.archydropro.AdjustFlowDirectioninStreams("Fdr", "FdrStr", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Layers\FdrStrAdj")

#7) Catchment Grid Delineation
arcpy.archydropro.CatchmentGridDelineation("FdrStrAdj", "StrLnk", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Layers\Cat")

#8) Catchment Polygon Processing
arcpy.archydropro.CatchmentPolygonProcessing("Cat", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb\Layers\Catchment")
```

```
29
30 #9) Adjoint Catchment Processing
31 arcpy.archydropro.AdjointCatchmentProcessing("DrainageLine", "Catchment", r"C:\Users\Jana\
       Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb\Layers\
       AdjointCatchment", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\
       Aufgabe1\Aufgabe1.gdb\Catchment_FS", "DrainageLine_FS")
```

3.2 Erzeugen von Flussnetzwerken

3) Der Unterschied zwischen Nodes/Links des schematischen Netzwerkes und Junctions/Edges des Hydro-Networks besteht darin, dass die Links im schematischen Netzwerk ihre geometrische Korrektheit abgelegt haben und stattdessen als einfache Geraden dargestellt werden, während die Links im Hydronetzwerk weiterhin topologisch und geometrisch korrekt abgebildet werden. Im Hydronetzwerk wird das Fließgewässer also sowohl topologisch als auch geometrisch korrekt dargestellt, wohingegen in dem schematischen Netzwerk nur die topologischen Beziehungen der Flussabschnitte von Bedeutung sind. Ein schematisches Netzwerk wird für Fragestellungen verwendet, bei welchen es egal ist, ob ein Flusssegment geometrisch korrekt verläuft oder nicht, weshalb die Information die Geometrie der Flusssegmente nicht mehr benötigt wird.

5) Darstellung der schematischen Fließgewässernetzwerkstruktur der Saale:

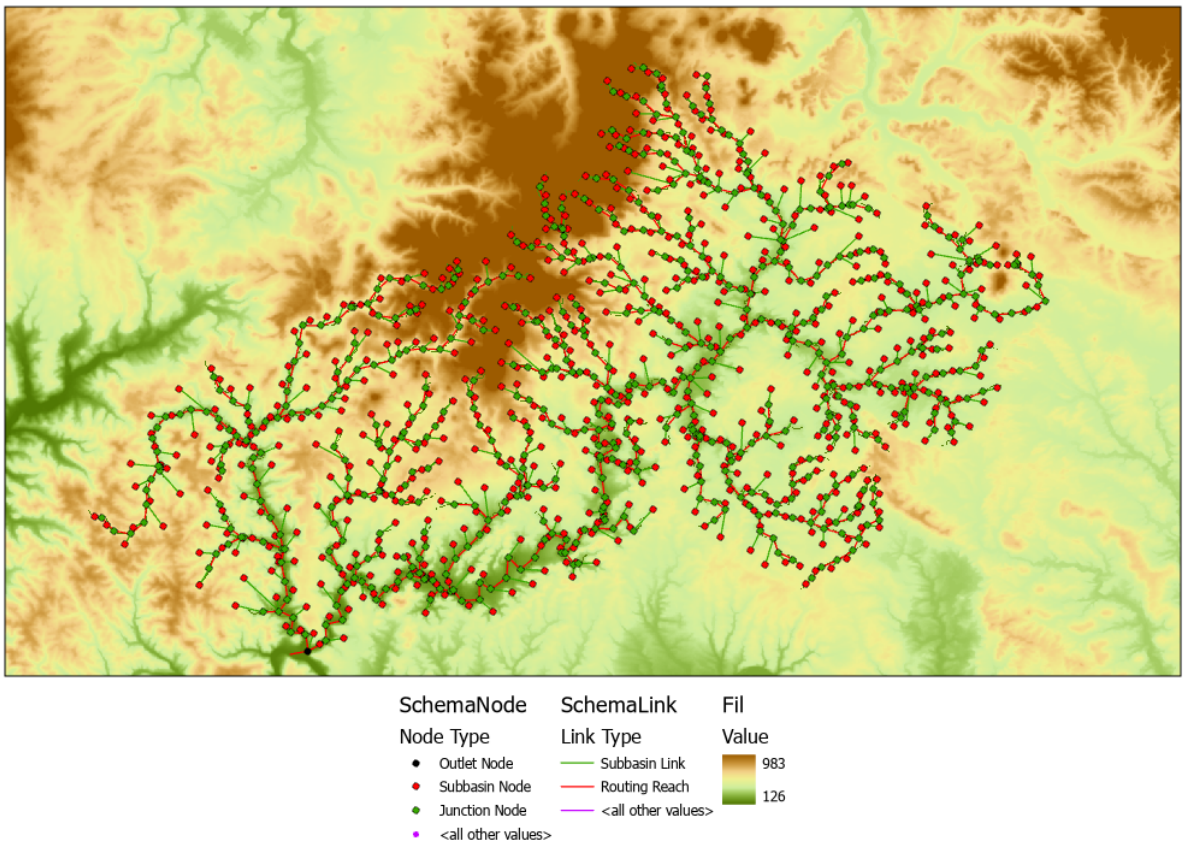

Abbildung 6: Schematisches Netzwerk mit gefülltem Geländemodell

Code/3_2.py

```
# Flow Accumulation
arcpy.archydropro.FlowAccumulation("FdrStrAdj", r"C:\Users\Jana\Dropbox\
    GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Layers\Fac")

# Drainage Point Processing
arcpy.archydropro.DrainagePointProcessing("Fac", "Cat", "Catchment", r"C:\Users\Jana\Dropbox\
    GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb\Layers\DrainagePoint")

# Create Hydro Network from Catchment
# falls ein Fehler bezueglich einer doppelten FeatureID auftritt, muss in der Tabelle
# der DrainageLine Featureclass die Spalte FeatureID geLoescht werden
arcpy.archydropro.CreateHydroNetworkfromCatchment("DrainageLine", "Catchment", "DrainagePoint
    ", "ArcHydro", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\
    Aufgabe1.gdb\Layers\HydroEdge", r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\
    Uebung3\Aufgabe1\Aufgabe1.gdb\Layers\HydroJunction")

# Generate Node Link Schema
arcpy.archydropro.GenerateNodeLinkSchema("HydroJunction", "HydroEdge", r"C:\Users\Jana\
    Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb\Layers\SchemaLink",
    r"C:\Users\Jana\Dropbox\GIS_Modell_Integration\Uebung\Uebung3\Aufgabe1\Aufgabe1.gdb\
    Layers\SchemaNode", "Catchment", None)
```

4 Bakterien

4.1 Ableiten der Abflusshöhe als Raster

1) Bei dem Resampling der mittleren Abflusshöhe wird ein bilinieares Resampling-Verfahren verwendet, da sich eine bilineare Interpolation gut für Anwendungen mit einem kontinuierlichen Wertebereich, wie z.B. der Abflusshöhe, eignet und außerdem zu einer Glättung der Daten führt.

2) Die höchste Abflusshöhe des Datensatzes beträgt 6245,25927734375 [m^3/a] und die kleinste Abflusshöhe des Datensatzes nimmt den Wert von 6,30471563339233 [m^3/a] an.

Code/4_1.py

```
import arcpy
from arcpy.sa import *
import math
ws = arcpy.env.workspace

# 1a) ein Fangraster und die Zellgroesse definieren, welche den Ergebnisrastern von Uebung 3
# entsprechen (z.B. SRTM_Saale_GK4), es muss ein spezifisches Band als snapRaster
# ausgewaehlt werden, hier SRTM_Saale_GK4.tif_Band_1
arcpy.env.snapRaster = "SRTM_Saale_GK4.tif_Band_1"
cellSize = arcpy.sa.Raster(arcpy.env.snapRaster).meanCellHeight

# MQ_Saale_HAD_GK4 zeigt die mittlere Abflusshoehe in [mm/a]
# Resampling der mittleren Abflusshoehe mit bilinearer Methode
MQ_Saale_HAD_GK4_Resample = arcpy.management.Resample("MQ_Saale_HAD_GK4",
    "MQ_Saale_HAD_GK4_Resample", cellSize, "BILINEAR")

# 1b) die resampelte Abflusshoehe ist noch in [mm/a] angegeben, es muss also noch eine
# Umrechnung in [m3/a] erfolgen
runoff = Raster("MQ_Saale_HAD_GK4_Resample") * 0.001 * cellSize * cellSize

# speichern
runoff.save(ws + "\\runoff")
```

4.2 Ableiten der Bakterieneinträge aus der Landnutzung

1) Die Landnutzungsdaten wurden über fernerkundliche Methoden erhoben. Insbesondere handelt es sich hierbei um Satellitendaten des europäischen Erdüberwachungsprogramms Copernicus, aufgrund welcher eine Landnutzungsklassifikation vorgenommen wurde.
Im Osten des Einzugsgebietes der fränkischen Saale sind vor allem Agrarflächen dominierend. In der Mitte des Einzugsgebietes und auch vereinzelnd an weiteren Stellen treten Waldflächen auf. Im westlichen Bereich treten vermehrt größere Gras- und andere Vegetationsflächen auf. Die Verteilung der Landnutzungsklassen auf das Einzugsgebiet der fränkischen Saale, aber auch Agraflächen sind dort vorhanden (allerdings nicht so viele wie im Osten). Über das ganze Einzugsgebiet verstreut finden sich immer wieder kleinere urbane Flächen. Die größten dieser

urbanen Flächen sind Bad Kissingen und Bad Neustadt an der Saale. In unmittelbarer Nachbarschaft zu diesen urbanen Flächen befinden sich auch immer wieder Industriegebiete.

3) Der Wertebereich geht von 0 bis 1.179.173.519.360 [cfu/a].

Code/4_2.py

