Dynamic Choropleth Maps - Using Amalgamation to Increase Area Perceivability: SUPPLEMENTARY MATERIAL

Liam McNabb
Visual & Interactive Computing Group
Swansea University
Swansea, Wales
661370@swansea.ac.uk

Robert S. Laramee
Visual & Interactive Computing Group
Swansea University
Swansea, Wales
r.s.laramee@swansea.ac.uk

Richard Fry
National Centre for Population Health and Wellbeing Research, Medical School
Swansea University
Swansea, Wales
r.j.fry@swansea.ac.uk

Algorithm 1 - Are polygons neighbors?

Input–
p_1 : polygon one
p_2 : polygon two

```
procedure isNeighbor(p1, p2)
    if isOverlapping( p1.boundingBox(),
                      p2.boundingBox() ) then
        return commonVertices(p1, p2)
    endIf
    return FALSE
```

Local Variables–
counter : number of matching vertices
$MIN = 2$: minimum number of matching vertices required to be neighbors

```
procedure commonVertices(p1, p2)
    counter = 0
    for i = 0; i < p1.length(); i++ do
        if p2.intersects(p1[i]) then
            counter++
            if counter ≥ MIN then
                return TRUE
            endIf
        endIf
    endFor (i)
    return FALSE
```

Desc: *Compares two polygons and tests for overlapping boundaries, $p_1 \cap p_2$. Returns true if the minimum number of common vertices are found.*

Algorithm 2 Contiguous Regions

Input– L_p : non-empty list of polygons
Output– $L_{islands}$: list of contiguous islands (or land masses)
Local Variables–
island: current island
neighborFound: flag designating if neighbor is part of existing island

```
procedure identifyContiguousRegions(L_p)
    // For each polygon
    while !L_p.isEmpty() do
        // Assume Island
        island = L_p.popFirst()
        // For each island
        for i = 0; j < L_islands.length(); i++ do
            // For each polygon on each island
            for j = 0; j < L_islands[i].length(); j++ do
                if isNeighbor(island, L_islands[i][j]) then
                    neighborFound = true
                    break
                endIf
            endFor (j)
            if neighborFound then
                island.appendList(L_islands[i])
                L_islands.removeIslandAt(i)
                i- -
            endIf
        endFor (i)
        L_islands.append(island)
    endWhile
    return L_islands
```

Desc: *Partitions a list of non-contiguous polygons into separate contiguous regions such as islands and land masses. A contiguous region has connected neighbors where no area is completely separated by water.*

Algorithm 3 Identify Boundary Range

Input–
start : starting index of shared boundary line
end : last index of shared boundary line
V_c : current polygon vertices in clockwise order
V_n : neighbor polygon vertices in clockwise order
Local Variables–
longestC: found by comparing distance between common vertices
common : list of found commonVertices

```
procedure IDENTIFYBOUNDARYRANGE(start, end, V_c, V_n)
    for int i = 0; i ¡ V_c.length(); ++i do
        if V_n.contains(V_c[i]) then
            common.append( V_c[i] )
        endIf
    endFor (i)
    longestC = longestSharedBoundaryChain( common, V_c )
    *end = V_c.indexOf(common[longestC])
    *start = V_c.indexOf(common[longestC.next()])
    return
```

Desc: *Identifies b_s for V_c with V_n as a neighbor. Required for parent node.*

Algorithm 4 Longest Shared Boundary Chain

Input–
common : list of found commonVertices
V_c : current polygon vertices in clockwise order

```
procedure LONGESTSHAREDBOUNDARYCHAIN( COMMON, V_c )
    if isLongestChain( V_c, &longest, V_c.indexOf(common.last()),
                       V_c.indexOf(common.first()) ) then
        longestIndex = common.length()-1
    endIf
    for i = 1; i ¡ common.length(); i++ do
        if isLongestChain( V_c, &longest, V_c.indexOf(common.at(i-1)),
                           V_c.indexOf(common.at(i)) ) then
            longestIndex = i
        endIf
    endFor (i)
    return longestIndex
```

New Input–
longestL: The current longest distance between two common vertices
currI: current index to test
nextI: next index to test
Local Variables–
length: length of current chain

```
procedure ISLONGESTCHAIN(V_c, LONGESTL, CURRI, NEXTI)
    length = nextI - currI
    if length ¡ 0 then
        length = length + current.size() - 1
    endIf
    if *longestL ¡ length then
        *longestL = length
        return true
    else
        return false
    endIf
```

Desc: *Identifies the longest absence of a common vertex. We can assume that this signifies the beginning and end points of b_s.*

Algorithm 5 Build Binary Tree

Input–
L_{contig} : contiguous list of polygon sorted by area
Local Variables–
neighborI : index of selected neighbor
p : parent node of two neighbor areas

```
procedure BUILDBINARYTREE(L_contig)
    if L_contig.length() > 1 then
        //Neighbor Selection
        neighborI = selectNeighbor(list) //neighbor of list.first()
        //Create Parent Area
        p = new Node()
        p.identifyVertices(L_contig.first(),L_contig.at(neighborI))
        p.setLeftChild(L_contig.first())
        p.setRightChild(L_contig.at(neighborI))
        L_contig.first().setParent(p)
        L_contig.at(neighborI).setParent(p)
        //Update Sorted List with Parent
        updateList(L_contig, p, i)
        return buildBinaryTree(L_contig)
    endIf
    // Base Case -¿ L_contig == 1
    return L_contig
```

