# Transforming geographic boundaries to prevent symbol overlap

Gian, Thomas

*Award date:*
2022

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Algorithms, Geometry & Applications

# Transforming geographic boundaries to prevent symbol overlap

*Master's Thesis*

Thomas Gian

*Supervision:*
dr. W. (Wouter) Meulemans

*Assessment committee:*
dr. W. (Wouter) Meulemans
dr. M. J. M. (Marcel) Roeloffzen
dr. F. V. (Fernando) Paulovich

30-08-2022

**Abstract**

In this project, we discuss proportional symbol maps, which represent a quantitative variable shown in the size of the symbol. For these maps, it is of importance that individual symbols can be recognized distinctly and that their relative sizes can be judged. An undesirable yet common occurrence is the overlap between symbols, which we aim to completely prevent. One approach is to minimize the displacement of the symbols so that there is no overlap, leaving the corresponding represented feature unchanged. We approach this problem by allowing the shape of the corresponding represented feature to be transformed. We first model the problem and then we design and implement an algorithm, which we then evaluate through various experiments.

# Contents

# Chapter 1

# Introduction

Proportional symbol maps are commonly used to visualize numerical data associated with geographic locations by placing a symbol on a map, where the size of the symbol represents the numerical value. These locations can be interpreted as coordinate points that reasonably represent the location of a geographic feature on the chosen map scale, such as cities on the scale of the world map. Another interpretation is that each location represents an aggregated district of regions, instead of any point therein, such as a country. In this thesis, we will consider the latter and work with maps of a set of countries.

The primary goal of proportional symbol maps is to enable users to accurately judge the sizes of the symbols, both in comparison to the legend to estimate the data values, and in comparison to each other to judge relative size. To achieve this goal, we want to prevent the overlap of symbols, as obscured symbols make it difficult to fully grasp its actual size. Additionally, the ratio between symbol sizes should be large enough such that the symbol size difference is evident to the map reader. Otherwise, most of the symbols look nearly the same size and the map ends up being rather uninformative.

However, in practice most proportional symbol maps have overlap between symbols, typically in regions with a high density of features or around the largest symbols. This leads to errors in size interpretations and large numbers of symbols obscure the underlying geographic reference map, making it difficult to recognize which feature each symbol represents. Preventing such overlap can be done by optimizing symbol sizes or by omitting a minimal number of symbols, which is NP hard, even for circular symbols of the same size [7]. However, these solutions are not particularly effective in providing a complete and legible visualization with many symbols. Optimizing symbol sizes may reduce its sizes such that the differences between symbol sizes becomes negligible on the map, making the detection of spatial patterns difficult and omitting symbols means that the output map is incomplete. An alternative is to displace symbols with minimal displacement such that there is no overlap, and additionally the geospatial characteristics

should carry over to the new positions [32].

We aim to deal with this problem from another perspective by transforming the boundary or shape of the underlying geographic feature that corresponds to each symbol in such a way that the entire symbol is fully contained within the feature. This will prevent symbol overlap because geographic features do not overlap each other. Additionally, we wish the transformation of the shape to be such that we obtain a large range of symbol sizes, in other words, every geographic feature must be able to contain a large circle. Finally, we want to maintain the topology of the features in terms of adjacency, which comes at the cost of distortion of the feature shapes. Thus, the transformation should aim to maintain the feature's shape for ease of recognition.

The secondary goal of proportional symbol maps include aesthetic appeal and a symbol shape that is easy to interpret. Although these symbols can be of any shape, it is common to use simple geometric shapes such as circles or squares whose relative sizes are easier to judge, especially in dense areas with overlapping symbols. Among the geometric symbols, the predominant shape has been circles as they are relatively easier to distinguish when they overlap [13]. Thus, we shall also use circle shapes for our symbols.

## 1.1 Contributions and organization

The primary contribution is to present an algorithm that outputs a proportional symbol map that has no symbol overlap and aims to provide a large range of symbol sizes for the visualization. We discuss our algorithm design decisions as well as their experimental results in hope of inspiring new algorithms to consider this novel approach.

In Section 1.2 we give an overview of related work regarding managing symbol overlap and shape distortion. Followed by the preliminaries in Chapter 2. In Chapter 3 we define the problem model and in Chapter 4 we present our algorithm along with a discussion behind the design decisions. We discuss the setup of the experiments and evaluation of our algorithm in Chapter 5 and finally in Chapter 6 we review future work to our approach.

## 1.2 Related work

**Symbol overlap** The general consensus is that allowing some overlap is acceptable, as eliminating all overlap often requires a significant reduction in symbol sizes to the point where its size is difficult to judge. The allowance of symbol overlap is often paired with the use of opaque symbols due to user preference, despite obscuring other symbols and the underlying map [18]. Nandanari [35] visualizes multi-dimensional data by merging overlapping

symbols based on the similarity of represented attributes and the ratio of their overlapping area.

Another approach that allows overlap is to arrange the order in which the opaque symbols are visible. Here, we consider two types of drawings of symbol arrangements. Physically realizable drawings can be seen as a drawing constructed from symbols cut out from sheets of paper, we can interleave them without breaking physical restrictions such as cutting the symbols. Stacking drawings are obtained by layering symbols on top of each other. The quality of the drawing depends on how well it enables the viewer to judge the symbol sizes, which is proportional to the portion of visible symbol boundaries. This leads to two possible optimization problems: maximizing the minimum visible boundary length of each symbol (the *max-min* problem) or maximizing the total visible boundary length over all symbols (the *max-total* problem). Cabello et al. [5] have shown both problems to be NP-hard for physically realizable drawings and they present an algorithm that solves the max-min problem for stacking drawings in $O(n^2 \log n)$ time, while the complexity for the max-total problem remained open. Kunigami et al. [25] present integer linear programming (ILP) models to solve the max-total problem for stacking drawings. They then evaluated their results against the max-min heuristic of Cabello et al. [5], which still performs well in terms of the max-total objective, and they show that their ILP solutions are significantly superior to the best heuristic solutions of Cabello et al.

In this thesis, we are concerned only with completely preventing symbol overlap in our proportional symbol map.

**Symbol overlap removal** In the context of graph layout, we can look at algorithms designed to remove node overlaps, which bears strong similarity to our case of removing symbol overlap. Here, we can consider our symbols to be nodes with no edge connections.

Dwyer et al. [10] present a constrained optimization algorithm that generates a linear number of constraints and then finds a solution to these constraints. Marriot et al. [30] have also presented quadratic programming algorithms that solve these constraints. Meulemans [32] removes overlap for square symbols with minimal displacement while maintaining the orthogonal order by rotating the squares 45 degrees using constraint programming as well.

Another approach is the use of force-based algorithms including cluster busting. The cluster busting procedure iteratively relocates nodes in a graph according to some measurable criteria. Lyons et al. [28] present a Voronoi cluster busting algorithm that iteratively forms a Voronoi diagram from the current graph layout and then moves each node to the center of its Voronoi cell until no overlaps remain. However, the nodes in the final drawing are unlikely to bear resemblance to the original graph layout due to the homoge-

neous distribution. Huang et al. [21] present a heuristic method that applies forces to the overlapping nodes such that they push each other away. Lamb [26] displaces nodes based on the amount of overlap between neighboring nodes and preserves the location of non-overlapping symbols by modifying a proximity stress model algorithm designed for graph layouts with overlapping nodes. Gansner and Hu [14] also use a stress model algorithm, where iteratively start with a Delaunay triangulation on the node centers and build the stress model on the edges of the triangulation. Nachmanson et al. [34] present an algorithm that, similarly to the previous paper, also starts with Delaunay triangulation, but it builds a minimum spanning tree on this triangulation to be used for removing node overlaps by letting the tree "grow". In other words, for each child node of some parent node it extends the edge connecting the parent and child if they overlap. Here, we keep the parent in a fixed location, but we translate the sub-tree of the child node.

**Symbol size range** Our intention of providing a large range of symbol sizes implies that we wish for each geographic feature to be able to contain a circle that is as large as possible. This is similar to the circle packing problem of fitting multiple uniformly sized and non-overlapping circles within a rectangle such that their radius cannot increase without causing overlap. Demaine et al. [9] show that such circle packing problems are NP-complete and because there are no deterministic solutions for NP-complete problems, we have to approximate a solution using metaheuristics such as genetic algorithms [8]. As geometric packing problems typically contain a large number of locally optimal solutions, the use of iterative improvement algorithms is ineffective because they become trapped in these local optima. We want to approximate the globally optimal solution by escaping these local optima. Theodoracatos and Grimsley [39] solve the packing of circles in a square using the simulated annealing technique [24], which is a metaheuristic to approximate a globally optimal solution in a large search space for an optimization problem. The main idea behind simulated annealing is that it iteratively attempts to modify the current solution by performing a move to improve the quality defined by an objective function. If this move improves the quality, then it is accepted; otherwise it may still be accepted with a random chance based on how many iterations have passed and the quality difference between the old and new solution. In addition to solving the circle-packing problem with simulated annealing, they also solve the packing of arbitrary simple polygons in a rectangular area by introducing various enhancements to their simulated annealing approach.

**Transforming geographic shapes** Similar to proportional symbol maps, an area cartogram is also a thematic map in which we portray numerical data for comparison between geographic places using visual size. Here, the

geographic size of a region feature is altered instead of using symbols, which trivially introduces shape distortion. We can broadly categorize area cartograms into three classes regarding the preservation of shape and topology [29]: contiguous (preserves topology, distorts shape), non-contiguous (preserves shape, but distorts topology), and diagrammatic (distorts both).

Our problem follows conditions similar to those of a contiguous shape-warping cartogram, also called irregular cartograms [29], as we scale and deform the shape of each region while maintaining adjacent edges. House and Kocmoud [20] present an algorithm for the construction of contiguous area cartograms by viewing the construction as a constrained optimization problem. Their approach follows that of simulated annealing and their scheme alternates goals of achieving correct region areas and adjusting region shapes, allowing for the balancing between statistical accuracy and shape accuracy. However, the algorithm converges quite slowly and can cause significant deformation of the global shape. Gastner and Newman [16] construct a contiguous cartogram by expressing the problem as an iterative diffusion process, where quantities are free to flow from one country to another until the density is the same everywhere. This approach allows for minimal cartographic error while keeping the region shapes recognizable.

For our proportional symbol map, not every region in the input will be shaped nicely to accommodate a relatively large inscribed circle to act as the symbol, so we must transform such region shapes without causing too much distortion. Van Dijk and Haunert [40] have presented an algorithm that enlarges a focus region on a map without removing parts of the map that lie outside this focus region, while minimizing distortion by solving a least-squares optimization problem. Their algorithm is fast enough to be used in real-time interactive applications, is capable of avoiding edge crossings, and can apply the enlargement to multiple areas on a map. The enlargement of the focus region depends on the boundary of the focus region as well as a zoom level parameter. However, when we individually enlarge multiple adjacent regions to accommodate relatively large inscribed circles, we may encounter difficulties with overlapping focus regions which presumably partially undo the enlargement. Alternatively, setting a single focus region containing these adjacent regions would maintain each individual region shape, but those may not be shaped to accommodate large inscribed circles.

# Chapter 2

# Preliminaries

**Polygon** A polygon $P$ is a cyclic sequence of vertices in $\mathbb{R}^2$.

**Simple polygon** A polygon $P$ is called a simple polygon if it adheres to the following properties: each vertex has exactly two edges, the number of edges is equal to the number of vertices, and we have no intersecting edges except for adjacent edges with a common endpoint.

**Regular polygon** A regular polygon is an $n$-sided polygon that is both equiangular and equilateral. The former means that all vertex angles are equal and the latter means that all edges are of equal length.

**Largest inscribed circle (LIC)** Given a simple polygon $P$, the $LIC$ of $P$ is the largest circle that can fit inside $P$. Here we allow the boundary of the circle to touch the boundary of the polygon.

**Bounding box** Given a set of 2-dimensional objects, the bounding box is the smallest axis-aligned rectangle that encloses the set.

**Doubly-connected edge list (DCEL)** A doubly-connected edge list consists of three collections of records [33]. The vertex record of a vertex $v$ stores the coordinates of $v$ in a field called $Coordinates(v)$ as well as a pointer $IncidentEdge(v)$ to an arbitrary half-edge with $v$ as its origin. The face record of the face $f$ stores a pointer $OuterComponent(f)$ to some half-edge on its outer boundary. We do not consider holes in the faces, as there are very few countries that lie entirely within another country, also known as enclaves. Here, we use the example of countries to represent the faces, as they are the input we use in our thesis. And finally, the half-edge record of the half-edge $e$ stores a pointer $Origin(e)$ to its origin, a pointer $Twin(e)$ to its twin half-edge, and a pointer $IncidentFace(e)$ to the face it bounds. We choose the origin such that $IncidentFace(e)$ lies to the left of $e$ when it is traversed from the origin to destination; in other words, we walk the face's

vertices counterclockwise. This record also stores the pointers $Next(e)$ and $Prev(e)$ to the next and previous edge on the boundary of $IncidentFace(e)$.

# Chapter 3

# Problem model

**Input** We assume that our input is a map represented as a connected collection of polygonal regions $\mathcal{P}_{input}$, that is, every pair of regions $P, Q \in \mathcal{P}_{input}$ is connected directly through adjacency $adj(P, Q)$ or transitively through other adjacent regions $\exists K \in \mathcal{P}_{input}(adj(P, K) \wedge adj(K, Q))$. For example, in Figure 3.1 the set of countries in Africa without Madagascar forms a connected collection of polygonal regions, as Madagascar is not connected to any other country. We also exclude polygonal regions with exactly one adjacent region.

Figure 3.1: Set of countries in Africa, where Madagascar is the disconnected red island.

**Constraint motivations** Directional relations between adjacent polygons can prove useful, as relative positions help with the recognition of deformed geographic areas. For example, it should be clear that Denmark lies to the north of Germany, even if the shape of Denmark has been distorted. There have been many studies on directional relation models using various

references such as centroids [19], surface areas [37], and bounding boxes [1, 17, 36]. Forwarding to Section 4.2, we have decided on an approach that iteratively modifies the position of one vertex at a time. Due to these arbitrary transformations to the polygon shapes and unpredictable positions of the LIC in our output, we choose not to use the retention of directional relation as a constraint.

Similarly to contiguous cartograms, our constraint is to preserve the topology by ensuring that the adjacencies of polygons in the input remain unchanged in the solution. In other words, if we consider the input drawing as a planar graph, then the solution must be a planar embedding of the graph with the same cyclic order of edges for the same vertices. To create a relatively bad (mirrored) directional relationship between a reference polygon and some target polygon, we would have to rotate the target around the reference which causes a visual swirl effect as shown in Figure 3.2. Due to the preservation of topology, we would have to move vertices of polygons adjacent to the target and reference polygons to even reach such state. This requires a large overhead in terms of rotating many vertices of different polygons around such reference polygon, which is unlikely to happen within an approach using randomization. Thus, the topology preservation constraint decreases the odds of a significant directional relation distortion throughout the solution as opposed to not having the constraint.

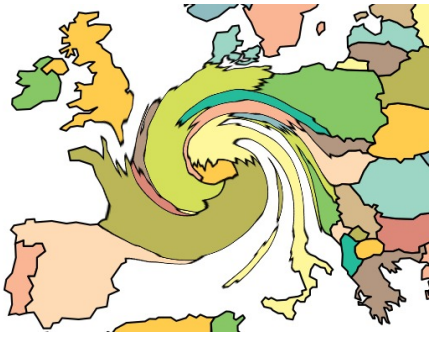

Figure 3.2: Example of the swirl effect on of West-Europe

Finally, we include a bounding box constraint that constrains the output polygons $\mathcal{P}_{output}$ to a specified allowed region $B$. Without this, the optimum solution would be to scale everything uniformly, resulting in a huge drawing. We set the allowed region $B$ to the bounding box of the connected collection of polygons $\mathcal{P}_{input}$.

**Constraints** In summary, we have the following constraints for our model:

- Each symbol $s$ is fully contained within its corresponding polygon $P \in \mathcal{P}$, where we allow the symbol $s$ to touch the boundary of $P$.

- All polygon adjacencies of $\mathcal{P}_{output}$ remain unchanged compared to the polygon adjacencies of $\mathcal{P}_{input}$.
- All polygons of $\mathcal{P}_{output}$ are drawn entirely inside the bounding box $B$.

**Quality measure motivations** Now that we have specified what makes a valid solution, we now discuss when said solution can be considered as a good solution. The region with the smallest LIC will form the bottleneck to how large our range of symbol sizes can be. We refer to this by $LIC_{min}$ and thus our quality measure would be to maximize the radius of $LIC_{min}$.

As we also modify the shape of the geographic areas associated with the symbols, we must take into account the shape similarity of the output polygons $\mathcal{P}_{output}$ compared to the input polygons $\mathcal{P}_{input}$. Most existing approaches to shape similarity measures are focused on only one type of geometric feature from the perspective of a mathematical representation, but this does not capture the complete shape from a visual perspective. For example, region based methods such as grid-based descriptors [27] represent shapes of polygons by focusing on statistically significant features that reflect global information, but reliance on region information leads to loss of the finer contour details. Fan et al. [11] proposed a shape similarity measure based on cognitive levels of the Gestalt psychology [41] using the context and texture of the polygons, as they are the basic elements of visual perception. Here, the context is a statistical grid that provides information on the relative positions of vertices of the contours and the texture information represents the spatial arrangement of intensities in a raster consisting of a matrix of cells organized into a grid. Their approach first extracts the contour on multiple scales and represents it as the context feature, which is then used in a texture analysis method to express its texture. The shape similarity is then obtained by comparing these texture features. They have also shown that their proposed shape similarity measure has superior accuracy over that of the commonly used turning function and Fourier descriptor. Using this measure, we aim to maximize the shape similarity of each output polygon compared to its input polygon.

**Quality measures** Our quality measures are as follows:

- Maximize the radius of $LIC_{min}$.
- Maximize the shape similarity between the input polygons $\mathcal{P}_{input}$ and output polygons $\mathcal{P}_{output}$.

We see that these two measures contradict each other, as the increase to $LIC_{min}$ radius implies the deformation of some polygon's shape which in turn decreases its shape similarity to its input polygon. We pay attention to this trade-off in the design of our algorithm to allow for a controlled balance between the two measures.

# Chapter 4

# The algorithm

In this chapter we first provide a high level overview of our algorithm and then discuss our design decisions to satisfy the constraints of the problem model. We shall use the term of faces and polygons interchangeably as they imply the same thing. In the subsequent sections we will discuss our approach to the quality measures of the problem model in more detail.

**Overview** Given a map consisting of connected polygonal regions as input, we first simplify the map by removing vertices and edges whose removal does not change the topology of the map and store their positions relative to the remaining simplified edges. We then transform the polygons by iteratively modifying the location of one vertex at a time following the simulated annealing technique. Here, the objective is to increase the $LIC_{min}$ radius as well as somewhat retaining the shape similarity between the simplified and transformed polygon. Once the simulated annealing terminates, we set every polygon's LIC radius to that of the $LIC_{min}$ and we reconstruct the shape by reinserting the removed vertices and edges without causing any overlap. For our final step, we recompute and update all the LICs once more, as this can only further increase the radius of $LIC_{min}$, which is then used as the maximum symbol radius for the proportional symbol map.

