Generalized Map Coloring for Use in Geographical Information Systems *

Robert Freimer
Caliper Corporation
1172 Beacon Street
Newton, MA 02461-1149
robert@caliper.com

ABSTRACT

We propose a new model for cartographic map coloring for use in Geographical Information Systems. Map coloring motivated the famous four-color problem in Mathematics. The published proofs of the four-color theorem yield impractical polynomial-time algorithms. Actual political maps often require generalizations to the standard four-coloring problem given the topology of some regions. We allow each region to have m disjoint pieces, which is Heawood's m-pire problem. We also count node adjacency between regions, i.e., two regions are adjacent if they share a common point. The adjacency graphs using node adjacency are known as map graphs. By combining m-pires with node and island adjacency, we formulate a new model to handle actual GIS instances. We implemented Brélaz's *Dsatur* heuristic, since no specific algorithm exists for coloring our resulting *cartographic graphs*. The choice works well in practice and we discuss the details of the implementation in TransCAD®.

Keywords

Coloring, Cartographic maps, *Dsatur* heuristic, GIS, Heawood's m-pire problem, Map Graphs

1. INTRODUCTION

In 1852, Francis Guthrie observed that four colors always sufficed for any cartographic map that he attempted to color in such a way that no two adjacent regions had the same color; he wondered whether this was always true [3, 32]. This simple conjecture remained open for many years and would become an important motivator to the field of Mathematics. It led to the concept of an abstract graph and the equivalent and more modern form of the problem: whether a planar graph can always be four-colored. Finally in 1976, the problem was solved in the affirmative using a long, but unsatisfying proof that included many pages of computer generated case analysis [2].

In the Mathematical literature, map coloring refers to assigning colors to simple connected planar regions based on edge adjacency; w.l.o.g., we can assume that the regions are polygonal [13]. This simplistic definition does not accurately reflect what cartographers face in practice because political countries may have multiple disconnected pieces; also node adjacency and islands should be considered for reasons of clarity.

In this paper, we return to Guthrie's original motivation and consider the problem of coloring cartographic maps, only in the modern setting of Geographical Information Systems. We review the current knowledge and consider two extensions which more closely reflect actual cartographic practice: regions with multiple disconnected pieces and more general definitions of adjacency. We use this more accurate model to create an actual implementation for use as part of a commercial GIS. We note that some instances require more than four colors with this model.

Surprisingly before our work, map coloring was not a feature included in any commercially available GIS software[1]. Most packages include chloropleth mapping, which colors regions using a thematic classification, but none had the capability of coloring by adjacency.

Numerous non-cartographic applications have provided the practical motivation for coloring algorithms. These include school timetables and other scheduling problems [24], various compiler optimizations [1], circuit layout and testing [34], and electronic bandwidth allocation [6].

The literature on planar and general graph theory is extensive. The book by Jensen and Toft[22] provides a very complete description of the current knowledge on graph coloring, including references to the original papers.

We begin in Section 2 with some basic definitions. We also formalize the relationship between map and graph coloring, so that results can be stated in either form as appropriate.

In Section 3, we consider the original planar maps and state the four-color theorem. Next in Sections 4 and 5, we discuss two previously known extensions: coloring maps with at most m pieces per region, better known as Heawood's m-pire problem [16], and coloring simple regions using a node adjacency rule.

By combining these two generalizations along with islands, we can formulate a sufficiently general model to han-

*An earlier version of this material appeared in Dr. Freimer's Ph.D Dissertation[12].

8th ACM Symposium on GIS 11/00 Washington, D.C., USA
© 2000 ACM ISBN 1-58113-319-7/00/0011...$5.00

[1]At least one third-party coloring extension exists for ArcInfo®. Manifold® has also released a coloring extension.

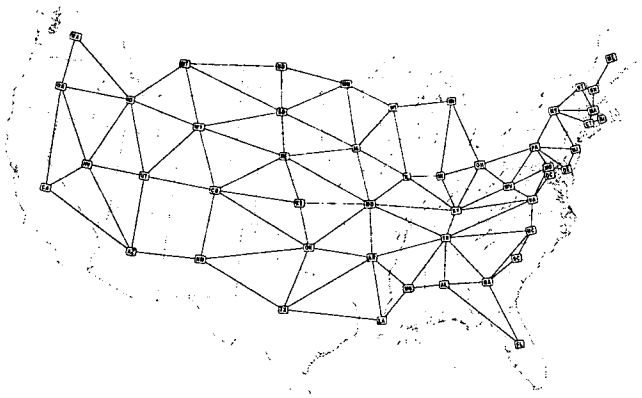

Figure 1: Correspondence between a map and its (edge) adjacency graph.

dle real GIS instances. In Section 6, we consider this important new model, whose adjacency graphs we call cartographic graphs. We discuss our implementation of Brélaz's *Dsatur* algorithm [5], which is a heuristic that works extremely well in practice on cartographic maps, producing colorings with a small number of colors.