Desc: *Builds hierarchy of polygons using a list of merge candidates recursively.*

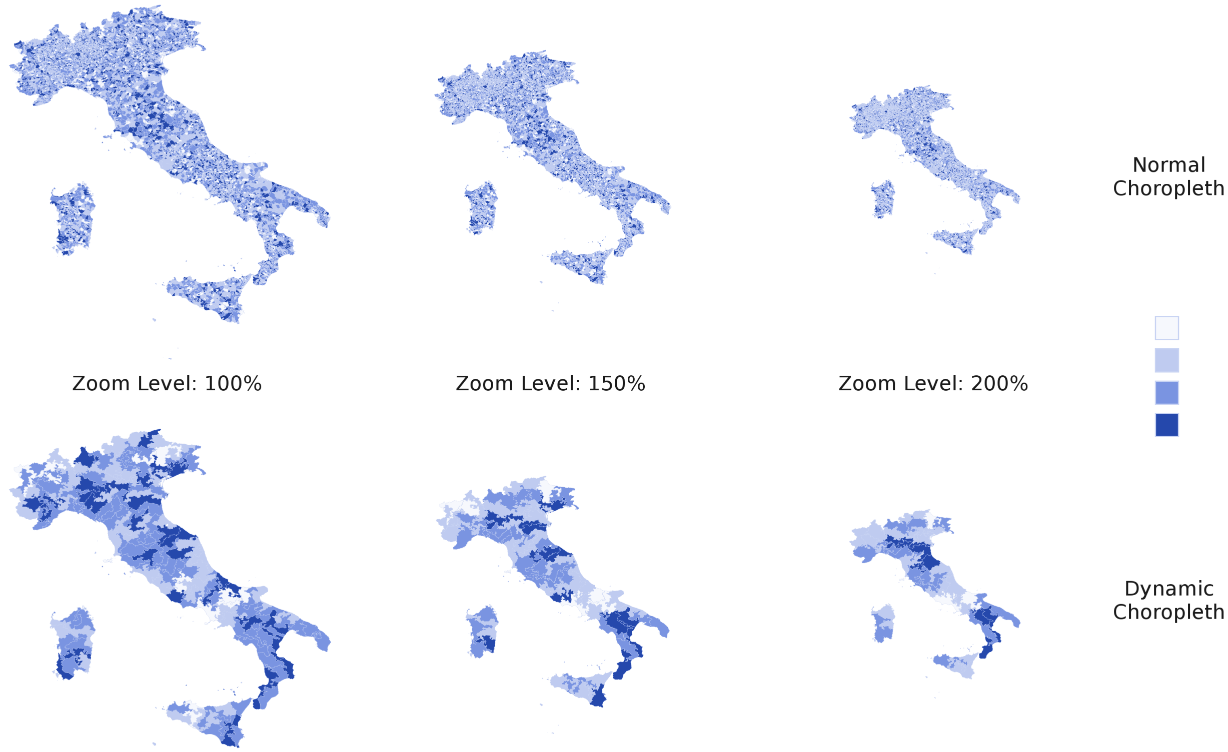

Fig. 1: An example of zooming out of Italy's administrative units where $m = 1\%$.

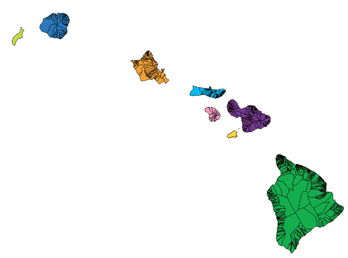

Fig. 2: Example of the contiguous regions procedure applied to the Ahupua'a boundaries of the state of Hawaii [2]. There are 10 visible contiguous islands each with their own color.

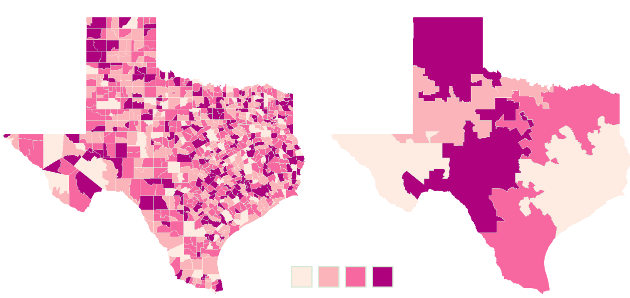

Fig. 3: Example of derived parent area's from original unit-areas. The example uses the State of Texas and shows multiple boundary merges [3] so that m is 15%. Our example uses a color palette from colorbrewer [1].

References

[1] C. A. Brewer, "Colorbrewer," 2017, accessed 2017/08/10. [Online]. Available: http://colorbrewer2.org/

[2] Hawaii, *hawaii.gov - Ahupua'a Boundaries (Historic Land Divisions)*, State of Hawaii - Office of Planning, accessed: 2017-9-26. [Online]. Available: http://planning.hawaii.gov/gis/download-gis-data/

[3] US-Texas, *Data.gov Texas, current county subdivision state-based*, US Census Bureau, Department of Commerce, accessed: 2017-09-26. [Online]. Available: https://catalog.data.gov/dataset/