```
# 2b) mit der Lookup Funktion ein Raster mit den EMC Werten aus dem Datensatz
# CLC_Saale extrahieren, Umrechnung von [cfu/100ml] zu [cfu/m3]
emc = Lookup("CLC_Saale", "EMCs_per_100ml") * 10000

# 2c) Resampling des EMC Rasters, sodass es von der Zellgroesse her zum Abflussraster runoff
# passt, Nearest Neighbour als Resamplingmethode verwenden, da es sich bei den EMC Daten
# um eine nicht numerische Groesse handelt
emc_Resample = Raster(arcpy.management.Resample("emc", "emc_Resample", cellSize, "NEAREST"))

# Berechnung der Bakteriellen Fracht, Einheiten: [cfu/m3] * [m3/a] = [cfu/a]
bakLand = emc_Resample * runoff

# speichern
bakLand.save(ws + "\\bakLand")
```

4.3 Akkumulation von Abflusshöhen und Bakterienfrachten auf die Catchments

Code/4_3.py

```
# 3) Akkumulation von Abflusshoehen und Bakterienfrachten auf die Catchments
# das Aggregieren auf die Catchments geschieht mit Hilfe der sa.ZonalStatisticsAsTable
# Funktion, 1=Abflusshoehe, 2=Landnutzungsbakterienfracht, 3=Tierhaltungsbakterienfracht
arcpy.ia.ZonalStatisticsAsTable("CatchID", "Value", "runoff", "ZonalSt_CatchID1", "DATA", "
    ALL", "CURRENT_SLICE", 90, "AUTO_DETECT")
arcpy.ia.ZonalStatisticsAsTable("CatchID", "Value", "bakLand", "ZonalSt_CatchID2", "DATA", "
    ALL", "CURRENT_SLICE", 90, "AUTO_DETECT")
arcpy.ia.ZonalStatisticsAsTable("CatchID", "Value", "BactLoad_cattle_g", "ZonalSt_CatchID3",
    "DATA", "ALL", "CURRENT_SLICE", 90, "AUTO_DETECT")
```

4.4 Simulieren der Frachten

Die Catchment.HydroID wird mit der Node.FeatureID verknüpft. Somit kommt man von den Catchments zu den Catchmentzentroiden (Knoten vom Typ 1). Die Node.DownElemID wird mit der Link.DownElemID verknüpft. Dadurch wird sich für jeden Knoten der flussabwärts nachfolgende Link gemerkt.

1) Die neu zugewiesene DownLinkID besagt für jedes Flusssegment, in welches Flusssegment es hineinfließt, also das nachfolgende Unterliegersegment. Durch Verknüpfung der Link.DownLinkID und der Node.DownElemID werden die Knoten quasi nicht mehr benötigt, da jeder Link nun seinen "Nachfolgerlink"kennt. Eine negative DownLinkID (z.B. -1) bedeutet, dass es sich hierbei um einen Auslasslink handelt, welcher keinen definierten Nachfolger hat. (Insbesondere wurde bei dem Auslasspunkt mit der DownLinkID -2 diese DownLinkID auf -2 gesetzt, damit

deutlich wird, dass die Auslasslinks mit der -1 als DownLinkID in diesen selbst erstellten Link hineinfließen sollen.

3) Aus den bisherigen Simulationen der Bakterienfrachten im Flussnetzwerk ergebeben sich folgende Maximalwerte:
   - Runoff: 901.864.182,9 [m^3/a]
   - Landnutzung: 21.249.914.534.585.900 [cfu/a]
   - Tierhaltung: 1.724.515.874.188.460.000 [cfu/a]
   - Landnutzung und Tierhaltung: 1.745.765.788.723.050.000 [cfu/a]

5) Darstellung der Bakterienfracht-Konzentration:

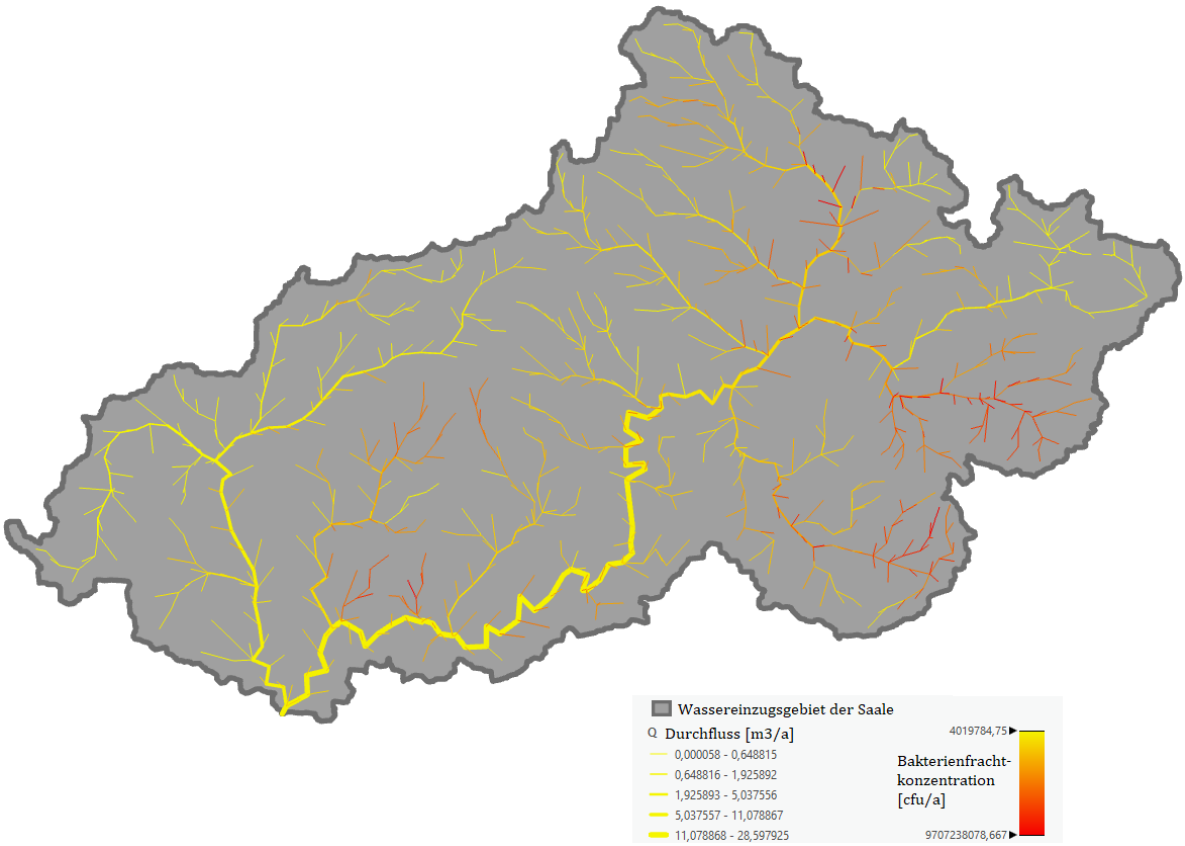

Abbildung 7: Karte bezüglich Durchfluss und Bakterienfrachtkonzentration

An der Mündung der Fränkischen Saale in den Main ergeben sich in vier verschiedenen Szenarien folgende Bakterienkonzentrationenen:

- Tierhaltung: 1.912.168.000 [cfu/m^3]
- Landnutzung: 23.562.210 [cfu/m^3]
- Landnutzung und Tierhaltung: 1.935.730.000 [cfu/m^3]
- Landnutzung, Tierhaltung, Abbau: 111.789.300 [cfu/m^3]

Code/4_4.py

```
# 4d) Simulation mit Gewichtung, Weight muss selbst ausgerechnet werden
# uebernimmt nur das feld reallength in die schema link tabelle
arcpy.management.JoinField("schema_link", "FeatureID", "drain_line", "HydrolD", "RealLength")

# Berechnen von Q
arcpy.CalculateField_management("schema_link", "Q", "!Runoff!/(365*24*60*60)", "PYTHON3", "",
    "FLOAT")

# Berechnen von der Fliessgeschwindigkeit V, ** fuer exponenten
arcpy.CalculateField_management("schema_link", "V", "0.7376 * (35.3147 * !Q!) ** 0.1035", "
    PYTHON3", "", "FLOAT")

# Traveltime: RealLength / V (anschliessend in 1/Tag)
arcpy.CalculateField_management("schema_link", "Traveltime", "!RealLength! / !V! /(60*60*24)"
    , "PYTHON3", "", "FLOAT")