We conclude with a couple of extensions in Section 7.

2. DEFINITIONS

We begin by formally defining map coloring. A *map* is a polygonal subdivision $\mathcal{M}$ of $\mathbb{R}^2$, with $n \geq 1$ regions (*countries*) with exactly one unbounded region. For $r \in \mathbb{N}$, a *r-coloring* of the map $\mathcal{M}$ is one that assigns a color $c \in \{1, 2, \ldots, r\}$ to each region, using at most r different colors, so that any two adjacent regions are colored using distinct colors. The minimum r for which $\mathcal{M}$ can be colored is called the *chromatic number* $\chi(\mathcal{M})$. A more through topological treatment of maps is available in [13], which shows for the context of coloring why it is sufficient to consider only polygonal subdivisions instead of a more general definition using simple closed Jordan curves.

Map coloring is closely related to the more general problem of *graph coloring*, which is concerned with coloring each vertex of a graph so that no two adjacent vertices, i.e. ones that share an edge, have the same color. More formally, let $G = (V, E)$ be a graph of $n = |V|$ vertices and $m = |E|$ edges. A coloring is a function $\rho : V \to \{1, 2, \ldots, r\}$ so that $\rho(v_1) \neq \rho(v_2)$ for every $e = (v_1, v_2) \in E$. The minimum r for which G has a coloring is denoted $\chi(G)$, the *chromatic number* of G.

The map coloring problem can easily be transformed into a graph coloring problem by building the adjacency graph G for a map $\mathcal{M}$. Let $V = \{\text{regions of } \mathcal{M}\}$ and add an edge $e = (\mathcal{R}_1, \mathcal{R}_2)$ for each pair of adjacent regions $\mathcal{R}_1$ and $\mathcal{R}_2$. Any coloring of G corresponds to a coloring of $\mathcal{M}$. Hence, map coloring is just graph coloring on a restricted subclass of graphs (see Figure 1). We will often use the equivalent graph form of the problem in this paper, following the practice of most of the coloring literature. See [13] for a much more rigorous presentation of this correspondence.

We assume in this paper that G does not contain any loops, i.e. edges of the form $e = (v, v)$. Let $\Delta(G)$ be the maximum degree of any vertex $v \in V$.

We make use of special notation for a classes of graphs named after Kuratowski. $K_{n,m}$ is the complete bipartite graph on n and m vertices.

Two regions are said to be *edge adjacent* if they share a common linear boundary (*map edge*) of $\mathcal{M}$. This is the common definition, when the map only contains simple polygons, i.e., one connected component per region. The class of such graphs defined by edge adjacency are those called *planar graphs*. If the regions are each allowed to have m pieces, then the resulting class are the *m-pire graphs*.

Two regions are said to be *node adjacent* if they share a common boundary corner point (*map node*) of $\mathcal{S}$ (e.g., Fourcorners, AZ, CO, NM and UT in Figure 1). If at most k regions meet at any node then the adjacency graph is called a *k-map graph* using the terminology of Chen et al.[7]. The 3-map graphs are the ordinary planar graphs. Collectively, the k-map graphs are called *map graphs*.

To match cartography, we sometimes may want to allow an "empty" region, which includes the unbounded portion of the plane. This uncolored region would correspond to the ocean and inland water bodies. Using this model, two regions are said to be *island adjacent* if they are visible across the water within some threshold distance d. I.e., some point p_1 of region $\mathcal{R}_1$ can "see" across the water along a straight line to some point p_2 of region $\mathcal{R}_2$ and the distance $|p_1p_2|$ is at most d. This is a superset of the node adjacency version, since node adjacent regions can vacuously see each other at distance 0. A similar model that required the water to be connected (no lakes) was examined by Jackson and Ringel[20]. They found slightly tighter bounds on $\chi(\mathcal{M})$ for these *island maps*, which are m-pires with the requirement that each region has some ocean shoreline.

In this paper we will consider the coloring problems defined by using edge, node and island adjacency. Adjacency graphs which result from the combination of island adjacency with m-pires will be called *cartographic graphs*.

3. PLANAR GRAPHS

In this section, we consider the classical problem of coloring maps using edge adjacency where each region must be connected. Planar graphs are the equivalent graph version of this problem, which have been extensively studied [28].

After more than a century, planar graphs were finally shown to be four-colorable in 1976 by Appel et al.[2], using a computer generated case analysis to reduce a graph to one of 1900+ unavoidable configurations. This has been greatly simplified by a new proof by Robertson et al.[31], which still has 633 cases. Summarizing their result:

THEOREM 1. **([2])** *Given a map $\mathcal{S}$, where each region in the subdivision is connected, then the map using edge adjacency can be four-colored.*

We now discuss algorithms to color planar graphs using between two and five colors. Two is possible only when the graph is bipartite, which can be easily checked in linear time. No polynomial-time algorithm is likely to exist for three-coloring, since Stockmeyer showed that deciding whether a planar graph can be three-colored is *NP*-Complete, even when $\Delta(G) = 4$ [15].

