

Alleviating the Modifiable Areal Unit Problem within Probe-Based Geospatial Analyses

Submission # 324

Abstract

We present a probe-based interface for the exploration of the results of a geospatial simulation of urban growth. Because our interface allows the user great freedom in how they choose to define regions-of-interest to examine and compare, the classic geospatial analytic issue known as the modifiable areal unit problem (MAUP) quickly arises. The user may delineate regions with unseen differences that can affect the fairness of the comparisons made between them. To alleviate this problem, our interface first alerts the user if it detects any potential unfairness between regions when they are selected for comparison. It then presents the dimensions with potential problematic outliers to the user for evaluation. Finally, it provides a number of semi-automated tools to assist the user in correcting their regions' boundaries to minimize the inequalities they feel could significantly impact their comparisons.

Categories and Subject Descriptors (according to ACM CCS): Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications; I.6.6 [Simulation and Modeling] Simulation Output Analysis

1. Introduction

Our application seeks to present the results of an urban growth simulation to policy analysts, urban planners, etc. such that they can analyze historical growth patterns, examine predicted trends, and compare the characteristics of development between different regions. We provide the user with the ability to probe the map-based data via selecting regions of any size and shape, resulting in coordinated visualizations reflecting those regions-of-interest, and to directly compare these regions-of-interest with each other.

However, by giving the user this freedom to select regions at such a wide range of shapes and sizes, we inadvertently make their analyses particularly vulnerable to unforeseen inequalities between regions being compared. For example, household level data, such as income or population, is aggregated into blocks to protect privacy. Depending on how one defines new regions cutting through these blocks, one can find different average values for the same locations. This is part of the long standing problem in the field of geography and spatial analysis, known as the modifiable areal unit problem (MAUP). Probe-based interaction is particularly prone to being effected by MAUP due to the inherent variability in areal units.

This prevalence of the MAUP in our application is compounded by the fact that the target audience does not necessarily have expert knowledge regarding all the “behind the scenes” data layers that have gone into guiding and dictating the underlying simulation’s behavior. For example, a policy analyst may understand the zoning limitations that constrain growth in a particular area, but is unlikely to understand the geologic barriers to construction in the same region, i.e. soil suitability and parcel slope.

To help alleviate the effects of the MAUP in our application, we have provided a number of enhancements to the previously available probe-based interface elements. First, when the user selects multiple regions to directly compare against each other, we evaluate the statistical distributions within the various dimensions and look for outliers with deviations that have the potential to be particularly problematic in the final analyses. When these are detected, we alert the user to them and provide an overview of the possible inequalities in each dimension that may affect their intended analysis. If the user decides that any of these inequalities might have a significant negative impact on their desired analysis, they can then choose to adjust them using a number of provided tools. These tools provide methods to manipulate the boundaries of regions to assimilate and discard land coverage types, grow and

shrink in advantageous directions, and trade area amongst themselves to attempt to bring their disparities within the user's selected bounds.

We illustrate the usefulness of these enhancements with an example scenario in which the analysis of urban sprawl growth patterns for a number of suburbs around a major metropolitan area is complicated by predefined city boundaries containing disproportionate amounts of water and protected land, which the underlying simulation specifically ignore.

2. Related work

The modifiable areal unit problem (MAUP) is a long standing, unsolved problem in geography sciences. It refers to the fact that when point data is aggregated into areal units, the variation in how the units, or regions, are delineated can cause significant variation in the aggregated values at any point. The issue itself has been long known, but the term MAUP was coined and the problem described in detail by Openshaw [Ope84]. It is primarily studied in regard to its effects on geospatial analyses of aggregated data in the field of socio-economics, politics, and epidemiology. [FW91] [OA99] [Arm95]

Traditionally, the MAUP is split into two components. The first, the scale problem, relates the choice in the number of regions being compared to its effects on the variation in the results of numerical analysis between those regions, especially when the source data was initially aggregated at a different resolution. We do not address this component in our system, as in our case, it is more of an issue with how the underlying datasets are generated from data at different granularities. (More on this in Section 4) Further, to address its slight appearance on the interaction side, it would require drastic changes to the user's freedom to select and compare any number of regions in an explorative manner. This is more applicable to situations in which the map's area is completely distributed into non-overlapping, space-filling regions, and not the disconnected and sparsely covering region selections commonly made in our probe-based interface. However, in the future, it might be worth considering the addition of automatic "split region" and "combine regions" behaviors if a sufficiently elegant method is devised to ensure these actions to not compromise the user's analytical tasks.

In this paper, we are primarily concerned with the second component of the MAUP, the aggregation problem. This problem relates the choice of where and how boundary lines are drawn between regions to the effect on variation in the resulting values for numerical analysis within those regions. An good example of this problem arises when working with census derived data. Due to privacy concerns, the individual household point data is never revealed. Instead, average values are given for "census blocks", which can be apartment complexes, city blocks, or arbitrary delineations of rural tracts of land. The choice in how to delineate these blocks has a direct and significant impact on the aggregated values. If the individual point data was instead aggregated into regions

delineated by different methods, say a regular grid, or by postal code, the values available at any particular point on the map point would likely show significant variation from the "census block" method. Thus, the MAUP problem is closely related to another often encountered problem in geography, the ecological fallacy, which states that it is wrong to make inferences as to the values of individuals in a region based on the aggregated values of that region.