# Weight berechnen
arcpy.CalculateField_management("schema_link", "Weight", "math.e ** (-1.5 * !Traveltime!)", "
    PYTHON3", "", "FLOAT")

# 4e) Berechnung der Konzentration
arcpy.CalculateField_management("schema_link", "Konzentration", "!LandTierAbbau! / !Runoff!",
    "PYTHON3", "", "FLOAT")

# Nach 4e) noch fuer die anderen 3 Szenarien die Konzentration berechnen
arcpy.CalculateField_management("schema_link", "KonzBakLand", "!BakLand! / !Runoff!", "
    PYTHON3", "", "FLOAT")
arcpy.CalculateField_management("schema_link", "KonzBakTier", "!BakTier! / !Runoff!", "
    PYTHON3", "", "FLOAT")
arcpy.CalculateField_management("schema_link", "KonzBakLandTier", "!BakLandTier! / !Runoff!",
    "PYTHON3", "", "FLOAT")
```

4.5 Monatliche Betrachtung

3) Darstellung des Jahresverlaufs der Abflussmenge:

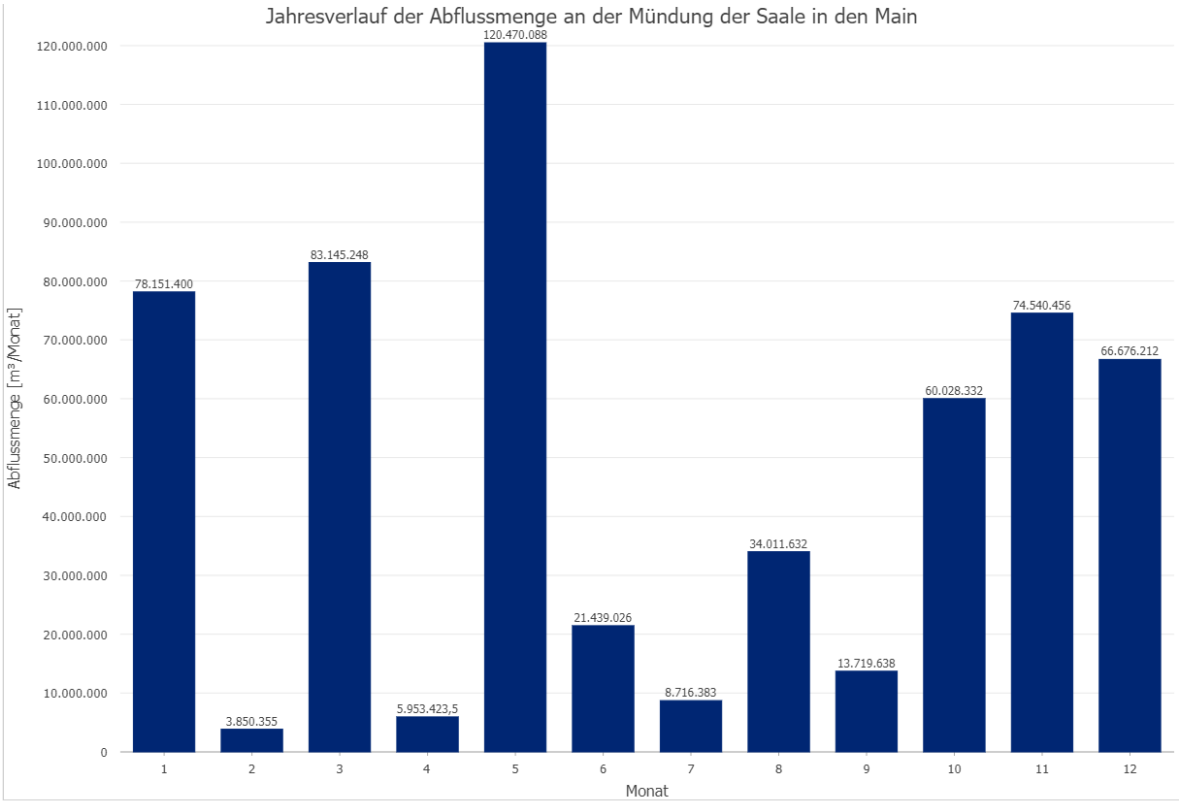

Abbildung 8: Jahresverlauf der Abflussmenge an der Mündung der Saale in den Main

4) Bisher wurde der Abfluss Q als durchschnittlicher Jahresabfluss bertrachtet, weshalb monatliche Schwankungen aufgrund von Trocken- oder Feuchtezeiten nicht in dem Ansatz berücksichtigt werden konnten. Da die Fließgeschwindigkeit als Abfluss geteilt durch die Querschnittsfläche definiert ist, gilt, je größer der Abfluss, desto höher ist die Fließgeschwindigkeit. Des Weiteren gilt, dass Eliminationsprozesse eine gewisse Zeit benötigen und somit von der Traveltime abhängig sind. Demnach findet bei einer verkürzten Traveltime ein geringerer Abbau der Bakterien statt als im Vergleich zu einer längeren Traveltime. Insgesamt erschließt sich daraus, dass bei einer höheren Abflussmenge eine höhere Fließgeschwindigkeit vorliegt und in Folge dessen, ein verminderter Abbau der Bakterien stattfindet. Vereinfacht angenommen würde die Anzahl an Bakterien in einem trockenen Monat mit geringer Abflussmenge demnach geringer sein, als in einem feuchten Monat mit einer hohen Abflussmenge. Auf Basis dieser Grundlagen wäre der Bezug direkt porportional.
Was hierbei jedoch nicht berücksichtigt wird, ist, dass in einem feuchten Monat mit ausgiebigen Niederschlägen auch mehr organisches Material abgetragen wird. Die

Masse an eingetragenem organischen Material wäre dann nicht mehr nur von der Landnutzung und insbesondere von der Tierhaltung innerhalb des Einzugsgebietes abhängig, sondern auch von dem Erosionsabtrag. Insgesamt hätte die Abtragung und Auswaschung von organischem Material vermutlich einen größeren Einfluss auf die Bakterienfracht, da der Eintrag weitaus größer ist als der Abbau. In diesem Fall, wäre der Bezug ebenfalls direkt porportional.
Im Vergleich zur Runoff Grafik würde die Grafik für die Bakterienfracht aufgrund des direkt porportionalen Bezuges etwa ähnlich aussehen, nur auf einer anderen Maßstabsskala, da die Frachten in cfu/a angegeben werden.
Die Bakterienkonzentration ist abhängig von der Wassermenge, also dem Runoff. Je größer die Abflussmenge ist, desto geringer ist die Bakterienkonzentration. Die Grafik würde dementsprechend genau andersherum aussehen.

5) Darstellung der Differenz von dem Monat November und einem Jahresdurchschnittsmonat:

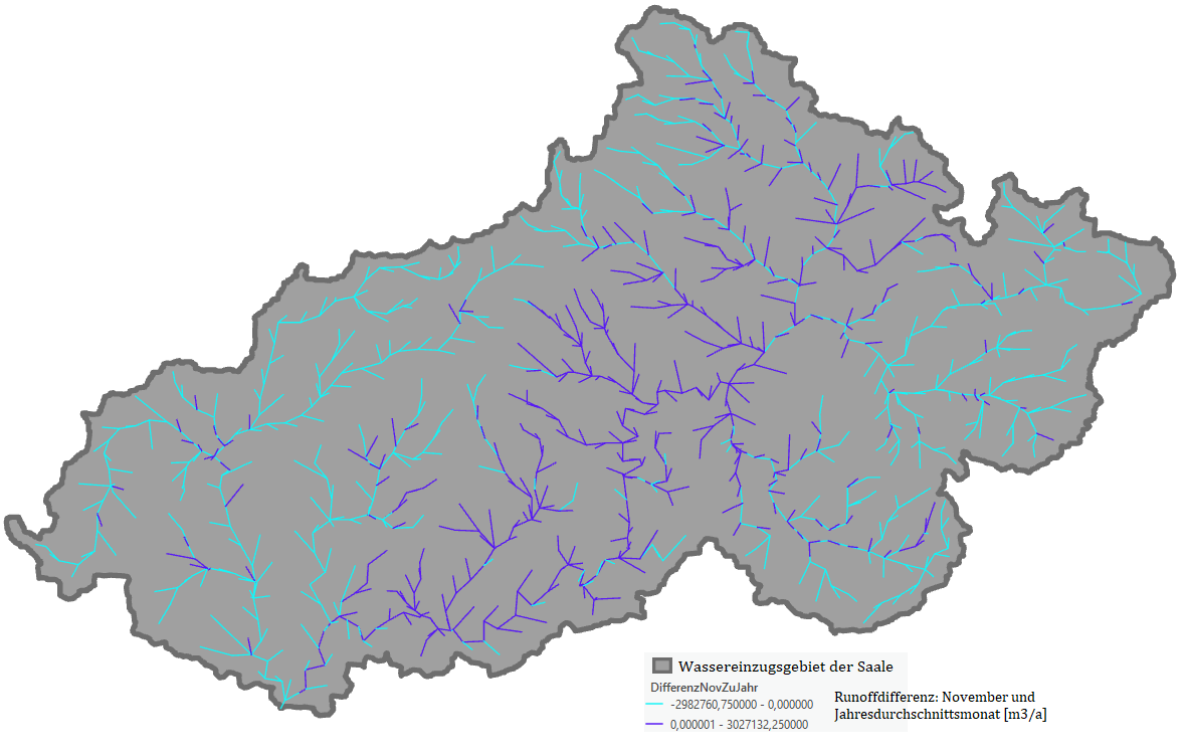

Abbildung 9: Differenz des Runoffs: Monat November und Jahresdurchschnittsmonat

Code/4_5.py