Each of the two proofs of Theorem 1 can be transformed to provide a theoretical algorithm for four-coloring planar graphs. Unfortunately, neither seems to be practical with regards to implementation. The algorithm derived from Appel

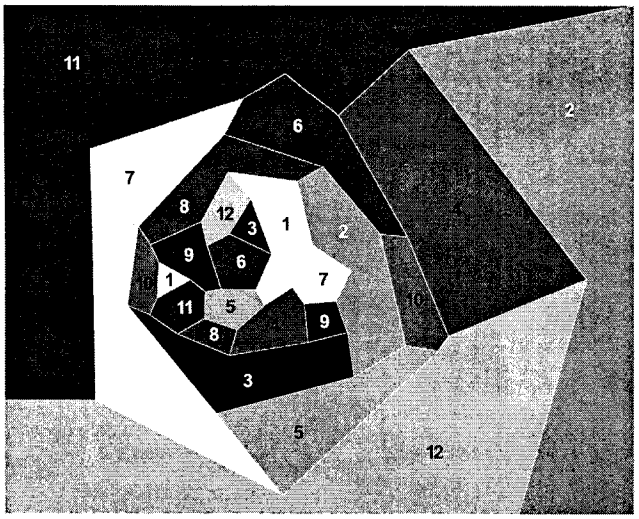

Figure 2: Heawood's 2-pire which requires 12 colors.

et al.'s proof has $O(n^4)$ time complexity with astronomical coefficients [30]. The somewhat simpler coloring algorithm from Robertson et al.[30] still has $O(n^2)$ time complexity and as far as we are aware has not been implemented.

Efficient and practical algorithms for five-coloring have been known for many years [28]. The most recent is due to Thomassen[35], whose proof that every planar graph is five-choosable, yields an extremely simple linear-time algorithm. Our implementation of this five-coloring algorithm works well in practice [12]; we have successfully colored all 1,081, 257 atomic polygons of Texas from 1998 TIGER/Line®.

Unfortunately, many geographic layers encountered are not planar and hence cannot be colored by Thomassen's algorithm. E.g., the edge adjacency graph of world countries has been non-planar since the demise of the Soviet Union; Azerbaijan is a 2-pire, which along with Georgia, Iran, Armenia, Russia and Turkey define a $K_{3,3}$-minor. However, it is still possible to four-color this graph.

A variety of other implementations and heuristics exist for coloring planar graphs [19, 27]. The LEDA[25] library also includes a FIVE_COLOR function for planar graphs. The only implementation by a cartographer that we found is an impractical exponential-time map coloring heuristic [11].

4. M-PIRES

In practice, cartographic regions may have multiple disconnected pieces. Coloring such regions is known as Heawood's Empire problem after the English mathematician, since in the late 1800's countries often had many colonies (e.g., the British Empire) and because of the convenient pun. Heawood[16] proved the upper bound on the number of colors required for an m-pire and gave a lower bound example for $m = 2$ (see Figure 2). The case $m = 1$ is Theorem 1. The lower bound examples for $m = 3, 4$ were discovered by Taylor[14]. Almost 100 years after the problem was first posed, the complete lower bound was proved by Jackson and Ringel[21] and later shortened by Wessel[38]. The tight bound is stated as follows:

THEOREM 2. **([16, 21])** *Given a map S, where each region in the subdivision has at most $m \geq 1$ disjoint connected components, then the map using edge adjacency can be colored with $\left\lfloor \frac{1}{2}(6m + 1 + \sqrt{(6m+1)^2 - 48}) \right\rfloor$ colors and for each m this bound is tight for some maps.*

From Heawood's proof, we have derived and implemented a linear-time algorithm that colors an m-pire with $6m$ or fewer colors using edge adjacency [12]. The algorithm turns out not to be useful for real maps, since $6m$ is almost always much larger than $\chi(\mathcal{M})$. E.g., the West Aleutians are comprised of 28 islands, so the algorithm uses 168 colors for the U.S. counties, which is absurd since five is sufficient.

When the number of pieces per region is unbounded, n colors may be required for an m-pire, since it is possible to simulate a general graph by using two pieces for each edge, one for each endpoint. Hence, finding the minimum coloring is *NP*-Complete by a reduction from general graph coloring.