Research into the MAUP problem in geospatial analysis fields tends to focus on either understanding the variance or error that can be generated through different scales and aggregations so as to understand the effects that the MAUP can have on analyses performed on the aggregated data [CHC95], or on developing methods to calculate optimal aggregation zones [Nak98]. In contrast, we are interested in monitoring the ways the user chooses to define their own areal units, and then figuring out if these delineations could produce misleading results based on the differences across multiple dimensions.

One of the most important differences between the MAUP situations commonly encountered in probe-based interfaces and those studied in the geospatial analysis field is that the MAUP research in the geospatial analysis field seems to focus primarily on space-filling regions that cover the map's entire extent, and share boundaries. While we do provide tools to deal with these conditions, we are primarily concerned with the disjunct regions, with large areas of unselected land, that are more common to our probe-based interaction. These have more room to grow, and adjustments of multiple regions are rarely zero-sum cases.

To the best of our knowledge there have been no similar visualization systems that attempt to find and alert users to potentially misleading dimensional inequalities between regions-of-interest being compared, and provide tools for the semi-automated adjustment of these questionable regions.

This application represents the next generation of probe-based interface, and the first to be released into the hands of actual end users. The considerations and tools for handling the MAUP detailed in this paper are one of the major new features that improve upon the original probe-based interaction groundwork [BDW*08]. We believe these improvements significantly strengthen the technique's power for geospatial analysis.

3. Application

In this section we describe our application, its background and basic functionality, as well as detailed descriptions of the MAUP overview and adjustment panels.

3.1 Background

Our application is the Urban Growth Decision Support System. It is designed to provide a highly interactive interface for policy makers, urban planners, etc. to explore and analyze both 30 years of historical urban growth and 25 years of predicted future growth. It focuses on a 240 km

(150 miles) wide region around a major metropolitan area characterized by significant urban sprawl.

Satellite imagery was used to classify historical land coverage as developed or undeveloped (e.g. natural vegetation versus impervious surfaces). Protected lands such as forests and parks were recorded as well. The currently remaining undeveloped land was then ranked by its attractiveness to new development. This was done by considering positive factors, e.g. distance to major employment centers, percentage of surrounding parcels already developed, and established infrastructure such as road density, as well as negative factors, e.g. slope of terrain. Then, by using forecasts of population growth for each region, and knowing how much land is used per person in each type of area (i.e. high density urban core, suburban fringe, etc), the appropriate amount of land was converted from undeveloped to developed for that particular time step, and the model was recalculated for the next time step. The results of this simulation process are highly detailed land coverage maps for multiple time steps ranging from 1976 to 2030.

The application was designed to run on a desktop for standard single-analyst usage, a laptop with projector for presentations to policy makers in the field, as well as on our multi-touch table for simultaneous collaborative use between multiple analysts and domain experts. A sample view of the application being used is shown in Figure 1.

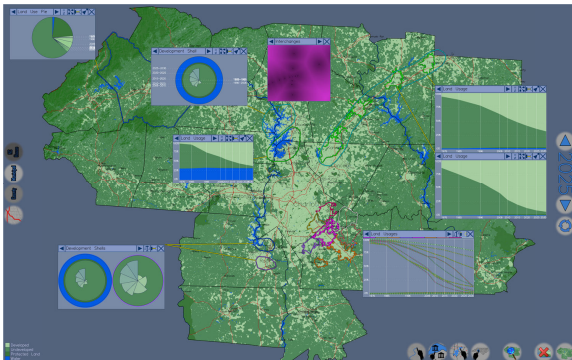


Figure 1: An example workspace in our application.

3.2 Probe creation

As a probe-based interface, the primary direct interaction with the map (aside from navigation) is to define regions-of-interest, which spawns coordinated probe interfaces allowing the analyst to examine the data within the associated region with a number of different visualizations. In our application we provide a wide variety of methods for selecting regions-of-interest. The most basic, and free form, methods are the ability to lasso or circle a region of any shape or size, or to “paint” region masks directly onto the map, using either the mouse or the users fingers (when run on a touch table). We also allow the user to select using the wide array of vector data commonly available from government geography databases. The user can thus select a variety of predefined regions, such as school

districts, city boundaries, voting districts, counties, water sheds, as well as combinations thereof.

This wide assortment of selection methods available provides great freedom in how the user can query the data, but it also exacerbates the MAUP which inherently arises in this type of analytical situation.

3.3 Comparisons

After the user has selected multiple regions-of-interest, each spawning its associated probe-interface, they can choose to combine these interfaces with each other to form comparison interfaces. In these interfaces, the visualizations pull the data from the individual regions-of-interest and plot it directly against each other. Upon the creation of a comparison interface, we calculate the statistical distribution of the regions across all relevant dimensions. If we determine that any of the regions being compared are potentially significant outliers within a particular dimension, then we alert the user by displaying a large flashing exclamation mark on that comparison window’s toolbar.