```
# 5a), Berechnung des Parameters S
S = Divide(Minus(25400,Times(254,"GCN250_ARCII_Saale_GK4.tif")),"GCN250_ARCII_Saale_GK4.tif")

# Parameter fuer Resampling bestimmen
# Wichtig ist es ein snapRaster zu setzen, da ansonsten das Resampling spaeter nicht funktioniert
arcpy.env.snapRaster = "CatchID"
cellSize = arcpy.sa.Raster(arcpy.env.snapRaster).meanCellHeight

# 5b) Laden des Multiband-Raster der Niederschlaege, extrahieren der einzelnen Monatsraster
# und Berechnung des Runoff pro Zelle [mm/month] nach Formel
monthlyRasters = arcpy.sa.Raster("HYRAS_2019_Saale_months_GK4.tif", True).getRasterBands()
for i in range(1,13):
    # Extrahieren eines Bands fuer einen Monat
    # monat = arcpy.ia.ExtractBand("HYRAS_2019_Saale_months_GK4.tif", [i])
    monat = monthlyRasters[i-1]
    # Berechnung des Runoffs Pe nach gegebener Formel in [mm/month]
    monat = ((monat - 0.2 * S) **2) / (monat + 0.8 * S)
    # Resampeln
    monat = arcpy.management.Resample(monat, "monat_res", cellSize, "BILINEAR")
    # Umrechnung in [m3/a]
    monat = Times(Times(Divide("monat_res", 1000), cellSize), cellSize)
    # Tabellen fuer die Catchment Aggregation berechnen
    ZonalStatisticsAsTable("CatchID", "Value", monat, ws +"\\PeCatchmentMonat" + str(i))

# 5c) Man kann leider nicht einfach so ein schoenes Balkendiagramm von der benoetigten Zeile erstellen, da ansonsten der Rest der Tabelle sonst
# ebenfalls mit in das Diagramm gepackt wird. Mit nur einer Zeile selektiert funktioniert es leider auch nicht. Deshalb nun Erstellung einer neuen
# Tabelle mit zwei Spalten (Monat und Abflussmenge), welche nur die eine Zeile fuer den Abflusslink beinhalten soll.
arcpy.management.CreateTable(ws, "Jahresverlauf")

# Spalte fuer Monate hinzufuegen
arcpy.management.AddField("Jahresverlauf", "Monat", "SHORT")
arcpy.management.AddField("Jahresverlauf", "Abflussmenge", "FLOAT")

# Die eine gesuchte Zeile in der Tabelle schema_link suchen und benoetigte Werte fuer die 12 Monate merken
fields = ["DownLinkID", "P_Monat1", "P_Monat2", "P_Monat3", "P_Monat4", "P_Monat5", "P_Monat6", "P_Monat7", "P_Monat8", "P_Monat9", "P_Monat10", "P_Monat11", "P_Monat12"]
cursor = arcpy.da.SearchCursor("schema_link", fields)

for i in cursor:
    if i[0] == -2:  # Auslasslinkzeile suchen
        gesuchteZeile = i # Zeile merken
        break   # Abbrechen, da das Gesuchte bereits gefunden wurde

# 12 neue Zeilen erstellen mit den Zahlen 1 - 12 fuer jeden Monat
fields = ["Monat", "Abflussmenge"]
cursor = arcpy.da.InsertCursor("Jahresverlauf", fields)
for i in range(1, 13):
    # Abflussmenge reinschreiben
    cursor.insertRow((i, gesuchteZeile[i]))

# cursor object loeschen
del cursor

# 5e) Berechnen der Differenz
arcpy.CalculateField_management("schema_link", "DifferenzNovZuJahr", "!P_Monat11! - (!Runoff!/12)", "PYTHON3", "", "FLOAT")
```

5 Geoprocessing

5.1 Python-Toolboxen kennen lernen

2) Die Klasse Toolbox stellt eine Art Werkzeugkasten dar. Innerhalb der init Methode der Klasse Toolbox findet die Initialisierung der Klasse statt. In einer Liste wird festgelegt, welche Werkzeuge in dieser Toolbox vorhanden sind (hier nur das WATER_tool). Außerdem werden Eigenschaften wie Alias und Label definiert. Der Name der Toolbox wird über den Namen der .pyt Datei bestimmt.
Die Klasse WATER_tool stellt das erste und einzige Tool in der Toolbox dar. In der init Methode für die Initialisierung der Klasse WATER_tool wird das überschreiben von Dateien erlaubt sowie ein Label und eine kurze Beschreibung gesetzt. Darüber hinaus verfügt die Klasse WATER_tool über eine getParameterInfo Methode, welche implizit vor dem Aufruf der execute Methode aufgerufen wird. In der getParameterInfo Funktion wird ein Eingabefeld für die Tabelle mit den Schema Links definiert. Dabei werden der Displayname, der Name, der Datentyp, der parameterTyp (required oder unrequired) und die Direction (Input oder Output) festgelegt. Als Rückgabewert liefert diese Funktion eine Liste der zuvor definierten Parameter zurück. Zuletzt besitzt die Klasse WATER_tool noch die execute Funktion. Diese ist der Kern des Tools und in ihr wird definiert, was das Tool kann und wie es funktioniert. Die execute Funktion erhält als zweiten Übergabeparameter die von der init Methode zurückgegebene Liste von definierten Parametern, sodass über die Attribute .value oder .valueAsText auf die einzelnen Parameter zugegriffen werden kann.

3) In der Klasse WATER_tool muss die execute Funktion erweitert werden, damit die über arcpy.AddMessage() hinzugefügten Nachrichten erscheinen.

5.2 Implementierung des WATER-Modells

3) Es gibt drei verschiedene Cursorarten, die danach unterschieden werden, was sie beim sequentiellen Durchlaufen der Daten mit den Daten machen können:

   - Searchcursor: zum Lesen von Daten
   - Updatecursor: zum Löschen oder Aktualisieren von Daten
   - Insertcursor: zum Einfügen von Daten

   Für das Ausgeben der HydroID und der SEQ_NR wird ein Searchcursor benötigt. Als Parameter erhält dieser die Tabelle mit den Schema Links, eine Liste der Spaltennamen, welche die HydroID und die SEQ_NR behinhalten, sowie eine SQL Clause, welche angibt, dass die Zeilen aufsteigend nach der SEQ_NR sortiert werden sollen.

5) Das Problem bei dieser Form der Darstellung der Nachbarschaftsbeziehungen besteht darin, dass es nicht möglich ist, bei einer Aufteilung eines Flusses in Richtung

flussabwärts (z.B. durch ein Hindernis oder so etwas wie eine Insel) die Frachtausleitung an mehrere Unterliegersegmente zu verteilen. Denn die Ausleitung wird immer nur an einen Link weitergegeben und es befindet sich für jeden Link nur eine DownLinkID in der Schema Link Tabelle, wohingegen bei einer abwärtsgerichteten Spaltung eines Flusses mehrere DownLinkIDs für einen Link oder eine andere Struktur benötigt werden würden. Am Auslasspunkt des Gewässers wird eine Fracht von 0,000165549968434 g/s berechnet.

6) Am Auslasspunkt wirt eine Konzentration von 5,78888025972457 ng/L erreicht. Die höchste Konzentration im Flussnetz beträgt 103,5 ng/L und die geringste Konzentration (ungleich 0) hat einen Wert von 0,684965171903786 ng/L.

Code/5.py

```
import arcpy
import math

# Toolbox definieren
class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""
        self.label = "GMI student"
        self.alias = ""

        # List of tool classes associated with this toolbox
        self.tools = [WATER_tool]

# erste Funktion der Toolbox, siehe Liste in zeile 12
class WATER_tool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "WATER_tool"
        self.description = "Student Version of the WATER Tool."
        arcpy.env.overwriteOutput = True # allow the override of old files

    # Diese Funktion wird implizit vor dem Aufruf von execute aufgerufen und
    # definiert ein eingabefeld als Parameteruebergabe.
    def getParameterInfo(self):
        """Define parameter definitions"""
        # schema links
        param_links = arcpy.Parameter(
            displayName="Schema links",
            name="in_schema_links",
            datatype="DEFeatureClass",
            parameterType="Required",
            direction="Input")