**Design decisions** Our problem model's first constraint is that all symbols must be fully contained within its corresponding polygon. This is always satisfied due to the definition of a polygon's LIC and the symbol size cannot exceed that of the $LIC_{min}$.

Next, we want all polygon adjacencies to remain unchanged in our solution. For every face, we know its adjacent faces by walking over the boundary and looking at the incident face of the twin half-edge. At the same time, we know in which order these adjacent faces appear and we can maintain the adjacencies by keeping at least one half-edge per adjacent face. We do not add additional vertices to a face, as it would function as a vertex interpolation that splits edges into smaller edges that are collinear, which does not

improve the shape similarity of the solution as the shape is essentially still the same. But it could be used to enable the LIC to become larger for polygons with very few vertices, as regular polygons with more vertices trivially have a better ratio of area to LIC radius compared to regular polygons with fewer vertices, since their shape becomes closer to that of a circle. However, since we use countries as input, it is highly unlikely that any of our input regions have a boundary of less than say five vertices, therefore we neglect this possibility. In fact, adding more vertices means that we will have a larger search space of vertices to move to another location to maximize the $LIC_{min}$ and it also increases the time needed to compute a polygon's LIC. Thus, for map simplification, we would like to have the minimal number of vertices per polygon while still maintaining its adjacent polygons in the correct order to lower the search space for simulated annealing. For every moved vertex, we would also have to recompute the LIC of all faces incident to the moved vertex, which can become quite computationally expensive, as we will require a large number of the said moves in later steps of our algorithm.

Now, we want to move vertices to increase the radius of $LIC_{min}$. If we were to disregard the polygons and only consider circles as the LICs, we would see that the optimal solution to maximize the $LIC_{min}$ is the circle packing problem of finding the maximum radius of $n$ uniform sized and non-overlapping circles and their arrangement such that they fit within a rectangle. As discussed in Section 1.2, this problem has been solved using the simulated annealing technique. However, our situation is more complex, as our circles are defined as the LIC of every polygon, and we also maintain the adjacencies of every polygon. We take care of these additional complexities by taking them into account in the objective function as well as in the iterative moves, and thus we also make use of simulated annealing.

At the end of our simulated annealing algorithm, we set every polygon's LIC radius to that of $LIC_{min}$ and lock its position. This represents the case where all regions have the same numerical value and hence have the same symbol proportions. Now, we want to reconstruct the original shapes of the underlying geographic areas for the shape similarity quality measure. We do this by reinserting the vertices that were removed during the map simplification to the locations they were at relative to the simplified edge. However, many vertices cannot be placed at such locations without overlapping the symbols or creating edge crossings. Our solution is to define a unique free space polygon $P_{free}$ for every simplified edge such that the reinsertion of the original vertices within $P_{free}$ will not break our model constraints. We then reconstruct the shape of the original border within this $P_{free}$ with an image resizing approach. Once we have performed this shape reconstruction for all simplified borders, we terminate our algorithm. Now we can use the drawing of the resulting $DCEL$ as our geographic map and every polygon's symbol can be resized for the display of some numerical data.

We see that the simulated annealing and shape reconstruction steps deal with the two quality measures of the problem model. These two measures contradict each other, as we want $LIC_{min}$ to take up as much area as possible within the bounding box, but at the same time, this decreases the total available free space for shape reconstruction. In Section 5.2, we will evaluate this trade-off and discuss its results.

## 4.1 Map simplification

We begin with our input which is a set of countries that will be represented as a $DCEL$ and we define our bounding box for this input. Some countries consist of multiple disconnected landmasses, such as the United Kingdom or Italy. In these cases, we simply take the largest polygon of the country to represent the country itself. There are more extreme cases, such as the Philippines and Indonesia, where there is no visible larger polygon to represent the country. We consider this to be an edge case along with the case when we have islands in our input.

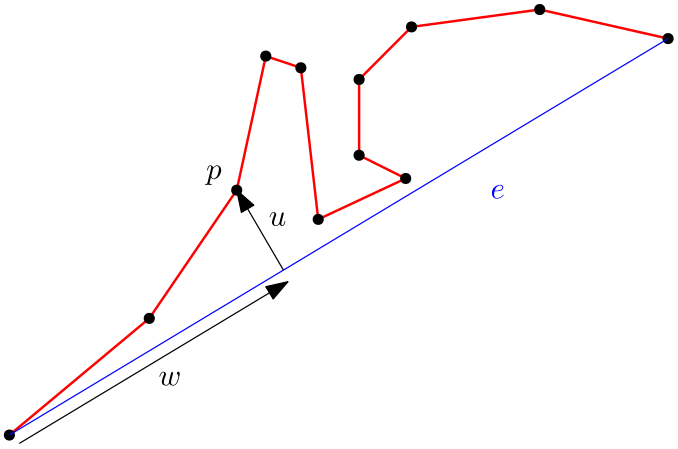


Figure 4.1: Calculation of the parallel and perpendicular offsets for vertex $p$ w.r.t. simplified edge $e$.

So the first step of our algorithm is to lower the number of vertices in every face by simplifying its shared borders with adjacent faces. Consider a shared border as a consecutive sequence of half-edges, where all half-edges point towards the same incident face and their twin half-edges also point to a same incident face. The simplified shared border $e$ consists of a single half-edge whose endpoints are the first and last vertex of the shared border. For every simplified border $e$, we store the offset position of every removed vertex $p$ relative to $e$ in a list $V_{rm}$. So, for every removed vertex $p \in V_{rm}$ we calculate the parallel vector $w$ and the perpendicular vector $u$ with respect to the simplified border $e$ as shown in Figure 4.1, and we store the relative

parallel and perpendicular distances $W = \frac{||w||}{||e||}$ and $U = \frac{||u||}{||e||}$. These values will be used later in the shape reconstruction step in Section 4.3.

## 4.2 Simulated annealing

Our objective is to maximize the $LIC_{min}$ radius using the simulated annealing technique by iteratively moving the vertices in the $DCEL$ to new positions. We shall refer to such iterations as epochs. Our initial solution is the state of our $DCEL$ after the map simplification step and our requirement is that the drawing of this $DCEL$ remains valid, that is, there must be no edge crossings and all vertices must be contained within the bounding box. Following the standard practice for simulated annealing approaches, we use a linear cooling schedule based on the iteration count [23]. Let the quality of the current and new solutions be $q$ and $q'$ respectively. The probability of accepting the new solution is 1 if $q' < q$, otherwise it is $exp(-\frac{q'-q}{T})$. Here, $T$ is the temperature variable whose value decreases based on the cooling schedule.

**Design choices** The amount of surface area we have within our bounding box is limited, we want the $LIC_{min}$ to take up as much space as possible, but all LICs are contained within some simple polygon $P$ that trivially has a larger surface area. To increase the radius of a LIC, we would need to increase the area of $P$ or reshape the perimeter of $P$ to be a regular polygon. Simply increasing the area of the polygon does not necessarily increase the LIC radius. For example, consider a simple square polygon whose area we can infinitely increase by stretching it horizontally while the LIC radius remains the same. Regular polygons have the maximum ratio of $\frac{area(LIC)}{area(P)}$, as the number of vertices in a regular polygon increases, it naturally approaches the shape of a circle. Thus, it is desirable to move the vertices so that the polygons approach the shape of a regular polygon.

Trivially, concave polygons stray further away from regular polygons in terms of their shape and the ratio of $\frac{area(LIC)}{area(P)}$ gets worse the more concave a polygon is. To discourage such moves that make polygons more concave, we include polygon concaveness in our measures. We do not want this measure to be scale invariant, as the concaveness of a large polygon can have a larger impact on the LIC radius compared to a small polygon. This means that the value range of this measure can be arbitrarily large for different inputs.

Now consider the case where all the polygons in our current solution are regular polygons, but the area sum of these polygons is merely a fraction of the area of the bounding box. Any vertex moves made to increase a polygon's LIC will deliver a worsened quality to the new solution due to the regularity measure and possibly the concaveness measure as well. To encourage this move, we consider increasing the ratio of area sum of all LICs

and the area of the bounding box as an improvement to the new solution.

During the initial iterations of simulated annealing, we are likely to accept moves that may cause polygons to become rather concave as part of escaping a local optimum. At the same time, this opens possibilities to position vertices such that their incident edges form rather narrow angles. This, in turn, will severely limit the available locations to which we can move adjacent vertices in subsequent iterations. We also generally do not desire the presence of narrow angles in our $DCEL$, as the smallest angle that a regular polygon can have is $\frac{\pi}{3}$ for a triangle, and this increases monotonically with the more vertices a regular polygon has. Therefore, we consider an angle to be narrow if it is smaller than $\frac{\pi}{6}$. However, it may be the case that the polygons already contain narrow angles after map simplification, but for the sake of increasing the radius of $LIC_{min}$ we still aim to remove them.

Finally, as the iterations pass, we get polygons whose regular shape does not accurately reflect its simplified shape after map simplification. We end up with a solution that indeed has a larger $LIC_{min}$ radius, but the recognizability of the polygons suffers. We will try to maintain some similarity between the polygon shape before and after simulated annealing. The shape similarity measure used in our problem model would be excessive for the few vertices that the polygons have here, as it performs an in-depth computation of the shape similarity by viewing its shape at multiple scales from global to local view. Instead, we use a turning function measure for shape similarity between two polygons, as it is reasonably easy to compute and its answers match that of human intuition [2]. This works for both convex and concave polygons, and it is translation, scale, and rotation invariant. A drawback of this shape measure is that it is highly sensitive to noise in the form of relatively small spikes in the contour. However, noise will not occur in our case as we always compare two polygons with the same number of vertices.

**Quality measures** We measure the quality of our solution as a weighted sum of the following quality measures, all values are non-negative and we consider lower values approaching zero to be of higher quality:

**C.1:** The maximum area that each polygon's LIC can have is upper bounded by $A = \frac{area(boundingbox)}{n}$, where $n$ is the number of polygons and $0 < area(LIC_{min}) < A$. To maximize the $LIC_{min}$, we want its area to approach the maximum area it can be. Thus, we measure $\frac{A - area(LIC_{min})}{A}$, where the value ranges from $(0, 1)$.

**C.2:** We measure the area difference $X$ between the bounding box and the area sum of all polygon's LICs denoted by $S$, normalized to the bounding box area. $X = \frac{area(boundingbox) - S}{area(boundingbox)}$, where the value ranges from $(0, 1)$.

**C.3:** Let $P$ denote a convex polygon and $n$ the number of vertices in $P$, we measure a polygon's regularity using the area and perimeter ratio [6] as $1 - \frac{area(P)}{(perimeter(P))^2} 4ntan(\frac{\pi}{n})$. This returns a value within the range [0, 1), where zero means that $P$ is a regular polygon. We do not apply this measure to concave polygons, as the measure is derived from alternative definitions of $\frac{area(P)}{area(R)}$, where $R$ is a regular $n$-gon with the same perimeter as $P$. Thus, we return a value of 1 for every concave polygon and use the concaveness quality measure (C.4) to evaluate concave polygons. We then take the average regularity measure for all polygons.

**C.4:** Let $P$ denote a polygon and $CH$ denote the convex hull of $P$. We measure the concaveness of each polygon on the basis of the difference between its convex hull perimeter and the polygon's perimeter. $\frac{perimeter(P) - perimeter(CH)}{perimeter(CH)}$ returns a value within the range [0, $\infty$), where zero means that the polygon is not concave. We then take the average concaveness measure for all polygons.

**C.5:** For every vertex $u \in V$, we compute the angles that are formed by its incident edges. If an angle $\alpha$ is smaller than $\frac{\pi}{6}$, then our measure returns $(\frac{(\pi/6) - \alpha}{\pi/6})^2$ with a value range of [0,1), where narrower angles approach value 1. Otherwise, for angles greater than or equal to $\frac{\pi}{6}$, we return zero. This measure then returns the sum of these narrow angle values.

If we used linear scaling here, we would set the weight to a high value, so that we would only allow narrower angles to be formed when the temperature was high. As the temperature decreases, it would become nearly impossible to ever accept a move that forms even the slightest narrower angle due to the high weight value. Using a lower weight value would enable acceptance of such moves, but it allows for extremely narrow angles to be created at high temperatures which will severely restrict subsequent moves. So instead, with quadratic scaling and a low weight, we will prevent extremely narrow angles to be created at high temperatures and yet allow moves that create slightly narrower angles when the temperature is lower. We only use this for the narrow angles measure, as narrow angles are prone to blocking our search for the optimal solution for subsequent epochs.

**C.6:** For each polygon $P$, we compare its turning function $\Theta_P$ with the turning function of its simplified polygon after map simplification $\Theta_{P'}$. The shape similarity is measured by computing the distance between the two turning functions, minimized with respect to the rotation of the polygons [2]. The value ranges from $[0, \infty)$, where zero means that their shape is identical. We return the sum of these turning function

distances for all polygons.

**Initial solution** We start by computing the quality of the initial solution, that is, the $DCEL$ after map simplification. To compute the LIC of every polygon, we need the following definition. The center of a polygon's LIC is also known as the *pole of inaccessibility*, which is the point that is the most distant from any point on the boundary of the polygon. This can be computed exactly for convex polygons using linear programming [31], but since we also work with concave polygons and do not require exact solutions, we look towards the randomized method. Garcia-Castellanos et al. [15] have presented an approximation algorithm that calculates this point by probing the polygon with points placed on an arbitrary sized grid distributed in the bounding box for the polygon. It then calculates the distance for every point to the polygon's boundary, selects the point with the longest distance, and repeats the process until the desired precision is achieved. In our implementation, we use a library inspired by this paper that gives us the center and radius of the LIC, but uses quadtrees instead by recursively subdividing cells and probing them for a better solution[1]. We compute this LIC for every face in our $DCEL$ and store its center position and radius in the face itself. We can find the $LIC_{min}$ by sorting by radius and we store this reference in the $DCEL$. The temperature $T$ is set to the maximum temperature $T_{max}$ and we calculate the quality of the solution with all our quality measures.

**Modification** To modify the current solution to a new one, we need to select a vertex to move to another position. We refer to this modification as an epoch. After every epoch, temperature $T$ decreases as per the linear cooling schedule. When the temperature is high, we pick an arbitrary vertex in our $DCEL$ to move, but as the temperature decreases we increase the likelihood of selecting a random vertex that lies on the boundary of the face with $LIC_{min}$ or a vertex adjacent to this. Let $k$ be a randomly generated number between (0,1) and let face $f_{min}$ be the face containing $LIC_{min}$, we regenerate $k$ and select to move a random vertex if $k < 0.05 + (\frac{T}{1.2T_{max}})^2$. If this condition fails, we select to move a random vertex adjacent to any vertex of $f_{min}$ if $k < 0.05 + (\frac{T}{1.1T_{max}})^2$. Otherwise, we select a random vertex of $f_{min}$. Observe that we always have at least a five percent chance to select a vertex that is not part of $f_{min}$, this choice was made to still allow shape modifications at low temperatures of faces that are not $f_{min}$. This is in accordance with the principles of simulated annealing, where lower temperatures make the technique behave as a greedy approach, yet we still

[1] A new algorithm for finding a visual center of a polygon. V. Agafonkin. https://blog.mapbox.com/a-new-algorithm-for-finding-a-visual-center-of-a-polygon-7c77e6492fbc Date of access: 13-03-2022.

maintain some degree of randomness.

Once we have selected a vertex $p$ to move, we must compute the positions to which it can be moved so that no edge crossings occur. In other words, the vertex $p$ must be in a location that is visible directly from all adjacent vertices. Let $p_{adj}$ denote some vertex adjacent to $p$. We want to compute the visibility polygons $P_{vis}$ for all $p_{adj}$ without edges incident to $p_{adj}$ and take their intersection, any location within the resulting polygon $P_{res}$ can be used to reposition vertex $p$ without creating edge crossings.

Calculating the visibility polygon can be done in $O(n)$ time for simple polygons [22], however, we do not have a simple polygon if $p_{adj}$ is incident to the outer face as shown in Figure 4.2.

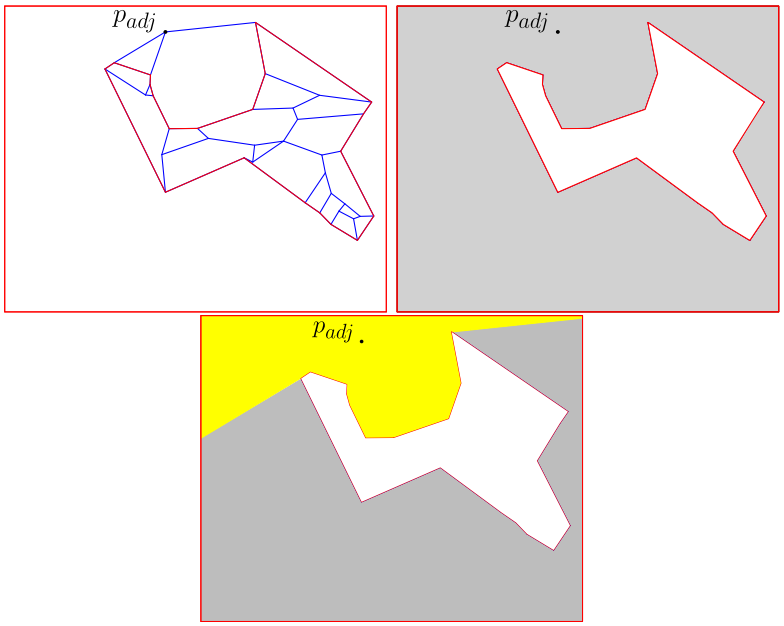


Figure 4.2: Visibility polygon example where vertex $p_{adj}$ is not in the interior of a simple polygon, but in a polygon (grey) with a hole. The red edges define the interior and exterior boundaries of the polygon.

Instead, we use the angular plane sweep algorithm of Asano [3]. This algorithm runs in $O(n \log n)$ time and $O(n)$ space and uses line segments as input. First, we get the set of line segments $L$ that are possibly visible from $p_{adj}$, these are the perimeter segments of the faces incident to $p_{adj}$ as well as the bounding box itself. To retrieve $L$, we take a half-edge $e$ of $p_{adj}$ and walk its boundary until we have $e.destination == p_{adj}$, then take the twin half-edge $e = Twin(e)$ and repeat the process until we have handled the boundaries of all incident faces. We also add the four sides of the bounding box to $L$ and we remove segment $(p, p_{adj})$ from $L$. Given this set $L$ of line segments, we must provide a viewpoint $z$ that lies in general position,

that is, not on any vertices in $L$. However, we always compute $P_{vis}$ where $z == p_{adj}$, so we slightly nudge the position of $z$ towards $p$ as a workaround. A faster approach that does not require this workaround would be to use the triangular expansion algorithm by Bungiu et al. [4], which has a worst case complexity of $O(n^2)$, but in practice its average runtime is two orders of magnitude faster than the sweep algorithm. We leave this open for future works to improve the runtime of simulated annealing.

Next, we move vertex $p$ to a random point inside $P_{res}$. We determine this point by sampling random points in the bounding box for $P_{vis}$ until it is inside $P_{vis}$. After moving the vertex, we efficiently update the quality measures by only recomputing the affected objects and taking the measure differences into account for our values. For $LIC_{min}$ and area difference we recompute the LIC of the polygons incident to the moved vertex $p$ and update $LIC_{min}$. We also recompute the measures of regularity, concaveness, and turning function distance for the incident faces. Finally we update narrow angles by computing the narrow angles for vertex $p$ and all of its adjacent vertices $p_{adj}$.