3.4 MAUP overview panel

From within a comparison interface, pressing the MAUP interface icon switches the interface to the MAUP overview panel. The purpose of this panel is to allow the user to evaluate any potentially problematic inequalities and choose which to take corrective action upon.

In the MAUP overview panel, each dimension has its own one-dimensional plot and action button, as shown in Figure 2. The plot itself is centered at the mean value for the dimension and expands three standard deviations above and below the mean on each side. Each region being compared is then plotted as a vertical line color coded to match the region. Regions beyond three standard deviations of the mean are plotted at the appropriate end of the plot. We highlight any regions that were determined to be outliers with a yellow indicator above the plot, as well as automatically selecting that dimension for corrective action. Under each dimension’s plot, there is a scale/measuring tool that allows the user to drag across the plot to quickly measure the actual range of values across a cluster of regions, as well as the actual value by which outliers deviate from these clusters.

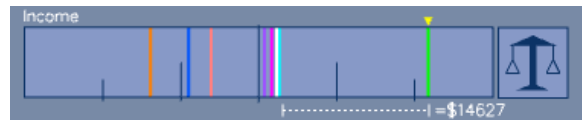


Figure 2: An example plot from the overview panel showing the distribution of eight regions in terms of average income. Notice that the green region has been flagged as an potentially problematic outlier, and that the user has measured how far it deviates from the upper end of the main cluster.

One shortcoming of this technique is that there is a limit to how many regions can be differentiated from each other with any color coding scheme. There are only so many distinctive colors, and after about ten regions, it becomes hard to distinguish which lines correspond to which regions. This can be overcome by labeling or highlighting the region on the map upon selection.

Upon entering the MAUP overview panel, the user can quickly assess the situation by viewing the highlighted dimensions with potentially problematic outliers and choose whether to either accept the suggested and automatically selected dimensions, or select and deselect dimensions at will. In practice, the user will rarely want to simply accept all of the suggested selections, as they are usually interested in looking at the differences between regions in at least one dimension. At this point, pressing the “Adjust selected dimensions” button transfers the user and any selected dimensions to the MAUP adjustment panel.



Figure 3: An example view of the MAUP overview panel.

3.5 MAUP adjustment panel

In the MAUP adjustment panel, each dimension that was selected in the MAUP overview panel is once again presented as a one-dimensional plot of the statistical distribution of the regions being compared. However, now the purpose of this graph is to adjust min and max values for the boundaries as targets to be used during the region adjustment procedures.

The user can move the ends of the selection box to correspond with existing or desired cluster boundaries. The actual value range within the proposed cluster is presented below the plot. A target value is also indicated by an upward pointing green triangle. Outliers outside the desired cluster boundaries are those that the adjustment algorithms will adjust until they either reach it, get as close as possible, or fall within the cluster boundary, depending on which adjustment method is being used.

Once the desired boundaries are set for each dimension, the user can choose from an assortment of adjustment tools, which are enabled or disabled based on the dimensions that have been selected for adjustment.

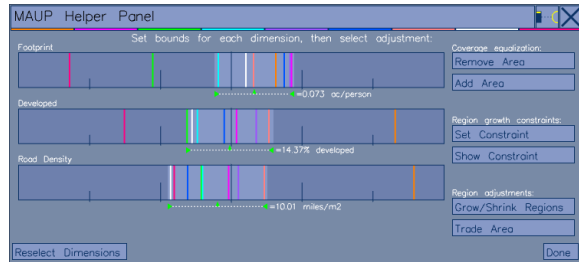


Figure 4: An example view of the MAUP adjustment panel.

Before adjustments are initiated, the user has the opportunity to use the region-of-interest selection tools to create constraints around regions, which they will not be allowed to grow beyond. For example, as shown in Figure 6, one might want to adjust the boundaries of local political jurisdictions which must always remain a subset of a larger political jurisdiction.

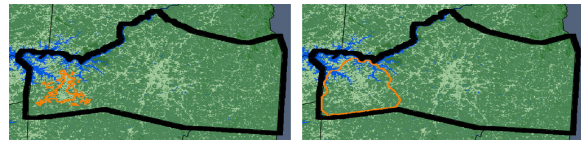


Figure 5: Before and after an add area adjustment of a city boundary (orange) within a constraint (thick black line) set for the county boundary that the city must remain inside.

The first, and most simple, adjustments available are “Add area” and “Remove area”. These are available for dimensions with categorical data, such as land coverage types, e.g. water, protected, etc. “Add area” attempts to expand regions that are below the minimum bound outwards into matching land types until either the target value is reached or until there is no available land within a reasonable distance. (“Reasonable” in this case is defined as how far we want to allow any added, non-contiguous regions to stray from the main region.) “Remove area” erodes the boundaries of regions that are above the maximum bound inwards, removing matching land types until either the target value is reached or there is no more available land of that particular type to remove. Both of these methods can be easily adjusted to maintain the existing connectivity of regions, however in practice this greatly reduces its effectiveness and ability to reach target values, and provides little more than an aesthetic benefit in analyses that do not require contiguous regions.