        # 2a) Erweiterung um drei weitere Parameter, dabei auf die Datentypen
        # der Spalten in der Tabelle achten
        # Aggregierte Abbaurate als Input vom Typ GPDouble:
        param_aggreAbbau = arcpy.Parameter(
            displayName = "Aggregierte Abbaurate",
            name = "in_aggreAbbau",
            datatype = "GPDouble",
            parameterType = "Required",
            direction = "Input"
        )

        # Spaltenname fuer Frachteinleitung pro Segment als GPString:
        param_spaltennameFrachtein = arcpy.Parameter(
            displayName = "Spaltenname fuer die Frachteinleitung pro Segment",
```

```
49              name = "in_spaltennameFrachtein",
50              datatype = "GPString",
51              parameterType = "Required",
52              direction = "Input"
53          )
54
55          # Spaltenname fuer das Zielfeld der Berechnung (Frachtausleitung) vom Typ GPString:
56          param_spaltennameFrachtaus = arcpy.Parameter(
57              displayName = "Spaltenname fuer die Frachtausleitung",
58              name = "in_spaltennameFrachtaus",
59              datatype = "GPString",
60              parameterType = "Required",
61              direction = "Input"
62          )
63
64          # params wird dann fuer die execute funktion als parameters uebergeben
65          params = [param_links, param_aggreAbbau, param_spaltennameFrachtein,
66              param_spaltennameFrachtaus]
67          return params
68
69
70      def execute(self, parameters, messages):
71          """The source code of the tool."""
72          # 1c) Nachricht in View Details ausgeben
73          arcpy.AddMessage("Starten des Tools Water_tool.")
74
75          # 2b) Ausgeben der eingegebenen Parameter
76          schemaLinks = parameters[0].valueAsText
77          abbaurate = parameters[1].value
78          lein = parameters[2].valueAsText
79          load = parameters[3].valueAsText
80
81          arcpy.AddMessage(schemaLinks)
82          arcpy.AddMessage(abbaurate)
83          arcpy.AddMessage(lein)
84          arcpy.AddMessage(load)
85          arcpy.AddMessage("------------------------------------------------")
86
87          # 2c) Vor der Ein/Ausgabe muessen die Attribute SEQ_NR aufsteigend sortiert werden
88          # die Attribute HydroID und SEQ_NR eines jeden Segments aufsteigend sortiert
89          # nach der SEQ_NR ausgeben
90          zeilen = arcpy.da.SearchCursor(parameters[0].valueAsText, ["SEQ_NR", "HydroID"],
91              sql_clause=(None, "ORDER BY SEQ_NR ASC"))
92          for i in zeilen:
93              arcpy.AddMessage("SEQ_NR: {0}, HydroID: {1}".format(i[0], i[1]))
94          arcpy.AddMessage("------------------------------------------------")
95
96          # 2d) Frachtberechnung:
97          # UpdateCursor zum aendern der einzelnen Felder in den Zeilen verwenden,
98          # wichtig wieder an die Sortierung denken!
99          zeilen = arcpy.da.UpdateCursor(schemaLinks, ["Travel_time_d", lein, load, "SEQ_NR",
100             "HydroID", "DownLinkID"], sql_clause=(None, "ORDER BY SEQ_NR ASC"))
101
102         # Abbaurate von 1/h in 1/s umrechnen, diese ist fuer alle Zeilen gleich
103         abbau = abbaurate / 60 / 60
104
105         # 2e) Initialisierung des NextDownDictonarys
106         # Dictonaries bestehen aus Key-Value Paaren
107         # ein bestimmter key kann nur maximal einmal im Dictonary vorhanden sein
108         NextDownDictonary = {}
109
110         for i in zeilen:
111             # 2e) Dictonary, i[4] = HydroID, i[5] = NextDownID = DownLinkID
112             # pruefen, ob der Link (HydroID) bereichts im Dictonary vorhanden ist
113             if i[4] in NextDownDictonary:
114                 # falls ja, wird der zu dem Schluessel HydroID gehoerende Value sich gemerkt
115                 add_emission = NextDownDictonary[i[4]]
116             else:
```

```
117                 # wenn der Link nicht im Dictonary ist, wird die emission auf 0 gesetzt
118                 add_emission = 0
119
120             # wenn die eigene Frachteinleitung eines Links Null ist, dann erhaelt er
121             # nur die Oberfracht (Fracht aus oberen Segmenten)
122             if i[1] is None:
123                 einleitung = add_emission
124             # wenn die eigene Frachteinleitung eines Links vorhanden ist, dann wird
125             # diese zur Oberfracht dazu addiert
126             else:
127                 einleitung = i[1] + add_emission
128
129             # wenn die Traveltime Null ist, dann ist die ausleitung gleich der
130             # einleitung und es findet kein Abbau statt
131             if i[0] is None:
132                 ausleitung = einleitung
133             # wenn eine Traveltime vorhanden ist, dann wird der Abbau berkuecksichtigt
134             else:
135                 # Traveltime von 1/tag in sekunden umrechnen
136                 travel = i[0] * 24 * 60 * 60
137                 ausleitung = einleitung * math.exp(- abbau * travel)
138
139             # die Ausleitung (Load_g_s) des Links dieser Zeile updaten
140             i[2] = ausleitung
141
142             # pruefen, ob bereits Frachten von dem Oberliegersegment in dem
143             # Unterliegersegment vorkommen
144             if i[5] in NextDownDictonary:
145                 # falls ja, wird die von diesem Link berechnete Ausleitungsfracht zu den
146                 # bekannten Frachten des Oberliegersegments aus dem Dictonary dazu addiert
147                 NextDownDictonary[i[5]] = NextDownDictonary[i[5]] + ausleitung
148             else:
149                 # falls noch keine Frachten von dem Oberliegersegment vermerkt sind, wird
150                 # nur die Ausleitungsfracht des Segments in das Dictonary eingetragen
151                 NextDownDictonary[i[5]] = ausleitung
152
153             # dem cursor Objekt sagen, dass es die Zeile speichern/updaten soll
154             zeilen.updateRow(i)
155
156         # 5f) Berechnen der Konzentration im neuen Feld Konzentration_ng_L:
157         # zuerst die Ausleitung von gramm/s in nanogramm/s, dann in Jahr (damit sich beim
158         # Teilen das Jahr herausrauskuerzt) umrechnen, bei runoff: 1 m3 = 1000 Liter
159         arcpy.management.CalculateField(schemaLinks, "Konzentration_ng_L",
160             "(!Load_g_s! * 1000 * 1000 * 1000 * 3600 * 24 * 365) / (!Runoff_m3_a! * 1000)",
161             field_type = "DOUBLE")
```

6 Urban Growth

Bei dem Vergleich des Anfangszustands mit Zeitschritt 1 ist zu erkennen, dass die Anzahl an urbanen Zellen innerhalb eines Zeitschritts nur leicht gewachsen ist (siehe Darstellung der Ergebnisse des Zellulären Automaten). Im weiteren Verlauf vergrößert sich die urbane Fläche in jedem Zeitschritt um ein paar Zellen. Insbesondere im oberen Teil des Ausschnitts von Zeitschritt 7 ist festzustellen, dass viele der ungeschützten Zellen zu urbanen Zellen geworden sind. Im unteren Teilausschnitt hat sich eine Treppenform gebildet, bei welcher die an die urbane Fläche grenzenden ungeschützten Zellen maximal 3 urbane Zellen als Nachbarn haben. Dadurch kann sich die urbane Fläche im unteren Teil nicht weiter ausbreiten.

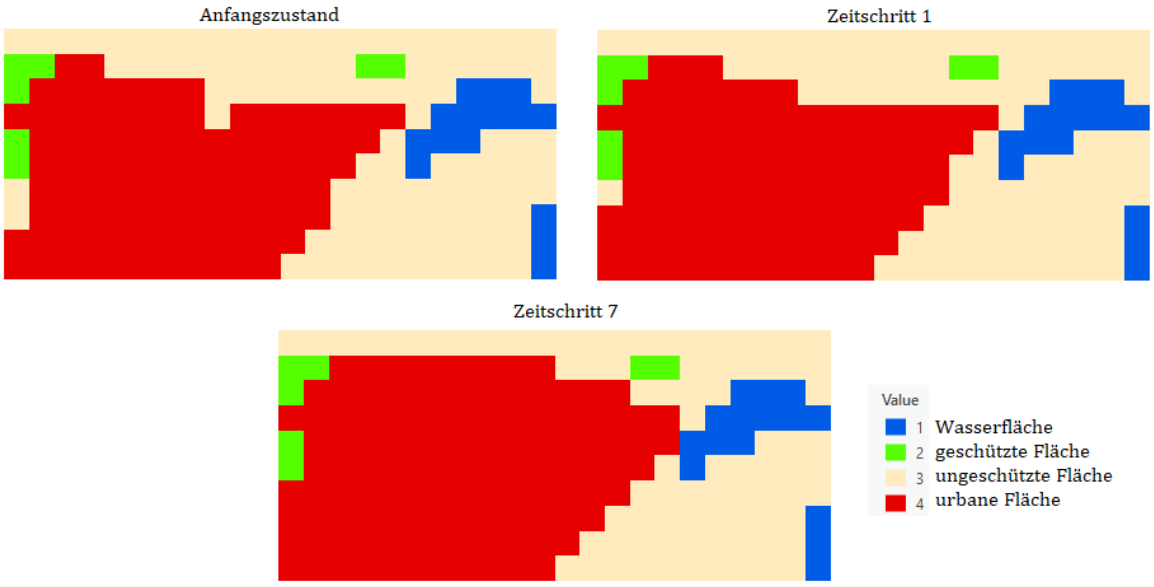

Abbildung 10: Darstellung der Ergebnisse des Zellulären Automaten

Eine Änderung der Regel 2 mit einer Nachbarschaft von mindestens 3 urbanen Nachbarzellen führt zu einem schnelleren Wachstum der Stadt, da diese Bedingung leichter zu erfüllen ist, als die Bedingung mit mindestens 4 urbanen Nachbarzellen.

Mögliche Modellverfeinerungen:

- Hagneigung: auf steilen Hängen ist es nur mit hohem Kostenaufwand möglich eine Bebauung vorzunehmen. Deshalb wäre eine Regel, die besagt, ab welchem Steigungswinkel eine Bebauung aus kostentechnischen Gründen nicht mehr lohnenswert wäre, eine mögliche Verbesserung des Modells.

- Bodeneigenschaften: verschiedene Böden haben sehr unterschiedliche Eigenschaften, die eine wichtige Rolle spielen könnten. Denn zum Beispiel sind organische Böden mit einem hohen Humusgehalt als Bauuntergrund nahezu ungeeignet, während dahingegen anorganische Böden (Ton, Kies, ...) viel besser als Bauuntergrund verwendet werden können.

- Infrastruktur: Für viele Menschen sind Anbindungen an den Fern- oder Nahverkehr sehr wichtig, aber auch andere Aspekte, wie z.B. eine gewisse Nähe zu Versorgungs- oder Bildungseinrichtungen können bei der Wahl des Ansiedlungsortes einen gewissen Einfluss haben.

- variable Nachbarschaften: die in einem Modell gewählte Nachbarschaft hat große Auswirkungen auf das Ergebnis des Modells. In gewissen Fällen kann es sinnvoll sein, die Nachbarschaft variabel zu halten. Insbesondere an den Ecken und Kanten eines Ausschnitts stellt sich die Frage, wie diese zu behandeln sind, da diese Zellen nicht über die volle Anzahl an Nachbarzellen im Vergleich zu innen liegenden

Zellen verfügen. Zudem könnte bei besonders attraktiven Flächen eine kleinere Nachbarschaft bereits ausreichend sein, da der Attraktivitätsfaktor stärker in die Gewichtung mit einfällt als das Einhalten einer immer zu gleich großen Nachbarschaft. Umgekehrt könnte bei eher weniger attraktiven Flächen eine größere Nachbarschaft gewählt werden, um darzustellen, dass eine Ansiedlung auf dieser Fläche weniger wahrscheinlich ist.

- Baurecht: In Niedersachsen existieren zum Beispiel gewisse Bestimmungen für die Flächeninanspruchnahme und Bodenversiegelung, die im Baurecht hinterlegt und zu berücksichtigen sind.

- Gewichtung: Bei den Regeln wäre es eventuell sinnvoll mit einer Art Gewichtung zu arbeiten. Beispiel: Eine mögliche Regel zur Hangneigung könnte besagen, dass ab einer Hangneigung von 30 Grad oder mehr, eine Versiegelung bzw. Bebauung der Fläche nicht mehr möglich wäre. Diese Regel könnte etwas feingranularer gestaltet werden, da bei einer Grundstücksauswahl zwischen einer Fläche mit 5 Grad Hangneigung und einer Fläche mit 20 Grad Hangneigung eine Bebauung laut Regel zwar auf beiden Flächen möglich wäre, aber dennoch die Fläche mit 5 Grad Hangneigung zu bevorzugen wäre. Auf ähnliche Art könnten auch die einzelnen Regeln untereinander stärker oder weniger stark mit ins Gewicht fallen, da nicht immer jeder Faktor gleichstark zu gewichten ist.

Code/6.py