## 4.3 Shape reconstruction

Now that we have handled the quality measure of maximizing the $LIC_{min}$, we focus on the quality measure of maximizing the shape similarity of the polygons in the output drawing compared to the input. Our goal is to retrieve the shape features back in the simulated annealing output polygons $\mathcal{P}_{SA}$.

**Definitions** Recall that each edge $e$ of polygon $P \in \mathcal{P}_{SA}$ represents a simplified shared border with another country or the sea. Let $e_{start}$,$e_{end}$ denote the start and end points of the edge $e$, and let $\vec{e_{\parallel}}$,$\vec{e_{\perp}}$ denote the direction vectors parallel and perpendicular (rotated $\frac{\pi}{2}$ counterclockwise) to the edge $e$ respectively. The original shape of each border is restored by reinserting the vertices in $V_{rm}$ that were removed during map simplification in Section 4.1, back into edge $e$. For every reinsertion vertex $p \in V_{rm}$, we calculate its position as $e_{start} + (\vec{e}_{\parallel} \cdot p.U \cdot ||e||) + (\vec{e}_{\perp} \cdot p.W \cdot ||e||)$, where $p.U$ and $p.W$ are the relative distances parallel and perpendicular to $e$. We define path $T = (p_1, ..., p_n)$ of $n = ||V_{rm}||$ consecutive reinsertion vertices and the entire reconstructed border including its endpoints is the path $T_{full} = (e_{start}, p_1, ..., p_n, e_{end})$.

**Free space polygon** As we wish to reconstruct the original shape by reinserting these vertices to their relative positions, it is important that we do not cause any edge crossings or overlap any symbols. To ensure that we do not break these constraints, we compute a free space polygon $P_{free}$ for every

edge $e$ such that the intersection of all free space polygons is empty and $e$ is inside $P_{free}$ as shown in Figure 4.3.

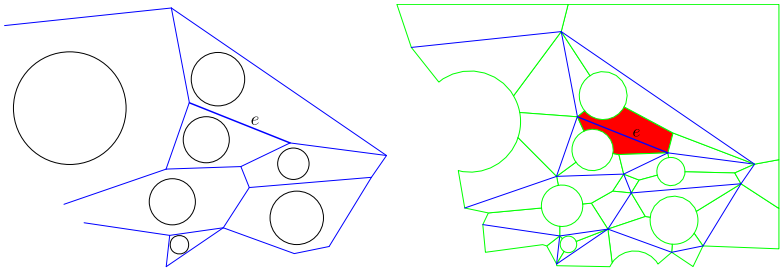


Figure 4.3: To the left we have partial view of the $DCEL$ and to the right we see the green outlines of the free space polygons for every edge. The free space polygon $P_{free}$ of edge $e$ is marked red.

Each edge $e$ has two adjacent free space polygons $P_{f1}, P_{f2}$ that join at $e$, one for each incident polygon. Free space polygon $P_{free}$ is the union of $P_{f1}, P_{f2}$ and will be used for the shape reconstruction. To compute $P_{f1}$, we look at edge $e$ in the context of one incident polygon and observe that its free space is either limited by the polygon's symbol or by another edge in the polygon. We temporarily disregard the polygon's symbol and focus on creating a planar subdivision of the polygon where each edge of the polygon belongs to exactly one unique face in the subdivision. To divide the free space for adjacent edges somewhat evenly, we take the interior bisector of the polygon's vertices to define the subdivision faces with. This idea coincides with the concept of the medial axis, which is a set of all points having more than one closest point on the polygon's boundary. Equivalently, it can be seen as the points that can be the center of a circle that lies entirely within the polygon and touches the polygon's boundary in at least two places. However, the medial axis of concave polygons can have curves which have compatibility issues in our implementation. In addition, we will have to create a planar subdivision for the polygon using the medial axis so that every edge is assigned a unique face. Therefore, we choose to use straight skeletons instead.

The straight skeleton of a polygon is defined by shrinking the polygon by translating each of its edges inwards parallel to themselves at a fixed rate. As the polygon shrinks, the vertices also move at a rate depending on the angle of the vertex. Once a moving vertex collides with a non-adjacent edge, the polygon is split in two by the collision and the shrinking process continues in each part. The straight skeleton is thus the set of curves traced out by the moving vertices during this shrinking process. It is identical to the medial axis for convex polygons, but for concave polygons the straight skeleton has fewer edges than the medial axis, and all of its edges are line segments. To compute the straight skeleton we use an algorithm by Felkel

and Obdrzálek [12] that supports both convex and concave polygons as well as polygons with holes. We then define $P_{f1}$ as the face incident to edge $e$ in the planar subdivision formed by the straight skeleton, minus the area of the polygon symbol. Here, we have set every polygon symbol radius to be equal to 99% of the $LIC_{min}$ radius, because if we use the full radius of $LIC_{min}$ some polygons will have their LIC touching its boundary, which causes a split in the free space polygon $P_{f1}$. We follow the same process for computing $P_{f2}$. When we need to compute the free space in the outer face of our $DCEL$ we can consider an enlarged version of the bounding box as the polygon and the floating component as the hole.

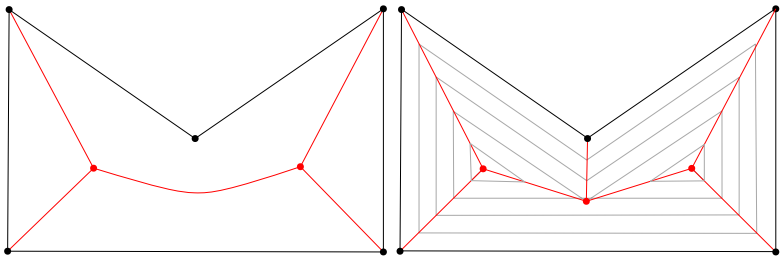

Figure 4.4: Medial axis (left) compared to the straight skeleton (right).

Now that we have $P_{free}$ for the edge $e$, we want to create $T_{full}$ such that all its edges are inside $P_{free}$. Ideally, if the vertices and edges of $T$ fit inside $P_{free}$ as well as edges $(e_{start}, v_1), (v_n, e_{end})$, then we can simply reinsert $T$. Otherwise, we look at two options in which we can make use of this free space for the shape reconstruction. The first is based on the technique of image resizing, and the second is based on image warping. The former is implemented in our algorithm, whereas the latter remains open for future work due to difficulties with its approach.

**Image resizing** Let $B_T$ be the bounding box of path $T$ and consider $B_T$ to be an image with $T$ as its contents. We compute the largest inscribed rectangle LIR within $P_{free}$ with the following two constraints. First, the LIR must touch or intersect the edge $e$. Second, the LIR must be aligned to the edge $e$. The width of the LIR is at most equal to the length of edge $e$, whereas the height is limited by the bounding box. In most cases, the width of $B_T$ ends up being close to or larger than that of the LIR. Thus, we resize the width of $B_T$ to that of the LIR. The height is then set based on the aspect ratio of $B_T$, however if this height exceeds that of the LIR, we will set it to the height of the LIR instead.

Now that we have resized the dimensions of $B_T$, we know that path $T$ is entirely contained within $P_{free}$. However, we still need to create edges $(e_{start}, p_1), (p_n, e_{end})$ to obtain $T_{full}$ and these could cause a crossing with $P_{free}$. If $height(B_T) < height(LIR)$ we can translate $B_T$ in the

direction perpendicular to edge $e$ to create more visually appealing edges $(e_{start}, p_1), (p_n, e_{end})$. That is, we translate $B_T$ such that we minimize the sum of distances of $v_1, v_n$ perpendicular to edge $e$.

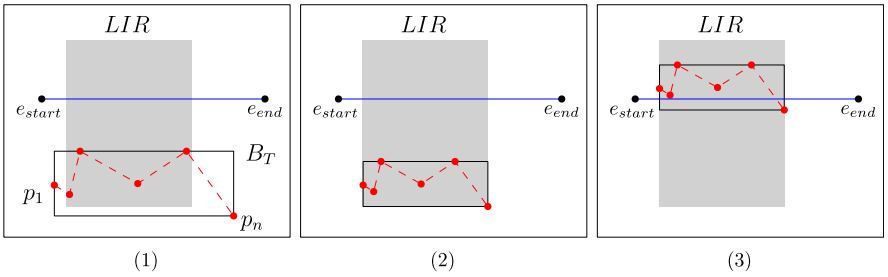


Figure 4.5: Example of resizing $B_T$ given an edge $e$ (blue), a path $T$ (red), and a LIR (grey). (1) Initial size and position of $B_T$. (2) Resized the dimensions of $B_T$ to fit in the LIR. (3) Translated the position of $B_T$.

However, to guarantee that there will be no crossings created for edges $(e_{start}, p_1), (p_n, e_{end})$, they must both be parallel to edge $e$ since we know that $e$ is always inside $P_{free}$. This is not always possible, as the endpoints of $e$ are locked in position. Instead, we add two intermediate vertices $p_k, p_l$ that are both on edge $e$ to replace $(e_{start}, p_1)$ with $(e_{start}, p_k), (p_k, p_1)$ and $(p_n, e_{end})$ with $(p_n, p_l), (p_l, e_{end})$. Vertex $p_k$ is the closest point on the boundary of $B_T$ to $e_{start}$ and similarly vertex $p_l$ is the closest point from $B_T$ to $e_{end}$. We can see a clear downside to this approach when the distance of $p_k$ and $p_l$ perpendicular to $e$ is rather large, causing a sudden spike in our contour as shown in Figure 4.6.

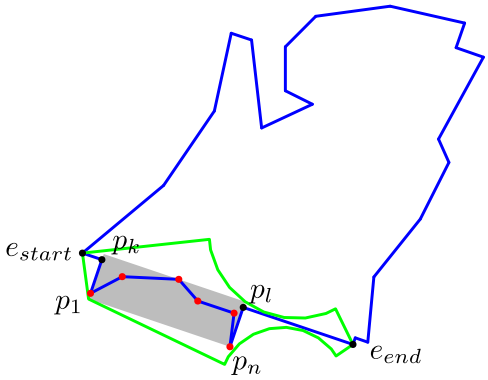


Figure 4.6: Example of a spike after shape reconstruction using intermediate vertices $p_k, p_l$ (black) to connect to path $T$ (red), where free space polygon $P_{free}$ for edge $e$ is shown with the green boundary and its LIR == $B_T$ is shown in grey.

The possibility of a crossing is now only present for edges $(p_k, p_1)$ and

$(p_n, p_l)$. We have a crossing between two edges $(a, b), (c, d)$ if $c, d$ lies on opposite sides of $(a, b)$ and $a, b$ also lies on opposite sides of $(c, d)$. We perform this crossing check for edge $(p_k, p_1)$ to find the last crossing edge $(p_i, p_{i+1})$ for $i = 1, ..., n-1$, and then we remove vertices $(p_1, ..., p_i)$ to connect $(p_k, p_{i+1})$ instead which has no crossings. Similarly for edge $(p_n, p_l)$, except we have $i = n, ..., 2$, we remove vertices $(p_n, ..., p_i)$, and we connect $(p_{i-1}, p_l)$.

The downside of this shape reconstruction technique is that the LIR does not optimally utilize the area of $P_{free}$ when $P_{free}$ is not quite rectangular, and consequently we could significantly downscale $B_T$ such that we lose out on the visibility of the finer details of the shape. Also, as shown in Figure 4.6, this approach is prone to causing spikes in the reconstructed shape.

**Image warping** The other approach follows the idea of image warping, where we do not restrict the transformation of $B_T$ to maintain a rectangular shape and it can be of the same shape as $P_{free}$. Swart [38] presents an algorithm that allows the warping of an arbitrary shaped image to that of a different arbitrary shape. This can cause more distortion in the reconstructed shape as we fully utilize the shape of $P_{free}$ which can cause the reconstructed shape to follow the shape of $P_{free}$ more than the original $T$.

The idea behind the image warping algorithm is that we create a mesh overlay of, for example, 16 x 16 points for $B_T$ and $P_{free}$. Here, we consider the mesh over $B_T$ to be the source mesh that is a square lattice with its points evenly distributed over $B_T$, and the target mesh is the mesh over $P_{free}$. Initially, the target mesh is the same as the source mesh, but we mark every point of the mesh that lies on the mesh boundary as a boundary point. These boundary points are always placed on the boundary of $P_{free}$. We then change the shape of the target mesh into that of $P_{free}$, by iteratively taking a random point in the mesh and setting it to the average position of its neighboring points. For boundary points, we also compute the average position of its neighbors, but set it to the closest point on the boundary of $P_{free}$. Once the algorithm has converged, we map the points from the quadrilateral faces in the source mesh to the target mesh, using some convenient interpolation scheme.

Although this algorithm works well for most arbitrary shapes according to the paper, we notice that it fails to create a target mesh that covers narrow areas of a polygon. As boundary points are always set to the closest point on the target polygon's boundary, it fails to expand the mesh point towards narrow areas as we can see in Figure 4.7. Locking boundary points to specific parts of the boundary would have to be done in a sophisticated manner, as its positioning and the density of points around certain contours will define the mesh.

Another problem we encounter is that the quadrilaterals of the target mesh are not always fully contained within $P_{free}$, especially around concave

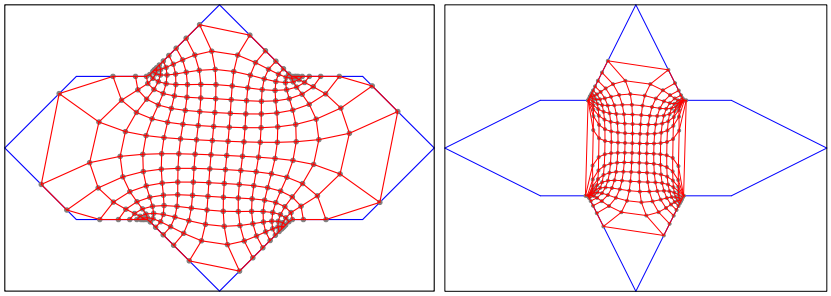

Figure 4.7: Example where the warped mesh cannot cover narrower areas.

parts of the polygon as we can see in Figure 4.8. To prevent the mapping of points in the source mesh to the target mesh from ending up outside the target polygon, we would need additional case handling for whichever interpolation scheme we choose to use.

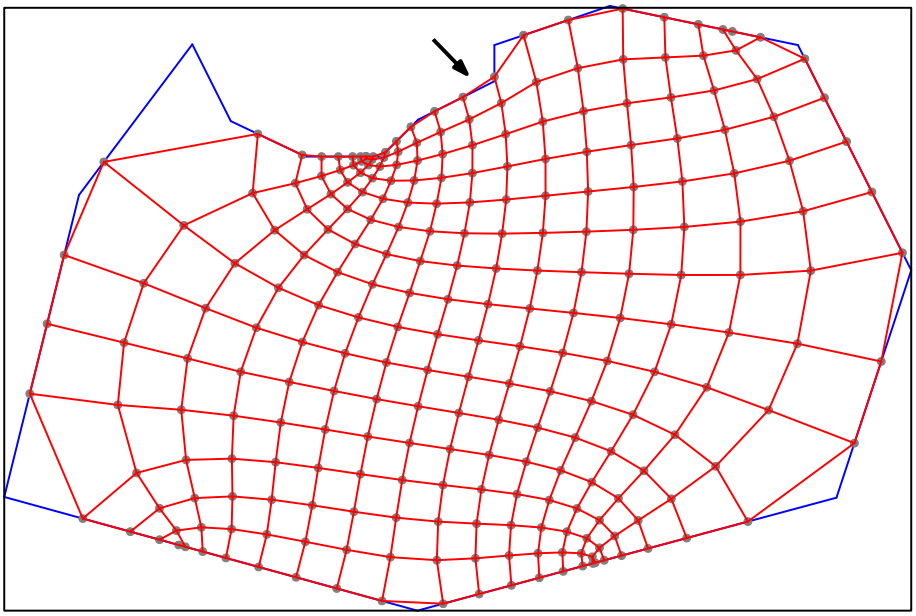

Figure 4.8: Example where quadrilateral faces of the target mesh are not fully contained within the target polygon.

Despite these issues, we have decided to leave this approach open for future works, as it seems like an interesting approach to fully utilize the free space polygon for shape reconstruction.

# Chapter 5

# Evaluation

In this chapter we discuss the experiments we have performed to evaluate the results obtained with the simulated annealing step and the shape reconstruction step. At the end of each section, we observe and discuss the visual quality of the output.

## 5.1 Simulated annealing

For the evaluation of our simulated annealing approach, we experimented with how weights influence the results, as well as how the quality measures interact with each other.

**Data** We run the experiments on a subset of countries in Western Europe, South-America and the Middle-East as shown in Figure 5.1. These subsets consist only of connected countries, that is, there are no islands. For the Western Europe dataset, the smallest few countries are surrounded by other countries and they are not directly adjacent to the outer face. For these countries to expand in size to accommodate a larger LIC, the simulated annealing technique must move the neighboring countries away. In the South-America dataset, we see the opposite, where the smallest countries are adjacent to the outer face and can expand their area in that direction. For the Middle-East dataset, we observe that it has relatively many narrow angles after map simplification and its overall shape is quite concave.

**Quantitative evaluation** We ran the simulated annealing for 30,000 epochs with different weight values. Due to randomization, we repeat every experiment 5 times. The execution time per experiment takes about 8 to 12 s on a 3.6 GHz AMD Ryzen 5 3600 CPU.

**Weights** We first set a default value for every weight and then vary these weights to evaluate its influence on the various quality measures. As the

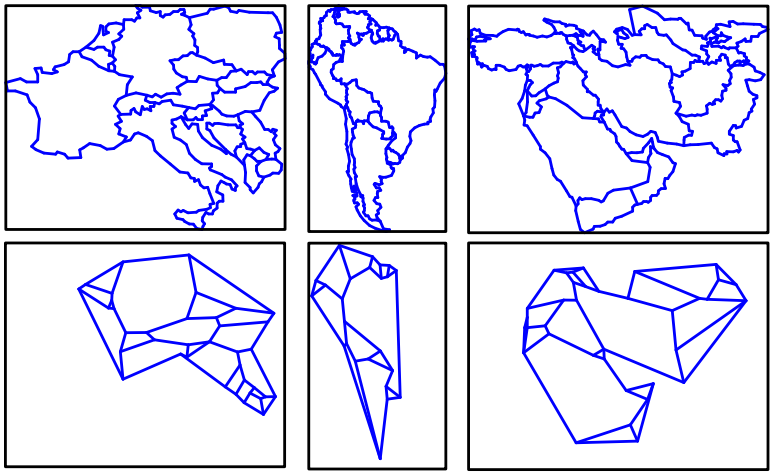

Figure 5.1: The data sets used for experimenting where the top is the original input and the bottom is after map simplification. We have subsets of Western Europe (left), South-America (middle), and the Middle-East (right).

parameter space is too large to experimentally explore, we set the default weights based on informal trials where we observed the visual results.

| Measure | Weight | Measure | Weight |
|---|---|---|---|
| C.1 $LIC_{min}$ Diff | 600 | C.4 Concaveness | 10 |
| C.2 Area Diff | 10 | C.5 Narrowness | 500 |
| C.3 Regularity | 10 | C.6 TF Distance | 400 |

Table 5.1: Default weights of the quality measures.