5. MAP GRAPHS

Cartographers often need to consider node adjacency for reasons of clarity given the complexity of many political regions. In this section, we will again consider maps with a single polygon per region, but this time use the node adjacency definition for coloring. The adjacency graphs arising from such maps (map graphs) can be divided into classes based upon k, the maximum number of regions which meet at a single node. Obviously, k-map graphs always require at least k colors. We will demonstrate in this section that some k-map graphs can require $\lfloor \frac{3}{2}k \rfloor$ colors and conjecture that this is also sufficient.

The node adjacency generalization was first considered by Ore and Plummer[22]. Restating their result:

THEOREM 3. *For $k \geq 3$, given a k-map graph G such that no region is completely surrounded by another, $\chi(G) \leq 2k$.*

This was strengthened slightly to $\chi(G) \leq 2k - 3$ for $k \geq 8$ by Borodin[22]. He also conjectured that:

CONJECTURE 1. *For $k \geq 3$, any k-map graph can be colored using $\lfloor \frac{3}{2}k \rfloor$ colors.*

We independently developed this conjecture while investigating this problem, as did [7]. Their motivation to extend the definition of planarity was a restricted version of the topological inference problem. They have an extremely complicated method for deciding in polynomial time whether a graph is 4-map and conjectured that this was also possible for any $k \geq 4$. The affirmative answer was provided by Thorup[36]. Collectively, they gave a useful description of the region configurations from which the maximum cliques arise, including *flowers* and *hamantaschen* using Thorup's terminology (see Figure 3). It is easy to see for the hamantasch that $\lfloor \frac{3}{2}k \rfloor$ colors may be required for a k-map graph, matching the conjectured upper bound.

For $k = 3$, Conjecture 1 is the Theorem 1. For $k = 4$, every 4-map graph can be shown to be *1-planar*, i.e., they can be embedded in the plane so that each edge is crossed by at most one other edge, which can be six-colored [36]. The conjecture remains open for $k \geq 5$.

Finding a minimum coloring for map graphs remains *NP*-Complete, since they are a generalization of planar graphs.

An interesting, but very restricted version of node adjacency coloring arises when coloring quadtrees using corner

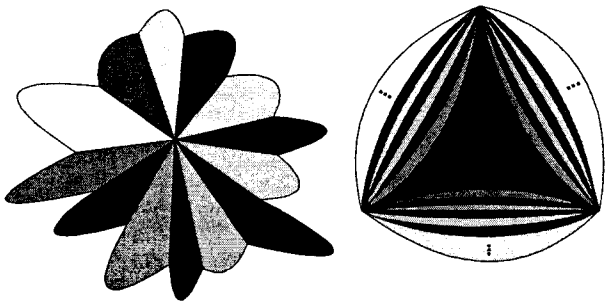

Figure 3: Two types of maximum cliques: (a) a flower and (b) a hamantasch.

adjacency. A *quadtree* is defined by starting with an initial square and recursively splitting some of the squares into four smaller squares. Clearly quadtrees are a special case of planar maps. A quadtree is *balanced*, if for any inside edge, the ratio of the lengths of the squares adjacent to the edge is within a factor of two. Bern et al.[4] showed for edge adjacency that $\chi(G) = 3$ for balanced quadtrees and $\chi(G) = 4$ otherwise. For node adjacency, they showed that $5 \leq \chi(G) \leq 6$ and gave a linear-time algorithm for six-coloring quadtrees.

6. CARTOGRAPHIC MAPS

In the preceding sections, we examined various generalizations to planar graphs. We will now consider the combination of island adjacency with m-pires. The resulting cartographic graphs are an attempt to model how actual maps are colored. Frequently, countries have multiple components and node adjacency needs to be considered for clarity. Nearby islands also need to be colored differently, so that different nations can be distinguished as in the Caribbean.

This new coloring problem remains *NP*-Complete, since cartographic graphs are a generalization of planar graphs.

An upper bound on $\chi(G)$ for cartographic graphs is $\Delta(G)$ +1, using Brooks Theorem for general graphs [22].

In this section, we review some of the methods from the literature for coloring general graph and explain our choice of Brélaz's *Dsatur* algorithm [5]. We provide the details of our implementation and discuss how it works well in practice with the class of adjacency graphs arising from actual GIS datasets.

6.1 Coloring Algorithms

Exact algorithms for coloring general graphs are impracticable given the *NP*-Completeness results. This has not prevented some attempts using various methods to speed them up [27], but they are usually prohibitively slow except on small examples (say a few hundred vertices) due to the exponential-time complexity. Approximation algorithms are primarily of theoretical interest, since the bounds that they guarantee are far from optimal. Even finding an approximation within a factor of 2 is also *NP*-Complete [15]. In fact, graph coloring serves as one of the canonical problems in Class IV of inapproximability problems [18].

Heuristics have provided the most frequent solution for practical graph coloring problems, since they typically find small colorings in a reasonable amount of time (polynomial). However, in order to achieve this performance, they need to be tuned for the particular class of input graphs. No heuristic is known that works well across the complete spectrum of general graphs.