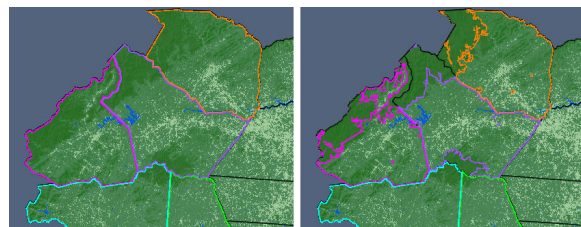


Figure 6: Before and after a remove area adjustment is made to remove protected wild lands (darkest green).

The other adjustment tools are more complicated, but are able to adjust dimensions with continuous data. The first is “Grow / Shrink regions”, which manipulates the boundaries of the regions both inward and outward at the same time, in an attempt to bring their values within the desired bounds. This is done through an iterative process consisting of simultaneous combinations of both removing and adding area at the edges of the regions to maximize movement towards the desired bounds while not exceeding the bounds set on other dimensions. The process completes when either the values for the selected dimensions fall within the desired bounds, or no more possible progress is achievable, e.g. no appropriate area is left available for removal. When using this tool, regions can both initially overlap as well as overlap after adjustments are made. If overlapping results are not desirable, regions can be prohibited from growing into each other.

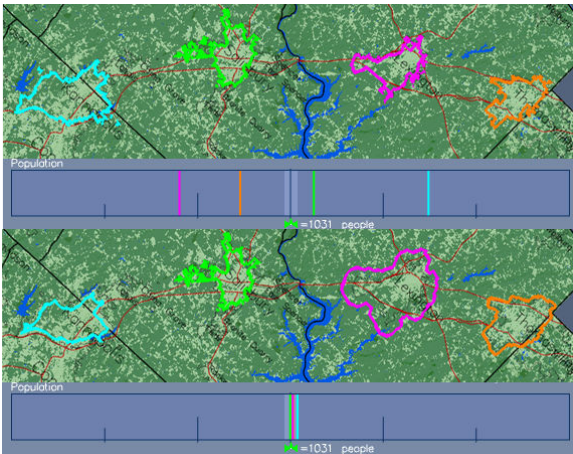


Figure 7: A grow/shrink adjustment of four regions to bring the population of each to be within ~1000 people of the mean. Above are the original predefined city boundaries and below are the results of the adjustment.

The final adjustment available here, “Trade area”, is the most complicated. It is used to adjust border-sharing and space-filling regions, such as political jurisdictions, which cannot overlap and must collectively cover a certain area completely, as opposed to the collections of disjunct and overlapping regions adjustable by the previous methods. It behaves much like the “Grow / Shrink regions”, in that it attempts to both grow and shrink portions of regions’ boundaries to bring values for selected dimensions within the desired ranges, but now it considers the costs and benefits of each boundary adjustment to the regions of each side of the boundary. Thus it is actually weighing the benefits of trading bits and pieces of area between the regions. It iteratively executes the most advantageous trades of area between regions, redrawing the boundaries of multiple regions in the process, until it achieves its goal or runs out of valid adjustments.

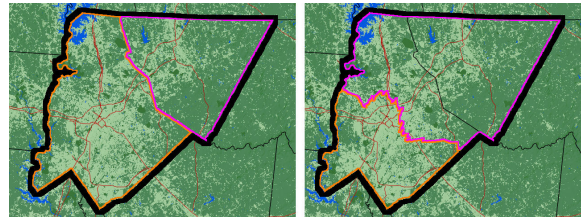


Figure 8: Before and after a trade area adjustment of two regions to make their populations equal. The thick black line is a constraint used to force the regions to stay within their non-shared boundaries.

Careful consideration is still required by the user as to choosing which dimensions to adjust, target bounds, and adjustment methods. However, the MAUP helper panel attempts to assist the user in making these choices through both helpful intuitive visualizations and enabling only those adjustment methods relevant for the selected dimensions.

4. Implementation

Our software accepts two main types of data. The first is vector data that is used to provide both reference, e.g. roads, city names, as well as semi-automated assisted selection techniques, e.g. “select city bounds.” We utilize the ESRI shapefile format for this type of data, as it is widely supported among all of our GIS collaborators.

The second data type is raster based data layers, in .tif format. These raster images provide the raw data for our application, such as land coverage and demographic information. For most variables, conversion from existing GIS formats to our raster based format is fairly straightforward. However, for many household based demographic variables, such as median income, consideration must be made with regard to ensuring the most accurate distribution of aggregate data to individual pixels, so as to minimize ecological fallacy effects.

In creating our population maps, for example, instead of merely dividing the population of a census block by the number of pixels within it to get a population value for each pixel, we utilized supplementary data, including satellite imagery, to perform dasymetric mapping. In this manner, if a census block contains farmland as well as an urbanized area, the pixels in the urbanized area would contain the majority of the population, while the farmland areas with no impervious surfaces would have near zero population values. This was very important for our application, where users are interested in the differences between developed and undeveloped areas, and can select their own regions cutting through census blocks.