Our primary goal is to maximize the $LIC_{min}$, but we limit our chances of reaching the global optimum if many narrow angles are created, which subsequently cause the availability of new vertex locations within the epoch to decrease substantially. Therefore, we set a high weight on narrow angles (C.5) to avoid these scenarios.

Our secondary goal is to have the resulting polygon's shape being similar to that of the simplified polygon for a better shape reconstruction in the next step of our algorithm. To effectively allow the increase of the $LIC_{min}$ radius, we need to expand out polygons to take up space in the outer face. These iterative moves will negatively impact the shape of the polygons (C.6) at the time of the epoch and are therefore discouraged; to counteract this, we use the area difference measure (C.2) that encourages the expansion of polygons.

Alternatively, we can completely disregard the simplified polygon shapes and focus only on the primary goal. This means that we would want every polygon to approach the shape of its regular polygon to allow maximum efficiency in the ratio of area and LIC. For this, we have the regularity (C.3)

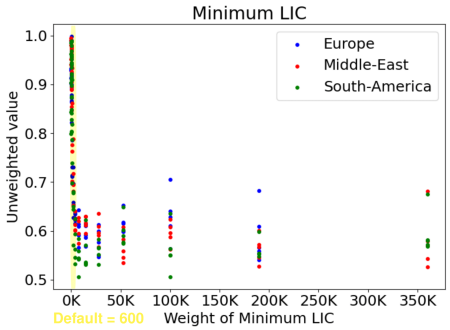


Figure 5.2: Varying weight of $LIC_{min}$ (C.1).

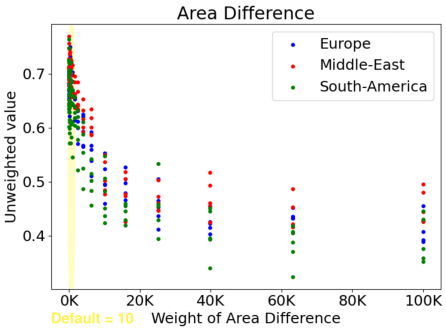


Figure 5.3: Varying weight of Area Diff (C.2).

and concaveness measures (C.4).

We validate our weights by varying each weight exponentially in an interval around our chosen default value, while keeping the other weights fixed. In Figures 5.2 and 5.3 we see some results in which we plot the unweighted value of the quality measure as a function of the corresponding weight. Observe that setting the weight higher for C.1 does not necessarily improve its value anymore starting from a weight around 9000. This is because within each epoch we can only see improvements to this measure if we were to move a vertex that is incident to the face $f_{min}$ containing $LIC_{min}$, but to increase this $LIC_{min}$ we may need to move surrounding polygons first to enable $f_{min}$ to become larger. In general, as we increase the weight of a quality measure, its unweighted value is expected to become lower (improve). We see the same pattern of converging and stabilizing values as the weights increase for the other quality measures as well, see Appendix A for the full figure. For our chosen weights, we see that the values are not quite stable, but we shall not simply increase the weight to the point where it starts to stabilize, as we expect there to be trade-offs between the measures. Instead, we will look at the dependencies between the quality measures by looking at how the change of one measure's weight affects the unweighted value of another.

**Quality measure dependencies** There are many quality measure pairs where changing the weight of one has a visible effect on the value of another. We explore these dependencies by looking at plots in which we vary one weight at a time and how it impacts the unweighted value of every other quality measure. The full figures for all plots are available in the Appendix A.

We start by varying the weight of $LIC_{min}$ (C.1), here we see that increasing the weight causes a noticeable increase (deterioration) in other quality measure values except for area difference (C.2). In fact, we observe a cor-

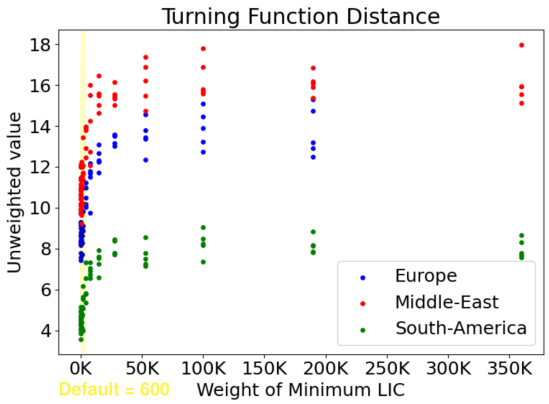


Figure 5.4: Effect of varying $LIC_{min}$ weight (C.1) on TF Distance (C.6)

relation between the value of $LIC_{min}$ and area difference when we look at Figures 5.2 and 5.3. This is expected, since increasing the radius of $LIC_{min}$ implies a greater area sum of LICs. For the other measures of (C.3) to (C.6), we generally see an increase in value starting from around a weight of 8000. Interestingly, from this weight on we have reached relatively stable values for $LIC_{min}$ and the further increase in weight only negatively impacts measures (C.3) to (C.6). In particular, we see an inverse correlation between $LIC_{min}$ in Figure 5.2 and the turning function distance (C.6) in Figure 5.4. As we place a greater emphasis on purely increasing the $LIC_{min}$, we see that consequently the shape of the polygons will show more distortion. Thus for the $LIC_{min}$ weight, we should set its weight somewhere in the range of below 8000 as well as balance it against the turning function distance measure.

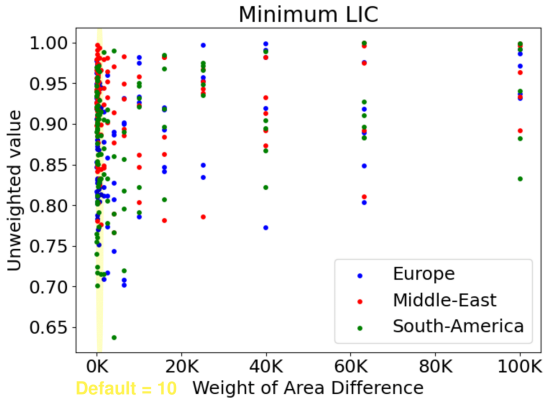


Figure 5.5: Effect of varying area difference weight (C.2) on $LIC_{min}$ value (C.1)

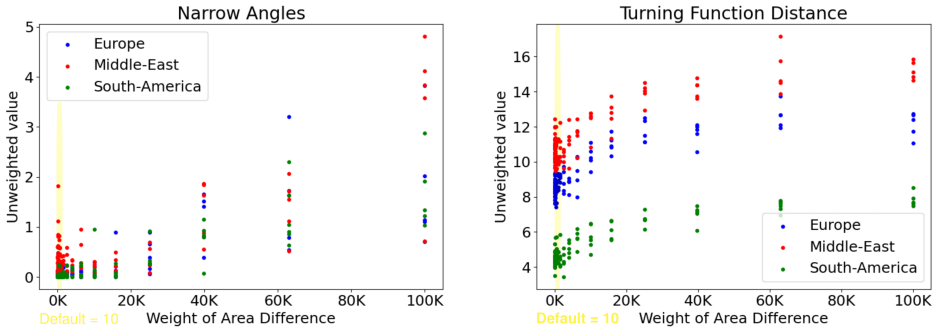


Figure 5.6: Effect of varying Area Diff (C.2) on narrow angles (C.5 left) and TF Distance (C.6 right)

When we vary the weight of the area difference (C.2), one might expect its correlation with $LIC_{min}$ (C.1) to be present as shown above. However, this is not necessarily the case since an improvement in the area difference does not directly impact the $LIC_{min}$ radius, because an epoch can expand any other polygon area besides the one containing $LIC_{min}$. We see this in Figure 5.5, where significantly higher weights will slowly show a more stable deterioration in the value of (C.1). The effect on regularity (C.3) and concaveness (C.4) shows no clear pattern, while narrow angles (C.5) are more prevalent at weights above 40,000 and the turning function distance value (C.6) slightly increases starting around a weight of 10,000 in Figure 5.6. Overall, varying the weight of the area difference does not appear to have much of an effect on other measures by itself.

For regularity (C.3) and concaveness (C.4), there are no clear dependencies other than the fact that the turning function distance trivially increases in value as the polygon shapes become more regular, as they may not have been regularly shaped after map simplification. Nudging epochs towards moving vertices to make its incident faces in the shape of a regular polygon does not necessarily increase the polygon area, thus its LIC radius does not increase much and there are no significant dependencies. As our default weight for these two measures are already set quite low, we have increased the range to which we vary its weight by a factor hundred, yet we see no dependency of other measures.

Recall that in Section 4.2 we have stated that narrow angles (C.5) would prevent reaching good values for the $LIC_{min}$ measure. In Figure 5.7 we observe that from weight 800 on we barely end up with any narrow angles in our output. As we get rid of the occurrence of narrow angles, we see regularity and concaveness improve slightly, whereas the turning function distance worsens particularly for the Middle-East dataset, whose polygon shapes contain many narrow angles after map simplification. As we prefer

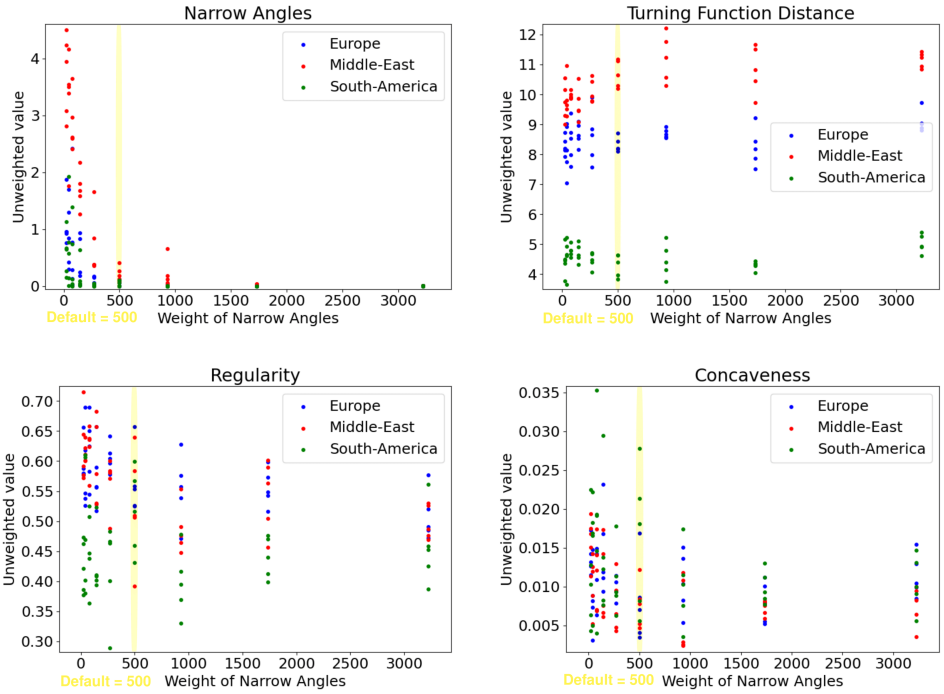


Figure 5.7: Effect of varying narrow angles (C.5) on itself (top left), turning function distance (C.6) (top right), regularity (C.3) (bottom left), and concaveness (C.5) (bottom right).

convex polygons over concave polygons during simulated annealing, we shall set the weight of narrow angles around 1500 such that we prevent narrow angles from occurring.

Lastly, we look at the dependencies of the turning function distance (C.6). In Figure 5.8 we this measure clearly making trade-off with the $LIC_{min}$ as well as the area difference measure.. As we use a higher weight, we are less likely to allow large distance movement of a vertex in an epoch, as it will distort the shape of the incident polygons. At some point, the weight is set so high that we are never able to accept any epoch moves any more and none of the quality measure values change compared to the start of simulated annealing. As we have seen with varying the weight of $LIC_{min}$, we need to balance the weight between these two measures. We do so with visual inspection on how the output drawing turns out in the qualitative evaluation, where we look at the overall shape recognizability of the output compared against the shapes we obtain after map simplification as well as how much larger our $LIC_{min}$ radius has gotten.

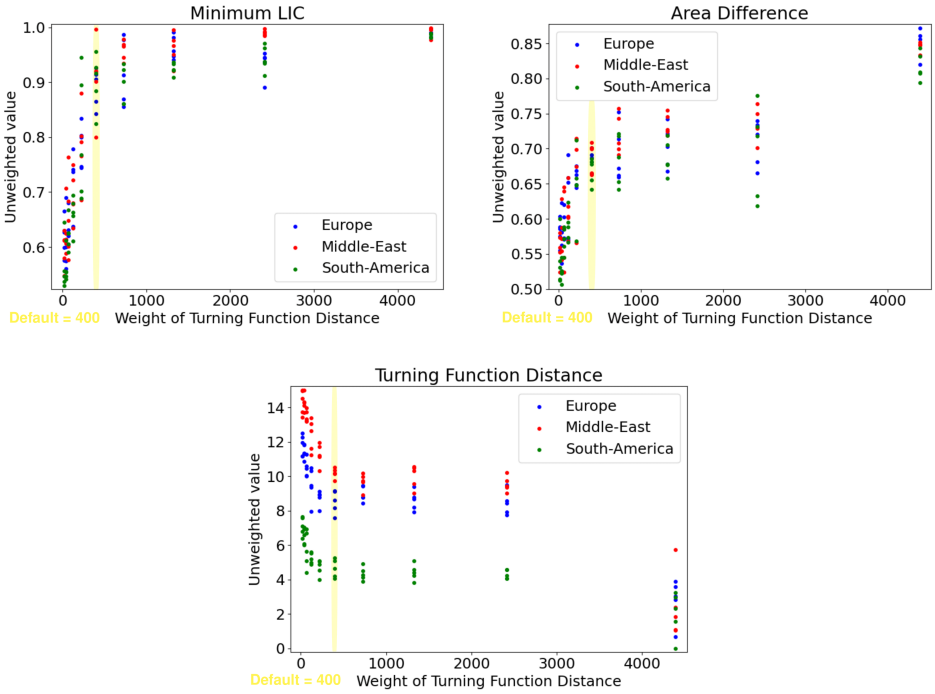


Figure 5.8: Effect of varying turning function distance weight (C.6) on $LIC_{min}$ (C.1) (top left), area difference (C.2) (top right), and itself (bottom).

**Qualitative evaluation** For this evaluation, we will primarily look at the effects of changing the weights on the Western Europe dataset. Each country is given a color for ease of recognition in the figures. Here, the face with the $LIC_{min}$ is Luxembourg, which is surrounded by other faces and we cannot simply extend its surface area towards the outer face. With the default weights, we can see in Figure 5.9 that the output manages to significantly increase the $LIC_{min}$ radius by a factor of 6.9 times compared to the $LIC_{min}$ radius in the simplified drawing. However, this happens at the cost of the shape similarity and we wish to retain more of the shape for the upcoming step of shape reconstruction.

We observe that the shapes of the countries individually does not end up looking similar to its shape after map simplification, which is what we always compare against. This is particularly the case for polygons with a higher vertex degree such as Germany (DE) and Austria (AT), so experiment with raising the turning function distance weight to 800 (case $A$) which should decrease the $LIC_{min}$ radius but bring back some of the shape similarity. In Figure 5.10 we can see that it causes the global shape of Europe to be brought back somewhat compared to filling up the entire bounding box with polygon

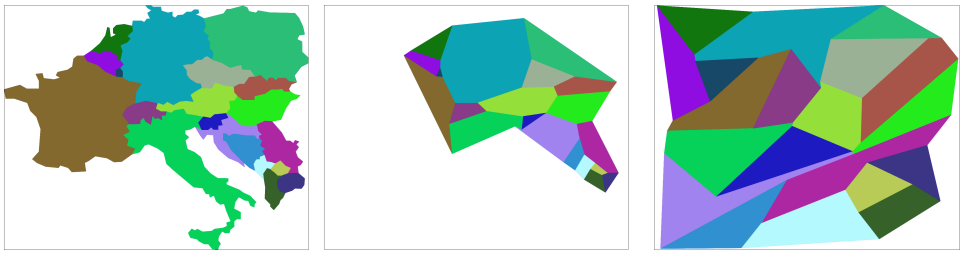

Figure 5.9: Drawing of Western-Europe dataset as input (left), after map simplification (middle), and after simulated annealing with default weights (right).

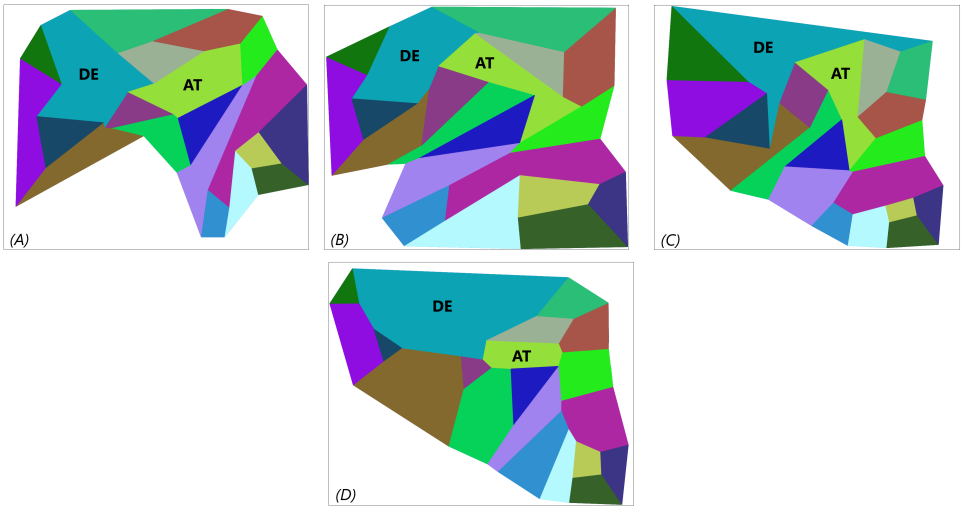


Figure 5.10: Outputs of Western-Europe after simulated annealing for cases $A, B, C, D$, where Germany and Austria are marked by DE and AT respectively.

surface area with the default weights. Here, our radius of the $LIC_{min}$ has increased by a factor of 5.3 times compared to 6.9 times previously.

Looking closer at the individual countries shapes, we see that some are rather concave which is not present in our simplified shape nor makes good use of the polygon's area for the LIC. So we raise the concaveness weight to 40,000 (case $B$) which should nearly prevent the presence of concave shaped polygons to enable a better shape for countries like Germany. We see that the $LIC_{min}$ radius consequently increases to a factor of 6.5 times as expected; however, simply making the polygons convex is not sufficient to obtain a larger LIC as well as a better shape similarity.