Heuristics can be broken down into two general classes: sequential (successive augmentation) and iterative improvement. See [27] for a complete discussion.

Sequential heuristics color the vertices of G, one at a time, using some order σ. Color choices, once made, are usually considered to be unchangeable. If only information available before any vertices have been colored is used to determine σ, then the method is called *static*. Commonly used static orderings include random, largest-first (degree) and smallest-last. A *dynamic* method makes use of the previously assigned colors in determining σ. A good comparison of sequential algorithms is provided by [29].

The most frequently used dynamic sequential heuristic is Brélaz's *Dsatur* algorithm [5] (a.k.a., the saturation algorithm): select the vertex with the greatest saturation (number of colors already used by neighbors), breaking ties using the largest degree vertex. This method typically works well in practice for sparse graphs. Turner[37] further supports this claim with a probabilistic analysis showing *Dsatur*'s average performance ratio is good for almost all sparse graphs. However, the worst-case behavior of *Dsatur* is poor: $O(n)$ [33]; they give three-colorable examples with $O(n)$ vertices, for which *Dsatur* uses n colors. *Dsatur* is often considered to be the best sequential method.

Various strategies have been employed to improve upon static colorings. *Iterated improvement* heuristics use a sequence of iterations that start with an initial coloring and search for better solutions. A variety of general optimization strategies have been used, including hybrid genetic algorithms [10], simulated annealing [23], and Tabu search [17]. While success has been reported with the general methods, specific coloring strategies such as iterated greedy [9] and distributed coloration neighborhood search [26] usually will do even better. Unfortunately, in either case the resulting heuristics usually perform well on only a small class of problems. We omit any further discussion of iterated heuristics, since they are complex and we do not feel that they are appropriate for our GIS problem.

Very little code for heuristics for general graph coloring is publicly available. Culberson[8] has provided a useful suite of programs for exploring different methods for graph coloring. Thanks to these, we decided to implement the *Dsatur* algorithm. An implementation of *Dsatur* is also provided with Mathematica® for graph coloring.

6.2 Implementation

We implemented the *Dsatur* heuristic in *C* to add cartographic coloring functionality to TransCAD®, which is a high-end GIS with many extensions for transportation, planning and logistics. A large amount of geographic data is included with the software, which provides many huge real examples for our experiments.

As is frequently the case for geometric algorithms, a fair portion of the implementation effort is spent on support routines and data structures, which are omitted from most theoretical descriptions. For our coloring implementations, this involved approximately half the time. For instance, we had to process the native geographic area layers to efficiently compute the different types of adjacency. TransCAD®'s

geographic database structures are optimized for drawing, storage and quick spatial queries. They are not designed to provide cheap and simple navigation around the plane graph bounding the faces of an area layer. So we developed functions to efficiently process an area layer and create a sparse memory-based structure to contain the adjacency graph. The structure, which contains the upper triangular portion of the symmetric adjacency matrix, is stored in an array. The first n cells contain records for the n regions. Each record includes the index to the first cell in its adjacency list. The lower half of the adjacency matrix is populated before one of our coloring algorithms is used.

Edge adjacency is computed by simply looping over all the map edges once, adding an entry to the adjacency matrix for the pair of regions which border an edge if it does not already exist.

For node adjacency, a temporary point database is created to efficiently locate map nodes. The edge adjacency processing step within the loop is expanded to process each endpoint of the edge. The presence of each node is tested against the point database and is added if missing. An in-memory list for each node is maintained to keep track of its incident regions. At the conclusion of the edge loop, the list of nodes is processed to update the adjacency matrix for every pair of regions incident to the node.

Island adjacency is computed by augmenting the point database used for node adjacency with extra shape points from edges which border water, region 0. Every 25^{th} internal shape point of each water edge is added; this sampling rate is employed to simplify the final processing step which occurs after the node adjacency has been computed. Each water node is processed in turn, looking at all the other water nodes within distance d. For any which are incident to different regions, the adjacency matrix is updated. Notice that the cartographic detail, i.e. number of shape points, of the layer has an important impact on the running time for computing island adjacency.

Once the desired adjacency has been computed, **DSATUR** is called. This assigns a region a color by updating the value of a specified database field. The area layer is then redrawn using a manual, list of values, chloropleth map on the color field.

We now formally state Brélaz's *Dsatur* algorithm [5]:

DSATUR(G):
1. Choose an uncolored vertex v with maximum saturation degree (the number of distinct colors already used by adjacent vertices), breaking ties using the largest vertex degree. For the initial iteration, this selects the vertex with the largest degree.
2. Color v with the lowest available color c.
3. Increment the saturation degree for each neighbor w of v for which this is the first use of c in w's neighborhood.
4. Stop if all vertices are colored, otherwise repeat Step 1.