The interface is written in C++ and uses OpenGL for all onscreen graphics. OpenCV [OCV09] is used to perform all image processing operations (erosion, dilation, etc.).

4.1 Statistical evaluation

Our statistical evaluation is quite simple, but is sufficient for our purposes. Upon comparison interface creation, the mean and standard deviation for each dimension is calculated by examining the precomputed values for all regions that are being compared. The number of standard deviations from the mean value is used to detect outliers. We use greater than two standard deviations from the mean as a threshold, over which we alert the user to the detected outlier and automatically select that dimension for adjustment. A more rigorous statistical evaluation could easily be substituted here if deemed necessary.

4.2 Adding and removing area

The “add area” and “remove area” functions, which expand or contract a regions boundary to include more or less of a particular categorical value, behave as follows: First we generate a search mask that is used to find candidate pixels to either add to or remove from the region. This process is visually explained in Figure 9.

We begin by extracting a binary image mask (A) describing what areas make up in the current region. If we want to add area, we perform morphological dilation on this mask, resulting in expanded mask (B). We then generate a search mask (C), which equals (B AND (NOT A)). The search mask is a ring around the outside of the original mask containing all pixels within the chosen kernel size (more on choosing this later) of, but not within, the original mask.

Likewise, if we want to remove area, we perform morphological erosion on the original mask, resulting in a shrunken mask (B). We then generate our search mask (C) as ((NOT B) AND A). This results in a ring around the inside of the original mask, with all pixels within the original mask’s boundary by no more than the kernel size.

After generating our search mask, we begin examining the pixels within individually to see if they match the categorical type we are interested in. If we are trying to add area, these pixels are set as true in our original mask defining the region. If we are removing area, they are set as false in the original mask. We continue this until either the desired number of pixels has been added or removed, or we run out of candidate pixels in the search mask. In the former case, we are done adjusting the region. In the latter case, we repeat the process, generating a new, further reaching, search mask.

After generating our search mask, we begin examining the pixels within individually to see if they match the categorical type we are interested in. If we are trying to add area, these pixels are set as true in our original mask defining the region. If we are removing area, they are set as false in the original mask. We continue this until either the desired number of pixels has been added or removed, or we run out of candidate pixels in the search mask. In the former case, we are done adjusting the region. In the latter case, we repeat the process, generating a new, further reaching, search mask.

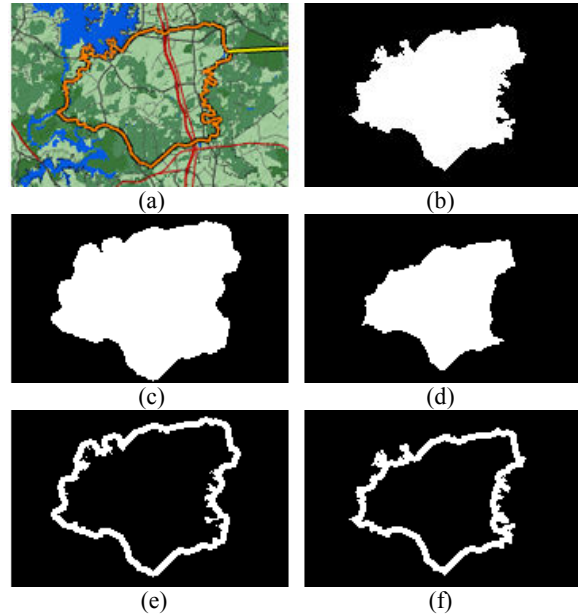


Figure 9: The process of calculating search masks for adding or removing area from a region. (a) is the region-of-interest to be adjusted on the map, (b) is the binary image mask for the area inside this region, (c) is the dilated mask, (d) is the eroded mask, (e) is the dilation search mask, and (f) is the erosion search mask. Notice that $(e) = (c) - (b)$ and that $(f) = (b) - (d)$.

Aside from achieving our target goal, there are two other stopping conditions: When removing area, we stop if there are no longer any new candidate pixels being generated, i.e. all possible pixels that can be removed have been removed. When adding area, we stop if a certain number of dilations have failed to unearth any candidate pixels that match our specific categorical type. The number of fruitless dilations dictates how far away new disjunct regions can stray from the original region.

The choice of kernel size for these morphological operations is a tradeoff between speed (less iterations required) and even growth (or reduction) patterns. Larger kernel sizes have a tendency to provide more candidate pixels than needed. The algorithm converts candidate pixels in a scanning pattern from the top left, and so this can result in growth mostly in the northern direction when kernel sizes are too large. Lower kernel sizes ensure that multiple concentric rings of candidates will be evaluated, resulting in a more even, outward growth. We have found a 7x7 kernel to be a good balance. By using a 3x3 kernel one can ensure that only those pixels that are directly connected to the edges of the region will be added or removed, and hence no new disconnected islands or holes will be generated.

4.3 Growing and shrinking regions

The “grow and shrink regions” function attempts to automatically augment the size and shape of regions,

independently of each other, in order to adjust outlying values in selected dimensions to be within the specified value range. Each region is checked to see if it has at least one value outside the desired range in any of the dimensions selected for adjustment. If so, we attempt to adjust this region, then move on to evaluate the next region.