Observe that there are multiple polygons whose shape is relatively long, but narrow, in other words it is far from being a regular polygon. Thus, we also increase the weight of the regularity to 40,000 (case $C$) and see that we

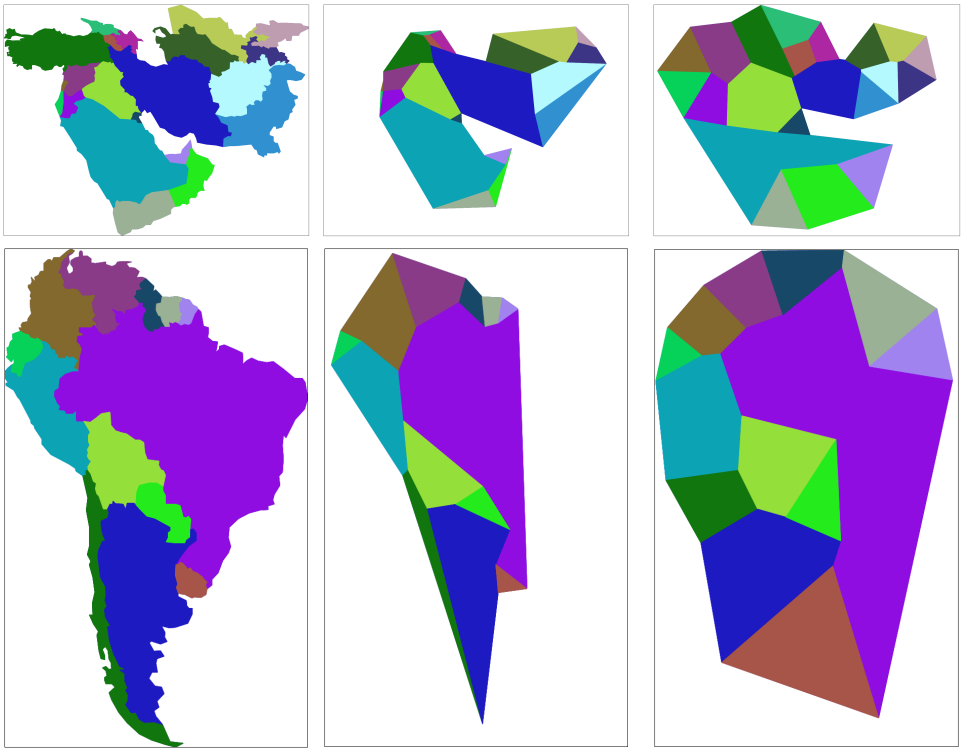

Figure 5.11: Drawing of the Middle-East (top) and South-America (bottom) dataset where we have the input (left), shape after map simplification (middle), and the output with the new weights (right).

are getting better shapes for the countries in the east. Unfortunately, our $LIC_{min}$ radius factor decreases from 6.5 to 4.0 times and we also see that Germany (DE) and Austria (AT) seem to become rather concave again due to the narrow angles formed by itself or its neighboring countries.

Therefore, we also increase the weight of narrow angles to 1500 which should effectively nearly prevent the occurrence of narrow angles (case $D$). Our resulting output looks quite similar to the simplified shape and our $LIC_{min}$ radius ends up to be increased with a factor of 3.7 times. The weights we end up with are quite different than the default weights, which had a larger emphasis on increasing the $LIC_{min}$ radius and did not succeed in keeping the resulting polygon shapes rather similar to the simplified polygons. With our new weights, we have shown a balance between increasing the $LIC_{min}$ radius and obtaining better polygon shapes.

Using these new weights, we also see similar results for the Middle-East and South-America datasets in Figure 5.11.

## 5.2 Shape reconstruction

In this section we evaluate our method of shape reconstruction through image resizing. Recall from Section 4.3 that we use 99% of the $LIC_{min}$ radius for all polygon symbols and that these symbols remove an area of $P_{free}$. We experiment with how changing this $LIC_{min}$ radius affects the shape similarity value between the original shape and the reconstructed shape. We also discuss the shape similarity values with respect to the observed reconstructed shapes.

**Data** We run the experiments on the output of simulated annealing, where we had three datasets: Western Europe, South-America and the Middle-East. Due to randomization in the simulated annealing output, we run our shape reconstruction on 10 different simulated annealing outputs for each dataset, and measure the shape similarity for every country.

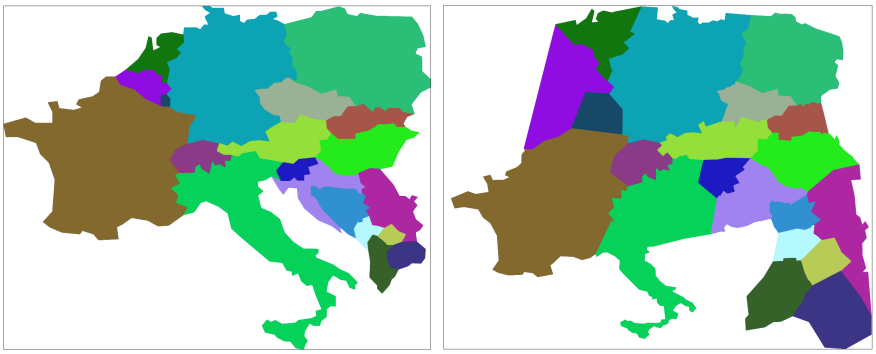

Figure 5.12: Comparing the original shapes of Western Europe (left) against our shape reconstruction output (right).

**Results** Let us first consider the Western Europe dataset, we look at the boxplot of the shape similarity values between the original shapes and the reconstructed shapes using 99% of the $LIC_{min}$ radius in Figure 5.13, where the whiskers denote the minimum and maximum values. The value of the shape similarity measure ranges between (0, 1], where 1 means that both shapes are identical. We see that the shapes generally achieve a similarity value around 0.65, which is decent if we observe the shapes we end up with in Figure 5.12.

To see the variance in polygon shapes for different simulated annealing output, we look in Figure 5.14 at minimum, median, and maximum value shapes of Switzerland (CHE) as it has a large similarity value ranges between (0.48, 0.89). Here, we see improvements to both the local and global shapes of the polygon that contribute to a higher similarity value. The reconstructed global shape of Switzerland depends on the output shape after

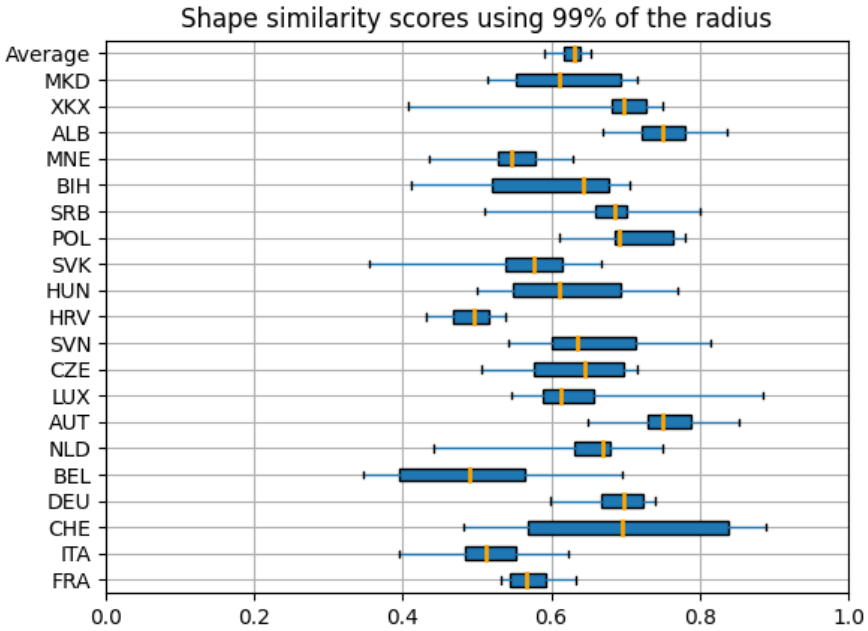


Figure 5.13: Shape similarity measure boxplot of the Western Europe dataset, where each country it denoted by its three-letter code.

simulated annealing, unlike Italy, where the majority of its global shape is defined by its "boot" shape, which is only retrieved during the shape reconstruction. We can see in Figure 5.12 that this "boot" shape of Italy is retained nicely, although it is down-scaled so that it may no longer define the countries global shape.

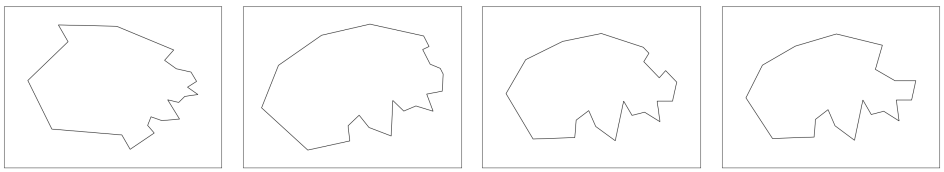

Figure 5.14: Polygon shape of Switzerland (CHE) with similarity values of 0.48, 0.70, 0.89, 1.0 (input) from left to right.

To see the effects of lowering the $LIC_{min}$ radius on the shape similarity values, we set the polygon symbol sizes to 10% of the $LIC_{min}$ radius, which makes the symbol have a negligible effect on removing an area from the free space polygon. The differences in similarity value that we obtain compared to having 99% of the $LIC_{min}$ radius are shown in Figure 5.15. Unfortunately,

we see that decreasing the $LIC_{min}$ radius overall has no significant impact on the shape similarity. We expected similarity improvements, as the smaller the $LIC_{min}$ becomes, the larger the $P_{free}$ becomes and consequently the LIR can also become bigger so we do not have to downsize the reconstructed shape as much.

| | run #1 | run #2 | run #3 | run #4 | run #5 | run #6 | run #7 | run #8 | run #9 | run #10 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FRA | -1.7% | 1.9% | 2.1% | 0.0% | 0.0% | 0.0% | 1.7% | 0.0% | 0.0% | 0.0% | 0.4% |
| ITA | -8.6% | -1.3% | -1.4% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | -1.1% |
| CHE | -1.5% | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% | 1.3% | 0.0% | 0.0% | 0.0% | -0.0% |
| DEU | 0.6% | 11.1% | -3.2% | 1.6% | -0.8% | 10.5% | 2.9% | 2.8% | -2.1% | 7.3% | 3.1% |
| BEL | -1.0% | -0.6% | 4.3% | 5.2% | -1.1% | -0.3% | -8.2% | -27.9% | 2.6% | -10.8% | -3.8% |
| NLD | -7.3% | -1.0% | -5.7% | 1.5% | 1.0% | -0.8% | 1.4% | -0.3% | -3.2% | -5.4% | -2.0% |
| AUT | 0.2% | -1.0% | -0.3% | 0.4% | 0.0% | 5.3% | 0.9% | 4.1% | 2.8% | 1.1% | 1.3% |
| LUX | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| CZE | 9.0% | -2.0% | 7.6% | -3.8% | -3.9% | -4.4% | -6.2% | -3.0% | -7.6% | 0.0% | -1.4% |
| SVN | 1.0% | -1.8% | -0.5% | 4.0% | -2.3% | -3.0% | -5.2% | 1.4% | -1.7% | 2.0% | -0.6% |
| HRV | 1.7% | 1.9% | 2.8% | -1.5% | 4.8% | -11.7% | -1.5% | -2.6% | 3.6% | 0.4% | -0.2% |
| HUN | -6.1% | 0.0% | 0.0% | 0.0% | 2.2% | 0.0% | 0.0% | -1.8% | 0.0% | 4.1% | -0.2% |
| SVK | -4.0% | 0.0% | 0.0% | -3.1% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | -0.7% |
| POL | 0.0% | 0.0% | 0.0% | 0.1% | -5.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | -0.5% |
| SRB | 0.9% | -0.6% | -1.3% | -1.9% | 0.0% | -0.2% | 7.0% | -5.7% | 0.0% | -7.8% | -1.0% |
| BIH | -14.3% | -6.3% | -30.4% | -8.0% | 7.9% | 25.9% | -33.5% | -2.9% | 0.1% | 1.5% | -6.0% |
| MNE | 9.6% | 1.8% | 1.9% | -7.1% | 3.2% | 5.1% | -1.0% | -0.3% | -2.3% | -0.8% | 1.0% |
| ALB | 1.1% | 1.9% | 2.4% | 0.2% | -12.9% | 2.0% | -6.4% | 1.0% | 4.4% | -3.3% | -1.0% |
| XKX | 3.1% | 0.1% | 0.0% | -0.2% | 0.0% | 1.5% | 0.0% | -2.9% | -1.1% | 0.0% | 0.0% |
| MKD | -0.6% | -4.5% | 1.7% | 0.7% | -1.5% | 5.9% | 5.1% | 1.2% | 2.3% | 1.7% | 1.2% |

Figure 5.15: Shape similarity differences between using 10% of the $LIC_{min}$ radius instead of 99% for the Western Europe dataset.

However, upon visual inspection of the changes that are made to the free space polygons and the LIR when we set all polygon symbols to 10% of the $LIC_{min}$ radius in Figure 5.16, we see that most of the LIR do not change. Some free space polygons do not have a LIR as the corresponding simplified edge has no reinsertion vertices. When the LIR does change, its aspect ratio may change to become narrower and taller instead, which covers a larger surface area, but might also end up "compressing" the reconstructed shape for a worse similarity value. So we see that our $LIC_{min}$ radius is generally too small to affect the $P_{free}$ area significantly, and also that the bottleneck to our shape reconstruction is rather the area of $P_{free}$ that is available to reinsert vertices in.

We perform the same experiments for the South-America and Middle-East datasets and observe their reconstructed shape in Figure 5.17. Again, we see that reducing the $LIC_{min}$ radius has little overall impact on the shape similarity for both datasets and the experiment results are generally similar

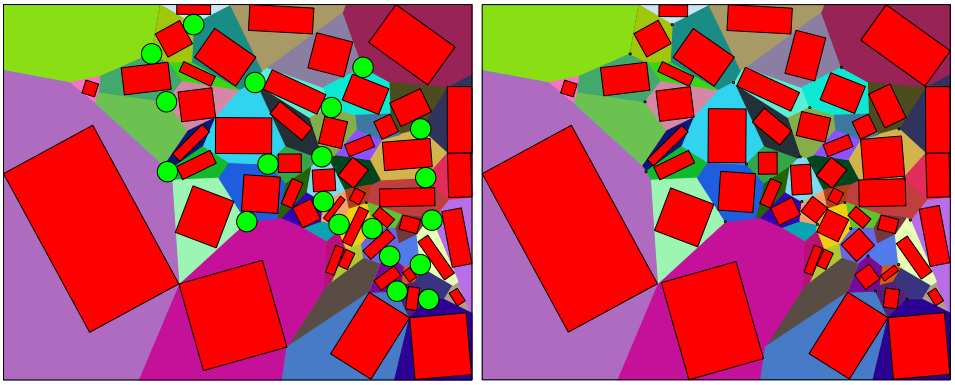

Figure 5.16: Example of the free space polygons for the Western Europe dataset, each in a different color with their LIR (red). On the left we have symbols (green) that are 99% of the $LIC_{min}$ radius, and on the right they are 10%.

to that of Western Europe. The full figures are available in the Appendix B. For the South-America dataset, we particularly notice the shape change to Chile, which originally had a long narrow shape that inefficiently accommodates a LIC, and in our output it lost all of its shape and become rather rectangular. Its border shape with the sea became significantly compressed such that the finer details are no longer visible.

Overall, we see that the shape reconstruction method of image resizing provides decent results when the LIR is relatively large enough for the reinsertion vertices. However, this LIR does not utilize the full free space polygon area and this remains open as a possibility for improvement upon the shape reconstruction.

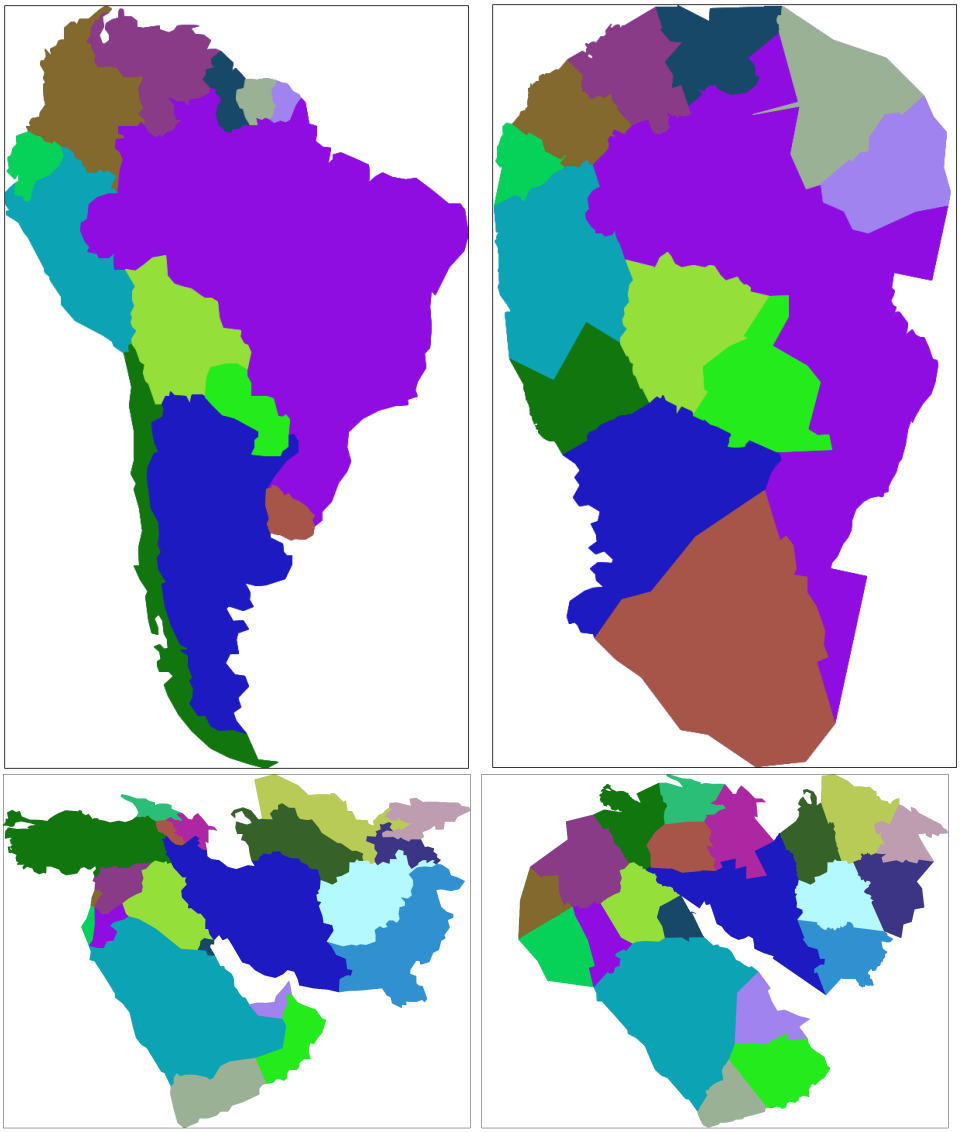

Figure 5.17: Comparing the original shapes (left) of South-America (top) and the Middle-East (bottom) against our shape reconstruction output (right).

# Chapter 6

# Conclusion

In this thesis, we have presented an algorithm that transforms the boundaries of the underlying geographic map to prevent symbol overlaps for proportional symbol maps. We first simplified the map by removing vertices and storing their relative position to the resulting simplified edge. Using the simulated annealing technique, we have transformed the simplified map so that the $LIC_{min}$ has increased by several factors, while still maintaining a sense of recognizability for the polygon shapes by experimenting with the quality measures used in the simulated annealing. Finally, we reconstruct the original shapes of the polygons by reinserting the removed vertices back to their relative positions. To prevent any crossings during this process, we have defined the notion of free spaces in which the entire reconstructed shape will be contained, and we evaluated the shape similarity between the original polygons and our output polygons. The last step of our algorithm sets the maximum symbol size to the radius of $LIC_{min}$. As shown in Figures 6.1, 6.2, 6.3, our resulting geographic maps clearly allow a larger symbol to be used for the smallest countries, while keeping the map relatively recognizable.