A priority queue based on an AVL tree is used to keep track of the saturation and vertex degrees of the uncolored vertices. A bit-vector of size $degree(v) + 1$ is maintained for each vertex v to keep track of the colors already used. This leads to an $O(m \log n)$ implementation, where $m = |E|$, since each saturation update for neighbors is handled by a deletion from and an addition to the priority queue and there can be $O(m)$ updates.

The strategy behind the *Dsatur* algorithm is that the most constrained vertex at each stage should be colored next to prevent running out of colors later. Ties are broken using the largest-first rule, since vertices with larger degree are often more difficult to color. This combination works well in practice on sparse graphs as can be seen from the experimental results presented in the next subsection.

Brélaz's original algorithm had $O(n^2)$ time complexity due to the initialization of bit-vectors. We employed Morgenstern's observation [27] that each bit-vector need only be of length $degree(v) + 1$ to eliminate that issue. Further information can be found in Morgenstern's dissertation, including a $\Theta(m)$ time PASCAL implementation based on lazy arrays, which didn't seem practical for our GIS application. Morgenstern did not provide any experimental results for his *Dsatur* implementation.

6.3 Performance

We ran our algorithm on a wide spectrum of instances, ranging from nationwide U.S. Census Block Groups to World Countries (see Figure 4 for U.S. Counties). The U.S. geography was derived from 1998 TIGER/Line®. The World Countries were derived from the Digital Chart of the World.

For each layer, we report in Table 1 the following statistics to better understand the complexity of the instance: the number of colors used for edge, node and island adjacency, the number of geographic regions in the layer, the number of geographic polygons in the layer, the maximum number of polygons comprising any single geographic region, the number of geographic boundary edges in the layer, the number of shape points in the layer, and the maximum and average node degrees.

We also report in Table 1 for each layer and type of adjacency, the running times in seconds for computing the adjacency graph and for coloring the adjacency graph. We ran our trials under Microsoft Windows® NT 4.0 with a 450MHz Pentium® II processor. Only the edge adjacency rule was used for the larger instances due to the long running times. Notice that the adjacency computations completely dominates the time required for determining the coloring, which is roughly linear to the number of adjacency graph edges.

For smaller instances, say less than 2,000 regions, the speed of the implementation is sufficiently good to be used in real-time while a user interacts with the software. For larger instances, it really should be run offline, saving the result in a field for later use. This works well in practice, since colorings do not need to be recomputed unless an area layer is changed.

Observe that the colorings produced (see Figure 4) are not close to being evenly balanced; the smaller colors appear with much greater frequency due Step 2 of **DSATUR**. Unfortunately changing this rule to randomly select a color often causes more colors to be used.

7. EXTENSIONS

We conclude this paper with a couple of extensions to cartographic map coloring.

Cartographers like to use colors evenly throughout a map for esthetic reasons. A couple of metrics that can be used for balancing the colors are the number of countries with each color [11] and the total area colored by each color. None of the algorithms discussed in this paper attempt to satisfy this requirement. A theorem of Hajnal and Szemerédi[22] shows for general graphs that balancing is possible when using the