The adjustment process for individual regions, which is visually explained in Figure 10, begins with the generation of search masks from both dilation and erosion operations on the region's mask, as detailed in Section 4.2. Now that we have these two masks, which form rings both inside and outside of the region's current boundary, we cut these masks up into a number of candidate sub-masks. This is done by finding the center of the region, and then generating a number of "pie slice" shaped masks emanating outwards from the center point (using OpenCV's `cvEllipse` function). We then generate our collection of candidate adjustment masks by computing the binary AND of each slice mask and the erosion and dilation masks. If it is desirable to restrict regions from growing into each other, the other regions' masks can be subtracted from the candidate masks.

The number of slices to cut the original erosion and dilation masks into is a tradeoff between speed of computation and accuracy. By making too few, and thus larger, slices, the regions are very restricted in their choice of growth directions, will not add or remove area as efficiently, and are less likely to reach their dimensional value goals. A reasonable solution is to choose a number of slices based on the current size of the region. Small regions (< 3km wide) may require as few as eight slices for sufficiently pleasing results, while larger regions (~30km wide) can benefit from as many as 30-40 slices. Another option here is to vary the number of slices on each pass, as the size of the region changes. By varying the number of slices in each pass, one also lessens the chance of unnatural looking radial patterns.

All non-zero candidate masks are processed as temporary regions-of-interest and evaluated to determine the values it contains for each dimension of interest. The temporary region-of-interest and mask are discarded, and the values are stored in a candidate adjustment object along with details including which operation type (erosion or dilation) and slice number was applied to generate it.

After all candidate masks are processed into a list of candidate adjustments, they are sorted in descending order according to the progress they would make in bringing the values for the dimensions that still need adjustment within their desired ranges. We then choose a subset of this list by starting at the top and evaluating if the adjustments would result in the region moving outside any of the bounds for the other dimensions, or overshooting our target values. By deciding how far down the list to evaluate on each pass, we can make a tradeoff between speed and optimum results. Selecting only the single best adjustment from the list results in only the locally optimal choice being made on each pass. Conversely, selecting all valid adjustments produces quick results, but they may be far from the optimal solution.

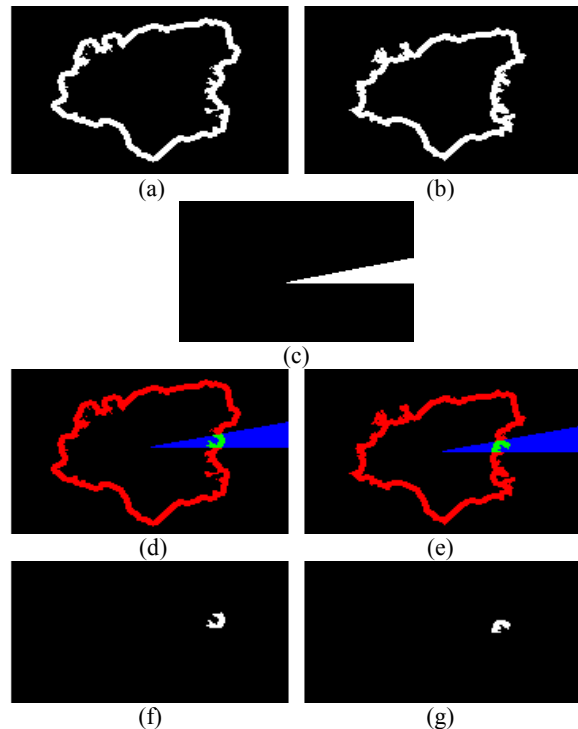


Figure 10: The process of creating candidate adjustment masks. (a) & (b) are the dilation and erosion search masks generated as in Figure 1. (c) is a sample of one of the many slice masks that are used to divide up the search masks. (d) & (e) show the slice mask superimposed on each search mask to show the Boolean operations. (f) & (g) are the resulting masks, representing candidate adjustments.

Once a subset of adjustments has been selected, we redraw two new slice masks, one containing all the slices that correspond to selected adjustments that were erosions, and one for the selected dilation slices. We AND these slice masks with the corresponding erosion and dilation masks. The resulting sliced erosion mask is then used to remove pixels from the regions mask, then the sliced dilation mask is used to add pixels to the regions mask.

This whole process repeats itself until either the values for the selected dimensions are all within their desired boundaries, or no candidate adjustments are found to be acceptable, and thus there are no more adjustments that can be made. In the latter case, a refinement can be made which increases the number of slices and searches again, with the candidate areas now of smaller area.

4.4 Trading area

The trade area function is similar to the "grow and shrink regions" function, but instead of adjusting the regions independently of each other, it adjusts regions with respect to each other. This is used for cases where regions border each other, and the user does not want them to overlap, but is willing to allow the boundary between them to move.

To accomplish the adjustment of multiple regions at once we use a modified greedy algorithm. Our solution makes the optimal choices on each pass but does not guarantee the best possible solution. It can however be fast enough to return results within a short enough amount of time (< 5 minutes) to maintain interactivity, whereas finding the optimal solution could take hours. It is merely a proof of concept implementation at this point, and future work must be done to make this adjustment as efficient and effective as possible.