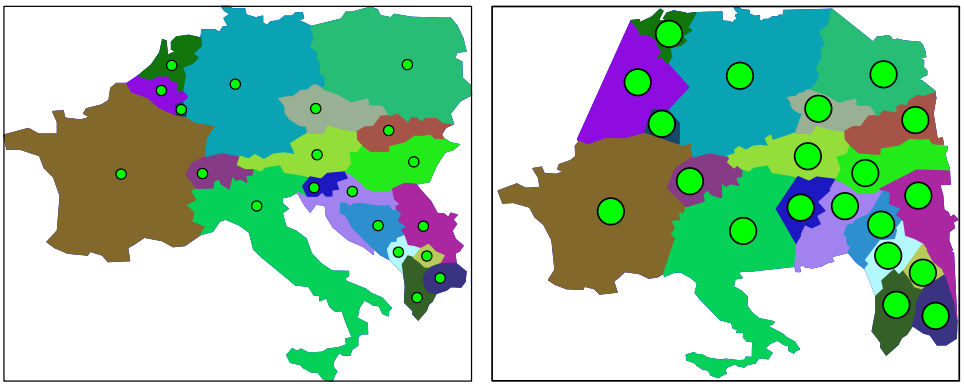

Figure 6.1: Comparing the $LIC_{min}$ of the input polygons (left) to the output polygons (right) for the Western Europe dataset.

There is still room for improvement in our approach. After the map sim-

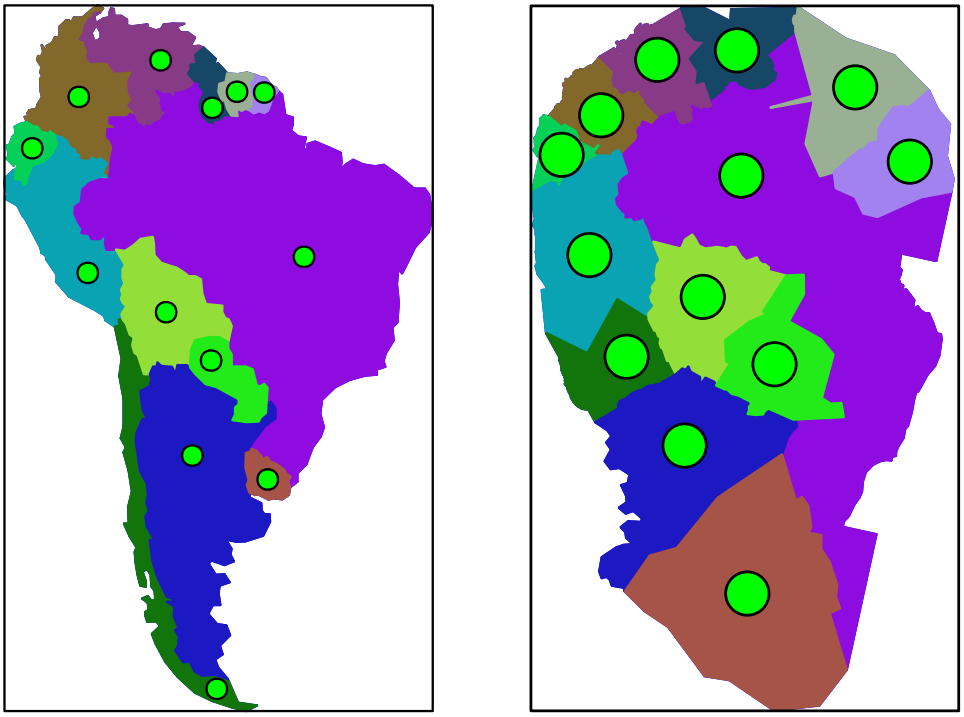

Figure 6.2: Comparing the $LIC_{min}$ of the input polygons (left) to the output polygons (right) for the South-America dataset.

plification step in Section 4.1, our simplified polygons may not accurately represent a simplified shape of the original polygon. As one of our quality measures in simulated annealing relies on maintaining this simplified shape, improving the polygon simplification to be of a similar shape to the original should lead to better global shape similarity in the shape reconstruction step. Our simulated annealing quality measures do not include a notion of directional relationship between polygons and relies only on the topology preservation to prevent significant overall directional distortion. We see with our results that the directional relationships in our output maps are generally quite good, with few exceptions where the relation between some reference and target countries changed from, for example, east to southeast. The introduction of some directional relationship model that does not categorize relationships in a small set of compass directions and is also not too restrictive as a quality measure in simulated annealing, could ensure even less directional distortion appearing in the output map. In Section 4.3, we have mentioned the technique of image warping to be an example where we utilize the entire free space polygon more effectively than the technique of image resizing. Other methods can be proposed here, and the comparison to our image resizing technique is done by comparing the shape similarity value differences for the same datasets.

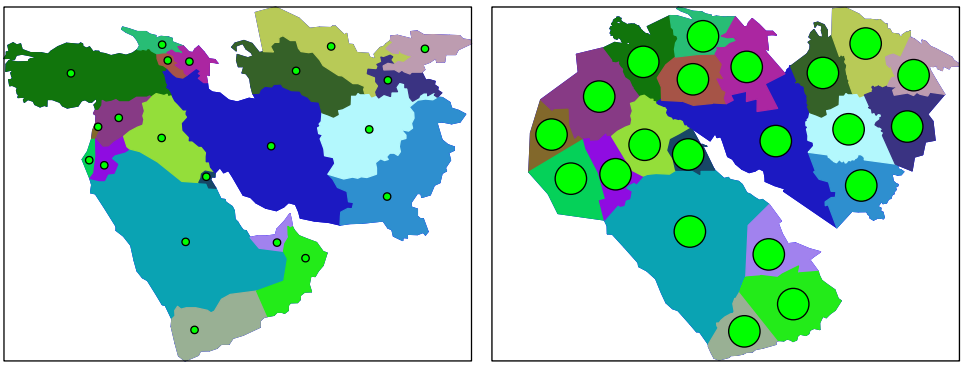

Figure 6.3: Comparing the $LIC_{min}$ of the input polygons (left) to the output polygons (right) for the Middle-East dataset.

# Bibliography


[1] Alia I. Abdelmoty and M. Howard Williams. Approaches to the representation of qualitative spatial relationships for geographic databases: A critical survey and possible extensions. In *Advanced Geographic Data Modelling*, pages 216–221, 1994.

[2] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, and J.S.B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216, 1991.

[3] Tetsuo Asano. An efficient algorithm for finding the visibility polygon for a polygonal region with holes. *IEICE TRANSACTIONS (1976-1990)*, 68(9):557–559, 1985.

[4] Francisc Bungiu, Michael Hemmer, John Hershberger, Kan Huang, and Alexander Kröller. Efficient computation of visibility polygons. 2014.

[5] Sergio Cabello, Herman J. Haverkort, Marc J. van Kreveld, and Bettina Speckmann. Algorithmic aspects of proportional symbol maps. *Algorithmica*, 58:543–565, 2009.

[6] Ramon Chalmeta, Ferran Hurtado, Vera Sacristán, and Maria Saumell. Measuring regularity of convex polygons. *Computer-Aided Design*, 45(2):93–104, 2013.

[7] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990.

[8] Kenneth De Jong, Jong William, and William Spears. Using genetic algorithms to solve NP-complete problems. 1998.

[9] Erik Demaine, Sandor Fekete, and Robert Lang. Circle packing for origami design is hard. 2010.

[10] Tim Dwyer, Kim Marriott, and Peter Stuckey. Fast node overlap removal. *Lecture Notes in Computer Science*, 2006:153–164, 2005.

[11] Hongchao Fan, Zhiyao Zhao, and Wenwen Li. Towards measuring shape similarity of polygons based on multiscale features and grid context descriptors. *ISPRS International Journal of Geo-Information*, 10:279, 2021.

[12] Petr Felkel and Stepán Obdrzálek. Straight skeleton implementation. In *Proceedings of Spring Conference on Computer Graphics*, pages 210–218, 1998.

[13] James John Flannery. The relative effectiveness of some common graduated point symbols in the presentation of quantitative data. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 8:96–109, 1971.

[14] Emden R. Gansner and Yifan Hu. Efficient, proximity-preserving node overlap removal. *J. Graph Algorithms Appl.*, 14:53–74, 2010.

[15] Daniel Garcia-Castellanos and Umberto Lombardo. Poles of inaccessibility: A calculation algorithm for the remotest places on earth. *Scottish Geographical Journal*, 123(3):227–233, 2007.

[16] Michael T. Gastner and M. E. J. Newman. Diffusion-based method for producing density-equalizing maps. In *Proceedings of the National Academy of Sciences*, volume 101, pages 7499–7504, 2004.

[17] Roop Kishor Goyal. *Similarity Assessment for Cardinal Directions between Extended Spatial Objects*. PhD thesis, 2000.

[18] Trevor Griffin. The importance of visual contrast for graduated circles. *Cartography*, 19(1):21–30, 1990.

[19] R Haar and University of Maryland at College Park. Computer Science Center. *Computational models of spatial relations*. 1976.

[20] Donald House and Christopher Kocmoud. Continuous cartogram construction. In *Proceedings Visualization '98 (Cat. No.98CB36276)*, pages 197–204, 1998.

[21] Xiaodi Huang, Wei Lai, A.S.M. Sajeev, and Junbin Gao. A new algorithm for removing node overlapping in graph visualization. *Information Sciences*, 177(14):2821–2844, 2007.

[22] B. Joe and R. B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT*, 27(4):458–473, 1987.

[23] Rozieana Khairuddin and Zaitul Zainuddin. A comparison of simulated annealing cooling strategies for redesigning a warehouse network problem. *Journal of Physics: Conference Series*, 1366:012078, 2019.

[24] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[25] Guilherme Kunigami, Pedro J de Rezende, Cid C de Souza, and Tallys Yunes. Optimizing the layout of proportional symbol maps: Polyhedra and computation. *Journal on Computing*, 26(2):199–207, 2014.

[26] David Lamb. An automated displaced proportional circle map using Delaunay triangulation and an algorithm for node overlap removal. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 52(4):364–370, 2017.

[27] Guojun Lu and Atul Sajjanhar. Region-based shape representation and similarity measure suitable for content-based image retrieval. *Multimedia Systems*, 7(2):165–174, 1999.

[28] Kelly A. Lyons, Henk Meijer, and David Rappaport. Algorithms for cluster busting in anchored graph drawing. *Journal of Graph Algorithms and Applications*, 2:1–24, 1998.

[29] Anna Markowska. Cartograms – classification and terminology. *Polish Cartographical Review*, 51(2):51–65, 2019.

[30] Kim George Marriott, Peter J Stuckey, Vincent Tam, and Weiqing He. Removing node overlapping in graph layout using constrained optimization. *Constraints*, 8(2):143 – 171, 2003.

[31] Oscar Martinez. An efficient algorithm to calculate the center of the biggest inscribed circle in an irregular polygon. 2012.

[32] Wouter Meulemans. Efficient optimal overlap removal: Algorithms and experiments. *Computer Graphics Forum*, 38:713–723, 2019.

[33] D.E. Muller and F.P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7(2):217–236, 1978.

[34] Lev Nachmanson, Arlind Nocaj, Sergey Bereg, Leishi Zhang, and Alexander Holroyd. Node overlap removal by growing a tree. In *Graph Drawing and Network Visualization*, pages 33–43, 2016.

[35] Nadine Nandanari. *A technique to reduce overlapping symbols on proportional symbol map of multi-dimensional data.* PhD thesis, Doctoral Dissertation. University of Tsukuba, 2016.

[36] Dimitris Papadias and Yannis Theodoridis. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138, 1997.

[37] Donna J. Peuquet and Zhan Ci-Xiang. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition*, 20(1):65–74, 1987.

[38] David Swart. Warping pictures nicely. In *Proceedings of Bridges 2011: Mathematics, Music, Art, Architecture, Culture*, pages 303–310, 2011.

[39] Vassilios E. Theodoracatos and James L. Grimsley. The optimal packing of arbitrarily-shaped polygons using simulated annealing and polynomial-time cooling schedules. *Computer Methods in Applied Mechanics and Engineering*, 125(1):53–70, 1995.

[40] Thomas C. van Dijk and Jan-Henrik Haunert. Interactive focus maps using least-squares optimization. *International Journal of Geographical Information Science*, 28(10):2052–2075, 2014.

[41] Johan Wagemans, James H. Elder, Michael Kubovy, Stephen E. Palmer, Mary A. Peterson, Manish Singh, and Rüdiger von der Heydt. A century of gestalt psychology in visual perception: I. perceptual grouping and figure-ground organization. *Psychological bulletin*, 138 6:172–217, 2012.

# Appendix A

# Simulated annealing experimental results

This appendix shows extended figures of the trial results for all quality measures presented in Section 4.2.

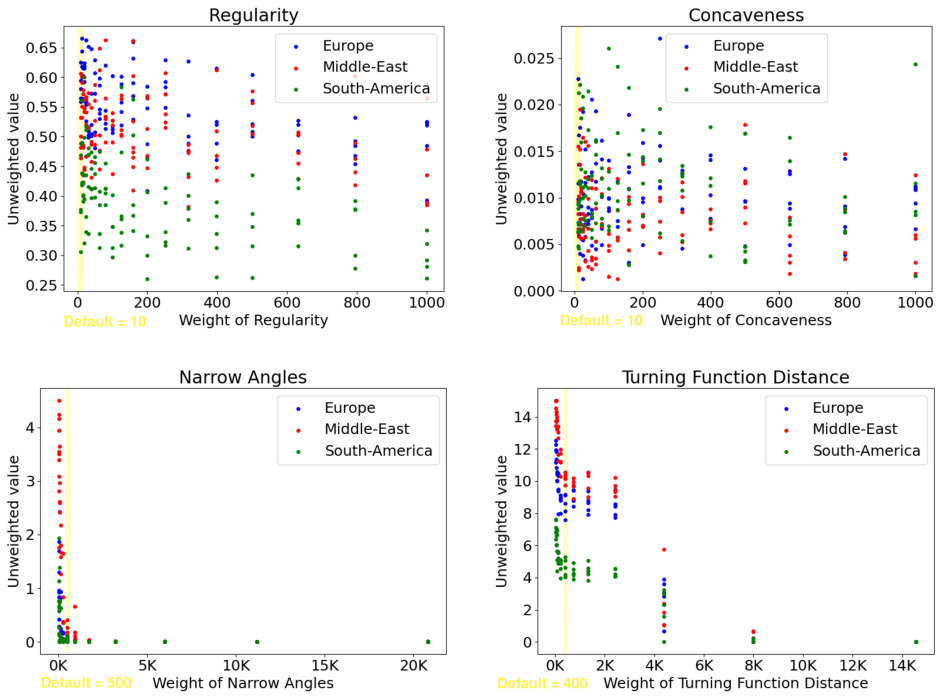


Figure A.1: Varying the weight of a quality measure in the objective function, showing the resulting unweighted value for the quality measure whose weight is varied.

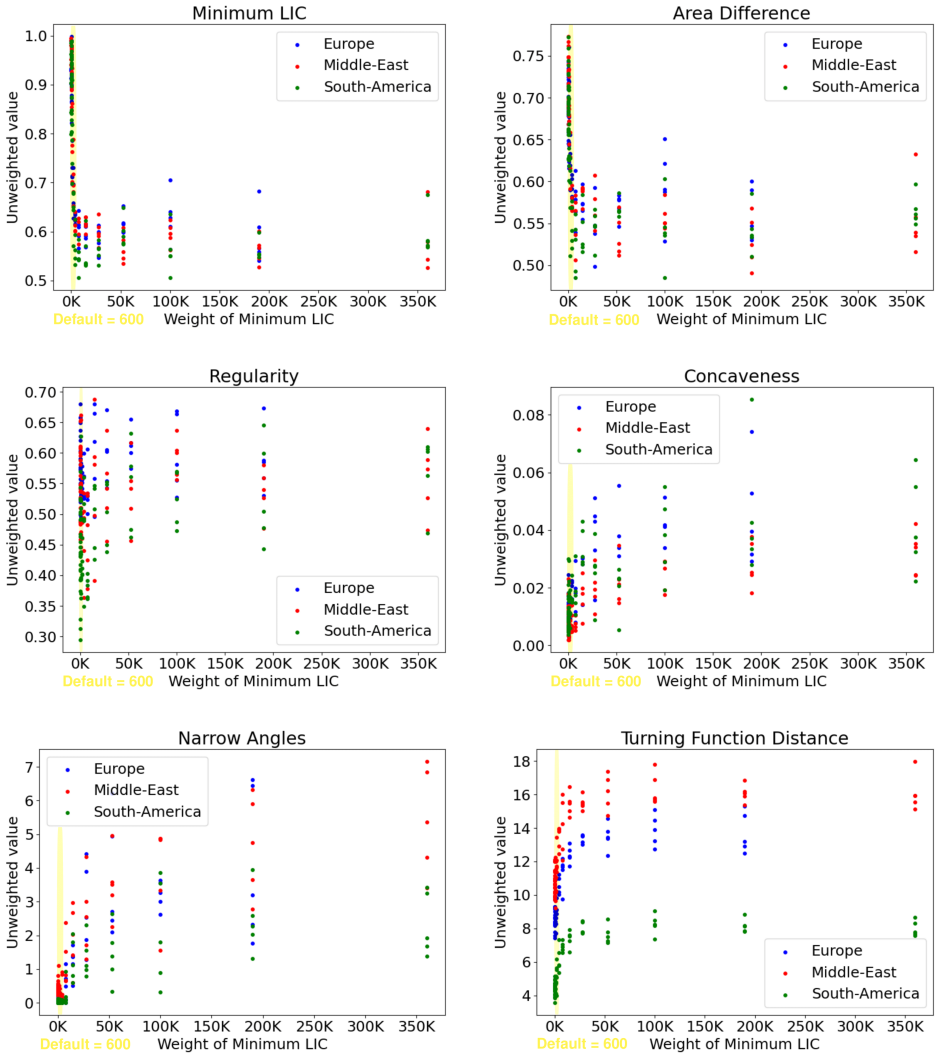


Figure A.2: Varying weight of $LIC_{min}$ (C.1), showing the resulting unweighted value for (C.1) to (C.6) from left to right, top to bottom.