```
import arcpy
import math
from arcpy.sa import *
ws = arcpy.env.workspace

# 1a) Ausschnitt von Australien definieren:
# vier Eckpunkte festlegen, Koordinaten koennen aus der Karte abgelesen werden,
# Nord und Ost sind positive Werte, Sued und West sind negative Werte
polyPoints = [arcpy.Point(152.38, -27.44), arcpy.Point(152.44, -27.44),
              arcpy.Point(152.44, -27.41),arcpy.Point(152.38, -27.41)]
# extrahieren des Polygons
extPolygonOut = ExtractByPolygon("globcov2006_aus.tif_Band_1",polyPoints, "INSIDE")

# 1b) nur die vier Zustaende des CA sollen in dem Raster enthalten sein
vierZustaende = Lookup(extPolygonOut, "RECL_VALUE")
# speichern des Rasters ueber die 4 Zustaende
vierZustaende.save(ws + "\\vierZustaende")

# 1c) Cellulaerer Automat
altesRaster_ = Raster(vierZustaende)
zeitschritte = 7

for t in range(1, zeitschritte + 1):
    # neues Raster erstellen, welches als binaere Maske wirkt,
    # eine Zelle darin ist entweder 0 (= nicht urban) oder 1 (= urban)
    binaereMaske_ = Con(altesRaster_, 1, 0, "Value = 4")
    # focalStatistics: berechnet fuer jede Eingabezellenposition eine Statistik der Werte
    # innerhalb einer angegebenen Nachbarschaft. Wenn man diese Funktion mit der Maske
    # aufruft, werden somit alle urbanen Felder um das betrachtete Feld aufsummiert und in
    # einem neuen Raster wird pro Zelle die Anzahl der benachbarten urbanen Felder ge-
    # speichert, da alles nicht urbane 0 in der Maske ist. 3x3 Quadrat = Moore Nachbarschaft
    anzahlUrbaneNachbarn_ = FocalStatistics(binaereMaske_,
        NbrRectangle(3, 3, "CELL"), statistics_type = "SUM")
```