We begin by generating a list of candidate adjustments in the manner described in Section 4.3, but this time we generate them for each region. We also now record not only the effects the adjustment would have on the region it was generated from, but also its converse effect on any other regions that either currently contain, or are proposed to contain it.

For each region and dimension that needs adjustment, we sort the list by how far the adjustments would move the outlying value into the desired bounds. We then start at the top of the list and look for adjustments that are advantageous (they move the value towards the target) and do not bring the values in other dimensions outside those bounds. Matching candidate adjustments have a preference value incremented each time they are chosen to be made by a region or dimension.

After all regions and their dimensions have been considered, we sort the list by preference value. We execute the top N adjustments from this list as long as they have a preference value of at least one. The choice of N , how many of the top requested adjustments to make, is another trade-off between speed and how close the results will be to the optimal solution. We like a value of 5% of the total number of candidate adjustments, but have used different values with varied success across situations.

When executing the top N adjustments, we follow the same process as in Section 4.3. However, when using these sliced dilation and erosion masks, we not only add or remove the pixels from the region the mask was generated from, but perform the opposite operation on the same pixels in the neighboring region. In this manner, area/pixels are not gained nor lost, but instead transferred between regions.

Once the adjustments are complete, we check to see if any regions still have values outside the desired bounds, if not then we are finished. If so, then we make another pass. If another pass results in no acceptable candidate adjustments, we can either stop, or increase the number of slices per region in an attempt to find smaller valid adjustments on another pass.

5. Scenario

In this example scenario, the analyst's goal is to compare the growth patterns, both historical and predicted, for a number of cities, and clusters of smaller cities, that are all suburbs of a major metropolitan area. The analyst wants to examine the relationship between the amounts of land that change to developed with each time step, and during what times development rates peaked, for each region relative to

the others. As shown in Figure 11, they have selected regions-of-interest using the city selection tool. However, some of the regions contain significant amounts of water, and others significant amounts of protected land. The simulation is programmed to ignore both of these land-cover types, and they will never get developed. Their presence can cause misleading results for analyses or visualizations that rely on ratios involving developing land. This effect can be seen in the "land use pies" visualization shown in Figure 12, where water (blue) and protected land (dark green) slices squeeze the developed and undeveloped land slices into remaining degrees of the circle. Comparing the angles between regions to see the relative amounts of growth that occurred in each time step is now misleading. The same amount of growth will appear smaller in the region with excess water.

The user is alerted by the flashing MAUP alert icon, and enters the MAUP overview panel. Here, as shown in Figure 13, the footprint (land developed per person), road density, undeveloped, protected, and water dimensions have been automatically selected due to outliers being detected within them. Not concerned with footprint or road density, the user unselects those dimensions. The user also unselects the undeveloped dimension, as the variations within that dimension are one of the aspects of the data they are interested in.

The user advances to the MAUP adjustment panel. As shown in Figure 14, they set the target bounds for each dimension around the regions with the least amounts of water and protected land. They then select the "remove area" tool to bring the other regions within those bounds. Figure 15 shows the water and protected space being removed from the regions with excesses. Finally, Figure 16 shows the pie charts, now free of the misleading distortions from excess water and protected land.

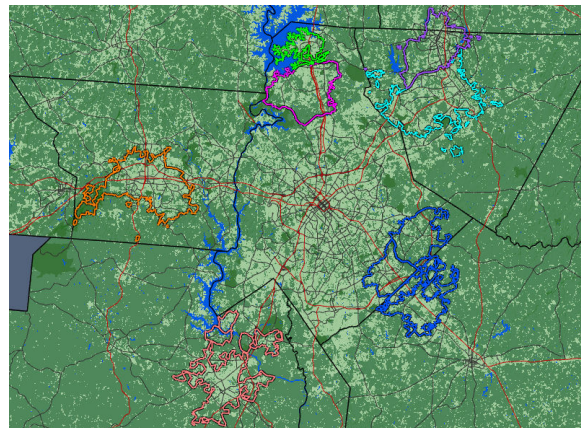


Figure 11: The selected regions in the example scenario.

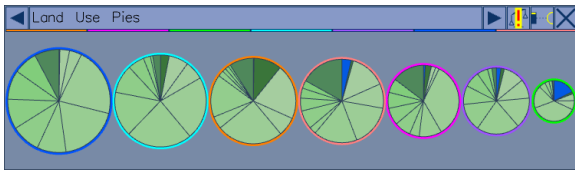


Figure 12: Pie charts of the amounts of land developed over each time step. Notice the amount of water in the 7th region and protected land (darkest green) in the 3rd region.



Figure 13: The MAUP overview panel from the scenario.

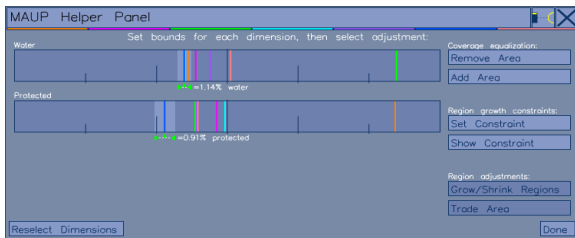


Figure 14: The MAUP adjustment panel from the scenario.