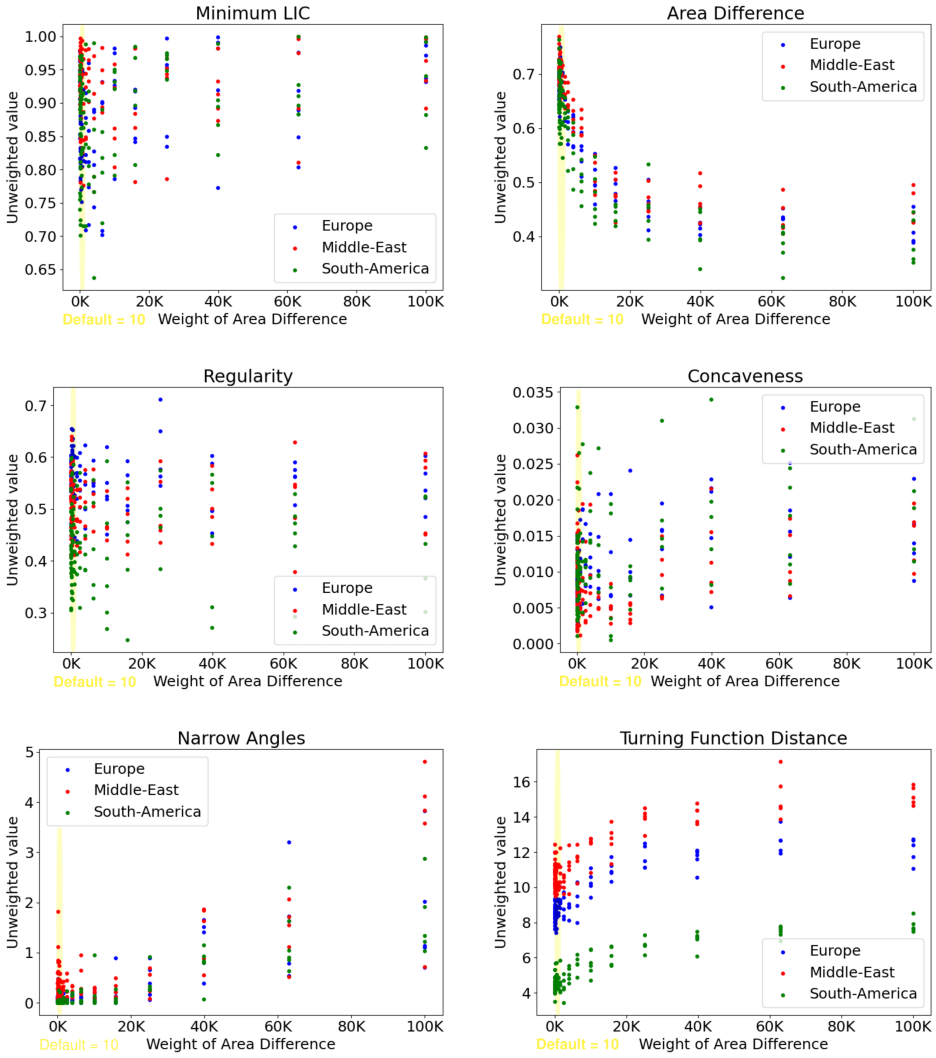


Figure A.3: Varying weight of area difference (C.2), showing the resulting unweighted value for (C.1) to (C.6) from left to right, top to bottom.

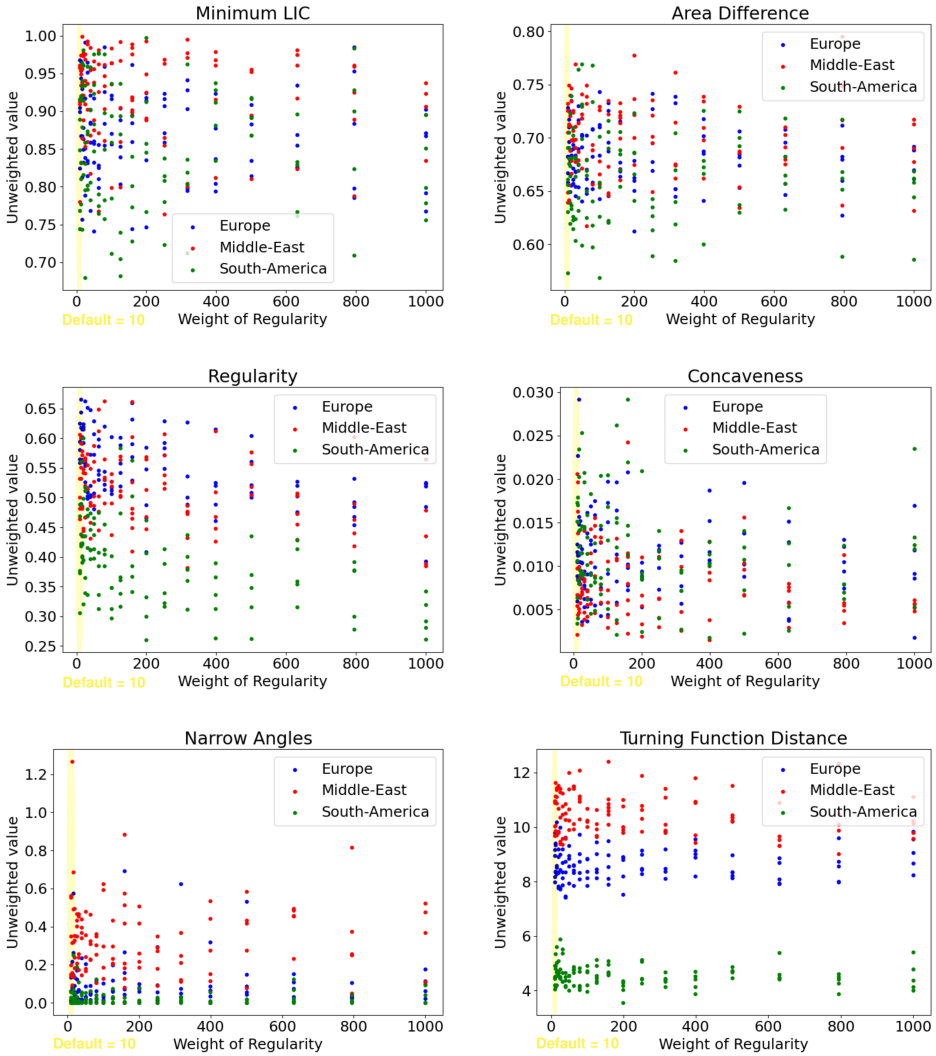


Figure A.4: Varying weight of regularity (C.3), showing the resulting unweighted value for (C.1) to (C.6) from left to right, top to bottom.

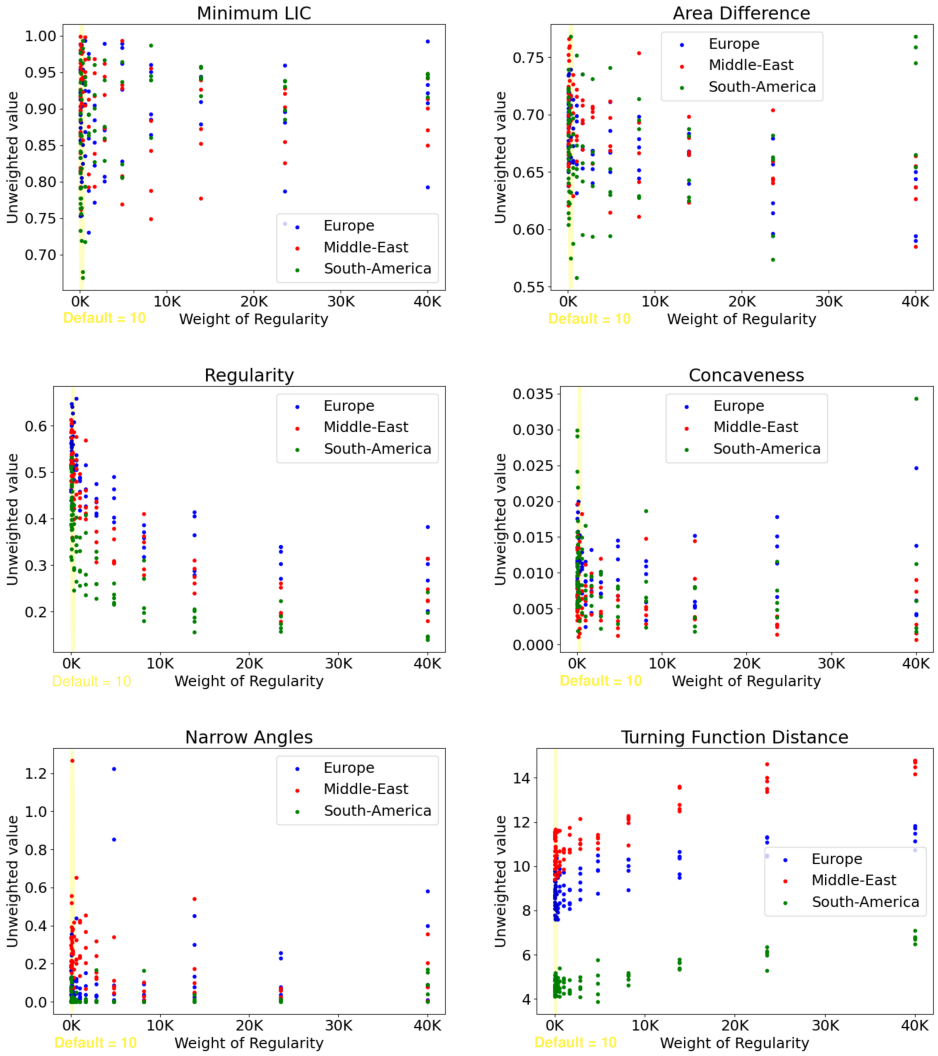


Figure A.5: A significantly larger varying weight of regularity (C.3), showing the resulting unweighted value for (C.1) to (C.6) from left to right, top to bottom.

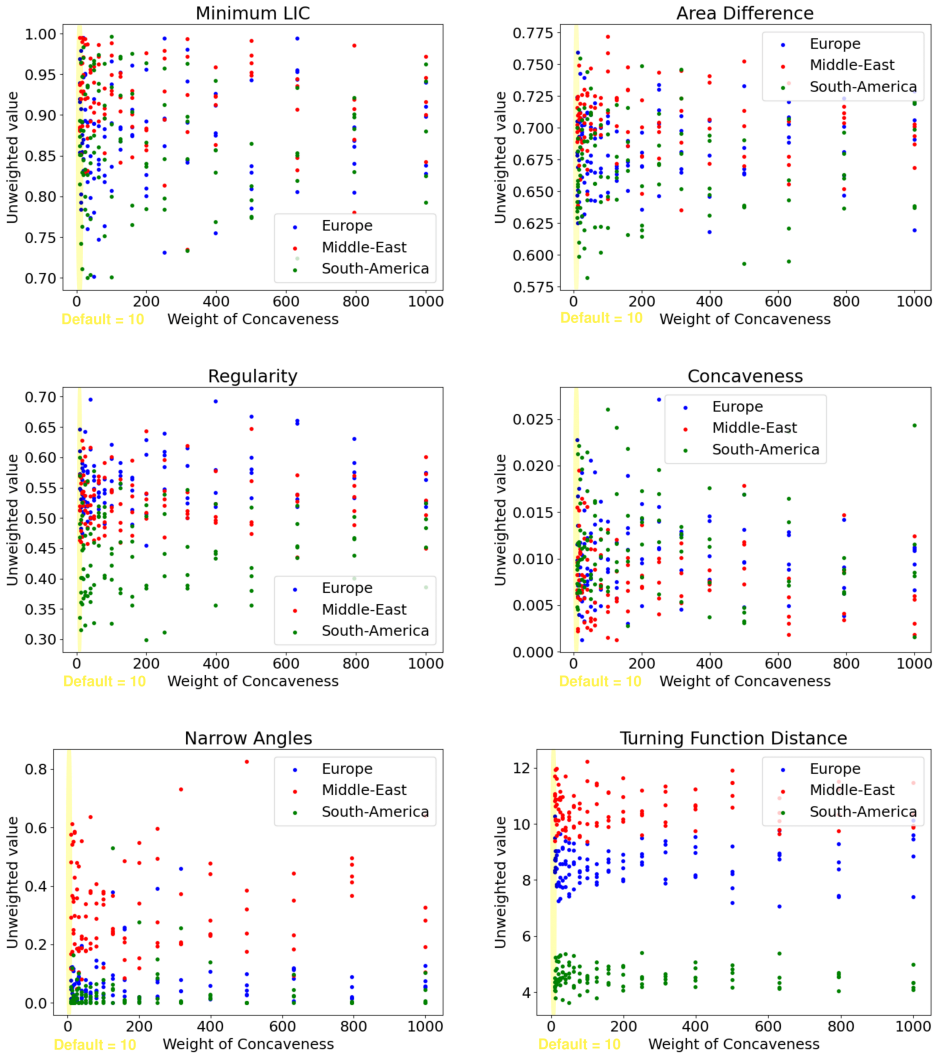


Figure A.6: Varying weight of concaveness (C.4), showing the resulting unweighted value for (C.1) to (C.6) from left to right, top to bottom.

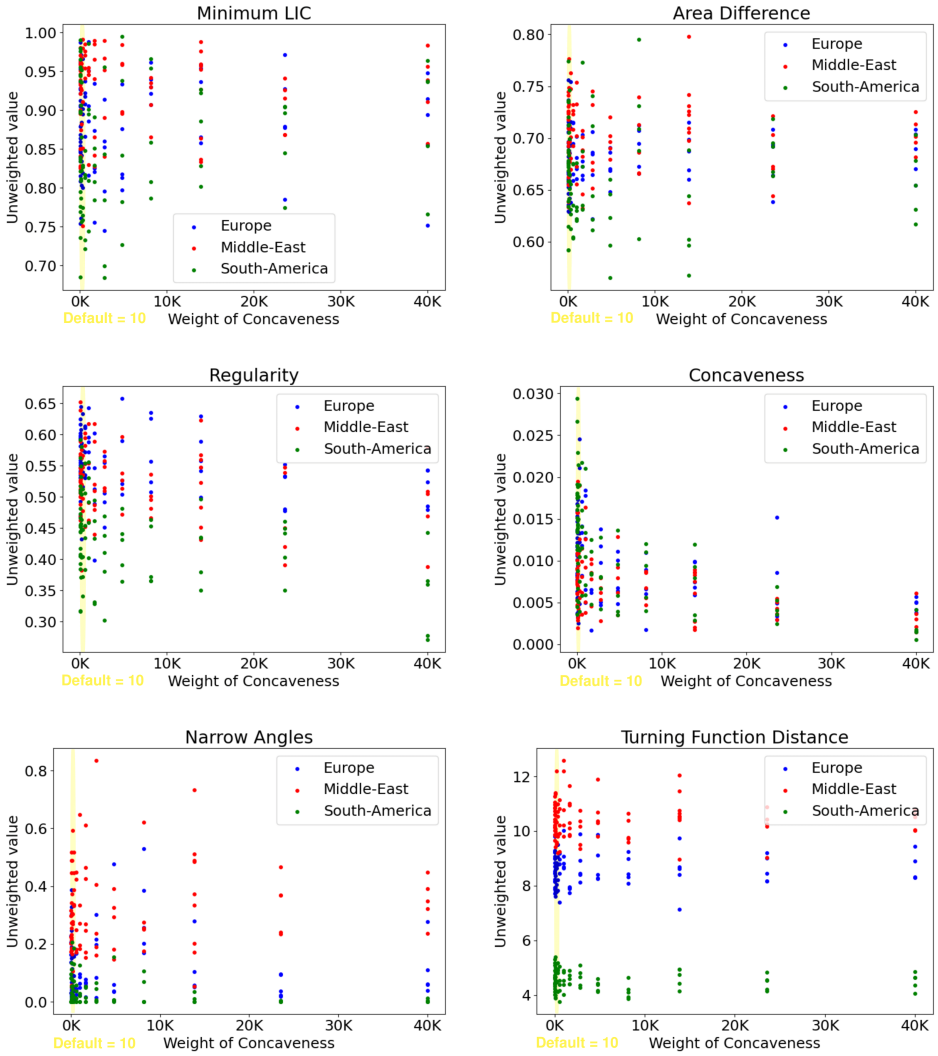


Figure A.7: A significantly larger varying weight of concaveness (C.4), showing the resulting unweighted value for (C.1) to (C.6) from left to right, top to bottom.

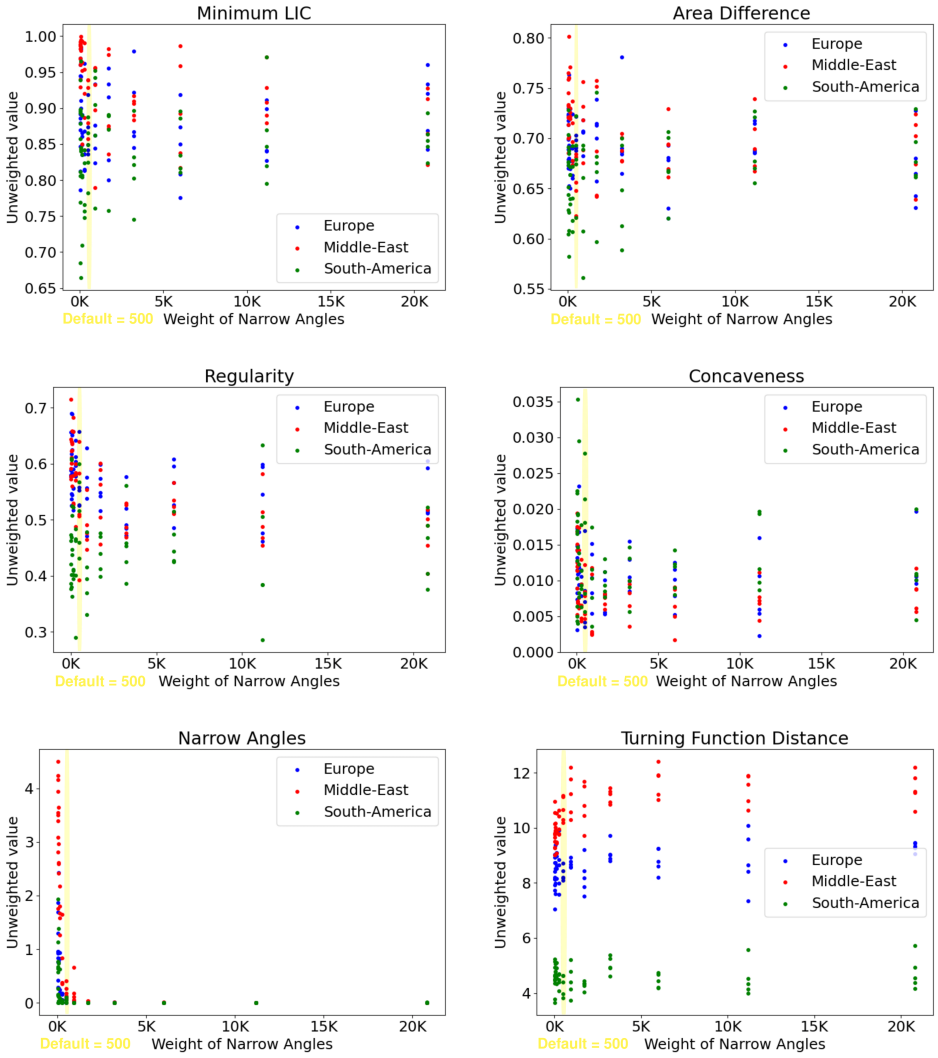


Figure A.8: Varying weight of narrow angles (C.5), showing the resulting unweighted value for (C.1) to (C.6) from left to right, top to bottom.