```
34      # wenn mehr als 4 Nachbarn urban sind UND die Zelle vom Typ 3 (ungeschuetzt) ist, dann
35      # wird das Feld auf 4 gesetzt, alle anderen Felder behalten den vorherigen Zustand
36      neuesRaster = Con(((anzahlUrbaneNachbarn_ >= 4) & (altesRaster_ == 3)), 4, altesRaster_)
37      # speichern des gerade berechneten Zeitschritts
38      neuesRaster.save(ws + "\\Zeitschritt" + str(t))
39      # altes und neues Raster fuer den naechsten Zeitschritt tauschen
40      altesRaster_ = neuesRaster
```

7 Urban Sprawl

Um im Vergleich zu dem vorherigen CA eine realistischere Modellierung vorzunehmen, werden nun weitere Ebenen hinzugenommen. Als Basisebene des CA dient die Landnutzung Australiens aus dem Jahr 2014 [vgl. Lymburner et al. 2015]. Zuerst wird eine Reklassifizierung der Landnutzung vorgenommen (siehe Reklassifizierung der Landnutzung). Dabei werden Weide-, Gras-, Agrarflächen und Buschland zusammengefasst, da diese in der Regel leicht zu versiegeln sind. Eine weitere Klasse enthält verschiedene Waldarten, da vor einer Bebauung erst eine Abholzung erfolgen müsste. Feuchtgebiete stellen eine Klasse für sich dar, da bei einer Bebauung mit speziellen Gefahre, wie zum Beispiel einer Absackung des Baugrunds, zu rechnen ist. Des Weiteren werden Seen und Dämme in einer Klasse zusammengefasst, da eine Versiegelung dieser eine Trockenlegung und besonders hohe Mehrkosten voraussetzt. Die Klasse für urbane Flächen bleibt weiterhin erhalten. Zuletzt stellen Minen und Steinbrüche eine eigene Klasse dar, denn diese werden in der Regel für die Gewinnung von Rohstoffen verwendet und stehen somit nicht als mögliche Versiegelungsfläche zur Verfügung.

Name	Klasse	Reklassifizierung
Mines and Quarries	1	5
Urban areas	35	4
Lakes and dams	3	1
Salt lakes	4	1
Irrigated cropping	5	3
Rain fed cropping	8	3
Irrigated pasture	6	3
Rain fed pasture	9	3
Irrigated sugar	7	3
Rain fed sugar	10	3
Wetlands	11	6
Alpine meadows	15	3
Open Hummock Grassland	16	3
Closed Tussock Grassland	14	3
Open Tussock Grassland	18	3
Scattered shrubs and grasses	19	3
Dense Shrubland	24	3
Open Shrubland	25	3
Closed Forest	31	2
Open Forest	32	2
Woodland	34	2
Open Woodland	33	2

Tabelle 1: Reklassifizierung der Landnutzung

Als zusätzliche Ebene wird die Hangneigung aus einem SRTM-Daten abgeleitet [vgl. NASA Shuttle Radar Topography Mission (SRTM) 2013], da eine Versiegelung von Flächen mit einem geringen Neigungswinkel einfacher ist, als bei Flächen mit einem hohen Hangneigungswinkel.
Zuletzt sollen außerdem auch Schutzgebiete, wie zum Beispiel Reservoire oder Nationalparks, berücksichtigt werden, da diese unter Naturschutz stehen und nicht versiegelt werden dürfen [vgl. Australian Government - Department of Agriculture, Water and the Environment 2017].

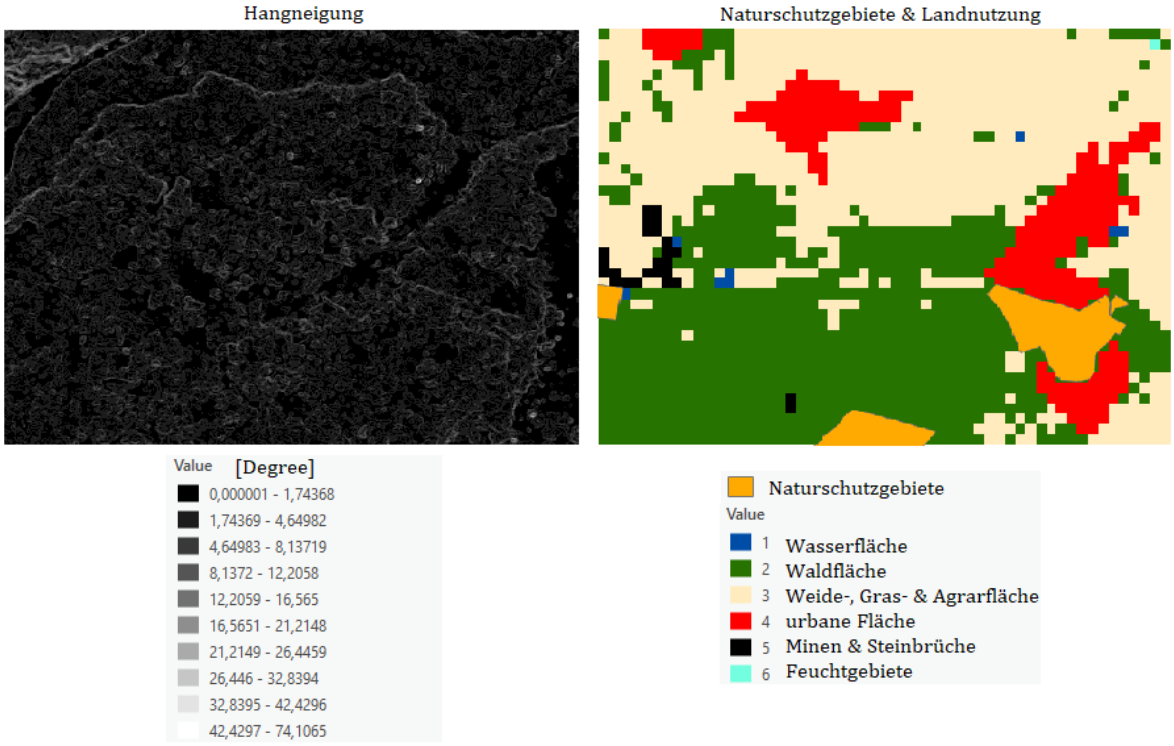

Abbildung 11: Darstellung der drei betrachteten Ebenen

Der CA operiert nach folgendem Regelwerk:
Eine Zelle wird in eine urbane Zelle umgewandelt, wenn sie eine eine Hangneigung von kleiner gleich 8,5 Grad, die Zelle zu keinem Naturschutzgebiet gehört und eine der folgenden Bedingungen zutrifft:

1) Die Zelle gehört der Klasse Weide- Gras- & Agrarfläche und hat in einer rechteckigen 4x4 Nachbarschaft mindestens 11 urbane Nachbarn.

2) Die Zelle gehört der Klasse Waldfläche an und hat in einer rechteckigen 3x3 Nachbarschaft mindestens 6 urbane Nachbarn.

3) Die Zelle gehört der Klasse Feuchtgebiete an und hat in einer rechteckigen 3x3 Nachbarschaft mindestens 7 urbane Nachbarn.

4) Die Zelle gehört der Klasse Wasserfläche an und hat in einer rechteckigen 3x3 Nachbarschaft mindestens 8 urbane Nachbarn.

Für die leicht bebaubare Klasse Weide-, Gras- & Agrafläche wurde eine 4x Nachbarschaft und eine Mindestanzahl von 9 urbanen Nachbarn gewählt, um zum einen abzubilden, dass bei der Entstehung von Siedlungsflächen häufig ein größere Fläche in Betracht gezogen wird, um dort nicht nur ein bis zwei Häuser zu bauen, sondern gleich eine ganze Wohnsiedlung. Zum Anderen wird dadurch der stärkere Zuwachs zu bereits größeren Siedlungsflächen verdeutlicht (eine Großstadt kann schneller wachsen als eine Kleinstadt), da es für sehr kleine Siedlungsflächen trotz der größeren 4x4 Nachbarschaft

schwieriger ist, 9 urbane Nachbarn aufzuweisen. Für Waldfläche, Feuchtgebiete und Wasserfläche wird je eine 3x3 Nachbarschaft mit einer relativ hohen Mindestanzahl von urbanen Nachbarzellen festgelegt. Dies ist damiz zu begründen, dass bei einer Versiegelung dieser Flächen ein Mehrkostenaufwand entsteht. Die Abholzung eines Waldgebietes ist einfacher als die Trockenlegung oder oberflächliche Bebauung einer Wasserfläche. Deshalb erfodert eine Waldfläche mindestens 6 urbane Nachbarzellen, Feuchtgebiete mindestens 7 und eine Wasserflächen mindestens 8 urbane Nachbarzellen, bevor die Zelle zu einer urbanen Zelle werden kann. Des Weiteren verhindert die Regel bezüglich der Naturschutzgebiete eine Versiegelung dieser, da diese rechtlich geschützt sind. Die Regel bezüglich der Hangneigung soll umsetzen, dass ab einer Hangneigung von größer als 8,5 Grad die Fläche nicht in eine urbane Zelle umgewandelt wird, da für die Versiegelung einer solchen Fläche eine kostenaufwendige Terrassierung zuvor durchgeführt werden müsste.

Die Ergebnisse (siehe Darstellung der Ergebnisse des Zellulären Automaten) über 6 Zeitschritte zeigen, dass sich im Vergleich die größere urbane Fläche im Osten stärker ausbreitet als die drei kleineren Flächen. Zudem ist die östliche Siedlungsfläche in Richtung Süden durch ein Naturschutzgebiet beschränkt und kann sich deshalb nicht mit der südöstlichen urbanen Fläche zu einer noch größeren urbanen Fläche verbinden. Aufgrund des Naturschutzgebietes wächst die südöstliche Siedlung nicht. Die kleine Siedlungsfläche im Nordwesten wächst ebenfalls nicht, dies liegt an der gewählten Mindestanzahl von 9 urbanen Nachbarn in einem 4x4 Raster. Denn diese Regel ermöglicht nur den größeren Siedlungsflächen ein schnelleres Wachstum, während kleine urbane Flächen, die diese Mindestgröße nicht erfüllen, auch nicht wachsen können, da sie für eine Besiedelung als eher unattraktiv angesehen werden.

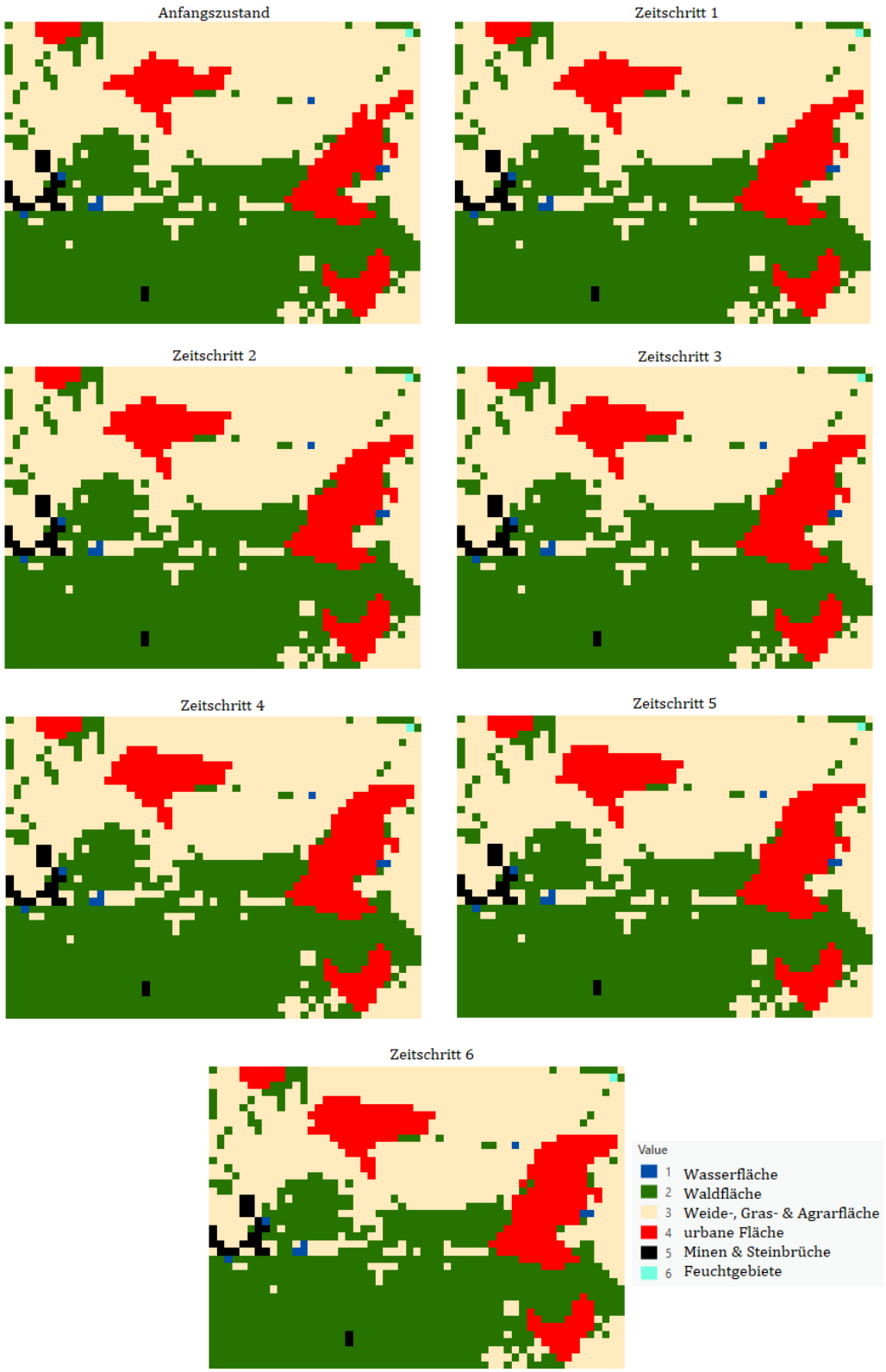

Abbildung 12: Darstellung der Ergebnisse des Zellulären Automaten

Das Modell könnte mit folgenden Ebenen noch weiter verfeinert werden:

1) Soziales Umfeld: Graph mit Bekanntschafts-/Freundschaftsbeziehungen (auf kleinerem Maßstab)

2) Infrakstruktur: Verkehrsanbindungen, Nähe zu Versorgungs- und Bildungseinrichtungen

3) Klima: nival, arid, semi-arid, humid (auf größerem Maßstab)

4) Gebiete mit einem gewissen Gefahrenpotential: Waldbrände in trockenen Gebieten, Überschwemmungsgefahren bei starken Niederschlägen oder in Gewässernähe, stärkere Luft- oder Umgebungsverschmutzungen (z.B. durch Kraftwerke oder industrielle Verbrennung), Naturkatastrophen (z. B. Erdbeben, Vulkanismus, Tsunamis, Tornados)

Code/7.py