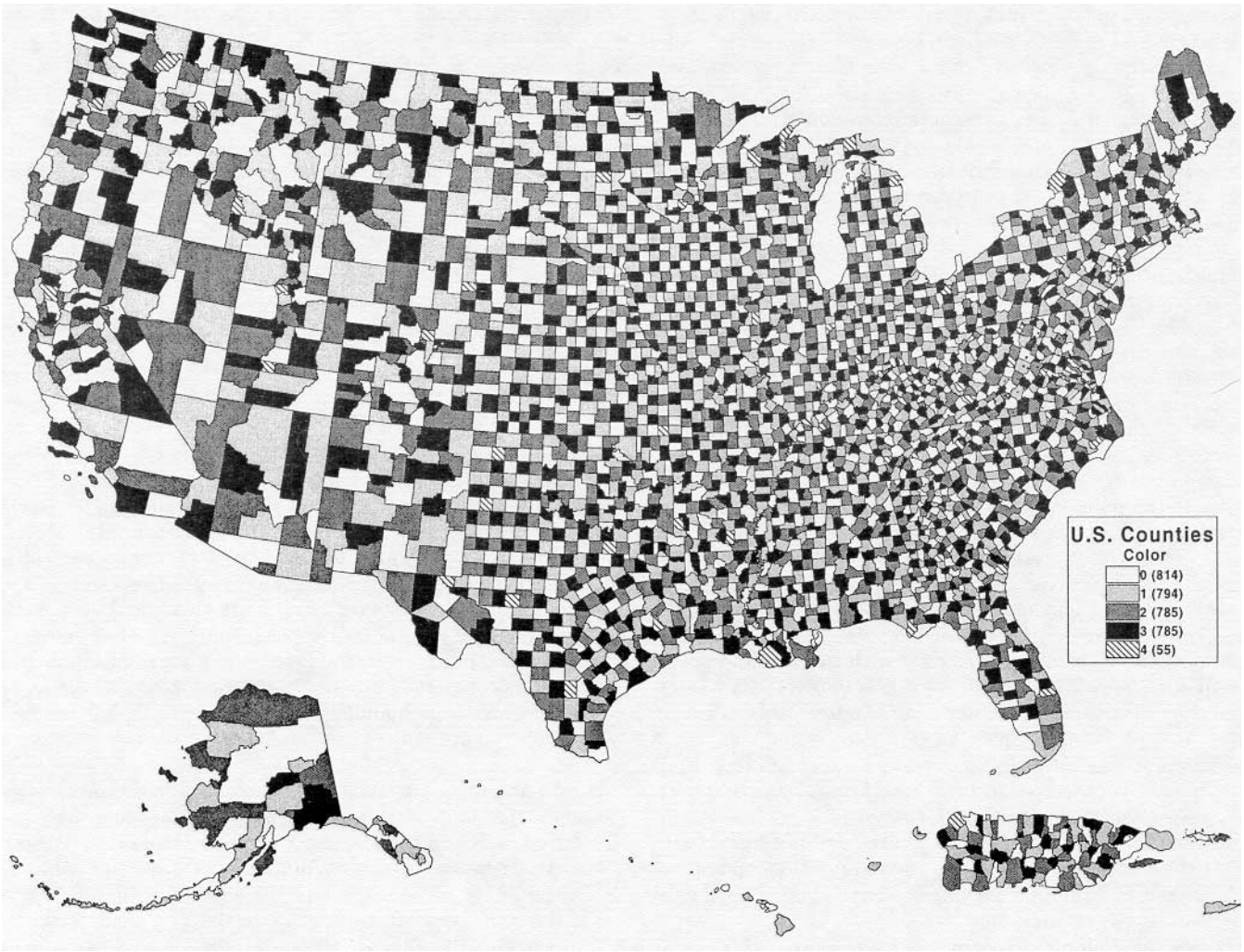

Figure 4: A *Dsatur* five-coloring of U.S. counties using the island adjacency rule.

Table 1: Combinatorial statistics and *Dsatur* running times (seconds) for various summary levels of geography.

Area Layer	Adj. Rule	*Dsatur* Colors	# Regions	# Polygons	Max. Polygons	# Bndry. Edges	# Shape Points	Max. Degree	Avg. Degree	Adj. Time	*Dsatur* Time
World Countries	Edge	4	260	27,007	5,225	27,646	1,508,574	14	2.40	36.24	0.12
	Node	4						14	2.40	63.32	0.06
	Island	5						16	2.90	1,035.96	0.05
U.S. States	Edge	4	57	314	49	435	302,621	8	3.79	7.54	0.03
	Node	4						8	3.86	8.43	0.03
	Island	4						8	4.25	52.86	0.03
U.S. Counties	Edge	5	3,233	3,575	28	10,613	1,779,943	13	5.60	51.32	0.58
	Node	5						14	5.83	60.56	0.59
	Island	5						14	5.95	125.82	0.59
U.S. County Subdivisions	Edge	5	36,244	43,794	89	110,651	5,829,124	35	5.19	191.77	6.32
	Node	6						35	6.19	290.37	7.19
	Island	6						35	6.24	592.76	7.17
U.S. Census Tracts	Edge	5	62,829	64,390	18	2,580,416	9,505,979	108	5.48	550.74	22.27
	Node	6						108	6.20	3,450.94	12.62
U.S. Census Block Groups	Edge	5	230,446	232,771	19	5,320,913	18,603,919	89	5.49	1,198.42	43.61

worst-case number of colors. Does an algorithm for cartographic graphs exist which guarantees a balanced coloring with a small number of colors for cartographic graphs?

Are there better definitions of island adjacency? For instance, consider the Voronoi diagram of the islands. Two regions might be adjacent if they share a Voronoi edge.

8. REFERENCES

[1] A. Aho, R. Sethi, and J. D. Ullman. *Compilers, principles, techniques and tools.* Addison-Wesley, Reading, MA, 1986.

[2] K. Appel and W. Haken. *Every Planar Map is Four Colorable.* Amer. Math. Soc., Providence, RI, 1989.

[3] D. Barnette. *Map Coloring, Polyhedra, and the Four-Color Problem.* Math. Assoc. Amer., Washington, DC, 1983.

[4] M. W. Bern, D. Eppstein, and B. Hutchings. Algorithms for coloring quadtrees. To appear in *Algorithmica*, 1999.

[5] D. Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, Apr. 1979.