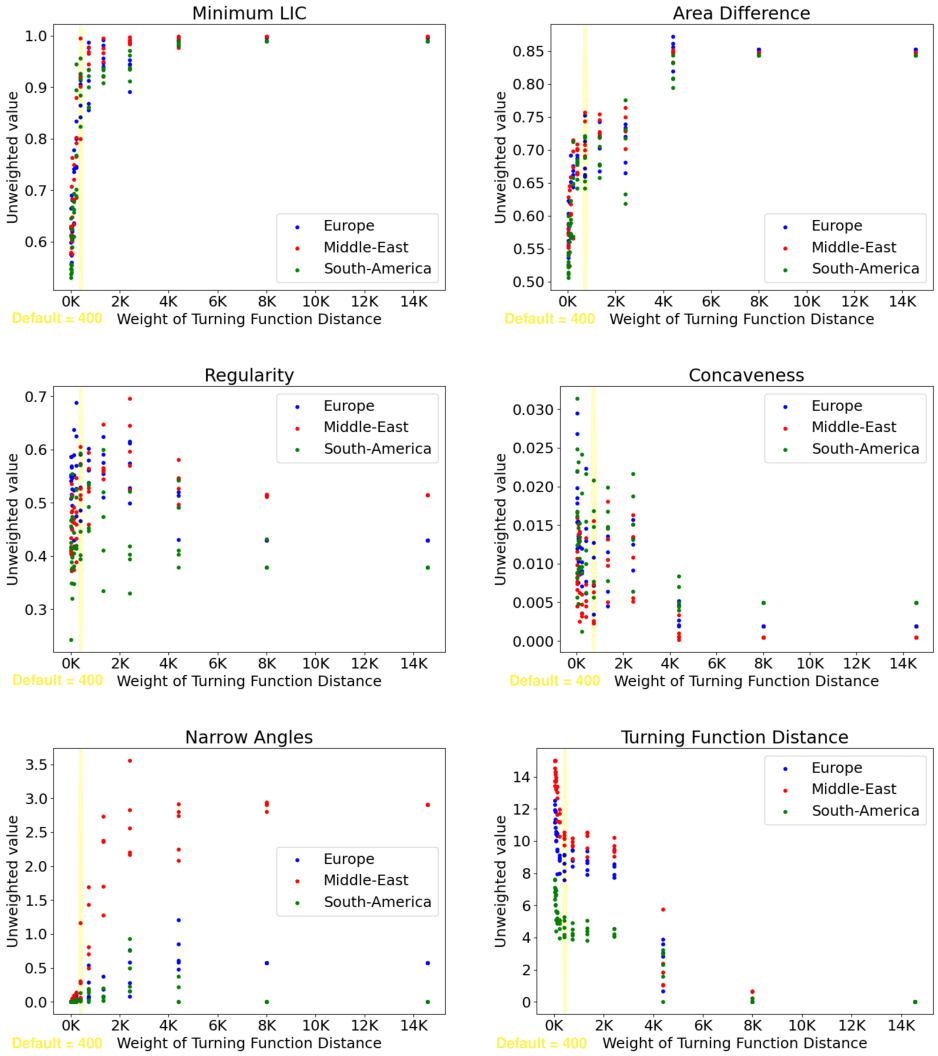


Figure A.9: Varying weight of turning function distance (C.6), showing the resulting unweighted value for (C.1) to (C.6) from left to right, top to bottom.

# Appendix B

# Shape reconstruction experimental results

This appendix shows the experiment results for the three datasets presented in Section 5.2.

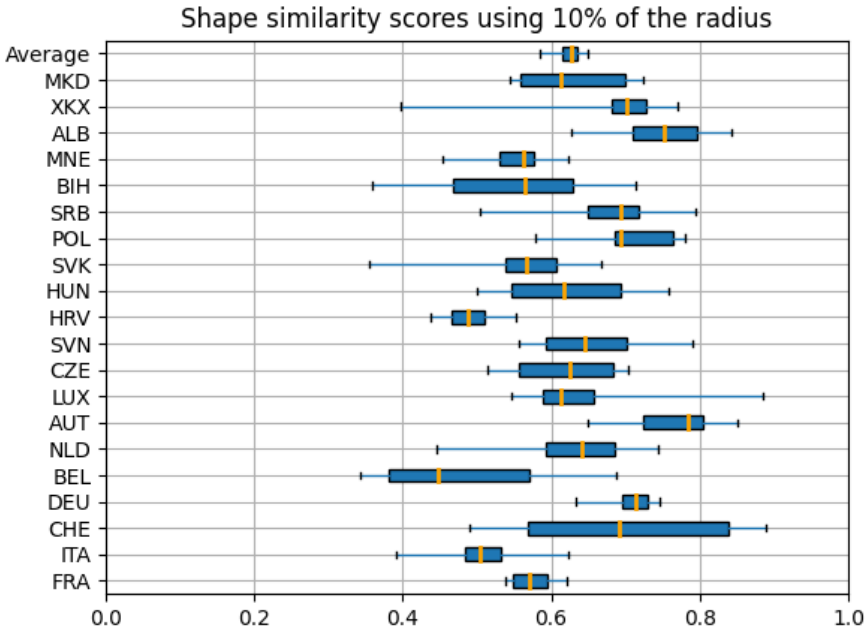


Figure B.1: Shape similarity measure boxplot for the Western Europe dataset, where each country is denoted by its three-letter code.

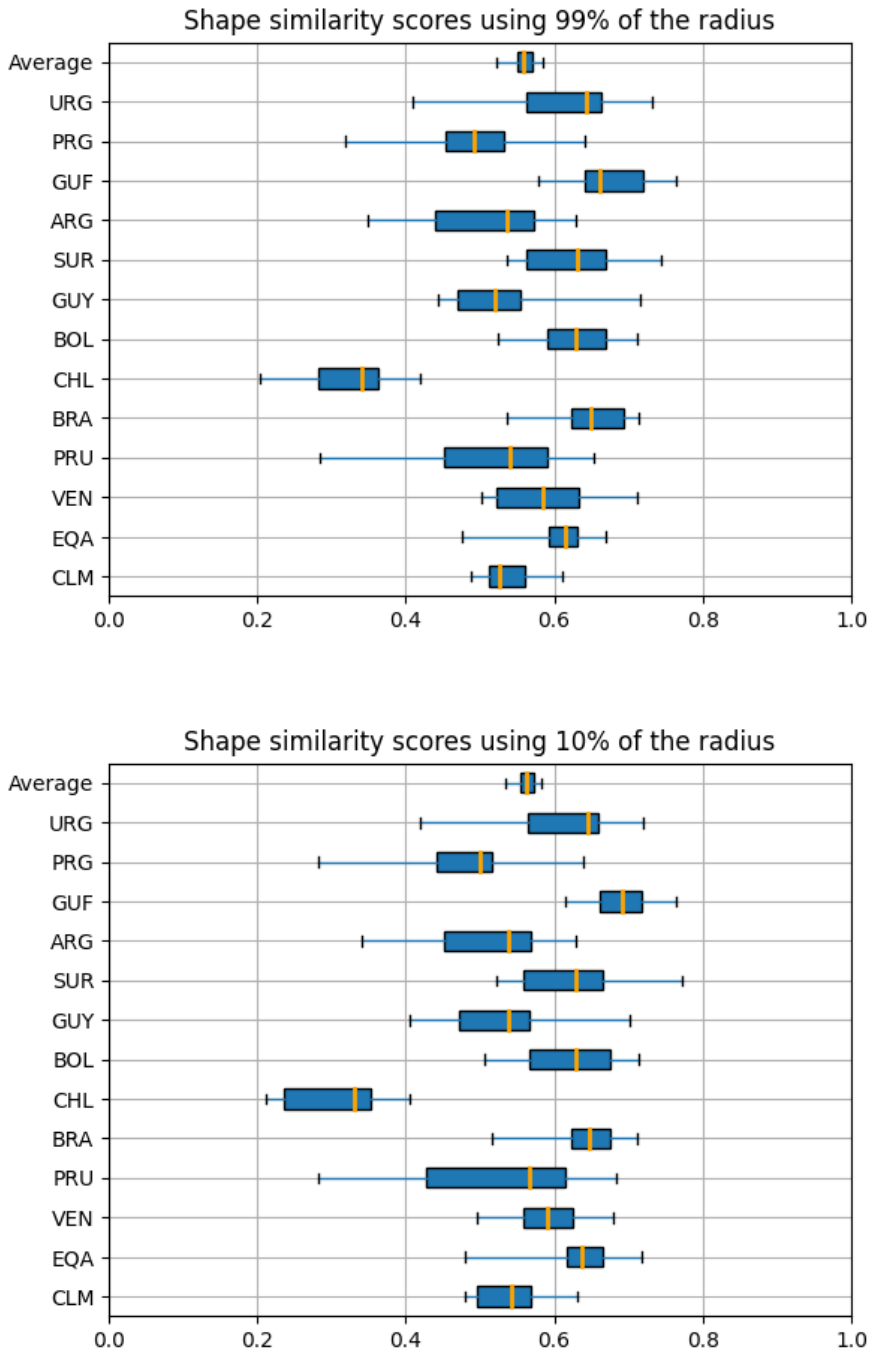


Figure B.2: Shape similarity measure boxplots for the South-America dataset, where each country is denoted by its three-letter code.

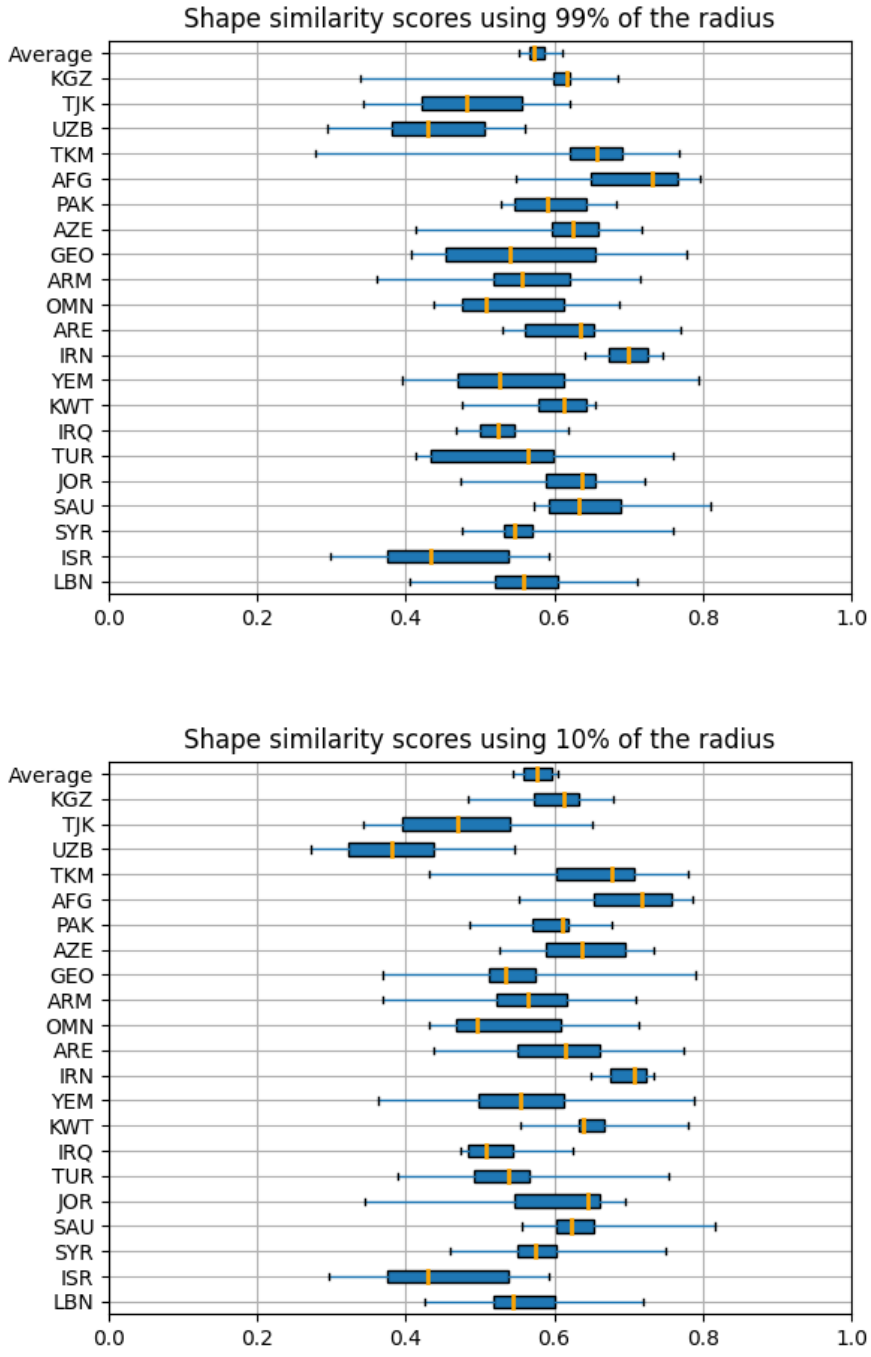


Figure B.3: Shape similarity measure boxplots for the Middle-East dataset, where each country is denoted by its three-letter code.

| | run #1 | run #2 | run #3 | run #4 | run #5 | run #6 | run #7 | run #8 | run #9 | run #10 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CLM | 12.1% | -5.8% | 3.4% | 1.4% | -5.8% | 1.1% | 3.8% | -2.0% | 3.3% | -5.8% | 0.6% |
| EQA | 12.7% | 2.7% | 2.6% | 11.8% | 5.8% | 40.9% | -2.7% | 12.3% | 2.9% | -24.3% | 6.5% |
| VEN | -2.8% | 1.3% | 14.8% | 3.6% | -4.6% | 11.3% | 2.7% | -4.7% | -2.3% | -3.6% | 1.6% |
| PRU | -2.1% | 5.0% | -10.8% | -0.7% | 26.0% | -0.5% | 0.2% | -15.7% | -4.2% | 5.6% | 0.3% |
| BRA | -3.8% | -0.7% | -0.2% | -6.0% | -0.9% | 1.5% | -0.1% | -1.4% | -0.4% | 0.9% | -1.1% |
| CHL | -23.7% | 1.9% | -11.6% | -3.5% | 4.1% | -11.8% | 2.9% | -3.6% | -9.7% | -4.7% | -6.0% |
| BOL | -5.1% | 1.3% | -2.7% | 0.3% | 1.1% | -1.2% | 1.5% | 4.5% | 0.9% | -3.5% | -0.3% |
| GUY | -8.7% | -1.9% | 6.9% | 13.3% | 20.4% | 1.5% | -0.5% | 0.1% | 5.6% | -23.9% | 1.3% |
| SUR | -2.2% | -1.8% | 1.8% | 6.8% | -9.2% | -1.0% | -10.2% | 3.8% | -2.3% | 14.3% | 0.0% |
| ARG | -0.1% | 0.3% | 0.7% | 0.5% | 5.2% | -0.4% | 2.5% | -2.5% | 2.9% | -1.2% | 0.8% |
| GUF | 0.2% | -1.2% | 6.6% | -1.0% | -0.9% | 9.2% | 0.5% | 13.1% | -0.3% | 3.1% | 2.9% |
| PRG | 3.9% | -42.7% | -6.9% | 1.2% | 5.0% | -4.1% | -0.4% | -2.1% | 12.1% | 3.5% | -3.1% |
| URG | -1.6% | 7.4% | -0.3% | -0.5% | 2.6% | -0.5% | -1.1% | 2.2% | 0.0% | 1.2% | 0.9% |

Figure B.4: Shape similarity differences between using 10% of the $LIC_{min}$ radius instead of 99% for the South-America dataset.

| | run #1 | run #2 | run #3 | run #4 | run #5 | run #6 | run #7 | run #8 | run #9 | run #10 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LBN | -0.2% | 0.6% | 21.4% | -0.1% | -14.2% | -1.7% | -10.3% | -4.1% | 2.6% | 1.2% | -0.5% |
| ISR | -2.5% | 0.0% | 0.0% | -1.4% | 0.0% | 0.0% | -1.3% | 0.0% | 0.0% | 0.0% | -0.5% |
| SYR | 1.3% | 4.2% | -10.5% | 6.3% | -3.3% | -1.2% | 1.3% | 2.5% | 9.1% | 4.0% | 1.4% |
| SAU | 0.0% | -2.9% | -2.0% | 1.3% | -1.5% | 0.8% | -12.1% | -0.3% | -0.2% | 3.1% | -1.4% |
| JOR | -0.5% | -41.9% | 4.2% | 1.4% | 5.7% | -4.0% | 0.4% | -6.0% | -3.6% | 11.2% | -3.3% |
| TUR | 27.2% | -3.6% | 0.1% | -8.3% | -5.7% | -5.8% | 6.9% | -1.5% | -1.9% | -0.9% | 0.7% |
| IRQ | 0.0% | 1.2% | 0.0% | 0.0% | -1.2% | -7.9% | 0.0% | 0.9% | -5.7% | -4.7% | -1.7% |
| KWT | -2.8% | 0.4% | 50.3% | 0.8% | -3.7% | 4.7% | 0.2% | 4.7% | 28.5% | 32.0% | 11.5% |
| YEM | -2.3% | -8.1% | 0.0% | 2.1% | 7.0% | 6.9% | -0.7% | 4.0% | -0.4% | 4.3% | 1.3% |
| IRN | -0.1% | 2.3% | 2.0% | -2.8% | 1.2% | -0.3% | -1.7% | -0.6% | 2.7% | -0.6% | 0.2% |
| ARE | -3.7% | 0.4% | 1.5% | 2.8% | 1.4% | -0.9% | 0.5% | -4.4% | -26.7% | -7.8% | -3.7% |
| OMN | 1.5% | -15.4% | -4.0% | -1.1% | -0.5% | 3.8% | -0.5% | 1.7% | 8.2% | -1.6% | -0.8% |
| ARM | -0.0% | -0.9% | -2.8% | 0.0% | 2.0% | 6.2% | 0.2% | 0.0% | 0.0% | 0.0% | 0.5% |
| GEO | 8.1% | -0.0% | 2.0% | 31.6% | 1.4% | -5.9% | -15.7% | -1.7% | -14.8% | 0.1% | 0.5% |
| AZE | -13.9% | -0.7% | 2.2% | 74.9% | -6.0% | -2.7% | -0.7% | -2.9% | 17.4% | -6.9% | 6.1% |
| PAK | -0.7% | -5.7% | -7.9% | -0.9% | -3.8% | 1.6% | 9.0% | 4.1% | 5.2% | -3.9% | -0.3% |
| AFG | 1.1% | -2.2% | -2.8% | 0.2% | -0.7% | -5.5% | 0.9% | -1.3% | 1.5% | -1.4% | -1.0% |
| TKM | 2.4% | 2.1% | -1.2% | 5.3% | 2.2% | -3.3% | 1.6% | 54.5% | 1.9% | 1.8% | 6.7% |
| UZB | -6.4% | -3.2% | 1.2% | -13.1% | -3.9% | -29.8% | -2.2% | -38.5% | -2.0% | -1.6% | -10.0% |
| TJK | -2.0% | -14.5% | 21.6% | 6.1% | -4.1% | -13.1% | 1.4% | 9.7% | -16.4% | -4.4% | -1.6% |
| KGZ | 0.7% | 42.5% | 1.1% | -1.9% | -15.5% | 61.2% | 2.4% | -0.8% | -8.9% | 8.1% | 8.9% |

Figure B.5: Shape similarity differences between using 10% of the $LIC_{min}$ radius instead of 99% for the Middle-East dataset.