```
import arcpy
import math
from arcpy.sa import *
ws = arcpy.env.workspace

# snapRaster defnieren, alle Raster sollen dann spaeter die gleiche Zellgroesse
# wie das Landnutzungsraster haben
arcpy.env.snapRaster = "DLCD_v2-1_MODIS_EVI_12_20130101-20141231.tif"
cellSize = Raster(arcpy.env.snapRaster).meanCellHeight

# Landnutzungsdaten vorbereiten: Auschnitt definieren, Sueden und Westen sind negativ
polyPoints = [arcpy.Point(150.6980128, -33.5824053), arcpy.Point(150.8270753, -33.5824053),
              arcpy.Point(150.8270753, -33.6749098), arcpy.Point(150.6980128, -33.6749098)]

# extrahieren des Auschnitts
extractedLandnutzung = ExtractByPolygon("DLCD_v2-1_MODIS_EVI_12_20130101-20141231.tif",
    polyPoints, "INSIDE")

# Mit Lookup nur die reklassifizierten 6 Klassen als Raster speichern
sechsZustaende = Lookup(extractedLandnutzung, "Reklass")


# Senken im DEM fuellen
DEM_senkenlos = Fill("output_SRTMGL1.tif")

# Hangneigung in Grad aus dem senkenlosen DEM berechnen
hangneigung = Slope(DEM_senkenlos, "DEGREE")

# Resampeln des Hagneigungsrasters auf die Zellgroesse des Landnutzungsrasters
hangneigungResampled = arcpy.management.Resample(hangneigung, "hangneigungResampled",
    cellSize, "BILINEAR")

# Ausschnitt fuer die Hangneigung extrahieren
extractedHangneigung = ExtractByPolygon(hangneigungResampled, polyPoints, "INSIDE")


# Mergen von Nichtschutzgebieten und Schutzgebieten, fuer ein lueckenloses Raster
arcpy.Merge_management(["capad", "Viereck"], "MergedSchutz")

# Schutzgebietelayer capad von 2014 in ein Raster ueberfuehren
schutzgebiete = arcpy.conversion.FeatureToRaster("MergedSchutz", "Schutz", "schutzgebiete",
    cellSize)

# Ausschnitt fuer die Schutzgebiete extrahieren
```

```
45 extractedSchutz = ExtractByPolygon(schutzgebiete, polyPoints, "INSIDE")
46
47
48 # Implementation des CA: die zu verwendenen Ebenen sind extractedSchutz, extractedHangneigung
49 # und sechsZustaende
50 altesRaster = Raster(sechsZustaende)
51 zeitschritte = 8
52
53 for i in range(1, zeitschritte):
54     # binaere Maske erstellen, die besagt ob eine Zelle urban(6) ist oder nicht urban ist
55     binaereMaske = Con(altesRaster, 1, 0, "Value = 4")
56     # Raster mit Anzahl berechneter Nachbarn fuer jede Zelle zuerst mit 3X3
57     anzahlUrbaneNachbarnDreiXDrei = FocalStatistics(binaereMaske, NbrRectangle(3, 3, "CELL"),
58         statistics_type = "SUM")
59     # und danach mit 4X4 Nachbarschaft
60     anzahlUrbaneNachbarnVierXVier = FocalStatistics(binaereMaske, NbrRectangle(4, 4, "CELL"),
61         statistics_type = "SUM")
62
63     # Regeln umsetzen: wann aendert sich eine zelle zu urban?
64     # 1. wenn im 4x4 mindestens 9 Nachbarzellen urban sind, wenn Zelle bebaubar (3) ist,
65     #    wenn die Hangneigung <= 8.5 ist, wenn es kein Naturchutzgebiet ist.
66     # 2. wenn im 3x3 mindestens 6 Nachbarzellen urban sind, wenn Zelle Wald (2) ist,
67     #    wenn die Hangneigung <= 8.5 ist, wenn es kein Naturchutzgebiet ist.
68     # 3. wenn im 3x3 mindestens 7 Nachbarzellen urban sind, wenn Zelle Feuchtgebiet (6) ist,
69     #    wenn die Hangneigung <= 8.5 ist, wenn es kein Naturchutzgebiet ist.
70     # 4. wenn im 3x3 mindestens 8 Nachbarzellen urban sind, wenn Zelle Wasser (1) ist,
71     #    wenn die Hangneigung <= 8.5 ist, wenn es kein Naturchutzgebiet ist.
72     neuesRaster = Con((extractedHangneigung <= 8.5) & (extractedSchutz == 0) & (((
       anzahlUrbaneNachbarnVierXVier >= 9) & (altesRaster == 3)) | ((
       anzahlUrbaneNachbarnDreiXDrei >= 6) & (altesRaster == 2)) | ((
       anzahlUrbaneNachbarnDreiXDrei >= 7) & (altesRaster == 6)) | ((
       anzahlUrbaneNachbarnDreiXDrei >= 8) & (altesRaster == 1))), 4, altesRaster)
73     # aktuellen Zeitschritt speichern
74     neuesRaster.save(ws + "\\zeitschritt" + str(i))
75     # auf naechsten Zeitschritt vorbereiten
76     altesRaster = neuesRaster
```

Literatur

Australian Government - Department of Agriculture, Water and the Environment: Collaborative Australian Protected Areas Database (CAPAD) 2014. (2017). URL: /http://www.environment.gov.au/fed/catalog/search/resource/details.page?uuid={1813C24E-42E7-4959-A09D-EFB9B7E6EF8E}
(letzter Abruf: 13.03.2022).

Lymburner, L.; Tan, P.; McIntyre, A.; Thankappan, M.; Sixsmith, J.: Dynamic Land Cover Dataset Version 2.1. (2015). URL: /https://ecat.ga.gov.au/geonetwork/srv/eng/catalog.search#/metadata/83868
(letzter Abruf: 13.03.2022).

NASA Shuttle Radar Topography Mission (SRTM): Shuttle Radar Topography Mission (SRTM) Global. (2013). URL: /https://doi.org/10.5069/G9445JDF
(letzter Abruf: 13.03.2022).