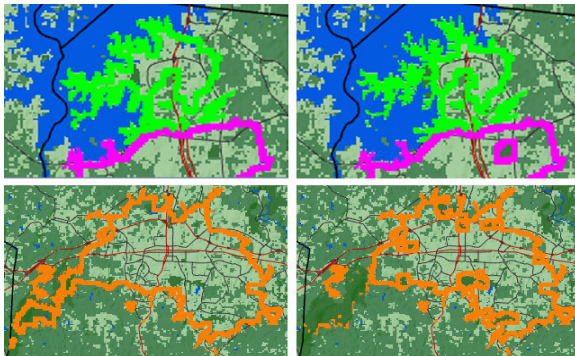


Figure 15: Regions before (left) and after (right) the adjustments made in the example scenario. Notice the removal of water (top) and protected land (bottom).

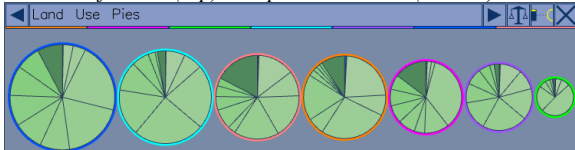


Figure 16: The same pie charts as in Figure 12 after adjustment to remove excess water and protected land.

5. Future Work

As identified in the related work section, we could attempt to address the “scale problem” component of the MAUP through the introduction of tools to split and combine regions-of-interest. This would require a more thorough understanding of how much modification of the user’s analysis is tolerable. For example, the current model lets the user ask and answer questions such as “How are areas A, B, and C like area D?”, whereas a split operation might turn this into “How are areas A, B, and C like these similarly sized subsets of area D?”

It is also worth examining the processes our collaborators use to de-aggregate data into our raster based input. This is an area in which both the scale component of the MAUP and the ecological fallacy are of supreme concern, as all analyses done within the interface rely on the accuracy of the underlying maps. This has been studied in spatial analysis literature, but there may be specific concerns or loopholes related to our particular usage of the derived rasters.

As noted in Section 4.4, our proof-of-concept “trade area” adjustment algorithm has much room for improvement. We hope to bring in collaborators with image processing and geography backgrounds to help improve both the speed and effectiveness of our current technique.

6. Conclusion

We have explored the origins of the modifiable areal unit problem (MAUP), and based on these understandings we have identified the ways in which probe-based geospatial applications are particularly susceptible to the MAUP. The user can probe the data by selecting their own regions-of-interest using a wide range of selection tools operating at a range of scales. When combined with the underlying raster-mapped data, generated from sources with different aggregation scales, the opportunities for the MAUP to affect the user’s analysis are infinite.

While we cannot easily solve the MAUP, we can plan for its appearance in our geospatial analysis applications. By alerting the user to any potential issues with the regions-of-interest they select to compare, we remove much of the possibility that the comparisons they make will be misleading or misinterpreted. Simple visualizations can provide quick indication of outliers in the distributions, allowing one to see at a glance what dimensions might become problematic in their analyses. Finally, by providing semi-automated tools to help the user understand these inequalities, and then correct their selections, we minimize the impact of these unintended problems that are inherent to probe-based interfaces, with their great freedom in region-of-interest selection choices.

References

[Arm95] C. Armhein. "Searching for the Elusive Aggregation Effect: Evidence from Statistical Simulations" *Environment & Planning A*, vol. 27, pp 105-120. 1995

[BDW*08] Thomas Butkiewicz, Wenwen Dou, Zachary Wartell, William Ribarsky, and Remco Chang. "Multi-Focused Geospatial Analysis Using Probes" *InfoVis 2008 / IEEE Transactions on Visualization and Computer Graphics (TVCG)* vol. 14, no 6. pp. 1165-1172, Nov/Dec, 2008.

[CHC95] S.L. Cutter, D. Holm, and L. Clark. "The role of geographic scale in monitoring environmental justice" *Risk Analysis*, vol. 16, no. 4, pp 517-525, 1995.

[FW91] A. S. Fotheringham and D.W.S. Wong. "The modifiable areal unit problem in statistical analysis" *Environment and Planning A*, vol. 23, pp 1025-1044. 1991.

[Nak98] Tomoki Nakaya. "An Information Statistical Approach to the Modifiable Areal Unit Problem in Incidence Rate Maps" *Environment and Planning A*, vol. 32, pp 91-109, 2000.

[Ope84] Stan Openshaw. "The Modifiable Areal Unit Problem" *Concepts and techniques in modern geography*, issue 38, Geo Books, 1984.

[OA99] Stan Openshaw, Seraphim Alvandies. "Applying Geocomputation to the analysis of spatial distributions." In *Geographic Information Systems: Principals and Technical Issues*. vol. 1, 2nd ed., 1999.

[OCV09] Open Computer Vision Library
<http://sourceforge.net/projects/opencvlibrary/>