[6] B. Chamaret, S. Ubeda, and J. Žerovnik. A randomized algorithm for graph coloring applied to channel allocation in mobile telephone networks. In T. Hunjak, L. J. Martić, and L. Neralić, editors, *Proc. 6th Intl. Conf. Oper. Res.*, pages 25–30, 1996.

[7] Z.-Z. Chen, M. Grigni, and C. H. Papadimitriou. Map graphs. Draft, Dept. Math. Sci., Tokyo Denki Univ., Hatoyama, Saitama, Japan, 1999.

[8] J. Culberson. *Graph Coloring Programs Manual.* Dept. Comput. Sci., Univ. Alberta, Edmonton, AB, 1997.

[9] J. C. Culberson and F. Luo. Exploring the k–colorable landscape with iterated greedy. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*, pages 245–284, Providence, RI, 1996. Amer. Math. Soc.

[10] C. Fleurent and J. A. Ferland. Genetic and hybrid algorithms for graph coloring. In G. Laporte and I. Osman, editors, *Metaheuristics in combinatorial optimization*, pages 437–461. J. C. Baltzer, 1996.

[11] D. Forrest. Colouring the political map. *Cartographic J.*, 33(2):141–147, Dec. 1996.

[12] R. Freimer. Shattering geometric objects and coloring geographical maps. Ph.D. Thesis, Cornell Univ., Ithaca, NY, 2000. `http://www.freimer.com/thesis.pdf`.

[13] R. Fritsch and G. Fritsch. *The Four-Color Theorem.* Springer-Verlag, New York, NY, 1998.

[14] M. Gardner. *The last recreations: hydras, eggs, and other mathematical mystifications.* Copernicus, New York, NY, 1997.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, NY, 1979.

[16] P. J. Heawood. Map colour theorem. *Quart. J. Pure Appl. Math.*, 24:332–338, 1890.

[17] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.

[18] D. Hochbaum, editor. *Approximation Problems for NP-Complete Problems.* PWS Publishing Company, Boston, MA, 1997.

[19] J. P. Hutchinson and S. Wagon. Programming tips: Four-coloring planar maps. *Mathematica Educ. Res.*, 6(1):42–51, 1997.

[20] B. Jackson and G. Ringel. Coloring island maps. *Bull. Austral. Math. Soc.*, 29(2):151–165, 1984.

[21] B. Jackson and G. Ringel. Solution of Heawood's empire problem in the plane. *J. Reine Angew. Math.*, 347:146–153, 1984.

[22] T. R. Jensen and B. Toft. *Graph Coloring Problems.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, New York, NY, 1995.

[23] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. *Oper. Res.*, 39(3):378–406, May-June 1991.

[24] F. T. Leighton. A graph colouring algorithm for large scheduling problems. *J. Res. National Bureau of Standards*, 84(6):489–503, 1979.

[25] K. Mehlhorn, S. Näher, M. Seel, and C. Uhrig. *The LEDA User Manual, Version 3.8*, 1999.

[26] C. Morgenstern. Distributed coloration neighborhood search. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*, pages 335–357, Providence, RI, 1996. Amer. Math. Soc.

[27] C. A. Morgenstern. *Algorithms for General Graph Coloring.* PhD thesis, Dept. Comput. Sci., Univ. NM, Albuquerque, NM, 1989.

[28] T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms.* North-Holland, Amsterdam, Netherlands, 1988.

[29] J. Peemöller. Numerical experiences with graph coloring algorithms. *European J. Oper. Res.*, 24(1):146–151, Jan. 1986.

[30] N. Robertson, D. P. Sanders, P. Seymour, and R. Thomas. Efficiently four-coloring planar graphs. In *Proc. 28th Annu. ACM Sympos. Theory Comput.*, pages 571–575, 1996.

[31] N. Robertson, D. P. Sanders, P. D. Seymour, and R. Thomas. The four colour theorem. *J. Combin. Theory Ser. B*, 70:2–44, 1997.

[32] T. L. Saaty and P. C. Kainen. *The Four-Color Problem.* Dover Publications, New York, NY, 1977.

[33] J. P. Spinrad and G. Vijayan. Worst case analysis of a graph coloring algorithm. *Discrete Appl. Math.*, 12(1):89–92, 1985.

[34] I. Stewart. Math recreations: Empires and electronics. *Scientific Amer.*, 277(3):92–94, Sept. 1997.

[35] C. Thomassen. Every planar graph is 5-choosable. *J. Combin. Theory Ser. B*, 62:180–181, 1994.

[36] M. Thorup. Map graphs in polynomial time. Unpublished manuscript, Dept. Comput. Sci., Univ. Copenhagen, Denmark, 1998.

[37] J. S. Turner. Almost all k-colorable graphs are easy to color. *J. Algorithms*, 9:63–82, 1988.

[38] W. Wessel. A short solution of Heawood's empire problem in the plane. *Discrete Math.*, 191:241–245, 1998.